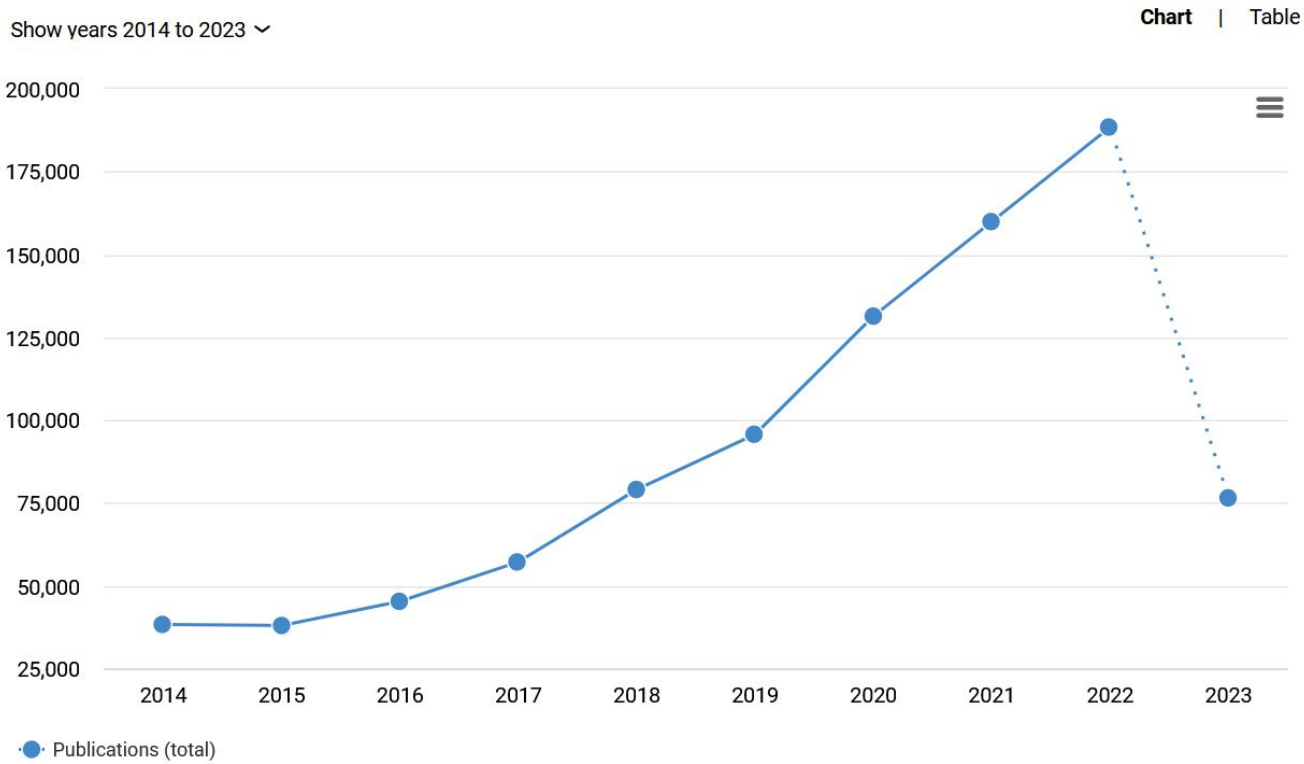
A Quick Introduction to Graph Neural Networks

Advanced Deep Learning Train-the-Trainer Workshop / 20th of June 2023

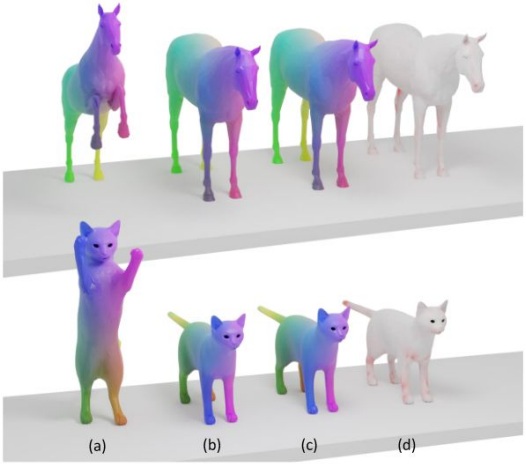
Rasmus Ørsøe

Technical University of Munich

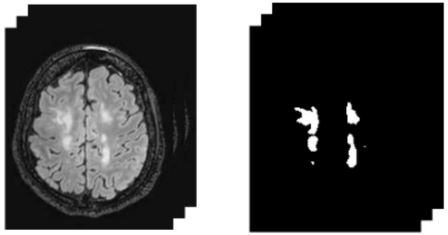


Number of publications matching “Graph Neural Networks”.

From dimensions.ai

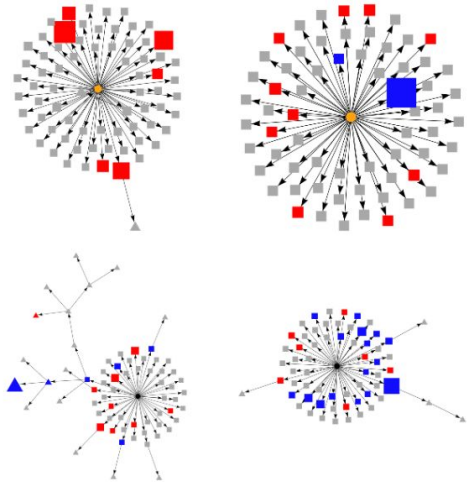


Bending Graphs: Hierarchical Shape Matching using Gated Optimal Transport

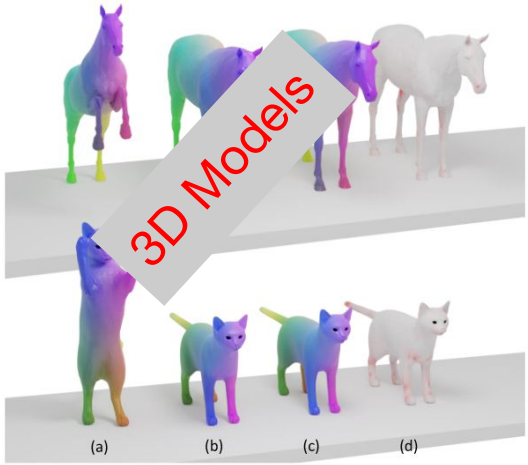


Beyond Voxel Prediction Uncertainty: Identifying brain lesions you can trust

Protein Fold Classification using Graph Neural Network and Protein Topology Graph

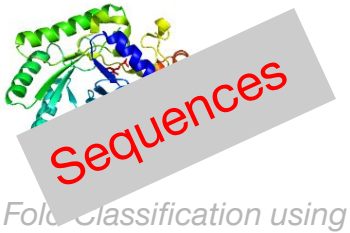


Modelling Social Context for Fake News Detection: A Graph Neural Network Based Approach

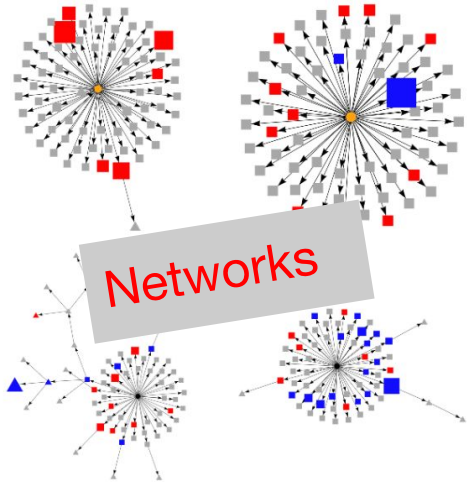


Bending Graphs: Hierarchical Shape Matching using Gated Optimal Transport

Beyond Voxel Prediction Uncertainty: Identifying brain lesions you can trust



Protein Fold Classification using Graph Neural Network and Protein Topology Graph



Modelling Social Context for Fake News Detection: A Graph Neural Network Based Approach

Overview of this talk

- **Graphs**
  - 7 Bridges of Königsberg
  - Modern Graph Theory

- **Graph Neural Networks**
  - Convolutions on Graphs
  - Exotic Learning Goals

- **Examples of applications in Physics**

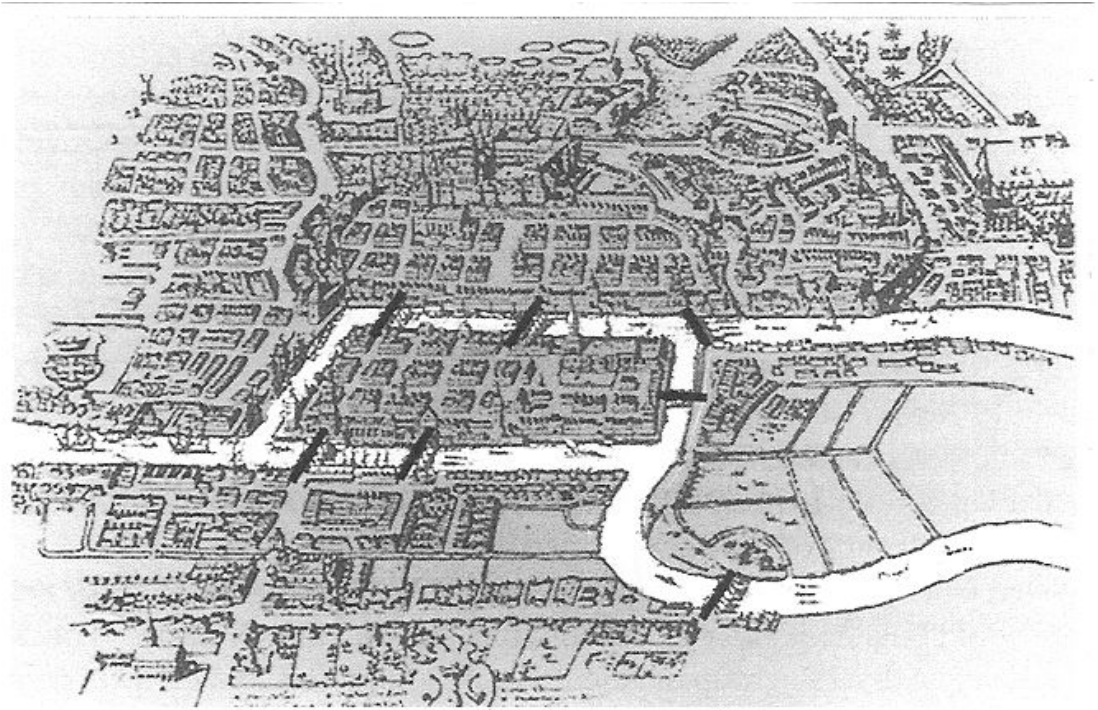
The 7 Bridges of Königsberg

The 7 Bridges of Königsberg

Historical math problem **solved by Euler in 1736**, widely regarded as the birth of graph theory.

Problem Statement:

Starting from anywhere you please, you must cross all bridges. However, you must only cross a bridge once!



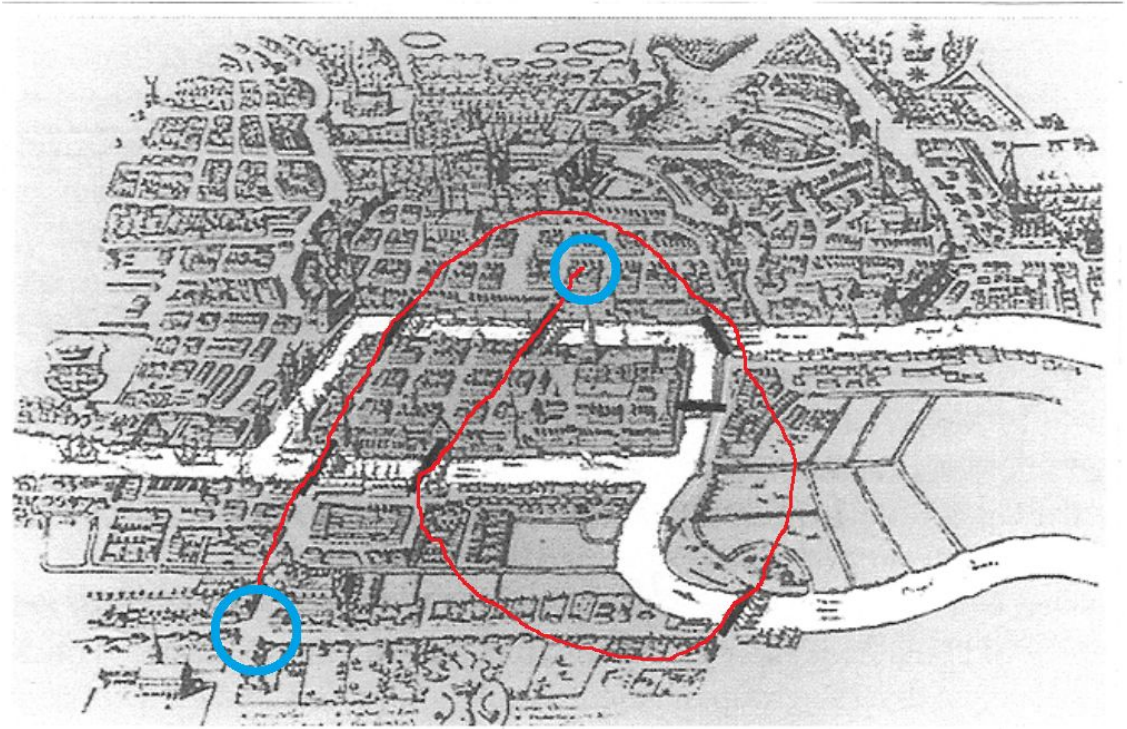
Map of Königsberg in 1800's, borrowed from Optimised Analysis and Visualisation of Metabolic Data Using Graph Theoretical Approaches

The 7 Bridges of Königsberg

Problem Statement:

Starting from anywhere you please, you must cross all bridges. However, you must only cross a bridge once!

