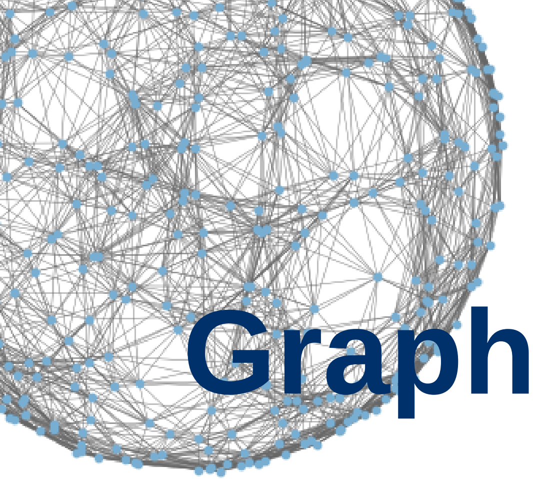
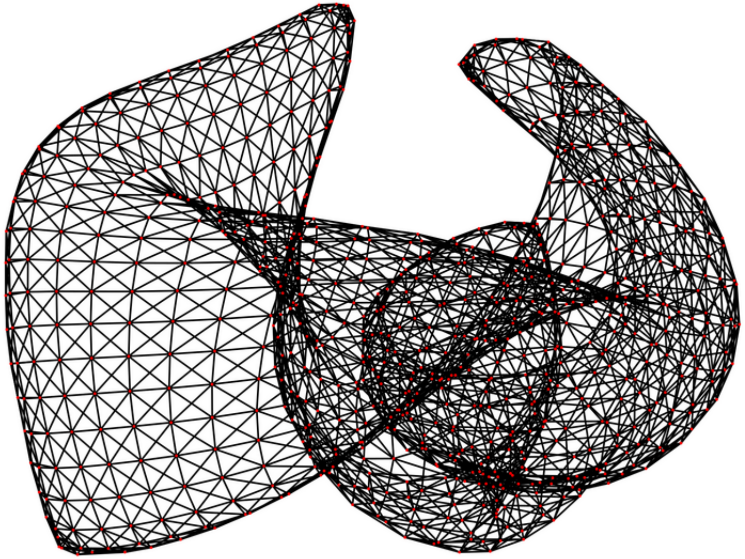


Graph Convolutional Networks

- Graphs and graph basics
- Convolutions on non-Euclidean domains
- Graph Convolutional Neural Networks
  - Spatial domain
  - Spectral domain



Jonas Glombitza

Deep Learning Train-the-Trainer workshop, Wuppertal, 9 - 10 June 2022

Time Schedule

Introduction: Graphs and Graph Convolutions

- Basics of graphs and graph theory

Graph Convolutional Networks

- *Example 1: Semi-supervised node classification using GCNs*

Convolutional in Spatial Domain

- EdgeConvolutions and Dynamic Graph Convolutional Neural Networks
- *Example 2: Cosmic-ray classification using DGCNNs*

Convolutions in Spectral Domain

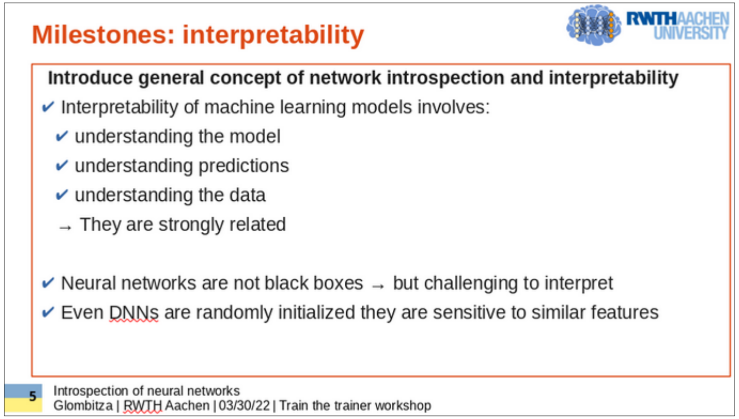
- Spectral graph theory
- Chebychev Convolutions (ChebNets)
- *Example 3: MNIST on graphs using ChebNets*

complexity

Structure

- Example lecture
  - introduction to graph networks

- Milestone slides:
  - pedagogical reasoning (and important points)



Milestones: interpretability

Introduce general concept of network introspection and interpretability

- ✔ Interpretability of machine learning models involves:
  - ✔ understanding the model
  - ✔ understanding predictions
  - ✔ understanding the data
  - → They are strongly related
- ✔ Neural networks are not black boxes → but challenging to interpret
- ✔ Even DNNs are randomly initialized they are sensitive to similar features

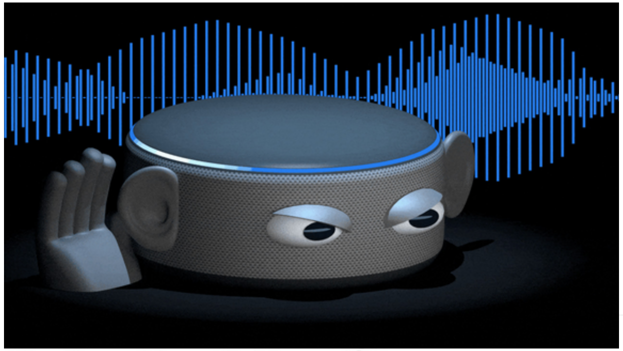
5 Introspection of neural networks
Glombitza | RWTH Aachen | 03/30/22 | Train the trainer workshop

Feel free to ask questions during the seminar!
Just raise your hand...

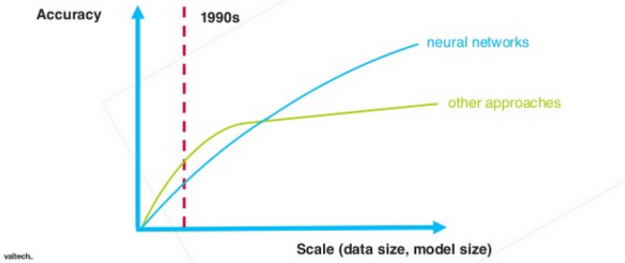
Deep Learning

- Outstanding results
  - Speech recognition
  - Image recognition → **Convolutions**

nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO
CONSERVATION
SONGBIRDS À LA CARTE
Illegal harvest of millions of Mediterranean birds
PAGE 452
RESEARCH ETHICS
SAFEGUARD TRANSPARENCY
Don't let openness backfire on individuals
PAGE 459
POPULAR SCIENCE
WHEN GENES GOT 'SELFISH'
Dawkins's calling card forty years on
PAGE 462
NATURE.COM/NATURE



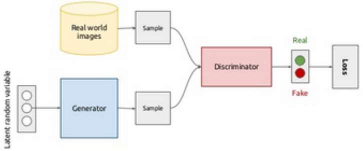
More Data + Bigger Models

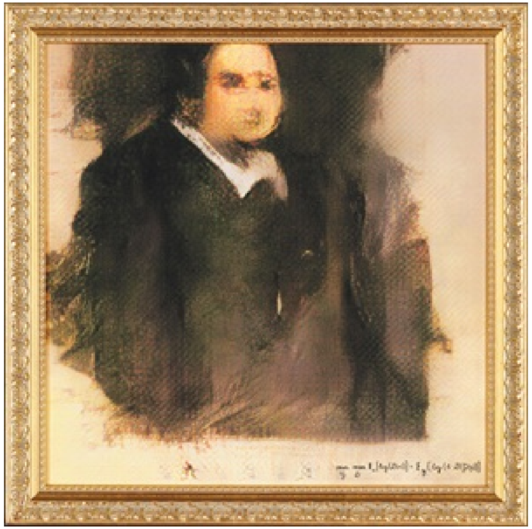


valtech.

https://www.scribd.com/document/355752799/Jeff-Dean-s-Lecture-for-YC-AI

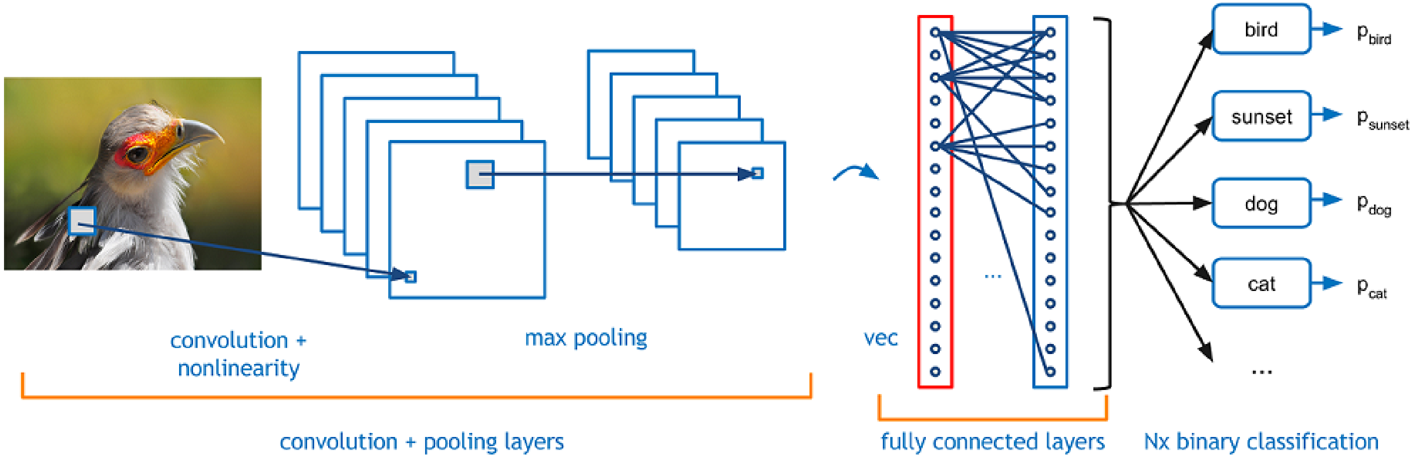
Generative adversarial networks (conceptual)



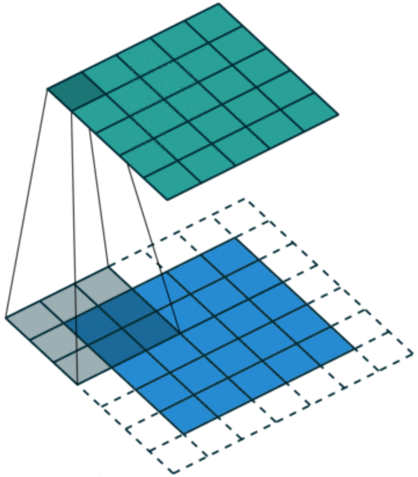


Convolutions

- Translational invariance
- Scale separation (hierarchy learning)
- Deformation stability (filters are localized in space)
- Parameters are independent from input size



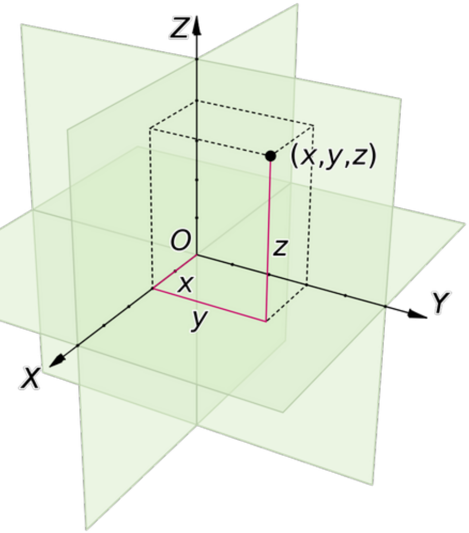
Adit Deshpande - https://adeshpande3.github.io/adeshpande3.github.io/

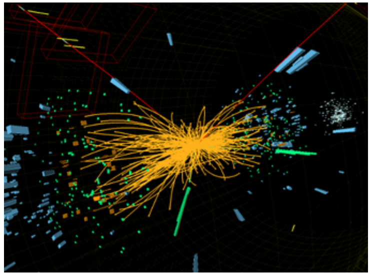


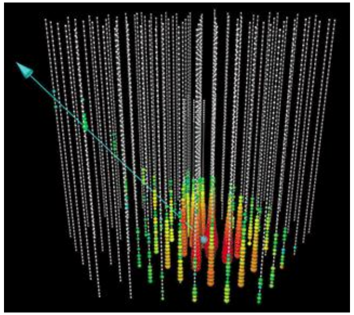
Paul-Louis Pröve,
Towards Data Science

Convolutions and Datasets

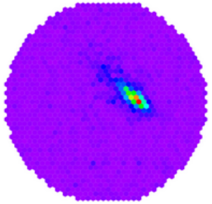
- Works in well defined euclidean space

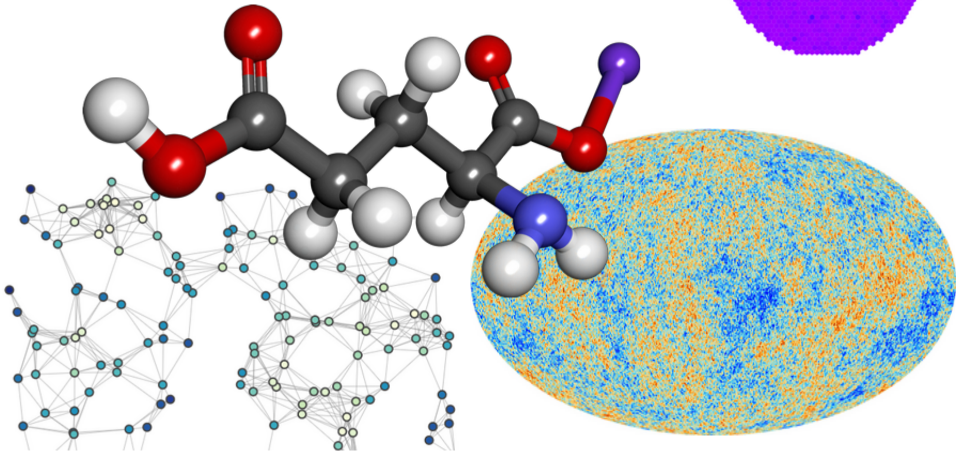






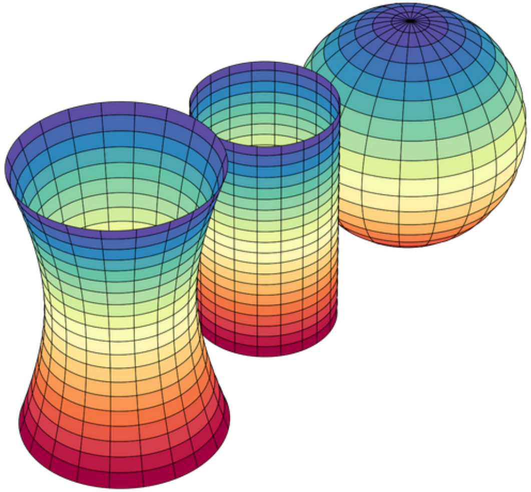
- physics data often feature different geometries



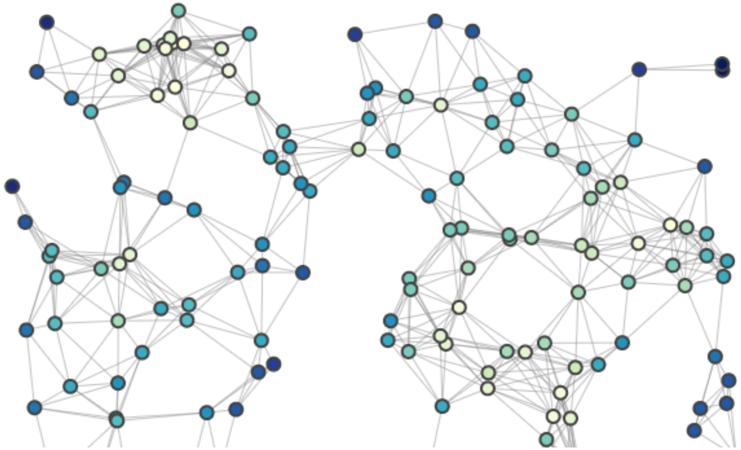


Generalization to Non-Euclidean Domains

- Defining convolutions, challenging on non-euclidean domains
  - Deformation of filters, changing neighbor relations
  - Non-isometric connections on graphs



- **Manifolds**



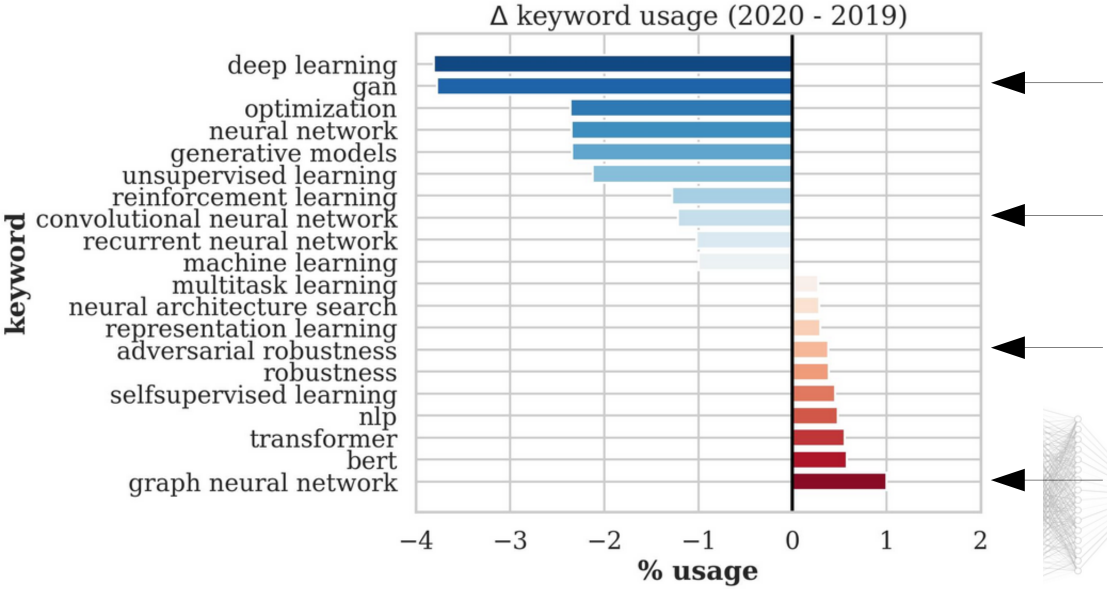
- **Graphs**

How can we generalize convolutions?

