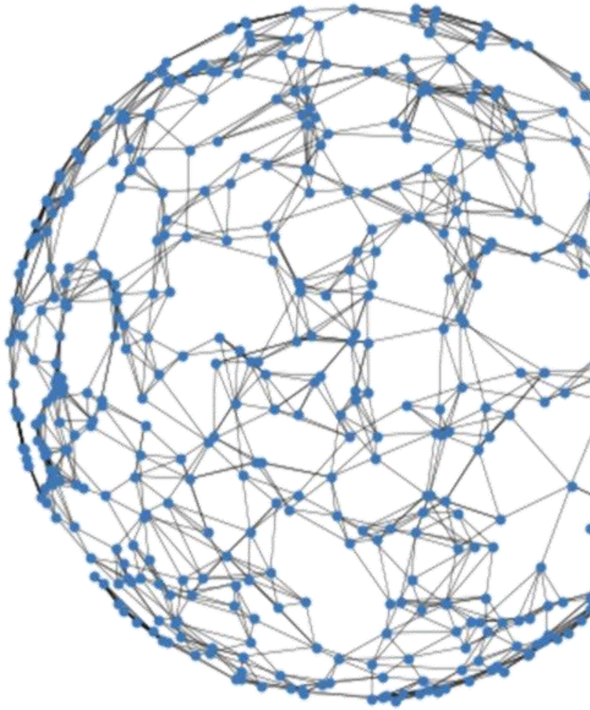
SPONSORED BY THE
Federal Ministry of Education and Research

III. Physikalisches Institut A | RWTH AACHEN UNIVERSITY

Hackathon: Graph Neural Networks

Conceptual Advances in Deep Learning for Research on Universe and Matter

ERUM DATA HUB

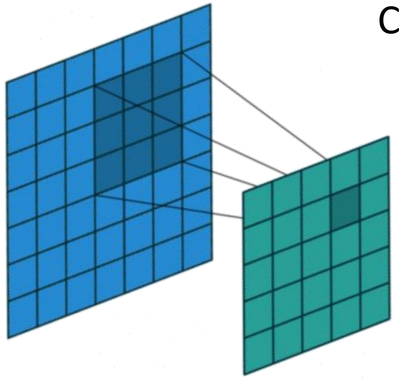


Niklas Langner
RWTH Aachen University

Introduction: Why use Graph Convolutions?

Convolutions are advantageous for tasks like pattern recognition as they **incorporate the structur of the data by design**

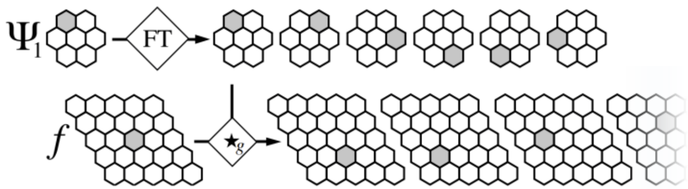
For example:



Convolutions on pixelized grids (e.g. images)

1)

Convolutions on hexagonal grids (e.g. Auger Surface Detector)



2)

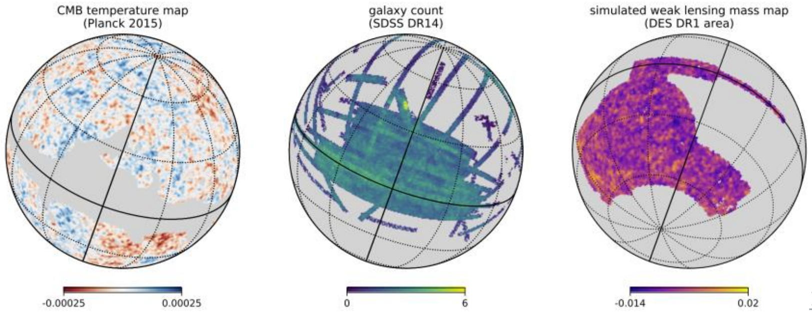
Problem: What if the structure is irregular, non-Euclidean or changes between measurements?

1) https://github.com/vdumoulin/conv_arithmetic

2) https://arxiv.org/abs/1803.02108v1

Examples of challenging data structures

1. **Spherical Data** (e.g. in HEALPix format)

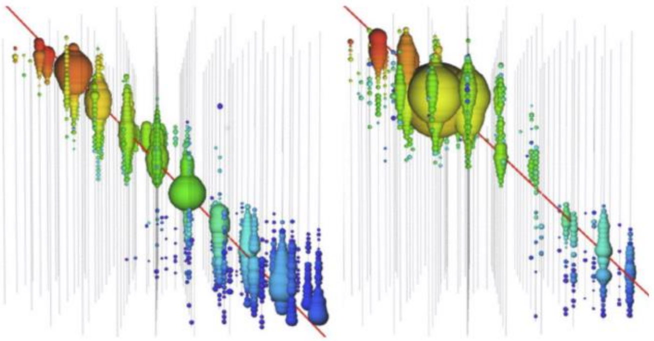


1)

Challenges:

- Convolutions should incorporate spherical shape
- Should be rotational invariant
- Using a 2D projection would lead to distortions

2. **Point Clouds** with fixed positions (e.g. detector hits)



2)

Challenges:

