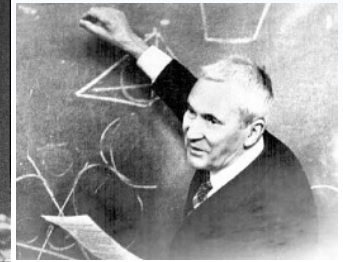
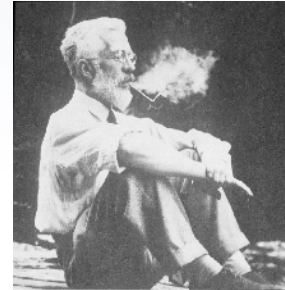
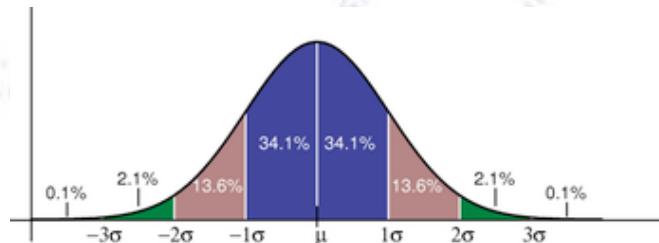


# Applied ML

## Graph Neural Networks (GNNs)



Troels C. Petersen (NBI)

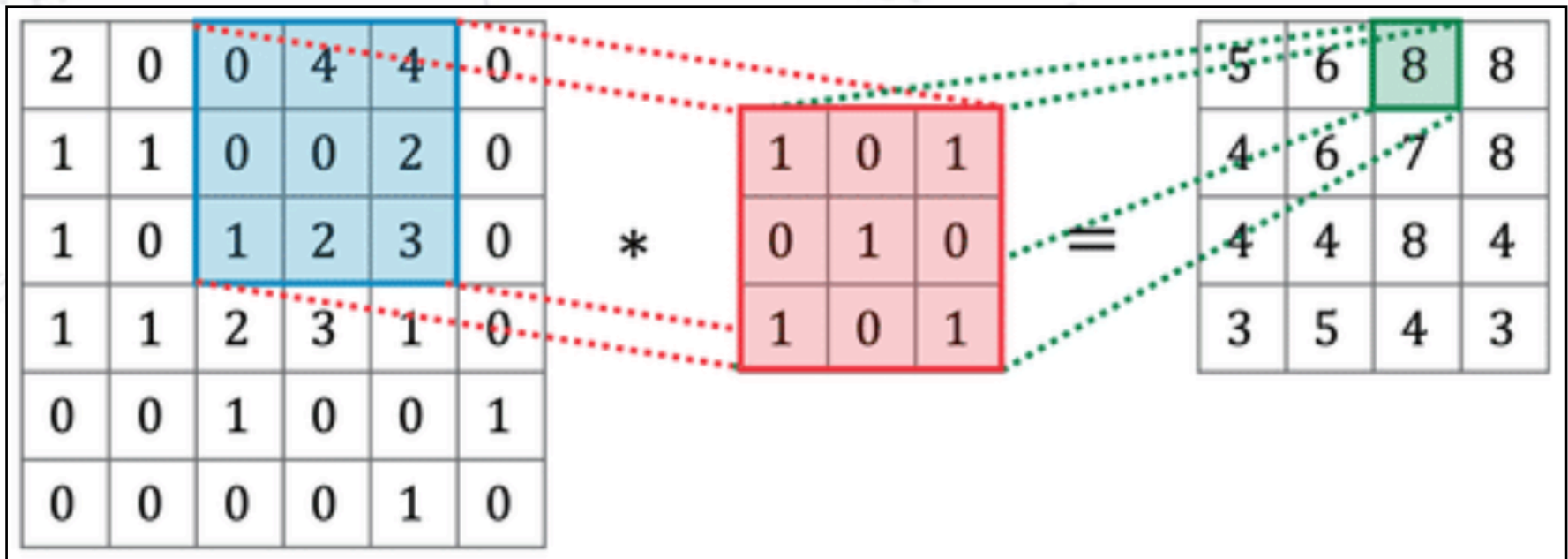


*"Statistics is merely a quantisation of common sense - Machine Learning is a sharpening of it!"*

# Motivation for GNNs

Let us consider images in a more abstract sense:

- They consist of (typically 3-4, RGB or CMYK) numbers in a **matrix structure**.
- The distance between neighbouring cells is **constant**.



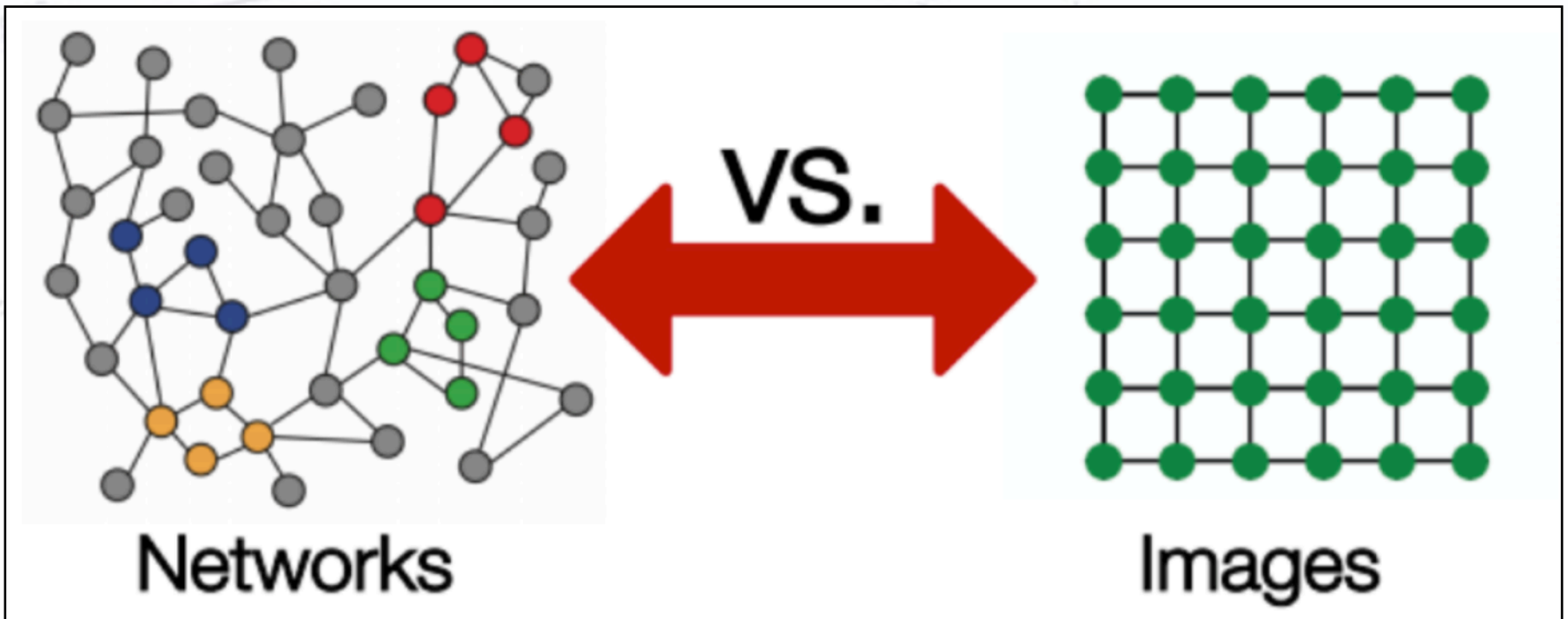
# Motivation for GNNs

Let us consider images in a more abstract sense:

- They consist of (typically 3-4, RGB or CMYK) numbers in a **matrix structure**.
- The distance between neighbouring cells is **constant**.

What if the data was not an image, but we wanted to use a CNN anyway?

This problem is not uncommon... (<https://arxiv.org/pdf/2101.11589.pdf>)



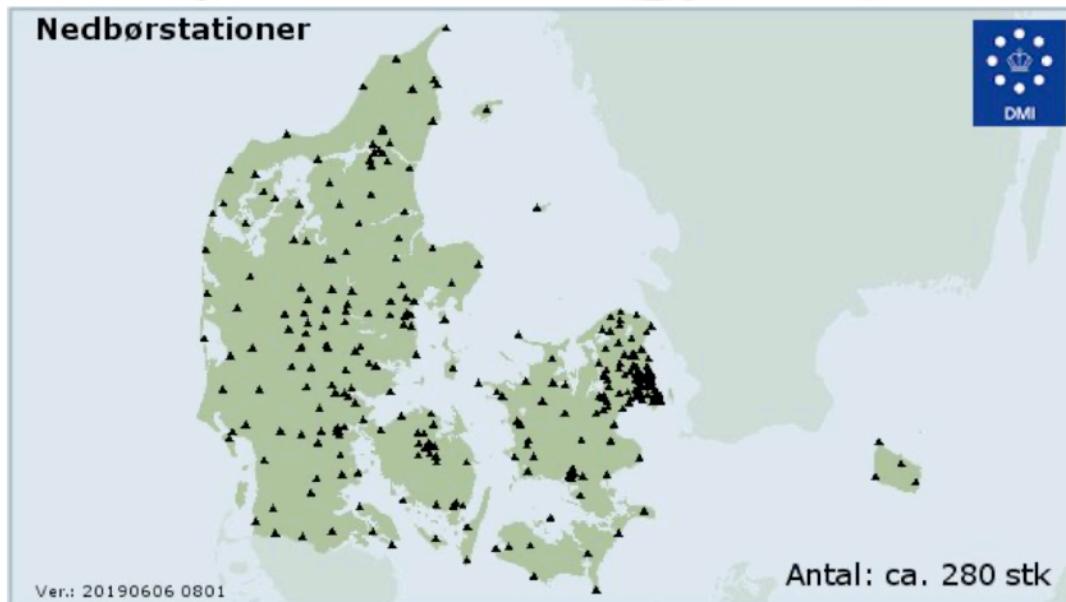


# Example of non-CNN data

Take the example of weather stations:

- There are about 280 weather stations distributed **non-uniformly** throughout Denmark.
- Each station provides say:  
[temperature, wind speed, wind direction, humidity, latitude, longitude]

What would an “image” of this data look like? And would a CNN work on it?





# Example of non-CNN data

Take the example of weather stations:

- There are about 280 weather stations distributed **non-uniformly** throughout Denmark.
- Each station provides say:  
[temperature, wind speed, wind direction, humidity, latitude, longitude]

What would an “image” of this data look like? And would a CNN work on it?

Nedbørstationer

By forcing problems with irregular geometry into images, we're shaping the problem to the tool, and not the tool to the problem!

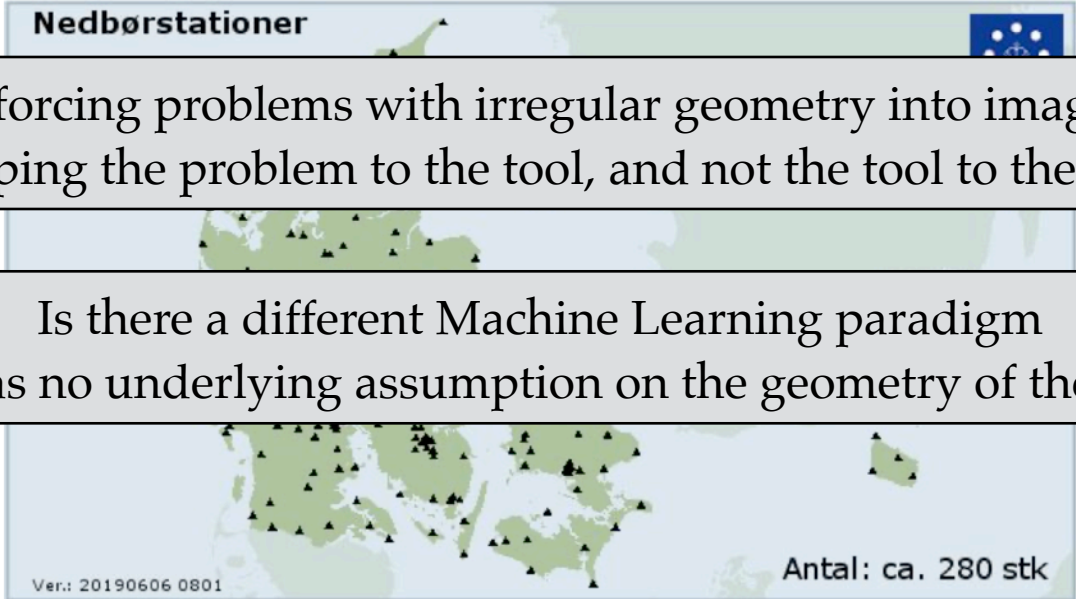


# Example of non-CNN data

Take the example of weather stations:

- There are about 280 weather stations distributed **non-uniformly** throughout Denmark.
- Each station provides say:  
[temperature, wind speed, wind direction, humidity, latitude, longitude]

What would an “image” of this data look like? And would a CNN work on it?



Nedbørstationer

A map of Denmark with numerous black triangle markers indicating the locations of precipitation stations. The map is titled 'Nedbørstationer' and includes a small European Union flag in the top right corner. The background is a light blue map of the country.

By forcing problems with irregular geometry into images, we're shaping the problem to the tool, and not the tool to the problem!

Is there a different Machine Learning paradigm that has no underlying assumption on the geometry of the data?