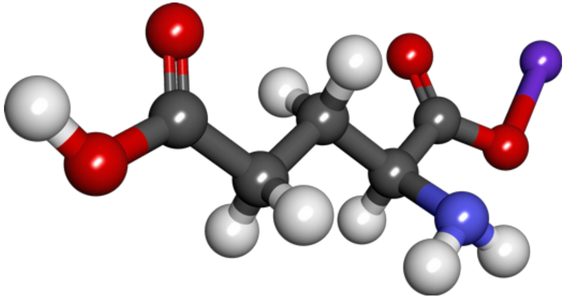
Deep Learning on Graphs

I. Introduction to graphs
II. Graph basics
III. Spectral graph theory

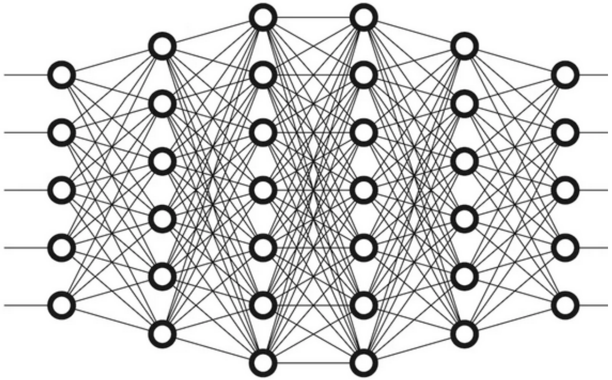
ICLR2020 submissions - growth



Types of Graphs

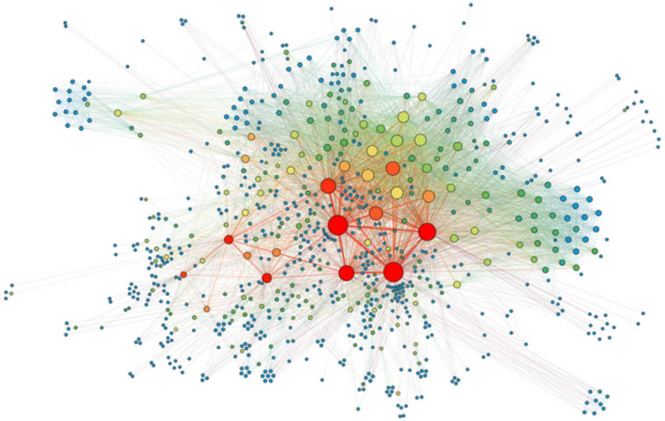


heterogeneous graph

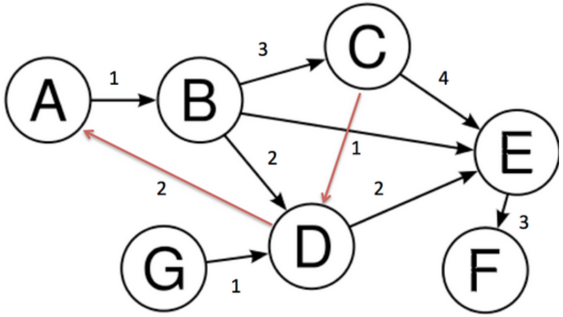


bipartite graph

graphs with edge information



undirected graph



directed graph

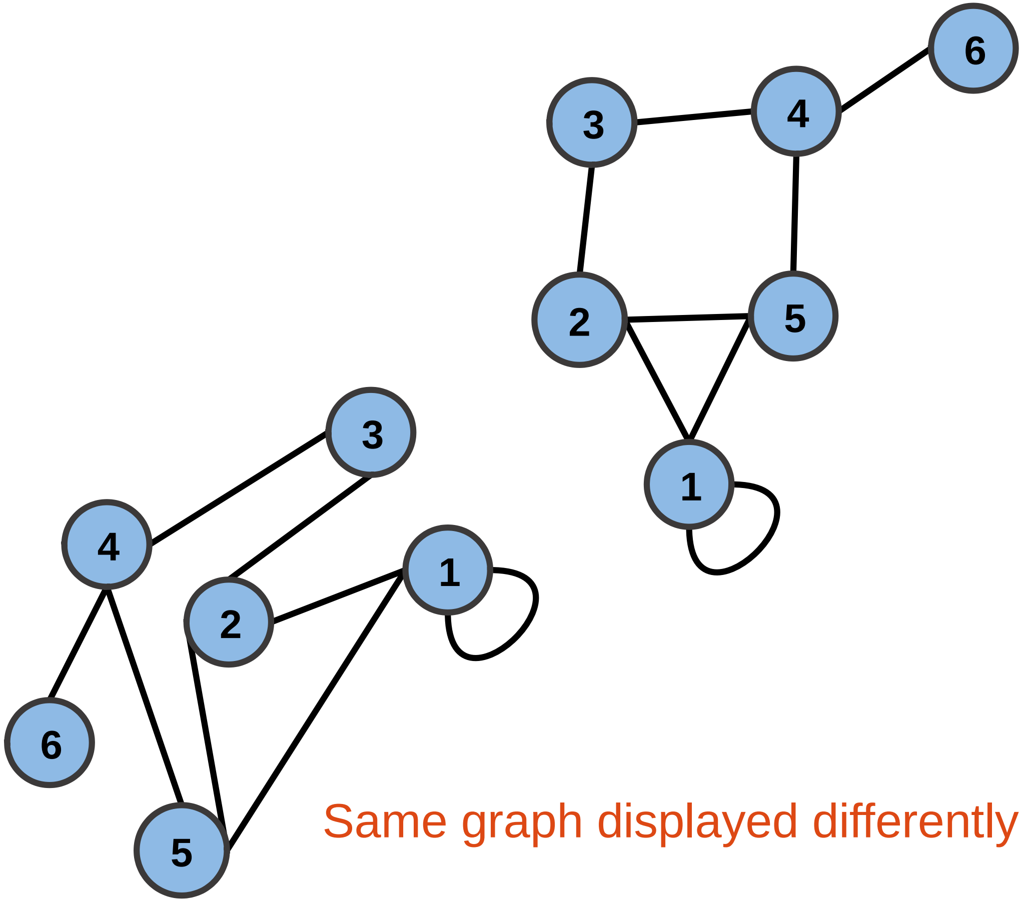
What is a Graph

$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$

- Graph is ordered pair
  - of nodes $\mathcal{V}$
  - and edges $\mathcal{E}$
- ➢ mainly defined by neighborhood

- Nodes have no order
  - ➢ **permutational invariance**

- challenging to visualize!

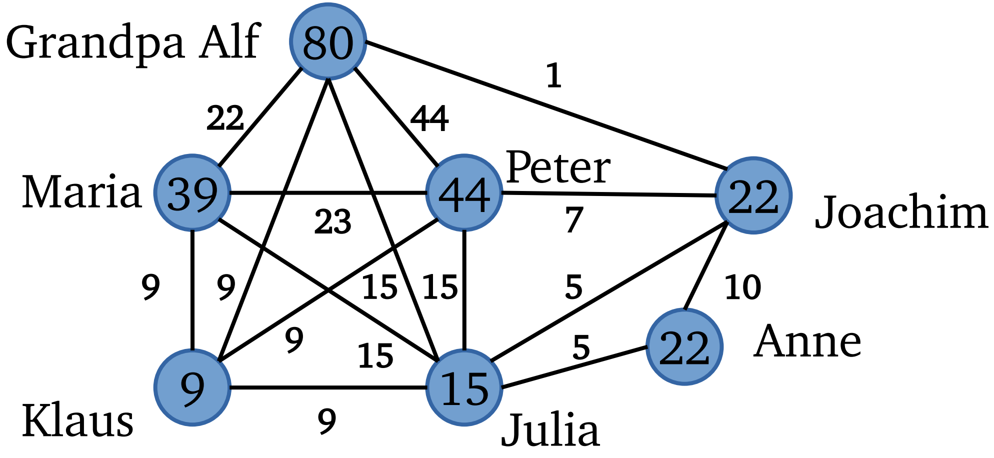


Same graph displayed differently

Example: Various Graphs

Social network

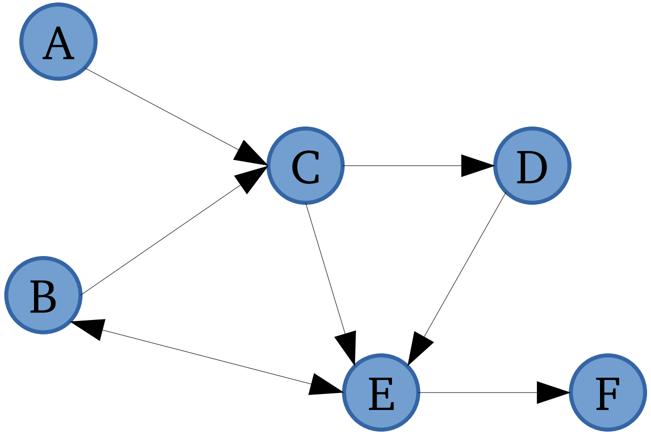
Bidirectional graph,
Including edge information



node = age of person
edge = age of relationship

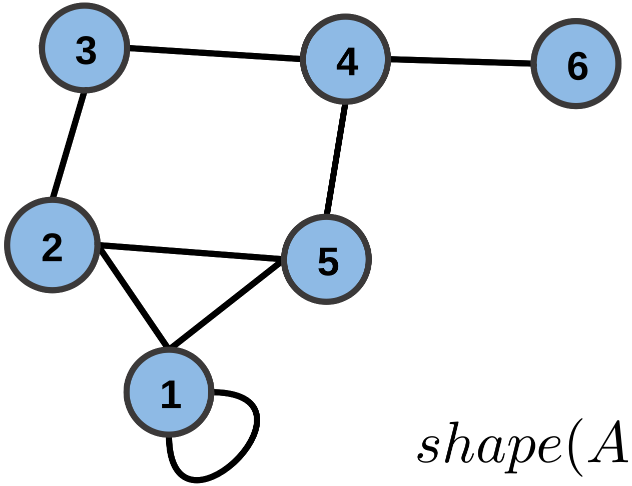
Production Chain

Directed graph



Adjacency Matrix

- Matrix to represent structure of graph
- Elements indicate edges of graph
- Symmetric for undirected graphs
- In general sparse



$$shape(A) = N \times N$$

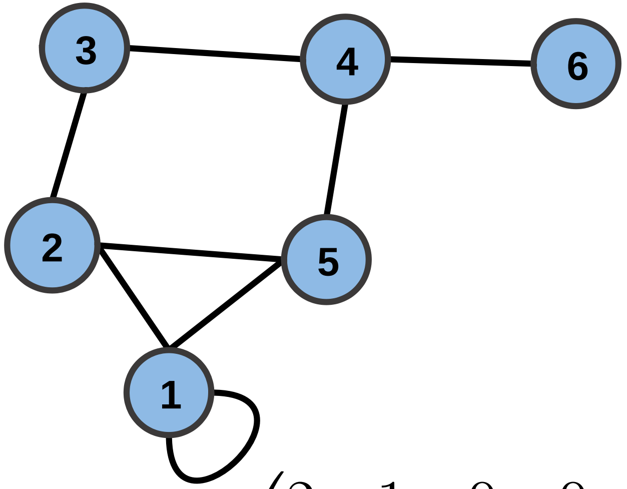
$$A = \begin{array}{cccccc|c} 1 & 2 & 3 & 4 & 5 & 6 & \\ \hline 2 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 1 & 1 & 4 \\ 1 & 1 & 0 & 1 & 0 & 0 & 5 \\ 0 & 0 & 0 & 1 & 0 & 0 & 6 \end{array}$$

- Used to propagate signals on the graph

$$A \cdot \underset{\text{signal}}{f} = \begin{pmatrix} 2 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Degree Matrix

- Elements count number of times edges terminate at each node
- Used used to normalize adjacency A
- $shape(D) = N \times N$



$$\begin{array}{cc} & \begin{array}{cccccc}1 & 2 & 3 & 4 & 5 & 6\end{array} \\ D = & \begin{pmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{array}{c}1\\2\\3\\4\\5\\6\end{array} \end{array}$$

$$A = \begin{pmatrix} 2 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Laplacian Matrix

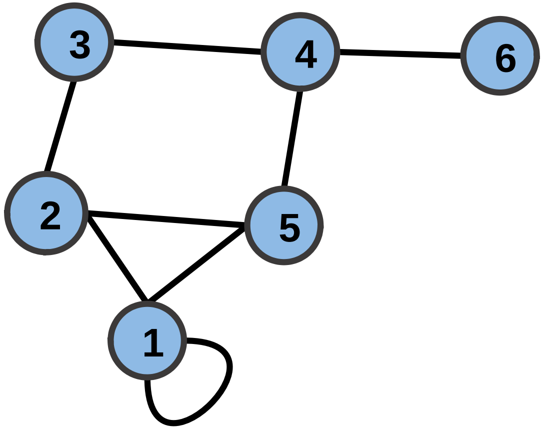
- Laplacian matrix L = normalized adjacency matrix A
  - $L = D - A$

- Difference between f and its local average
- Core operator in spectral graph theory

- Symmetric normalized Laplacian:
  - Eigenvalues do not depend on degree of nodes

$$L^{\mathrm{sym}} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

- Discrete version of Laplace operator



f = function acting on the graph

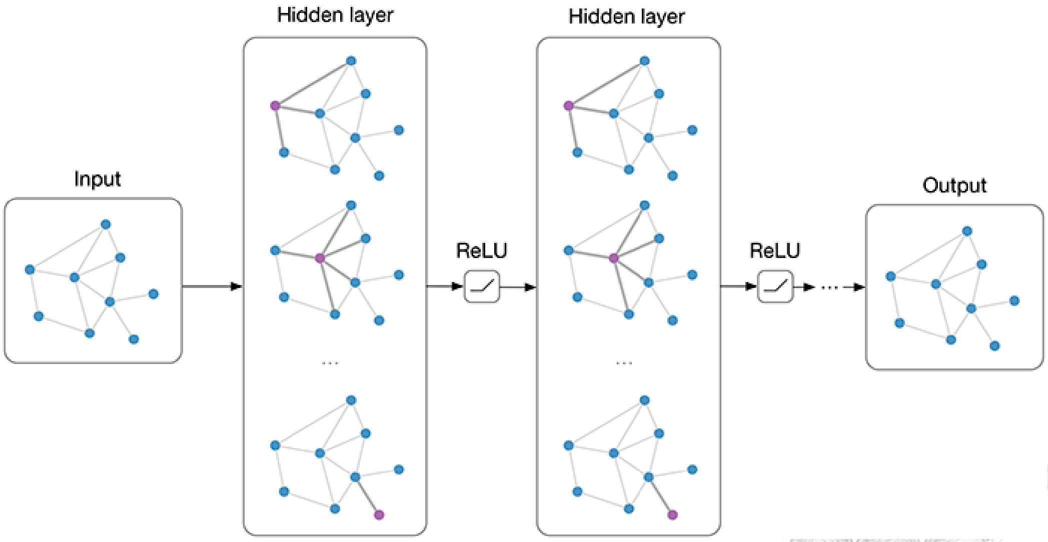
Milestones: Graph Networks

Introduce general concept of graphs and graph networks
Convolutions beyond euclidean domains (beyond image-like data)
Needed for following chapters (particularly convolutions in spectral domain)

- ✓ Motivate Graph Convolutional Networks (prior: neighborhood relation)
- ✓ Connect to your research (data structure / graph-like?)
- ✓ Introduction to graphs:
  - ✓ are collection of edges and nodes (plenty of representations)
  - ➢ defined by neighborhood
- ✓ Introduce basics of spectral graph theory
  - ✓ Adjacency, Degree, Laplacian

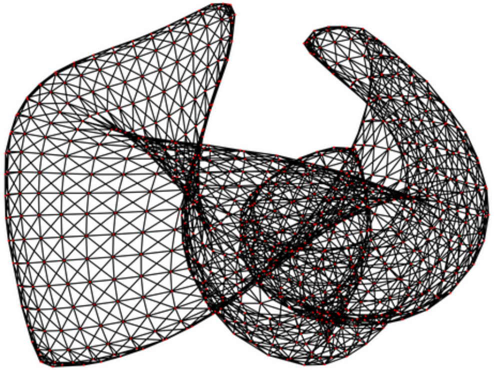
Graph Convolutional Networks

I. Propagation rule for GCN
II. Connection to CNNs
III. Semi-supervised classification



Thomas Kipf, Max Welling
arXiv:1609.02907

Natural Images vs. Graphs



- Collection of pixels (node)
  - Node (pixel) holds feature vector
  - Dense (rarely sparse)
  - Discrete, regular (symmetric)

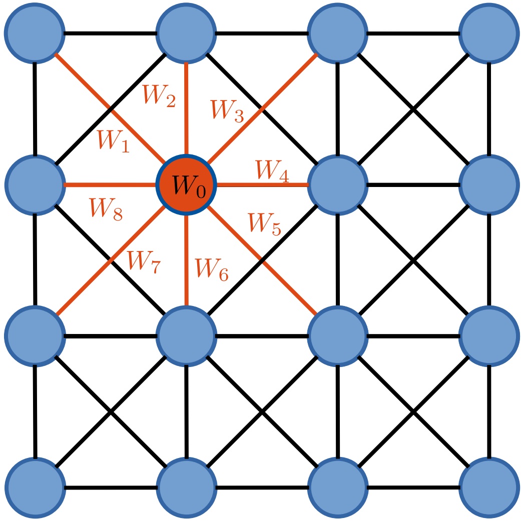
- Images feature euclidean space

- Collection of nodes and edges
  - Node + edge holds feature vector
  - Can be dense or sparse
  - Continuous non-symmetric positions

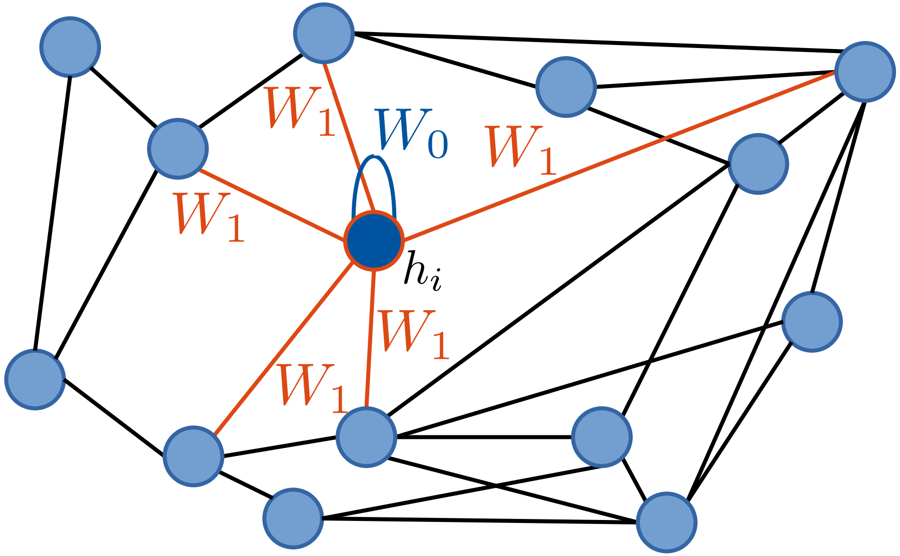
- Graphs can feature “arbitrary” domains

Graph Convolutional Networks

2D Convolution on regular grid



Convolution on Graph



- Channel-wise weight-sharing!
- Node-wise weight-sharing!
- Propagation rule for GCN:

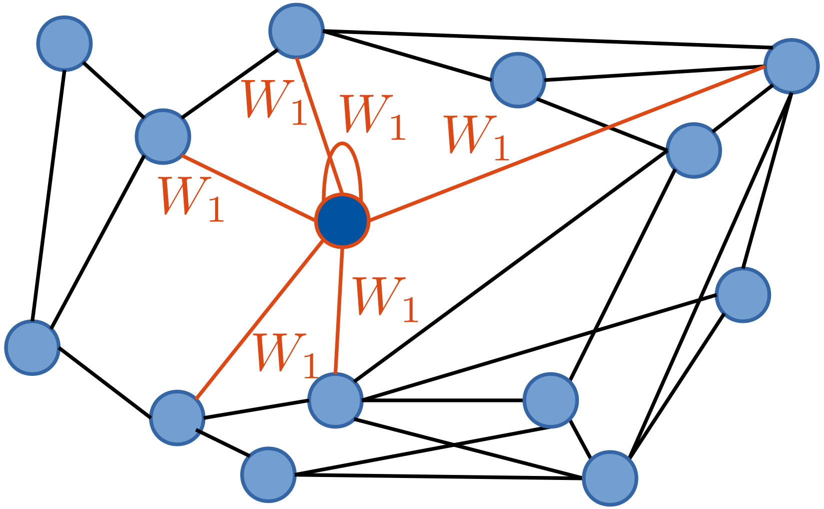
$$h_i^{(l+1)} = \sigma(h_i^{(l)} W_0^{(l)} + \underbrace{\sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}}}_{\text{Average over neighbors}} h_j^{(l)} W_1^{(l)})$$

Deep Learning for Physics Research, World Scientific

Graph Convolutional Networks

- In general more easy $W_0 = W_1$
- Self coupling: same weight as neighbors
  - Very simple → works surprisingly good

- Node-wise weight-sharing!



$$h_i^{(l+1)} = \sigma\left(h_i^{(l)} W_1^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} h_j^{(l)} W_1^{(l)}\right)$$

Mathematical Formulation

- Input $H^{(0)} = X$, input data (signal)
  - $shape(X) = N \times D$

number of nodes

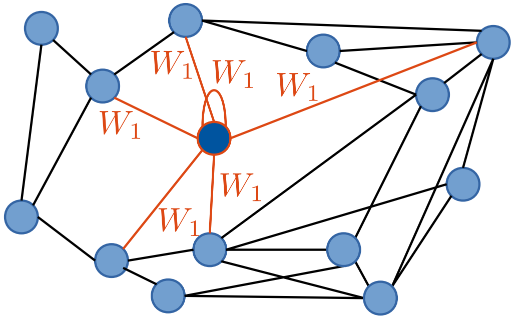
dimension of input feature (per node)

- Weight signal with neighborhood using adjacency matrix A, $shape(A) = N \times N$
  - $H = f(X, A) \sim AH$

number of kernels (new features)

- Apply transformation using weight matrix W, $shape(W) = D \times F$
  - $H = \sigma(AXW^{(l)})$