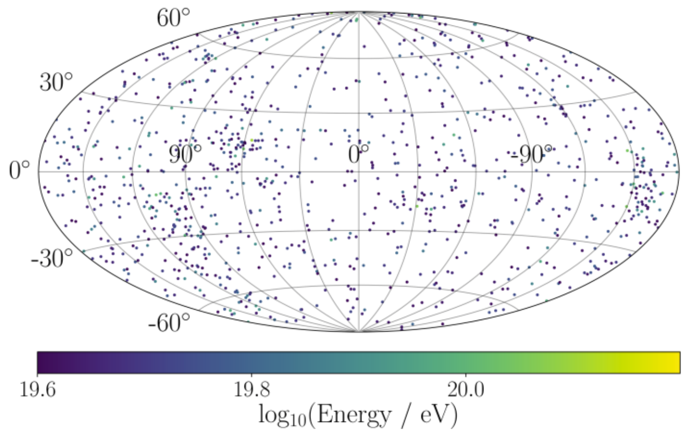
- Irregular geometries
- Sparsity

1) https://arxiv.org/abs/1810.12186

2) https://arxiv.org/abs/1809.06166

Examples of challenging data structures

3. Point clouds with **continuously distributed** positions (e.g. cosmic-ray arrival directions)



Challenges:

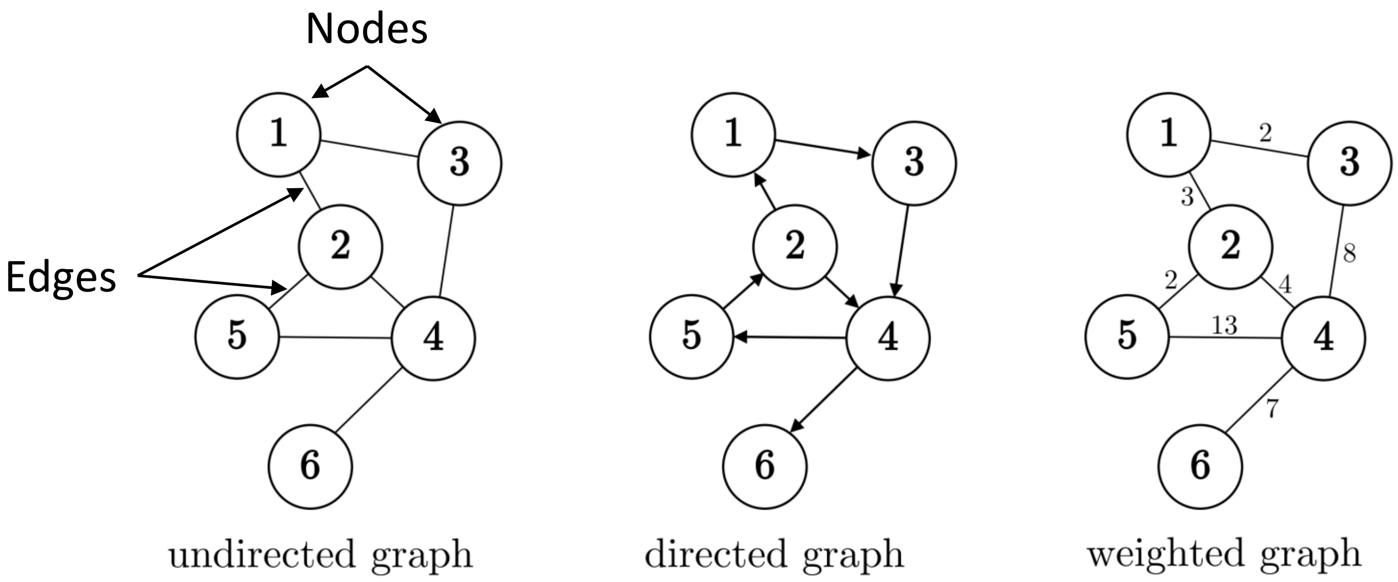
- No fixed positions / grid
- Positions change for each dataset
- Sparsity

4. …

Using **graph convolutions** can solve these problems!

What are Graphs?

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an ordered pair of **nodes** $\mathcal{V}$ and **edges** $\mathcal{E}$



Describing a graph:

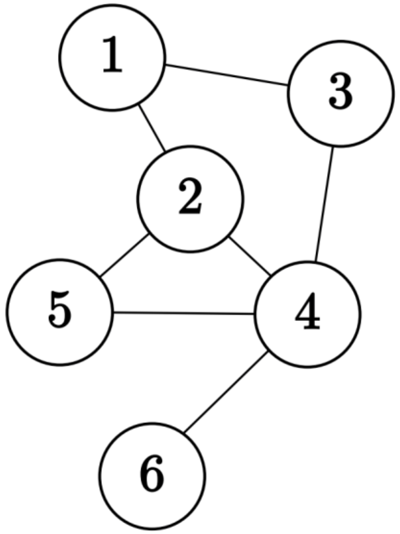
- Defined by connections / neighborhoods
- Changing order of nodes still describes the **same graph**
    - **Permutational invariance**

Describing a Graph: Adjacency Matrix A

Consider graph with N nodes

- **Adjacency matrix A** of shape $(N \times N)$ represents structure of the graph
- $A_{ij} = 1$ of node i is connected to node j and otherwise 0

