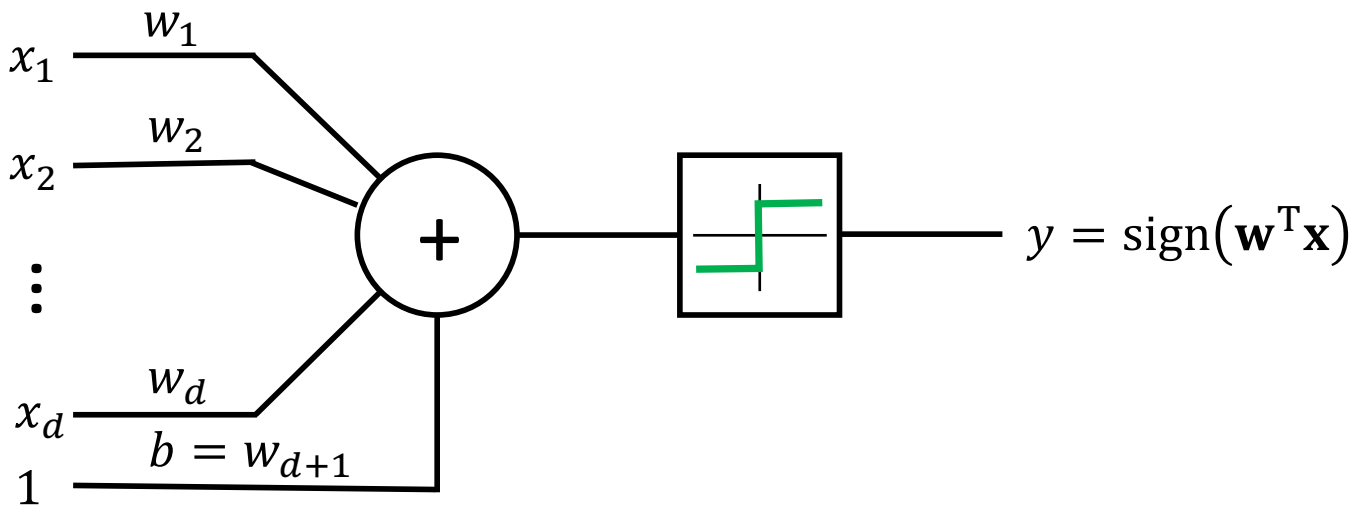
Geometric Deep Learning

going beyond Euclidean data

Michael Bronstein

Imperial College London / Twitter

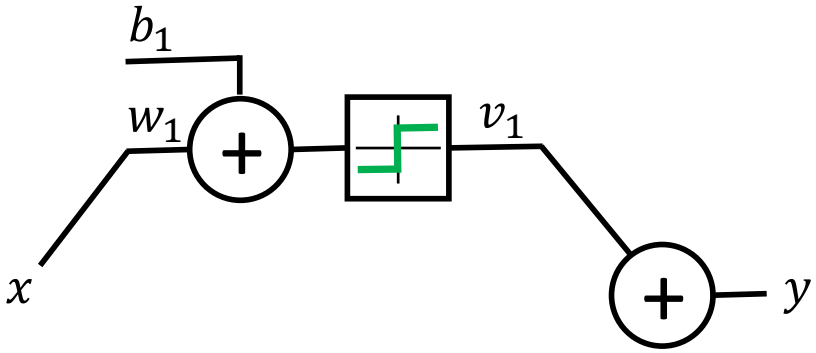
Perceptron



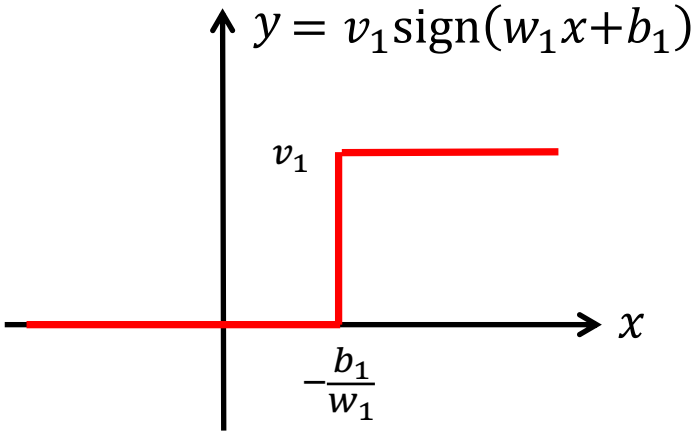
