On reconstruction with Graph Neural Networks (GNN) in the IceCube experiment

Troels C. Petersen

Niels Bohr Institute / University of Copenhagen

GraphNeT
Graph Neural Networks for Neutrino Telescope Event Reconstruction

Oxford University, 23rd of September 2022

Outline & Aim

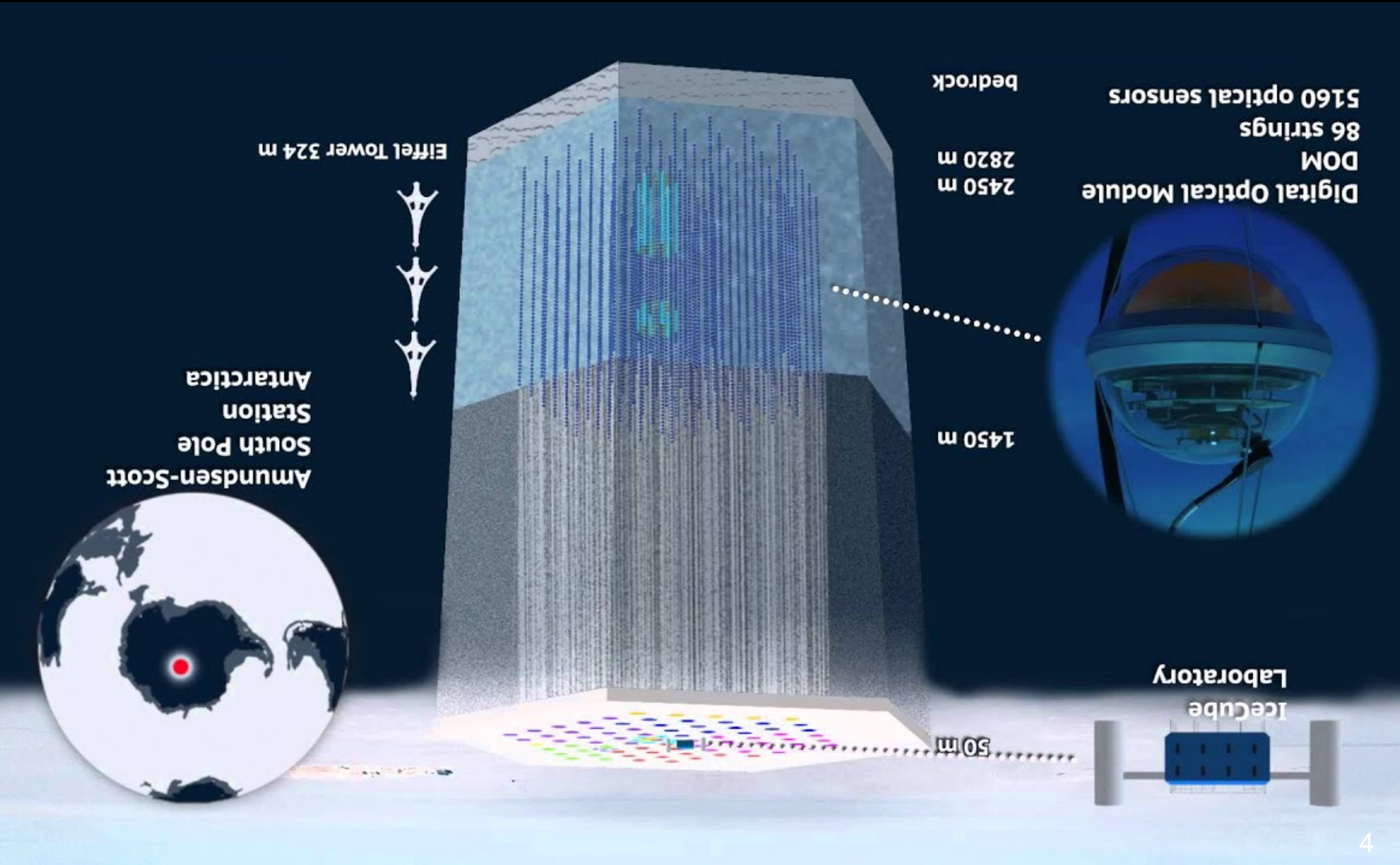
In the following, I will talk about:

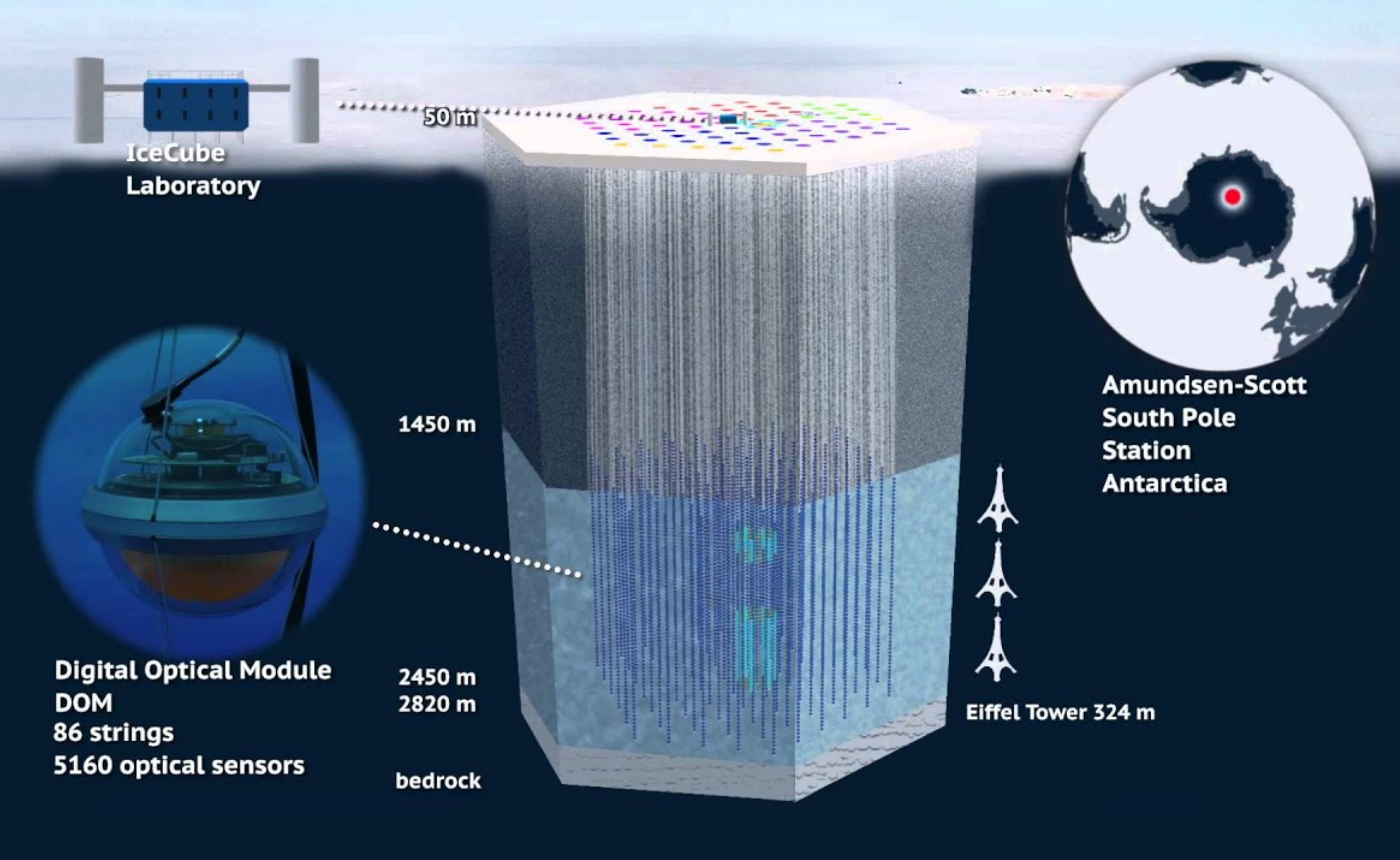
- The IceCube detector and physics program.
- The IceCube neutrino signals.
- Neutrino reconstruction algorithms.
- The workings of Graph Neural Networks (GNN).
- How GNNs might be used to expand neutrino astronomy.

In the following, I hope to convey that:

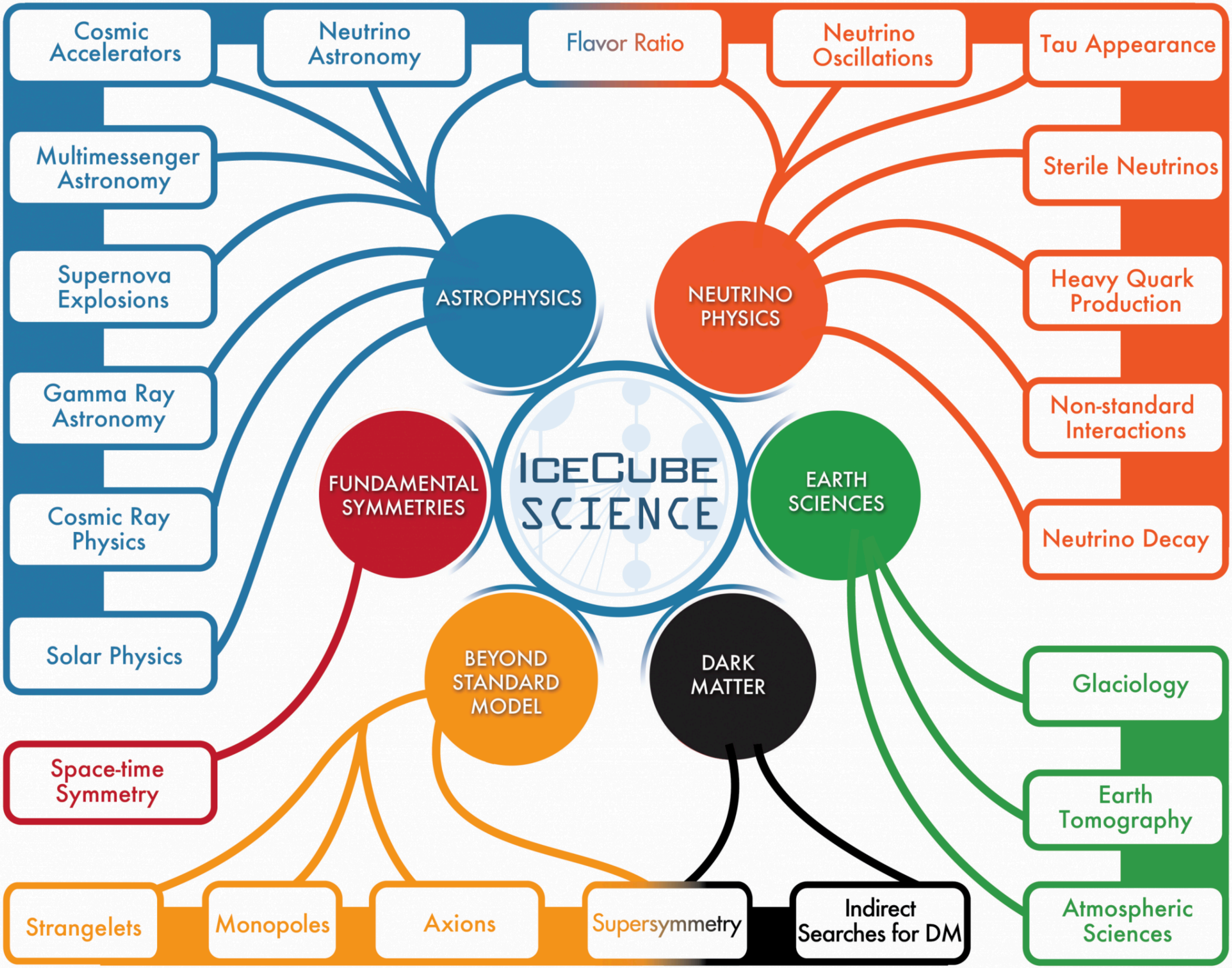
IceCube Detector & Physics

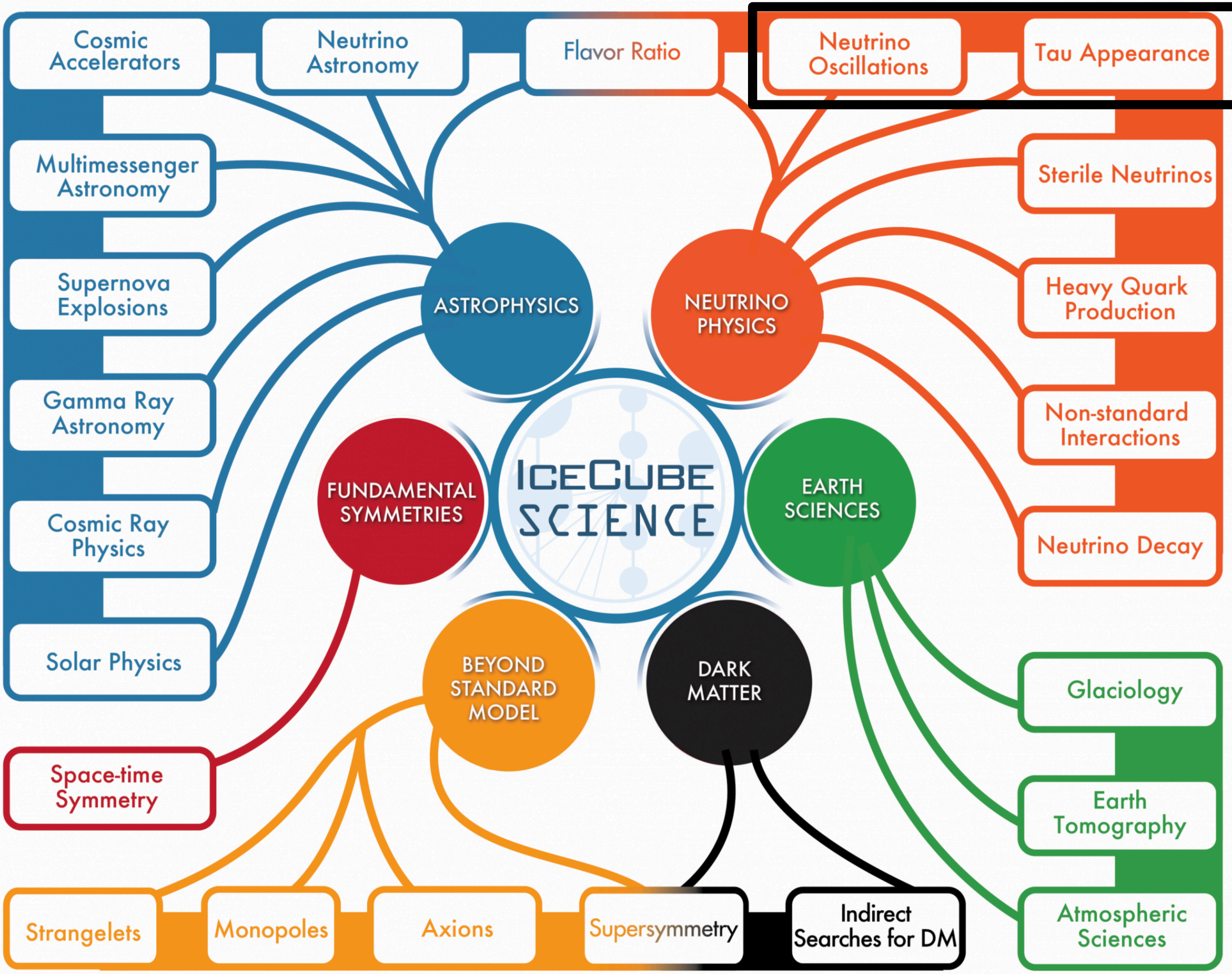
IceCube experiment (South Pole)



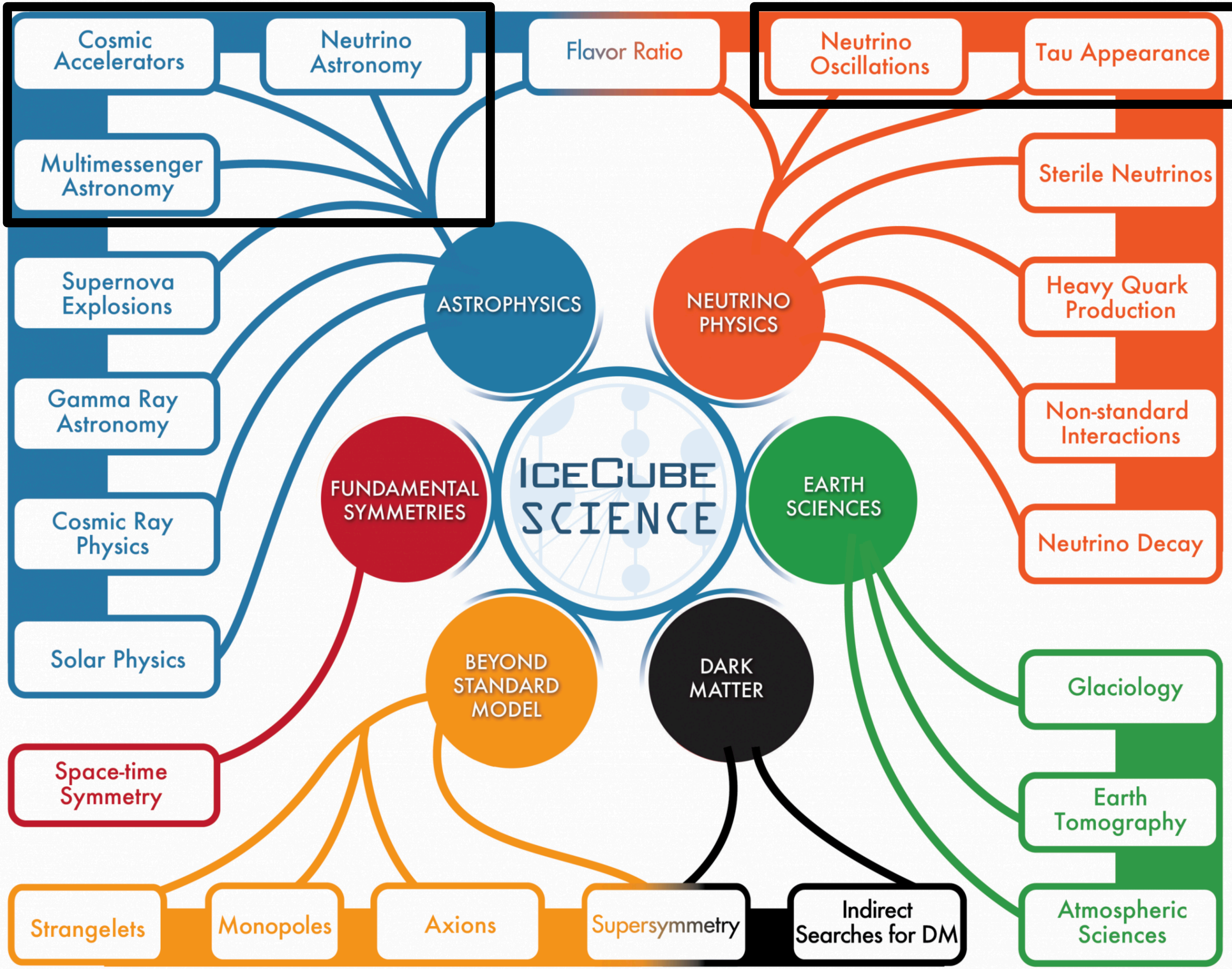


IceCube experiment (South Pole)