Example



$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

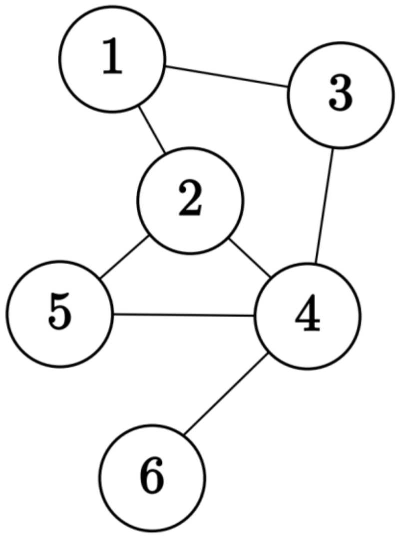
(For a weighted graph the entries are weighted accordingly.)

Describing a Graph: Degree Matrix $\boldsymbol{D}$

Consider graph with N nodes

- Diagonal **degree matrix $\boldsymbol{D}$** of shape $(N \times N)$
- Counts how often edges terminate at each node: $D_{ii} = \sum_i A_{ij}$

Example

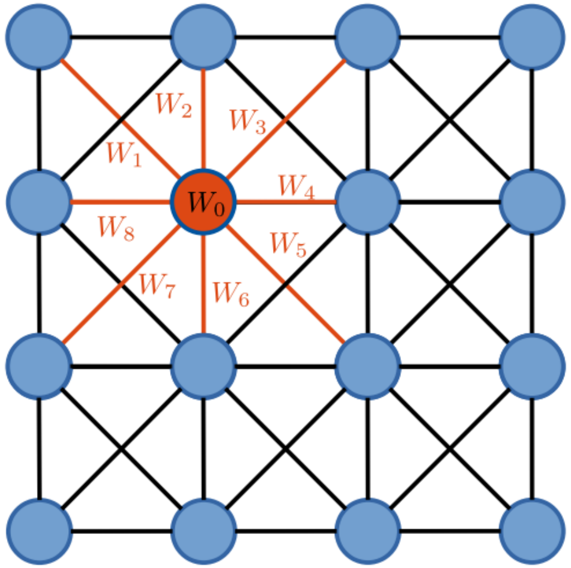


$$D = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

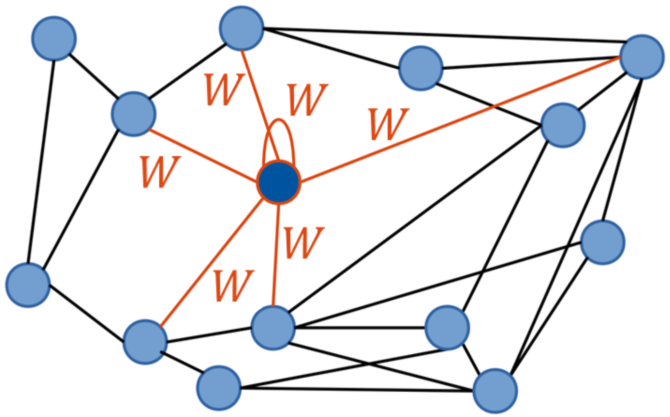
(For a weighted graph the entries are weighted accordingly.)

Graph Convolutions

Convolutions on pixelized grids

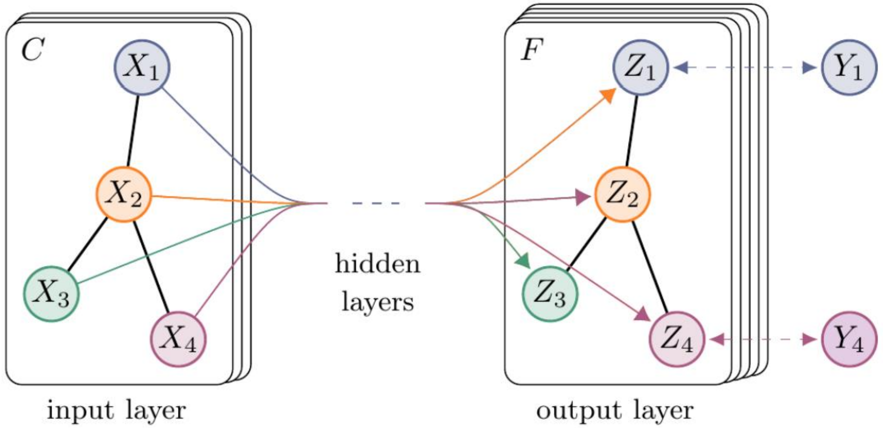


Convolutions on graphs



$\mathcal{N}_i$: Neighbors of node h_i

$$h_i^{l+1} = \sigma\left(h_i^l W^l + \underbrace{\sum_{j\in\mathcal{N}_i} \frac{1}{c_{ij}} h_j^l W^l}_{\text{Average over neighbors}}\right)$$