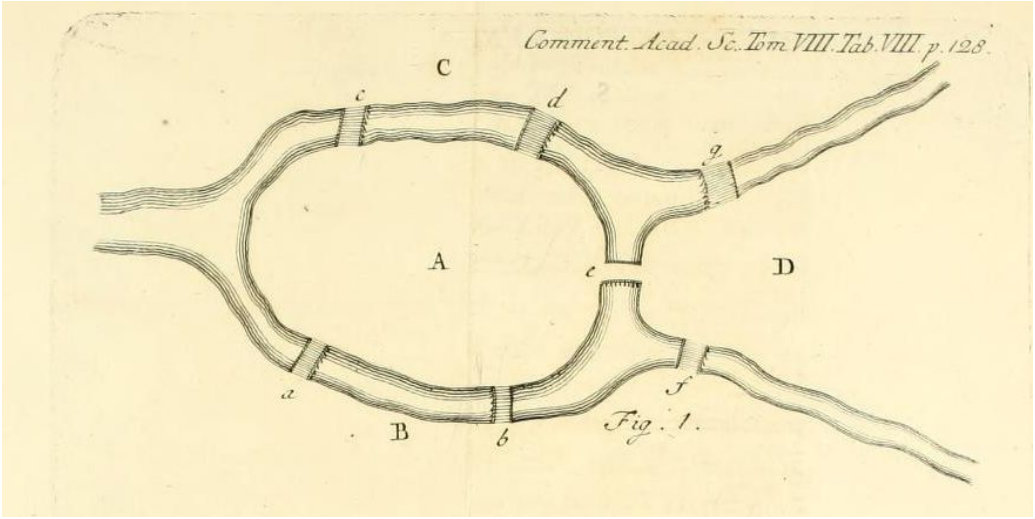
At the time, this **was considered to be a numerical problem**, as no set of first principles existed



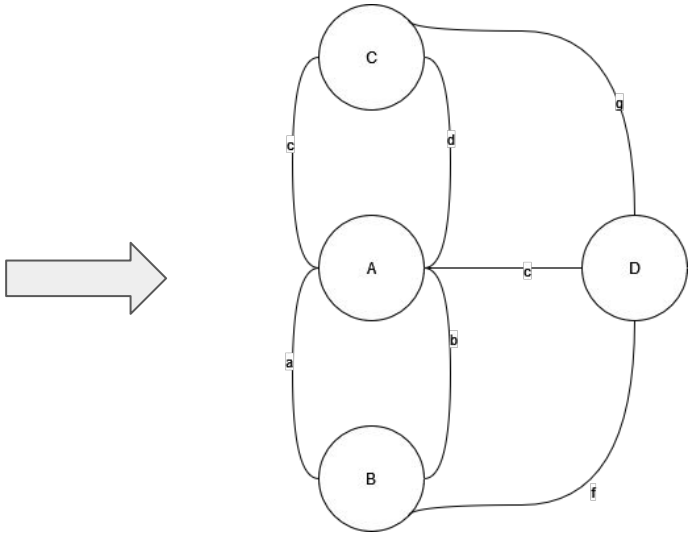
Map of Königsberg in 1800's, borrowed from Optimised Analysis and Visualisation of Metabolic Data Using Graph Theoretical Approaches

The 7 Bridges of Königsberg

Euler's Perspective:
The problem is comprised of landmasses and connections between them.
Perhaps principles can be deduced from such abstraction.



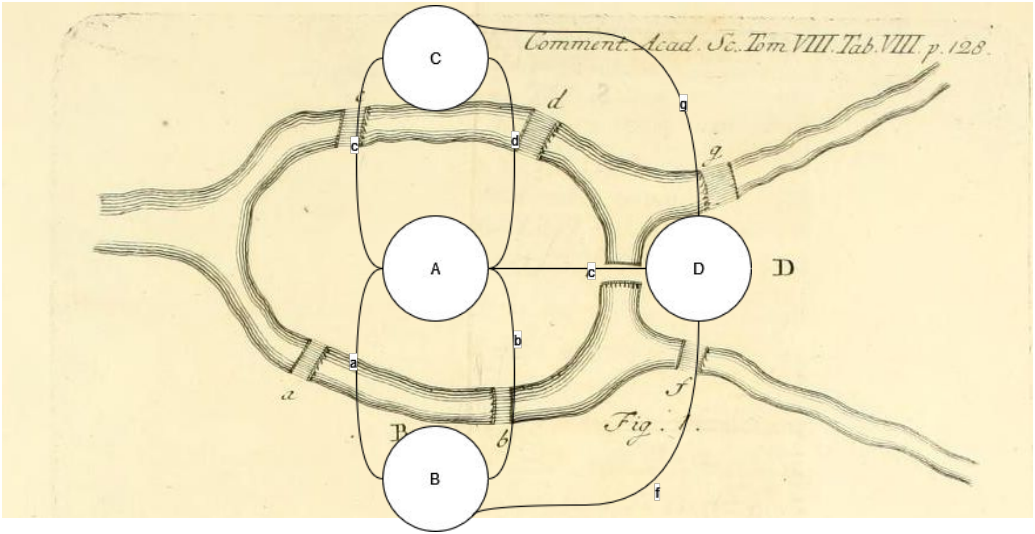
Original Drawing from Euler's solution in 1736



Euler's solution as a modern-day graph

The 7 Bridges of Königsberg

Euler's Perspective:
The problem is comprised of landmasses and connections between them.
Perhaps principles can be deduced from such abstraction.



Original Drawing from Euler's solution in 1736,
with a graph representation on top

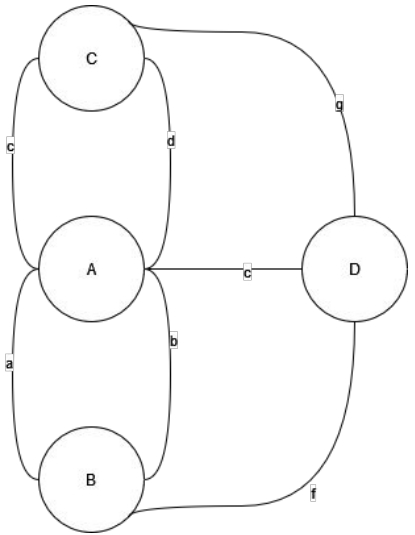
The 7 Bridges of Königsberg

Euler's Theorem:

The problem has a solution if at most two land masses have an uneven number of bridges connected to it.

The theorem was proven by Carl Hierholzer in the 1870's.

It means that **there is no solution** to the The 7 Bridges of Königsberg!



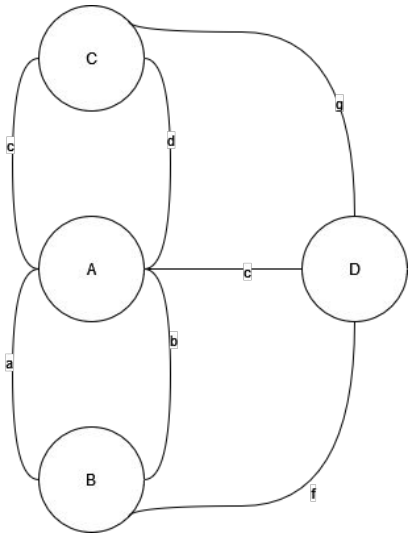
Euler's solution as a modern-day graph

Why is *that* interesting?

Seemingly complex problems can become simple if represented as graphs.

Euler was able to use quantities specific to graphs (the number of bridges connected to each landmass) to quickly assess if the problem is solvable and if so, where a potential path may start and end.

Today, there are many such known quantities and many different kinds of graphs….



Euler's solution as a modern-day graph

Graph Theory

Graph basics

A graph is a collection of two things

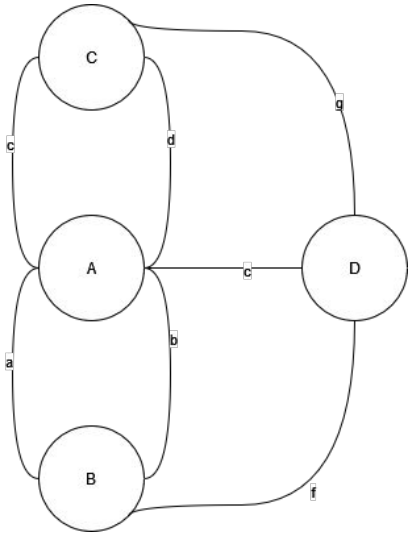
Nodes (landmasses)
a.k.a. “Vertices”

Represents data (“node features”)

Edges (bridges)
a.k.a. “Connections”

Implies relationship between data
Can also represent data (“edge features”)

Euler’s graph is an undirected, homogeneous graph

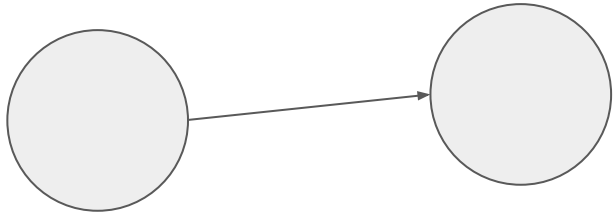


Euler’s solution as a modern-day graph

Graph Basics

A directed graph has an implied direction in it's edges.

Directed edges can used to model the flow of information.



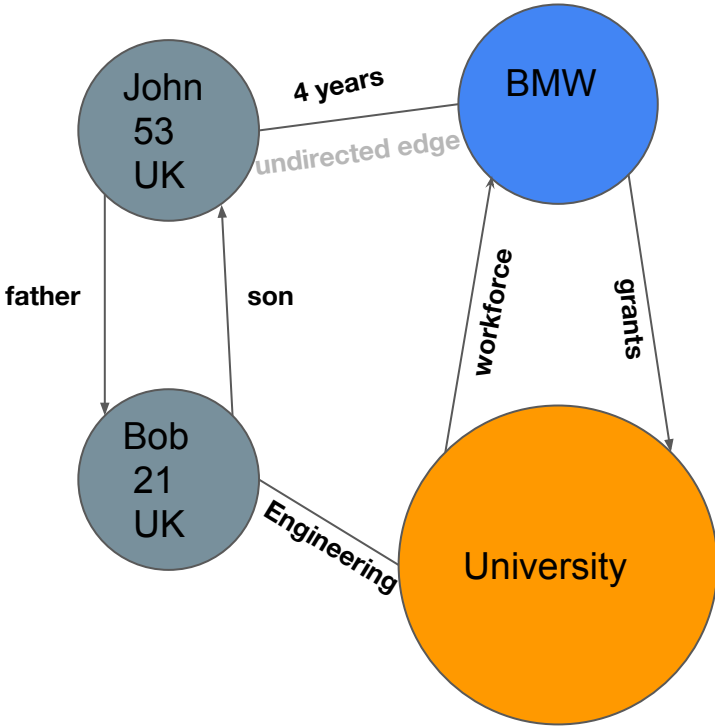
Simple example of a directed graph

Graph Basics

The Euler graph had a homogenous node type (landmasses) and edge type (bridges).

Heterogeneous graphs contain different kinds of nodes and edges

Can be used to model complex data structures



Example of a heterogeneous graph, with multiple kinds of edges and nodes.