simplest neural network

Rosenblatt 1957

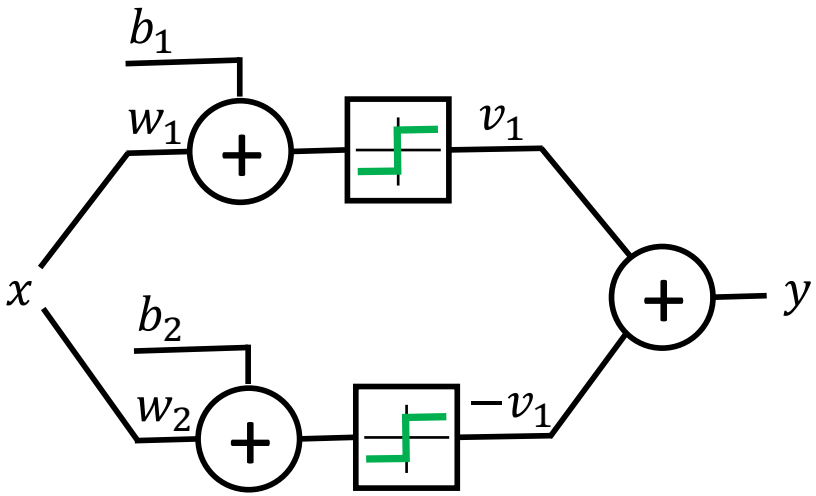
Multilayer perceptron

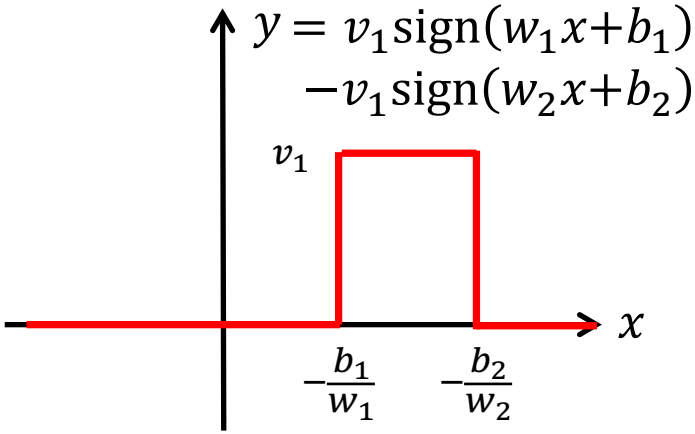


two-layer perceptron

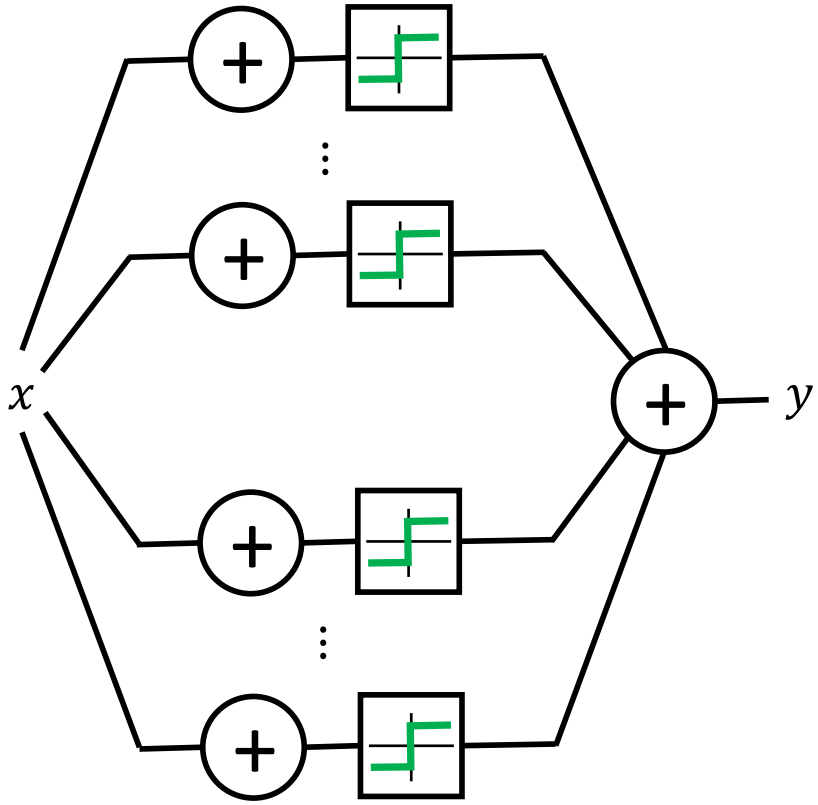