- As A do not include self loops, we have to add them:
  - $\hat{A} = I + A$



Normalization

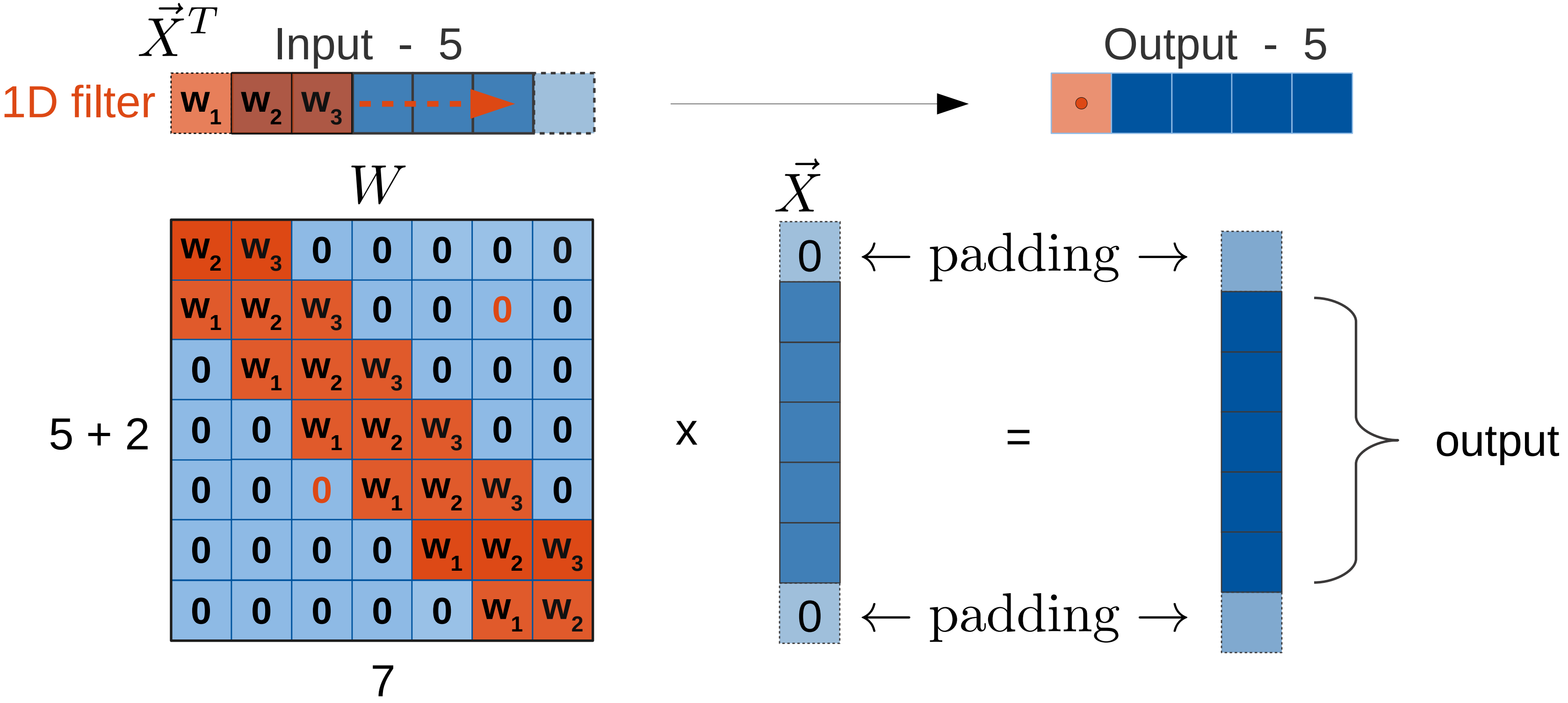
- Normalization needed in deep learning
  - Input / output normalization + batch / feature normalization
  - Weight normalization

- $\hat{A} = I + A$ is not normalized
  - Each multiplication would change feature scale!
- Normalize new adjacency matrix using *degree matrix* $\hat{D}$ of $\hat{A}$ (average over neighbor nodes)
  - $A \rightarrow \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}}$

- Final propagation rule: $f(H^{(l)}, A) = \sigma\left(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right)$
  - Can be repeated for each layer, by sharing graph structure A

Recap: Convolutional Operation

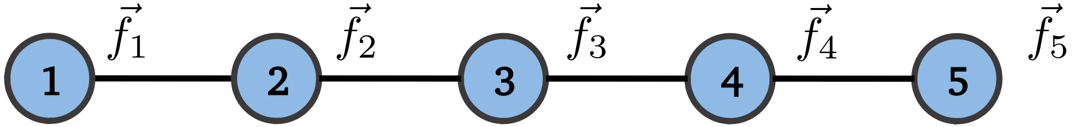
- Fully connected layers are special case of convolutional layers



$$W = \begin{pmatrix} w_2 & w_3 & 0 & 0 & 0 & 0 & 0 \\ w_1 & w_2 & w_3 & 0 & 0 & 0 & 0 \\ 0 & w_1 & w_2 & w_3 & 0 & 0 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 & 0 & 0 \\ 0 & 0 & 0 & w_1 & w_2 & w_3 & 0 \\ 0 & 0 & 0 & 0 & w_1 & w_2 & w_3 \\ 0 & 0 & 0 & 0 & 0 & w_1 & w_2 \end{pmatrix}$$

- Strong prior on **local correlation** and **translational invariance**

Graph Convolution



3-dim. feature vector at each node

$$\mathbf{H}^{(l+1)} = \sigma(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$$

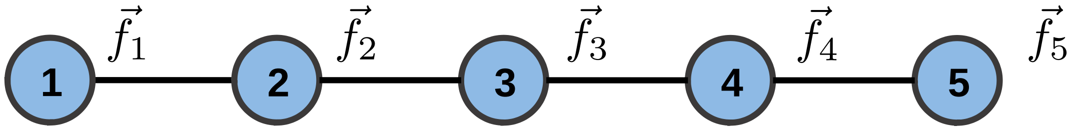
Similar to CNN weight matrix!

$$\begin{array}{ccccccc}
\hat{\mathbf{A}} & & \mathbf{H}^{(0)} & & \mathbf{W} & & \mathbf{H}^{(1)} \\
\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} & \text{x} & \begin{pmatrix} \vec{f}_1 \\ \vec{f}_2 \\ \vec{f}_3 \\ \vec{f}_4 \\ \vec{f}_5 \end{pmatrix} & \text{x} & \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix} & = & \begin{pmatrix} \dots \\ \dots \\ \dots \\ \dots \\ \dots \end{pmatrix}
\end{array}$$

shape: 5 x 5 5 x 3 3 x 2 5 x 2

Graph Convolution

- Convolutional layers are special case of graph convolutional layers



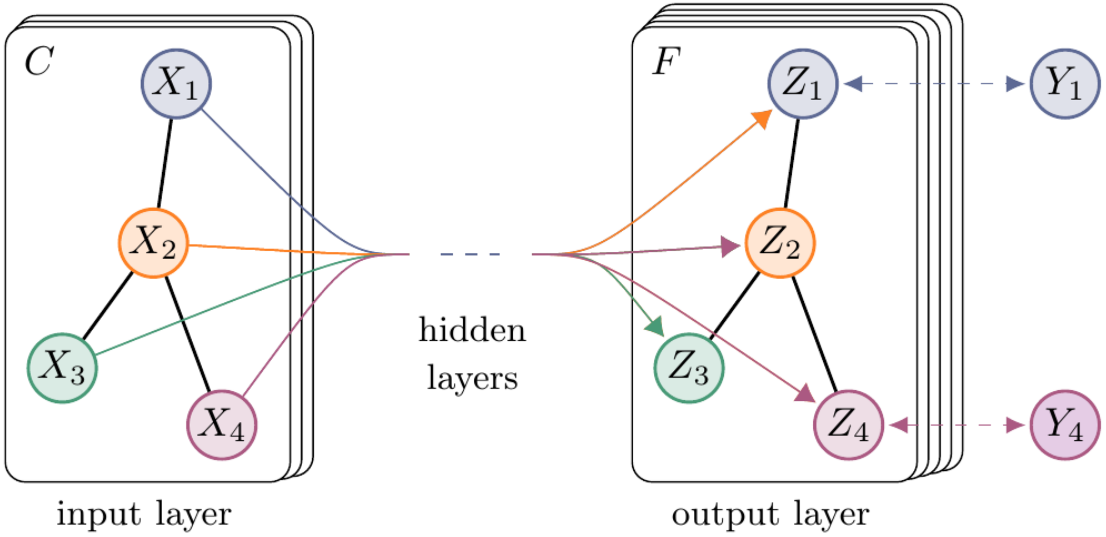
$$\mathbf{H}^{(l+1)} = \sigma(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$$

$$shape(H^{(l)}) = N \times D$$
$$shape(A) = N \times N$$
$$shape(W) = D \times F$$

- Output: 5 nodes
  - structure (fixed) shared over model
- Graph convolution
  - 6 adaptive weights
- Cartesian convolution
- 3x2x3=18 adaptive weights, neglecting bias, translational invariance + filtersize = 3

$$\begin{array}{ccccc} \hat{\mathbf{A}} \cdot \mathbf{H}^{(0)} & & \mathbf{W} & & \mathbf{H}^{(1)} \\ \begin{pmatrix} \vec{f}_1+\vec{f}_2 \\ \vec{f}_1+\vec{f}_2+\vec{f}_3 \\ \vec{f}_2+\vec{f}_3+\vec{f}_4 \\ \vec{f}_3+\vec{f}_4+\vec{f}_5 \\ \vec{f}_4+\vec{f}_5 \end{pmatrix} & \mathrm{x} & \begin{pmatrix} \mathbf{w}_{11} & \mathbf{w}_{12} \\ \mathbf{w}_{21} & \mathbf{w}_{22} \\ \mathbf{w}_{31} & \mathbf{w}_{32} \end{pmatrix} & = & \begin{pmatrix} \dots \\ \dots \\ \dots \\ \dots \\ \dots \end{pmatrix} \\ 5 \times 3 & & 3 \times 2 & & 5 \times 2 \end{array}$$

Graph Convolutional Network - GCN

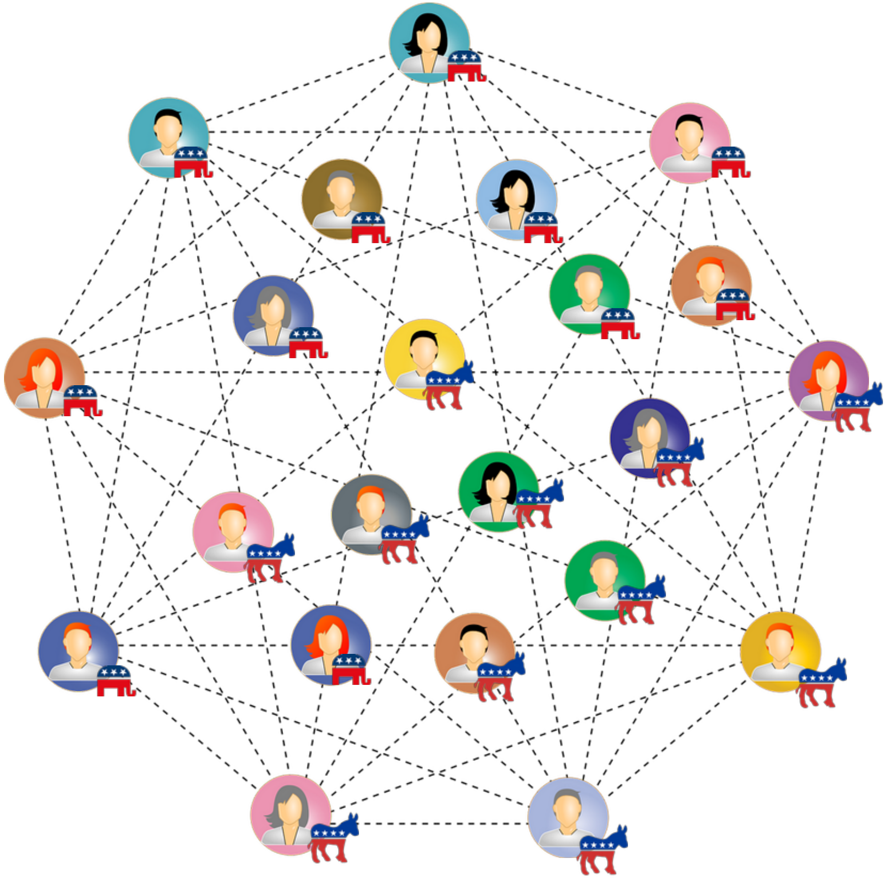


(a) Graph Convolutional Network

arXiv:1609.02907

- Share graph structure over model
  - Calculate once $A = \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}}$ during pre-processing
- Aggregate neighborhood information in every node
  - $H^{(l+1)} = \sigma(AH^{(l)}W^{(l)})$

Node Classification – social network

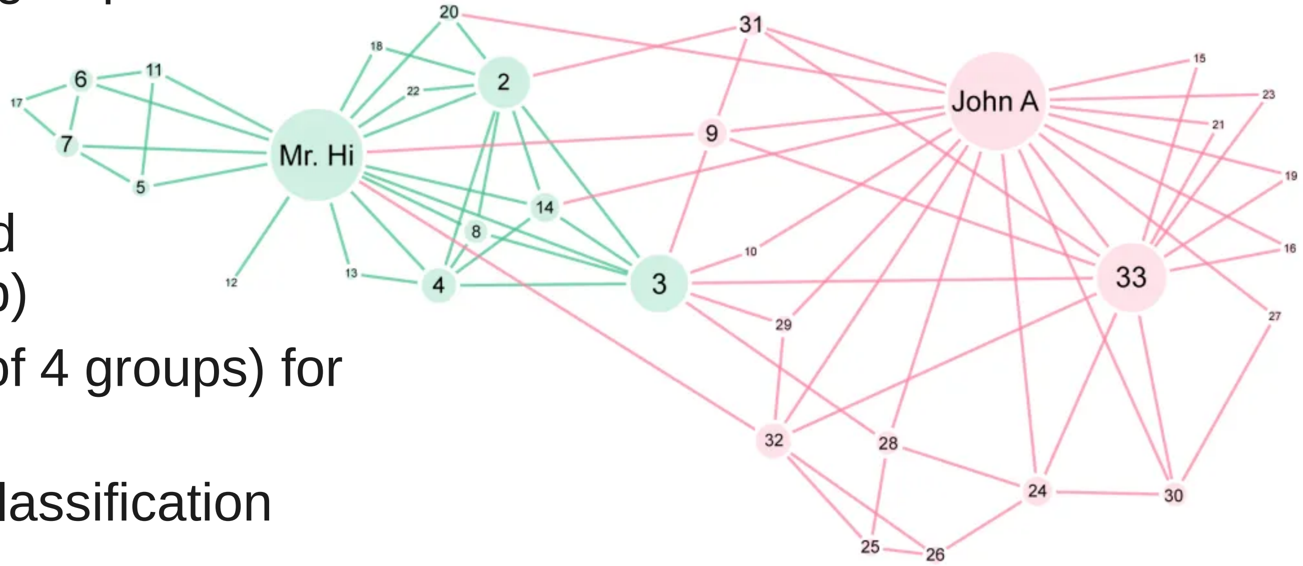


- Node Classification of single graph
  - Social network
- Clustering / classification of nodes
  - Voting behavior of individual persons

- Semi-supervised
  - use few labels || rest of nodes masked
- Unsupervised
  - without label information

Example: Zachary's Karate Club

- “Historical” data set
- Social network of university karate club
  - Edges represent social relationships outside the club
- Conflict between administrator “John. A” and trainer “Mr. Hi”
  - Karate Club splits in 4 groups



Task

- Given a single graph and 4 labels (1 of each group)
- Identify membership (1 of 4 groups) for every person
- Semi-supervised node classification

Milestones: Graph Convolutional Networks

Investigate graph structure ("social networks analysis")
Soft introduction to GNNs (connect to graph theory and CNNs)

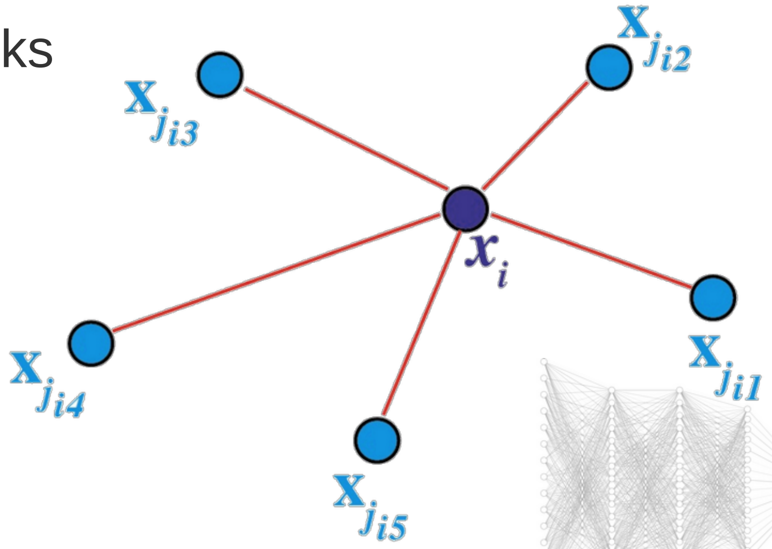
- ✓ Discuss Graph Convolutional Networks
  - ✓ basic structure & working principle of many GNN architectures
  - ✓ analyze graph-like data
  - ✓ adjacency matrix similar to weight matrix in CNNs
  - ✓ Each node uses the same adaptive parameter
    - identical to 1x1 convolution

Convolutions in the Spatial Domain

I. Edge-Convolutions

II. Dynamic Graph Convolutional Neural Networks

III.Physics example



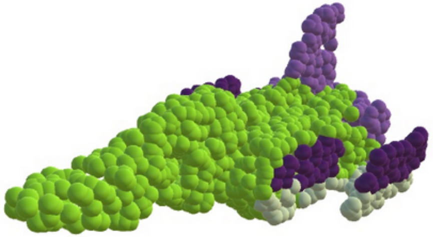
Y. Wang et al., ArXiv:1801.07829

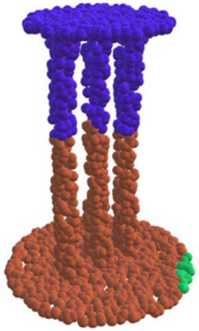
M. Simonovsky, N. Komodakis, ArXiv:1704.02901

Convolution in Spatial Domain

- Graphs feature permutational invariance of nodes
- Orientation of nodes meaningless

- Whats with networks *embedded* in a (spatial) domain?
  - Node position is important!
  - Not only neighborhood relationship!





https://arxiv.org/abs/1801.07829

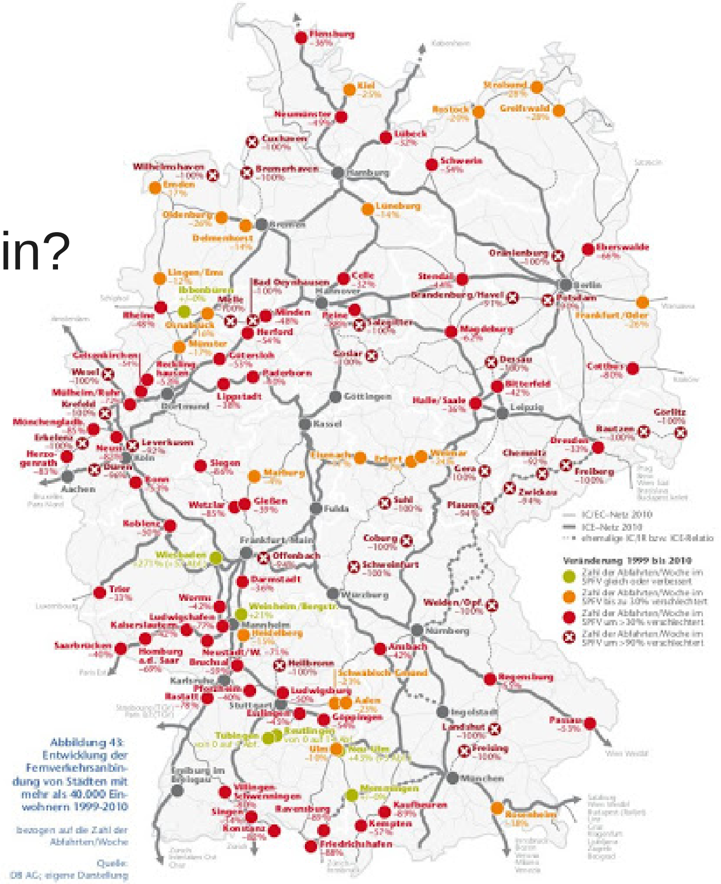
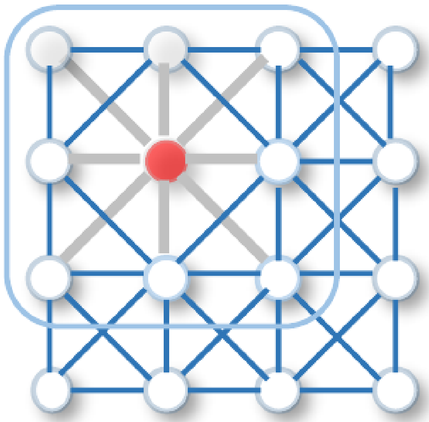


Abbildung 43: Entwicklung der Fernverkehrsanbindung von Städten mit mehr als 40.000 Einwohnern 1999-2010
bezogen auf die Zahl der Abfahrten/Woche
Quelle: DB AG; eigene Darstellung
IC/EC-Netz 2010
ICE-Netz 2010
ehemalige IC/IR bzw. ICE-Relatio
Veränderung 1999 bis 2010
Zahl der Abfahrten/Woche im SPFV gleich oder verbessert
Zahl der Abfahrten/Woche im SPFV bis zu 30% verschlechtert
Zahl der Abfahrten/Woche im SPFV um >30% verschlechtert
Zahl der Abfahrten/Woche im SPFV um >90% verschlechtert
Flensburg
Kiel
Hamburg
Bremen
Hannover
Berlin
Leipzig
Dresden
Kassel
Göttingen
Fulda
Frankfurt/Main
Köln
Dortmund
Aachen
Mannheim
Nürnberg
Stuttgart
Karlsruhe
München
Ingolstadt