Graph basics

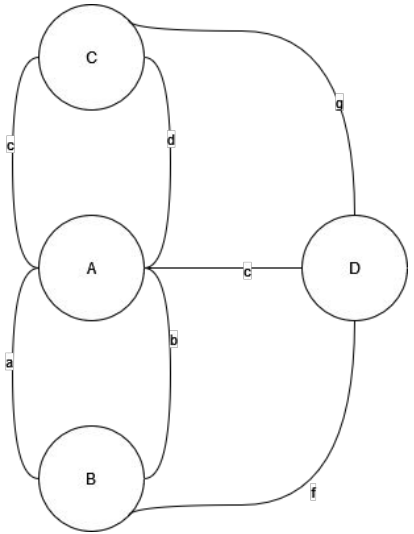
In addition to nodes and edges, graphs also have

Adjacency Matrix

$$\mathrm{A}_{ij} := \begin{cases} 1, & \text{if there is an edge between } n_i \text{ and } n_j \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{bmatrix} & A & B & C & D \\ A & 0 & 1 & 1 & 1 \\ B & 1 & 0 & 0 & 1 \\ C & 1 & 0 & 0 & 1 \\ D & 1 & 1 & 1 & 0 \end{bmatrix}$$

Adjacency Matrix for the Euler graph



Euler's solution as a modern-day graph

Graph basics

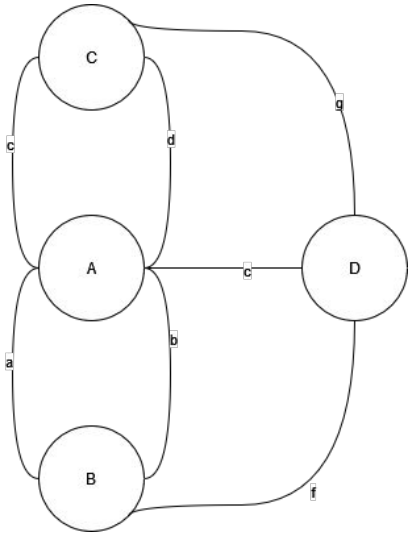
In addition to nodes and edges, graphs also have

Adjacency Matrix

$$\mathrm{A}_{ij} := \begin{cases} 1, & \text{if there is an edge between } n_i \text{ and } n_j \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{bmatrix} & A & B & C & D \\ A & 0 & 1 & 1 & 1 \\ B & 1 & 0 & 0 & 1 \\ C & 1 & 0 & 0 & 1 \\ D & 1 & 1 & 1 & 0 \end{bmatrix}$$

Adjacency Matrix for the Euler graph



Euler's solution as a modern-day graph

Degree Matrix (number of edges)

$$\mathrm{D}_{ij} := \begin{cases} \text{degree}(n_i), & \text{if i = j} \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{bmatrix} & A & B & C & D \\ A & 5 & 0 & 0 & 0 \\ B & 0 & 3 & 0 & 0 \\ C & 0 & 0 & 3 & 0 \\ D & 0 & 0 & 0 & 3 \end{bmatrix}$$

Degree Matrix for the Euler graph

Graph basics

In addition to nodes and edges, graphs also have

Graph Laplacian

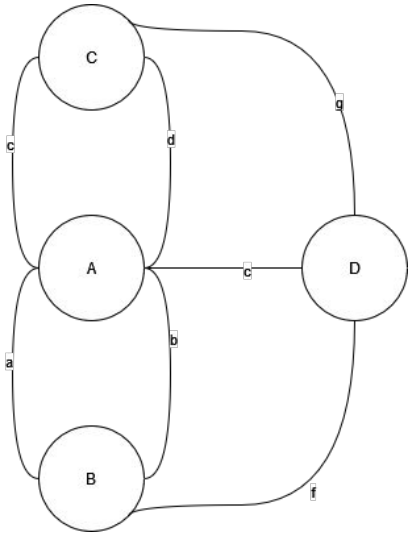
$$L := D - A$$

$$\begin{bmatrix} & A & B & C & D \\ A & 5 & 0 & 0 & 0 \\ B & 0 & 3 & 0 & 0 \\ C & 0 & 0 & 3 & 0 \\ D & 0 & 0 & 0 & 3 \end{bmatrix} - \begin{bmatrix} & A & B & C & D \\ A & 0 & 1 & 1 & 1 \\ B & 1 & 0 & 0 & 1 \\ C & 1 & 0 & 0 & 1 \\ D & 1 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} & A & B & C & D \\ A & 5 & -1 & -1 & -1 \\ B & -1 & 3 & 0 & -1 \\ C & -1 & 0 & 3 & -1 \\ D & -1 & -1 & -1 & 3 \end{bmatrix}$$

Laplacian for the Euler graph

A graph analogue to the Laplacian operator on continuous functions (divergence of gradients)

Used in various modified versions in *some* GNNs.



Euler's solution as a modern-day graph

Graph basics

In addition, one also has

Graph Laplacian

$$L := D - A$$

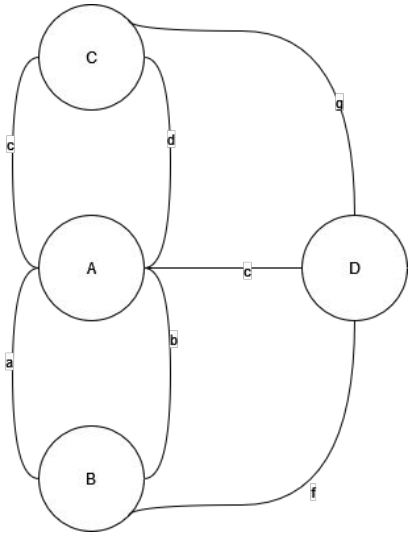
$$\begin{bmatrix} & A & B & C & D \\ A & 5 & 0 & 0 & 0 \\ B & 0 & 3 & 0 & 0 \\ C & 0 & 0 & 3 & 0 \\ D & 0 & 0 & 0 & 3 \end{bmatrix} - \begin{bmatrix} & A & B & C & D \\ A & 0 & 1 & 1 & 1 \\ B & 1 & 0 & 0 & 1 \\ C & 1 & 0 & 0 & 1 \\ D & 1 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} & A & B & C & D \\ A & 5 & -1 & -1 & -1 \\ B & -1 & 3 & 0 & -1 \\ C & -1 & 0 & 3 & -1 \\ D & -1 & -1 & -1 & 3 \end{bmatrix}$$

Laplacian for the Euler graph

In fact, the interpretation of the Laplacian operator has lead to two competing fields in graph theory:

Spatial
Uses laplacian as-is; local connectivity is used

Spectral
Decomposes Laplacian into Fourier modes; the graph is often treated as one signal



Euler's solution as a modern-day graph

Images are graphs

An image is a special kind of graph

- Nodes represents pixels
- Edges are drawn to form a grid

By definition:
The distance between neighbouring pixels in an image is constant.

When you represent your data as an image, you're implying that the geometry of your data is grid-like.

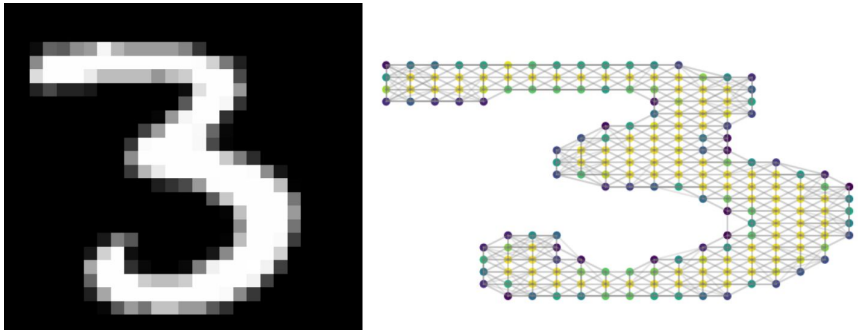


Illustration from exercise; shows graph representation of image.

Images are graphs

The selling point for graphs is their flexibility as ***a data representation vehicle***

Pro's:

- No artificial constraints
- "Generalized image"
- Can represent (probably) any data
- Leads to many definitions of convolutions

Con's

- Graph representation itself becomes a "hyperparameter"

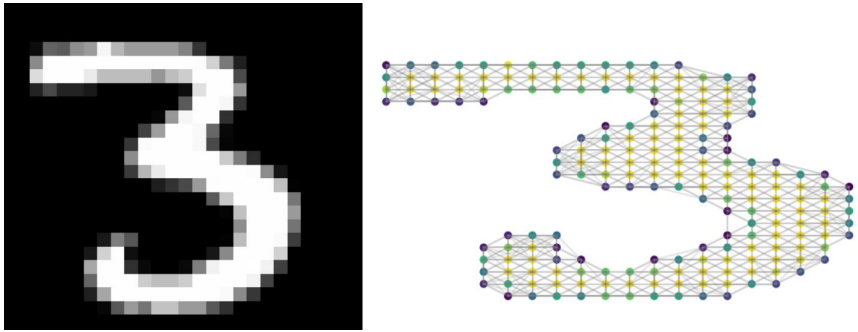


Illustration from exercise; shows graph representation of image.

Graph Neural Networks

Graph Neural Networks

GNNs are neural networks that act on graph structured data. Most are *convolutional* graph networks (GCNs), where the abstractness of graphs leads to many different definitions of “convolution”.

Generally, a convolution on a graph is a function

where $f(X, A) : \mathbb{R}^{n \times d} \longrightarrow \mathbb{R}^{n \times k}$

X : [n,d]-dimensional node features
A : adjacency matrix
n : number of nodes
d : number of node feature dimensions
k : dimension of node feature embedding space

and is equivariant to permutations of the node ordering. Extending the symmetries of GCNs is an active area of research

It is the definition of “convolution” that sets GNNs apart

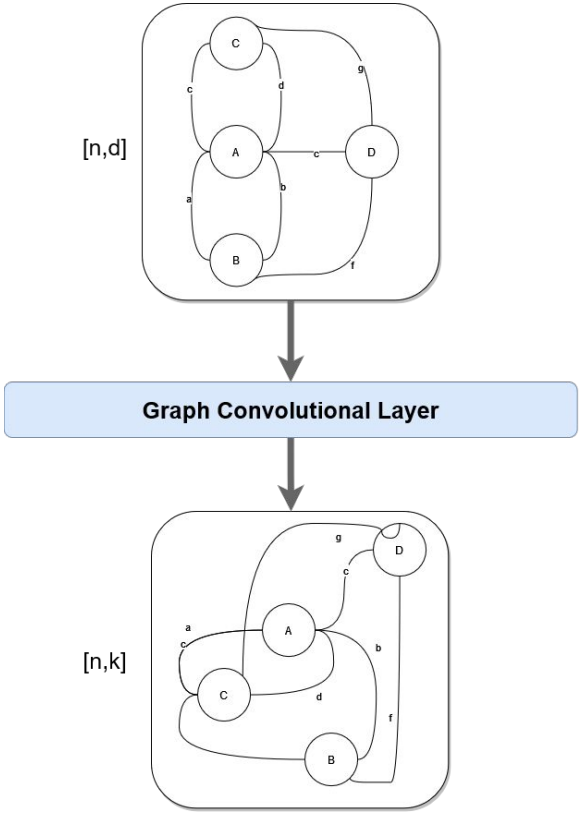


Illustration of a GCN layer outputting an embedded graph

Graph Neural Networks

The two different camps in Graph Theory leads to two competing methodologies for GNNs:

Spatial GNNs

Uses local structure to infer, classify or feature extract. Uses message passing to update node features iteratively. Has a wide range of applications.
By far the most common.

Spectral GNNs

Decomposes Laplacian into Fourier modes; the graph is often treated as one signal. All node features updated at once. Narrower range of applications.

It's an active area of research to map advances in one camp to the other, and some progress has been made *.