ICECUBE SCIENCE
ASTROPHYSICS
NEUTRINO PHYSICS
FUNDAMENTAL SYMMETRIES
EARTH SCIENCES
BEYOND STANDARD MODEL
DARK MATTER
Cosmic Accelerators
Neutrino Astronomy
Flavor Ratio
Neutrino Oscillations
Tau Appearance
Multimessenger Astronomy
Sterile Neutrinos
Supernova Explosions
Heavy Quark Production
Gamma Ray Astronomy
Non-standard Interactions
Cosmic Ray Physics
Neutrino Decay
Solar Physics
Glaciology
Space-time Symmetry
Earth Tomography
Strangelets
Monopoles
Axions
Supersymmetry
Indirect Searches for DM
Atmospheric Sciences



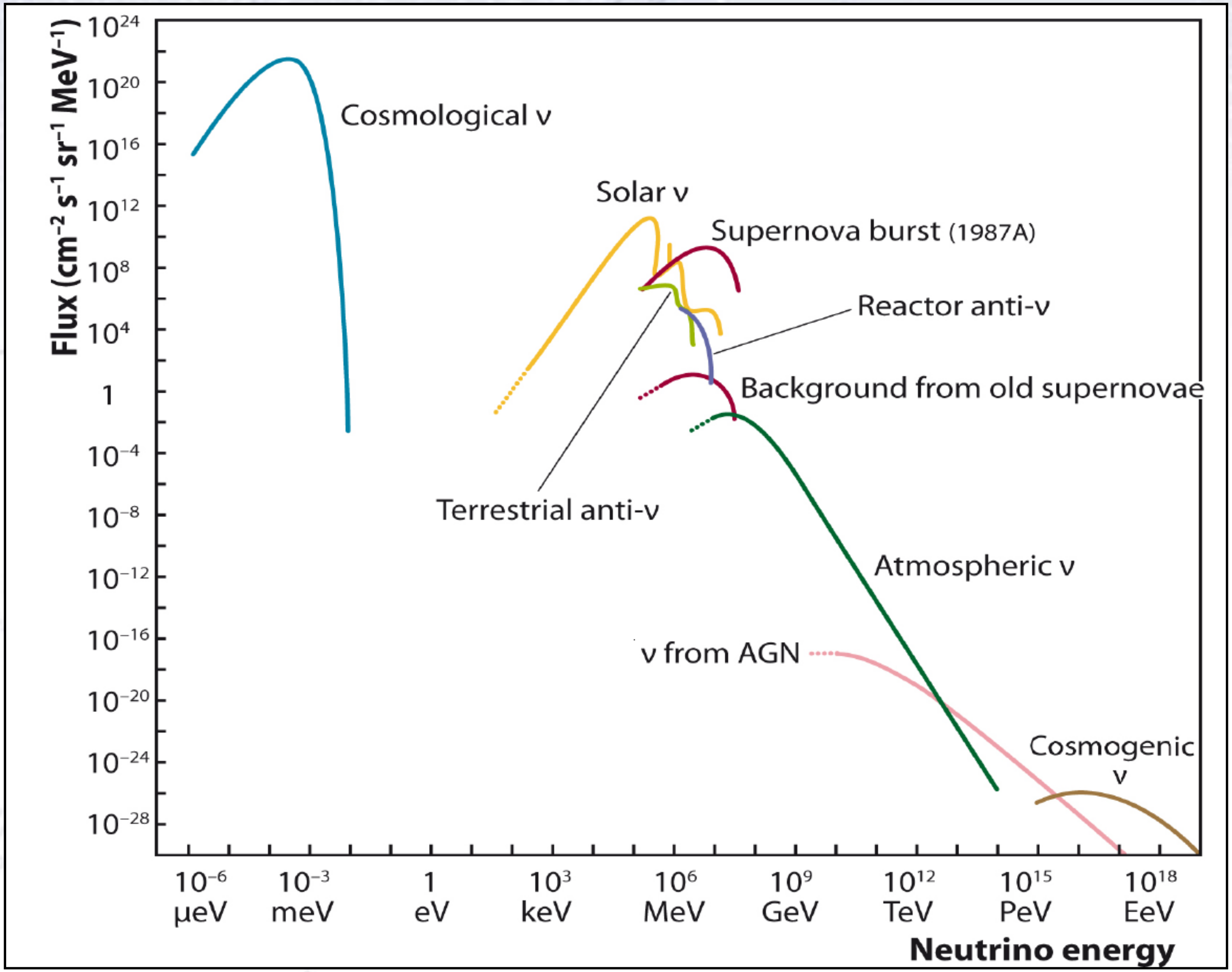
Cosmic Accelerators
Neutrino Astronomy
Flavor Ratio
Neutrino Oscillations
Tau Appearance
Multimessenger Astronomy
Sterile Neutrinos
Supernova Explosions
ASTROPHYSICS
NEUTRINO PHYSICS
Heavy Quark Production
Gamma Ray Astronomy
Non-standard Interactions
FUNDAMENTAL SYMMETRIES
ICECUBE SCIENCE
EARTH SCIENCES
Cosmic Ray Physics
Neutrino Decay
Solar Physics
BEYOND STANDARD MODEL
DARK MATTER
Glaciology
Space-time Symmetry
Earth Tomography
Strangelets
Monopoles
Axions
Supersymmetry
Indirect Searches for DM
Atmospheric Sciences


Cosmic Accelerators
Neutrino Astronomy
Flavor Ratio
Neutrino Oscillations
Tau Appearance
Multimessenger Astronomy
Sterile Neutrinos
Supernova Explosions
ASTROPHYSICS
NEUTRINO PHYSICS
Heavy Quark Production
Gamma Ray Astronomy
Non-standard Interactions
ICECUBE SCIENCE
FUNDAMENTAL SYMMETRIES
EARTH SCIENCES
Cosmic Ray Physics
Neutrino Decay
Solar Physics
BEYOND STANDARD MODEL
DARK MATTER
Glaciology
Space-time Symmetry
Earth Tomography
Strangelets
Monopoles
Axions
Supersymmetry
Indirect Searches for DM
Atmospheric Sciences

IceCube neutrino signals

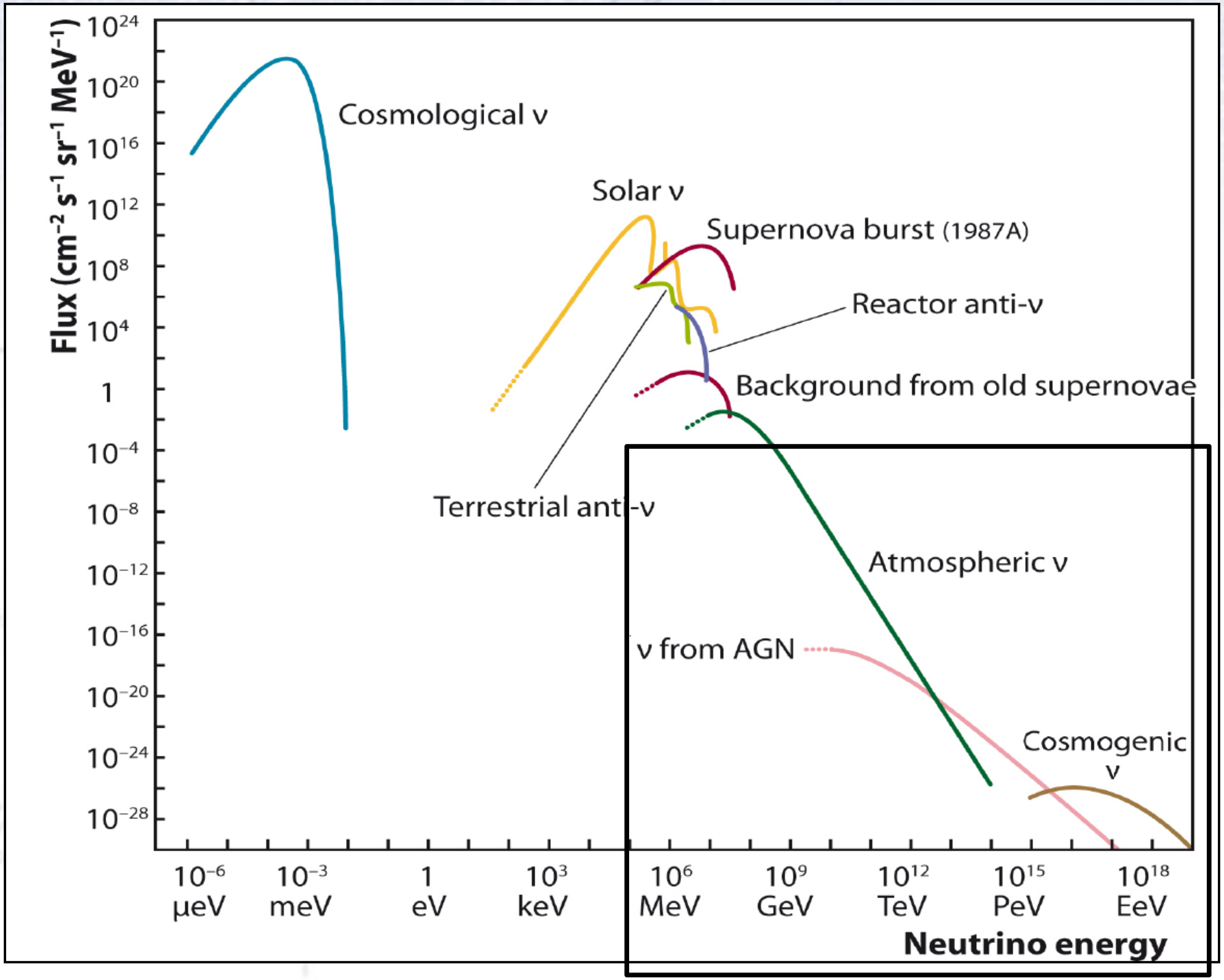
Signals in IceCube

The neutrino spectrum is largely unknown, but expected to be as follows:



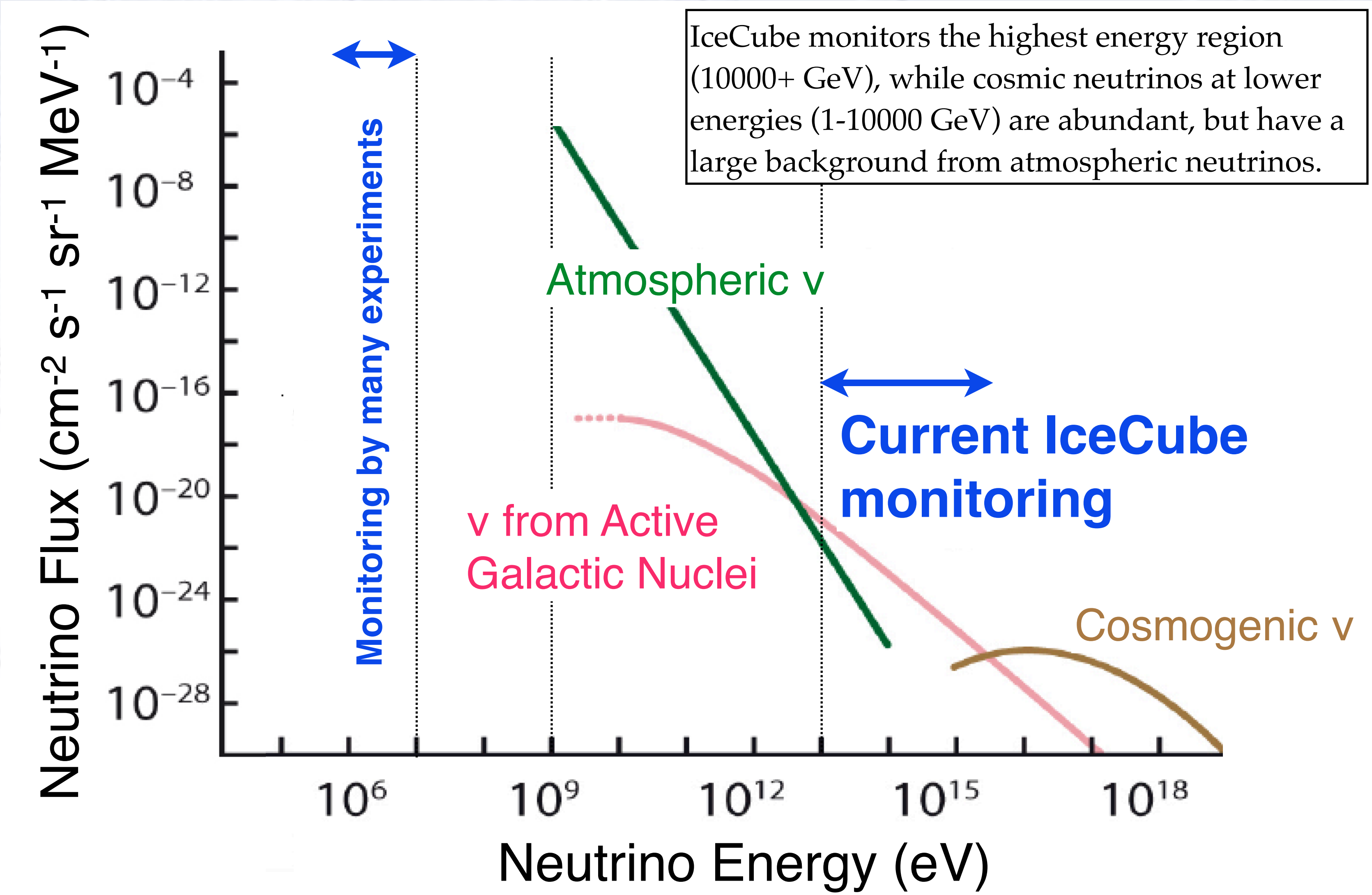
Signals in IceCube

The neutrino spectrum is largely unknown, but expected to be as follows:



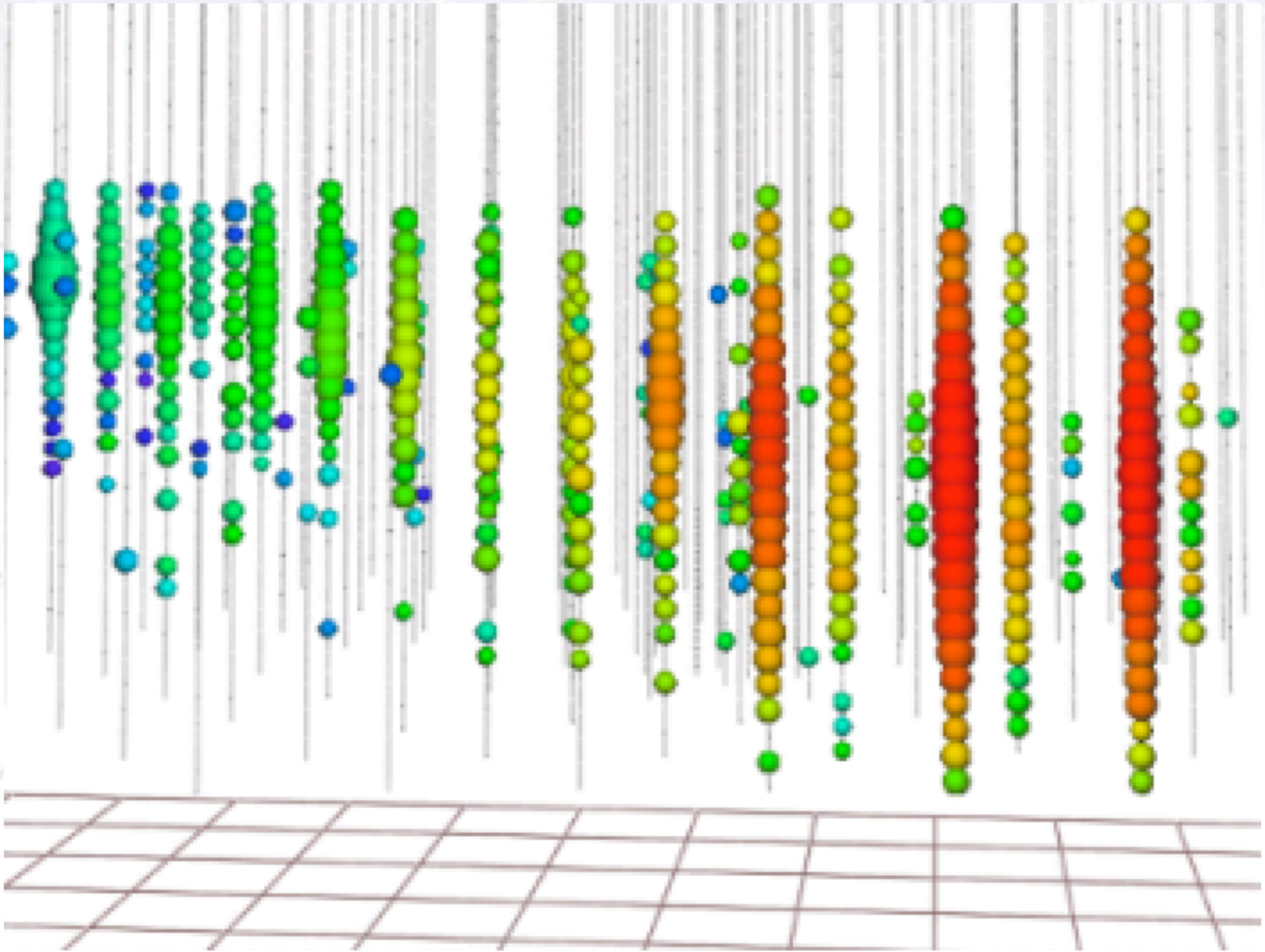
Signals in IceCube

The neutrino spectrum is largely unknown, but expected to be as follows:



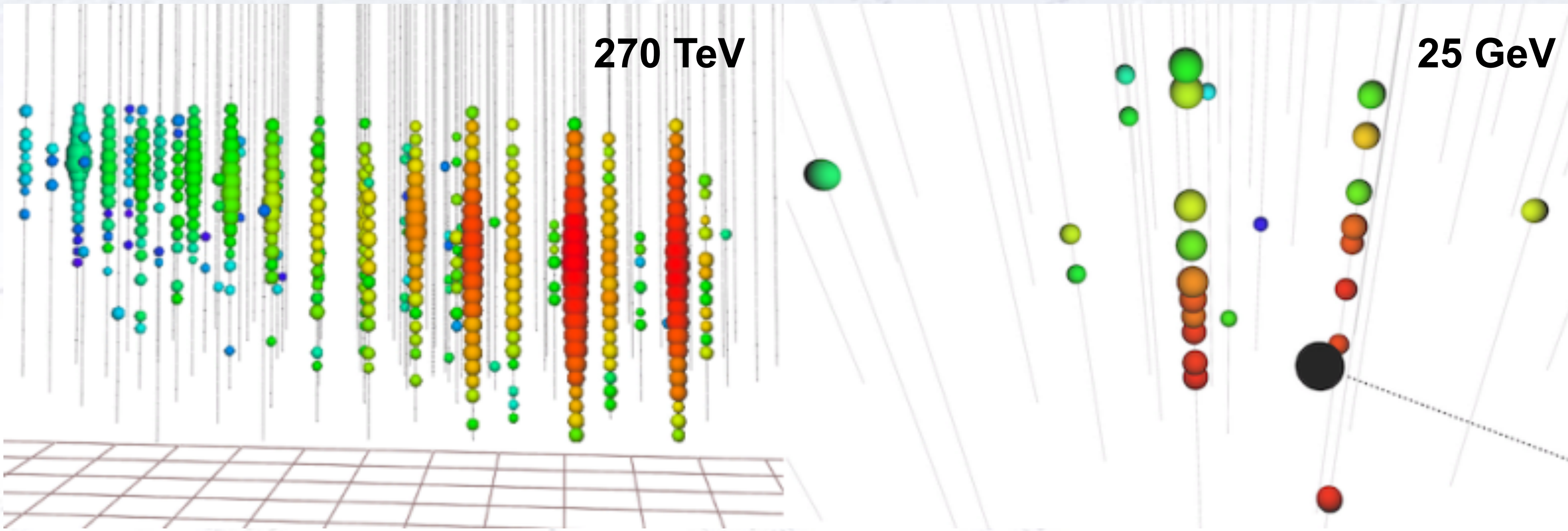
Signals in IceCube

The trigger/read-out rate is about 2500 Hz, while the true neutrino rate is mHz, i.e. one every 6 minutes! You might imagine these looks like below…



Signals in IceCube

The trigger/read-out rate is about 2500 Hz, while the true neutrino rate is mHz, i.e. one every 6 minutes! You might imagine these looks like below…



…but - alas - that is not the case! Most have very low energy, and result in 10-20 signal pulses, which are mixed with a similar amount of background pulses from noise.

