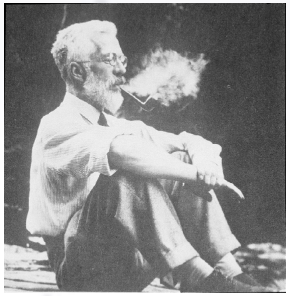
Applied ML

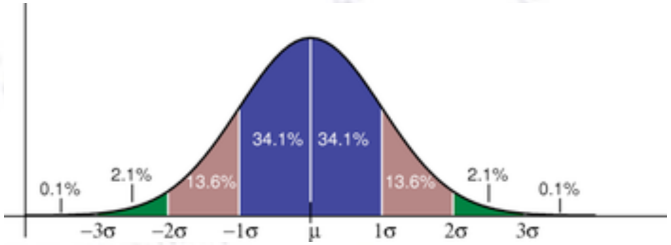
Graph Neural Networks








Troels C. Petersen (NBI)

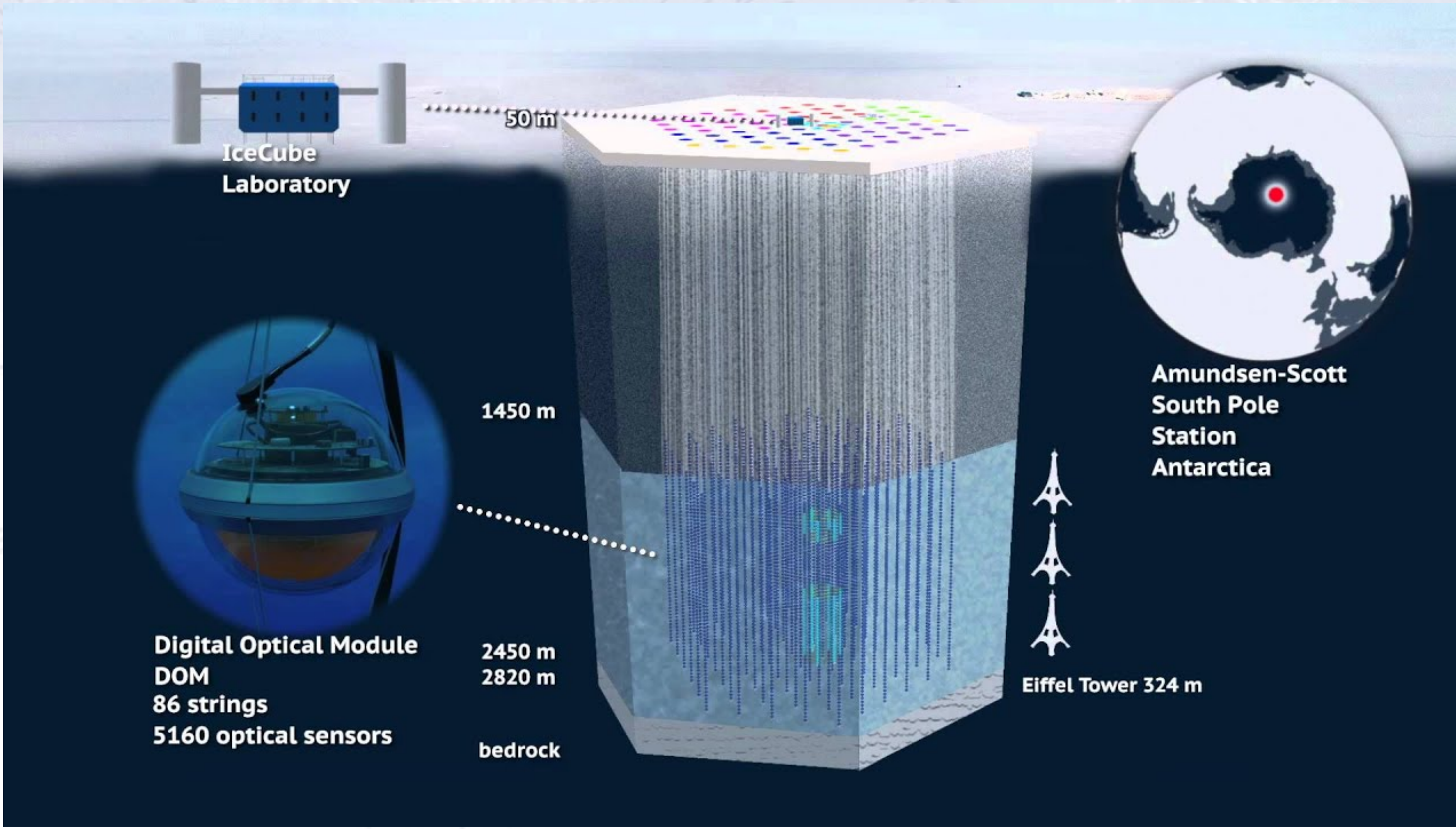


"Statistics is merely a quantisation of common sense - Machine Learning is a sharpening of it!"