*** see this review from 2020.**

Graph Neural Networks

Learning tasks are generally

graph-level

Maps graph to a single collection of targets

$$\mathrm{GNN}(g) : [n, d] \longrightarrow [1, k]$$

(Classical regression, classification)

node-level

predictions are produced for each node in the graph

$$\mathrm{GNN}(g) : [n, d] \longrightarrow [n, j]$$

(node classification & regression, link prediction*)

n : number of nodes
d : node feature dimensions

*depends on method

Spatial GCNs (Message passing)

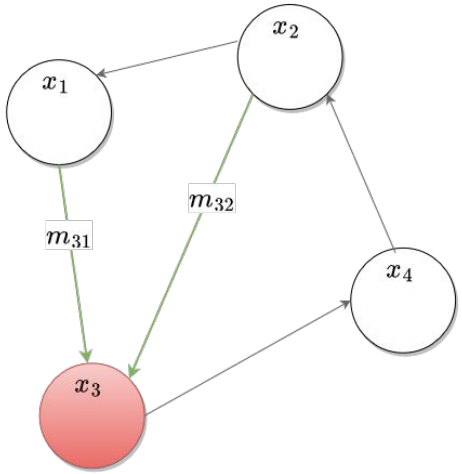
Spatial GNNs capture local information

"Neighboring nodes" contribute to the convolution

Cannot gather information outside of a certain neighbourhood size

Edges determine the flow of messages

A learned function is applied to transform node feature pairs [target, source] into messages. **Weights are shared.**



$$m_{ij} = f_{learned}(x_i, x_j)$$

$$\tilde{x}_i = Aggr(m_{i1}, m_{i2}, .. m_{in})$$

Typical message-passing operation on a source node.

Graph Attention Networks (GATConv)

Adds self-attention to message-passing, presented in *Graph Attention Networks* (2018)

An array of node features

$$\mathbf{h} = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}$$

Is passed through an attention mechanism

$$a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$$

Which computes attention messages

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$$ (masked attention)

That are converted to attention coefficients via

$$\alpha_{ij} = \mathrm{softmax}_j(e_{ij})$$

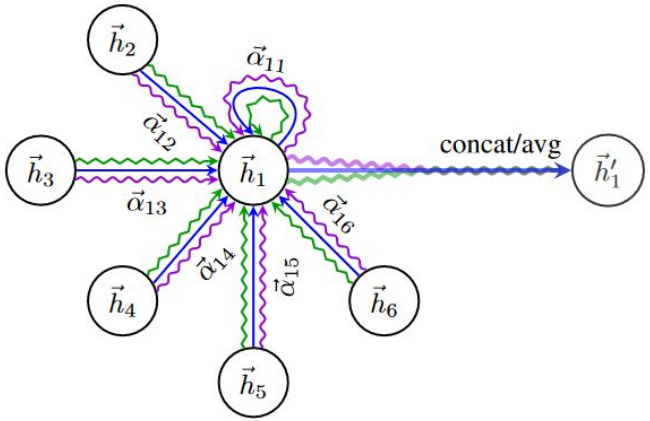


Illustration of multihead (n = 3) attention coefficients used in message passing. From *Graph Attention Networks* (2018)

$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\vec{h}_j\right)$$

Spatial GCNs (Message passing)

Example: EdgeConv

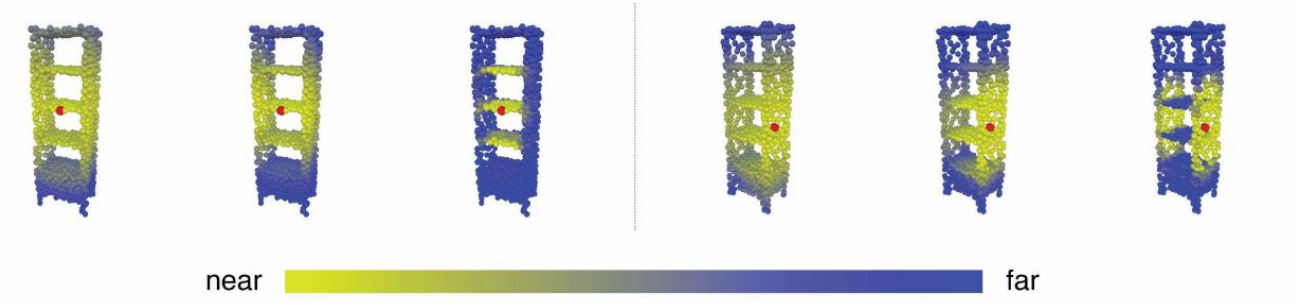
A convolutional operator on graphs designed for segmentation analysis of 3D point clouds. (**Geometric learning**)

Introduced in *Dynamic Graph CNN for Learning on Point Clouds (2019)*

EdgeConv

Goal:
Cluster nodes together into neighbourhoods that are related - “segments”.

Method:
Represent point clouds as graphs.
Interpret the convolution as a spatial transform in the latent space, and re-compute the edges based on new, latent positions.



From *Dynamic Graph CNN for Learning on Point Clouds*

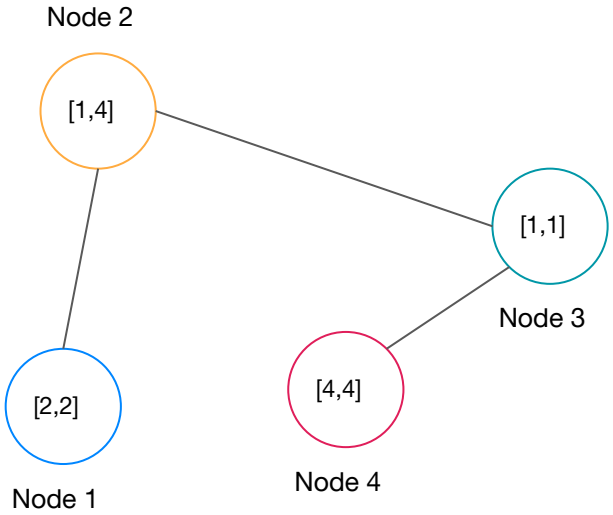
EdgeConv

EdgeConv ‘convolutes’ the graph by updating the values in each node in the graph by considering the values in the nodes that it is connected to.

The update of values of the j’th node is done via

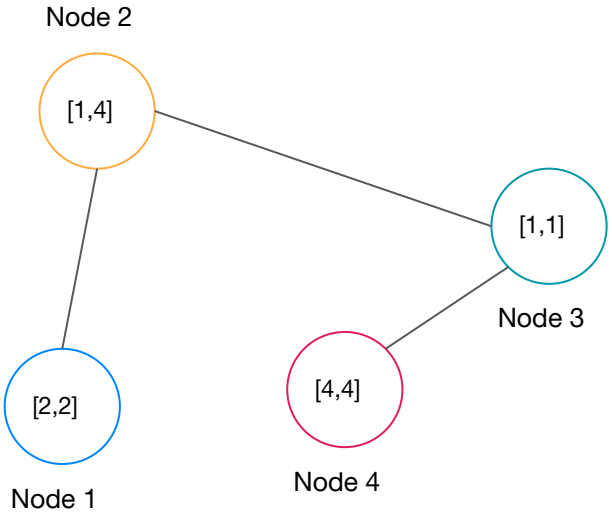
$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j)$$

Where f is a learned function (a neural net)



EdgeConv

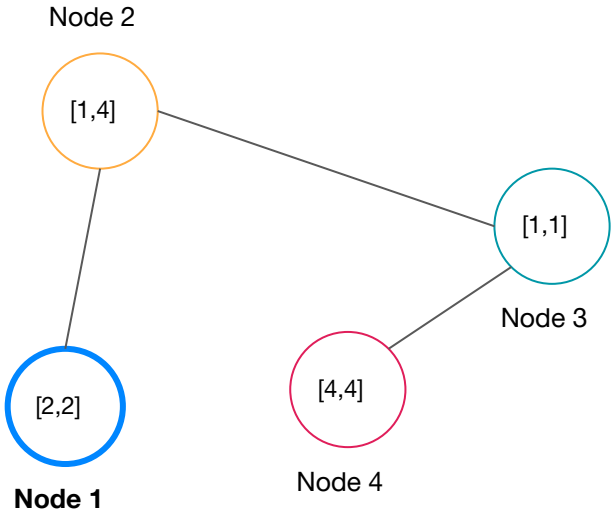
$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0$$ for simplicity



EdgeConv

$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0 \quad \text{for simplicity}$$

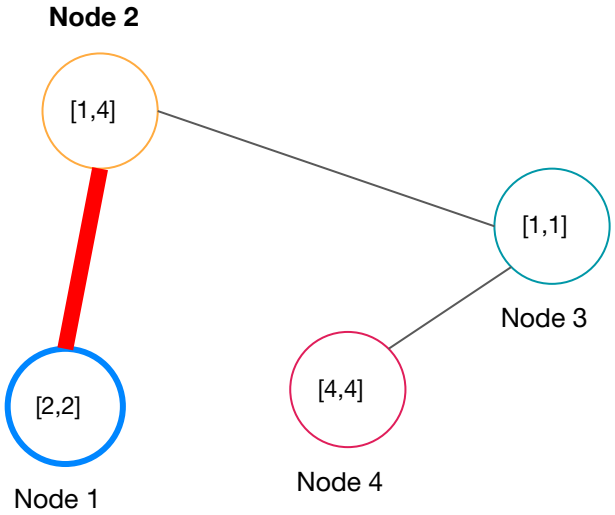
$\boxed{\tilde{x}_1} =$



EdgeConv

$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0 \quad \text{for simplicity}$$

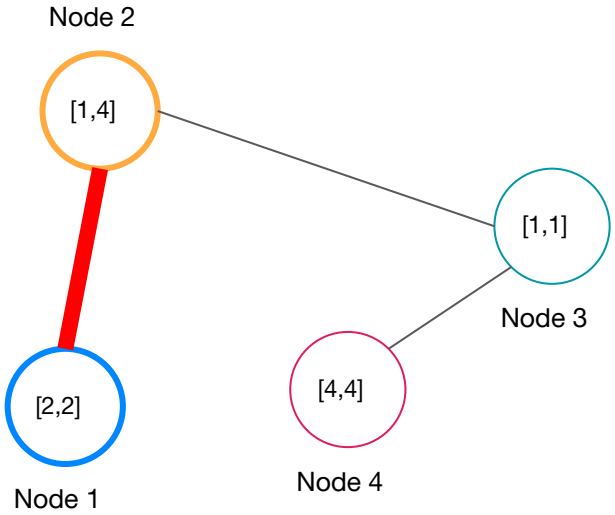
$$\tilde{x}_1 = f(x_1, x_1 - x_2)_2 =$$



EdgeConv

$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0 \quad \text{for simplicity}$$

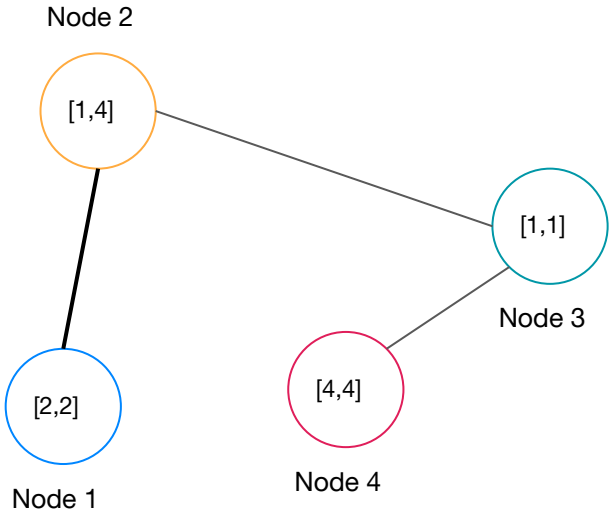
$$\tilde{x}_1 = f(x_1, x_1 - x_2)_2 = f([2,2], [2,2] - [1,4])_2 = [2,2,1,-2]$$