Here, the challenge starts!

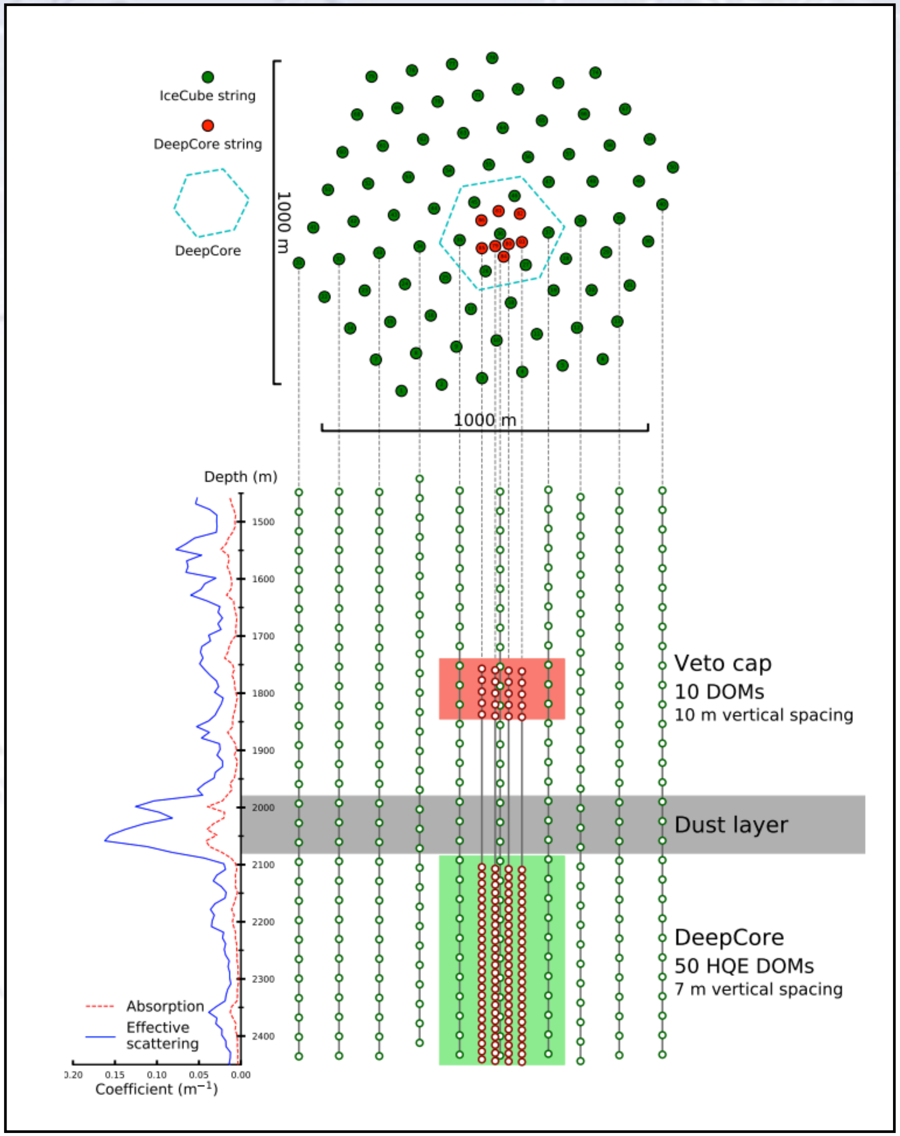
More complications

On top of the irregular detector geometry, the ice also has its irregularities.

Different depths have different absorption and scattering lengths, due to **dust layers** from past ages.

Furthermore, the ice in the holes that the strings are installed in has **bubbles**.

Finally, the ice seems to have a preferred direction, due to **crystal structures** in the glacial flow.

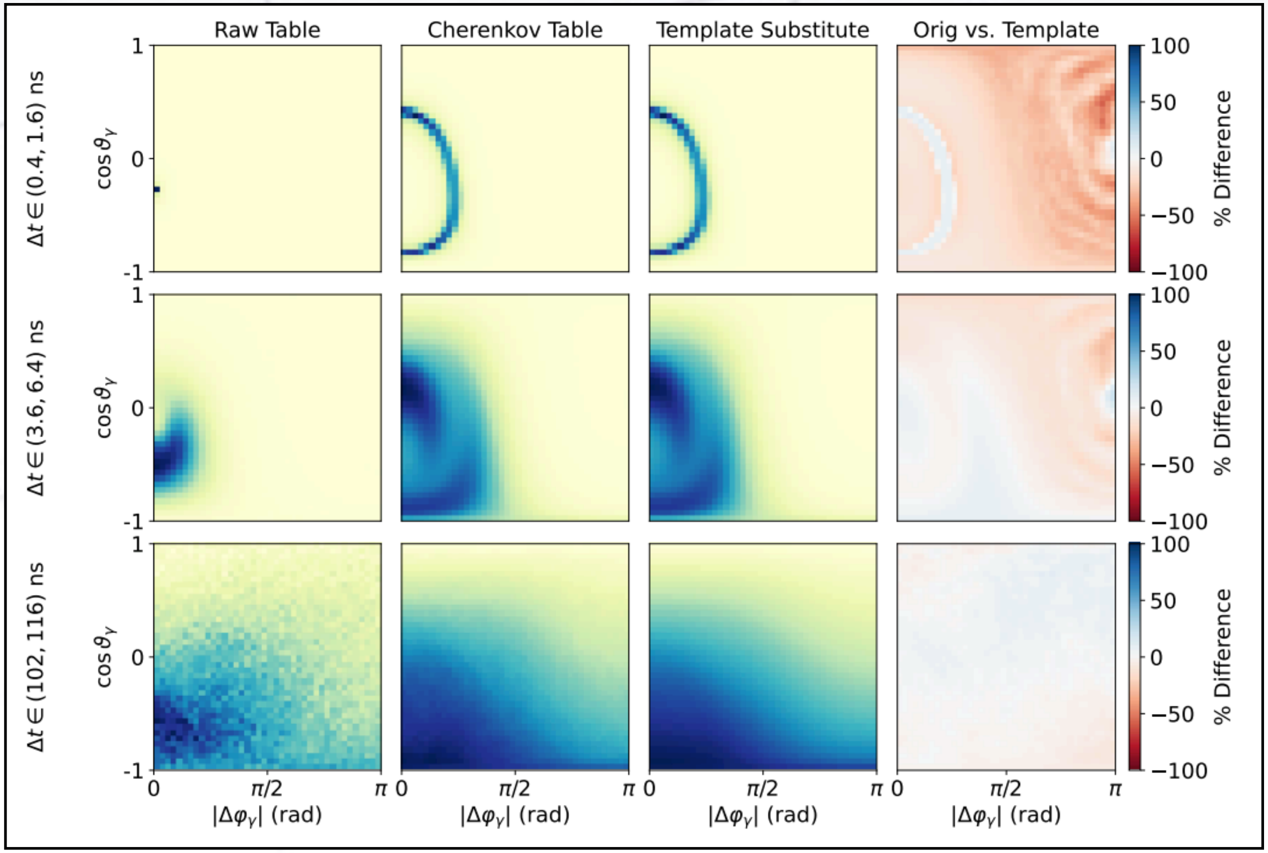


Current reconstruction methods

Current reconstruction methods

The **RETRO reconstruction algorithm** was created around 2019, and is a likelihood based method for LOW energy neutrinos (Arxiv: 2203.02303).

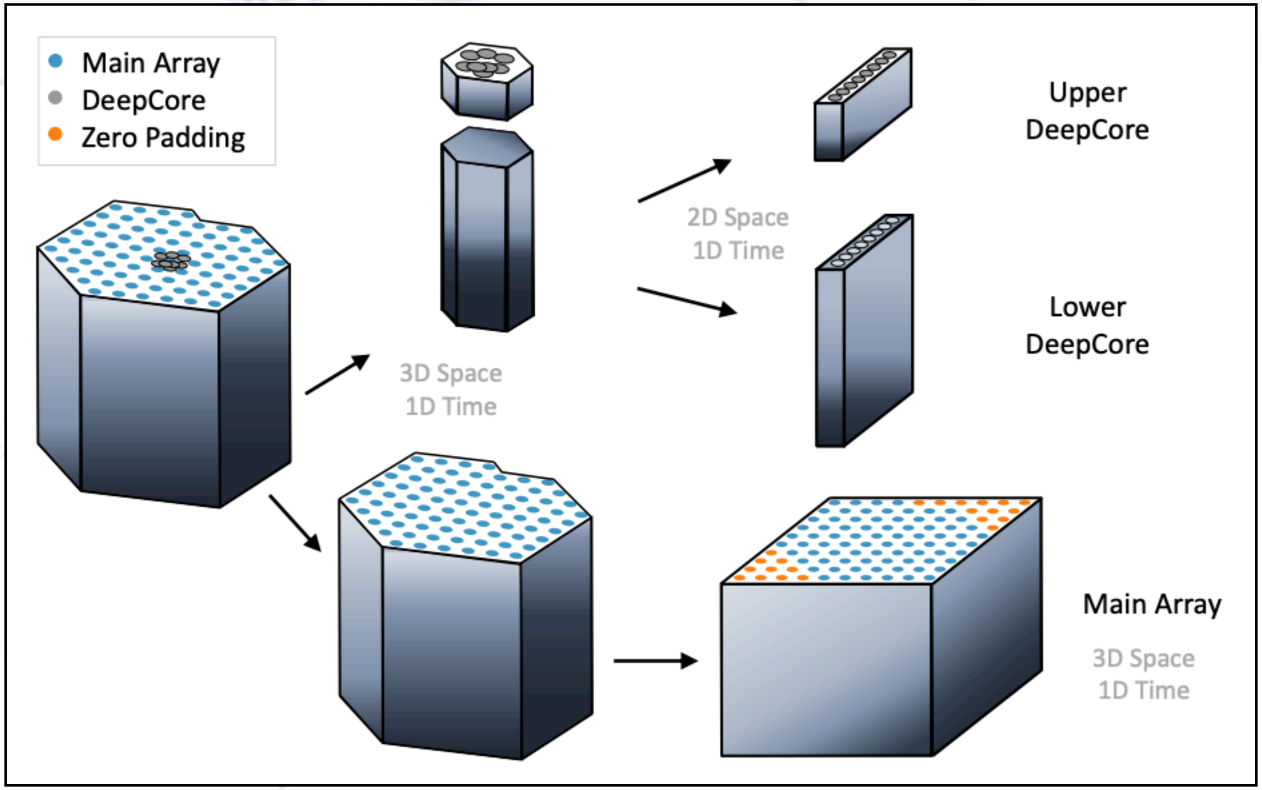
It uses large tables created from simulating millions of neutrino interactions, to create templates. While very accurate, it takes about 10-40 seconds to run.



Current reconstruction methods

The **DNN reconstruction algorithm** was created in 2021, and is a CNN based method for HIGH energy neutrinos (Arxiv: 2101.11589).

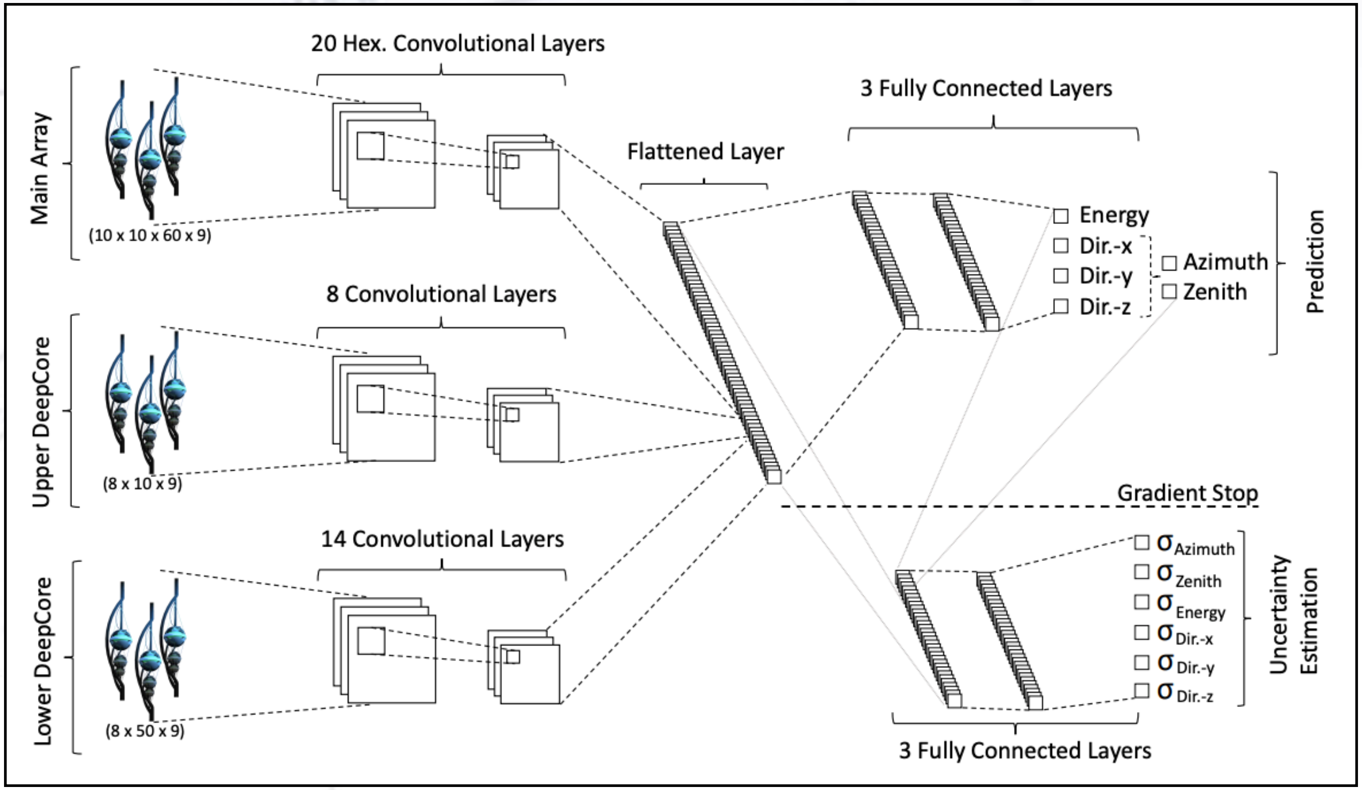
It subdivides the irregular IceCube array, and applies a CNN approach to each of these, which are then combined in a "normal" NN. It takes about 1ms to run.



Current reconstruction methods

The **DNN reconstruction algorithm** was created in 2021, and is a CNN based method for HIGH energy (cascade: e/tau) neutrinos (Arxiv: 2101.11589).

It subdivides the irregular IceCube array, and applies a CNN approach to each of these, which are then combined in a "normal" NN. It takes about 1ms to run.



Reconstruction discussion

The **RETRO algorithm** is very smart, and uses known physics along with a good deal of optimisation. The result is very accurate predictions.

The drawback is the speed. At 10-40 seconds/event, it takes months to run through and reconstruct the IceCube dataset (11 years!).

The **DNN algorithm** is also very smart. It utilises the fact that it is OK to introduce various transformations of the input data, as long as a Neural Network at the end gets a chance to combine it. Like all ML methods, it is also fast.

However, the method works best for high energy neutrinos, which have a large number of pulses (1000+), while details (and hence performance) are partially lost at lower energies.

Reconstruction discussion

The **RETRO algorithm** is very smart, and uses known physics along with a good deal of optimisation. The result is very accurate predictions.

The drawback is the speed. At 10-40 seconds/event, it takes months to run through and reconstruct the IceCube dataset (11 years!).

The **DNN algorithm** is also very smart. It utilises the fact that it is OK to introduce various transformations of the input data, as long as a Neural Network at the end gets a chance to combine it. Like all ML methods, it is also fast.