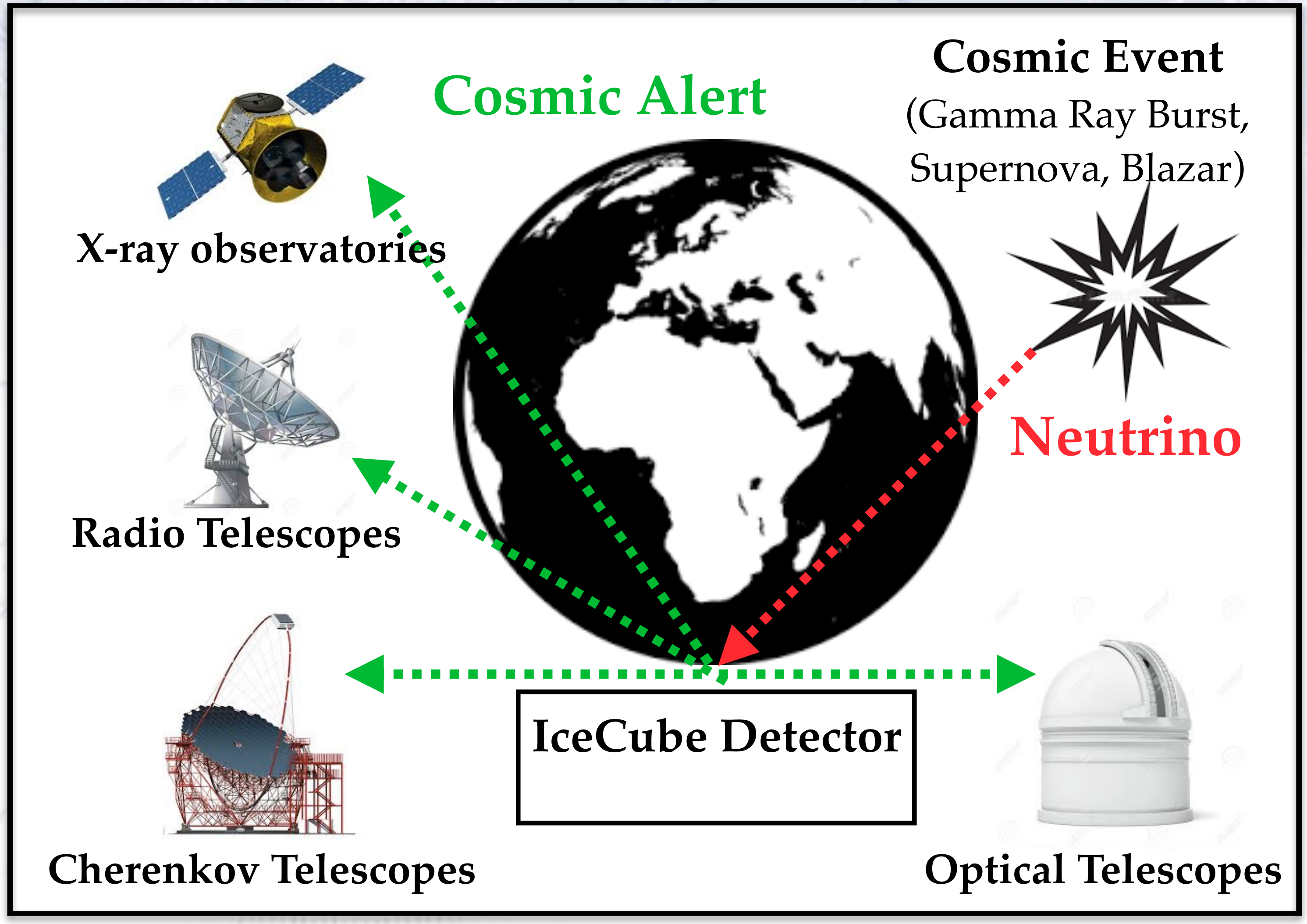
IceCube

IceCube counting house in the setting sun, sitting on top of 5160 Digital Optical Modules 1450-2450m below the surface.

IceCube experiment (South Pole)