EdgeConv

$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0$$ for simplicity

$$\tilde{x}_1 = [2, 2, 1, -2]$$

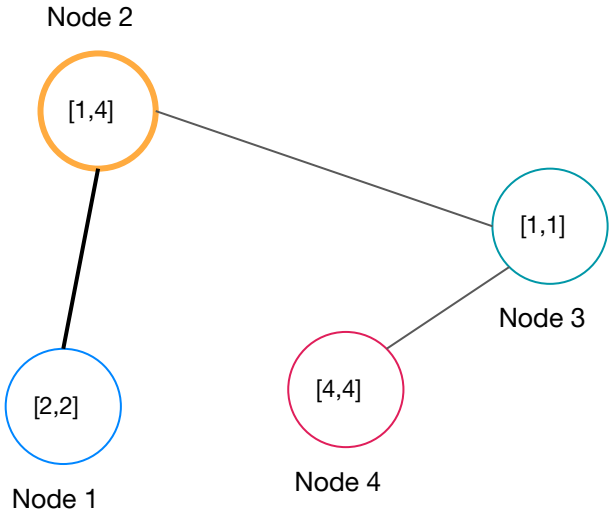


EdgeConv

$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0$$ for simplicity

$\tilde{x}_1 = [2, 2, 1, -2]$

$\tilde{x}_2 =$

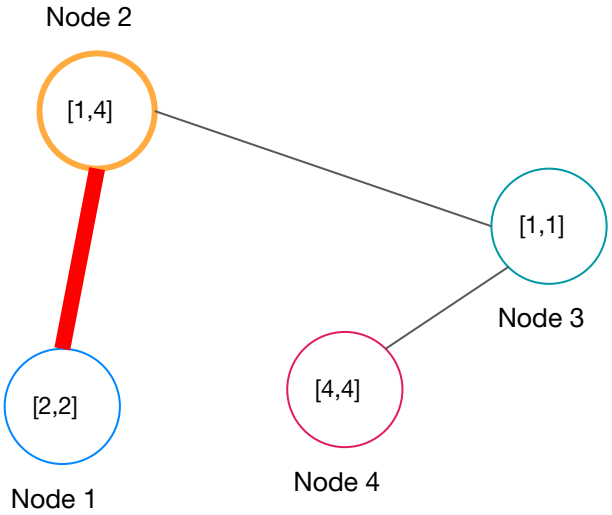


EdgeConv

$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0$$ for simplicity

$$\tilde{x}_1 = [2, 2, 1, -2]$$

$$\tilde{x}_2 = f(x_2, x_2 - x_1)_1 +$$

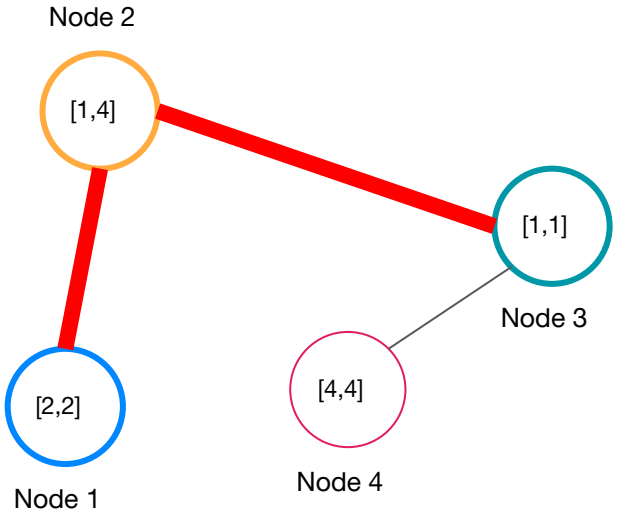


EdgeConv

$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0 \quad \text{for simplicity}$$

$$\tilde{x}_1 = [2, 2, 1, -2]$$

$$\tilde{x}_2 = f(x_2, x_2 - x_1)_1 + f(x_2, x_2 - x_3)_3$$

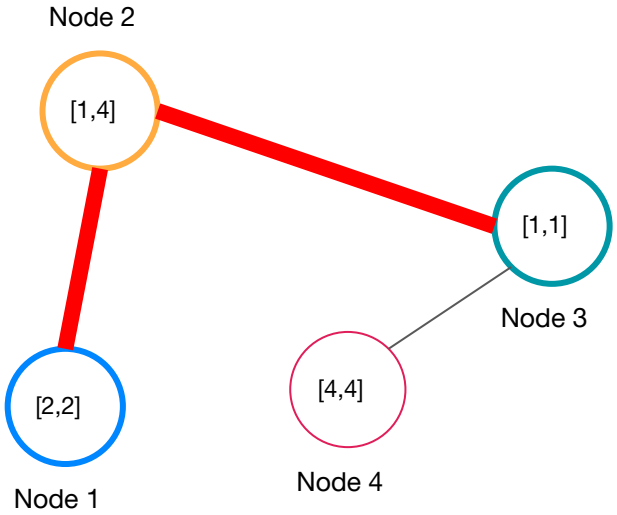


EdgeConv

$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0 \quad \text{for simplicity}$$

$$\tilde{x}_1 = [2, 2, 1, -2]$$

$$\begin{aligned}\tilde{x}_2 &= f(x_2, x_2 - x_1)_1 + f(x_2, x_2 - x_3)_3 \\ &= f([1,4], [1,4] - [2,2])_1 + f([1,4], [1,4] - [1,1])_3 \\ &= [1, 4, -1, 2] + [1, 4, 0, 3] \\ &= [2, 8, -1, 5]\end{aligned}$$



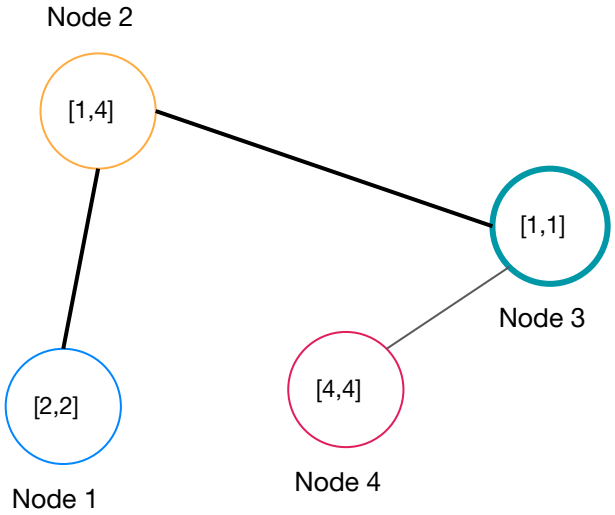
EdgeConv

$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0 \quad \text{for simplicity}$$

$\tilde{x}_1 = [2, 2, 1, -2]$

$\tilde{x}_2 = [2, 8, -1, 5]$

$\boxed{\tilde{x}_3} =$



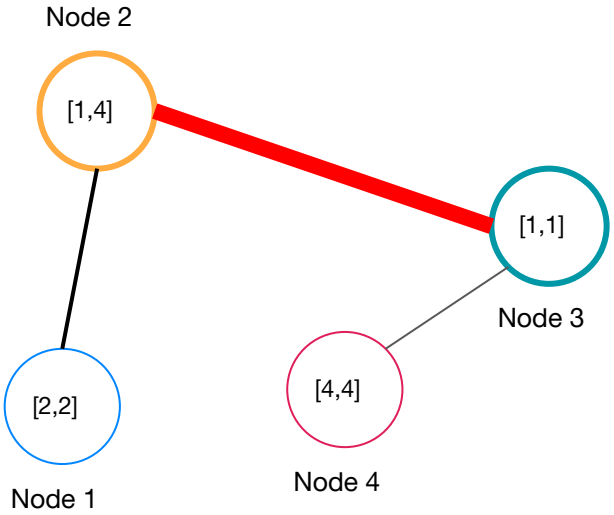
EdgeConv

$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0 \quad \text{for simplicity}$$

$$\tilde{x}_1 = [2, 2, 1, -2]$$

$$\tilde{x}_2 = [2, 8, -1, 5]$$

$$\tilde{x}_3 = f(x_3, x_3 - x_2)_2 +$$



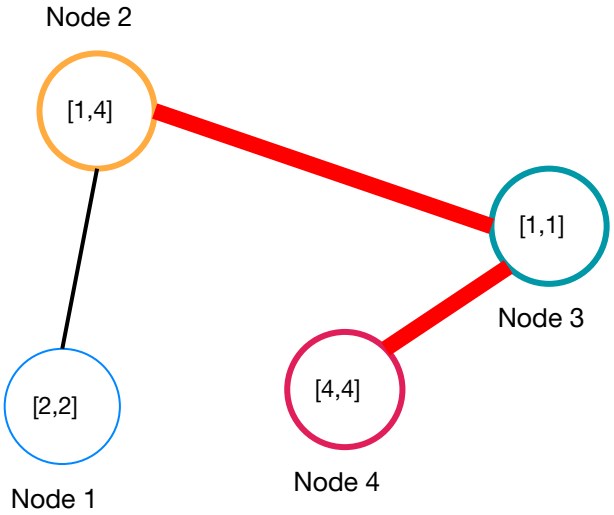
EdgeConv

$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0 \quad \text{for simplicity}$$

$$\tilde{x}_1 = [2, 2, 1, -2]$$

$$\tilde{x}_2 = [2, 8, -1, 5]$$

$$\tilde{x}_3 = f(x_3, x_3 - x_2)_2 + f(x_3, x_3 - x_4)_4$$



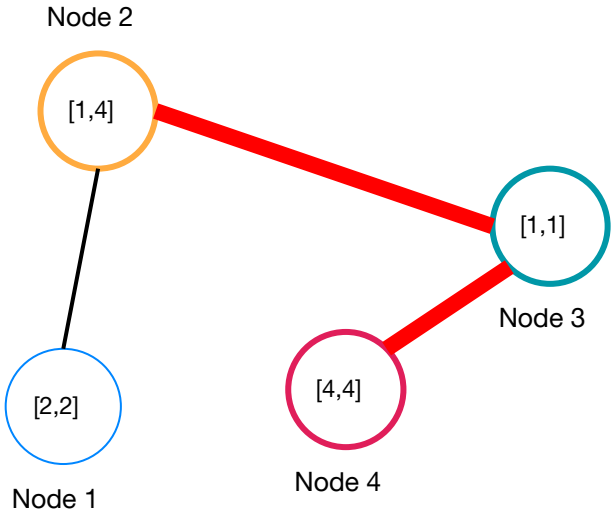
EdgeConv

$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0 \quad \text{for simplicity}$$

$$\tilde{x}_1 = [2, 2, 1, -2]$$

$$\tilde{x}_2 = [2, 8, -1, 5]$$

$$\begin{aligned} \tilde{x}_3 &= f(x_3, x_3 - x_2)_2 + f(x_3, x_3 - x_4)_4 \\ &= f([1,1], [1,1] - [1,4])_2 + f([1,1], [1,1] - [4,4])_3 \\ &= [1, 1, 0, -3] + [1, 1, -3, -3] \\ &= [2, 2, -3, -6] \end{aligned}$$



EdgeConv

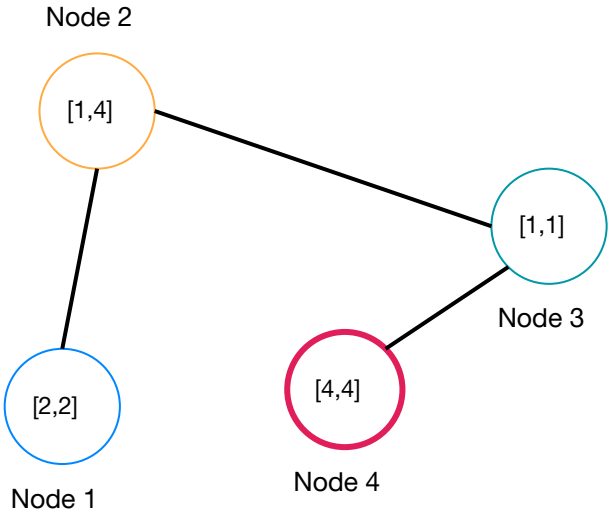
$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0 \quad \text{for simplicity}$$

$$\tilde{x}_1 = [2, 2, 1, -2]$$

$$\tilde{x}_2 = [2, 8, -1, 5]$$

$$\tilde{x}_3 = [2, 2, -3, -6]$$

$$\boxed{\tilde{x}_4} =$$



EdgeConv

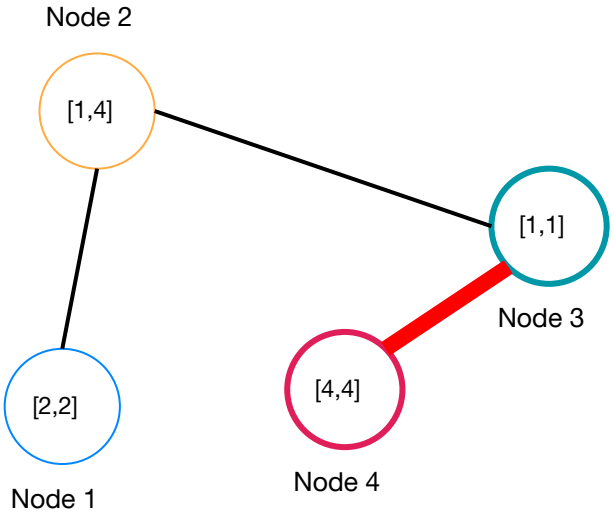
$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0 \quad \text{for simplicity}$$