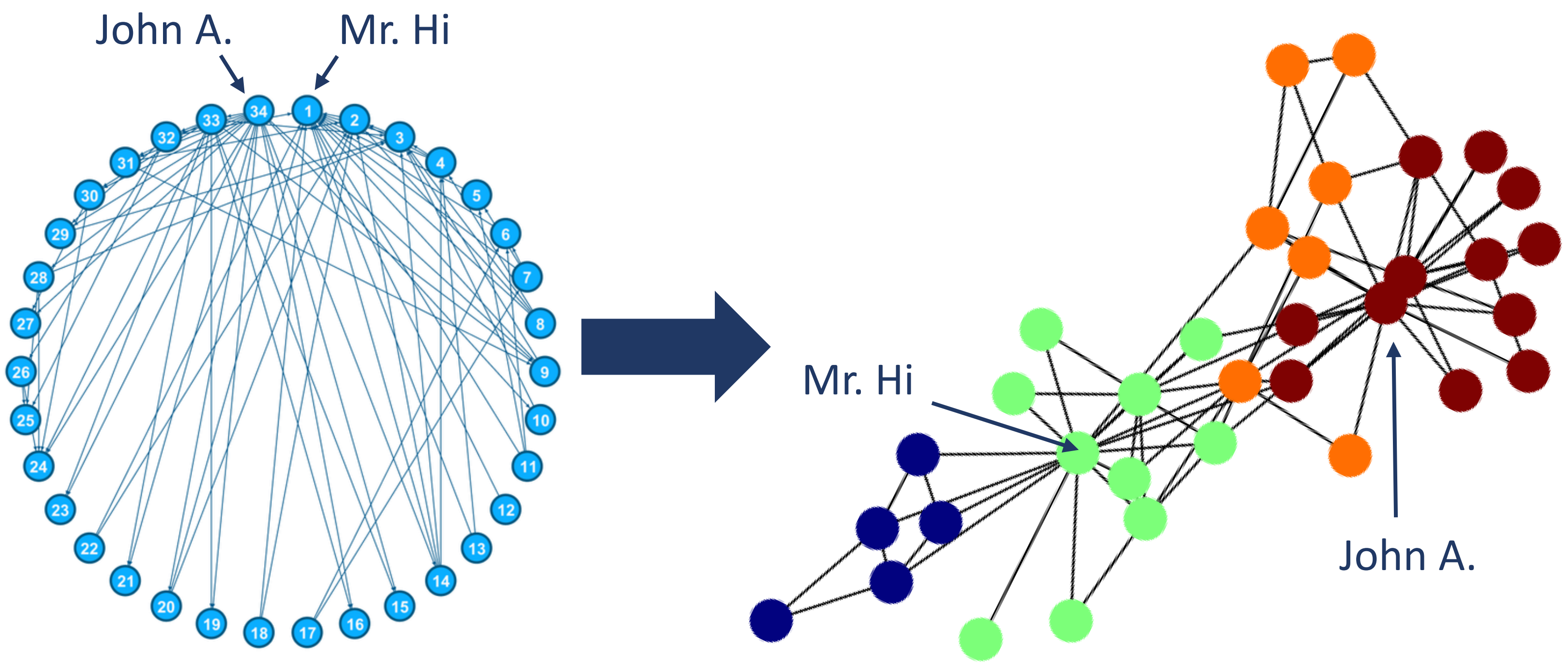
https://arxiv.org/abs/1609.02907

Images: Deep Learning in Physics Research, World Scientific

Example: Zachary’s Karate Club

- **Social network**: University karate club with **34 members**
- Key figures: Administrator “John A.” and instructor “Mr. Hi”
- Conflict between John A. and Mr. Hi **splits club into multiple groups**

  → Represent social network as graph and **classify groups using graph convolutions**



VISPA Example!

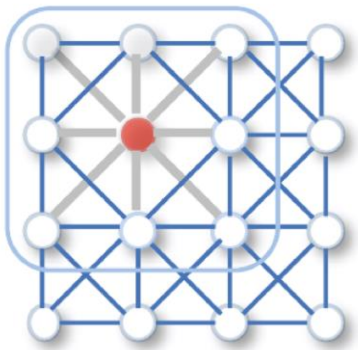
https://commons.wikimedia.org/wiki/File:Social_Network_Model_of_Relationships_in_the_Karate_Club.png

Graph Convolutions in the Spatial Domain

- **Nodes now have positions** that are considered, no longer only connections
- Analogous to classical CNN but in **continuous space**

Classical CNN

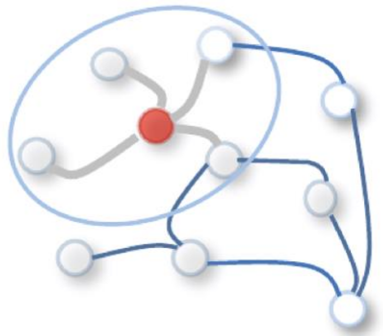
Discrete grid positions



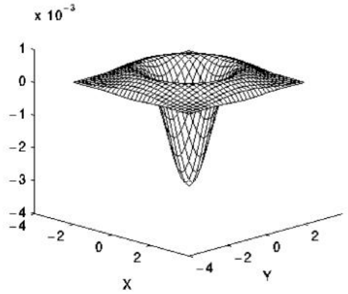
Discrete filter

$$\mathbf{D}^2_{xy} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Continuous Graph Convolution