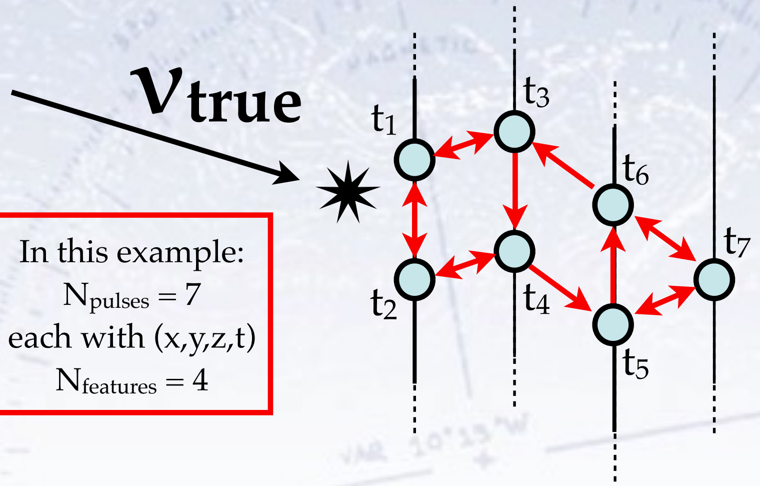


Seeing the Universe in ν light



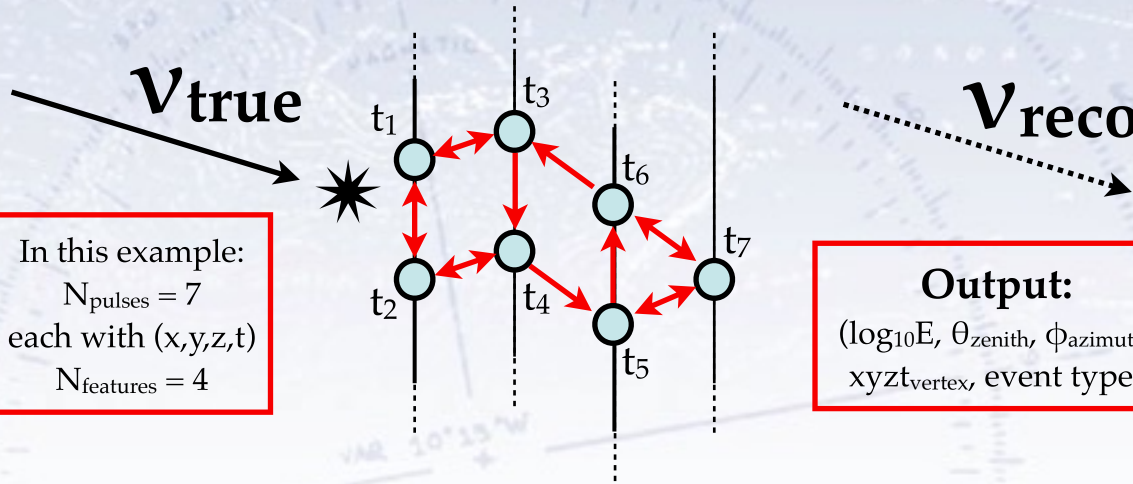
GNN model

Details of GNN reconstruction



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

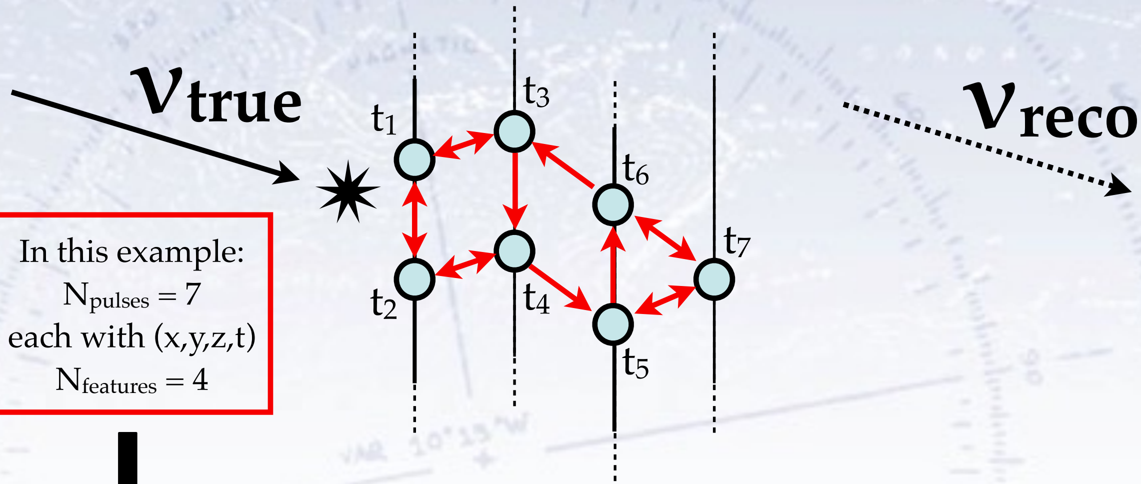
Details of GNN reconstruction



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

Output:
($\log_{10}E$, θ_{zenith}, $\phi_{azimuth}$, $xyzt_{vertex}$, event type)

Details of GNN reconstruction



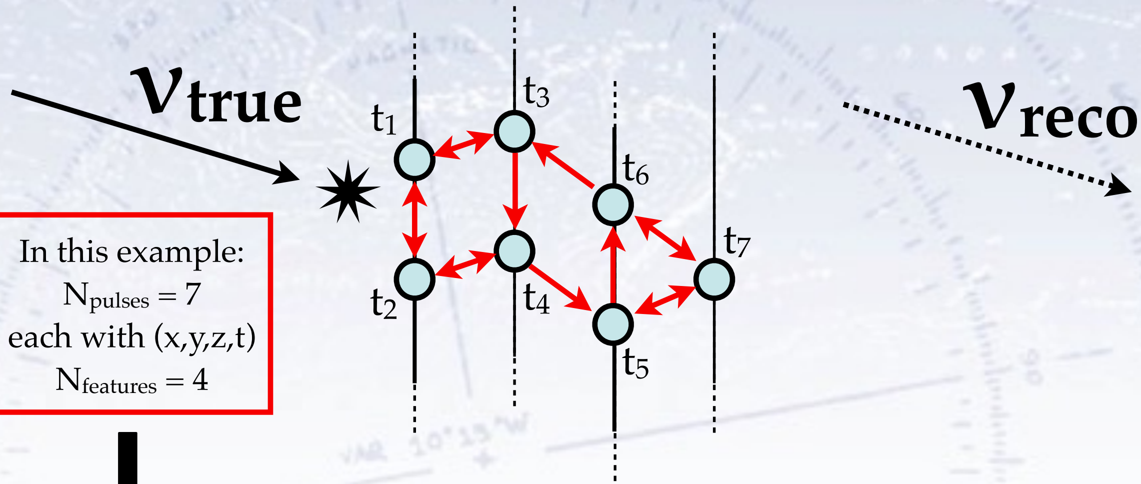
In this example:
$N_{pulses} = 7$
each with (x,y,z,t)
$N_{features} = 4$