$$\tilde{x}_1 = [2, 2, 1, -2]$$

$$\tilde{x}_2 = [2, 8, -1, 5]$$

$$\tilde{x}_3 = [2, 2, -3, -6]$$

$$\tilde{x}_4 = f(x_4, x_4 - x_3)_3$$



EdgeConv

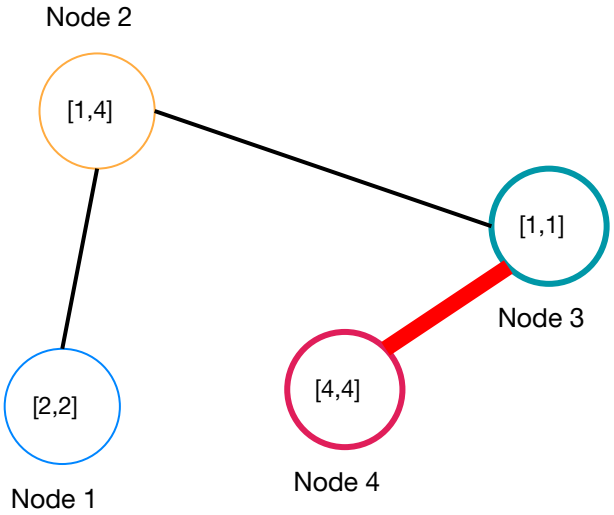
$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0 \quad \text{for simplicity}$$

$$\tilde{x}_1 = [2, 2, 1, -2]$$

$$\tilde{x}_2 = [2, 8, -1, 5]$$

$$\tilde{x}_3 = [2, 2, -3, -6]$$

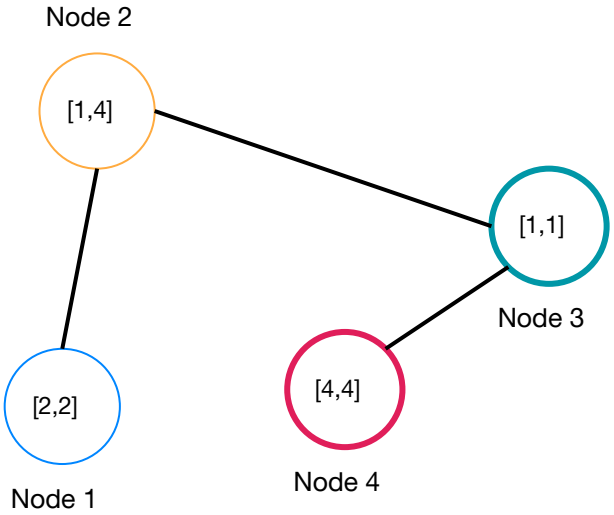
$$\tilde{x}_4 = f(x_4, x_4 - x_3)_3 = f([4,4], [4,4] - [1,1])_2 = [4, 4, 3, 3]$$



EdgeConv

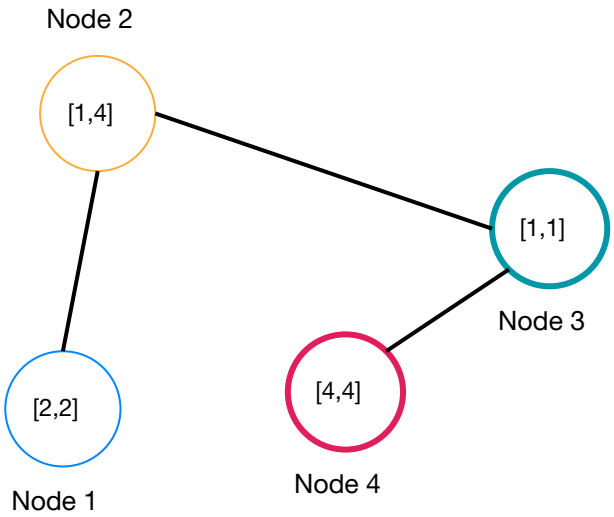
$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0 \quad \text{for simplicity}$$

$$\tilde{x}_1 = [2, 2, 1, -2]$$
$$\tilde{x}_2 = [2, 8, -1, 5]$$
$$\tilde{x}_3 = [2, 2, -3, -6]$$
$$\tilde{x}_4 = [4, 4, 3, 3]$$



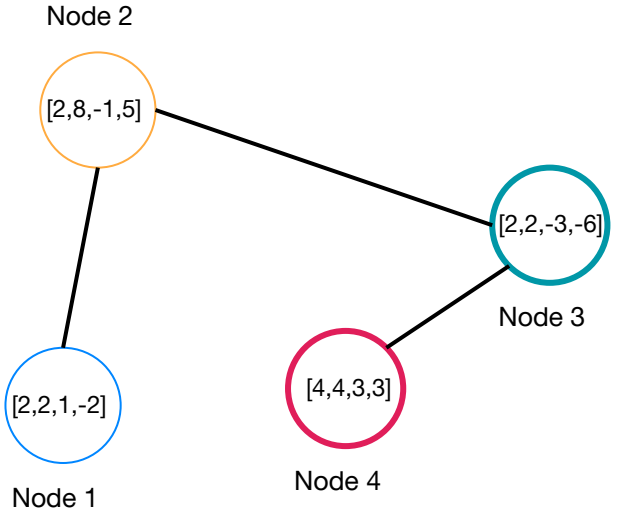
EdgeConv

$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0 \quad \text{for simplicity}$$



Input graph

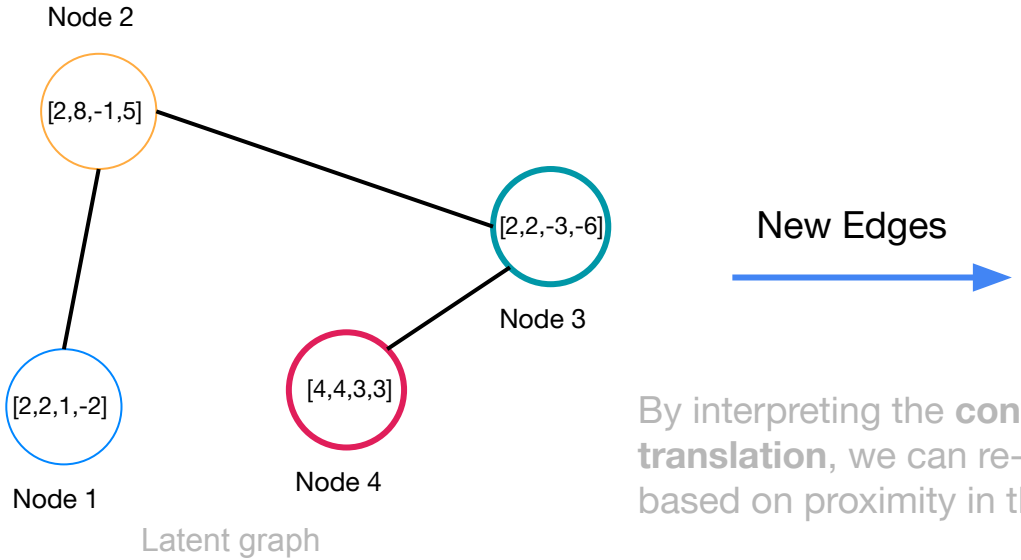
EdgeConv



Latent graph

EdgeConv

$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0 \quad \text{for simplicity}$$

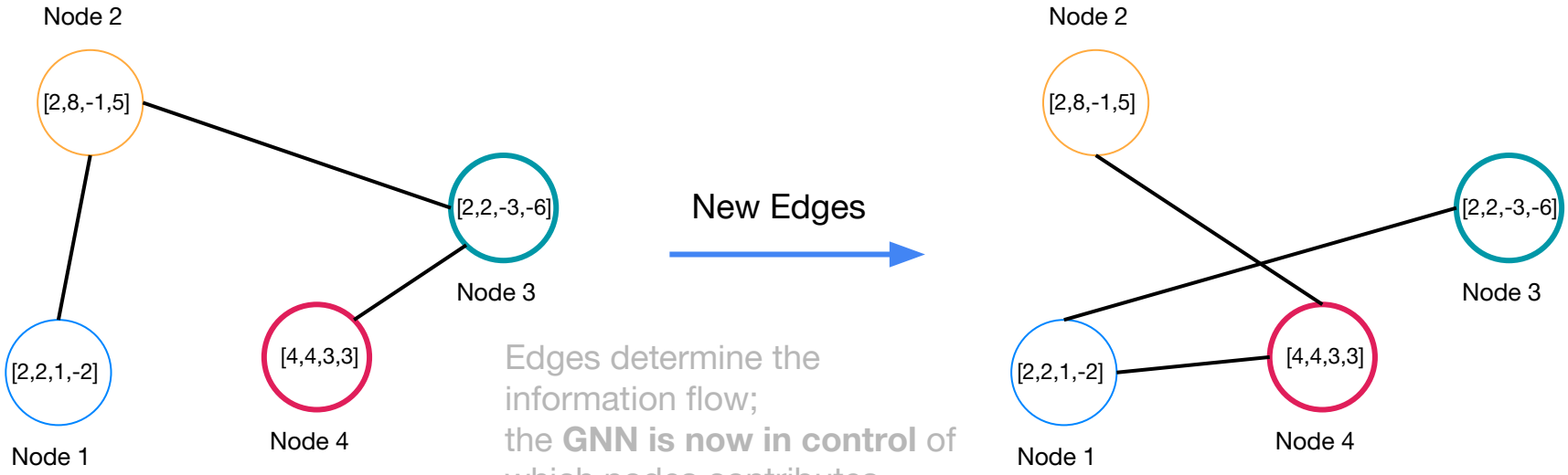


Latent graph

By interpreting the **convolution as a translation**, we can re-assign edges based on proximity in the latent space

EdgeConv

$$\tilde{x}_i = \sum_{k=1}^{n_{neighbours}} f(x_i, x_i - x_j) \quad , \; f(x) = 1 \cdot x + 0 \quad \text{for simplicity}$$



Edges determine the information flow;
the **GNN is now in control** of which nodes contributes