Ver.: 20190606 0801

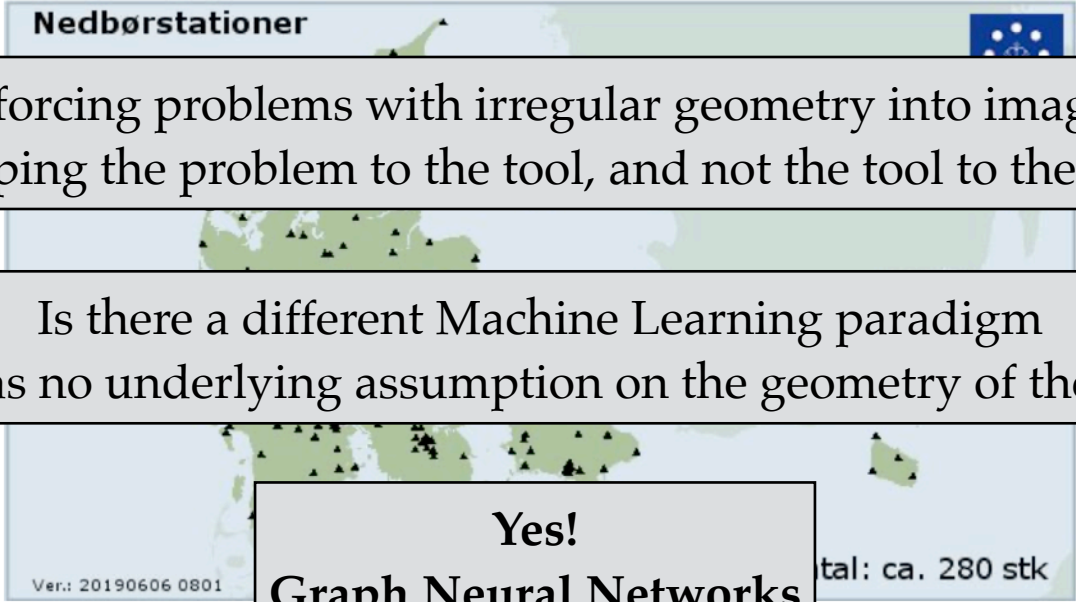
Antal: ca. 280 stk

# Example of non-CNN data

Take the example of weather stations:

- There are about 280 weather stations distributed **non-uniformly** throughout Denmark.
- Each station provides say:  
[temperature, wind speed, wind direction, humidity, latitude, longitude]

What would an “image” of this data look like? And would a CNN work on it?



Nedbørstationer

By forcing problems with irregular geometry into images, we're shaping the problem to the tool, and not the tool to the problem!

Is there a different Machine Learning paradigm that has no underlying assumption on the geometry of the data?

Yes!

Graph Neural Networks

tal: ca. 280 stk

Ver.: 20190606 0801



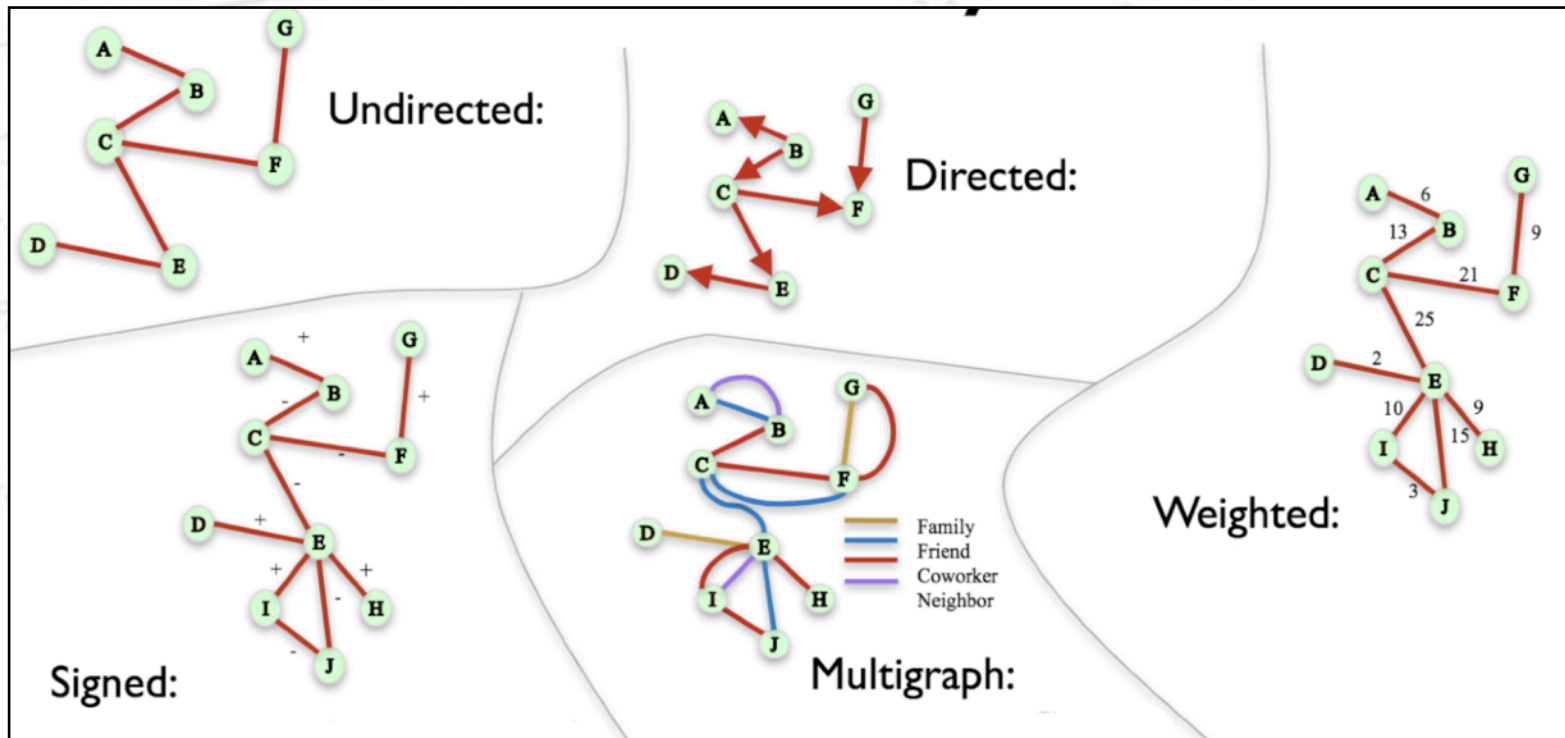
# Graph Definition

A **graph**  $G$  is defined as a combined pair  $G = \{V, E\}$  consisting of:

- **Nodes:** A set  $V$ , that typically contain input features (also called vertices)
- **Edges:** A set  $E$  of pair nodes, thus connecting the nodes (also called links)

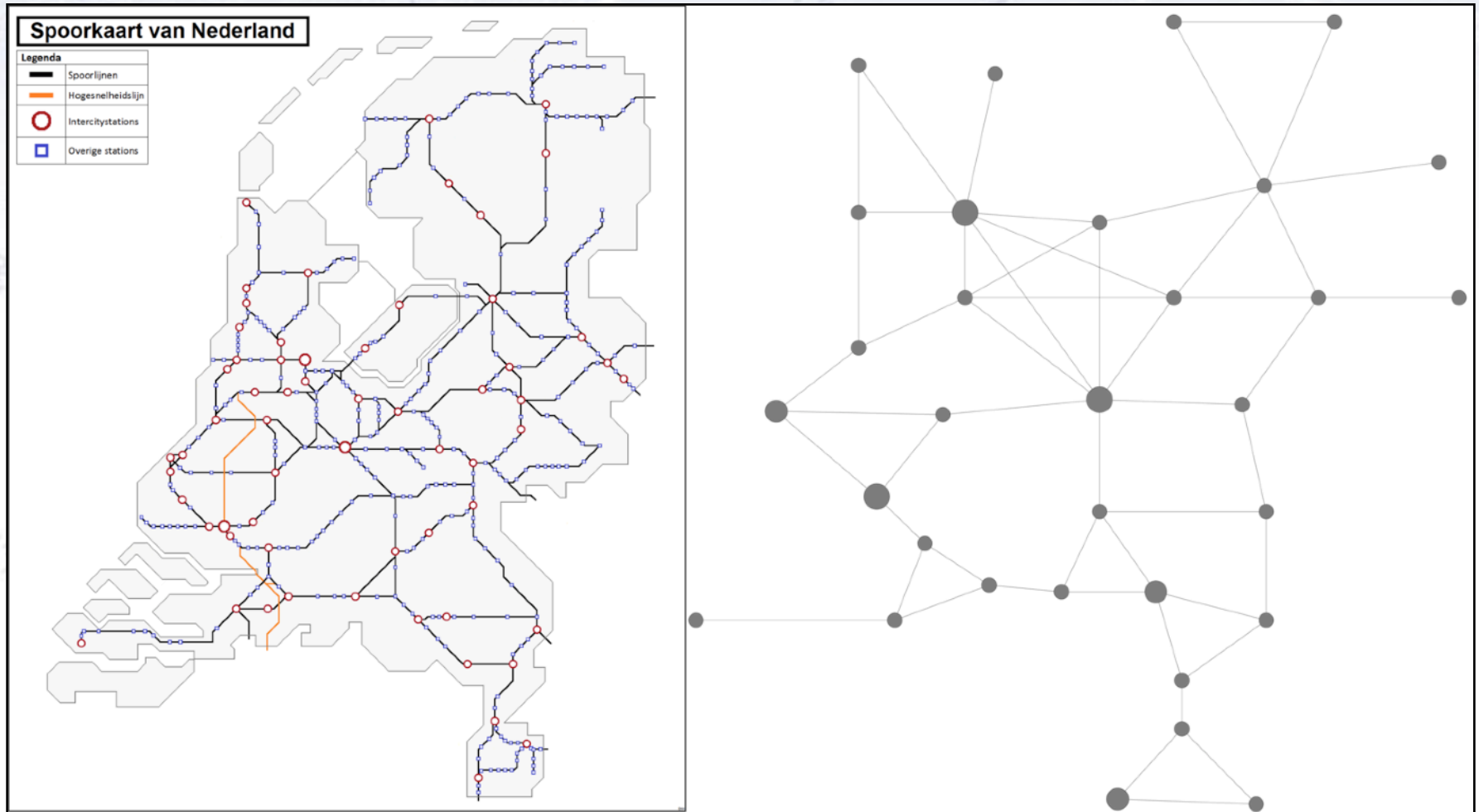
In plain words:

You got a list of points (nodes) that are somehow connected (with edges).



# Dutch railways as a graph

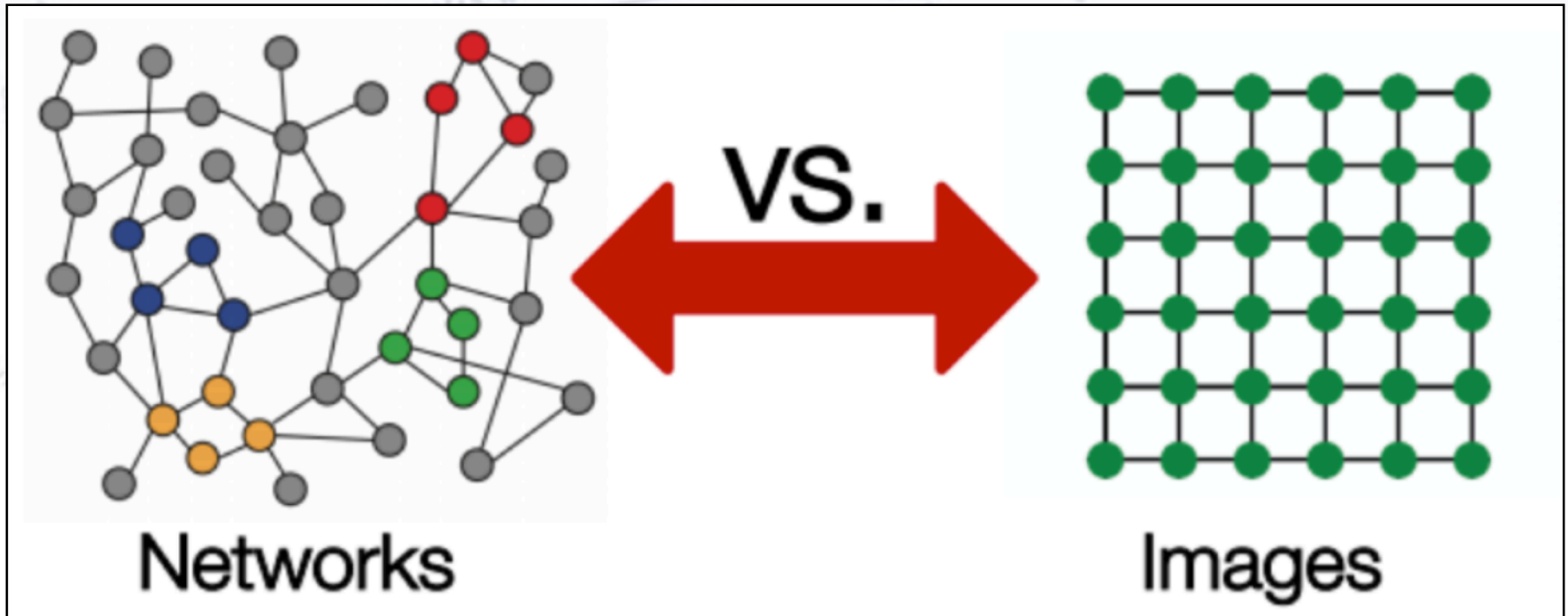
It takes a little imagination to see things as graphs... but once you do, everything starts looking like graphs!



# Geometrical data

Unlike e.g. images, **graphs have no underlying assumption on the geometry of the data**. This structure has to be specified **by the user** using the edges.

Many of the techniques in Machine Learning that you have been introduced to are also available for graphs (convolution, LSTM, Attention, Auto-Encoder, etc.)