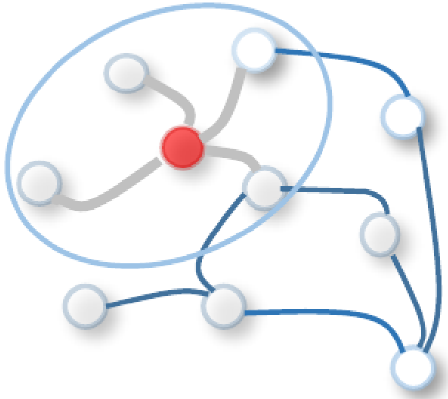
Convolution in Spatial Domain

- Images with discrete and continuous pixel coordinates

regular grid: equidistant positions



continuous grid positions

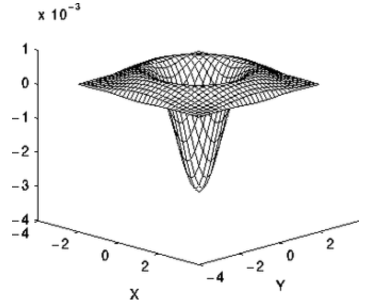


- Learned filter

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

$$\xrightarrow{\text{......?}}$$

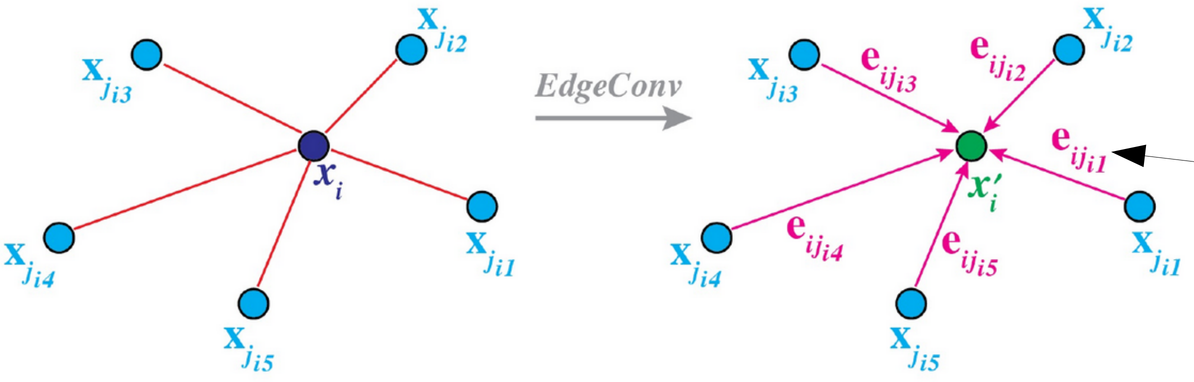
Transition of discrete filter to continuous filter

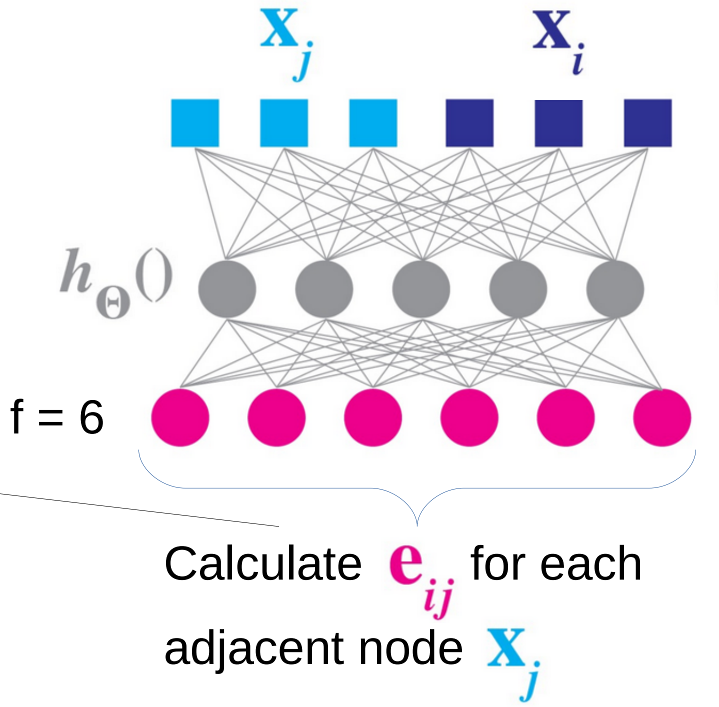


https://arxiv.org/abs/1901.00596

Edge Convolution

- ✗ For continuous pixelization → matrix becomes gigantic and sparse
- ➔ Approximate discrete f-dimensional kernel continuously using neural network
- Network applied at each pixel using:
  - central pixel x_i
  - relation to neighbor pixels eg. x_j or $x_i - x_j$
- Outputs f-dimensional feature vector





Calculate $\mathbf{e}_{ij}$ for each adjacent node $\mathbf{x}_j$

https://arxiv.org/abs/1801.07829

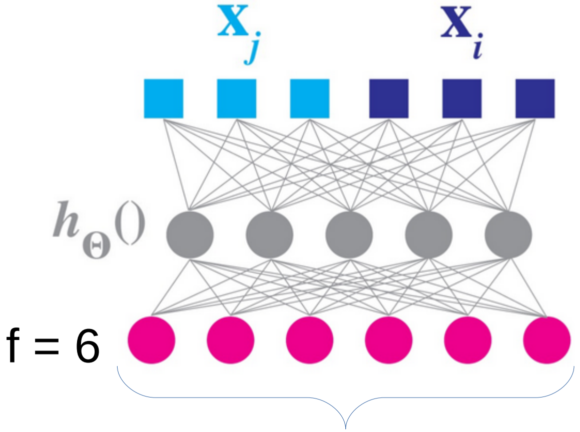
Edge Convolution

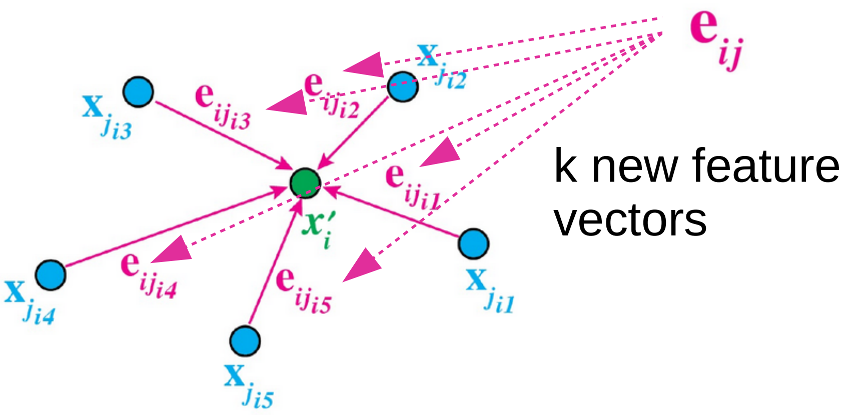
- Convolution acts on neighborhood $\mathbf{x}_i$ yielding for each node:
  - k new features $\mathbf{e}_{ij}$ (one for each neighbor)
  - feature dimension depends on features of $h_\Theta()$
  - **Parameters shared over edges**

- Aggregate neighborhood information
- Aggregation operation flexible:
  - e.g.

$$x'_i = \max_{j\in N_i} e_{ij}$$

$$x'_i = \langle e_{ij} \rangle_{j\in N_i}$$



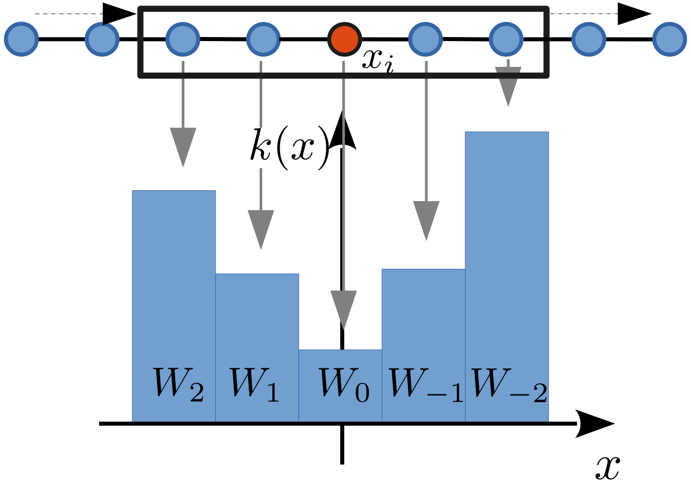


k new feature vectors

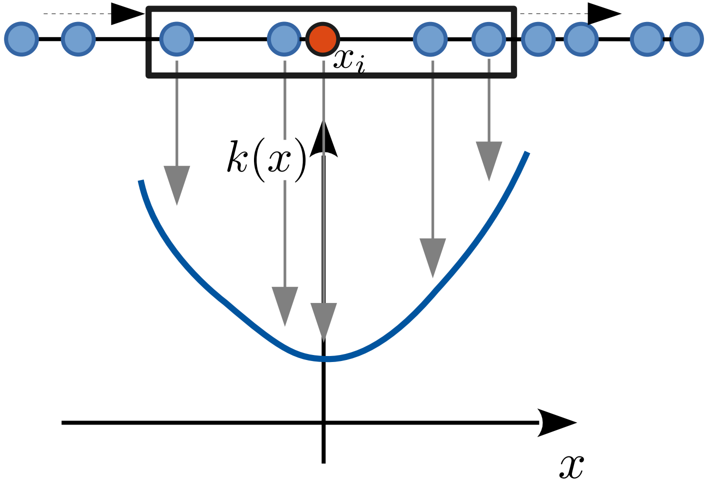
https://arxiv.org/abs/1801.07829

Convolution vs Edge Convolution

discrete grid positions

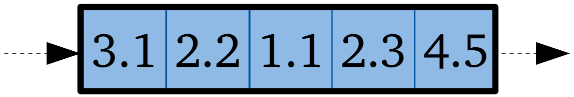


continuous grid positions



discretized kernel:

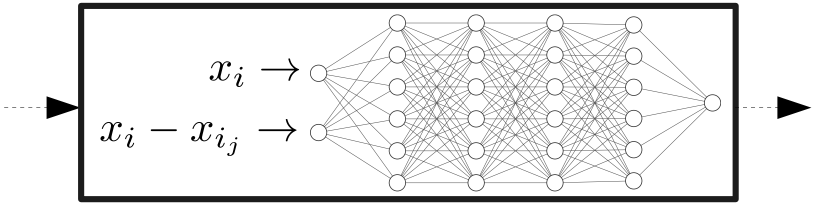
$$k(x)\hat{=}\sum_{k=-p}^{p} W_k x_{i-k}$$



3.1	2.2	1.1	2.3	4.5

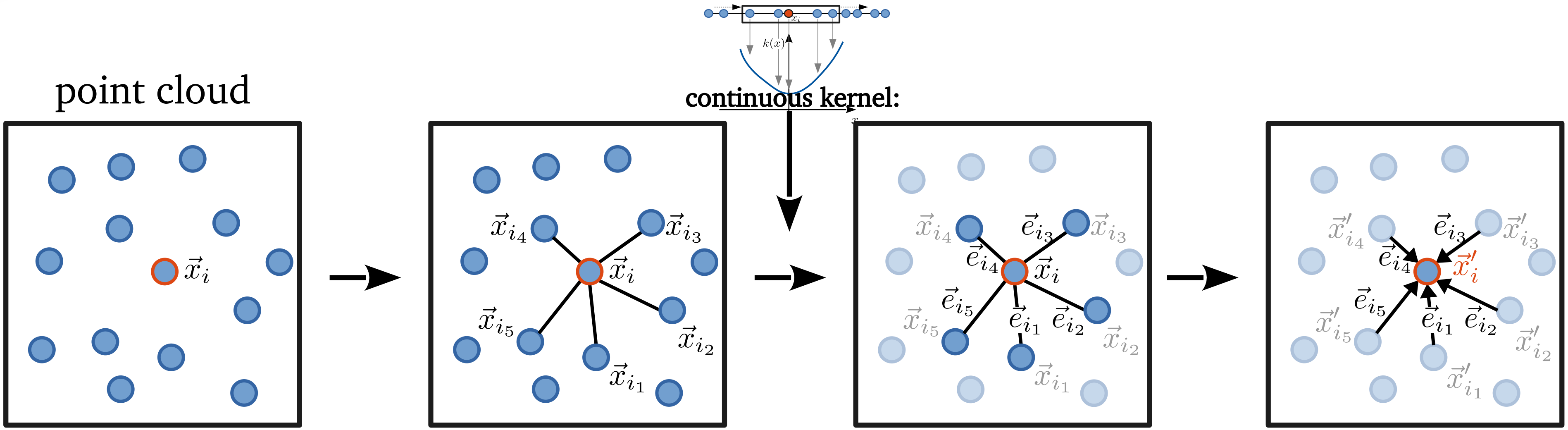
continuous kernel: $k(x)\hat{=}h_\theta(x_i, x_{i_j})$

e.g.: $k(x)\hat{=}h_\theta(x_i, x_i - x_{i_j})$



Deep Learning for Physics Research, World Scientific

Summary: Edge Convolution



construction of directed graph

→ search k nearest neighbors

estimation of edge features

$$\vec{e}_{i_j} = h_\theta(\vec{x}_i, \vec{x}_{i_j})$$

aggregation over neighborhood

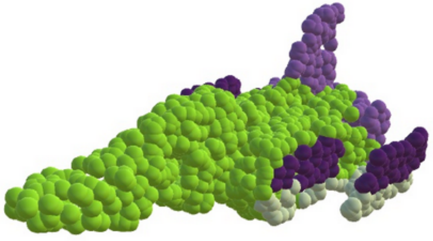
$$\vec{x}'_i = \square_{j=1}^{k} \, \vec{e}_{i_j}$$

e.g. $\vec{x}'_i = \sum_{j=1}^{k} \vec{e}_{i_j}$

Deep Learning for Physics Research, World Scientific

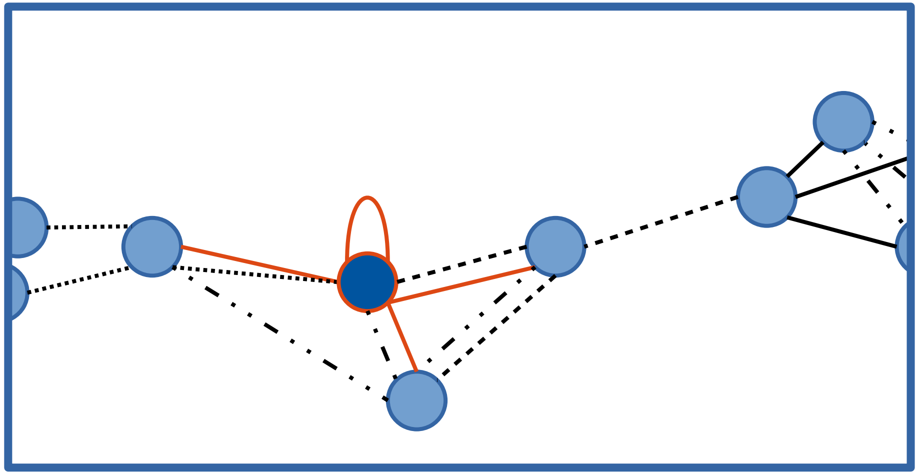
Dynamic Edge Convolution

- Before applying EdgeConv
  - Define underlying graph
- Find neighbors using kNN clustering
  - Smallest euclidean distance in feature space
  - Directed graph



https://arxiv.org/abs/1801.07829

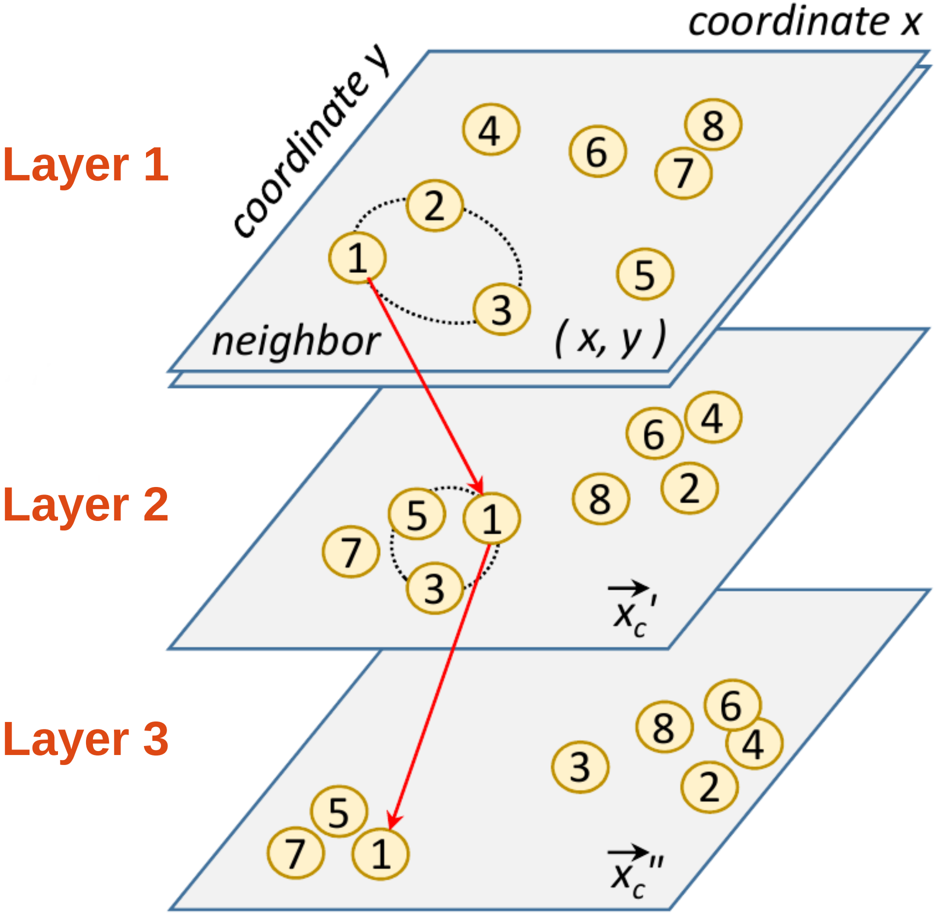
- Edges can be updated in each layer
  - **neighbors change in feature space**
  - *Dynamical* update of graph



Dynamical Graph Update

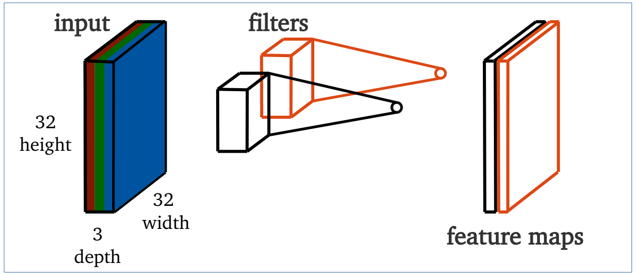
- In each layer neighbors of nodes change
- Update of graph using kNN
- DNN can not directly learn neighbor relations
  - **kNN has no gradient**

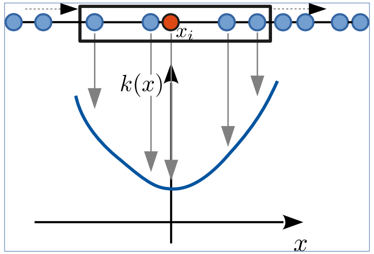
- Implicit clustering of nodes
  - Nodes with same features are embedded similar
    - Become neighbors

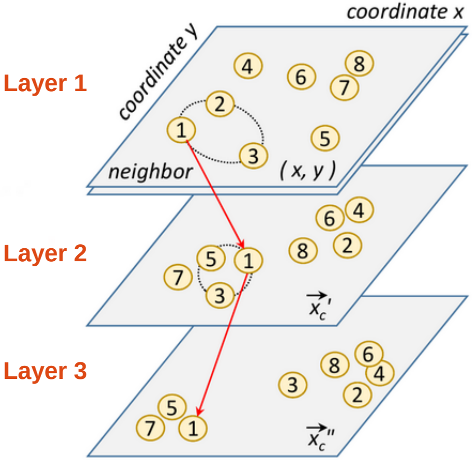


Erdmann et al.: Identification of Patterns in Cosmic-Ray Arrival Directions using Dynamic Graph Convolutional Neural Networks

Convolution vs. Dynamical Convolution







Erdmann et al.: Identification of Patterns in Cosmic-Ray Arrival Directions using Dynamic Graph Convolutional Neural Networks

Similarities:

- Localized convolution
- Operation exploits data structure (translation, rotation, permutation)
  - depends on your chosen $h_\theta(\vec{x}_i, \vec{x}_{i_j})$
  - Weight sharing over pixel positions

Differences:

Image: conv. at positions over features

- Neighbor points **stay** neighbors

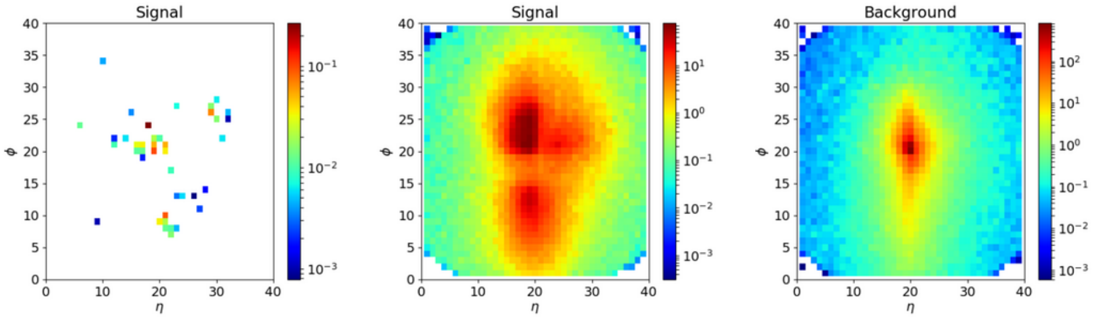
Graph: conv. at features over features

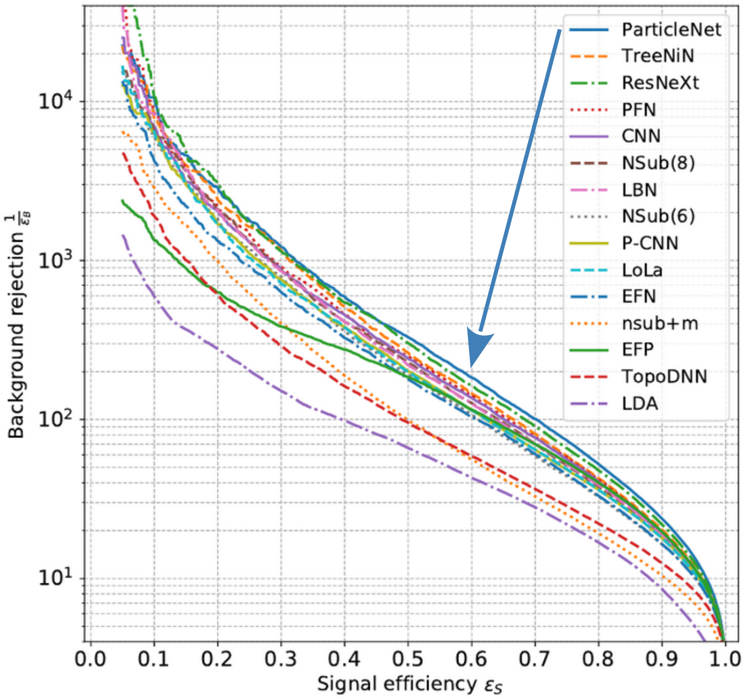
- **Neighbors can change!**

Example: Jet Tagging via Particle Clouds

- Challenge in high-energy physics
- Input: Particle cloud
  - Permutational invariance!
- Classify jets into: **1.** top quarks **2.** background

- ParticleNet won championship
  - Using 3 EdgeConv Layer

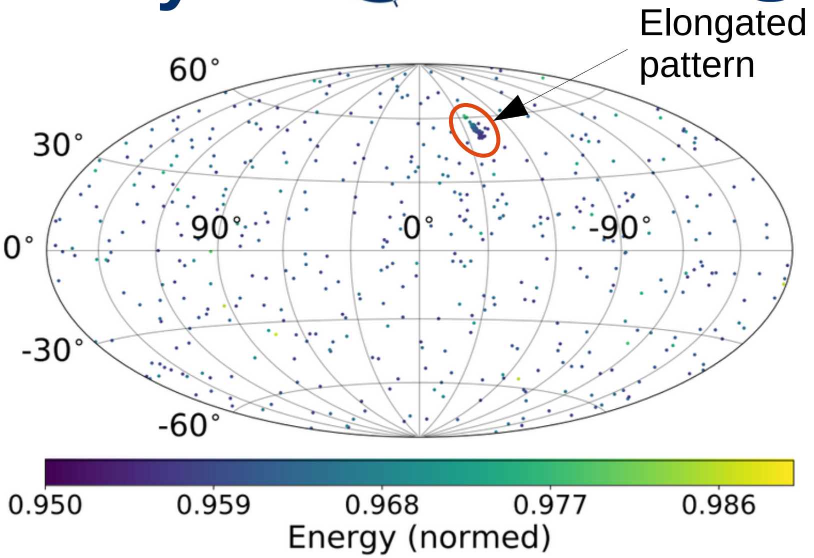




https://arxiv.org/abs/1902.09914

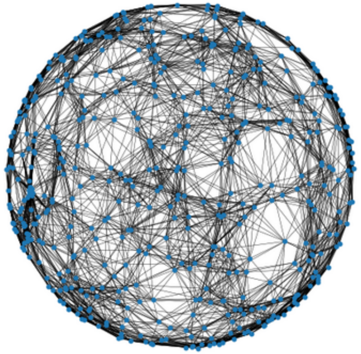
Example: Classification of Cosmic Rays

- Ultra-high energy cosmic rays deflected by galactic magnetic field
- Cosmic rays induce characteristic pattern when arriving at the earth



Task

- Given skymap of 500 cosmic rays
- Using EdgeConvs classify if skymap contains
  I. Signal from single significant source
  II. Only isotropic background



Visualize formed graph in each EdgeConv layer in physics space

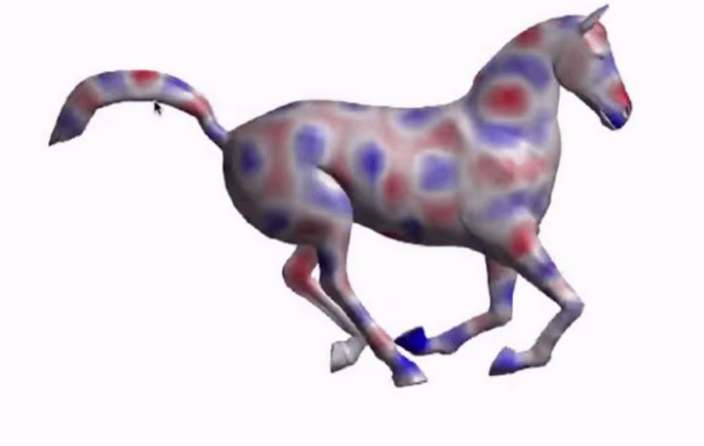
Milestones: Edge Convolutions

Analyze embedded graphs and/or point clouds *(non-Euclidean domains)*
Discussion of Edge Convolutions. Extend CNNs to continuously-distributed data (on non-Euclidean domains), then introduce dynamic graph update.
To simplify: figuratively connect to CNNs.

- ✔ Edge Convolutions similar to CNNs (natural extension to non)
  - ✔ steps: (graph construction, feature estimation, aggregation) (last 2: CNN-like)
  - ✔ discrete kernels (CNNs) → continuous kernel (EdgeConv)
  - ✔ application in parallel not recursively
  - ✔ very flexible (success depends on engineering of kernel & operation)
  - ✔ Dynamic graphs: data with less prior on local correlations (act more globally) no backpropagation through kNN

Convolutions in the Spectral Domain

I. Spectral graph theory
II. Stable and localized filtering
III. Chebychev Convolutions

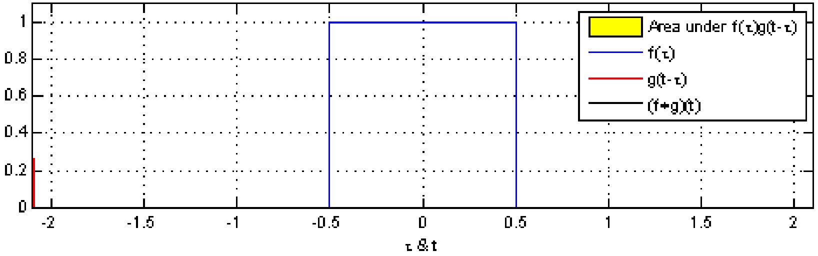


M. Defferrard, X. Bresson, P. Vandergheynst, arXiv:1606.09375

J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, arXiv:1312.6203

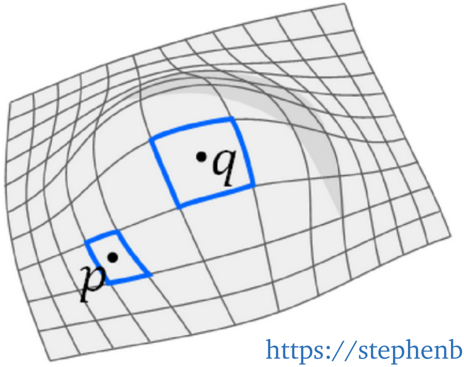
Convolution on non-Euclidean Manifolds

- $$(f * g)(x) := \int_{\mathcal{R}^n} f(\tau)g(x-\tau)\mathrm{d}\tau$$



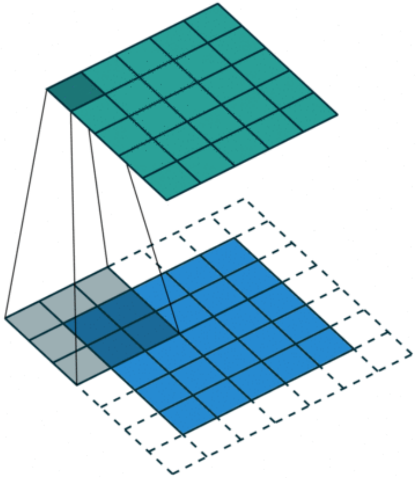
- Convolution has to include curvature of manifold
  - Filters get distorted

- How to convolve?



https://stephenbaek.github.io/projects/zernet/

- How to make it fast?



Paul-Louis Pröve,
Towards Data Science

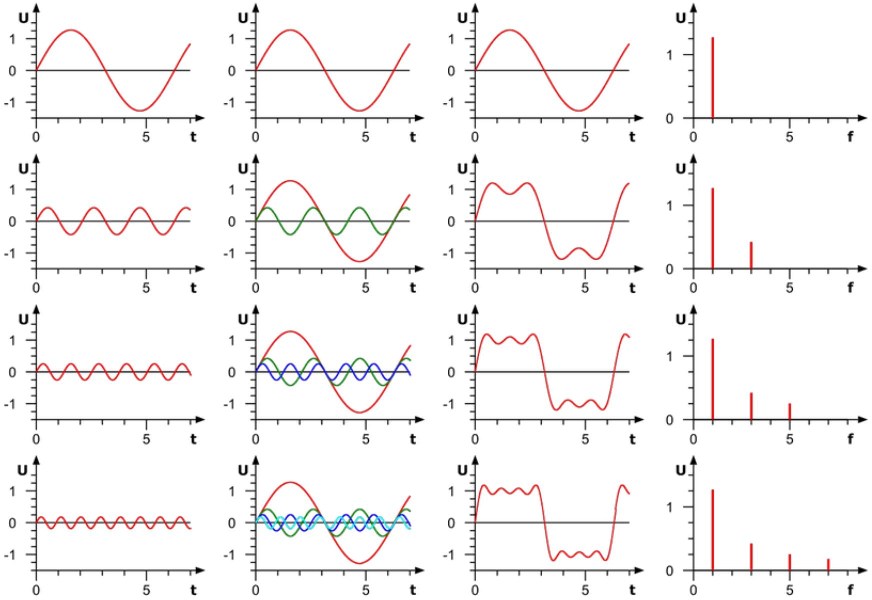
Convolutional Theorem

- Convolution acts pointwise in Fourier domain
  - $\mathcal{F}\{f * g\} = \hat{f} \cdot \hat{g} = \hat{g} \cdot \hat{f}$
  - in Fourier domain matrices are diagonal!

$$\mathcal{F}\{f\} = \hat{f} = \Phi^T f$$

- Accelerate computation
  - $f * g = \Phi(\Phi^T f \cdot \Phi^T g) = \Phi \hat{g} \Phi^T f$

- But need to do Fourier transformation!
  - need eigenvectors of Fourier domain



Graph Laplacian

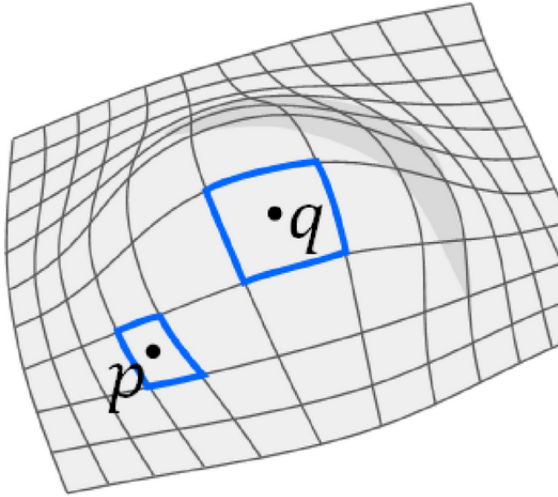
- Laplace matrix L is discrete version of Laplace operator Δ
- Laplace operator encodes smoothness/”curvature” of manifold (2nd derivative)

$$\Delta f = (\nabla \cdot \nabla) f = \operatorname{div}(\operatorname{grad}\ f) = \sum_{k=1}^{n} \frac{\partial^2 f}{\partial x_k^2}$$

Λ = matrix of eigenvalues

Φ = matrix of eigenvectors

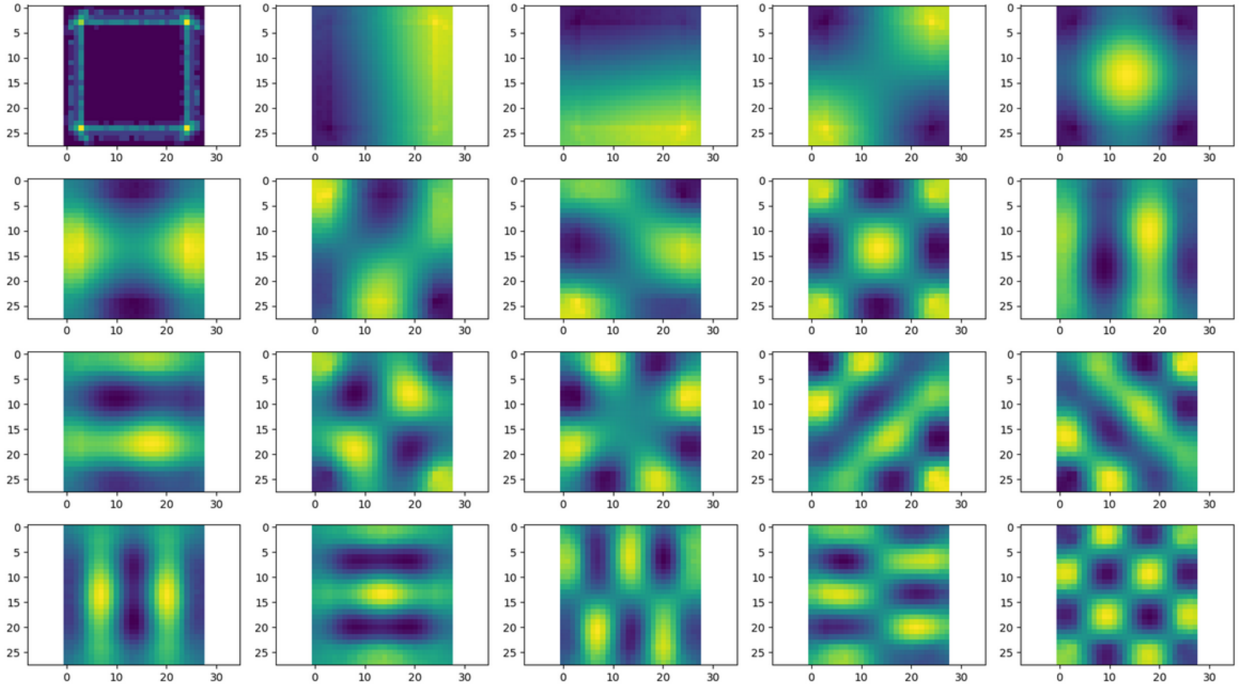
- Eigenfunctions of Laplacian form orthonormal basis
  - $\Delta f = \lambda f$, for graphs $Lf = \lambda f \rightarrow L = \Phi \Lambda \Phi^T$
- Solution directly connected to Fourier space
- Fourier basis = Laplacian eigenvectors/eigenfunctions
  - $-\frac{d^2}{dx^2} \exp^{ikx} = k^2 \exp^{ikx}$



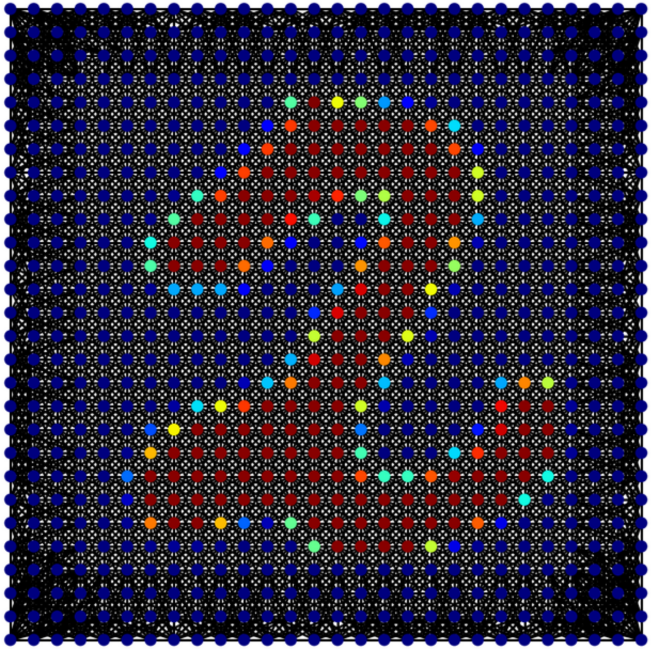
https://stephenbaek.github.io/projects/zernet/

Eigenvectors of Graph Laplacian

- 20 first eigenvectors of L → remember: eigenvectors are also Fourier basis!



representation of Laplacian eigenvectors in spatial domain
→ Fourier modes of graph (modes are not localized!)



- MNIST sample
  Graph k=20

Spectral Convolutions

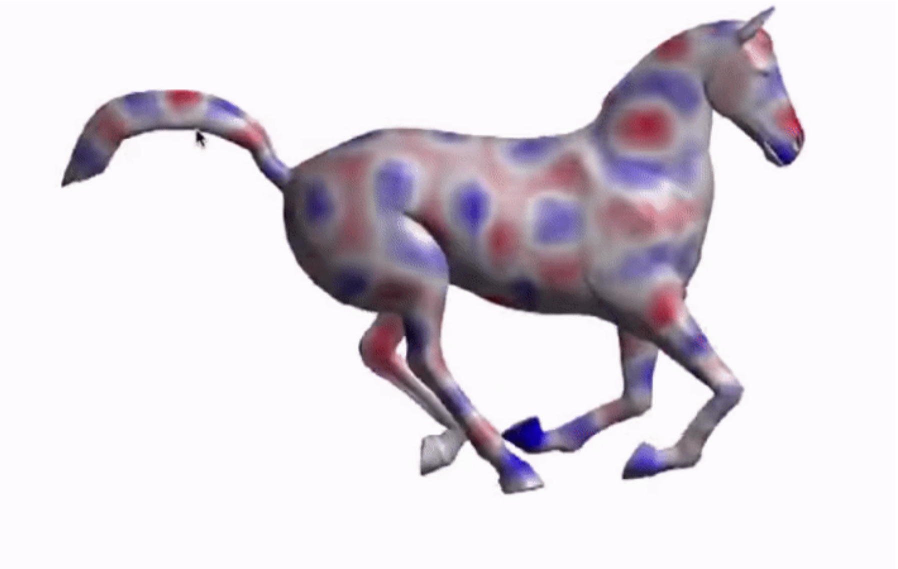
- We can perform the convolution in the spectral domain
  - Signal $X^{(l)}$
  - Weight matrix $W^{(l)}$
- $X^{(l+1)} = \Phi(\Phi^T X^{(l)} \cdot \Phi^T W^{(l)})$

  $= \Phi \hat{W}_\theta^{(l)} \Phi^T X^{(l)}$
- $\hat{W}_\theta^{(l)} = \mathrm{diag}(\theta_1, ..., \theta_n)$

Adaptive parameters in Fourier domain

Problems:

- Weights scale with number of graph nodes
  - act global! No prior on local features!
- $\hat{W}_\theta^{(l)}$ strongly depends on L (Λ, Φ)
  - strong domain dependency → bad generalization performance!