EdgeConv

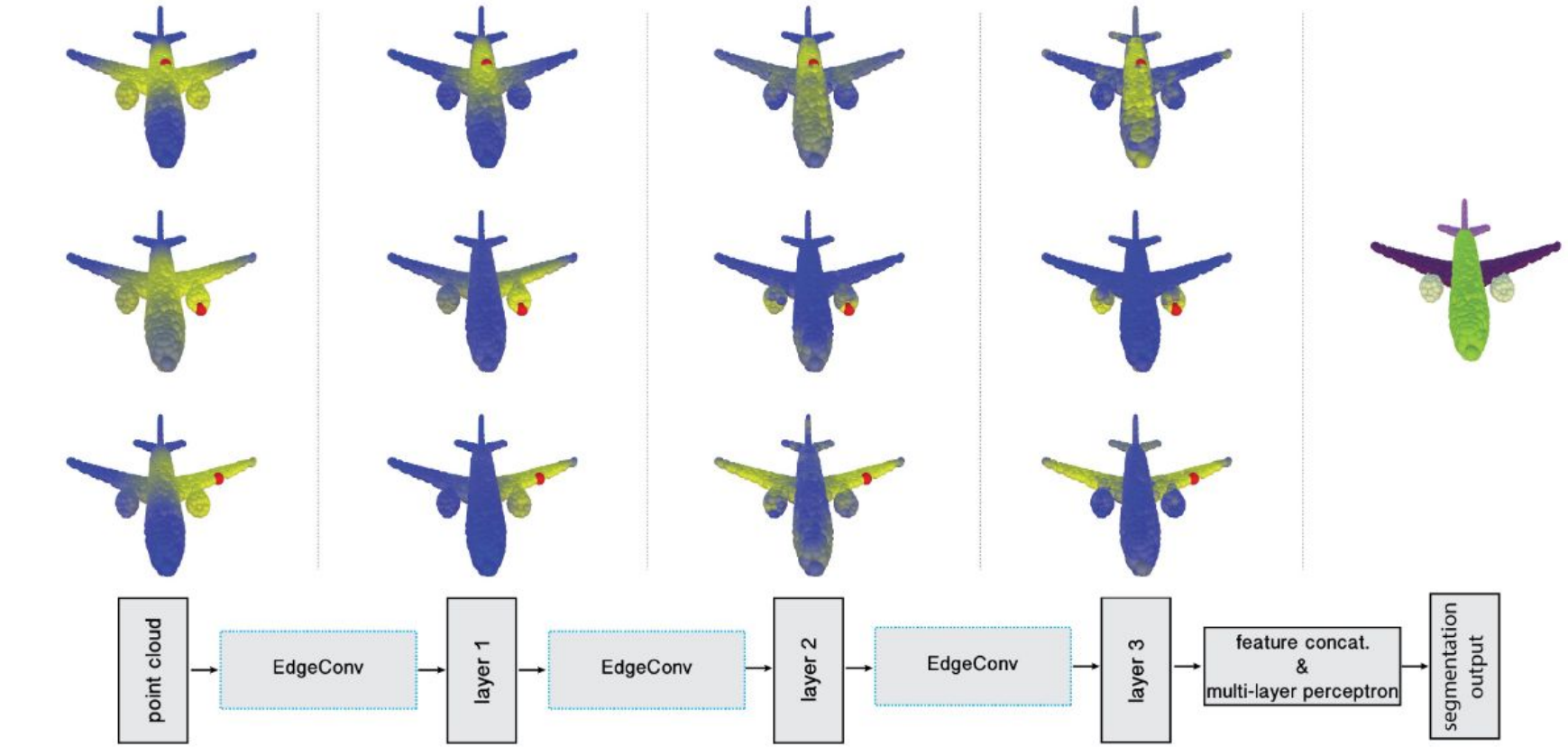
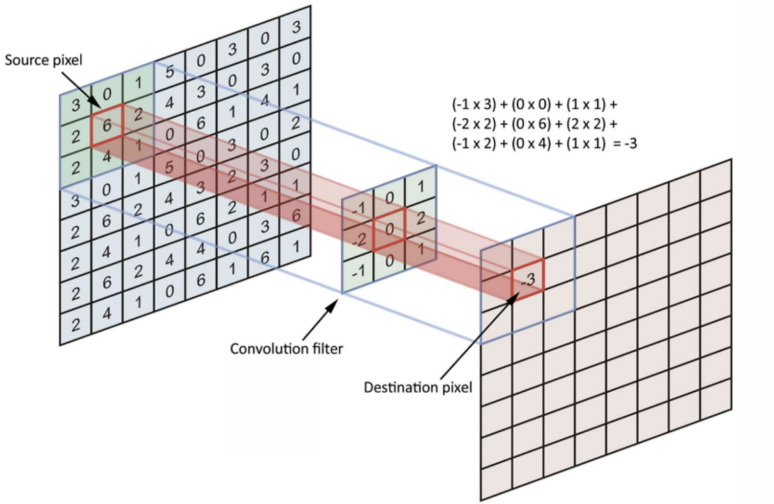
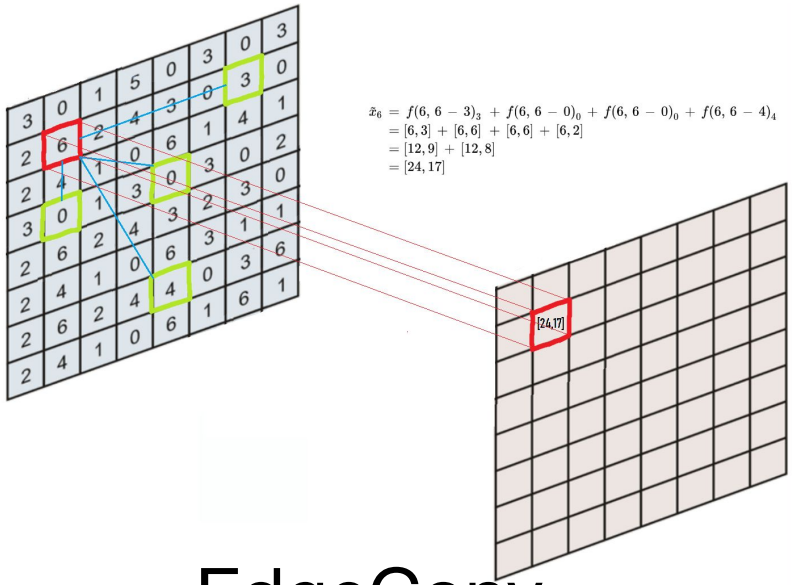


Figure from original paper

CNN Convolutions vs. EdgeConv



vs.



CNN Convolutions

EdgeConv

Examples of applications in Physics

Learning Feynman Diagrams using Graph Neural Networks

Learning Feynman Diagrams using Graph Neural Networks

Nodes : interaction vertices
Features - not clear

Edges : particles
Features - particle data

Target : matrix elements averaged over helicity

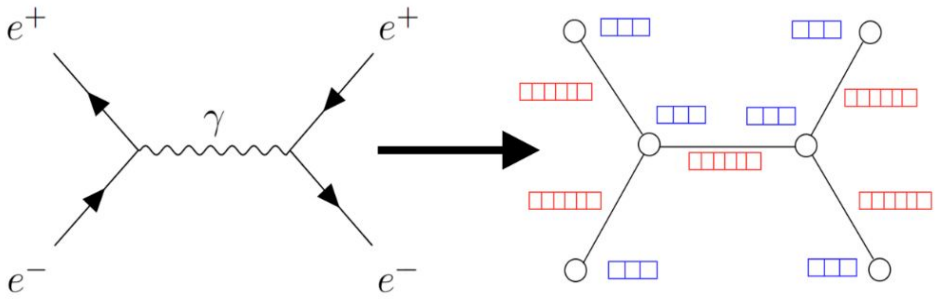


Figure 4: Encoding a Feynman diagram as a graph. The blue vectors are the node features and the red are the edge features.

Borrowed from *Learning Feynman Diagrams using Graph Neural Networks*

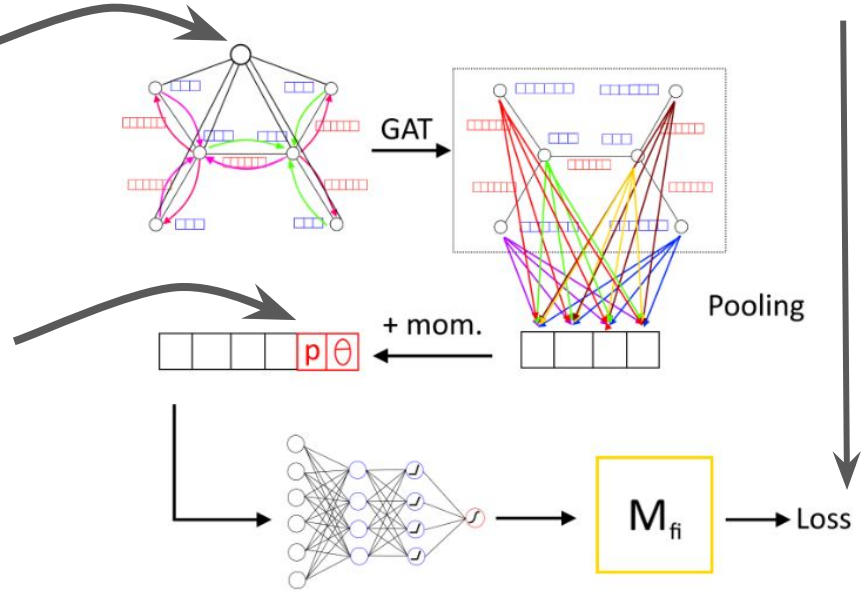
Each training example is parameterized with (polar angle, momentum). Momentum is in the range [1, 1000] GeV.

Learning Feynman Diagrams using Graph Neural Networks

L1 Loss

"Virtual node" connected to all other nodes. Contains interaction strength of QCD, QED and weak force. Improved performance.

Momentum is concatenated to learned graph representation



(b) Top left image depicts the action of the GAT layer. Top right is a portrayal of the output of the GAT layers being pooled into a graph representation and momenta added. This is passed to an FCN to get the predictions.

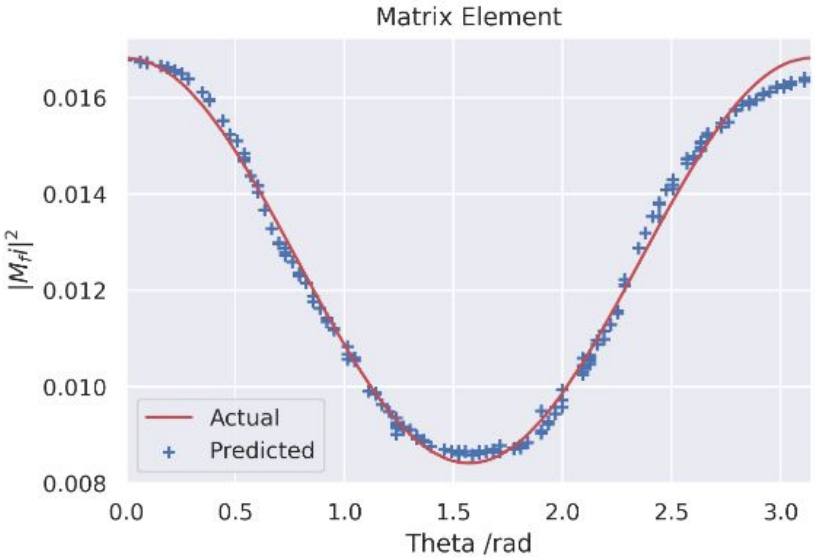
Graph-level task

Borrowed from [Learning Feynman Diagrams using Graph Neural Networks]

Learning Feynman Diagrams using Graph Neural Networks

A strong proof-of-concept

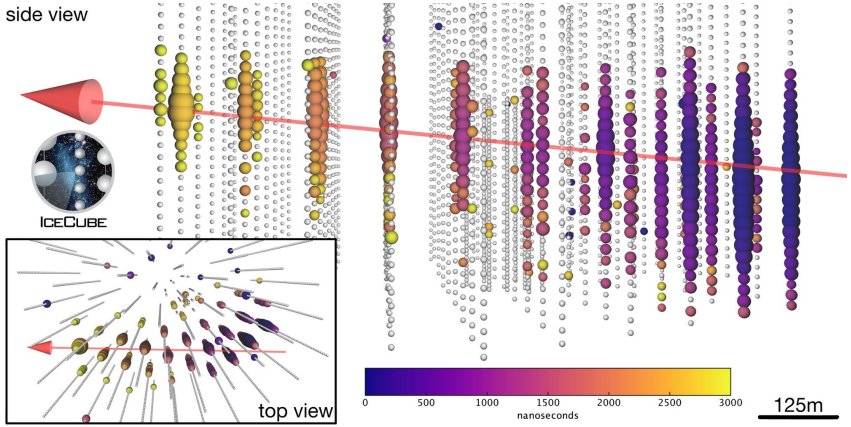
Accurate up to first or second decimal point



Test metric	Data
Test accuracy (1d.p.)	99.00%
Test accuracy (2d.p.)	81.50%
Test accuracy (3d.p.)	11.50%
Test L1 Loss	0.0049

Borrowed from *Learning Feynman Diagrams using Graph Neural Networks*

Node Classification for noise rejection in IceCube Neutrino Observatory

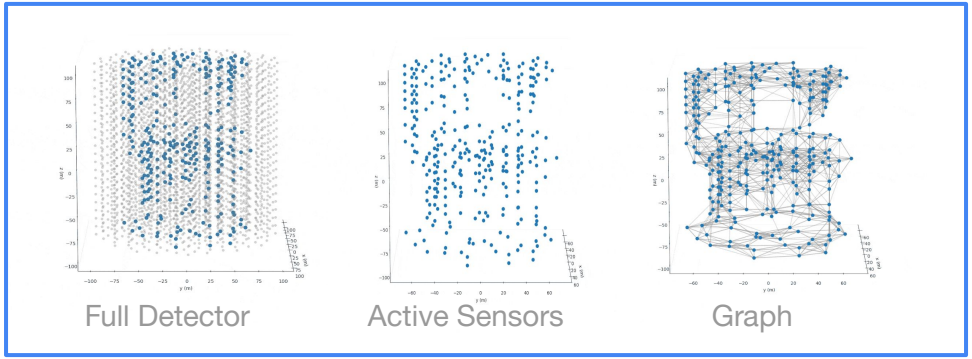


Node Classification in IceCube

Data from neutrino telescopes are ***geometric time series.***

An event is a series of measurements of light in an interaction window by sensors placed in the ice. Each sensor often can measure light (**so-called pulse**) multiple times in the same interaction window.