# Graph Convolution

The **graph convolution** proceeds much like for CNNs/images, as the **output is another graph**, possibly of different dimensionality. There are many different types of convolutions, **edgeconv** (<https://arxiv.org/abs/1801.07829>) considered below:

$$\tilde{x}_j = \sum_{i=1}^n f(x_j, x_j - x_i)$$

The “tilde” denotes the updated node

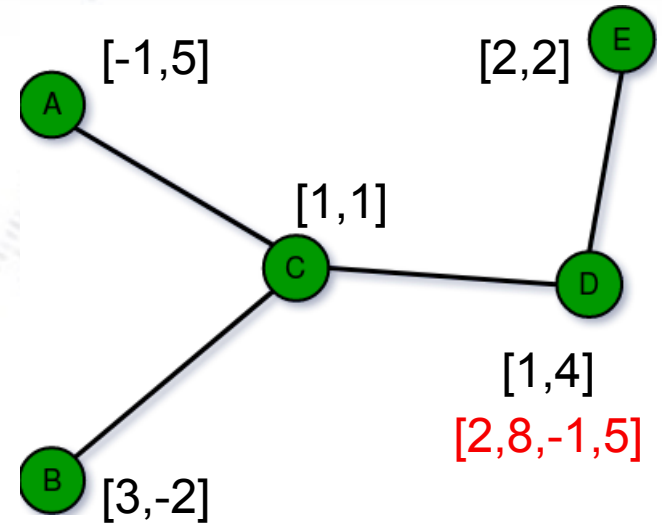
# Graph Convolution

The **graph convolution** proceeds much like for CNNs/images, as the **output is another graph**, possibly of different dimensionality. There are many different types of convolutions, **edgeconv** (<https://arxiv.org/abs/1801.07829>) considered below:

$$\tilde{x}_j = \sum_{i=1}^n f(x_j, x_j - x_i)$$

The “tilde” denotes the updated node, and so if we applied the edgeconv operator with  $f(\mathbf{x}) = \mathbf{1} * \mathbf{x} + \mathbf{0}$  to node D, we would obtain:

$$\begin{aligned}\tilde{x}_D &= f(x_D, x_D - x_C) + f(x_D, x_D - x_E) \\ &= f([1, 4], [1, 4] - [1, 1]) + f([1, 4], [1, 4] - [2, 2]) \\ &= f([1, 4], [0, 3]) + f([1, 4], [-1, 2]) \\ &= f([1, 4, 0, 3]) + f([1, 4, -1, 2]) \quad (\text{by concatenation}) \\ &= 1 \cdot [1, 4, 0, 3] + 1 \cdot [1, 4, -1, 2] \\ &= [2, 8, -1, 5]\end{aligned}$$



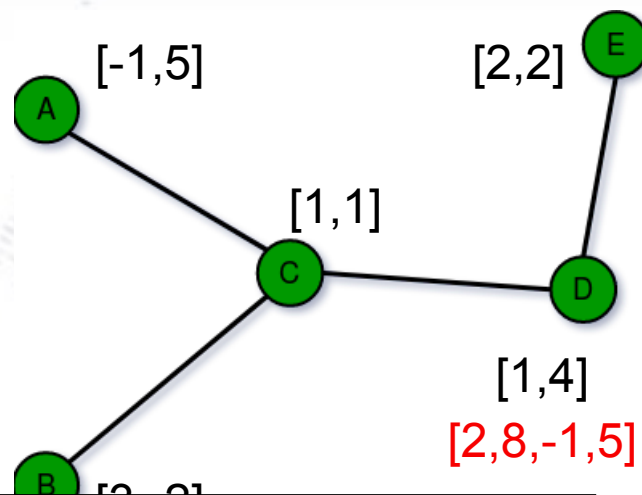
# Graph Convolution

The **graph convolution** proceeds much like for CNNs/images, as the **output is another graph**, possibly of different dimensionality. There are many different types of convolutions, **edgeconv** (<https://arxiv.org/abs/1801.07829>) considered below:

$$\tilde{x}_j = \sum_{i=1}^n f(x_j, x_j - x_i)$$

The “tilde” denotes the updated node, and so if we applied the edgeconv operator with  $f(\mathbf{x}) = \mathbf{1} * \mathbf{x} + \mathbf{0}$  to node D, we would obtain:

$$\begin{aligned}\tilde{x}_D &= f(x_D, x_D - x_C) + f(x_D, x_D - x_E) \\ &= f([1, 4], [1, 4] - [1, 1]) + f([1, 4], [1, 4] - [2, 2]) \\ &= f([1, 4], [0, 3]) + f([1, 4], [-1, 2]) \\ &= f([1, 4, 0, 3]) + f([1, 4, -1, 2]) \quad (\text{by concatenation})\end{aligned}$$



**However...**

**This only shows the start of a Graph Neural Network, not how to continue!**

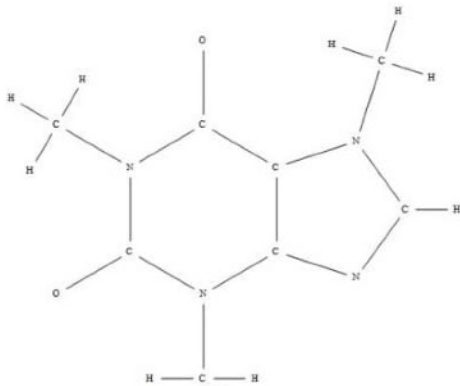




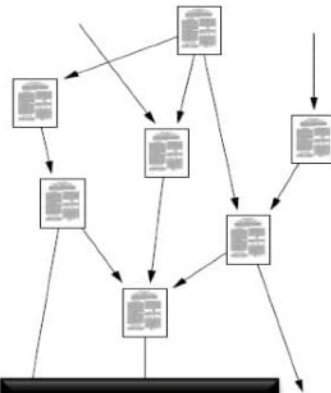
# Example of Application

# Current example usages

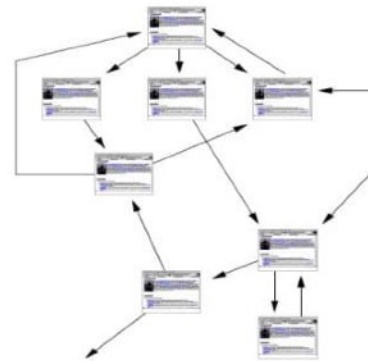
The graph examples/solutions are starting to enter the scene in many places:



Molecules



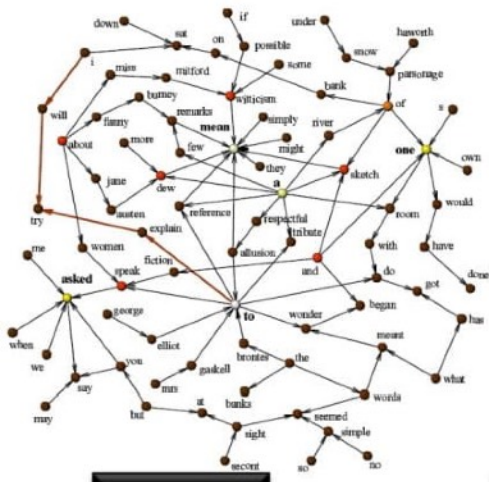
Knowledge



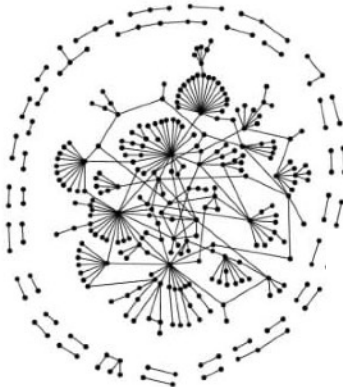
Information



Brain/neurons



Genes



Communication

```
def encode(obj):  
    """  
    Encode a (possibly nested)  
    dictionary containing complex values  
    into a form that can be serialized  
    using JSON.  
    """  
    e = {}  
    for key, value in obj.items():  
        if isinstance(value, dict):  
            e[key] = encode(value)  
        elif isinstance(value, complex):  
            e[key] = value.real + j*value.imag  
    return e  
  
import ast  
tree = ast.parse(" ")
```

Software



Social

# Classification from point clouds

Imagine a point cloud simply giving a (long) series of coordinates...

## Computer Vision (EdgeConv)

Dynamic Graph CNN for Learning on Point Clouds • 1:9

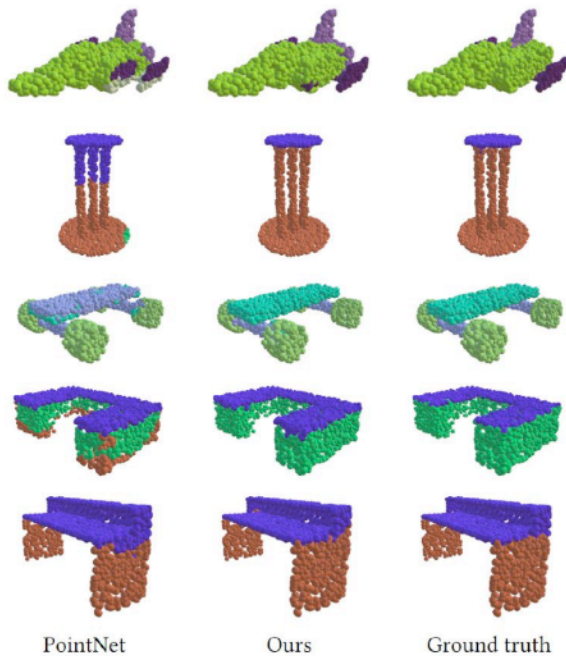
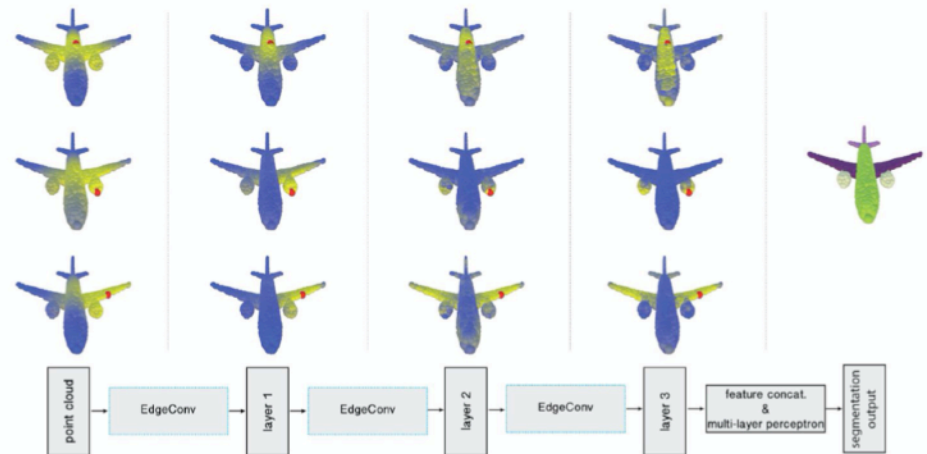


Fig. 7. Compare part segmentation results. For each set, from left to right: PointNet, ours and ground truth.

Dynamic Graph CNN for Learning on Point Clouds

YUE WANG, Massachusetts Institute of Technology  
YONGBIN SUN, Massachusetts Institute of Technology  
ZIWEI LIU, UC Berkeley / ICSI  
SANJAY E. SARMA, Massachusetts Institute of Technology  
MICHAEL M. BRONSTEIN, Imperial College London / USI Lugano  
JUSTIN M. SOLOMON, Massachusetts Institute of Technology



<https://arxiv.org/pdf/1801.07829.pdf>



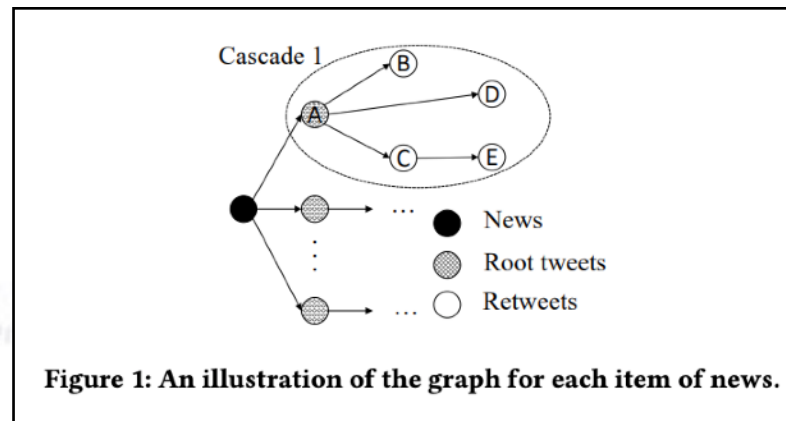
# Fake News detection

News stories can be related through graphs, where information is added.

## Graph Neural Networks with Continual Learning for Fake News Detection from Social Media

Yi Han, Shanika Karunasekera, Christopher Leckie

Although significant effort has been applied to fact-checking, the prevalence of fake news over social media, which has profound impact on justice, public trust and our society, remains a serious problem. In this work, we focus on propagation-based fake news detection, as recent studies have demonstrated that fake news and real news spread differently online. Specifically, considering the capability of graph neural networks (GNNs) in dealing with non-Euclidean data, we use GNNs to differentiate between the propagation patterns of fake and real news on social media. In particular, we concentrate on two questions: (1) Without relying on any text information, e.g., tweet content, replies and user descriptions, how accurately can GNNs identify fake news? Machine learning models are known to be vulnerable to adversarial attacks, and avoiding the dependence on text-based features can make the model less susceptible to the manipulation of advanced fake news fabricators. (2) How to deal with new, unseen data? In other words, how does a GNN trained on a given dataset perform on a new and potentially vastly different dataset? If it achieves unsatisfactory performance, how do we solve the problem without re-training the model on the entire data from scratch? We study the above questions on two datasets with thousands of labelled news items, and our results show that: (1) GNNs can achieve comparable or superior performance without any text information to state-of-the-art methods. (2) GNNs trained on a given dataset may perform poorly on new, unseen data, and direct incremental training cannot solve the problem---this issue has not been addressed in the previous work that applies GNNs for fake news detection. In order to solve the problem, we propose a method that achieves balanced performance on both existing and new datasets, by using techniques from continual learning to train GNNs incrementally.