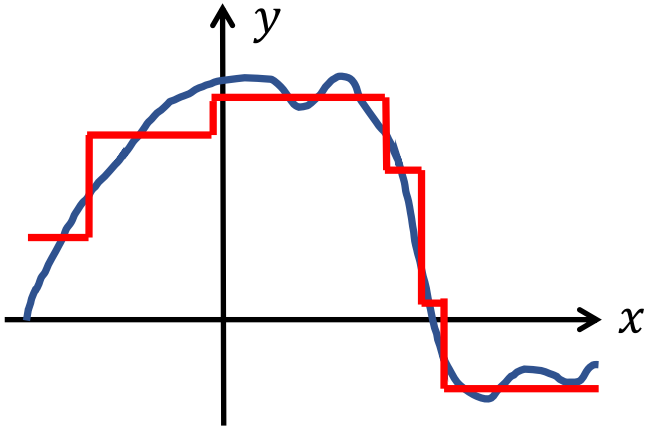
Multilayer perceptron





Multilayer perceptron = universal approximator



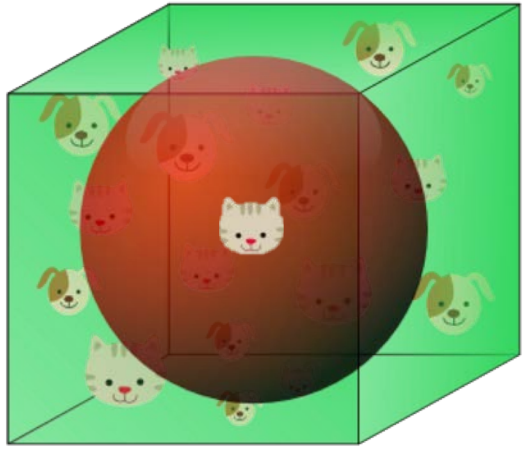


multilayer perceptron can approximate a continuous function to any desired accuracy