Continuous grid positions



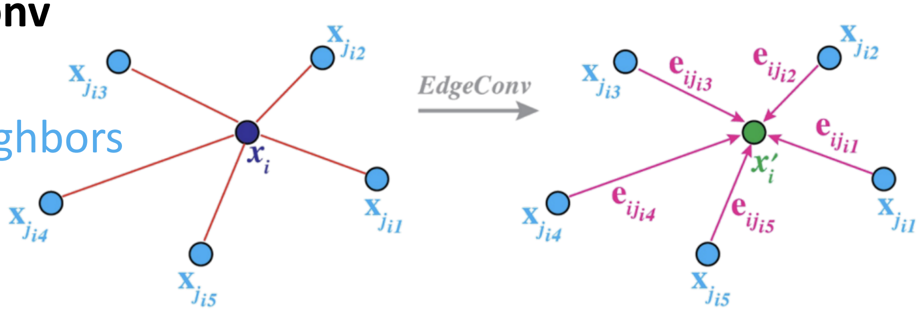
Continuous filter

https://arxiv.org/abs/1901.00596

EdgeConv[1] Operation

- Implement continuous filter as **function** h
- One implementation: **EdgeConv**

$k = 5$ nearest neighbors



- Node $x_i \in \mathbb{R}^F$: Calculate **edge features**

$$\boxed{e_{ij} = h_\theta(x_i, x_j) \qquad h_\theta : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}^{F'}}$$

for each node x_j that is connected to x_i with an edge

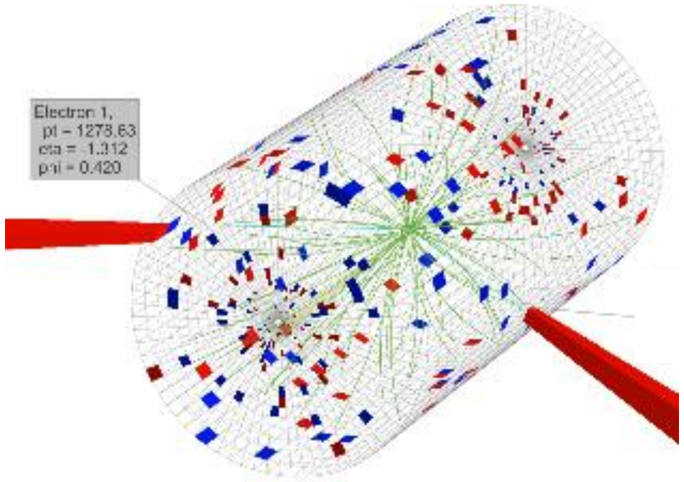
- h_θ with **trainable parameters** θ (e.g. a neural network) allows network to learn **approximation of optimal kernel function**
- Get „Feature Map“ by performing channel-wise symmetric **aggregation** $\square$ (e.g. mean)

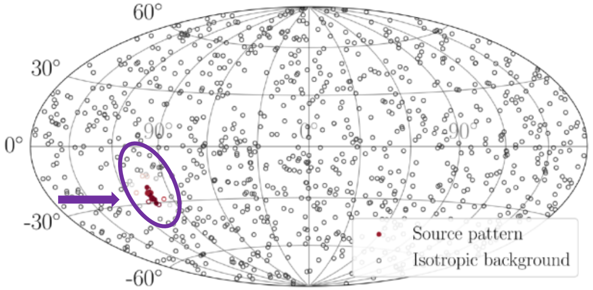
$$\boxed{x_i' = \mathop{\square}_{j:(i,j)\in\mathcal{E}} h_\theta(x_i, x_j)}$$

⇨ Transform graph with N nodes in F dimensions to graph with N nodes in F' dimensions
→ **New nearest neighbors (“dynamic”)!**