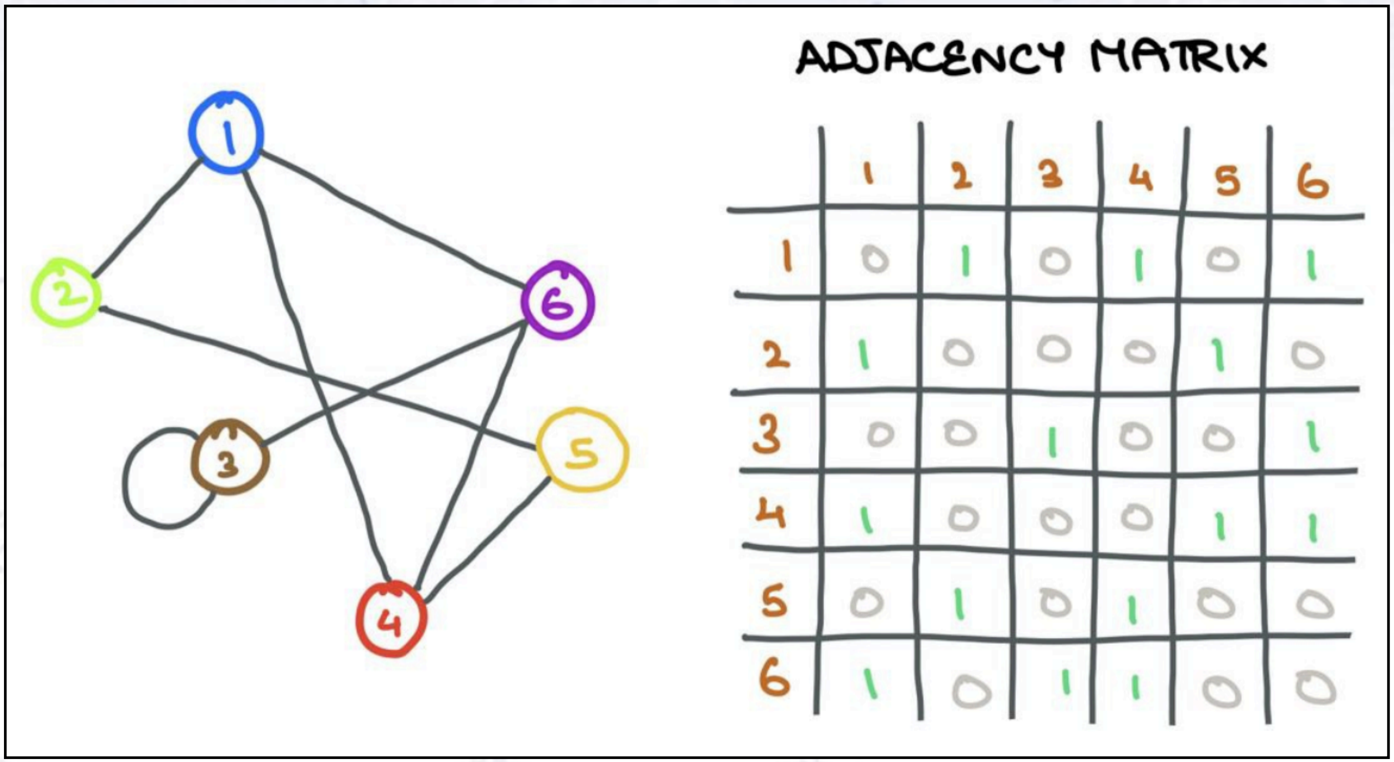
However, the method works best for high energy neutrinos, which have a large number of pulses (1000+), while details (and hence performance) are partially lost at lower energies.

How to solve the problem of variable input size and irregular geometry?

Enter Graph Neural Networks

Graphs

A graph consists of nodes (containing information) and edges (node connections). The connectivity does not need to be bijective, and the number of connections may vary (but doesn't in our case).



In the case of IceCube, the pulses (or DOMs) are the nodes, and these are connected in "some way", upon which a Neural Network is applied.

Graph Neural Networks

A Graph Neural Network (GNN) is a Neural Network applied to a graph, that is it takes a graph as input, and outputs as an NN.

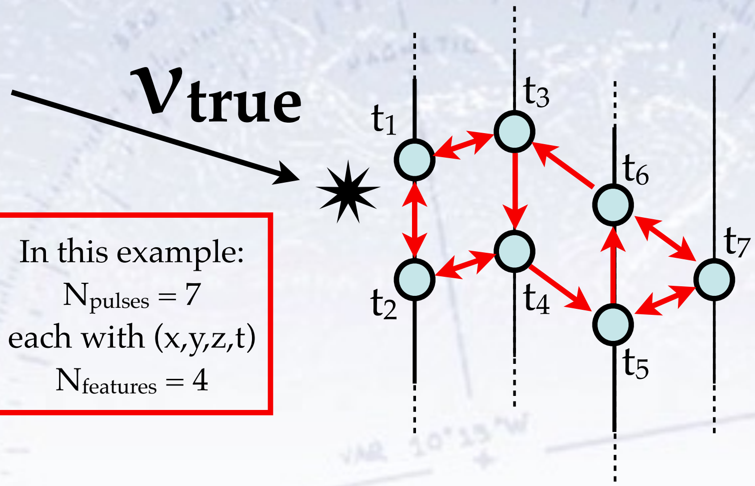
Just like in a Convolutional Neural Network (CNN), the process consists of applying several convolutions. In the GNN case, this is done with a convolutional operator. An example of this is EdgeConv (Arxiv: 1801.07829):

$$x^{\text{update}} = \sum_{i}^{\text{Neighbours}} NN(x, x_i)$$

In the end, all the nodes are aggregated together, resulting in a fixed size vector to apply an ordinary NN on, yielding the desired estimates.

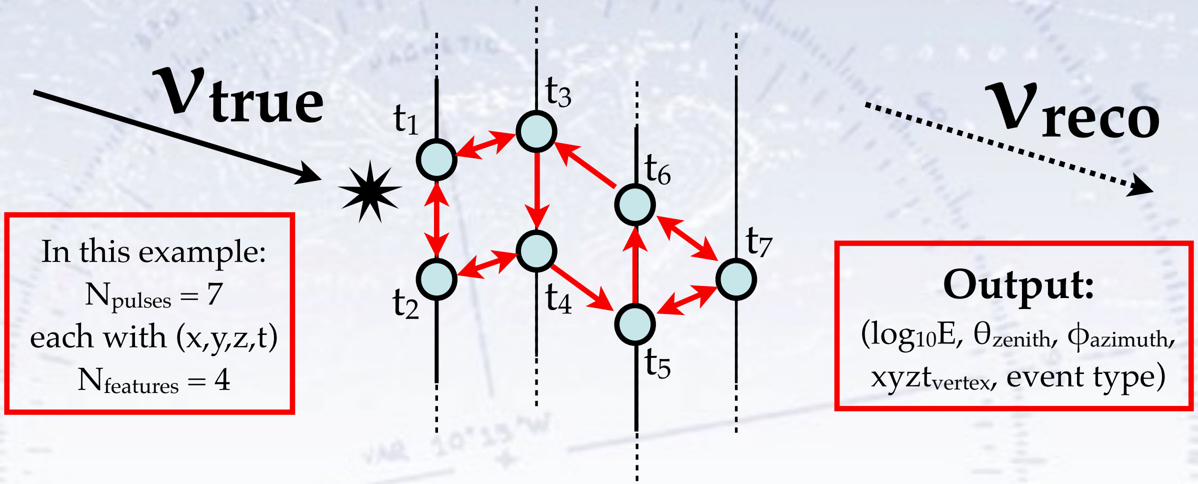
Sounds complicated? Let us have a look in more detail…

Details of GNN reconstruction

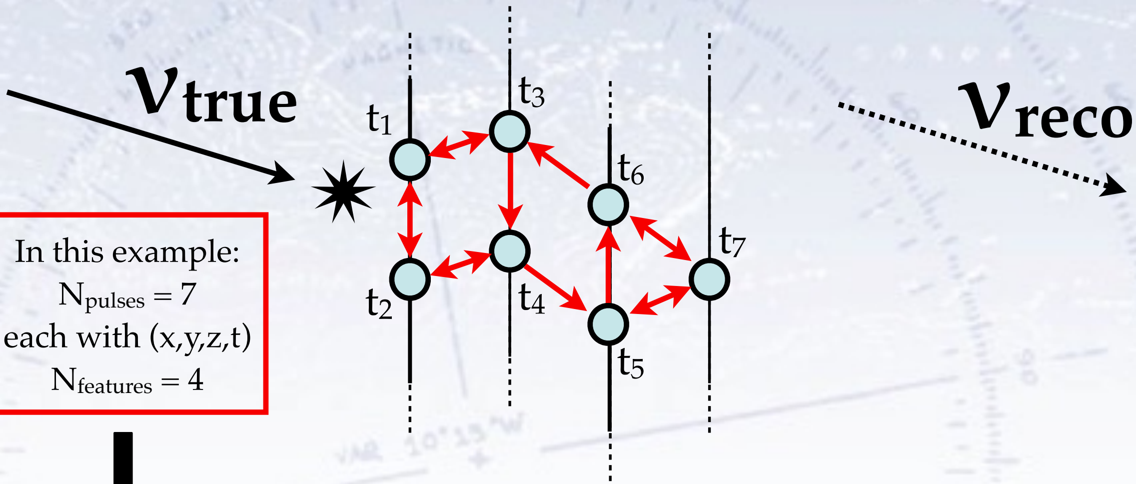


In this example:
$N_{pulses} = 7$
each with (x,y,z,t)
$N_{features} = 4$

Details of GNN reconstruction



Details of GNN reconstruction



In this example:
$N_{pulses} = 7$
each with (x,y,z,t)
$N_{features} = 4$

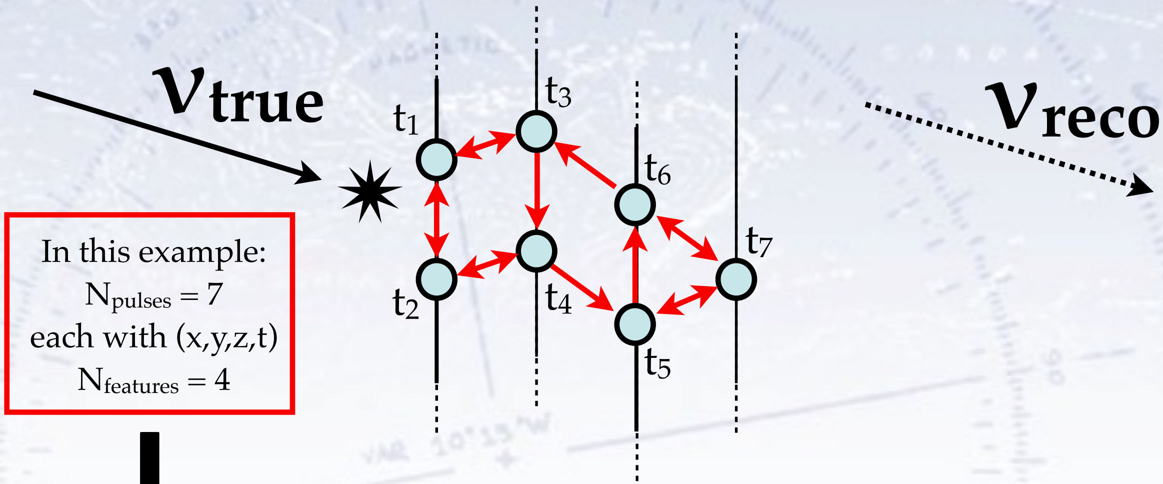
$$\vec{v}_1 = [x_1 \, y_1 \, z_1 \, t_1]$$
$$\vec{v}_2 = [x_2 \, y_2 \, z_2 \, t_2]$$
$$\vdots$$
$$\vec{v}_7 = [x_7 \, y_7 \, z_7 \, t_7]$$

Input:

$N = N_{pulses} \times N_{features}$

The input features of a node are combined with that of N (=2) nearby nodes

Details of GNN reconstruction



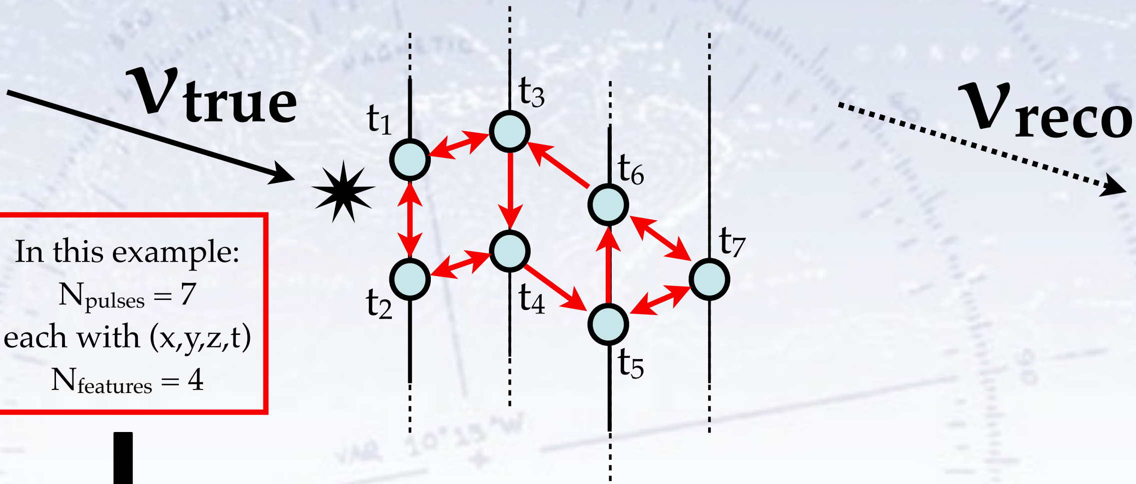
$$
\begin{array}{lcl}
\vec{v}_1 = [x_1\, y_1\, z_1\, t_1] & \xrightarrow{EC(\vec{v}_1,\vec{v}_2,\vec{v}_3)} & [g_{11} \dots g_{1N_1}] \\
\vec{v}_2 = [x_2\, y_2\, z_2\, t_2] & & [g_{21} \dots g_{2N_1}] \\
\vdots & \xrightarrow{EC(\vec{v}_4,\vec{v}_5,\vec{v}_6)} & \vdots \\
\vec{v}_7 = [x_7\, y_7\, z_7\, t_7] & & [g_{71} \dots g_{7N_1}]
\end{array}
$$

Input:
$N = N_{pulses} \times N_{features}$

Convolution(s):
$N = N_{pulses} \times N_1$

The input features of a node are combined with that of N (=2) nearby nodes through an NN (MLP0) function, yielding an (abstract) vector for each node. This can be repeated (not shown).

Details of GNN reconstruction



In this example:
$N_{pulses} = 7$
each with (x,y,z,t)
$N_{features} = 4$

$N_{all} = N_{features} + N_1$

$$\vec{v}_1 = [x_1\, y_1\, z_1\, t_1] \xrightarrow{EC(\vec{v}_1, \vec{v}_2, \vec{v}_3)} [g_{11} \dots g_{1N_1}] \qquad [x_1\, y_1\, z_1\, t_1\;\; g_{11} \dots g_{1N_1}]$$

$$\vec{v}_2 = [x_2\, y_2\, z_2\, t_2] \qquad [g_{21} \dots g_{2N_1}] \qquad [x_2\, y_2\, z_2\, t_2\;\; g_{21} \dots g_{2N_1}]$$

$$\vdots \xrightarrow{EC(\vec{v}_4, \vec{v}_5, \vec{v}_6)} \vdots \qquad \vdots$$

$$\vec{v}_7 = [x_7\, y_7\, z_7\, t_7] \qquad [g_{71} \dots g_{7N_1}] \qquad [x_7\, y_7\, z_7\, t_7\;\; g_{71} \dots g_{7N_1}]$$

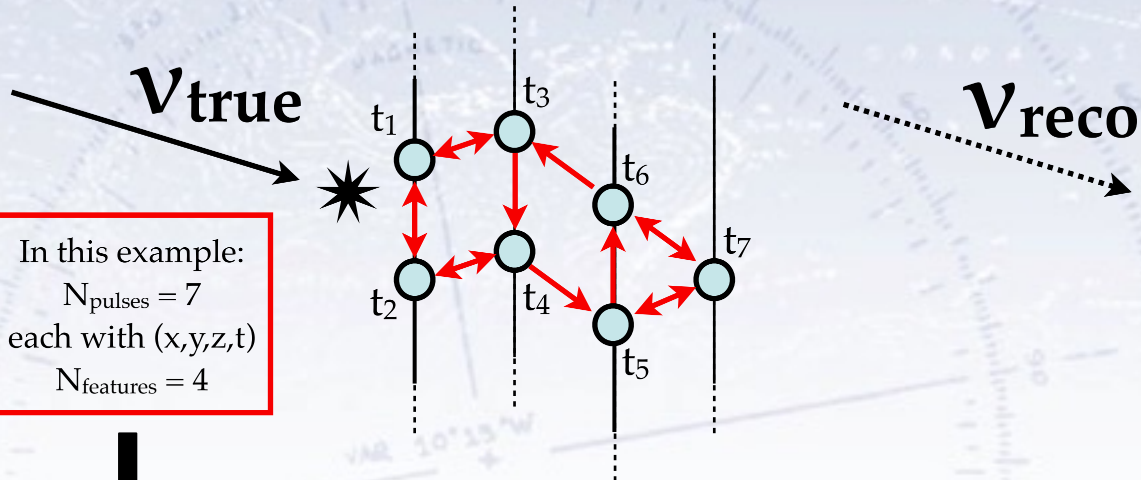
Input:
$N = N_{pulses} \times N_{features}$

Convolution(s):
$N = N_{pulses} \times N_1$

Concatenation:
$N = N_{pulses} \times N_{all}$

The input features of a node are combined with that of N (=2) nearby nodes through an NN (MLP0) function, yielding an (abstract) vector for each node. This can be repeated (not shown). All the features are then combined (concatenated) into long vectors,

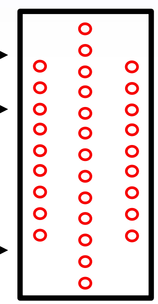
Details of GNN reconstruction



In this example:
$N_{pulses} = 7$
each with (x,y,z,t)
$N_{features} = 4$

$N_{all} = N_{features} + N_1$

$$
\begin{array}{lcccc}
\vec{v}_1 = [x_1\, y_1\, z_1\, t_1] & \xrightarrow{EC(\vec{v}_1, \vec{v}_2, \vec{v}_3)} & [g_{11} \ldots g_{1N_1}] & [x_1\, y_1\, z_1\, t_1\;\; g_{11} \ldots g_{1N_1}] \rightarrow & \rightarrow [h_{11} \ldots h_{1N_{all}}] \\
\vec{v}_2 = [x_2\, y_2\, z_2\, t_2] & & [g_{21} \ldots g_{2N_1}] & [x_2\, y_2\, z_2\, t_2\;\; g_{21} \ldots g_{2N_1}] \rightarrow & \rightarrow [h_{21} \ldots h_{2N_{all}}] \\
\vdots & \xrightarrow{EC(\vec{v}_4, \vec{v}_5, \vec{v}_6)} & \vdots & \vdots & \vdots \\
\vec{v}_7 = [x_7\, y_7\, z_7\, t_7] & & [g_{71} \ldots g_{7N_1}] & [x_7\, y_7\, z_7\, t_7\;\; g_{71} \ldots g_{7N_1}] \rightarrow & \rightarrow [h_{71} \ldots h_{7N_{all}}]
\end{array}
$$



Input: $N = N_{pulses} \times N_{features}$

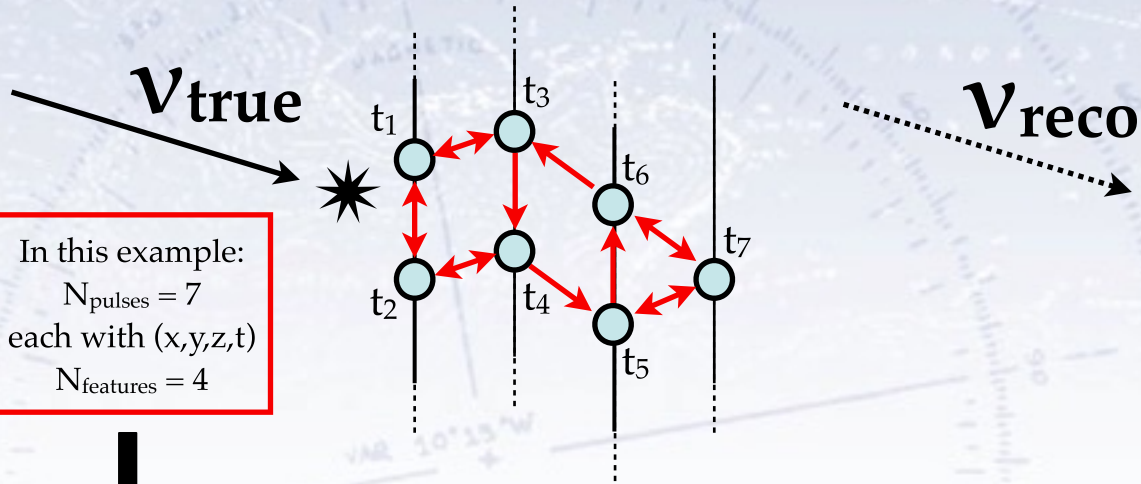
Convolution(s): $N = N_{pulses} \times N_1$

Concatenation: $N = N_{pulses} \times N_{all}$

MLP1: $N_{all} \rightarrow N_{many} \rightarrow N_{all}$

The input features of a node are combined with that of N (=2) nearby nodes through an NN (MLP0) function, yielding an (abstract) vector for each node. This can be repeated (not shown). All the features are then combined (concatenated) into long vectors, which are again put through an NN (MLP1) function with a large hidden layer.