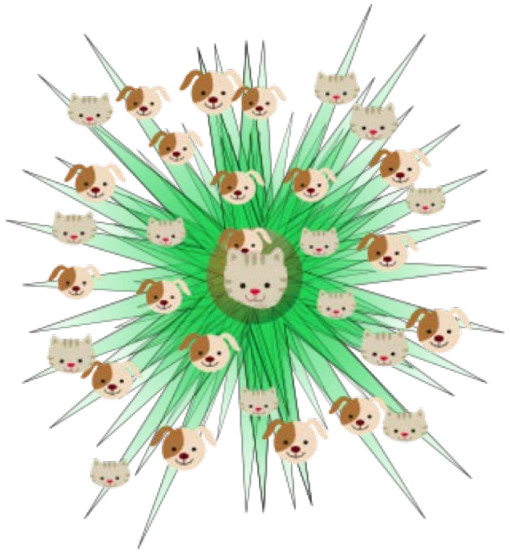
Cybenko 1989; Hornik 1991

Curse of dimensionality

$$\pi\left(\frac{1}{2}\right)^2 \approx 0.785$$



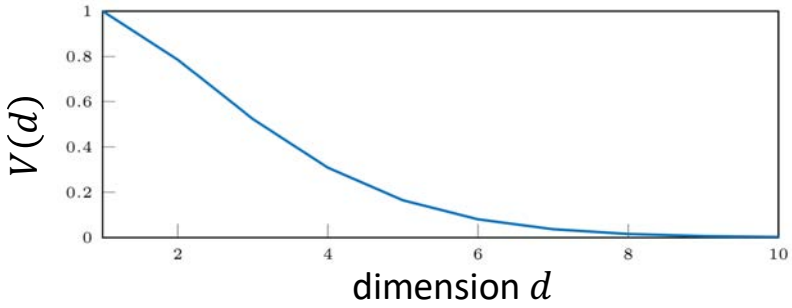
$$\frac{4}{3}\pi\left(\frac{1}{2}\right)^2 \approx 0.52$$



$$\approx 0.0159$$

Volume of ball inscribed in a unit hypercube

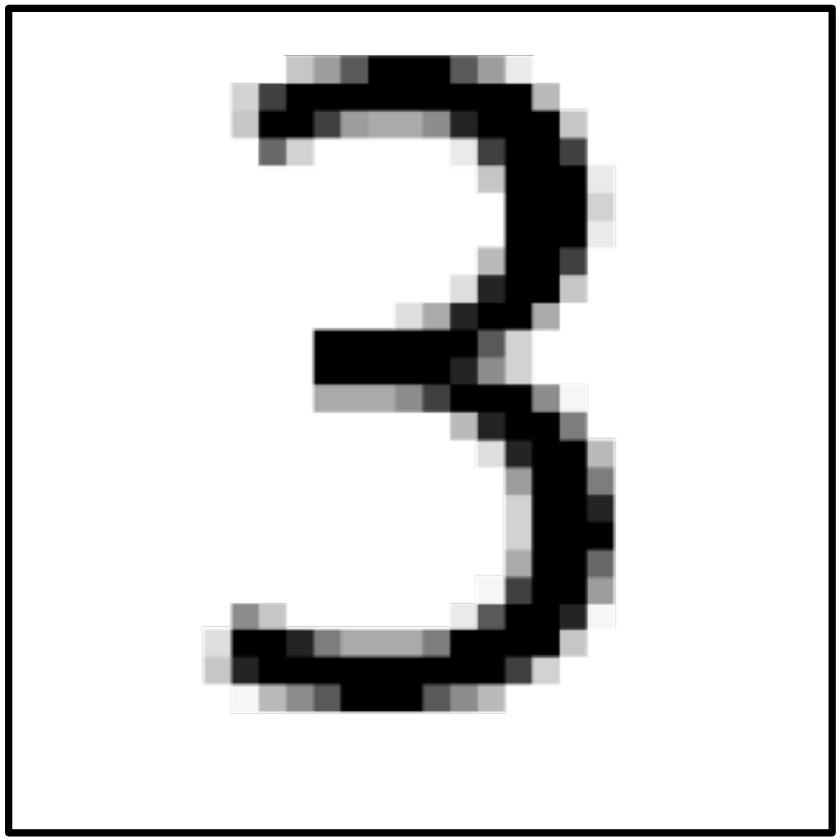
$$V(d) = \frac{\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}+1\right)2^d} \qquad \sim O\left(c^{-d}\right)$$