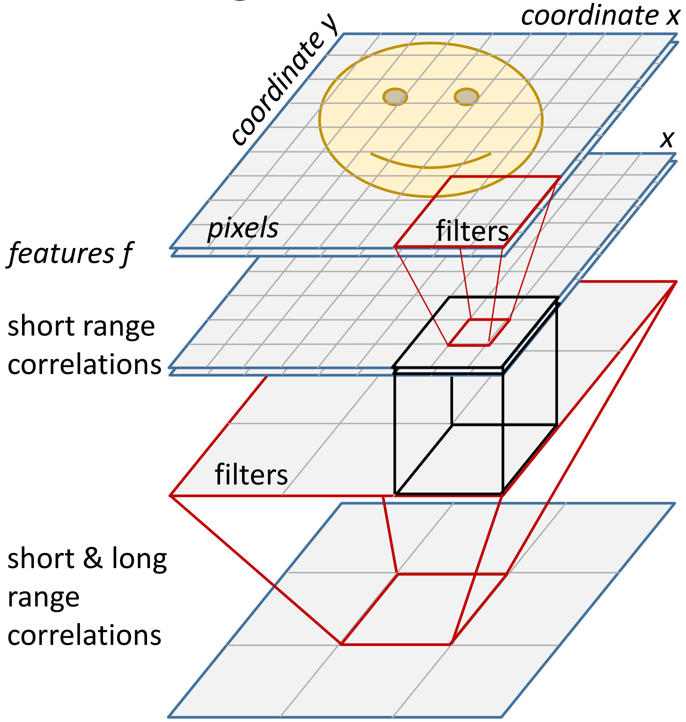
1) https://arxiv.org/abs/1801.07829

Convolution: Classic versus Graph network



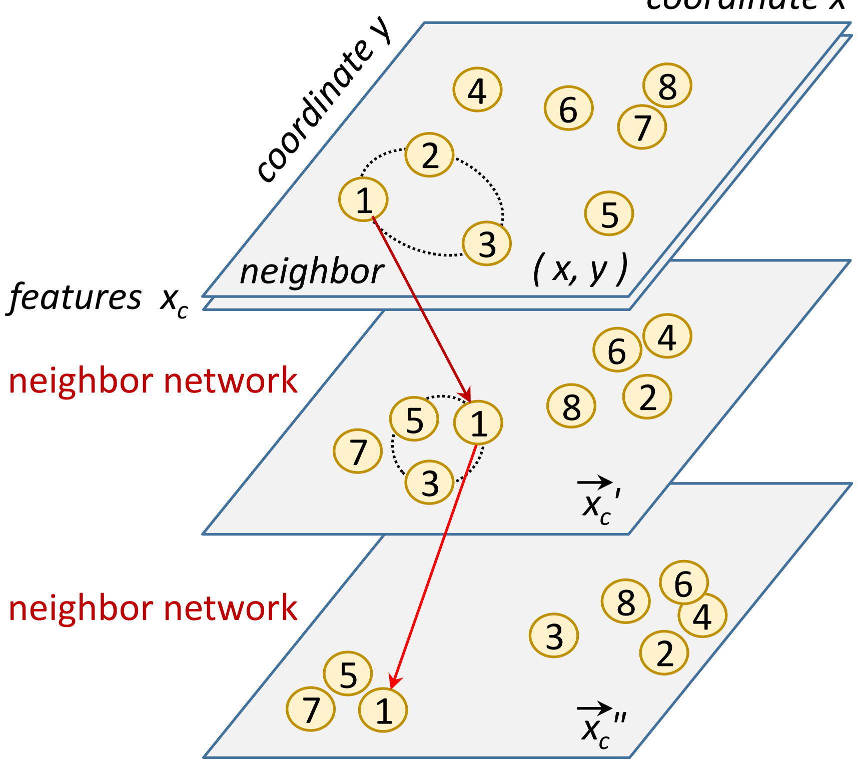


Classic: grid



- dense pixel occupation
- object identification
- analysis: single pixel contributions

Graph: neighbors



- sparse or dense neighbor occupation
- pattern recognition
- single objects 1 identification

Martin Erdmann

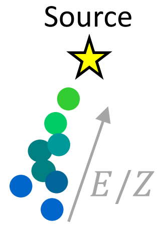
Example: Classification of Cosmic-Ray Arrival Directions

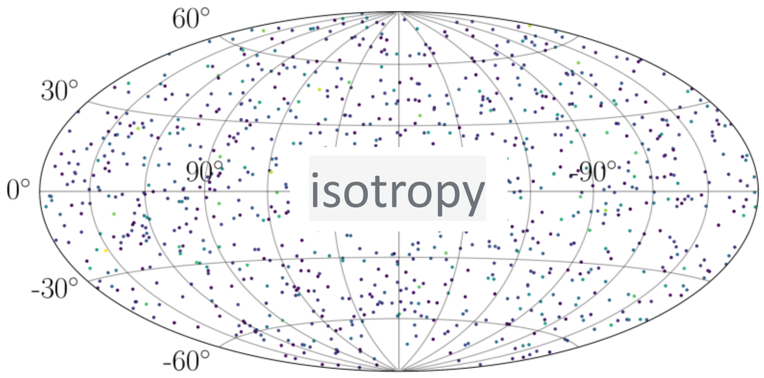
Ultra-high-energy cosmic rays from source

Passage through Galactic magnetic field

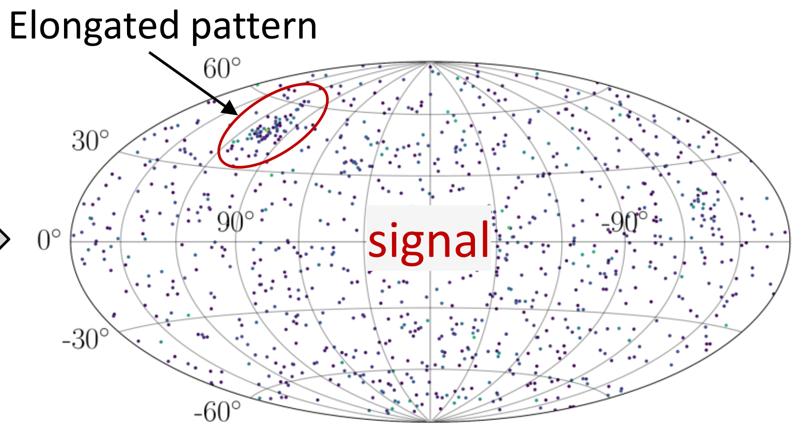
Deflection Depending on Energy and Charge

Pattern in arrival directions





Classification task



VISPA Example!

https://www.sciencedirect.com/science/article/pii/S0927650520300992

Describing a Graph: Graph Laplacian $\boldsymbol{L}$

Consider graph with N nodes

- **Unnormalized graph Laplacian $\boldsymbol{L}$** of shape $(N \times N)$
- Defined by $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$
- Discrete version of the Laplace operator