<https://arxiv.org/pdf/2007.03316v2.pdf>

# Relating medicin usage

How to related the usage of different drugs? Well, maybe with graphs...

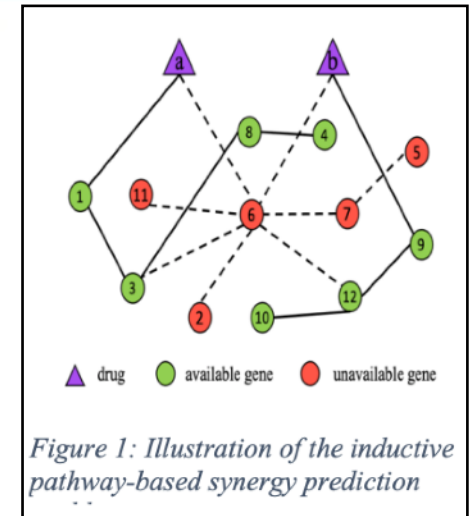
## Interpretable Drug Synergy Prediction with Graph Neural Networks for Human-AI Collaboration in Healthcare

Zehao Dong, Heming Zhang, Yixin Chen, Fuhai Li

We investigate molecular mechanisms of resistant or sensitive response of cancer drug combination therapies in an inductive and interpretable manner. Though deep learning algorithms are widely used in the drug synergy prediction problem, it is still an open problem to formulate the prediction model with biological meaning to investigate the mysterious mechanisms of synergy (MoS) for the human-AI collaboration in healthcare systems. To address the challenges, we propose a deep graph neural network, IDSP (Interpretable Deep Signaling Pathways), to incorporate the gene-gene as well as gene-drug regulatory relationships in synergic drug combination predictions. IDSP automatically learns weights of edges based on the gene and drug node relations, i.e., signaling interactions, by a multi-layer perceptron (MLP) and aggregates information in an inductive manner. The proposed architecture generates interpretable drug synergy prediction by detecting important signaling interactions, and can be implemented when the underlying molecular mechanism encounters unseen genes or signaling pathways. We test IDSP on signaling networks formulated by genes from 46 core cancer signaling pathways and drug combinations from NCI ALMANAC drug combination screening data. The experimental results demonstrated that 1) IDSP can learn from the underlying molecular mechanism to make prediction without additional drug chemical information while achieving highly comparable performance with current state-of-art methods; 2) IDSP show superior generality and flexibility to implement the synergy prediction task on both transductive tasks and inductive tasks. 3) IDSP can generate interpretable results by detecting different salient signaling patterns (i.e. MoS) for different cell lines.

### Drug Synergy:

“ An interaction between two or more drugs that causes the total effect of the drugs to be greater than the sum of the individual effects of each drug. A synergistic effect can be beneficial or harmful.”



<https://arxiv.org/pdf/2105.07082.pdf>

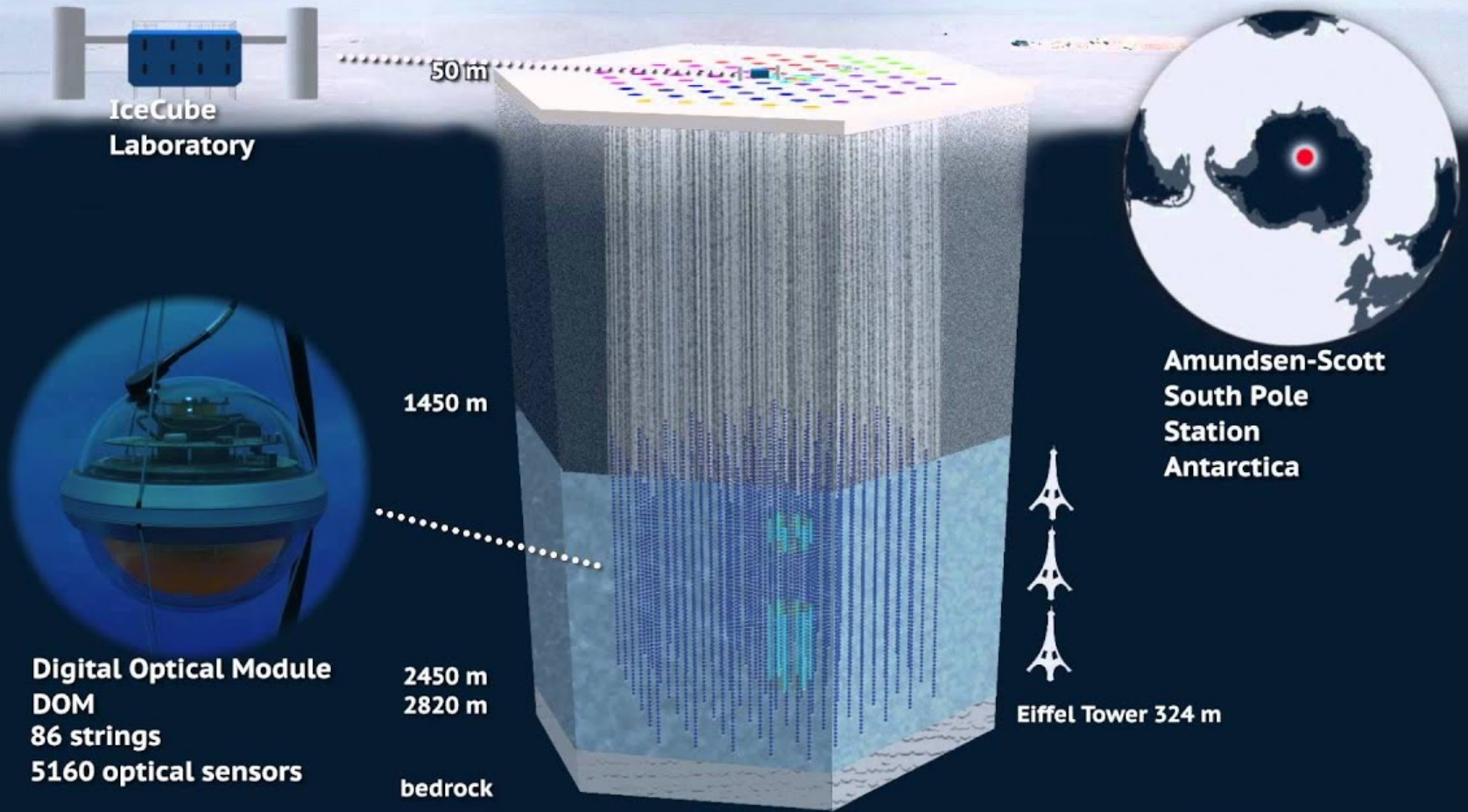


A photograph of the IceCube counting house, a multi-level industrial structure with metal stairs and railings, situated in a snowy, icy landscape. The structure is flanked by two large, cylindrical storage tanks. The scene is captured during sunset or sunrise, with a warm, orange glow in the sky. The word "IceCube" is overlaid in a large, black, serif font in the center of the image.

# IceCube



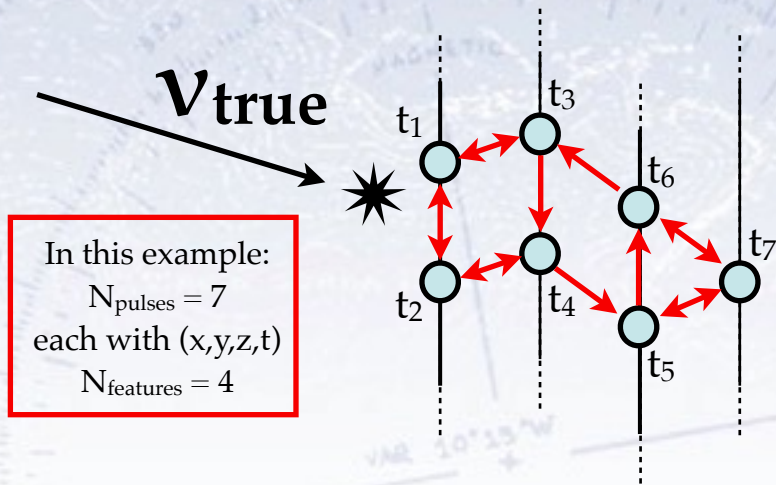
# IceCube experiment (South Pole)



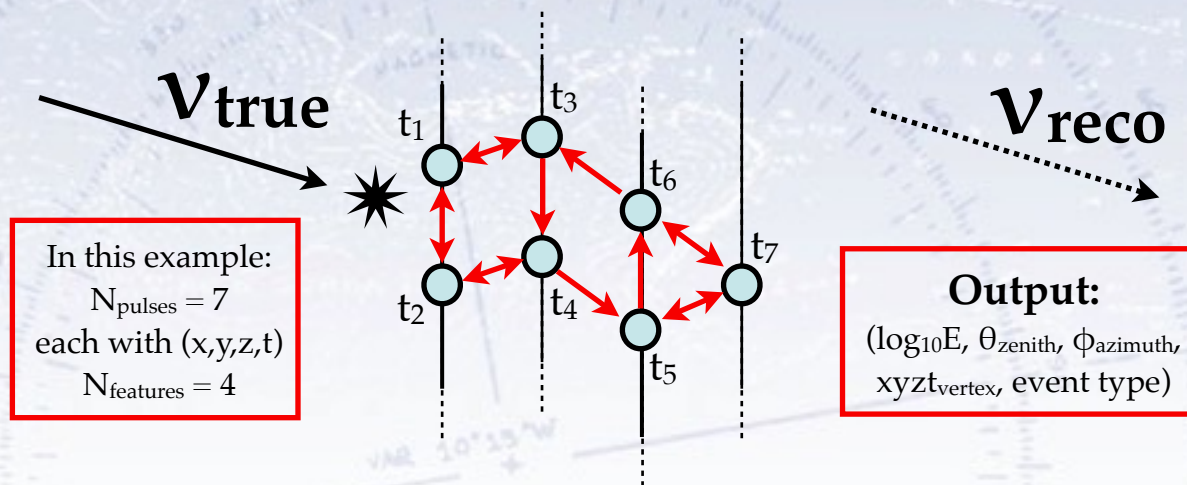


# IceCube GNN model

# Details of GNN reconstruction

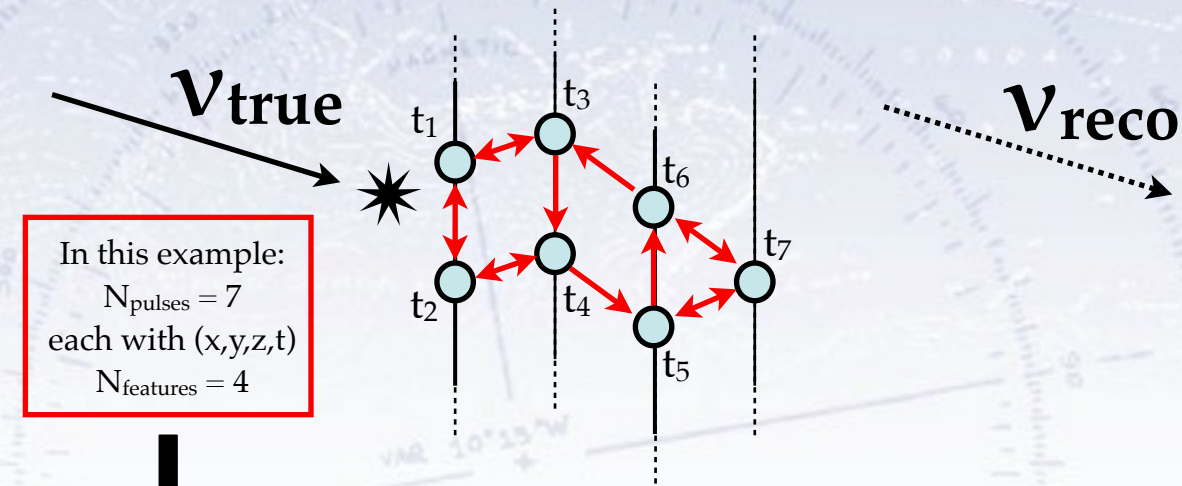


# Details of GNN reconstruction





# Details of GNN reconstruction



$$\vec{v}_1 = [x_1 \ y_1 \ z_1 \ t_1]$$

$$\vec{v}_2 = [x_2 \ y_2 \ z_2 \ t_2]$$

$\vdots$

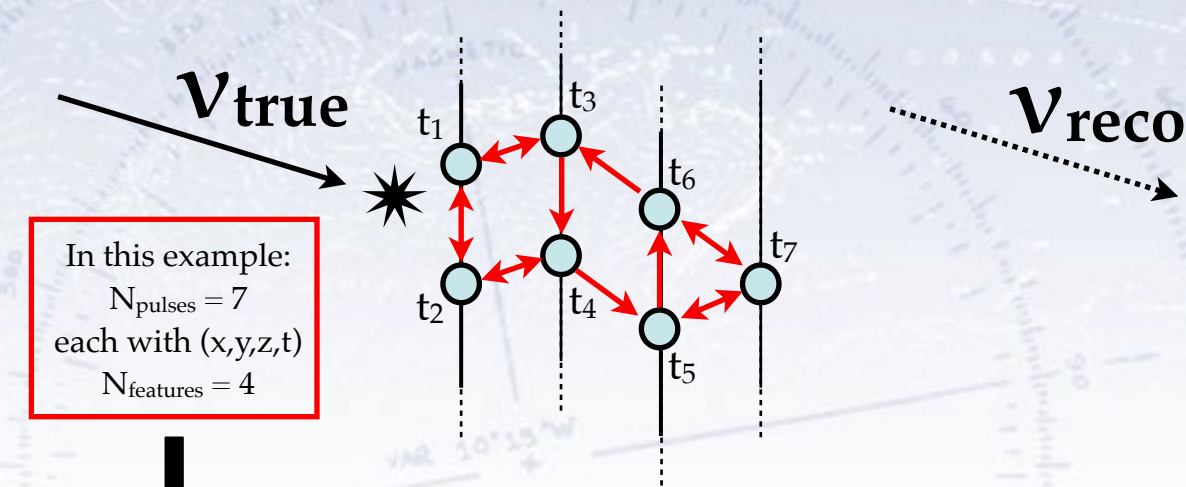
$$\vec{v}_7 = [x_7 \ y_7 \ z_7 \ t_7]$$

**Input:**

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

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

# Details of GNN reconstruction



$$\begin{aligned}
 \vec{v}_1 &= [x_1 \ y_1 \ z_1 \ t_1] \xrightarrow{EC(\vec{v}_1, \vec{v}_2, \vec{v}_3)} [g_{11} \dots g_{1N_1}] \\
 \vec{v}_2 &= [x_2 \ y_2 \ z_2 \ t_2] \xrightarrow{\hspace{1.5cm}} [g_{21} \dots g_{2N_1}] \\
 &\vdots \xrightarrow{EC(\vec{v}_4, \vec{v}_5, \vec{v}_6)} \vdots \\
 \vec{v}_7 &= [x_7 \ y_7 \ z_7 \ t_7] \xrightarrow{\hspace{1.5cm}} [g_{71} \dots g_{7N_1}]
 \end{aligned}$$