Details of GNN reconstruction



In this example:
$N_{pulses} = 7$
each with (x,y,z,t)
$N_{features} = 4$

$[\Sigma_1 \ldots \Sigma_{N_{all}}]$
$[\mu_1 \ldots \mu_{N_{all}}]$
$[\wedge_1 \ldots \wedge_{N_{all}}]$
$[\vee_1 \ldots \vee_{N_{all}}]$

sum(Σ), mean(μ) ↑ max($\wedge$), min($\vee$)

$N_{all} = N_{features} + N_1$

$$\vec{v}_1 = [x_1\, y_1\, z_1\, t_1] \xrightarrow{EC(\vec{v}_1, \vec{v}_2, \vec{v}_3)} [g_{11} \ldots g_{1N_1}] \quad [x_1\, y_1\, z_1\, t_1\;\; g_{11} \ldots g_{1N_1}] \rightarrow \text{MLP1} \rightarrow [h_{11} \ldots h_{1N_{all}}]$$

$$\vec{v}_2 = [x_2\, y_2\, z_2\, t_2] \qquad [g_{21} \ldots g_{2N_1}] \quad [x_2\, y_2\, z_2\, t_2\;\; g_{21} \ldots g_{2N_1}] \rightarrow \text{MLP1} \rightarrow [h_{21} \ldots h_{2N_{all}}]$$

$$\vdots \xrightarrow{EC(\vec{v}_4, \vec{v}_5, \vec{v}_6)} \vdots \qquad \vdots \qquad \vdots$$

$$\vec{v}_7 = [x_7\, y_7\, z_7\, t_7] \qquad [g_{71} \ldots g_{7N_1}] \quad [x_7\, y_7\, z_7\, t_7\;\; g_{71} \ldots g_{7N_1}] \rightarrow \text{MLP1} \rightarrow [h_{71} \ldots h_{7N_{all}}]$$

Input: $N = N_{pulses} \times N_{features}$

Convolution(s): $N = N_{pulses} \times N_1$

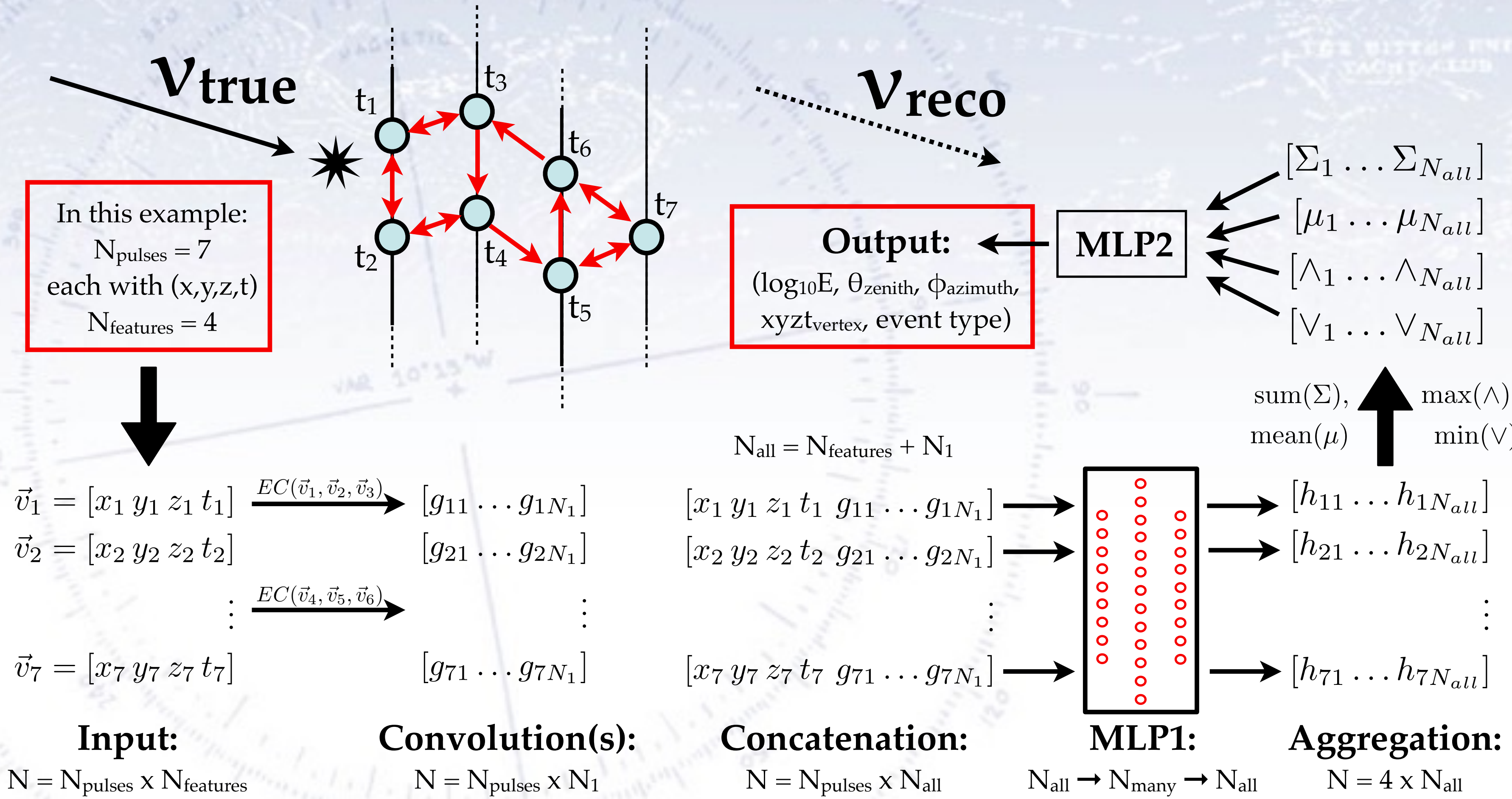
Concatenation: $N = N_{pulses} \times N_{all}$

MLP1: $N_{all} \rightarrow N_{many} \rightarrow N_{all}$

Aggregation: $N = 4 \times N_{all}$

The input features of a node are combined with that of N (=2) nearby nodes through an NN (MLP0) function, yielding an (abstract) vector for each node. This can be repeated (not shown). All the features are then combined (concatenated) into long vectors, which are again put through an NN (MLP1) function with a large hidden layer. The outputs are aggregated in four ways: Min, Max, Sum & Mean, breaking the variation with number of nodes.

Details of GNN reconstruction



The input features of a node are combined with that of N (=2) nearby nodes through an NN (MLP0) function, yielding an (abstract) vector for each node. This can be repeated (not shown). All the features from all the convolutions are then combined (concatenated) into long vectors, which are again put through an NN (MLP1) function with a large hidden layer. The outputs are aggregated in four ways: Min, Max, Sum & Mean, breaking the variation with number of nodes. These are then fed into a final NN (MLP2), which outputs the estimated type(s) and parameters of the event.

Further specifics of DynEdge

In DynEdge, there are several "enlargements" compared to the previous illustration of the GNN architecture. These are essentially:

- We use 6 input features: x, y, z, t, charge, and Quantum Efficiency.
- We convolute each node with the nearest 8 nodes (not two).
- We do 4 (not 1) convolutions, each with 256 inputs and outputs.
- The concatenation is of all convolution layers and the original input.
- We also include global statistics in the final NN.
- In the results to be shown, we have trained separate GNNs for each output.

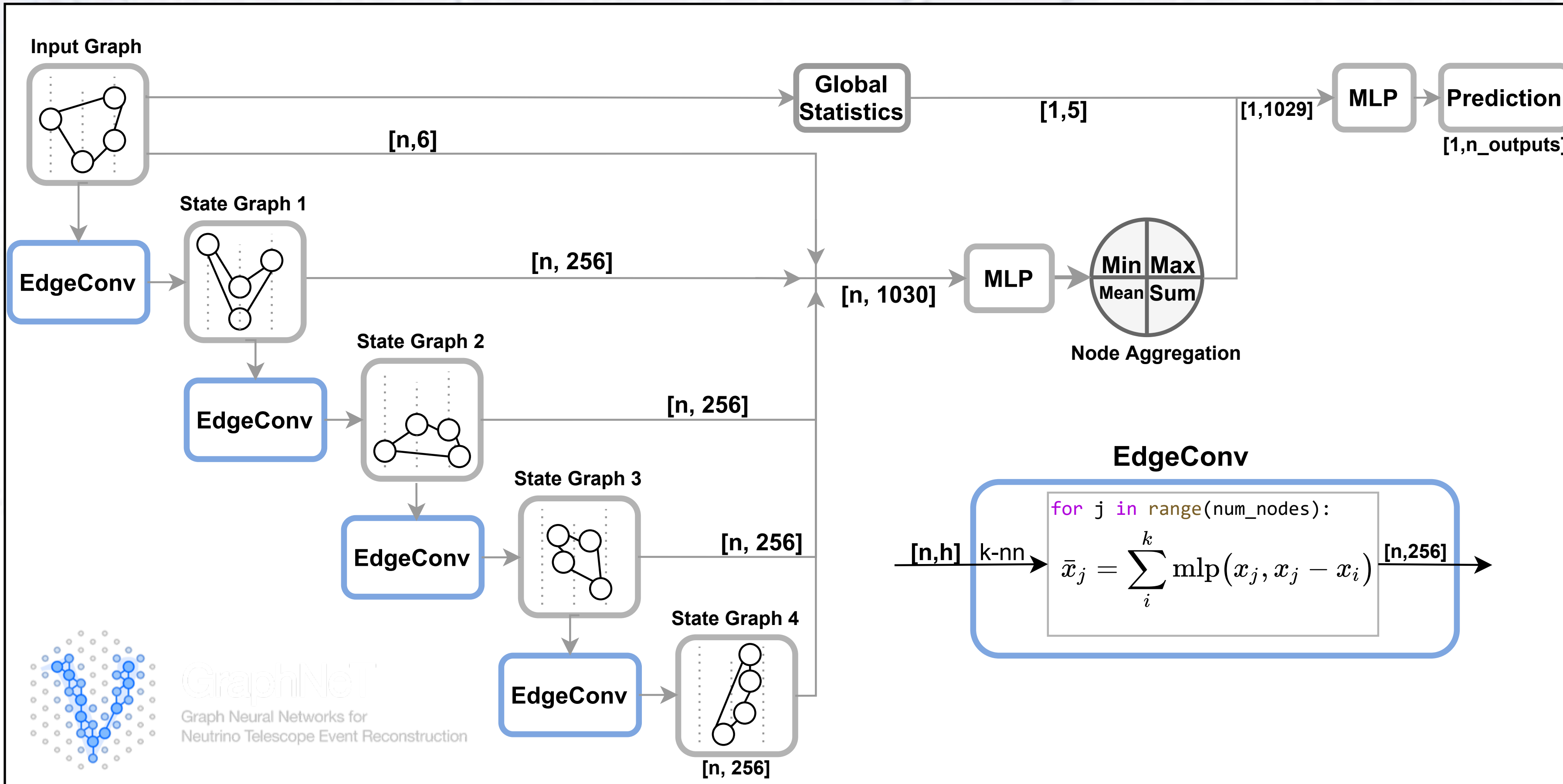
The repeated convolutions allows all signal parts to be connected.
The EdgeConv convolution operator ensures permutation invariance.

The number of model parameters is about 750.000 for the angular regressions, while the energy only requires 150.000. In principle one can go down to 70.000 parameters, but there is no reason for this - it is already a "small" ML model.

The GNN model is build in PyTorch Geometric using "DynEdge" architecture.

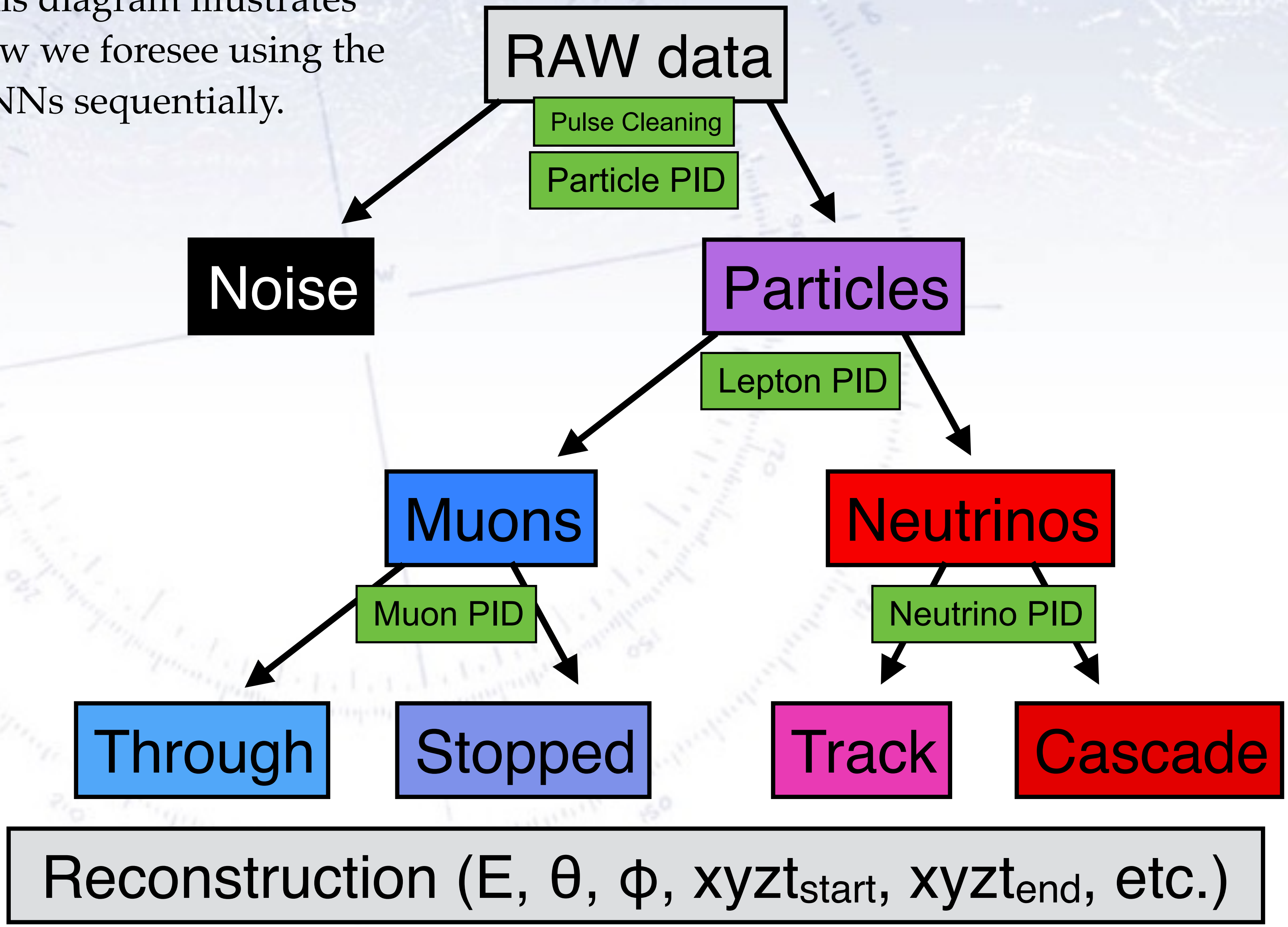
GraphNet

Our GNN model is outlined below, and is part of a larger package that we call GraphNet. Using GraphNet, we want to make it "simple" to reconstruct data from Neutrino Telescopes in general.



GNN classification overview

This diagram illustrates how we foresee using the GNNs sequentially.



GraphNet - Team

GraphNeT
Graph Neural Networks for
Neutrino Telescope Event Reconstruction

GraphNet is our attempt at putting GNN models for IceCube (and others) using the "DynEdge" architecture build in PyTorch Geometric into an easily available software package.

https://github.com/graphnet-team/graphnet

Troels C. Petersen

Project part: Inspiration, physics, detector, and coordination.
Period: First thoughts (with Andreas) in 2018.
Type: Regular job!
Goal: A great ML reconstruction, and fun getting there!

Andreas Soegaard

Project part: Eventually, probably all parts
Period: September 2021 (Marie-Curie Fellow).
Email: andreas.soegaard@nbi.ku.dk?
Result: GitHub repository.

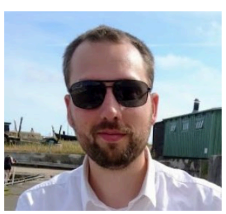
Philipp Eller

Kaare Endrup Iversen

Project part: GNN Upgrade reconstruction, Neutrino oscillation analysis
Period: August 2021 - May 2022 (Master Thesis).
Email: nvc889@alumni.ku.dk
Result: GitHub repository.

Leon Bozianu

Project part: GNN classification and reconstruction of muons, MC-data calibration
Period: August 2021 - May 2022 (Master Thesis).
Email: qzr746@@alumni.ku.dk
Result: GitHub repository.

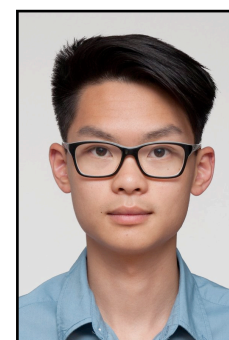


Morten Holm

Project part: GNN reconstruction, Neutrino oscillation analysis?
Period: February 2022 - December 2022 (Master Thesis).
Email: qgf305@alumni.ku.dk
Result: GitHub repository.