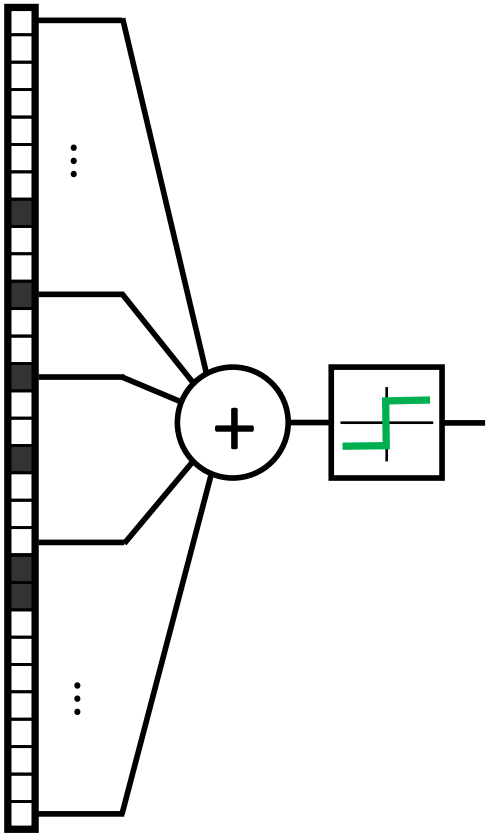
to approximate a continuous function $f: \mathbb{R}^d \rightarrow \mathbb{R}$ with ϵ accuracy one needs $O\left(\epsilon^{-d}\right)$ samples