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

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



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

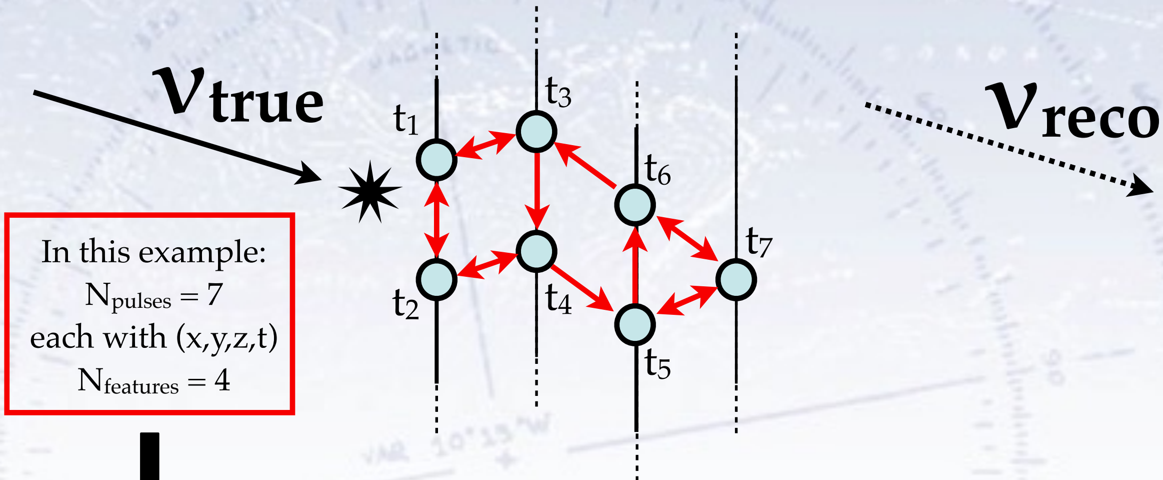
$$
\begin{array}{lcl}
\vec{v}_1 = [x_1\, y_1\, z_1\, t_1] & \xrightarrow{EC(\vec{v}_1,\vec{v}_2,\vec{v}_3)} & [g_{11} \dots g_{1N_1}] \\
\vec{v}_2 = [x_2\, y_2\, z_2\, t_2] & & [g_{21} \dots g_{2N_1}] \\
\vdots & \xrightarrow{EC(\vec{v}_4,\vec{v}_5,\vec{v}_6)} & \vdots \\
\vec{v}_7 = [x_7\, y_7\, z_7\,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

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



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

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

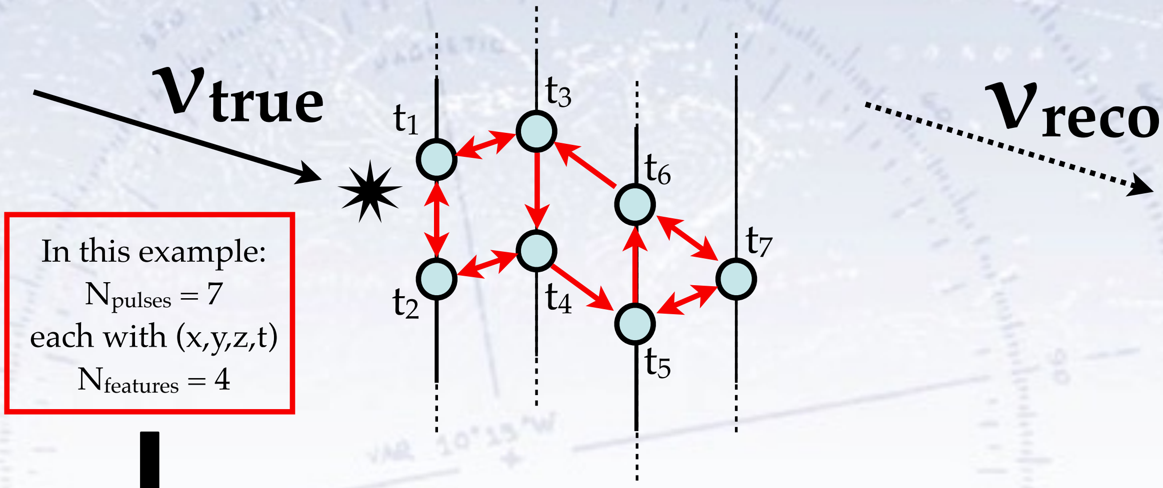
Input:
$N = N_{pulses} \times N_{features}$