1. For each sensor we associate a time series.
2. Each sensor has a unique position in the ice
3. The placement of sensors is irregular

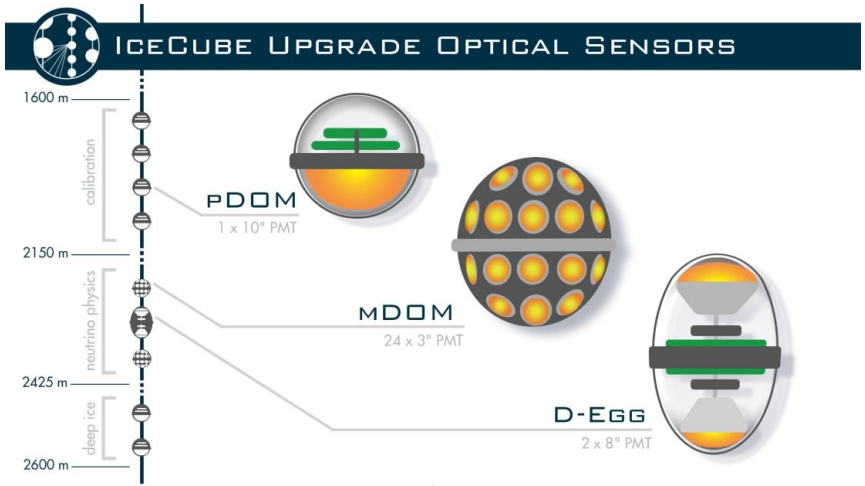


Mock simulation of a neutrino telescope. Illustrates the associated graph

Node Classification in IceCube

“The goal of IceCube Upgrade is to provide world-leading sensitivity to neutrino oscillations and to take unique measurements of tau neutrino appearance with high precision. It also serves as a R&D platform for the future IceCube-Gen2 experiment.”

Physics Potential of the IceCube Upgrade



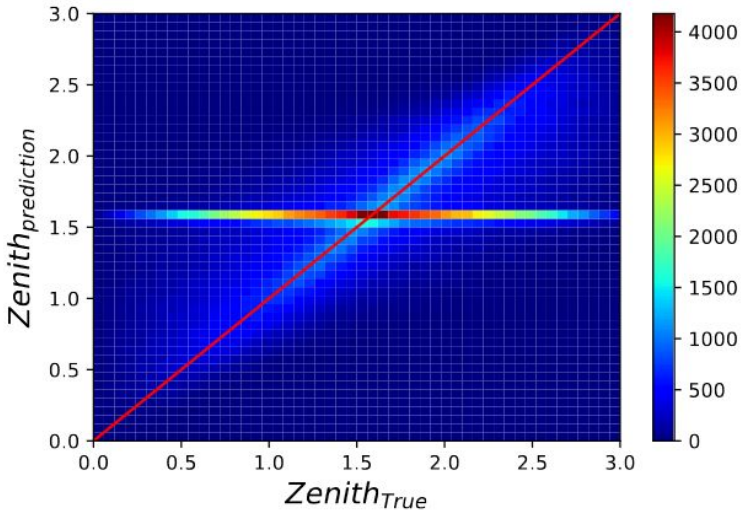
IceCube Press Release

Around 700 new modules

Node Classification in IceCube

On average, events have 70% noise, making them practically unreconstructable.

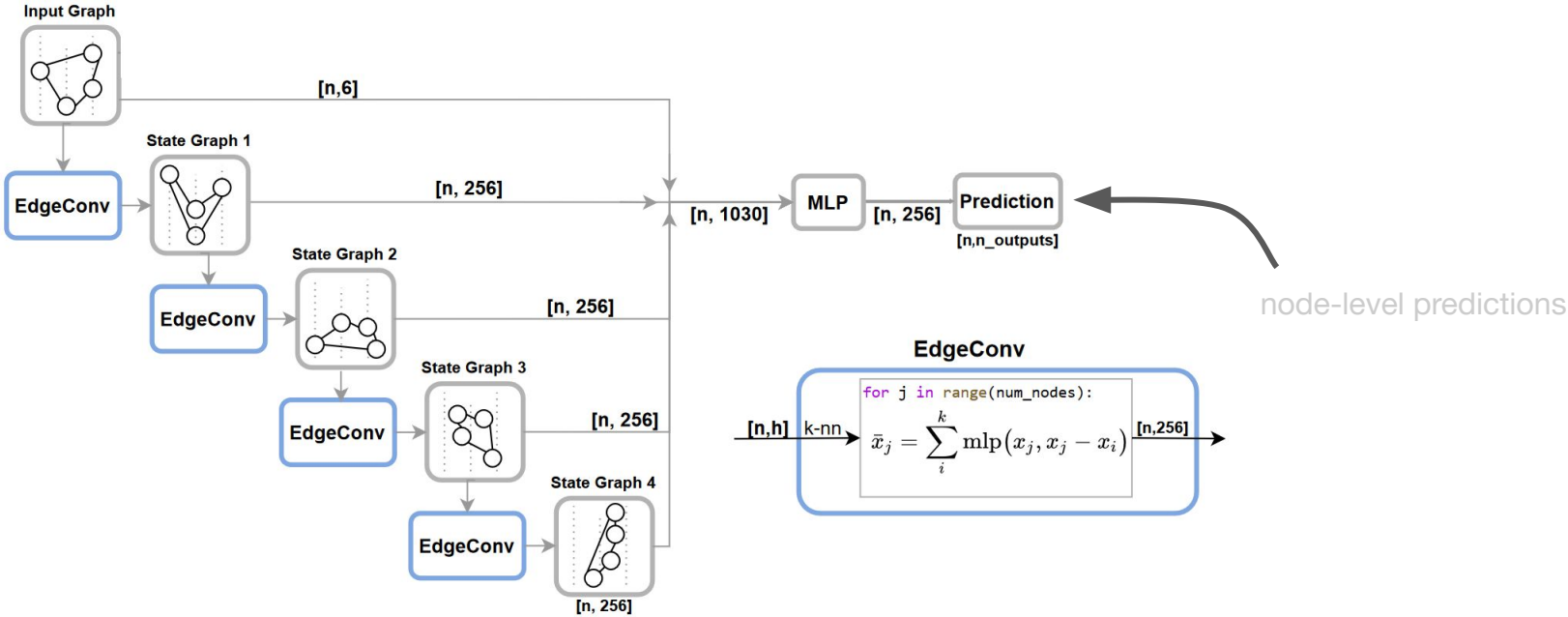
We phrased noise cleaning as a node classification problem, and used DynEdge to clean the events.



DynEdge trained on uncleaned events with an expected poor performance.

Node Classification in IceCube

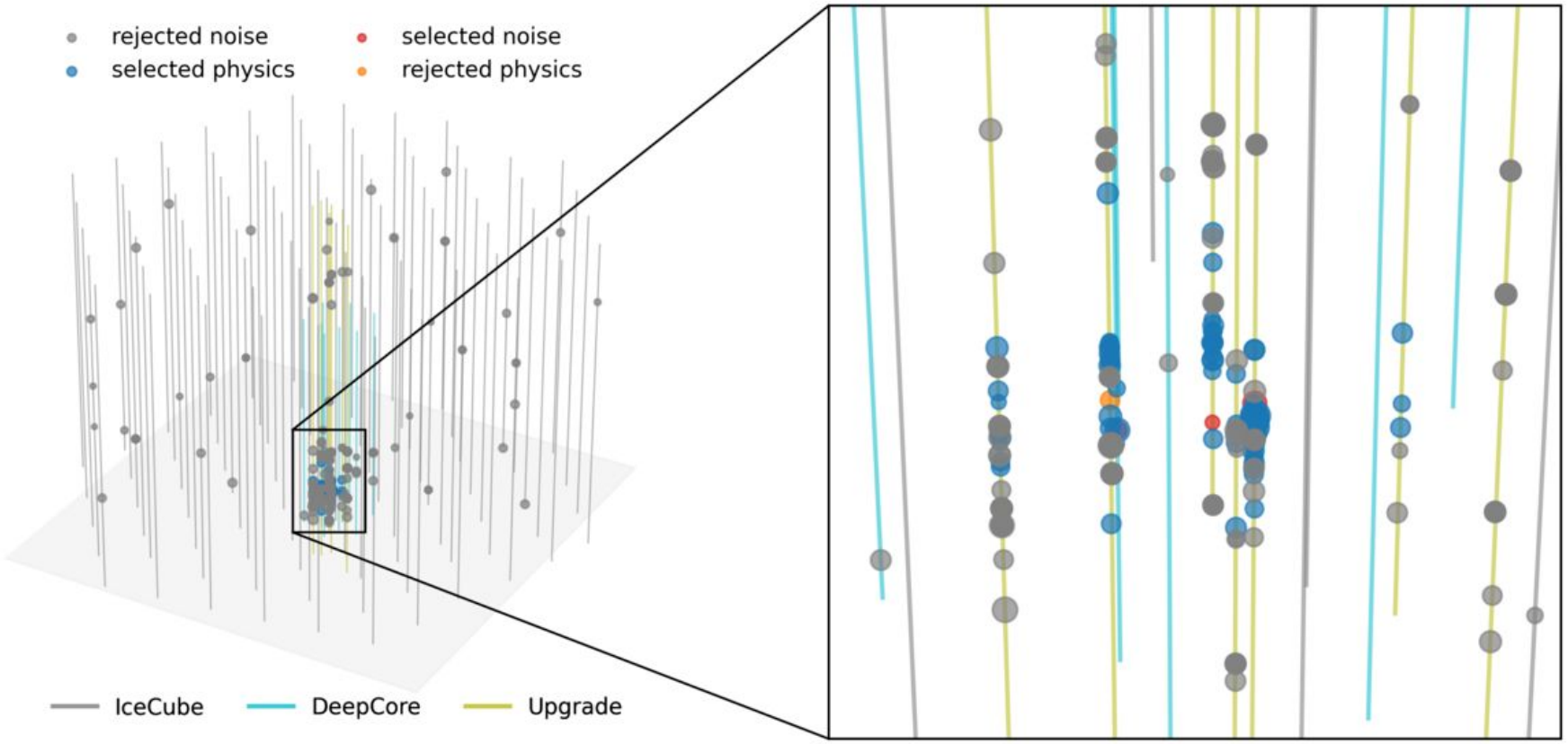
We removed the node-aggregation layers such that DynEdge outputs node-level predictions.



From ***Graph Neural Networks for low-energy event classification & reconstruction in IceCube***

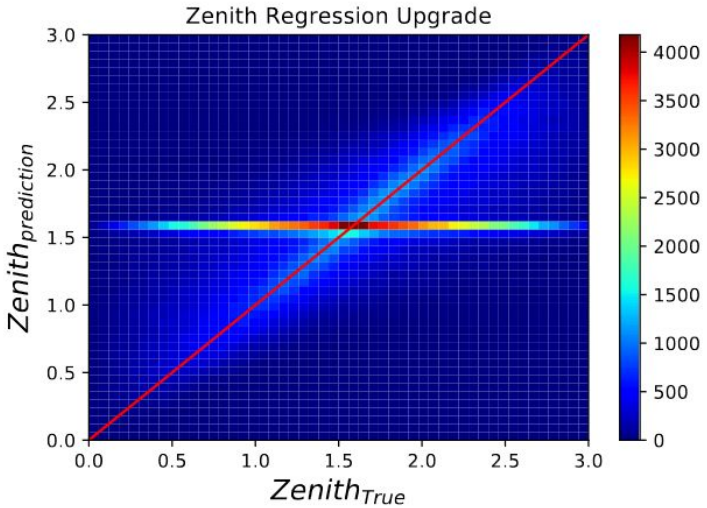
graphnet

Node Classification in IceCube

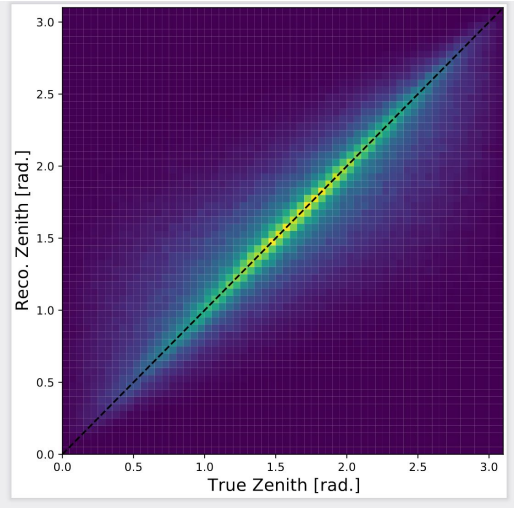


On average, events after cleaning contains 6.8% noise and retains 91% signal

Node Classification in IceCube



DynEdge trained on uncleaned events with an expected poor performance.



DynEdge trained on cleaned events

Other applications in physics

Vast majority of use cases of GNNs in physics is collider experiments and astro-particle physics. They often phrase their tasks as geometric learning problems.

You can find a dated (but good) review here

Thank you for listening!

GraphNeT

Graph Neural Networks for
Neutrino Telescope Event Reconstruction

icecube/graphnet

Exercises - GNN on MNIST

Showcases EdgeConv on graphs for image classification on the classic MNIST dataset.

Open this link to go to the exercises:

Notebook (start as GPU session)

Solution (If you get stuck and I'm busy)