**Input:**

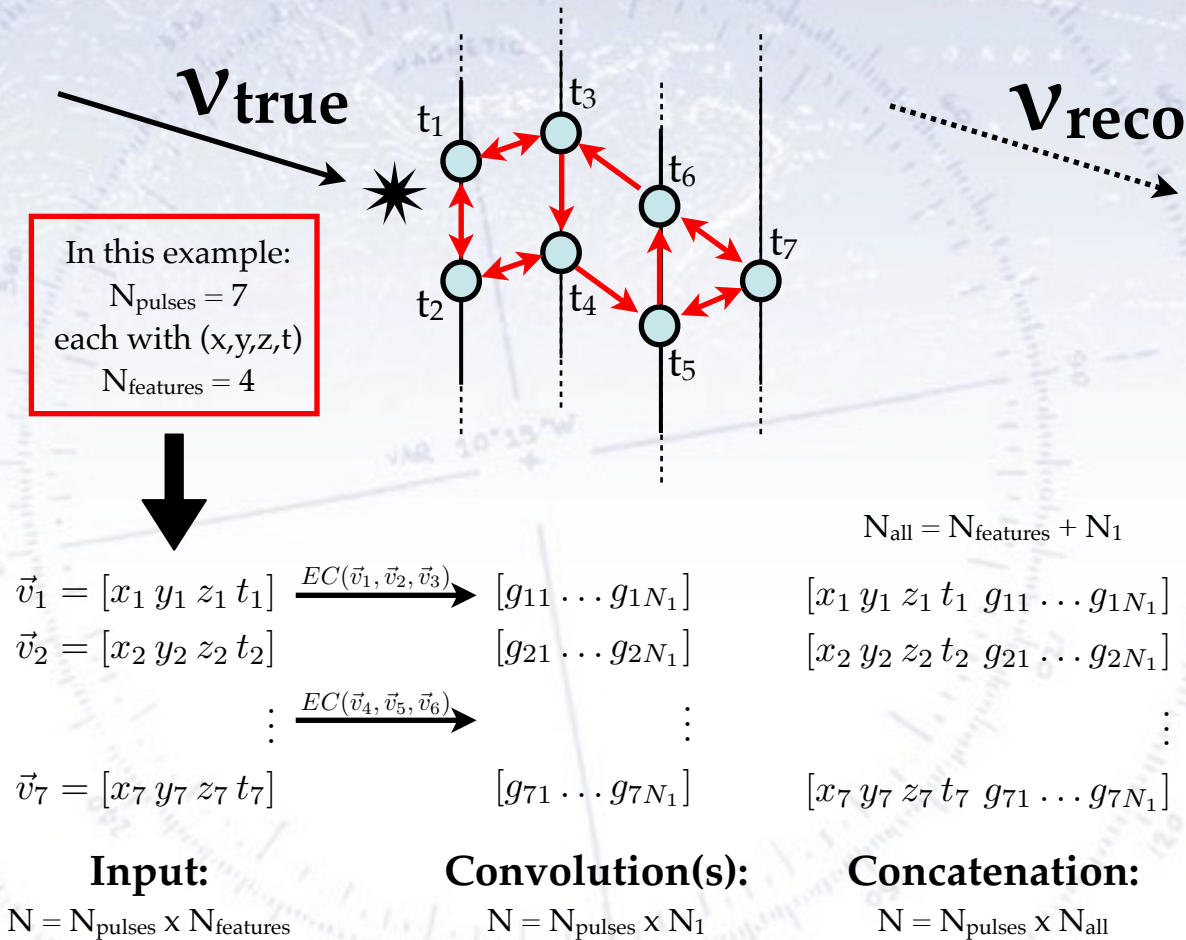
$$N = N_{\text{pulses}} \times N_{\text{features}}$$

**Convolution(s):**

$$N = N_{\text{pulses}} \times N_1$$

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).