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



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

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

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

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

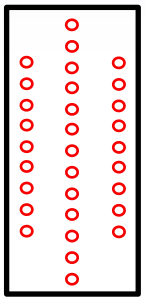
$N_{all} = N_{features} + N_1$

$$[x_1\, y_1\, z_1\, t_1\;\, g_{11} \ldots g_{1N_1}] \rightarrow \text{MLP1} \rightarrow [h_{11} \ldots h_{1N_{all}}]$$

$$[x_2\, y_2\, z_2\, t_2\;\, g_{21} \ldots g_{2N_1}] \rightarrow \text{MLP1} \rightarrow [h_{21} \ldots h_{2N_{all}}]$$

$$\vdots \qquad \vdots$$

$$[x_7\, y_7\, z_7\, t_7\;\, g_{71} \ldots g_{7N_1}] \rightarrow \text{MLP1} \rightarrow [h_{71} \ldots h_{7N_{all}}]$$



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

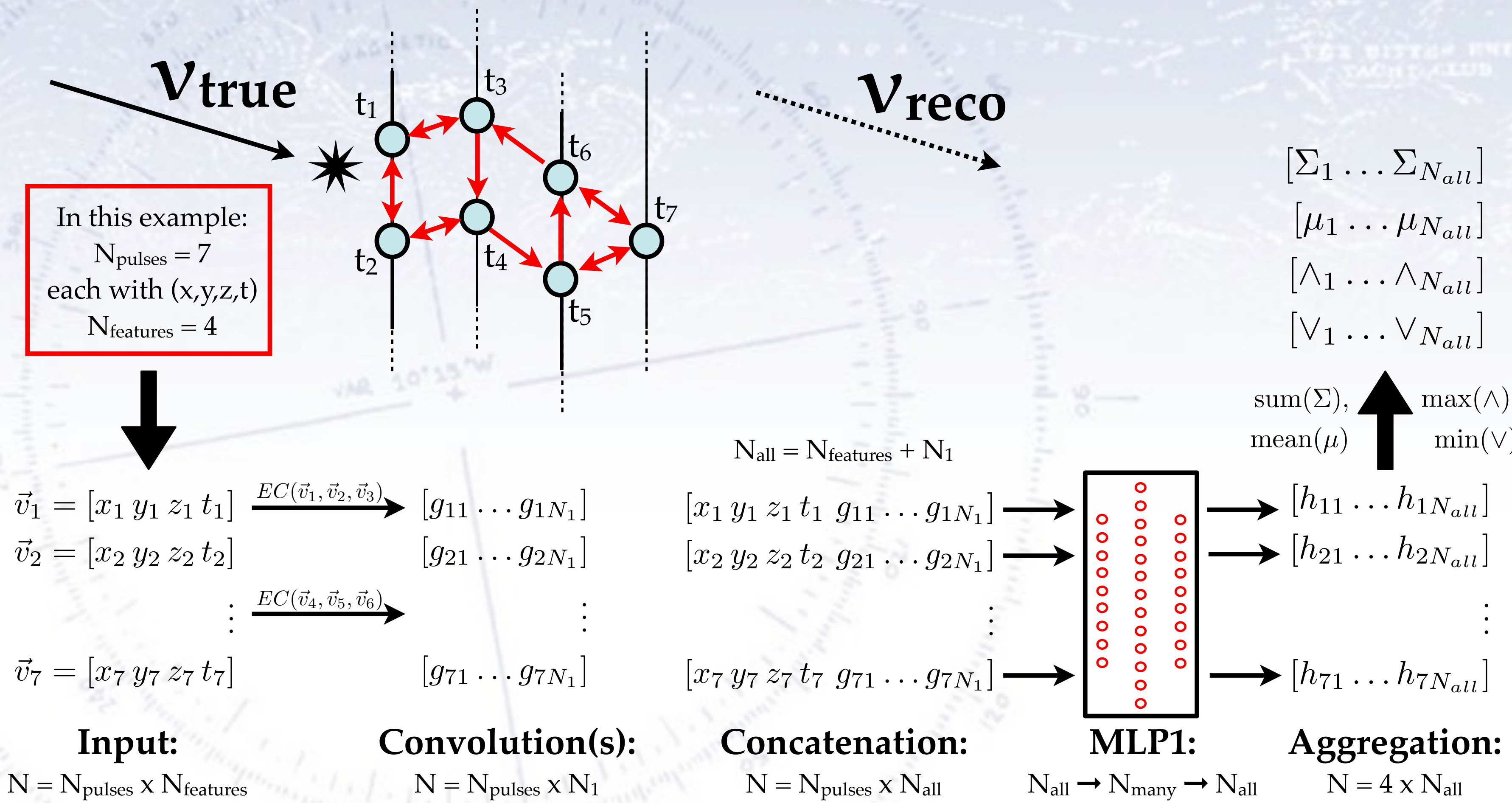
Convolution(s): $N = N_{pulses} \times N_1$