Rasmus F. Oersoe

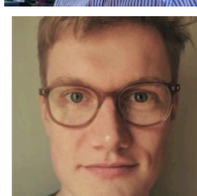
Project part: Graph Neural Net (PyTorch) reconstruction, data curration, etc.
Period: July 2020 - May 2021 (Master Thesis).
Email: pcs557@alumni.ku.dk
Result: GitHub repository.



Martin Minh

Mads Ehrhorn

Project part: CNN and TCN reconstruction, data curration, etc.
Period: September 2019 - February 2021.
Results: Master Thesis, Thesis Defence, and GitHub repository.



Bjoern Moelvig

Project part: RNN/GRU reconstruction, loss function exploration
Period: September 2019 - October 2020.
Results: Master Thesis, Thesis Defence, and GitHub repository.

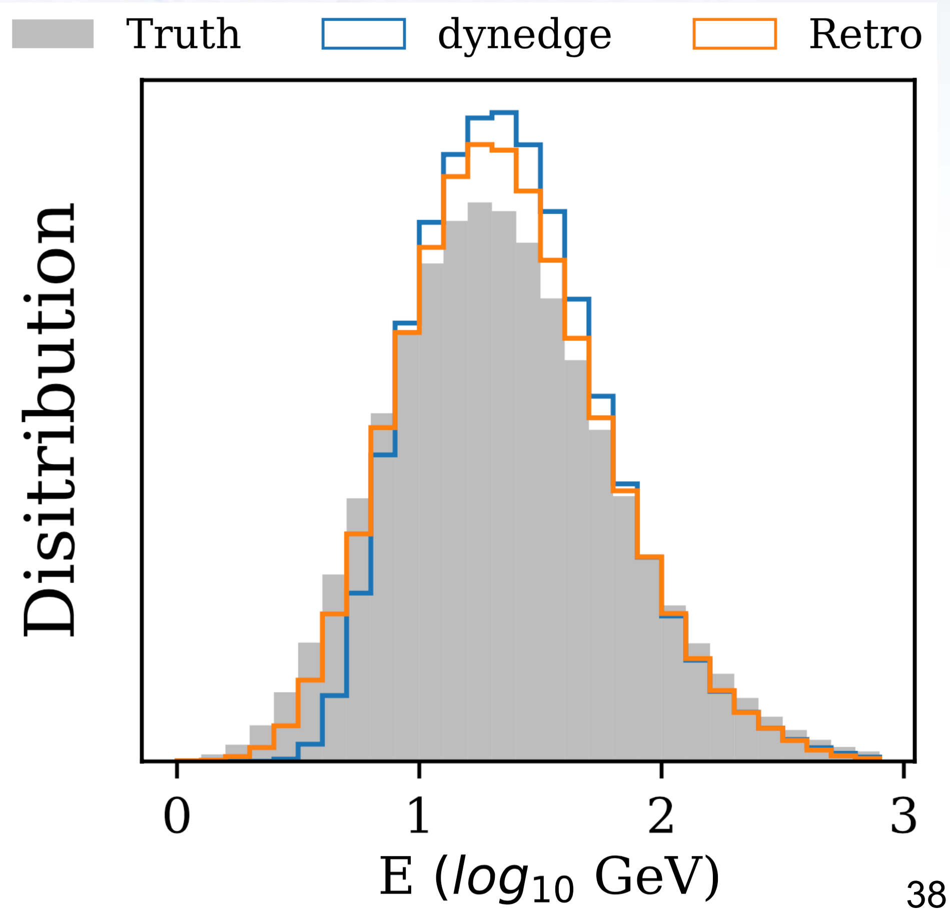
GNN performance

GraphNet - Results

The following slides will show the resulting GNN performance (also compared to RetroReco) for track and cascade neutrino events.

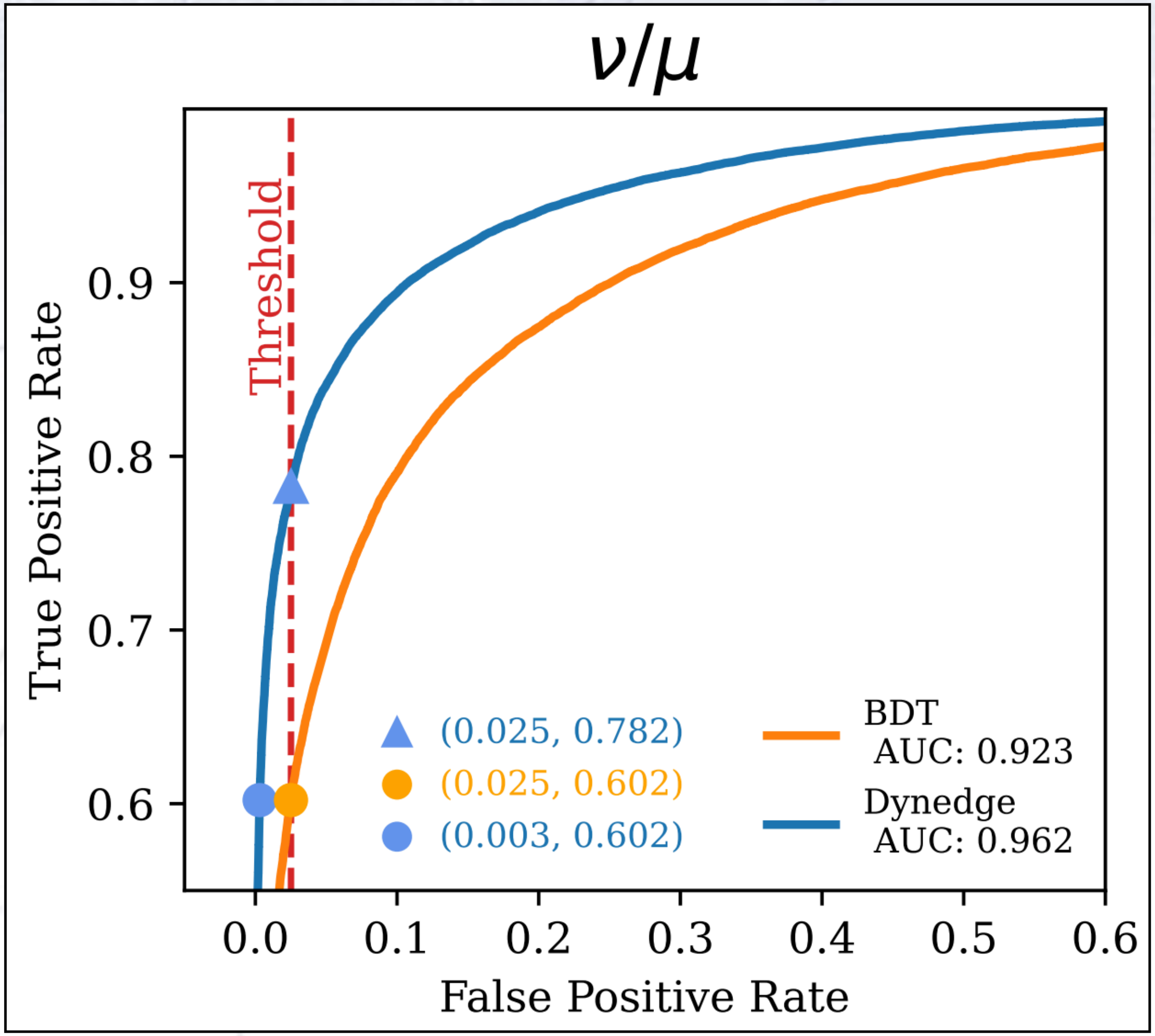
The performance depends on the neutrino energy, so all plots are made as a function of energy.

Since neutrino statistics is limited at low (< 3 GeV) and high (> 300 GeV) energies, the GNN performance there is not expected to be optimal.



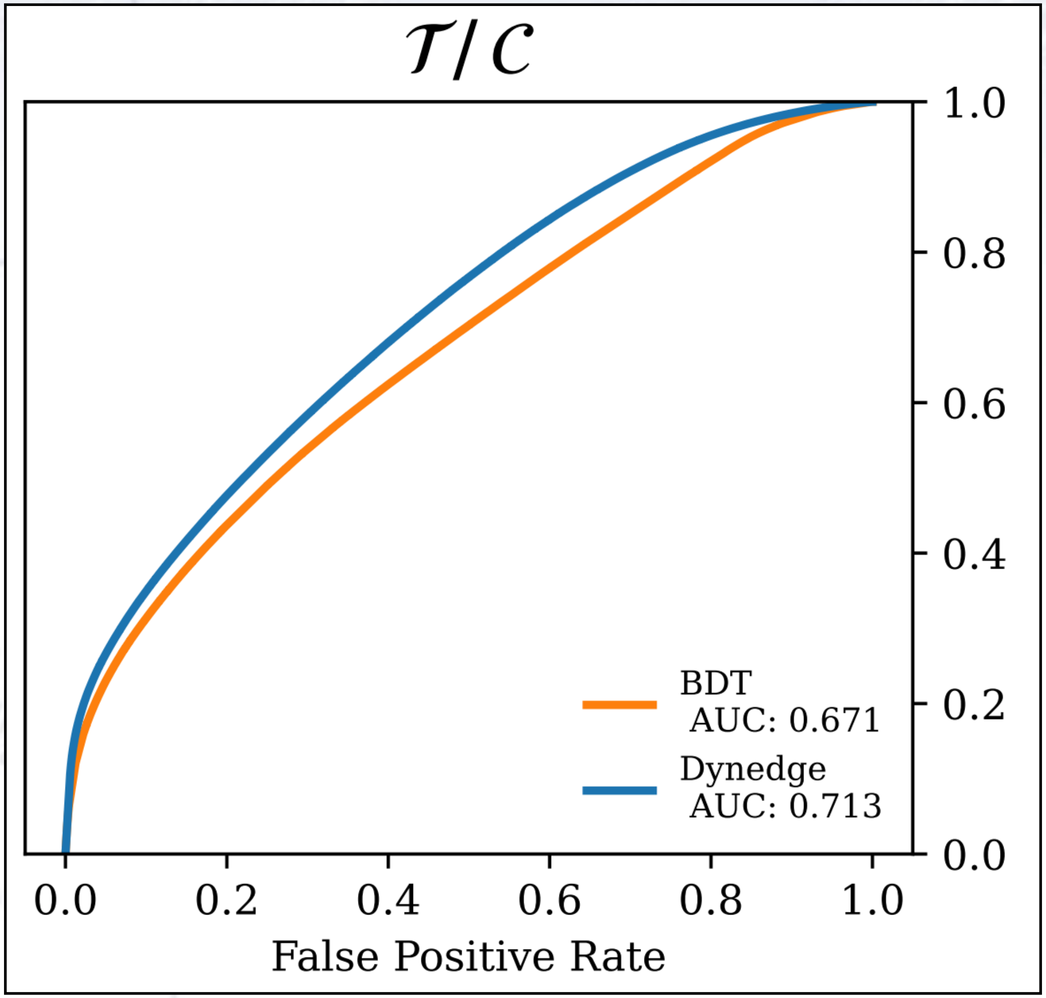
GraphNet - Results

Separation between signal (neutrinos) and background (muons) is improved.



GraphNet - Results

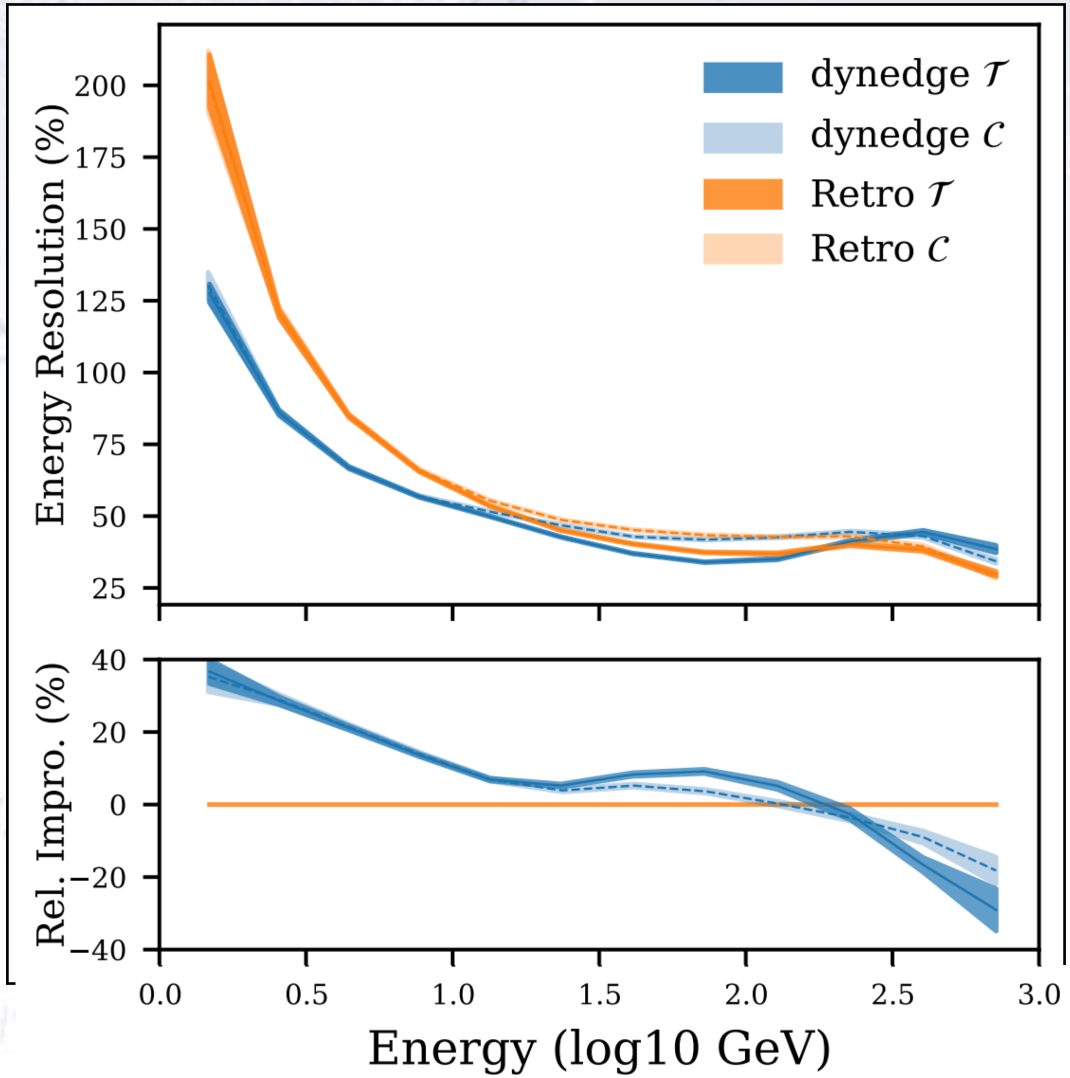
The (difficult) separation between track-like (T, from muons) and Cascade (C, from electrons and taus) is also improved, though less overall.



Reconstruction results

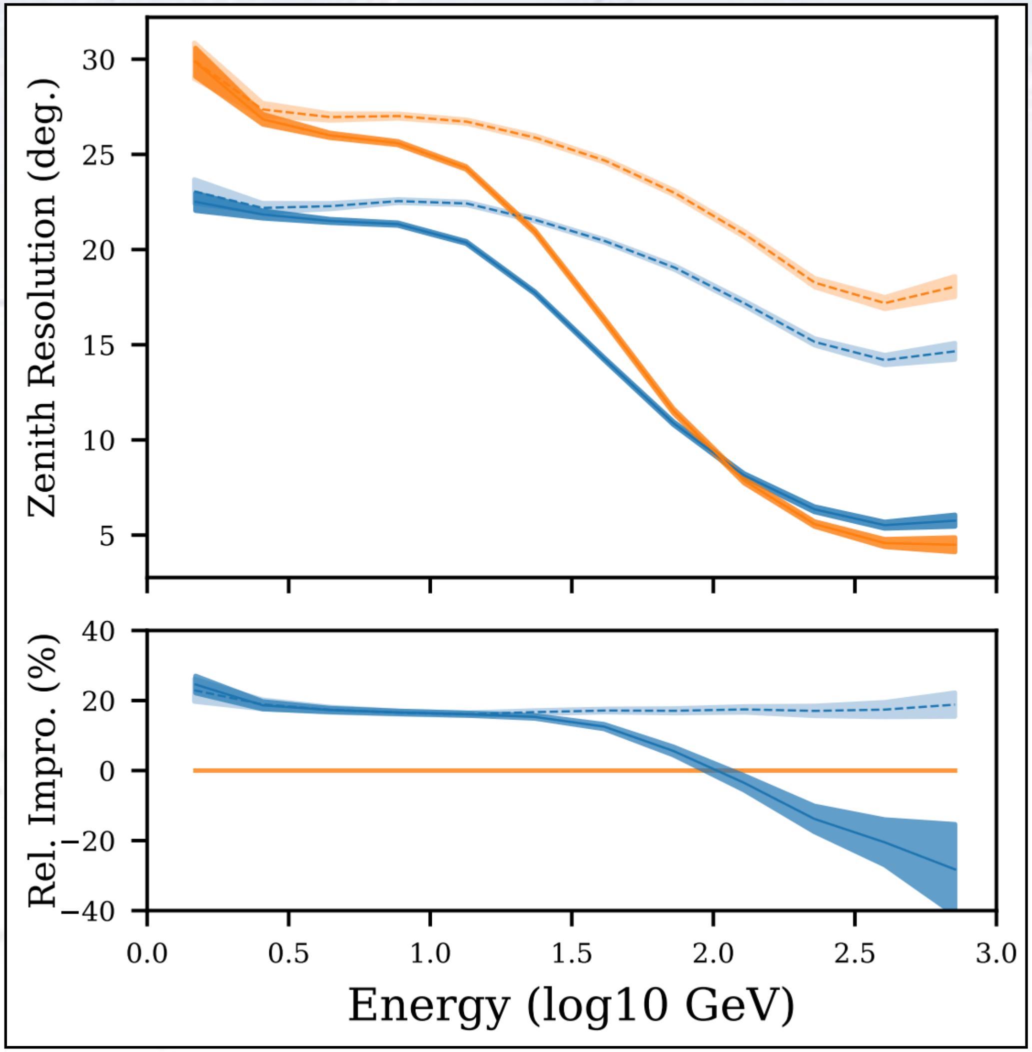
GraphNet - Energy

The energy is optimised by minimising LogCosh of $\log_{10}E_{pred} - \log_{10}E_{true}$



GraphNet - zenith angle

The angular performance uses a 2D Von-Mises Fisher loss function, where the task is to predict (cosθ, sinθ) and an uncertainty parameter.