- **(Symmetric) normalized graph Laplacian**: $L^{\text{sym}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

$$\boxed{L^{\text{sym}} = U \Lambda U^T}$$

$\uparrow$

L^{sym} diagonalized by Fourier basis

$$\boxed{\begin{aligned} U &= [u_0, \dots, u_{N-1}] \in \mathbb{R}^{N \times N} \\ \Lambda &= \operatorname{diag}([\lambda_0, \dots, \lambda_{N-1}]) \in \mathbb{R}^{N \times N} \end{aligned}}$$

Graph Convolutions in the Spectral Domain

$$\boxed{L^{\mathrm{sym}} = U \Lambda U^T}$$

Use to define **convolution** of **graph signal** f:

1. Multiplication of U^T with f yields **Fourier transform**

$$\hat{f} = \mathcal{F}_{\mathcal{G}}\{f\} = U^{\mathrm{T}} f \qquad \mathcal{F}_{\mathcal{G}}^{-1}\{\hat{f}\} = U\hat{f} = f$$

2. **Convolution theorem**

$$f * g = \mathcal{F}^{-1}\big\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\big\}$$

3. Convolution operation of f with **kernel function** h

$$h(L^{\mathrm{sym}}) f = U\left(h(\Lambda) U^{\mathrm{T}} f\right)$$

- Convolution **depends on structure** of the graph (i.e. on L)
- Computationally demanding / challenging to implement
- **Efficient implementations** exist, e.g. using Chebyshev polynomials (see arXiv:1606.09375)

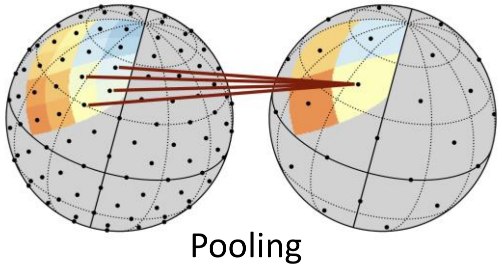
Using Spectral Graph Convolutions

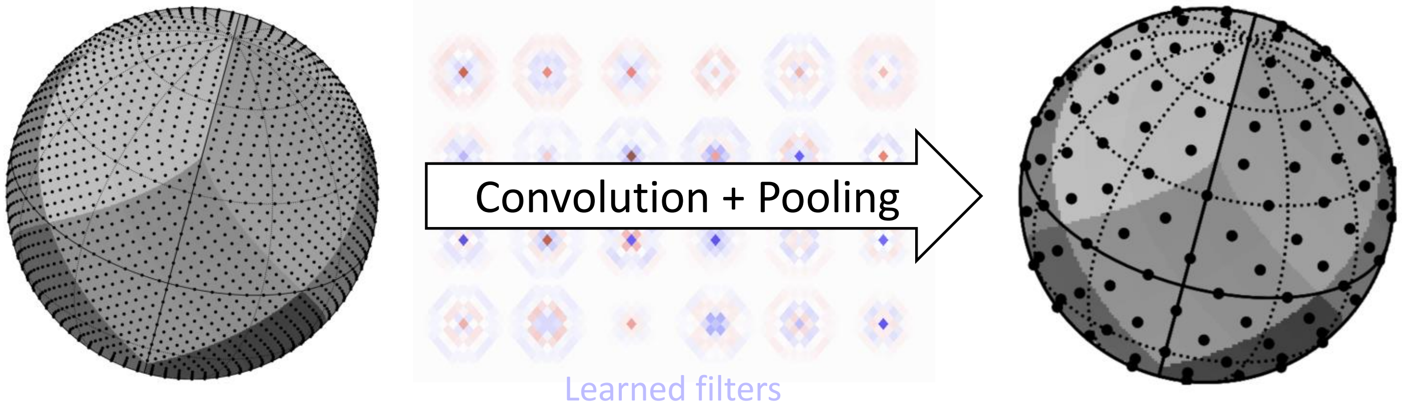
- **Graph structure does not change** by convolution itself → can be followed by pooling
- Can assign **properties** to each node and interpret the node itself as corresponding to one **position** (thus incorporating spatial relations)
- **Edges** chosen depending on the task, e.g. by calculating **nearest neighbors**
- Convolution defined **based on the graph** → graph **has to stay the same** between different datasets

Suited for:

→ Detector hits with a detector layout that does not change

→ Spherical data in the HEALPix format (example: **DeepSphere**[1])