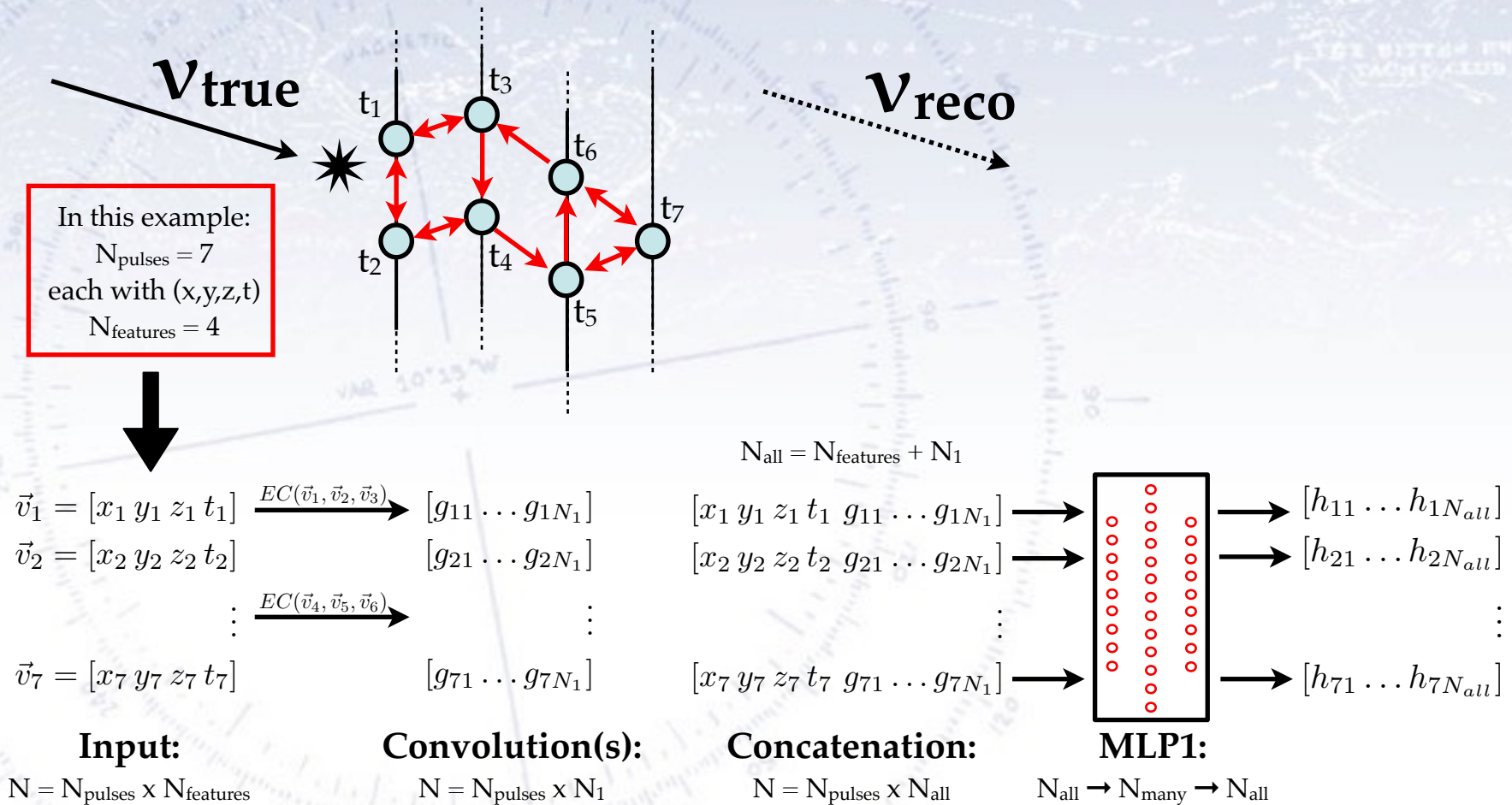
# Details of GNN reconstruction



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,

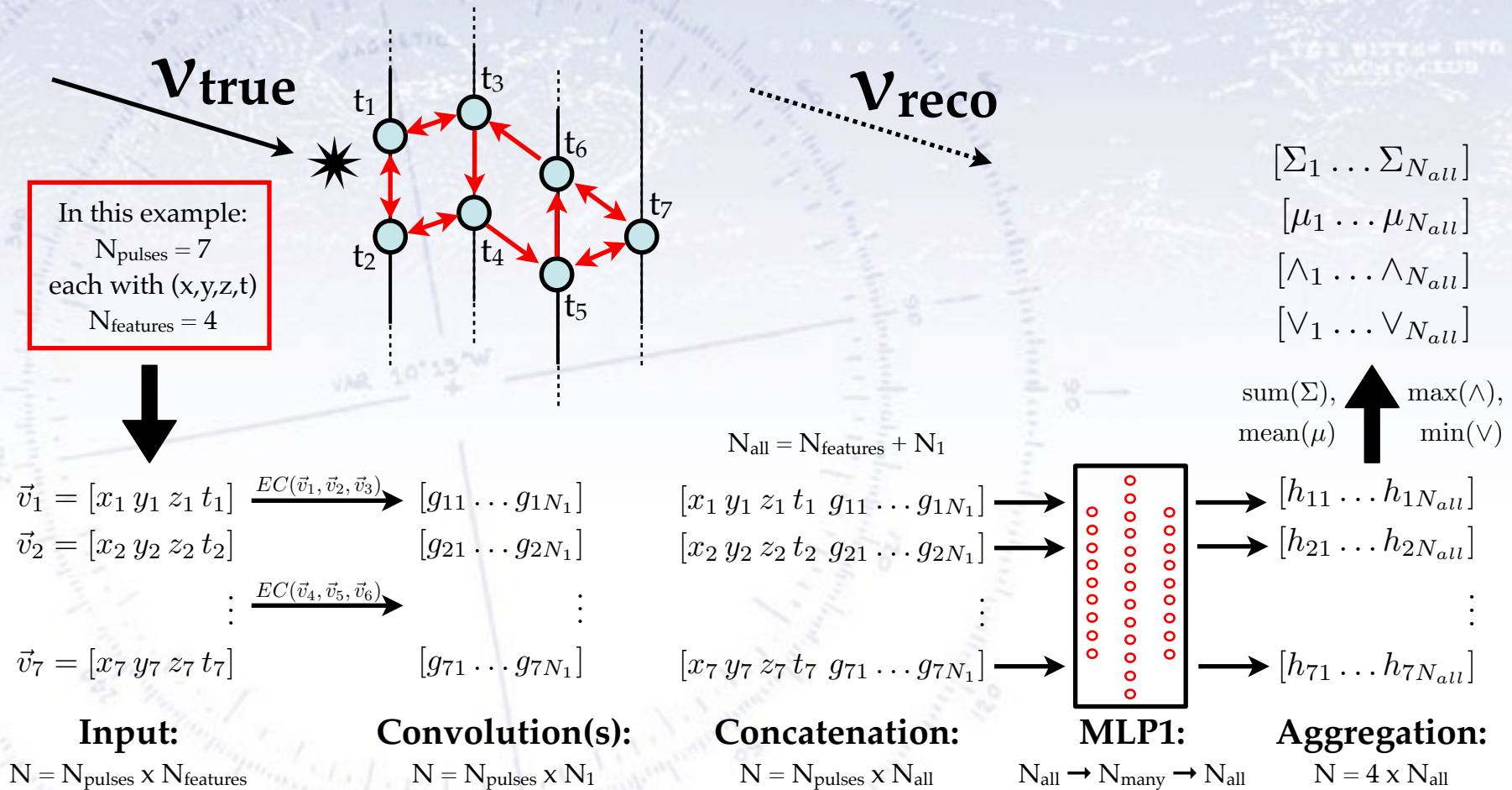


# Details of GNN reconstruction



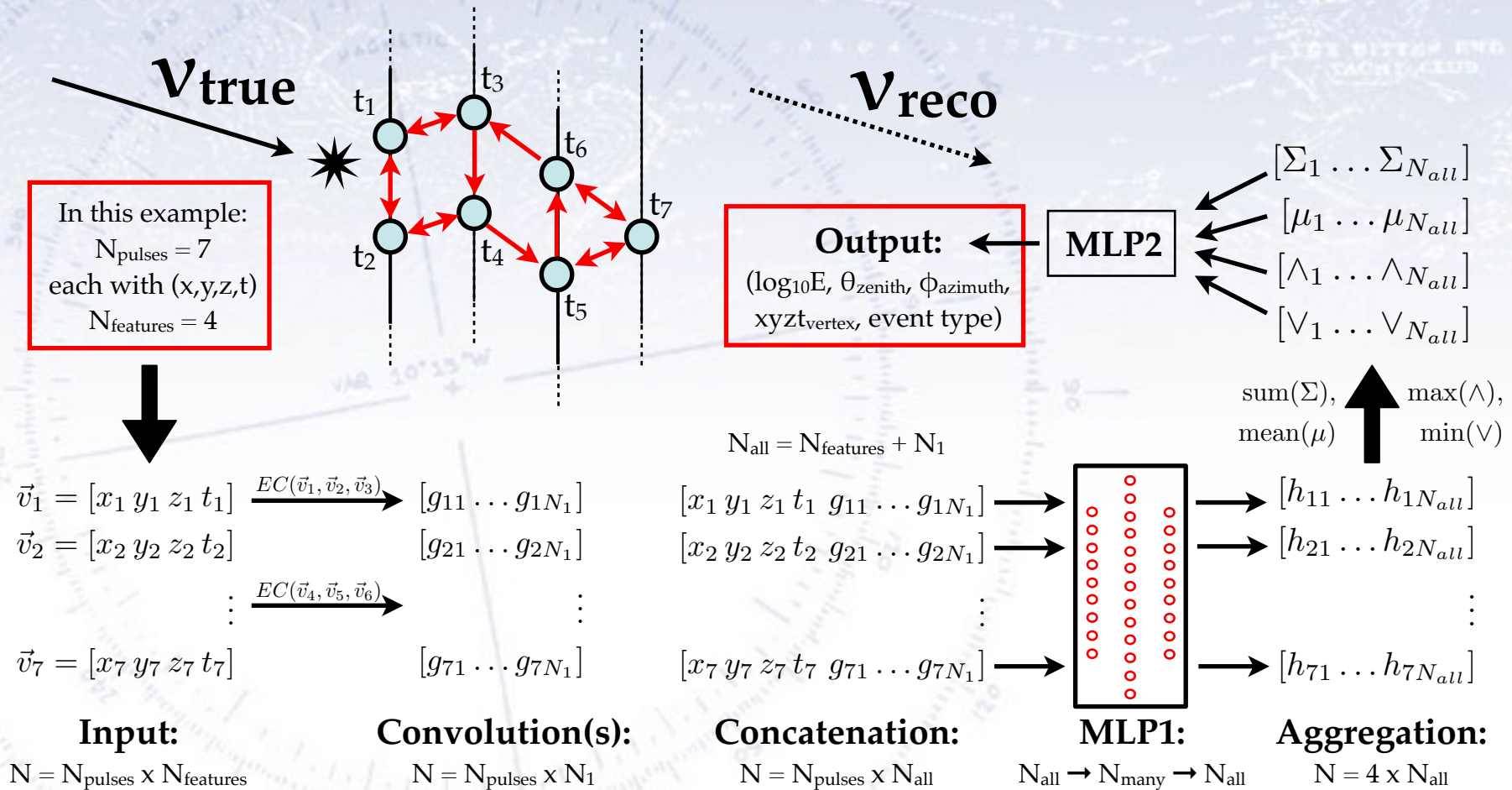
The input features of a node are combined with that of  $N$  ( $=2$ ) nearby nodes through an NN (MLP0) function, yielding an (abstract) vector for each node. This can be repeated (not shown). All the features are then combined (concatenated) into long vectors, which are again put through an NN (MLP1)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

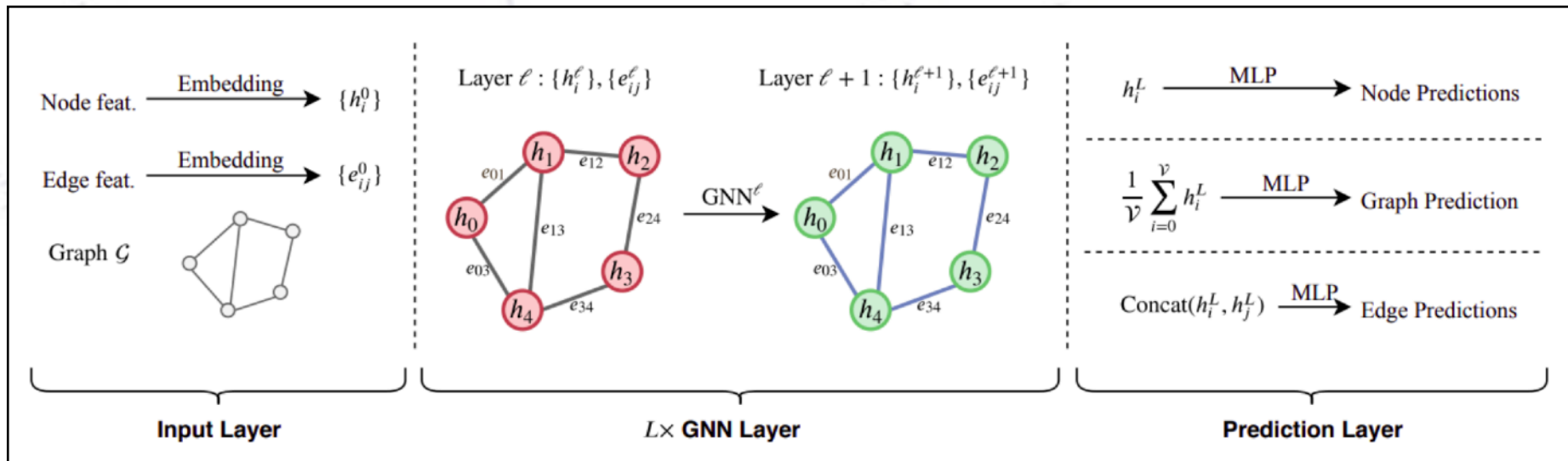
The EdgeConv convolution operator ensures permutation invariance.

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.

# What can GNN predict?

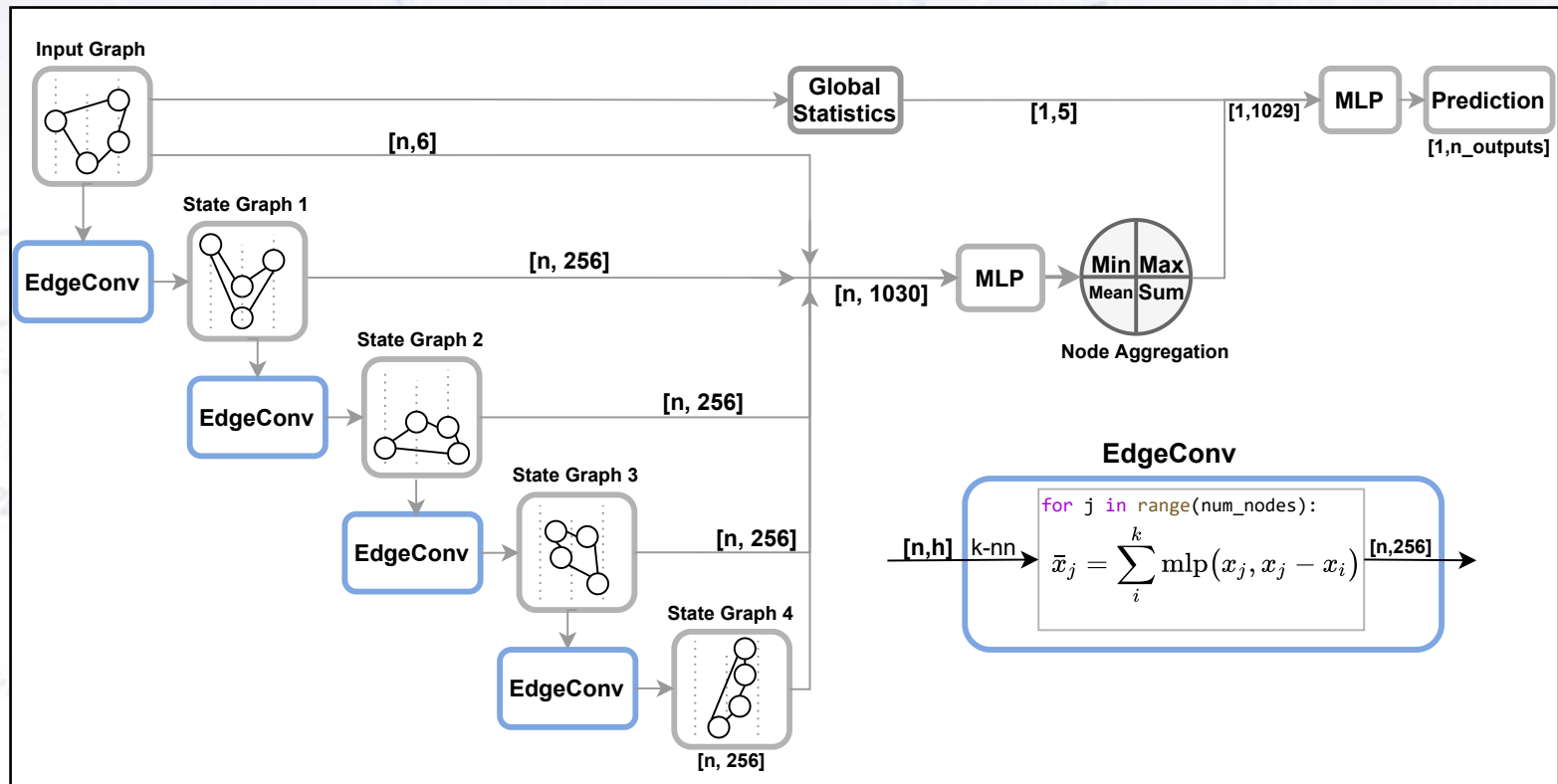
GNNs are capable to make three sorts of predictions:

- Node prediction (is this node signal or noise?)  
Obtained simply from an MLP on the convoluted node features.
- Edge prediction (is this edge important or not?)  
Obtained from an MLP on (concatenated) pairs of node features.
- Graph prediction (is this graph an X or not? What is the Y of this graph?)  
Obtained through an MLP on a summary of the graph nodes.  
Here, there are several options of dimensionality and aggregation.



# GraphNet

The GNN model is outlined below, which is also the figure for our IceCube GNN paper (<https://arxiv.org/abs/2209.03042>).

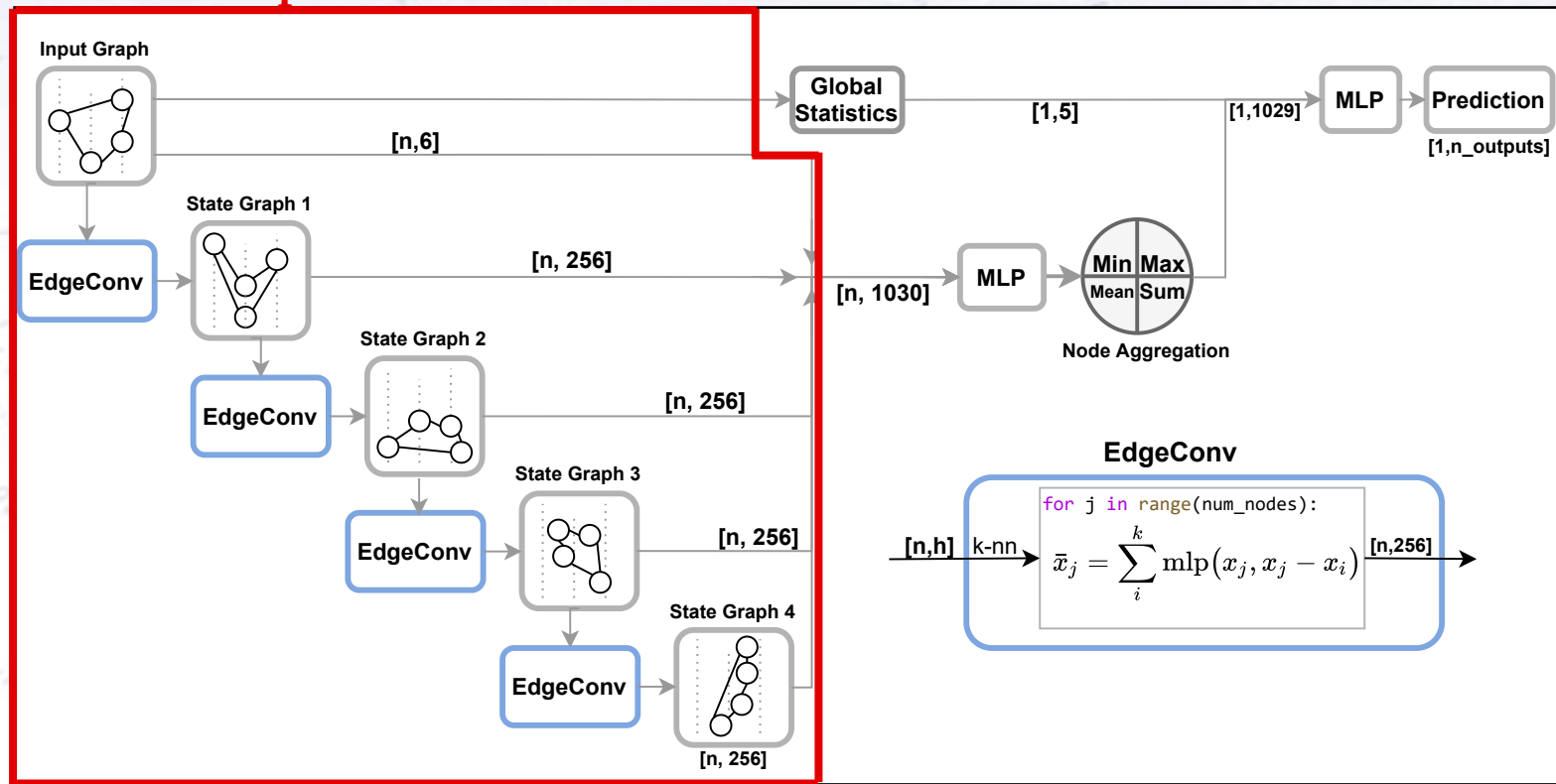




# GraphNet

The GNN model is outlined below, which is also the figure for our IceCube GNN paper (<https://arxiv.org/abs/2209.03042>).