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

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

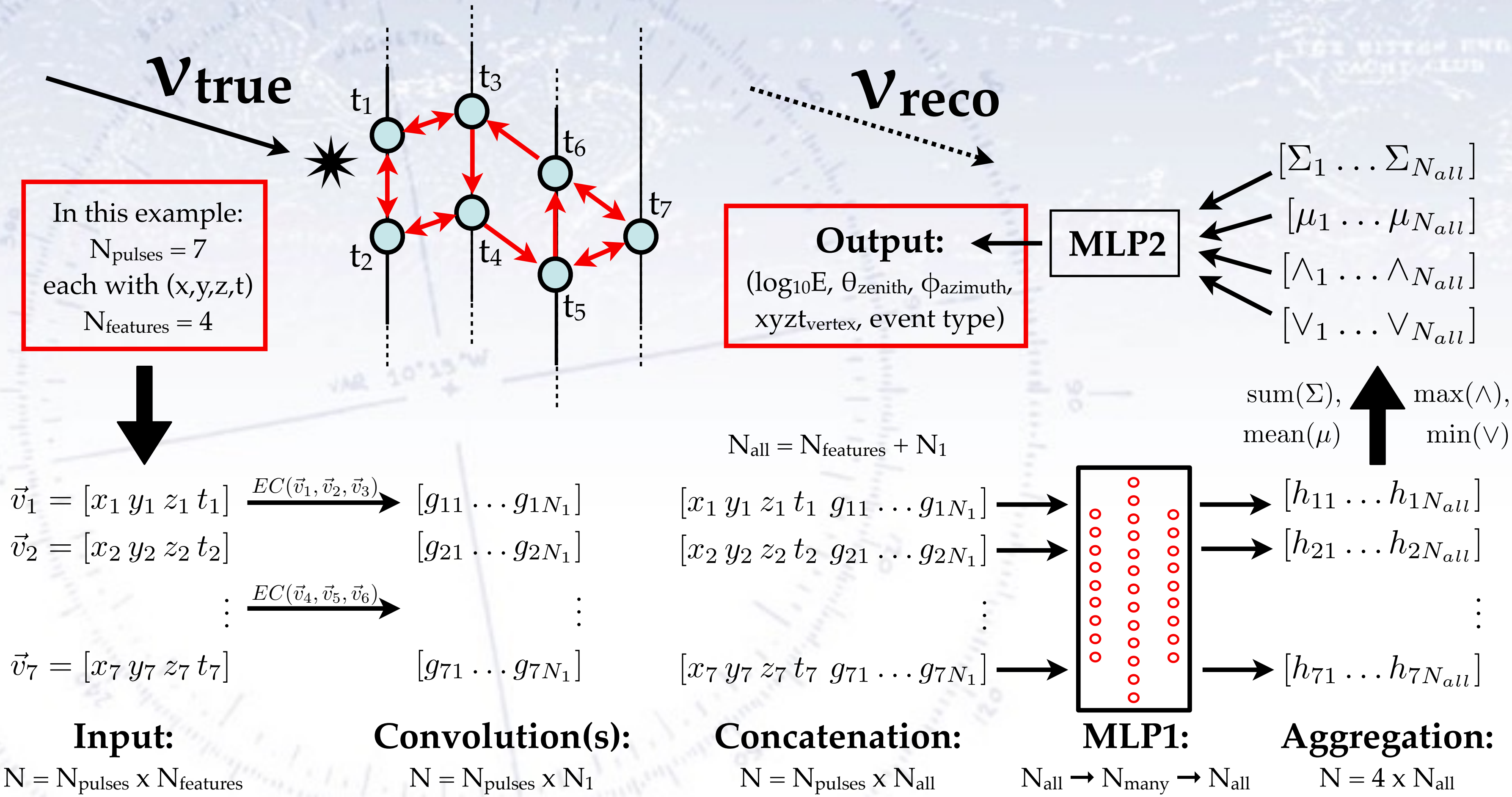
The input features of a node are combined with that of N (=2) nearby nodes through an NN (MLP0) function, yielding an (abstract) vector for each node. This can be repeated (not shown). All the features are then combined (concatenated) into long vectors, which are again put through an NN (MLP1) function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

Details of GNN reconstruction



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:

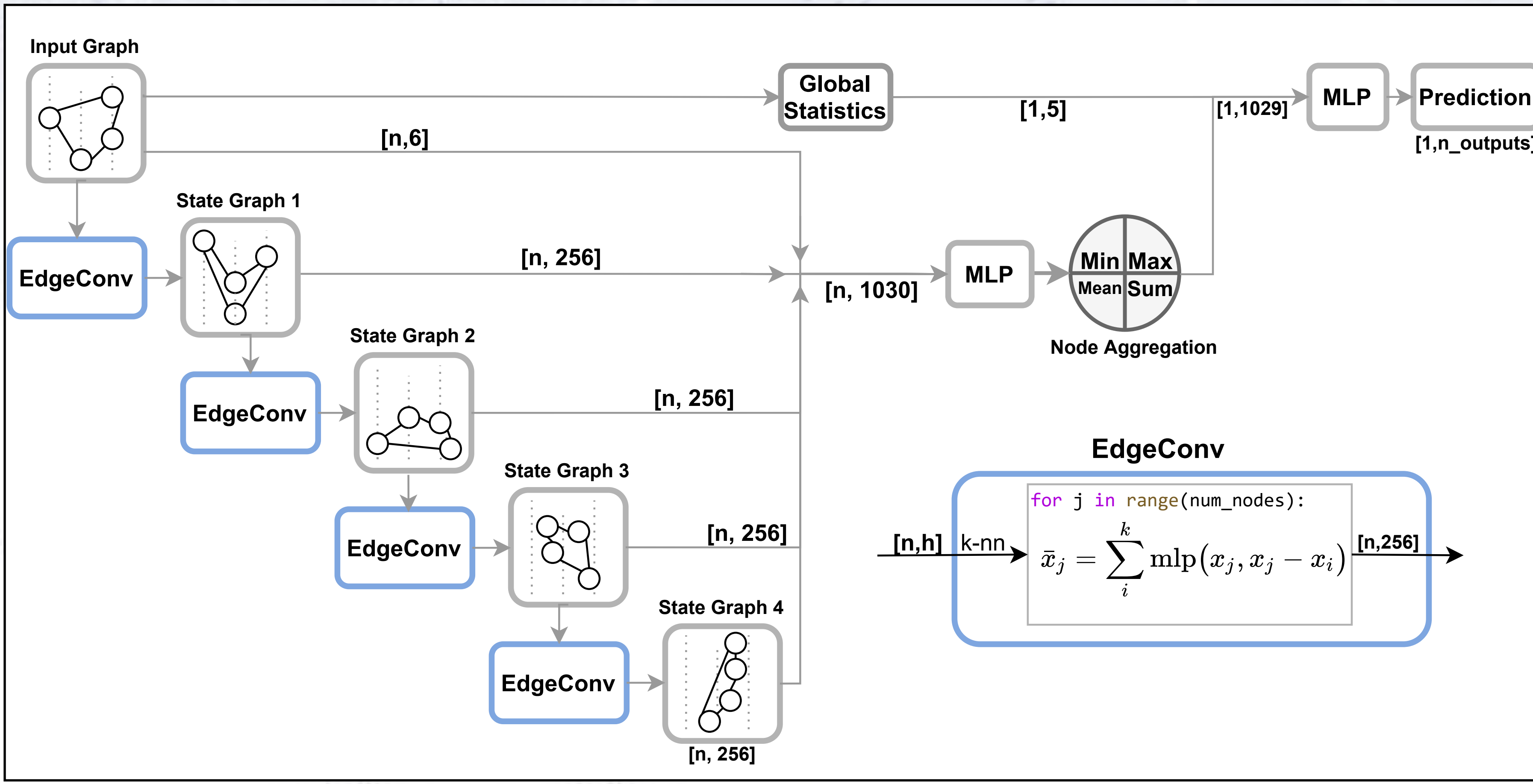
- We use 6 input features: x, y, z, t, charge, and Quantum Efficiency.
- We convolute each node with the nearest 8 nodes (not two).
- We do 4 (not 1) convolutions, each with 192 inputs and outputs.
- The concatenation is of all convolution layers and the original input.
- In the results to be shown, 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.

GraphNet

The GNN model is outlined more simply below, which is also the (current) figure for the GNN paper in process.



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/icecube/graphnet/

We are writing our results up in an IceCube paper (responded to several rounds of feedback and comments).

If possible, we also want to make it into a Kaggle competition (first thoughts).

GraphNet people

The original idea came through discussions with Jason Koskinen (NBI), where the “reconstruction bottleneck” became apparent.
With the arrival of Graph Neural Network, **Andreas** and I made a Marie-Curie Fellowship application… which took a while! Meanwhile, I had first Mads and Bjørn, and later **Rasmus** as master students working on the project.

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!

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.

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

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

Andreas Soegaard
Project part: Eventually, probably all parts
Period: September 2021 (Marie-Curie Fellow).
Email: andreas.soegaard@nbi.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.

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.

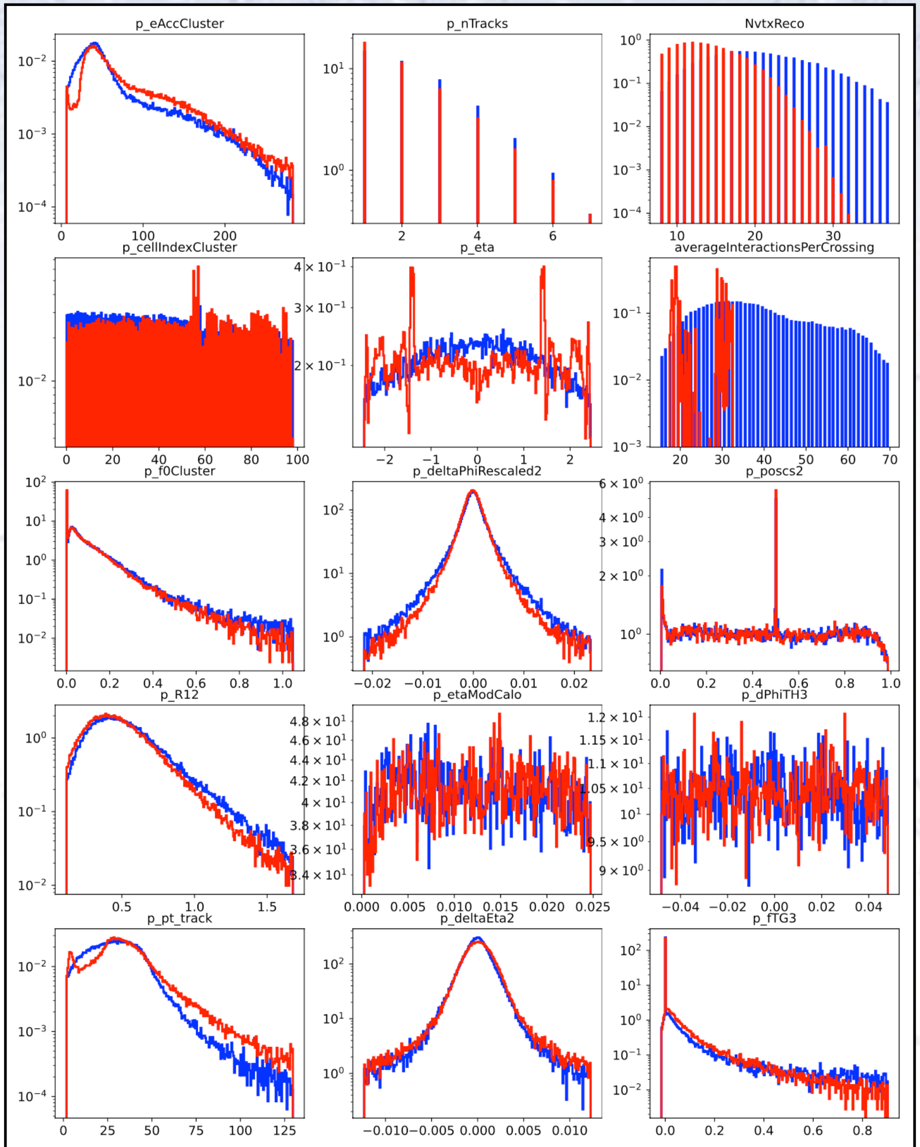
Bjoern Moelvig
Project part: RNN/GRU reconstruction, loss function exploration
Period: September 2019 - October 2020.
Results: Master Thesis, Thesis Defence, and GitHub repository.