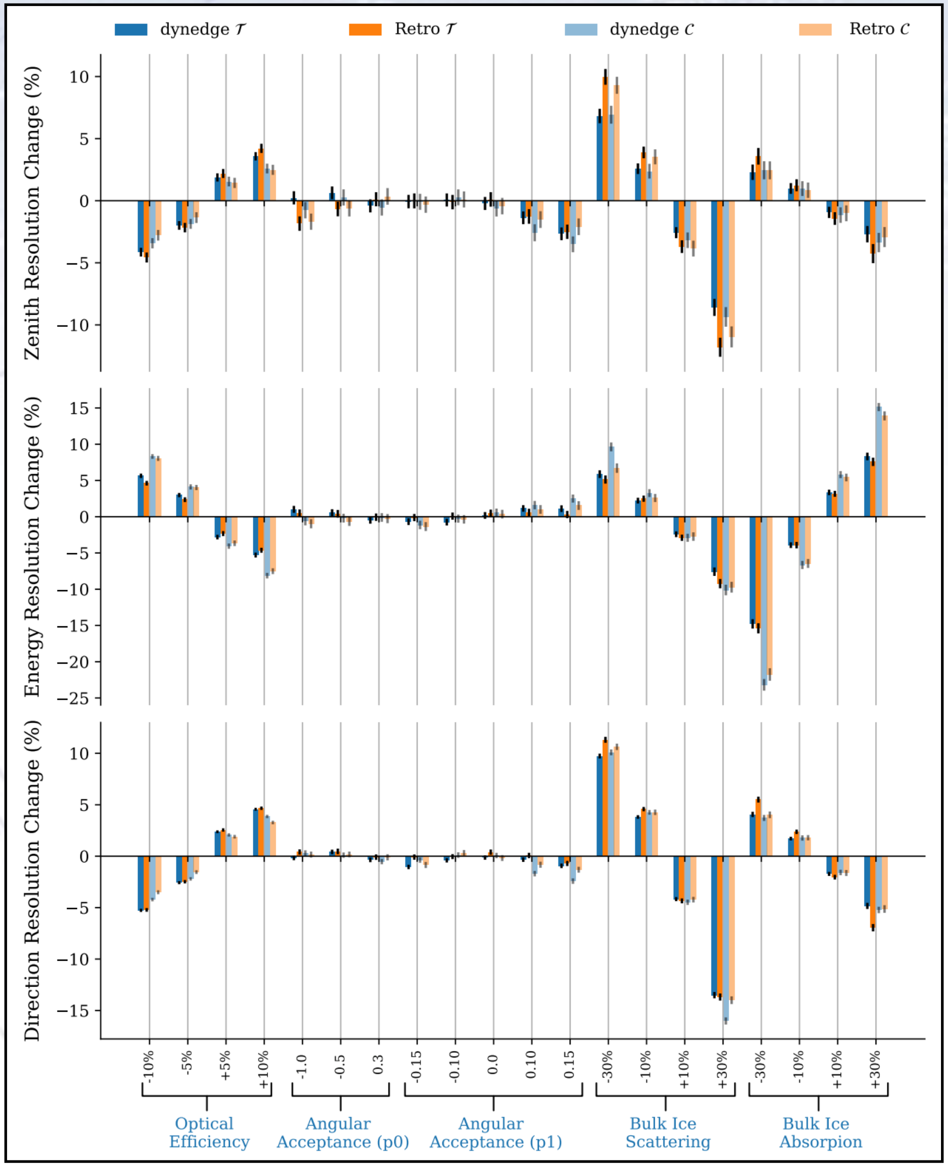
GraphNet - Robustness test

In order to test, to what extend the GNN reconstruction is robust to systematic changes in the ice properties and the detector response, it has been applied to a standard set of simulated systematic data sets.

We see the expected pattern in energy and angle.

The overall conclusion is, that it has the same "good" robustness as the RetroReco algorithm.

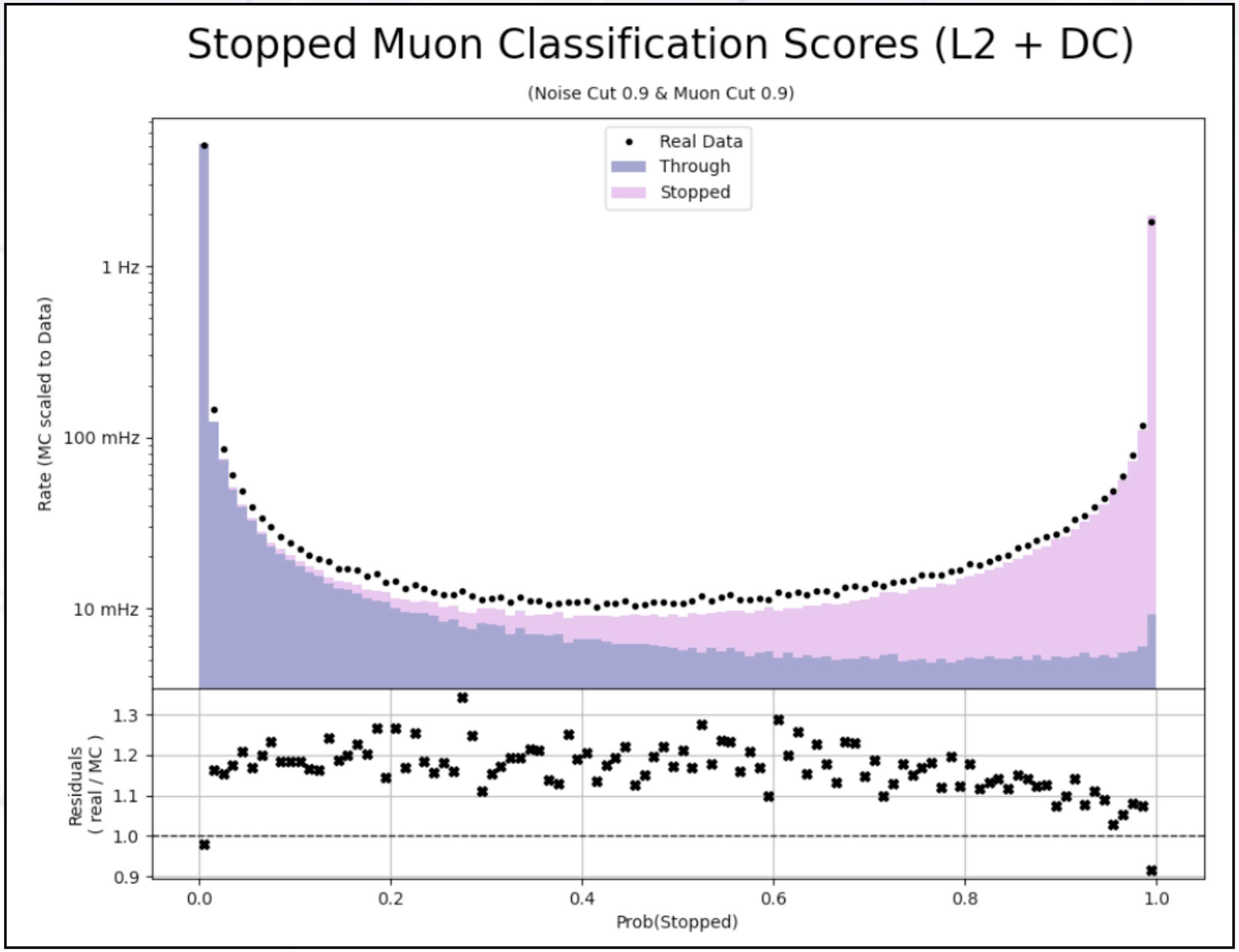
This suggests, that the reconstruction shifts with systematics are inherent, and not possible to avoid.



Hot-of-the-press
Not-even-preliminary

Running on (1 day of) real data

The ability to identify muons, and divide them into "stopped" or not is shown below, with data overlaying MC. The MC still needs scaling, re-weighting, etc…. but as first result on raw data, this is reasonably good:



Speed

GraphNet - Speed

Of course an ML algorithm is faster than a (difficult) likelihood minimisation.

However, the GNN speed never fail to amaze us, illustrated as follows:

The OscNext analysis required the reconstruction of the 21 systematic sets, which is a total of **143.000.000 neutrino events**.

At ~40 s/event, this took 2.5 months on 1000 CPUs. The power cost alone is 10.000+ DKr. We tried to do the same reconstruction on a single GPU…

GraphNet - Speed

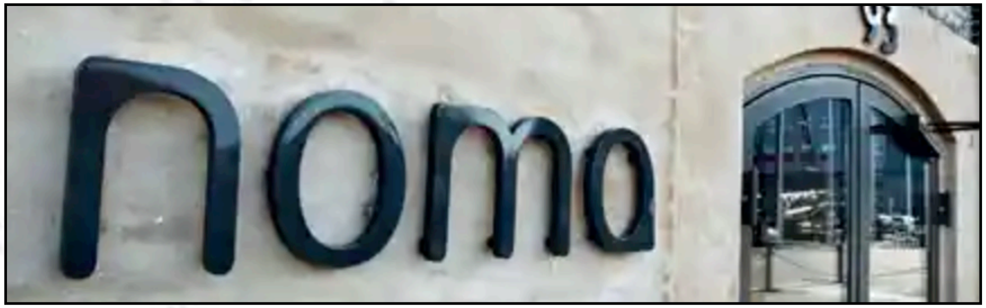
Of course an ML algorithm is faster than a (difficult) likelihood minimisation.

However, the GNN speed never fail to amaze us, illustrated as follows:

The OscNext analysis required the reconstruction of the 21 systematic sets, which is a total of **143.000.000 neutrino events**.

At ~40 s/event, this took 2.5 months on 1000 CPUs. The power cost alone is 10.000+ DKr. We tried to do the same reconstruction on a single GPU…

You could submit the job,
go to NOMA for dinner,
bring a friend/colleague,
get the full menu + wine menu,
have a fantastic evening…



GraphNet - Speed

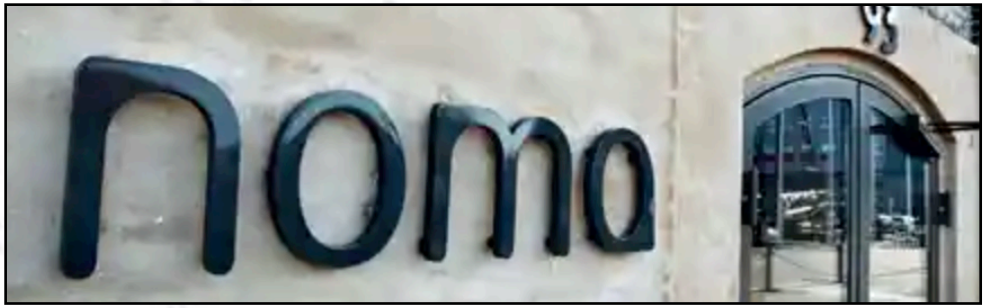
Of course an ML algorithm is faster than a (difficult) likelihood minimisation.

However, the GNN speed never fail to amaze us, illustrated as follows:

The OscNext analysis required the reconstruction of the 21 systematic sets, which is a total of **143.000.000 neutrino events**.

At ~40 s/event, this took 2.5 months on 1000 CPUs. The power cost alone is 10.000+ DKr. We tried to do the same reconstruction on a single GPU…

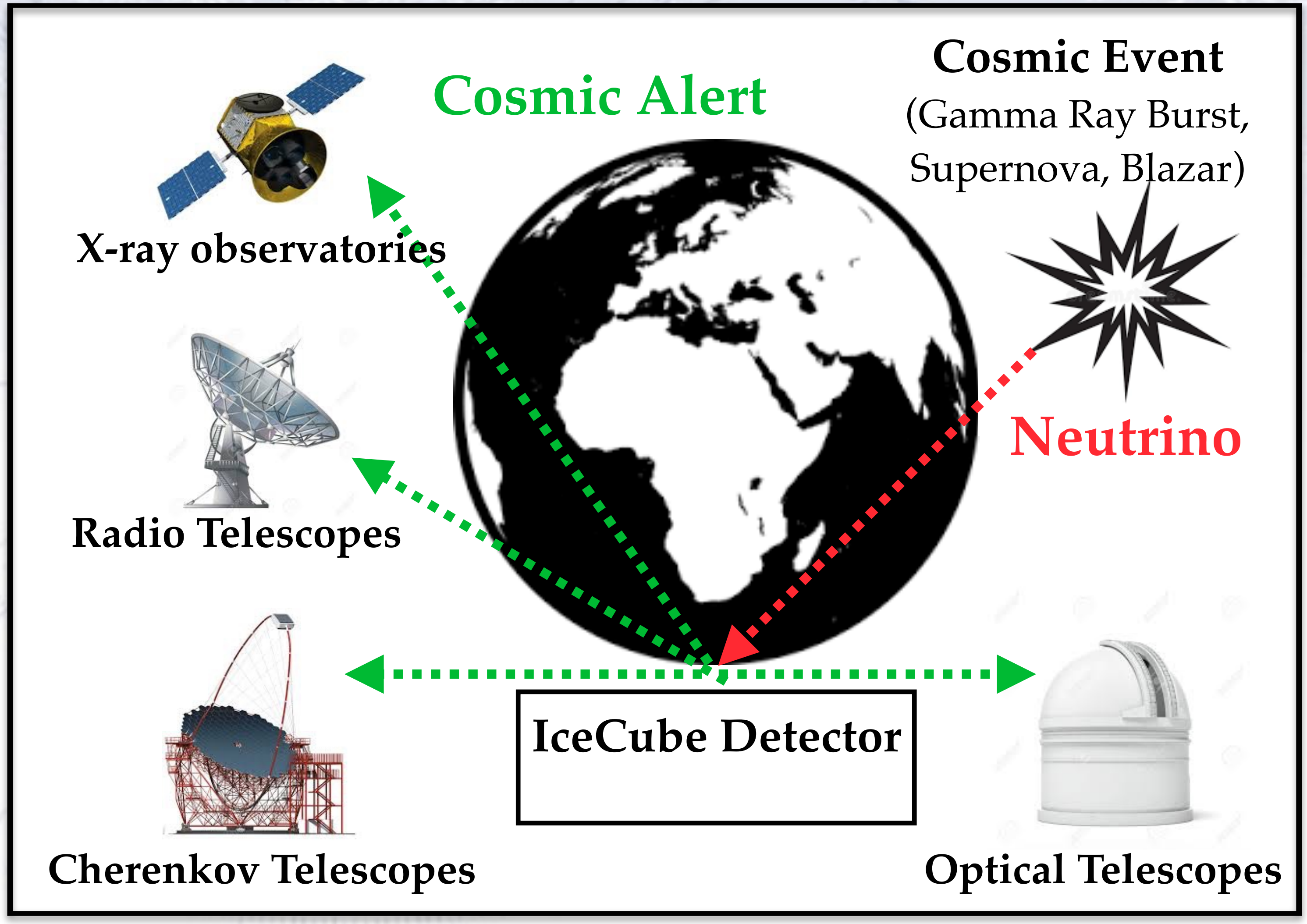
You could submit the job,
go to NOMA for dinner,
bring a friend/colleague,
get the full menu + wine menu,
have a fantastic evening…



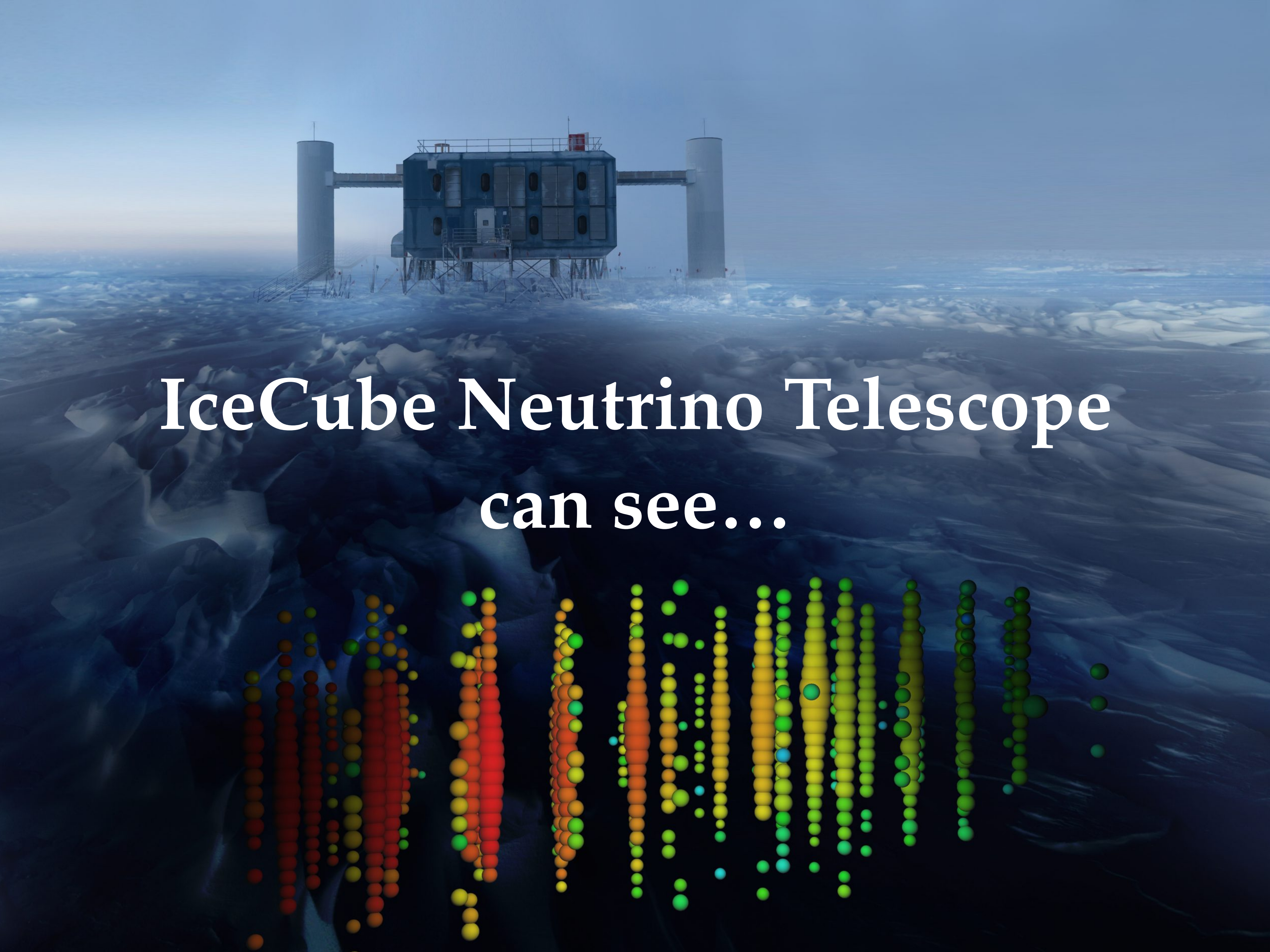
And you would still be richer and not be sober, when the job was done…
(7 hours, 10 DKr.)