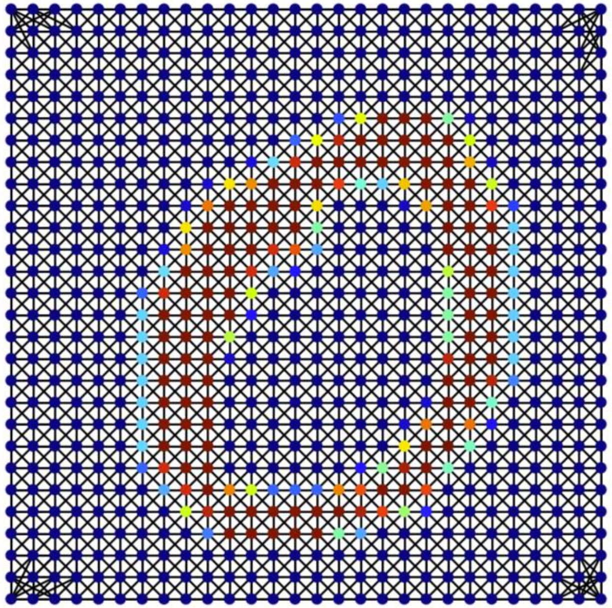


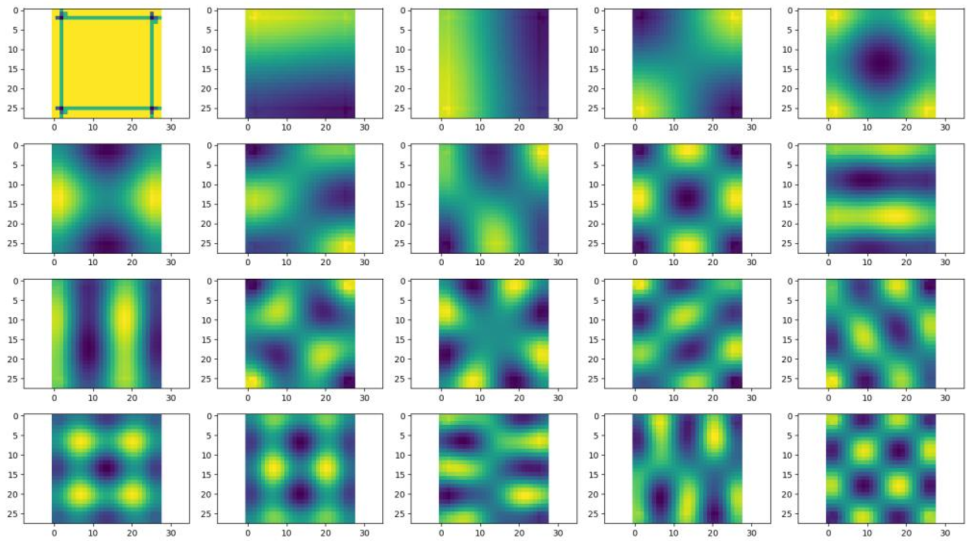


1) https://arxiv.org/abs/1810.12186

Example: MNIST

Images can also be treated as graphs with 1 pixel ≙ 1 node.



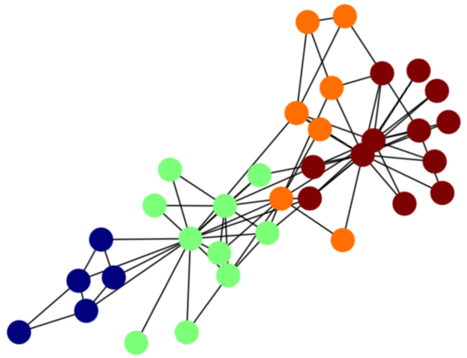


First 20 eigenvectors of L

VISPA Example!

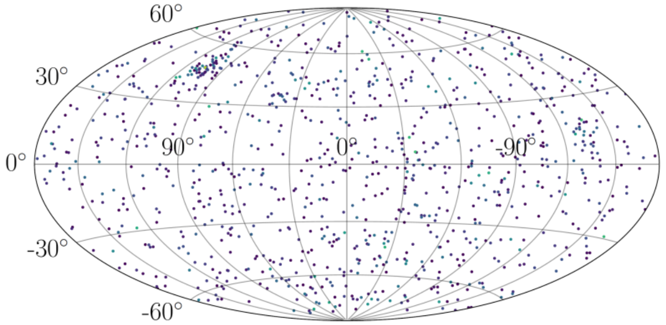
Graph Neural Networks Overview

Data: One fixed graph



Graph Convolutional Network

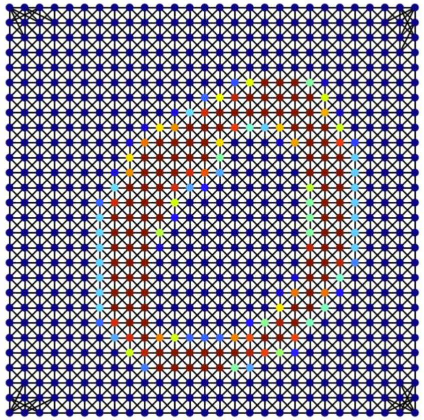
Data: Multiple graphs, changing shape

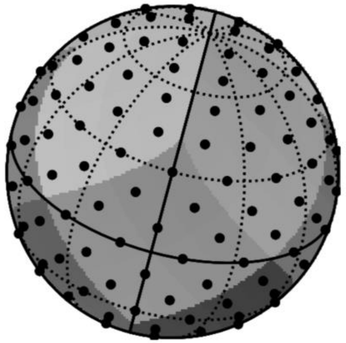


e.g. point clouds

Spatial (Dynamic) Graph Convolution

Data: Multiple graphs, identical shape





Spectral Graph Convolution

Often possible to apply **more than one** approach to a problem (**exact position → spatial graph convolution, positions pixelized on grid → spectral graph convolution, …**)

Find best-suited approach based on shape of data, its symmetries and performance/efficiency