NIPS2017: M. Bronstein, J. Bruna, A. Szlam, X. Bresson, Y. LeCun

Smoothing in Spectral domain

- Approximate $\hat{W}_\theta$ in spectral domain $\tau(L)f = \Phi\tau(\Lambda)\Phi^T f$

$$\Phi(\hat{W}_\theta \Phi^T f) = \Phi \begin{pmatrix} \tau_\theta(\lambda_1) & & \\ & \ddots & \\ & & \tau_\theta(\lambda_n) \end{pmatrix} \Phi^T f$$

- $\hat{W}_\theta \approx \tau_\theta(\lambda) = \sum_{k=1}^{K} \theta_k f_k(\lambda)$

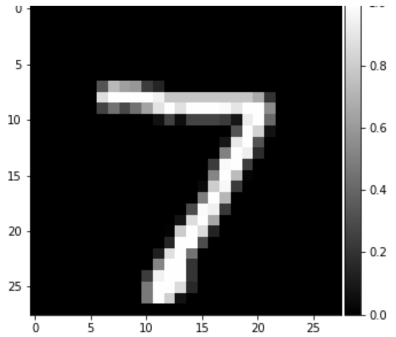
some function

adaptive parameters

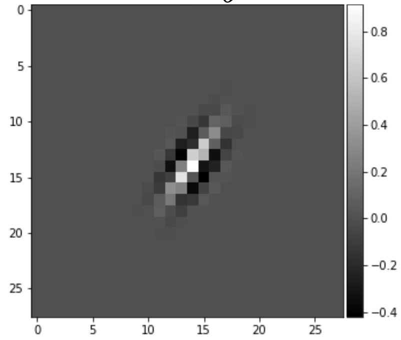
- Learn only K parameters → parameter reduction ✓
- For K << N, $\hat{W}_\theta$ gets smooth in spectral domain ✓
  - Spectral theory: filter become local!

proposed by Bruna et al. https://arxiv.org/abs/1312.6203

MNIST image≙Signal f



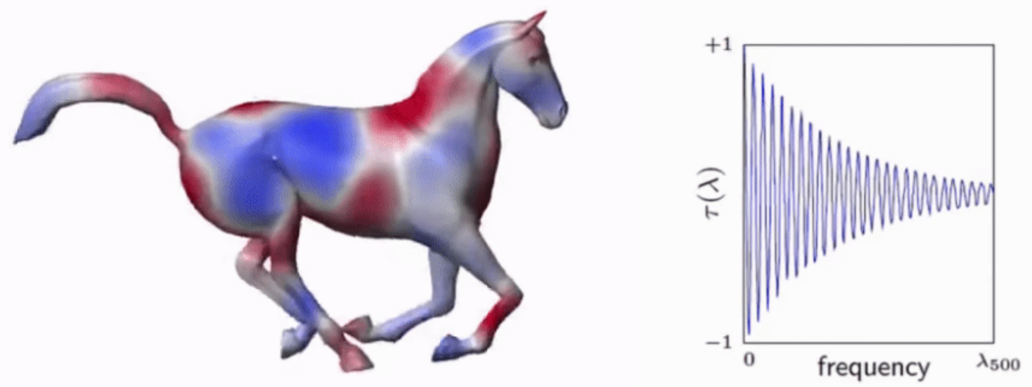
$\Phi\hat{W}_\theta$

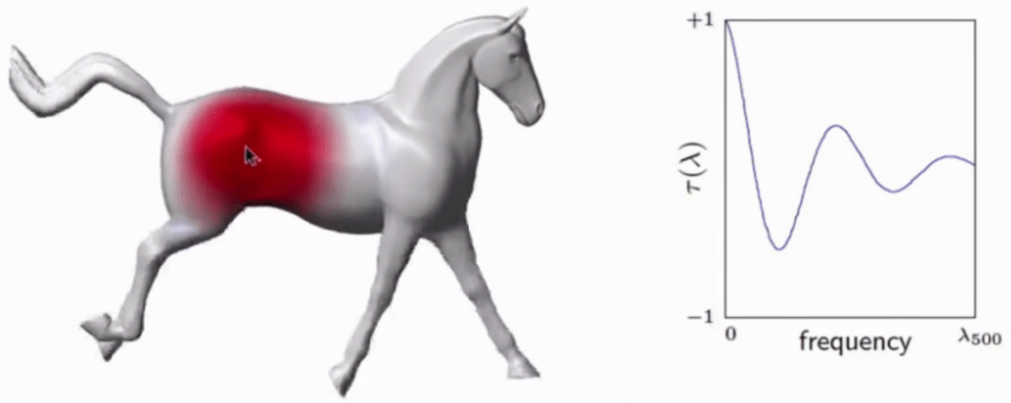


Boris Knyazev, Towards data science

Stable and Localized Filters

Underlying manifold (graph) is changing → change of graph Laplacian





NIPS2017: M. Bronstein, J. Bruna, A. Szlam, X. Bresson, Y. LeCun

- Non-smooth spectral filter
  - Not stable and delocalized

- Smooth spectral filter
  - stable and localized

Chebychev Convolution

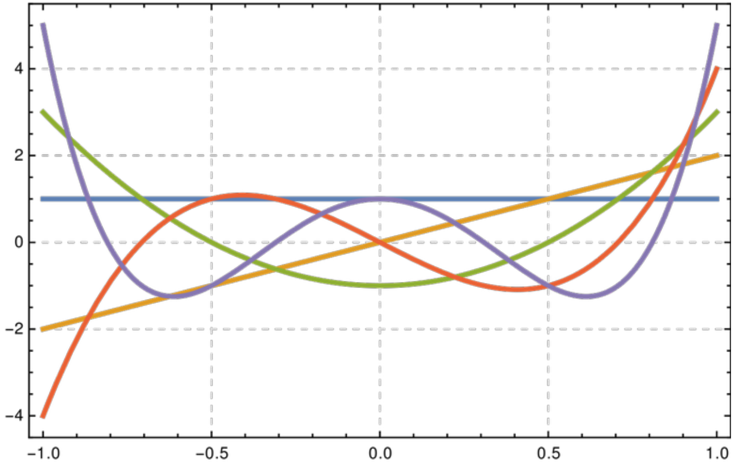
- Use “Chebychev polynomials” for approximation in spectral domain

$$\Phi(\hat{W}_\theta \Phi^T f) = \Phi \hat{W}_\theta(\Lambda) \Phi^T f = \hat{W}_\theta(L) f$$

$$\hat{W}_\theta(L) f \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L}) f \qquad \tilde{L} = \frac{2}{\lambda_{\max}} L - I$$

- Chebychev polynomials are recursively defined
  - $T_{n+1}(x) = 2x\, T_n(x) - T_{n-1}(x)$

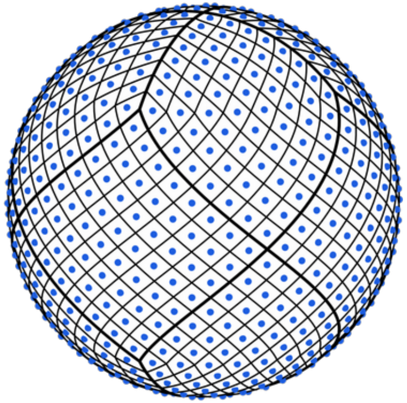
- As $T_0(L) = I,\ T_1(L) = L$
  - Calculate approximation recursive
  - No need for expensive decomposition!

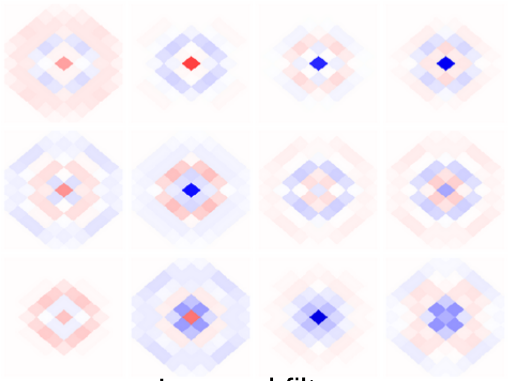


Example: DeepSphere

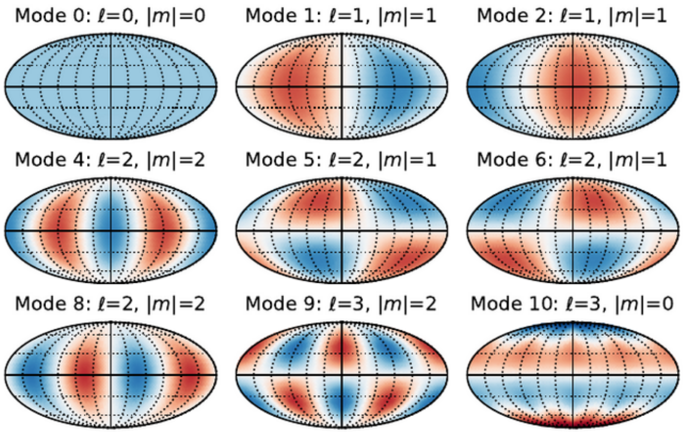
- Convolution on sphere
  - Use pixelization of HEAPix
  - Defines adjacency matrix

- Convolution via Chebychev expansion
  - Framework allows to process spherical data
  - Several properties can be changed
    - but not very modular





Learned filters



Crosscheck: eigenvectors of Laplacian

https://github.com/SwissDataScienceCenter/DeepSphere

https://arxiv.org/abs/1810.12186

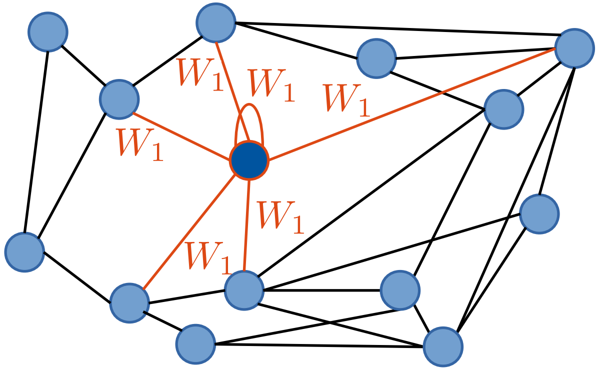
First Order Approximation of ChebNet

- Approximation of Chebychev:

$$g * x = \Phi \hat{g}_\theta \Phi^T x \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L}) x$$

- Evaluate for k=1
  - $g * x \approx \theta_0 x + \theta_1 (L - I) x$, setting $\lambda_{\max} \approx 2$
    $= \theta_0 x - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$
- Setting $\theta_0 = -\theta_1$ and remembering $\hat{A} = I + A$
  - $g * x \approx \theta_1 \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} x$

add self connection



- Propagation rule of GCN (Part I.) $f(X, A) = \sigma\left(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} X W\right)$
  - **GCN is first order approximation of ChebNet!**

Milestones: Spectral Graph Convolutions

Process graph-like structure (e.g., Non-Euclidean domains)
Introduce concept of convolutions on manifolds and think of graphs as an approximation of the manifold. Perform convolution in spectral domain instead spatial domain. Helpful illustration: show Fourier modes of graph in spatial domain

- ✓ Perform convolution in spectral domain (acts point-wise in spectral domain)
  - ✓ eigenvectors (total = number of nodes) of Laplacian are Fourier basis

    “kernels / modes” are not localized and domain dependent
  - ✓ solution I: smooth filters in spectral domain
  - ✓ solution II: perform Chebychev expansion of graph Laplacian
  - ✓ elegant way to define convolutions on Non-Euclidean domains

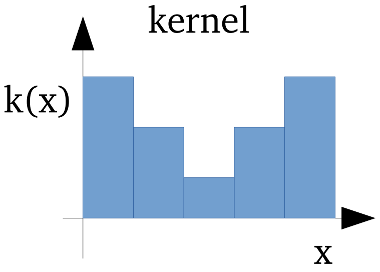
Graph networks and graph convolutional networks

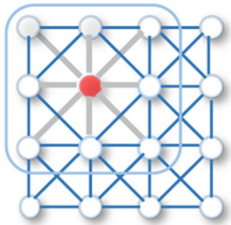
CNNs:

- define convolution on **regular** and **Euclidean** grid (matrix multiplication)
- convolution on special form of graph
- CNNs are <u>very</u> fast
- simple & straight-forward implementation of translational invariance
- straight-forward pooling
- can usually not deal well with sparsity

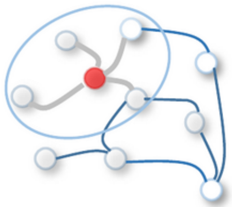
GCNNs:

- define convolution on **graphs**
- very flexible → exploit many symmetries
- can be applied to continuously distributed data → no pixelization (sparse data)
- powerful on non-regular domains
- powerful on non-Euclidean domains
- complex pooling operations
- many versions and implementations
- can be slow (for non-sparse data)

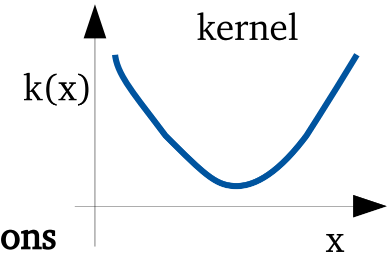


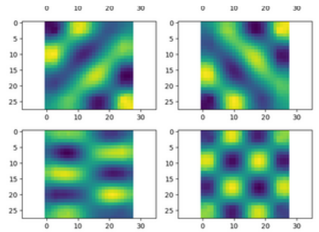


regular grid



Continuous grid positions





Take-Home Message

> "In AI, 'system' should be understood as including the human engineers. Most of the 'data → generalization' conversion happens during model design." - F. Chollet

After starting with standard methods (FCN, CNN, RNN)

- Is your model able to exploit all symmetries in data?
  - are important features missing?
  - is architecture supporting the underlying data structure (e.g. various sensors)
- ➢ Choose architecture which best fits for your symmetry!

- ➢ Graph Convolutional Networks are very flexible
  - ➢ powerful option for complex data structures
  - ➢ BUT: expect no improvements for simple/regular, e.g., image-like data!

Milestones: Graph Networks

How to exploit structured, graph-like, non-regular, non-Euclidean data?

1. Introduce graphs: nodes, edges, and adjacency matrix (Laplacian)

2. Perform convolution on simple bidirectional graph (social network) → GCN

3. Extend convolutions to embedded graphs (discrete → continuous kernel)

4. Perform convolutions in Fourier domain (spectral convolutions)

Complex mathematical framework, interesting: GCN is 1st order of ChebNet

For illustration: try to add many figurative examples

✔ Idea: Complicated data → construct graphs to define meaningful convolutions
→ reduce parameters by setting prior on local correlations / underlying symmetry

✔ perform graph convolution in spatial domain (filters localized in space)

✔ perform graph convolution in spectral domain (filter learned in Fourier domain)

✔ **Exploit underlying symmetry of given data**, expert knowledge needed!

Links & Resources

[1] M. Erdmann, J. Glombitza, G. Kasieczka, U. Klemradt, Deep Learning for Physics Research, World Scientific, 2021

[2] Francois Chollet: Deep Learning with Python, MANNING PUBLICATIONS

[3] Deep Learning (Goodfellow, Bengio, Courville), MIT Press, ISBN: 0262035618

[4] An Introduction to different Types of Convolutions in Deep Learning, Paul-Louis Pröve

https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

[5] Michael M. Bronstein et al. : Geometric deep learning: going beyond Euclidean data: ArXiv:1611.08097

[6] Thomas Kipf, Max Welling: ArXiv:1609.02907

[7] M. Defferrard, X. Bresson, P. Vandergheynst: ArXiv:1606.09375

[8] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun: ArXiv:1312.6203

[9] Y. Wang et al.: ArXiv:1801.07829

[10] M. Simonovsky, N. Komodakis: ArXiv:1704.02901

[11] E. Hoogeboom, J. Peters, T. Cohen, M. Welling: ArXiv/1803.02108

[12] Boris Knyazev, Towards data science, Tutorial on Graph Neural Networks for Computer Vision and Beyond

[13] M. Bronstein, J. Bruna, A .Szlam, X. Bresson, Y. LeCun: Tutorial Geometric Deep Learning on Graphs and Manifolds, https://www.youtube.com/watch?v=LvmjbXZyoP0&t=3813s, NIPS2017

ERLANGEN CENTRE FOR ASTROPARTICLE PHYSICS

FAU

Graph Neural Networks

HANDS-ON

Jonas Glombitza

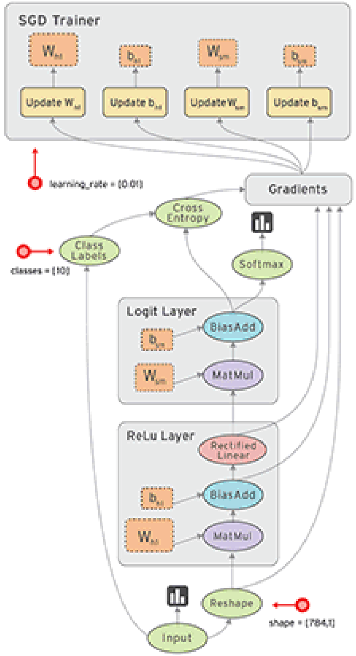
http://www.deeplearningphysics.org/

TensorFlow

“Open source software library for numerical computation using data flowing graphs”

- **Nodes** represent mathematical operations
- **Graph edges** represent multi dimensional data arrays (**tensors**) which **flow** through the graph

- Supports:
  - CPUs and **GPUs**
  - Desktops and mobile devices
- Released 2015, stable since Feb. 2017
- Developer: Google Brain



Keras

- Will use Keras in this tutorial (TensorFlow backend) - https://keras.io
  - High-level neural networks API, written in Python
- Concise syntax with many reasonable default settings
- Useful callbacks for monitoring the training procedure
- Nice Documentation & many examples and tutorials + useful extensions
- Ships with TensorFlow

- We use tf.keras 2.2.4-tf // TensorFlow 2.1

Additional Software

- We use Spektral in this tutorial, version 0.2.0
- Python library for deep learning on graphs
- Based on Keras and TensorFlow

Spektral

https://github.com/danielegrattarola/spektral

- Alternative for PyTorch users:

PyTorch geometric

https://github.com/rusty1s/pytorch_geometric

NetworkX

- For visualization of graphs we use NetworkX

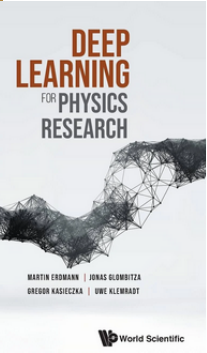
Tryout Deep Learning Yourself!

Find many physics examples at:

http://www.deeplearningphysics.org/

For example:

- CNNs, RNNs, **GCNs**
- GANs and WGANs
- Anomaly detection, Denosing AEs
- Visualization & introspection and more


DEEP LEARNING FOR PHYSICS RESEARCH
MARTIN ERDMANN | JONAS GLOMBITZA
GREGOR KASIECZKA | UWE KLEMRADT
World Scientific

Tutorial

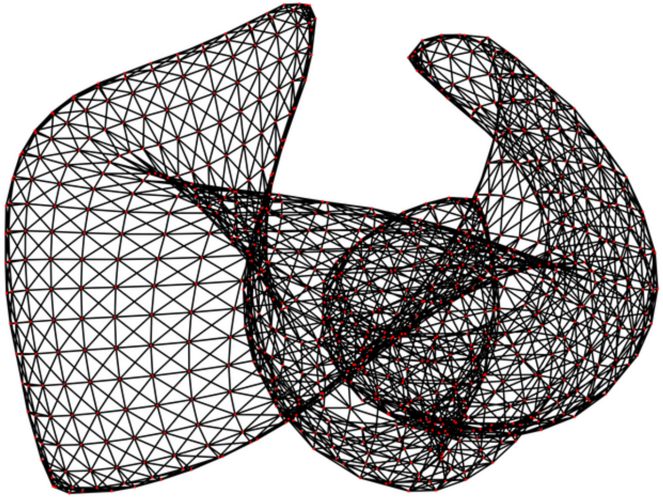
- Open exercise page
  https://github.com/DeepLearningForPhysicsResearchBook/deep-learning-physics/
- open Colab link and login with your Google Account

- **Exercise 10.1:**
  Semi-supervised node-classifcation

  Open in Colab

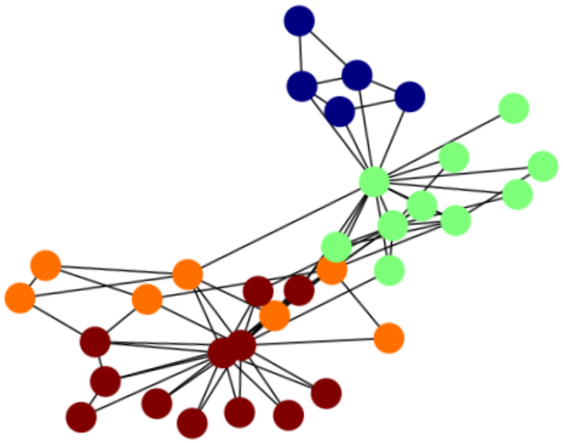
- **Exercise 16.1:**
  Classification of cosmic rays