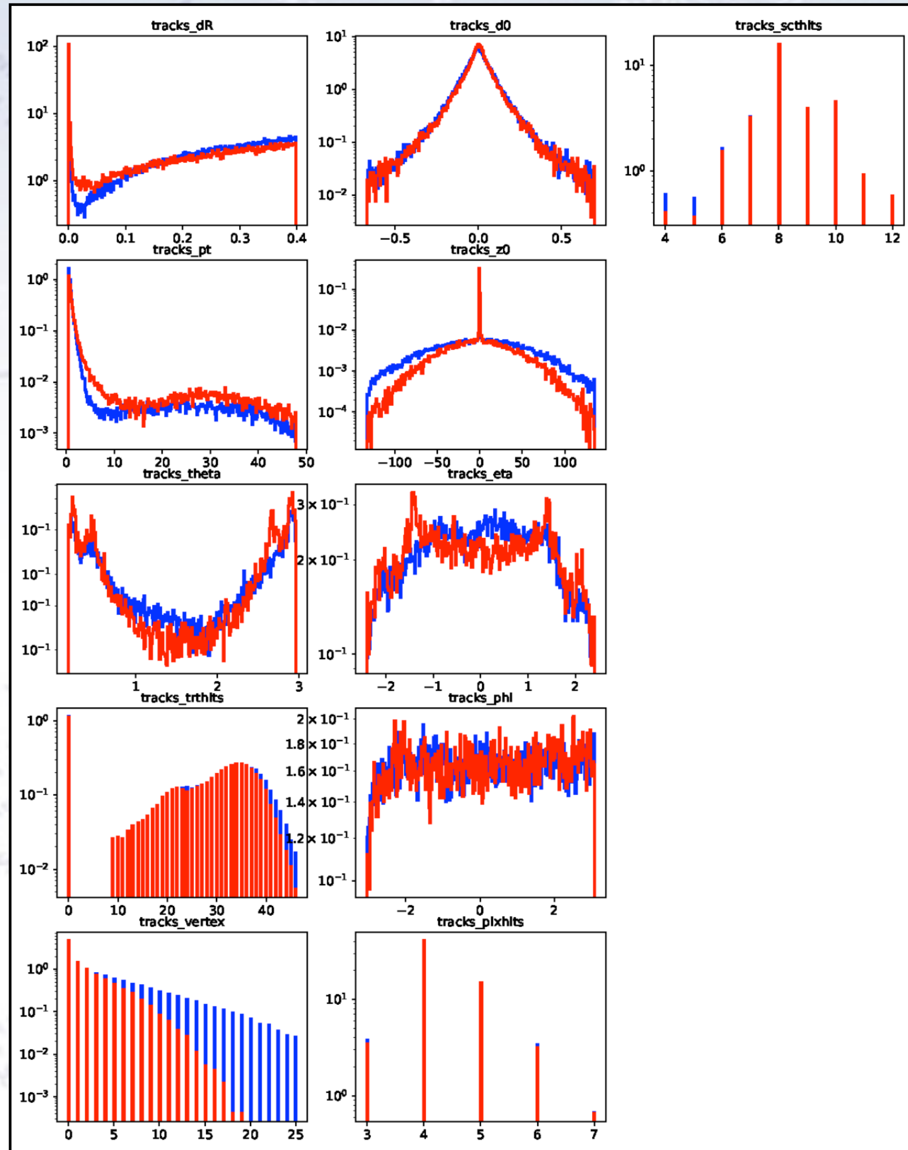
Philipp Eller

Martin Minh

Bonus slides

The input variables






SCIENCE
Formidlingspris 2015
Hard Rock


GraphNet Workshop
Niels Bohr Institute
5th - 7th of May 2022