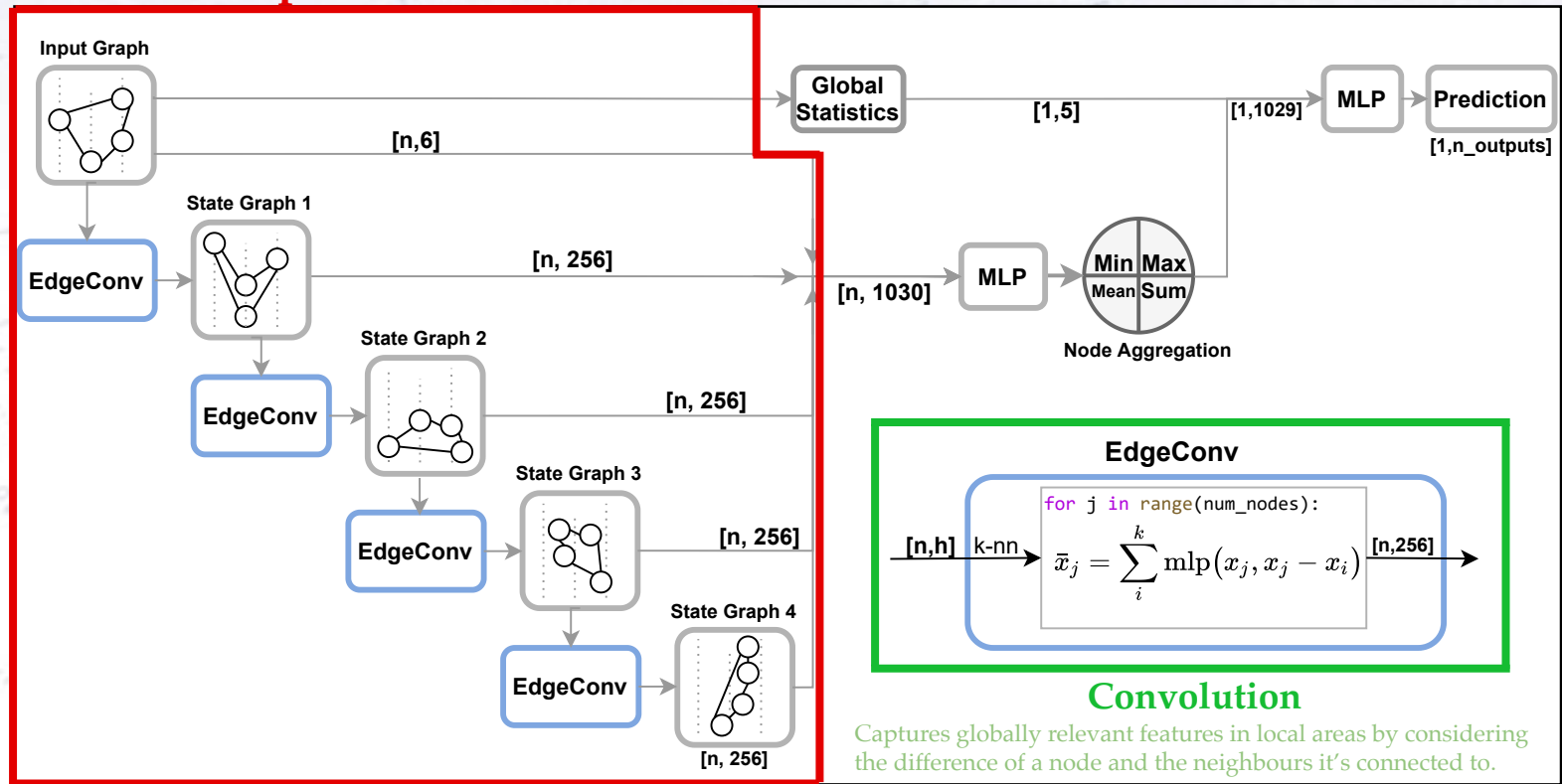
## Graph Convolutions



# GraphNet

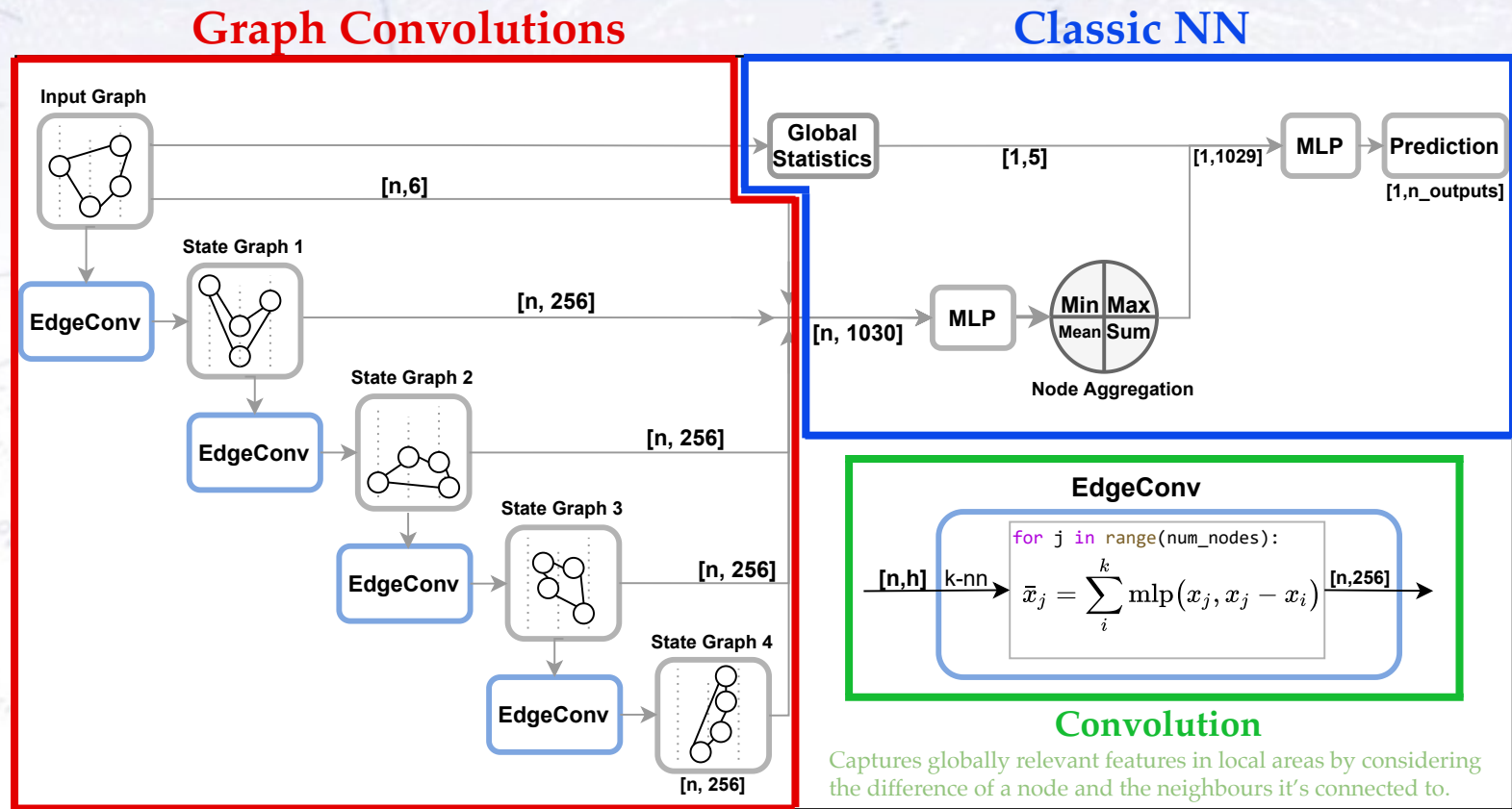
The GNN model is outlined below, which is also the figure for our IceCube GNN paper (<https://arxiv.org/abs/2209.03042>).

## Graph Convolutions



# GraphNet

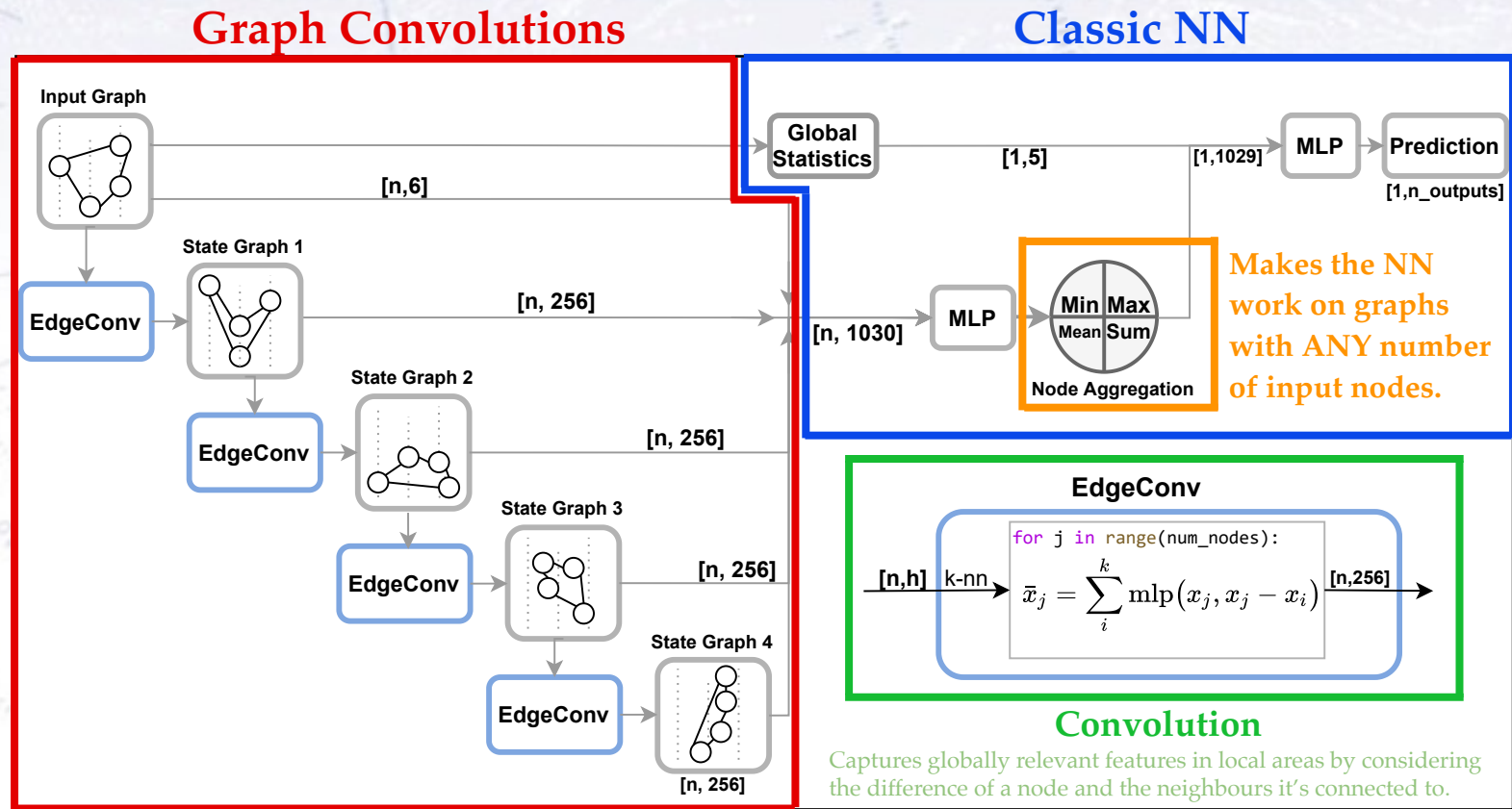
The GNN model is outlined below, which is also the figure for our IceCube GNN paper (<https://arxiv.org/abs/2209.03042>).





# GraphNet

The GNN model is outlined below, which is also the figure for our IceCube GNN paper (<https://arxiv.org/abs/2209.03042>).





# 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>

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

The IceCube challenge was also made into a Kaggle competition - .

# 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 curation, 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 curation, etc.