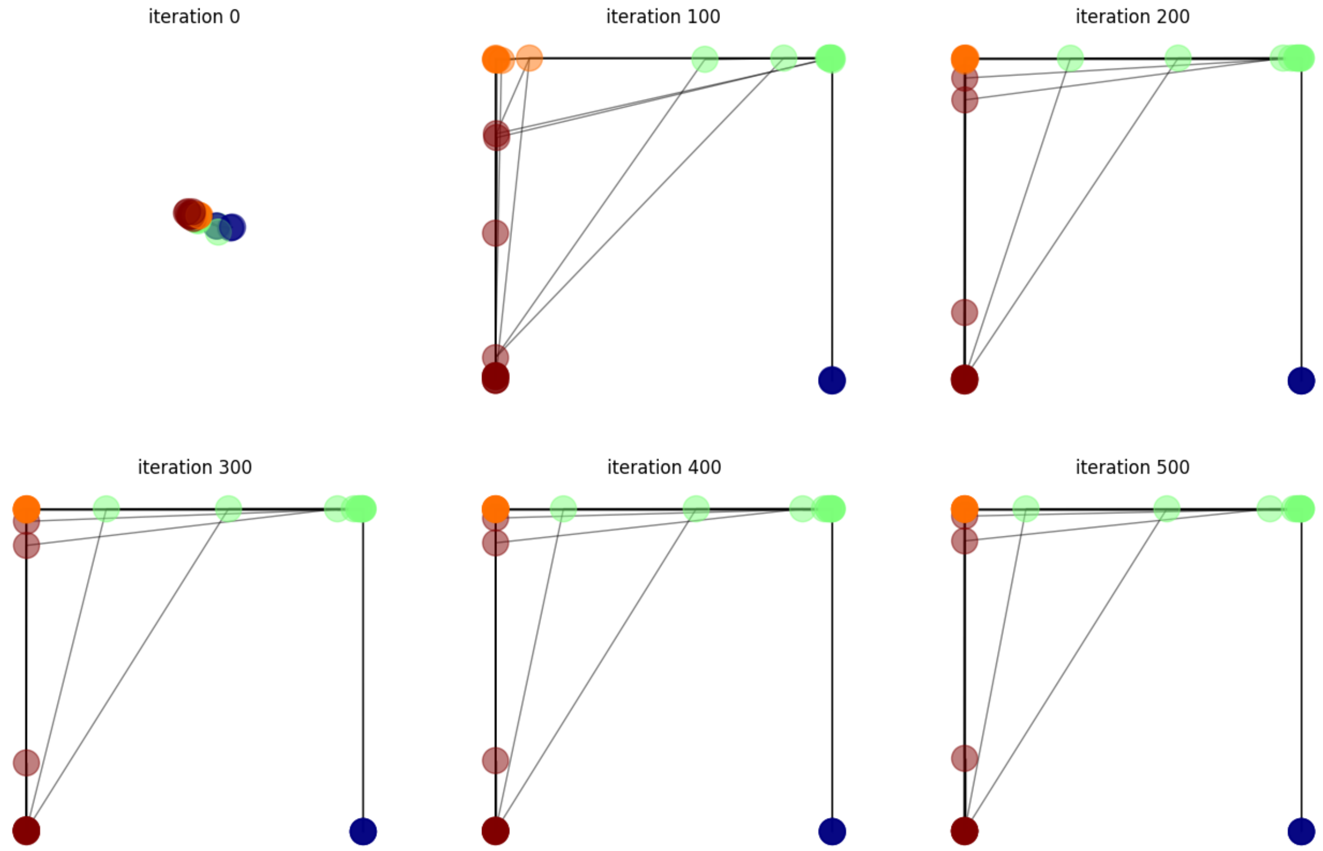
  Open in Colab



Practice 1 – Karate Club Network

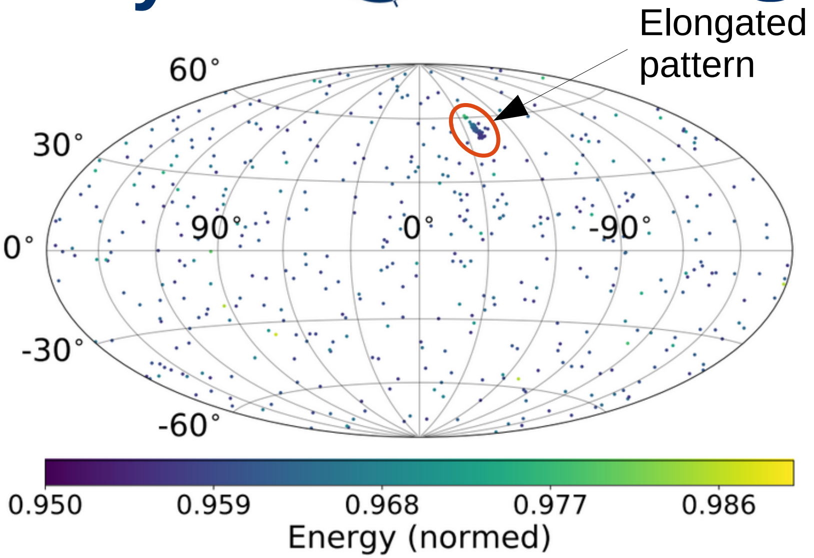
- Tune learning rate
- Increase iterations
- Well connected labels





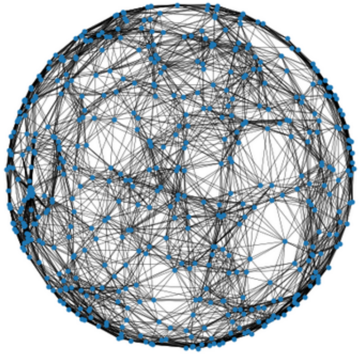
Example Classification of Cosmic Rays

- Ultra-high energy cosmic rays deflected by galactic magnetic field
- Cosmic rays induce characteristic pattern when arriving at the earth



Task

- Given skymap of 500 cosmic rays
- Using EdgeConvs classify if skymap contains
  I. Signal from single significant source
  II. Only isotropic background



Visualize formed graph in each EdgeConv layer in physics space

Helpful Comments on the Code

- Modify kernel network → change $h_{\Theta}()$ of EdgeConv

```
def  kernel_nn(data, nodes=16):
    d1, d2 = data  # get xi ("central" pixel) and xj ("neighborhood" pixels)
    dif = layers.Subtract()([d1, d2])
    x = layers.Concatenate(axis=-1)([d1, dif])
    x = layers.Dense(nodes, use_bias=False, activation="relu")(x)
    x = layers.BatchNormalization()(x)
    return  x
```

Dynamic + fixed graph updates

- Used *fixed* graph by passing in each layer the very first points_input

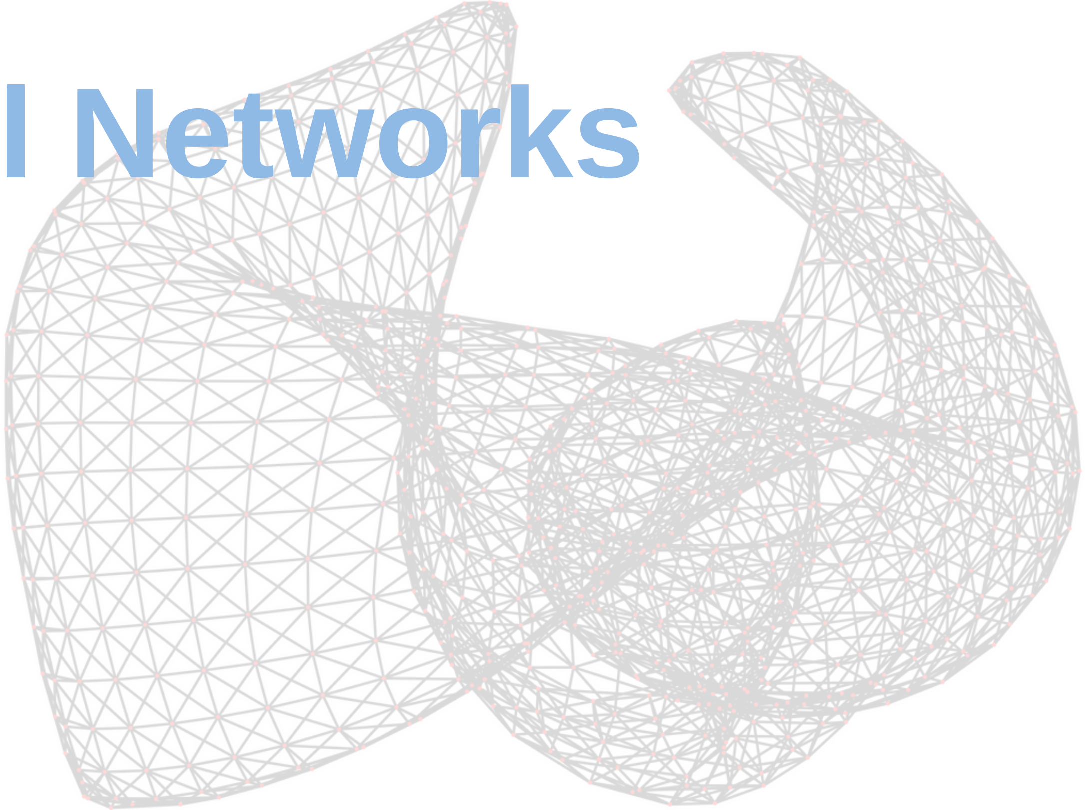
```
x = EdgeConv(lambda a: kernel_nn(a, nodes=8), next_neighbors=5)([points_input, feats_input])
```

- Use *dynamic* graph update by passing only produced feature dimension x

```
x = EdgeConv(lambda a: kernel_nn(a, nodes=16), next_neighbors=8)(x)
```

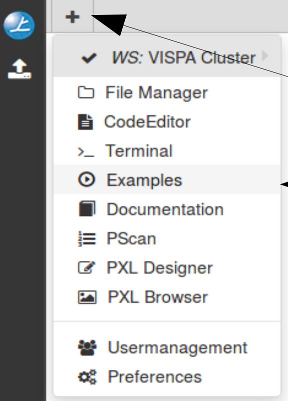
Graph Neural Networks

Backup



Jonas Glombitza, Martin Erdmann

Deep Learning Train-the-Trainer workshop, Wuppertal, 9 - 10 June 2022

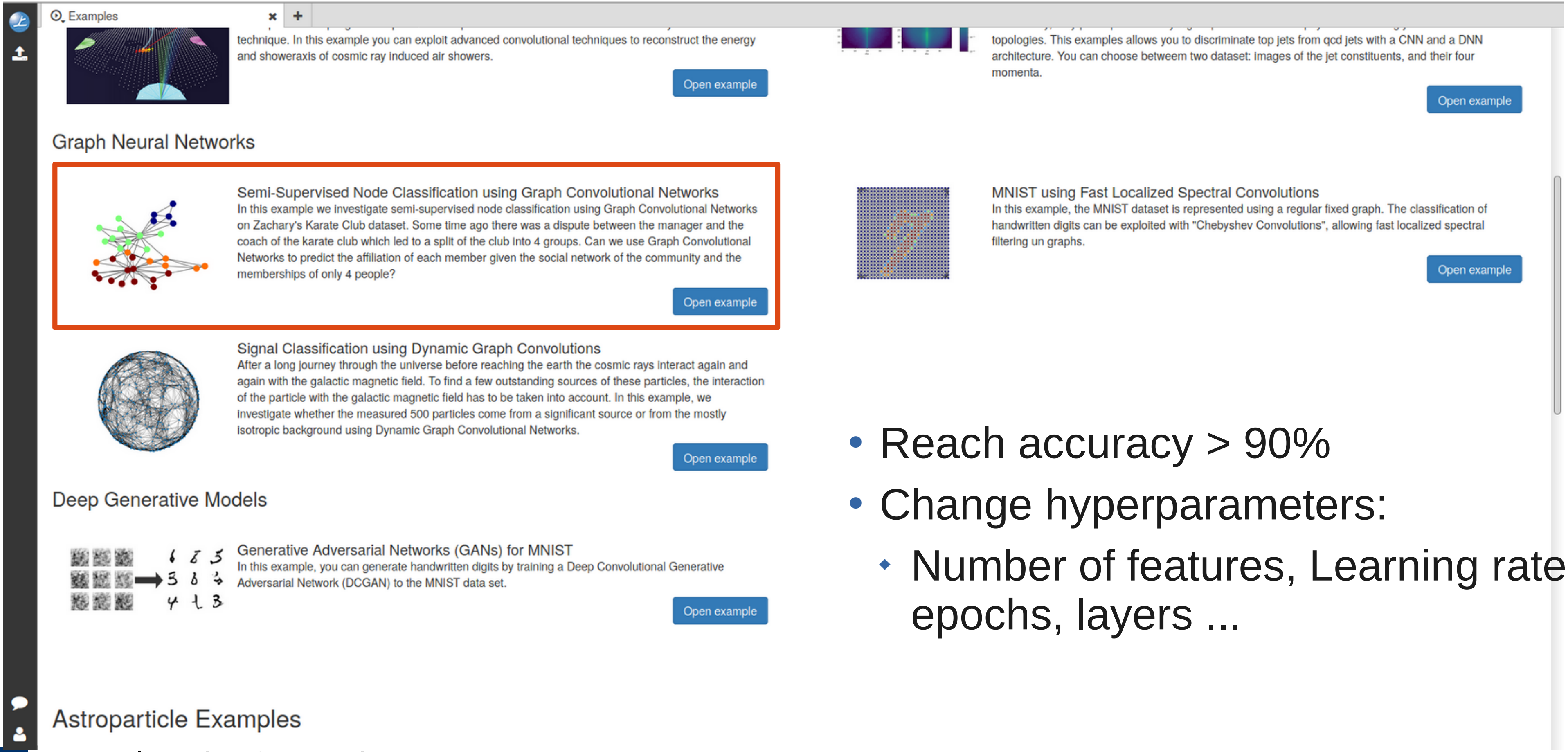


Opens the example page

VISPA

- Developed in Aachen (group of Martin Erdmann)
- GPU extension
  - 20x NVIDIA GTX 1080
  - 3x RTX 6000, 6x RTX 5000
- Accessible via https://vispa.physik.rwth-aachen.de/

Practice I



technique. In this example you can exploit advanced convolutional techniques to reconstruct the energy and showeraxis of cosmic ray induced air showers.

topologies. This examples allows you to discriminate top jets from qcd jets with a CNN and a DNN architecture. You can choose betweem two dataset: images of the jet constituents, and their four momenta.

Graph Neural Networks

Semi-Supervised Node Classification using Graph Convolutional Networks

In this example we investigate semi-supervised node classification using Graph Convolutional Networks on Zachary's Karate Club dataset. Some time ago there was a dispute between the manager and the coach of the karate club which led to a split of the club into 4 groups. Can we use Graph Convolutional Networks to predict the affiliation of each member given the social network of the community and the memberships of only 4 people?

MNIST using Fast Localized Spectral Convolutions

In this example, the MNIST dataset is represented using a regular fixed graph. The classification of handwritten digits can be exploited with "Chebyshev Convolutions", allowing fast localized spectral filtering un graphs.

Signal Classification using Dynamic Graph Convolutions

After a long journey through the universe before reaching the earth the cosmic rays interact again and again with the galactic magnetic field. To find a few outstanding sources of these particles, the interaction of the particle with the galactic magnetic field has to be taken into account. In this example, we investigate whether the measured 500 particles come from a significant source or from the mostly isotropic background using Dynamic Graph Convolutional Networks.

Deep Generative Models

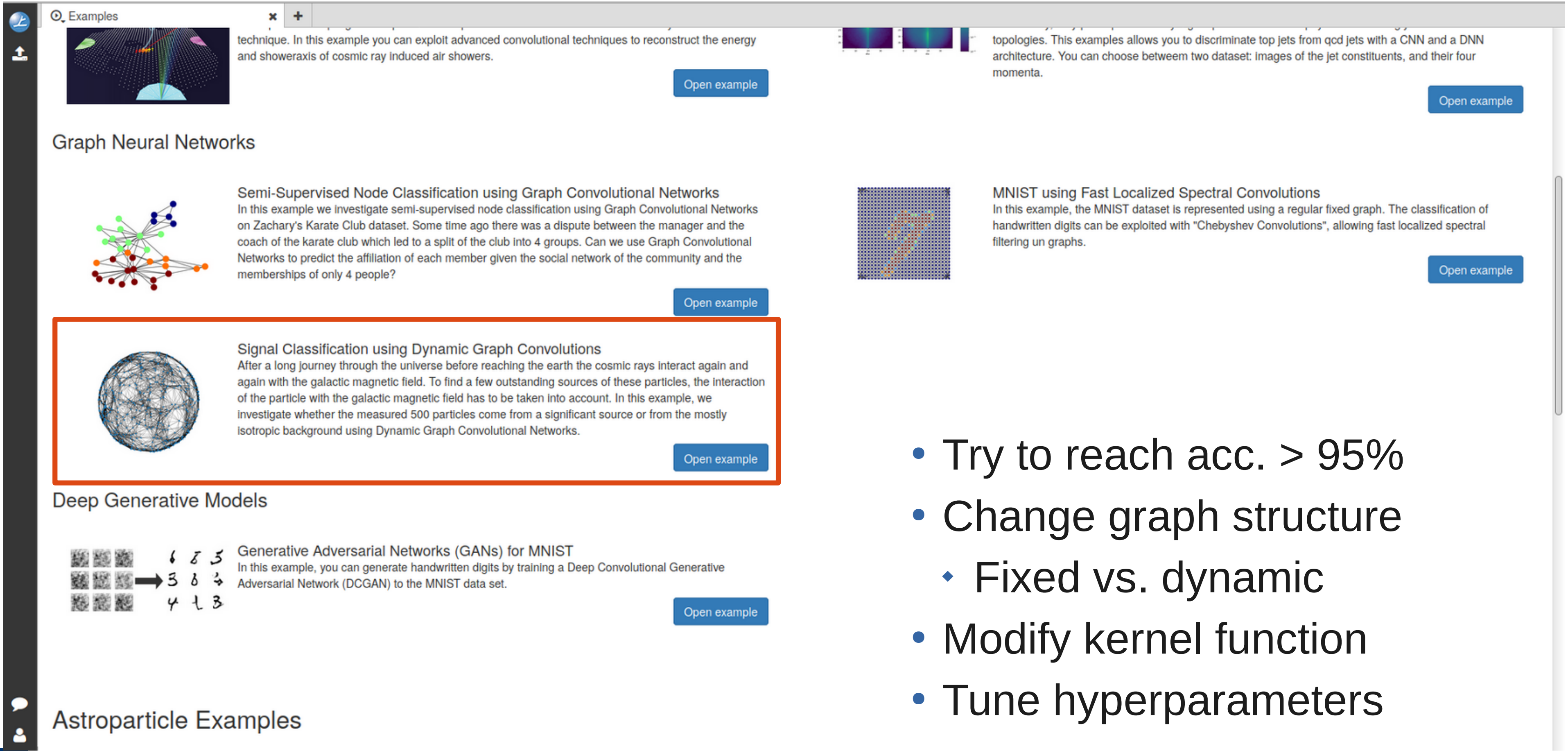
Generative Adversarial Networks (GANs) for MNIST

In this example, you can generate handwritten digits by training a Deep Convolutional Generative Adversarial Network (DCGAN) to the MNIST data set.

Astroparticle Examples

- Reach accuracy > 90%
- Change hyperparameters:
  - Number of features, Learning rate, epochs, layers ...

Practice II



technique. In this example you can exploit advanced convolutional techniques to reconstruct the energy and showeraxis of cosmic ray induced air showers.

topologies. This examples allows you to discriminate top jets from qcd jets with a CNN and a DNN architecture. You can choose betweem two dataset: images of the jet constituents, and their four momenta.

Graph Neural Networks

Semi-Supervised Node Classification using Graph Convolutional Networks
In this example we investigate semi-supervised node classification using Graph Convolutional Networks on Zachary's Karate Club dataset. Some time ago there was a dispute between the manager and the coach of the karate club which led to a split of the club into 4 groups. Can we use Graph Convolutional Networks to predict the affiliation of each member given the social network of the community and the memberships of only 4 people?

MNIST using Fast Localized Spectral Convolutions
In this example, the MNIST dataset is represented using a regular fixed graph. The classification of handwritten digits can be exploited with "Chebyshev Convolutions", allowing fast localized spectral filtering un graphs.

Signal Classification using Dynamic Graph Convolutions
After a long journey through the universe before reaching the earth the cosmic rays interact again and again with the galactic magnetic field. To find a few outstanding sources of these particles, the interaction of the particle with the galactic magnetic field has to be taken into account. In this example, we investigate whether the measured 500 particles come from a significant source or from the mostly isotropic background using Dynamic Graph Convolutional Networks.

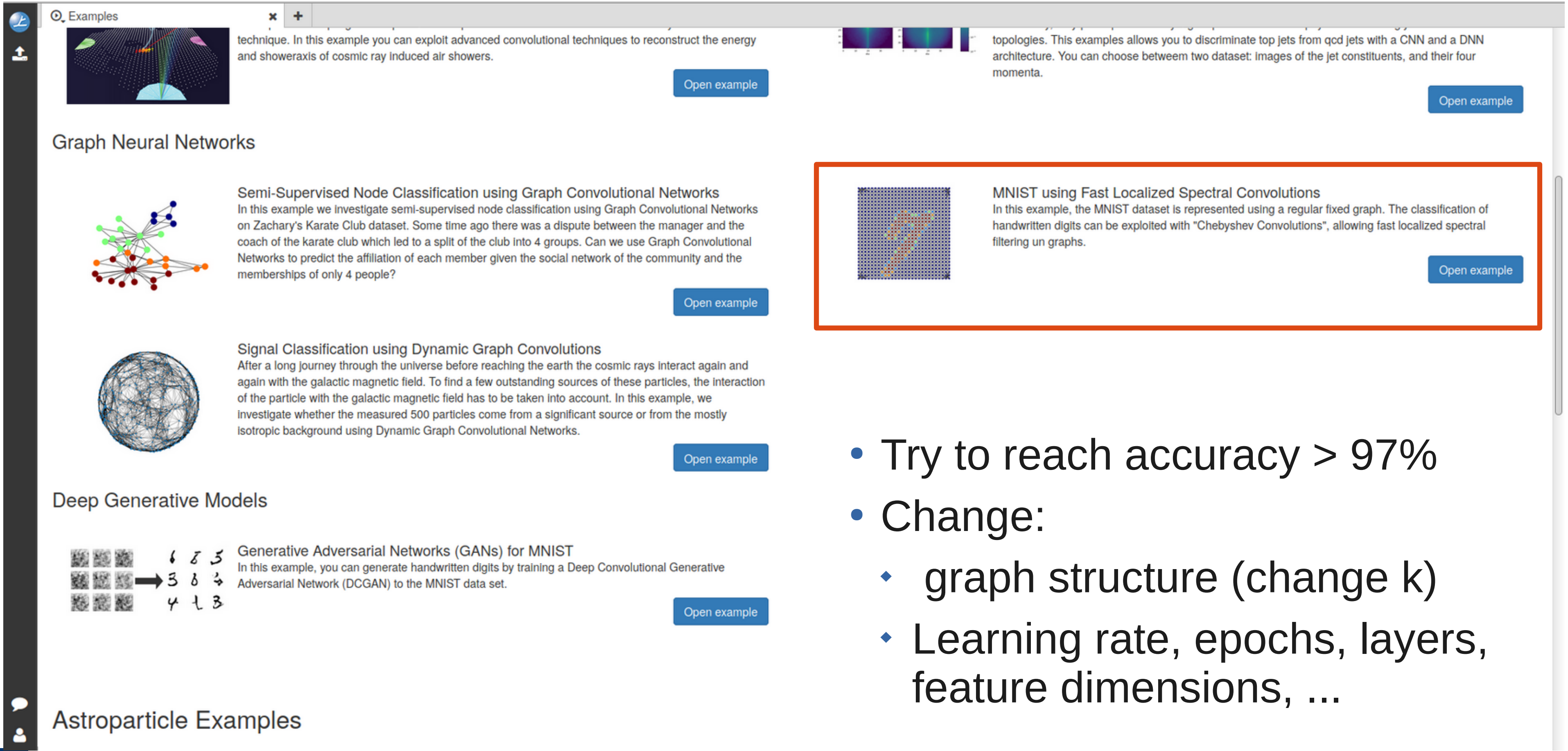
Deep Generative Models

Generative Adversarial Networks (GANs) for MNIST
In this example, you can generate handwritten digits by training a Deep Convolutional Generative Adversarial Network (DCGAN) to the MNIST data set.