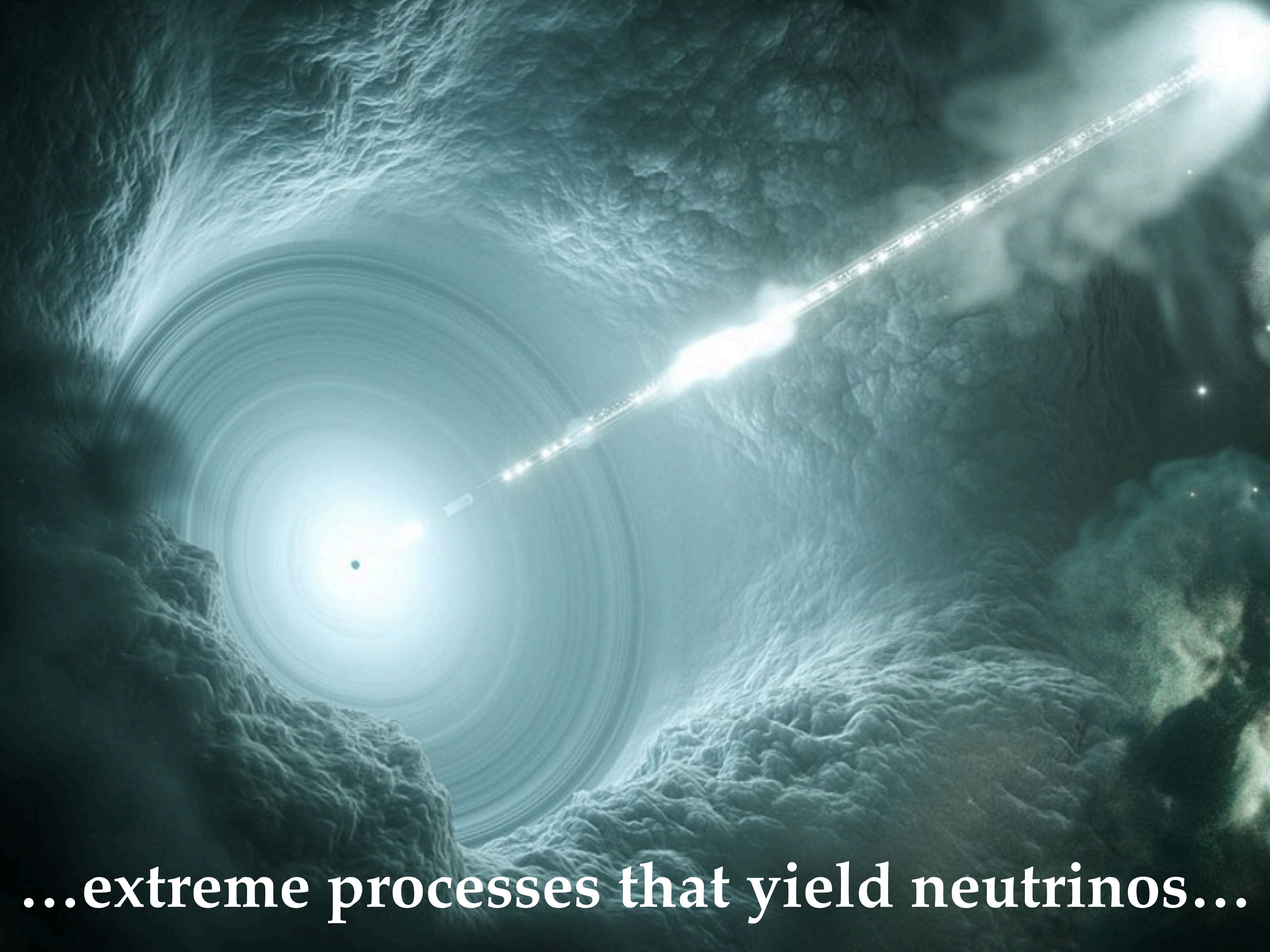
Dreaming

Seeing the Universe in ν light

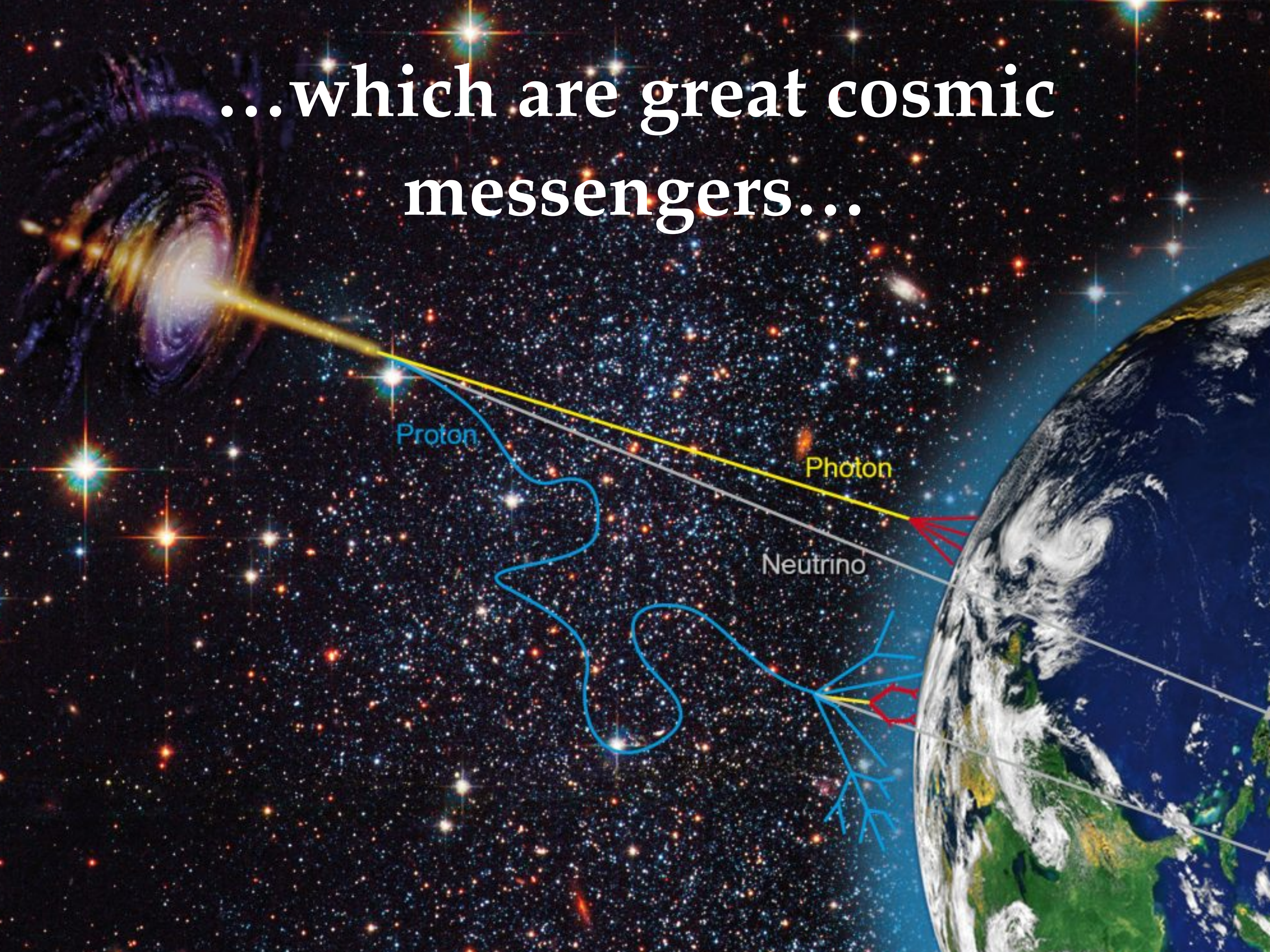


Seeing the Universe with neutrinos

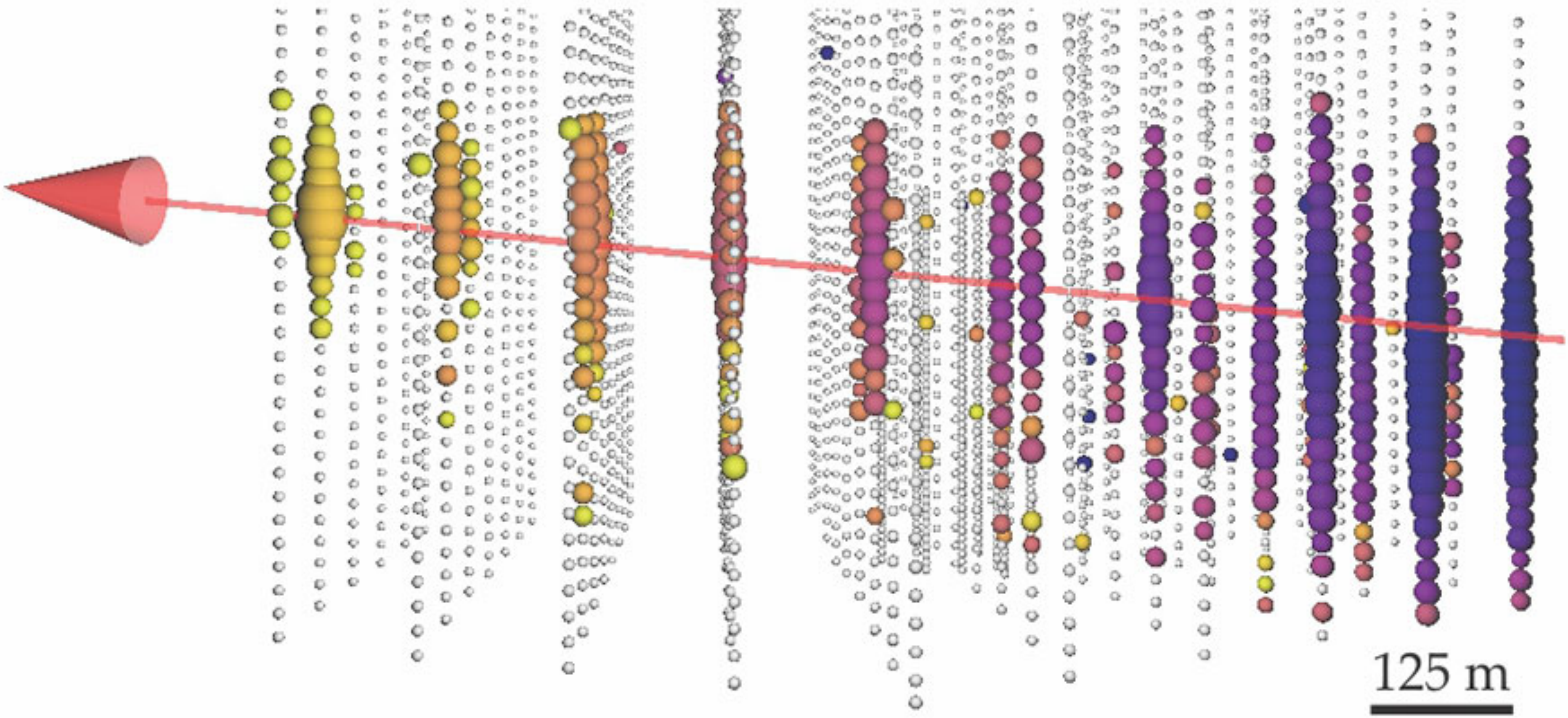

IceCube Neutrino Telescope
can see…

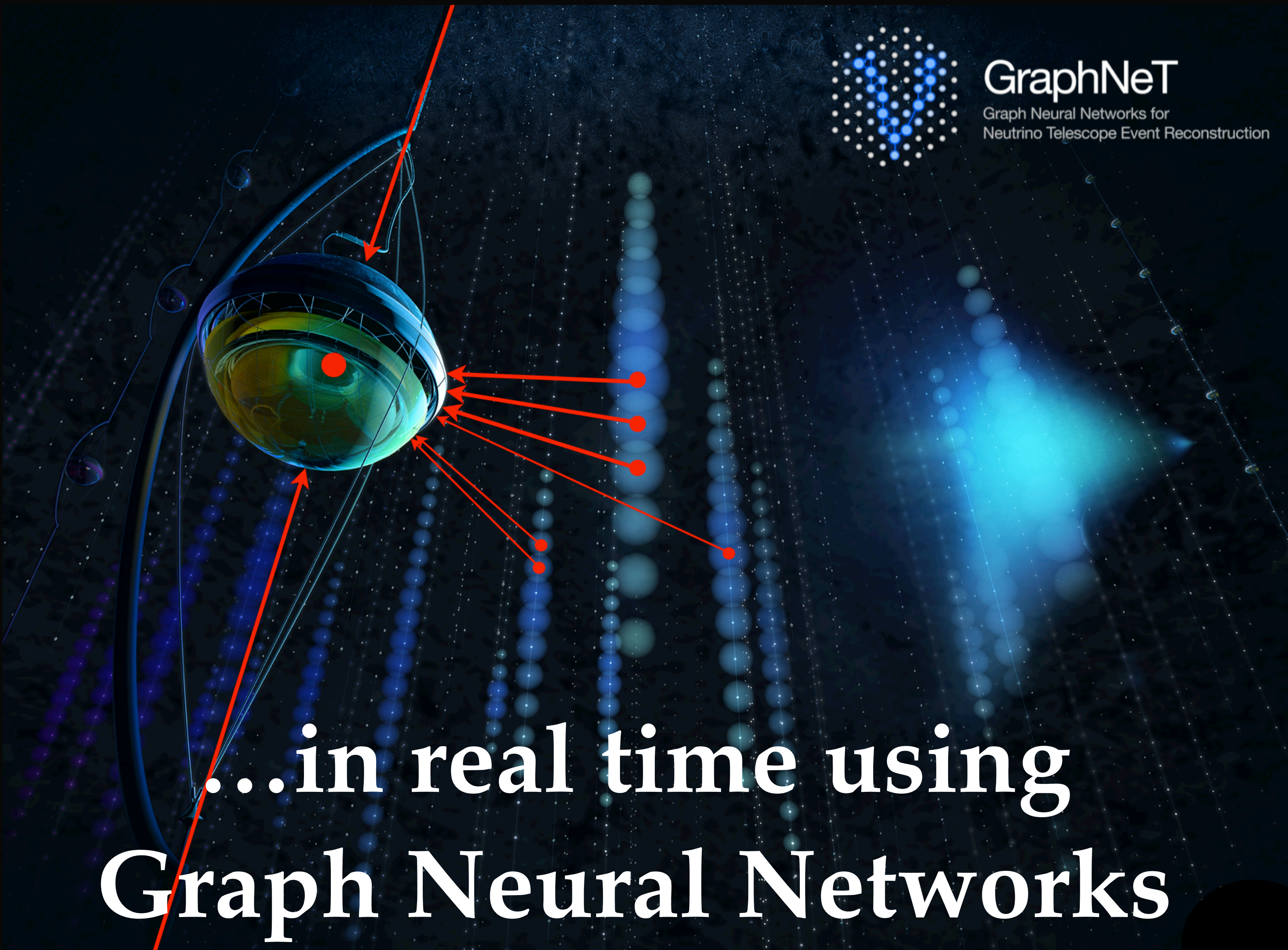

…extreme processes that yield neutrinos…

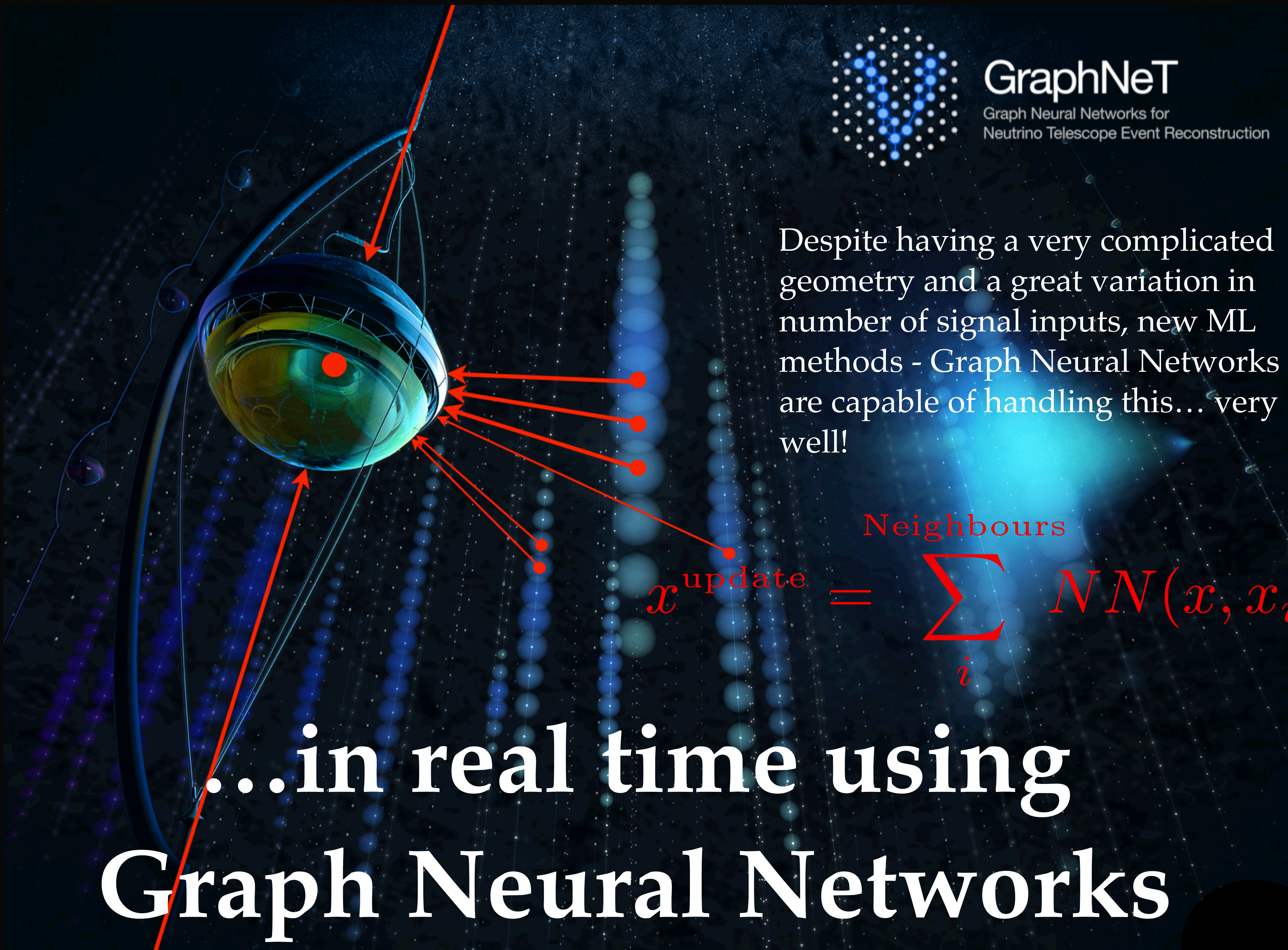
…which are great cosmic messengers…



…that can be observed by IceCube ν-Telescope…




GraphNeT
Graph Neural Networks for
Neutrino Telescope Event Reconstruction
…in real time using
Graph Neural Networks


GraphNeT
Graph Neural Networks for
Neutrino Telescope Event Reconstruction
Despite having a very complicated geometry and a great variation in number of signal inputs, new ML methods - Graph Neural Networks - are capable of handling this… very well!
$$x^{\text{update}} = \sum_{i}^{\text{Neighbours}} NN(x, x_i)$$
…in real time using
Graph Neural Networks

Bonus slides


SCIENCE
Formidlingspris 2015
Hard Rock