**Period:** July 2020 - May 2021 (Master Thesis).

**Email:** pcs557@alumni.ku.dk

**Result:** [GitHub repository](#).



## **Bjoern Moelvig**

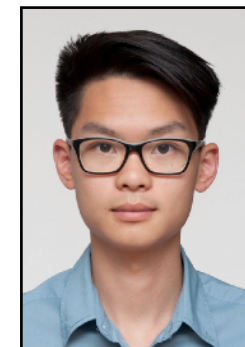
**Project part:** RNN/GRU reconstruction, loss function exploration

**Period:** September 2019 - October 2020.

**Results:** [Master Thesis](#), [Thesis Defence](#), and [GitHub repository](#).



**Philipp Eller**



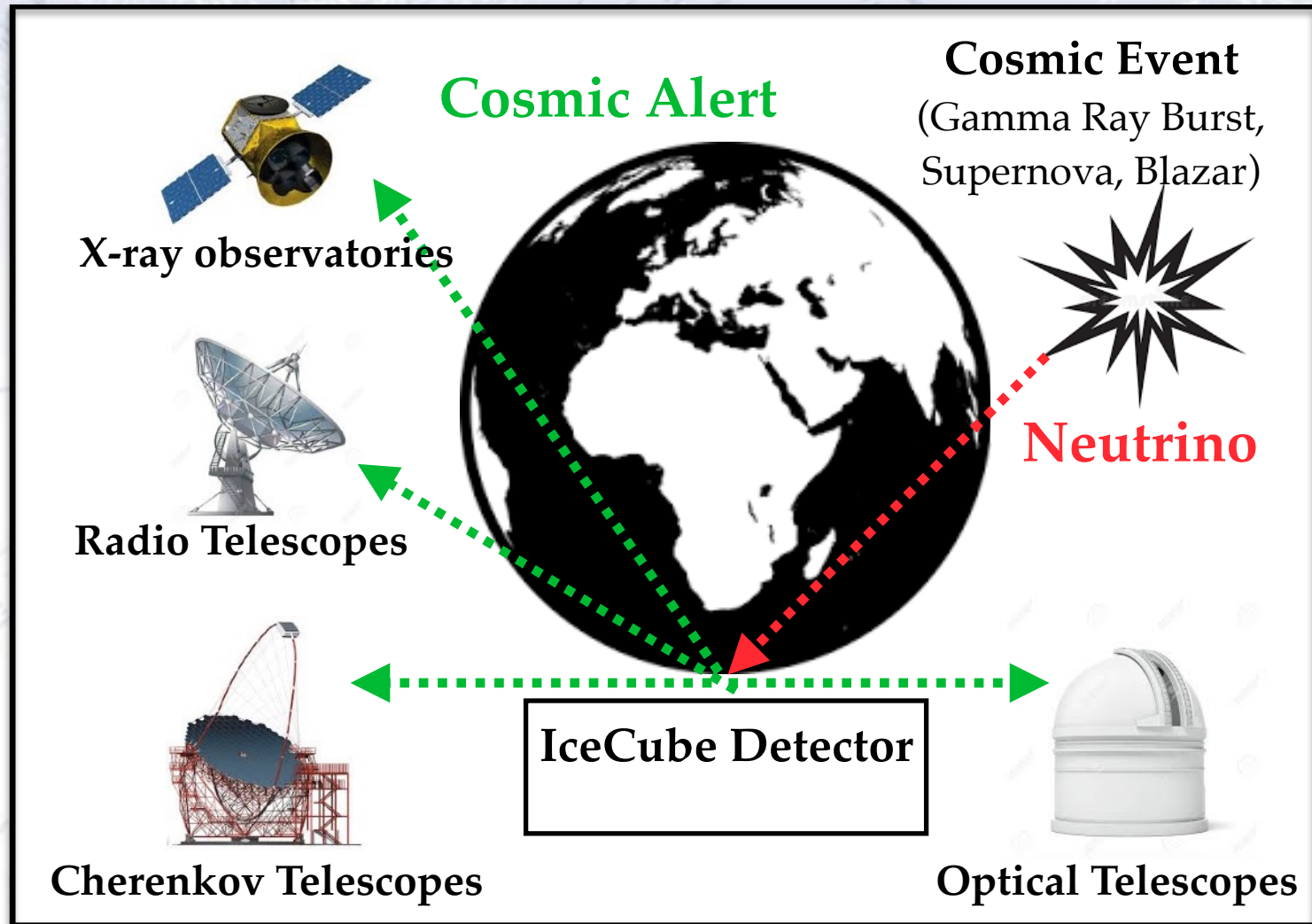
**Martin Minh<sub>38</sub>**





# Dreaming

# Seeing the Universe in $\nu$ light



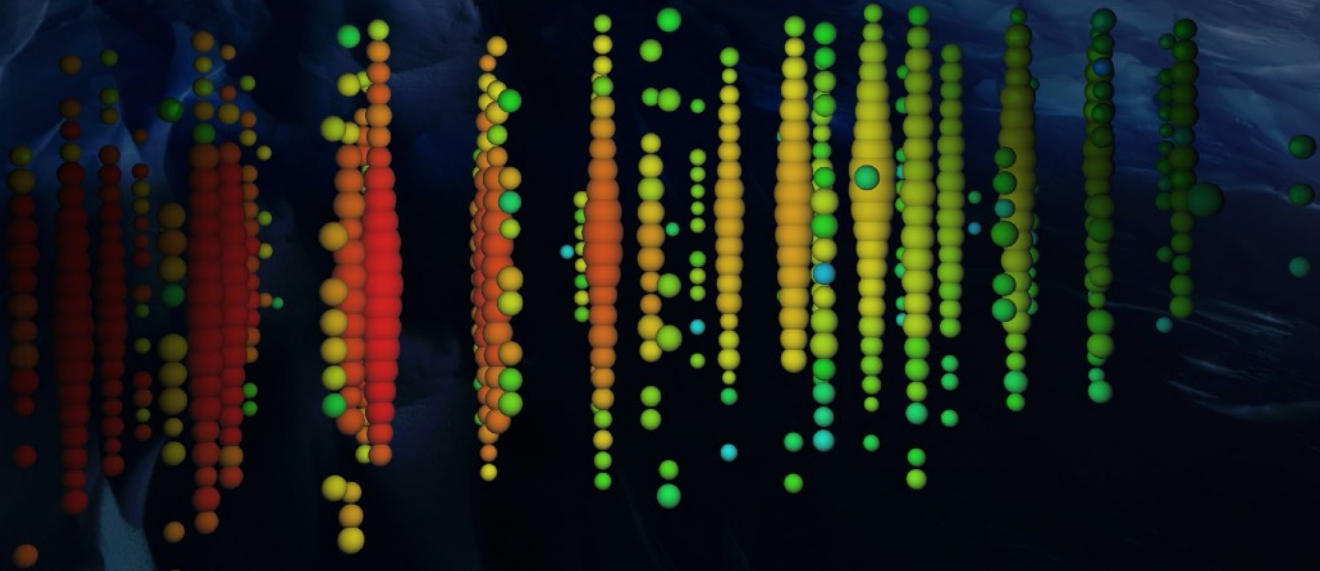


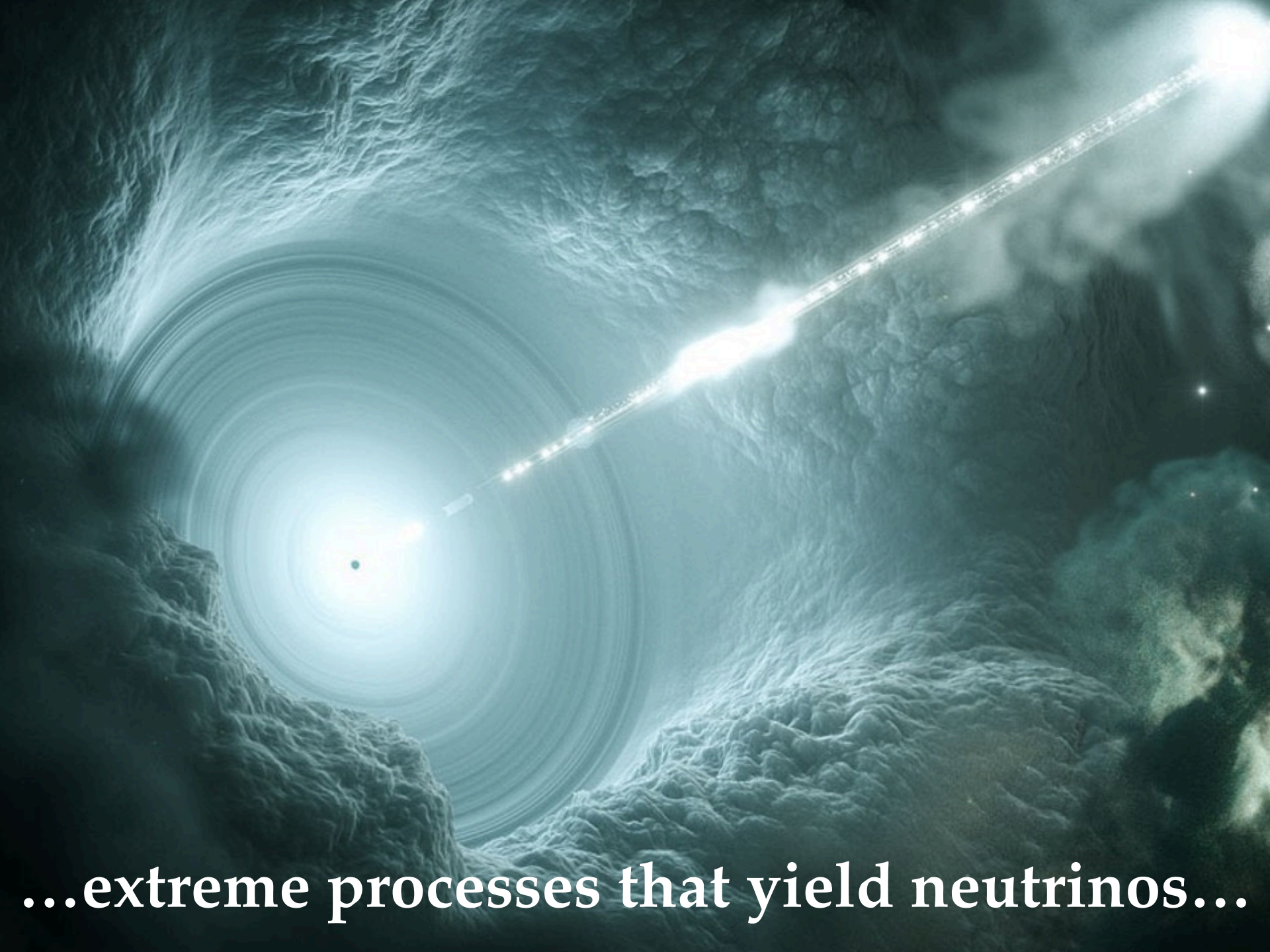
# Seeing the Universe with neutrinos





# IceCube Neutrino Telescope can see...





...extreme processes that yield neutrinos...

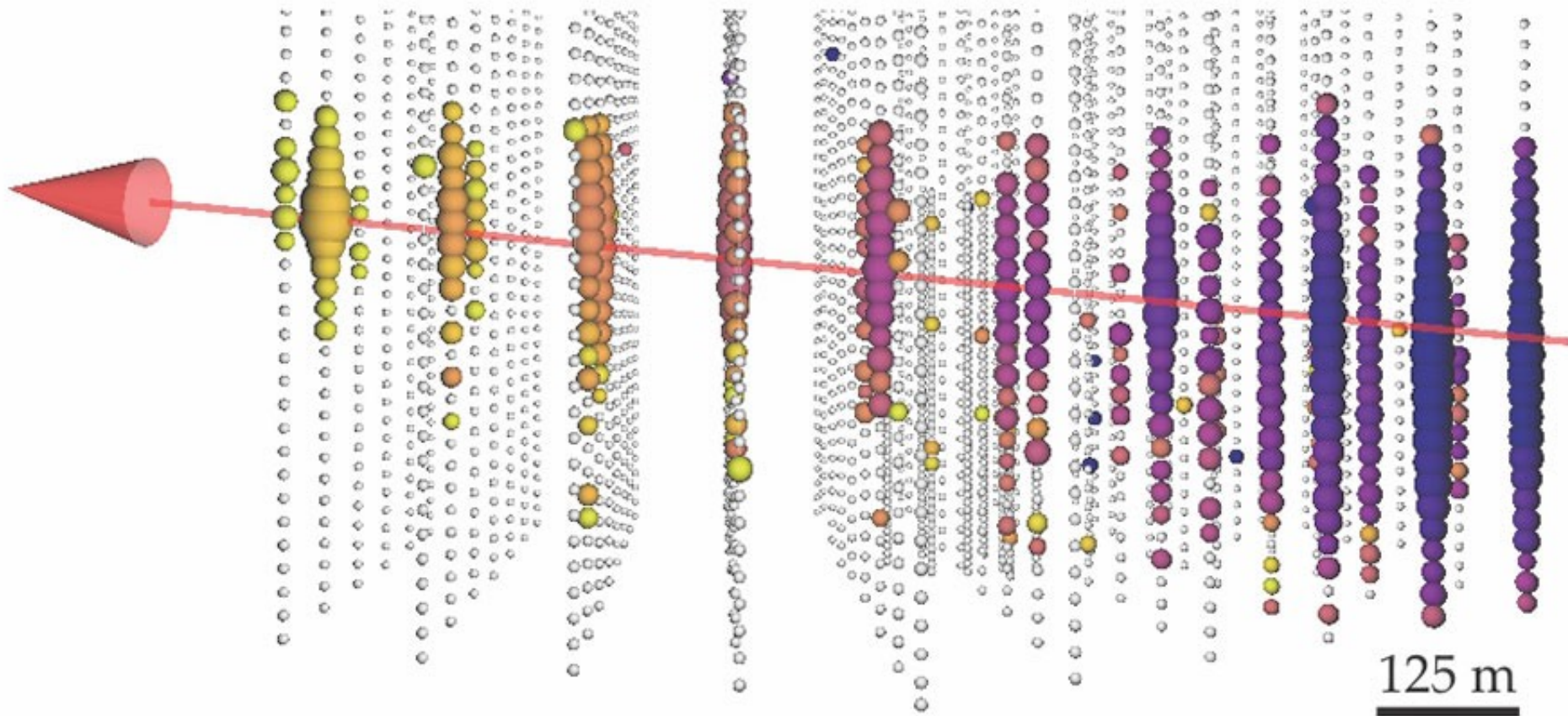


...which are great cosmic messengers...





...that can be observed by  
IceCube  $\nu$ -Telescope...



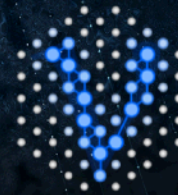


GraphNeT

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