Astroparticle Examples

- Try to reach acc. > 95%
- Change graph structure
  - Fixed vs. dynamic
- Modify kernel function
- Tune hyperparameters

Practice III



technique. In this example you can exploit advanced convolutional techniques to reconstruct the energy and showeraxis of cosmic ray induced air showers.

topologies. This examples allows you to discriminate top jets from qcd jets with a CNN and a DNN architecture. You can choose betweem two dataset: images of the jet constituents, and their four momenta.

Graph Neural Networks

Semi-Supervised Node Classification using Graph Convolutional Networks

In this example we investigate semi-supervised node classification using Graph Convolutional Networks on Zachary's Karate Club dataset. Some time ago there was a dispute between the manager and the coach of the karate club which led to a split of the club into 4 groups. Can we use Graph Convolutional Networks to predict the affiliation of each member given the social network of the community and the memberships of only 4 people?

MNIST using Fast Localized Spectral Convolutions

In this example, the MNIST dataset is represented using a regular fixed graph. The classification of handwritten digits can be exploited with "Chebyshev Convolutions", allowing fast localized spectral filtering un graphs.

Signal Classification using Dynamic Graph Convolutions

After a long journey through the universe before reaching the earth the cosmic rays interact again and again with the galactic magnetic field. To find a few outstanding sources of these particles, the interaction of the particle with the galactic magnetic field has to be taken into account. In this example, we investigate whether the measured 500 particles come from a significant source or from the mostly isotropic background using Dynamic Graph Convolutional Networks.

Deep Generative Models

Generative Adversarial Networks (GANs) for MNIST

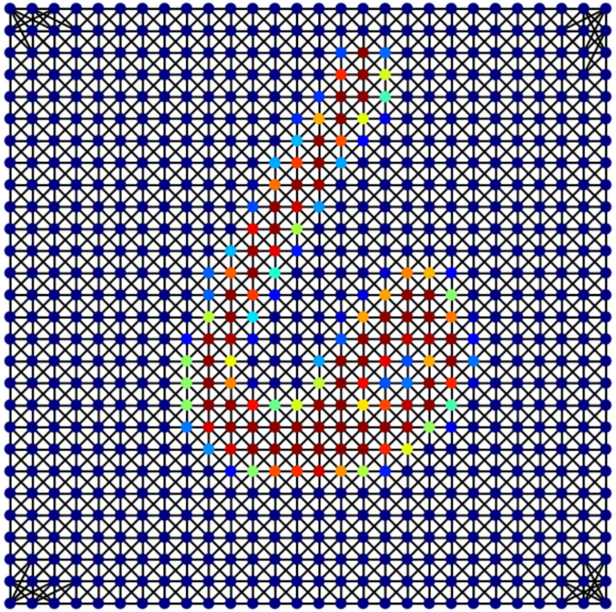
In this example, you can generate handwritten digits by training a Deep Convolutional Generative Adversarial Network (DCGAN) to the MNIST data set.

Astroparticle Examples

- Try to reach accuracy > 97%
- Change:
  - graph structure (change k)
  - Learning rate, epochs, layers, feature dimensions, ...

Example MNIST

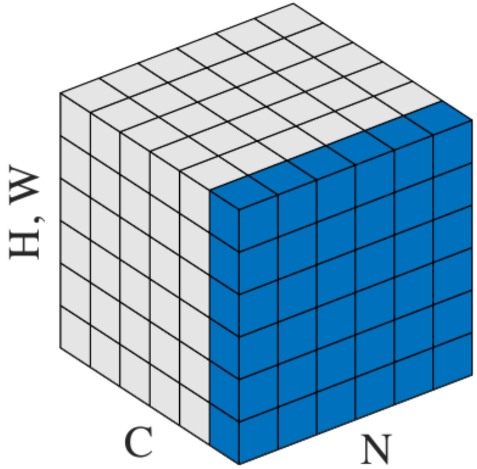
- Projection of MNIST on graph
  - Each nodes has 8 neighbors (kNN clustering)
  - Fixed domain (adjacency matrix fixed)
- Use ChebNet to classify handwritten digits
- MNIST
  - 10 classes
  - Training 50k samples
  - Testing 10 samples



Batch Normalization

- Calculate batch-wise for each channel:
  - Mean: μ_B
  - Variance: σ_B^2
  - Add free parameters $\gamma,\ \beta$ to change scale and mean

➢ $$y = \frac{x - \mu_B}{\sigma_B}\gamma + \beta$$



- Makes DNN robust against poor initializations
- Helps with vanishing gradient / less sensitive to high learning rates
- Has regularizing effect (no large weights, noise because of batch dependency)
- Reduce internal covariate shift

➢ **Very successful for convolutional architectures**

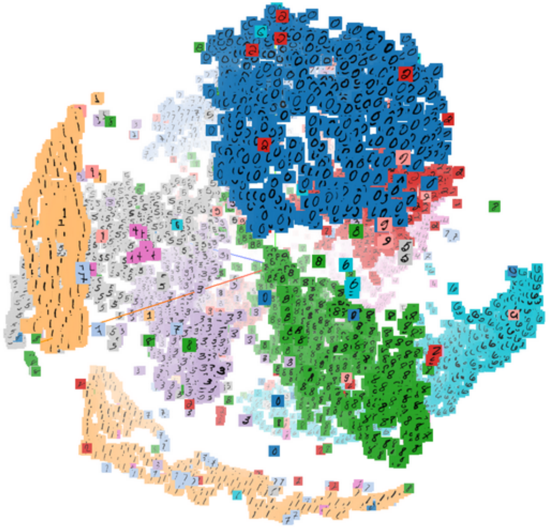
Embedding

- To visualize machine learning models
- Project vectors of high dimensional space on low dimensional manifold

- Good classifier need high separation capability
  - especially at latest layers

- Most simple embedding
  - Neural network layer with 2 dimensional uttput

3D embedding of MNIST

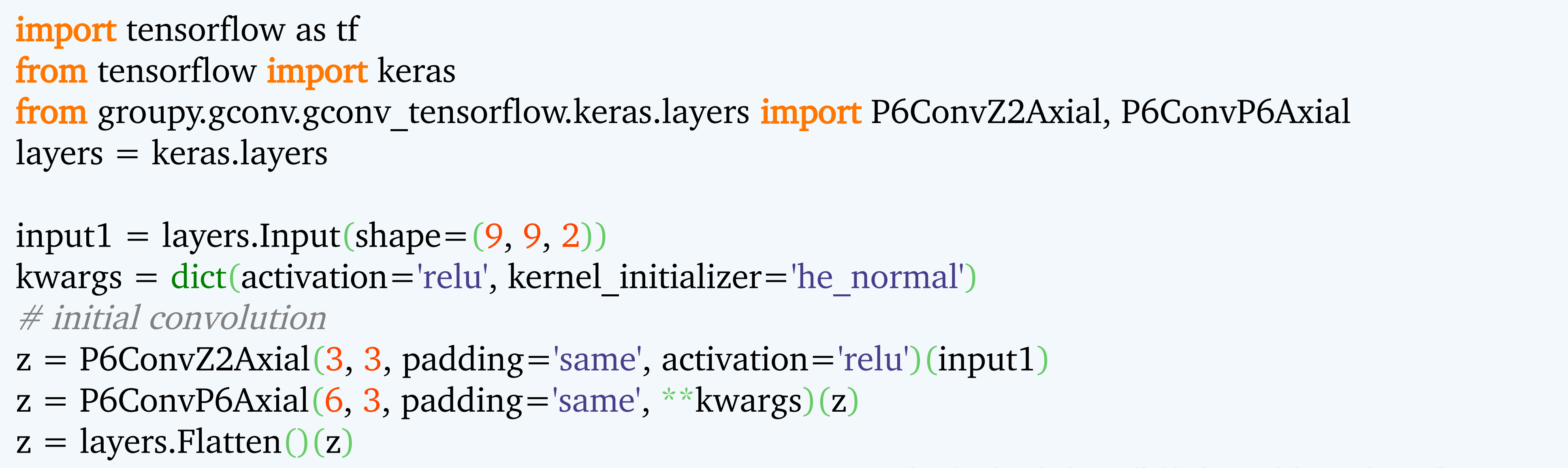


https://projector.tensorflow.org/

```
x = GraphConv(2, activation='tanh', name="embedding")([x, fltr_in])
```

Code Example

- Filter-size 3: → 7 adaptive parameter
- Filter-size 5: → 19 adaptive parameters
- Need data in axial coordinates
- Beta implementation of keras / tf layers by Lukas Geiger



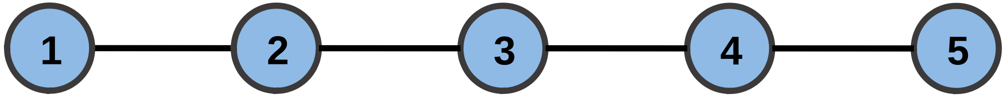
```python
import tensorflow as tf
from tensorflow import keras
from groupy.gconv.gconv_tensorflow.keras.layers import P6ConvZ2Axial, P6ConvP6Axial
layers = keras.layers

input1 = layers.Input(shape=(9, 9, 2))
kwargs = dict(activation='relu', kernel_initializer='he_normal')
# initial convolution
z = P6ConvZ2Axial(3, 3, padding='same', activation='relu')(input1)
z = P6ConvP6Axial(6, 3, padding='same', **kwargs)(z)
z = layers.Flatten()(z)
```

Check GitHub: https://github.com/ehoogeboom/hexaconv

Graph Convolution

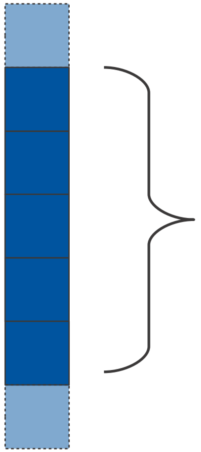
- Convolutional layers are special case of Graph convolutional layers



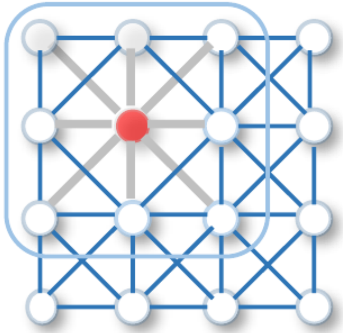
$$H^{(l+1)} = f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)})$$

$$shape(H^{(l)}) = N \times F$$
$$shape(A) = N \times N$$
$$shape(W) = D \times F$$

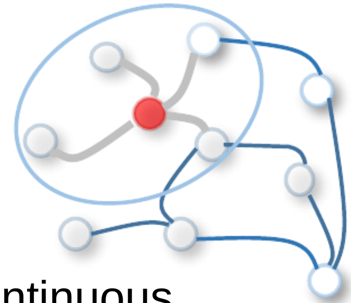
$$\begin{array}{c} A \\ \begin{bmatrix} 0&0&0&0&0&0&0 \\ 0&1&1&0&0&0&0 \\ 0&1&1&1&0&0&0 \\ 0&0&1&1&1&0&0 \\ 0&0&0&1&1&1&0 \\ 0&0&0&0&1&1&0 \\ 0&0&0&0&0&0&0 \end{bmatrix} \end{array} \times \begin{array}{c} H \\ \begin{bmatrix} 0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0 \\ 0&0&0&1&0&0&0 \\ 0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0 \end{bmatrix} \end{array} \times \begin{array}{c} W \\ \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \end{bmatrix} \end{array} =$$



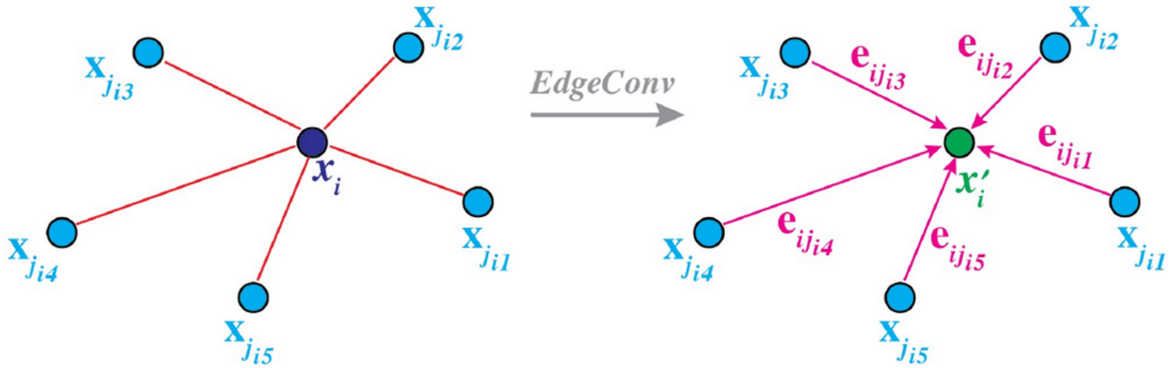
Summary: Dynamical Graph Convolution



Discrete
grid positions



Continuous
grid positions



Input: size x (features)

1. Search **k next neighbors**
2. **Convolve** signals
   → size x (k, channels)
3. **Aggregate** signals
   → size x (channels)

→ Repeat if you want

$$\boldsymbol{x}'_i = \mathop{\square}_{j=1}^{k} \boldsymbol{h}_{\boldsymbol{\Theta}}(\boldsymbol{x}_i, \boldsymbol{x}_{i_j}) = \bar{\boldsymbol{h}}_{\boldsymbol{\Theta}}(\boldsymbol{x}_i, \boldsymbol{x}_{i_j} - \boldsymbol{x}_i),$$

Use DNN

$$x_i = \sum_j \mathrm{ReLU}(\boldsymbol{\theta}_m \cdot (\mathbf{x}_j - \mathbf{x}_i) + \boldsymbol{\phi}_m \cdot \mathbf{x}_i),$$

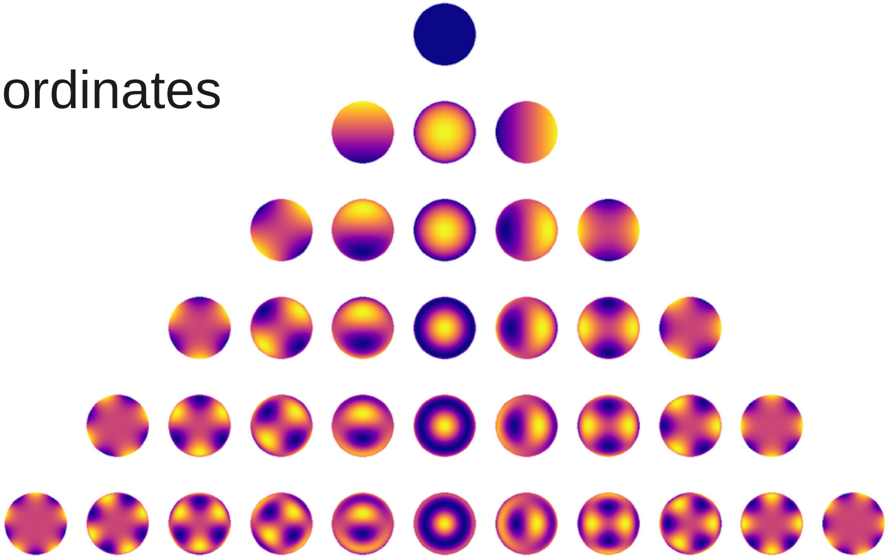
Example: Spherical Harmonics

$$\left(\frac{\partial^2}{\partial\vartheta^2}+\frac{\cos\vartheta}{\sin\vartheta}\frac{\partial}{\partial\vartheta}+\frac{1}{\sin^2\vartheta}\frac{\partial^2}{\partial\varphi^2}\right)Y_{lm}(\vartheta,\varphi)=-l(l+1)Y_{lm}(\vartheta,\varphi)$$

- e.g. Schrödinger’s equation for hydrogen atom
  - angular component breaks down to $\hat{\mathbf{L}}^2=-\hbar^2\Delta_{\theta,\varphi}$

- Eigenfunctions of Laplacian in spherical coordinates
  - $\Delta_{\theta,\phi}f=\lambda f$

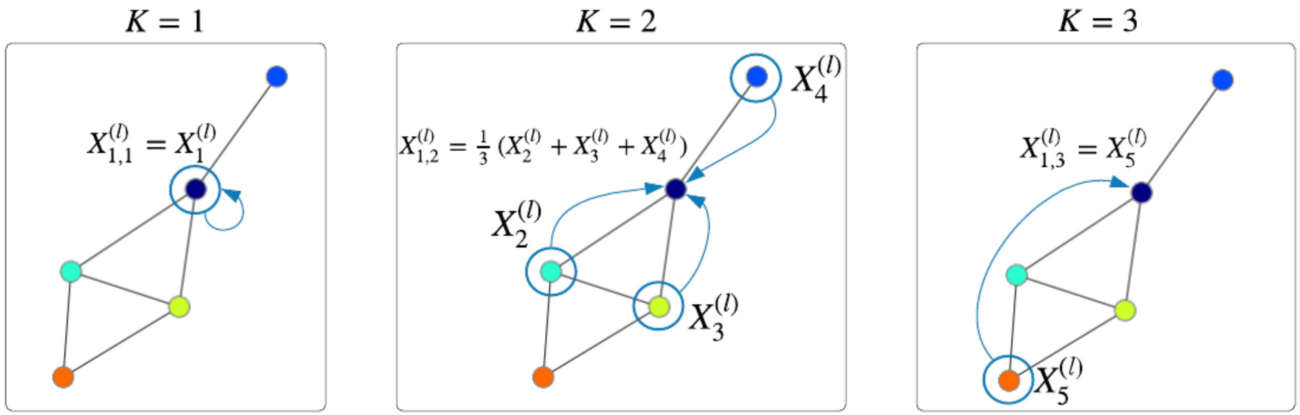
- Spherical harmonics
  - complete and orthonormal set of eigenfunctions of angular component



https://rodluger.github.io/

Illustrative Chebychev expansion

- Using the Chebychev expansion can be seen as
  - $\sum_{k=0}^{K-1} \theta_k A^k$, weighting the neighborhood with the adjacency matrix
- Precise A^k: element ij= number of walks of length n from node i to node j


$K = 1$
$X_{1,1}^{(l)} = X_1^{(l)}$
$K = 2$
$X_{1,2}^{(l)} = \frac{1}{3}(X_2^{(l)} + X_3^{(l)} + X_4^{(l)})$
$X_4^{(l)}$
$X_2^{(l)}$
$X_3^{(l)}$
$K = 3$
$X_{1,3}^{(l)} = X_5^{(l)}$
$X_5^{(l)}$

Result of the Chebyshev convolution: $X_1^{(l+1)} = [X_{1,1}^{(l)}, X_{1,2}^{(l)}, X_{1,3}^{(l)}]W^{(l)}$

Boris Knyazev, Towards Data Science

First Order Approximation

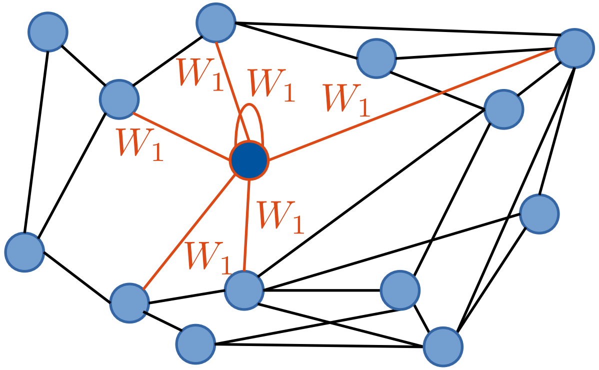
- Approximation of Chebychev:

$$g * x = \Phi \hat{g}_\theta \Phi^T x \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L}) x$$

- Evaluate for k=1
  - $g * x \approx \theta_0 x + \theta_1 (L - I) x$, setting $\lambda_{\max} \approx 2$

    $= \theta_0 x - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$
- Setting $\theta_0 = -\theta_1$ and remembering $\hat{A} = I + A$
  - $g * x \approx \theta_1 \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} x$

add self connection



- Propagation rule of GCN (Part I.) $f(X, A) = \sigma\left(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} X W\right)$
  - **GCN is first order approximation of ChebNet!**