Figure: Vision Dummy

Computer vision problems

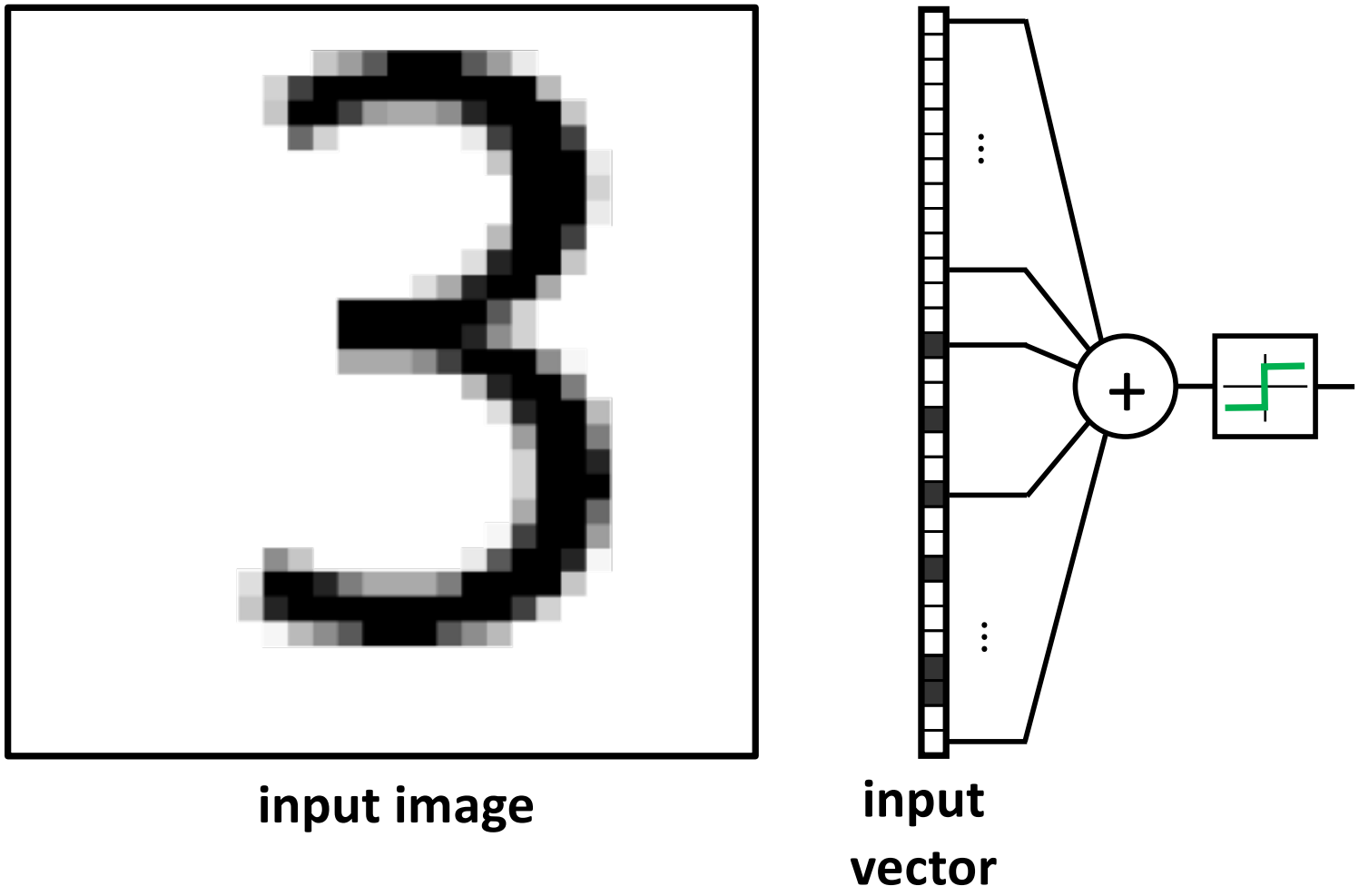


input image



input vector

Computer vision problems

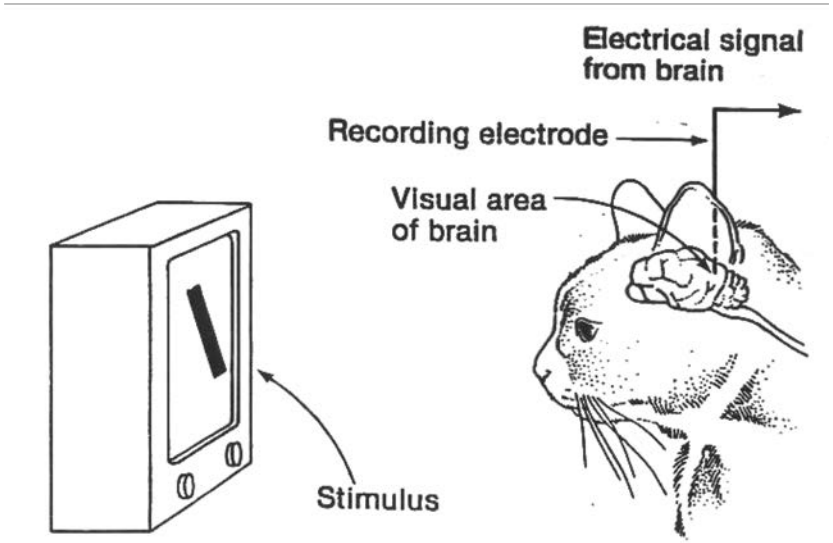


input image

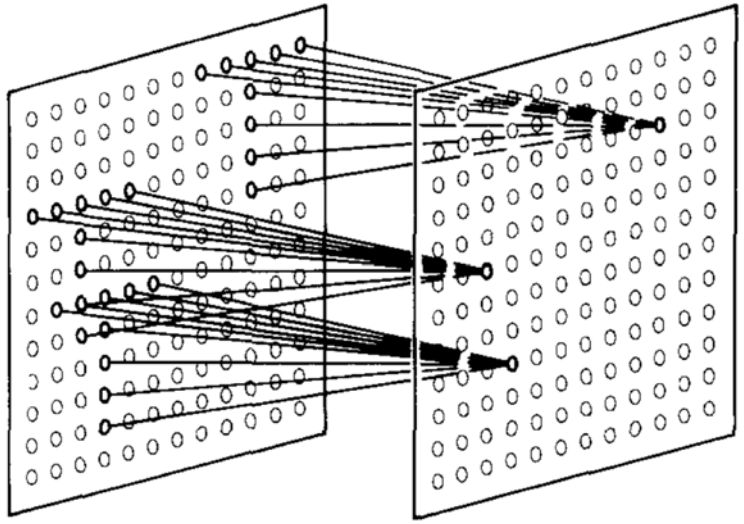
input vector

must learn shift invariance from data!

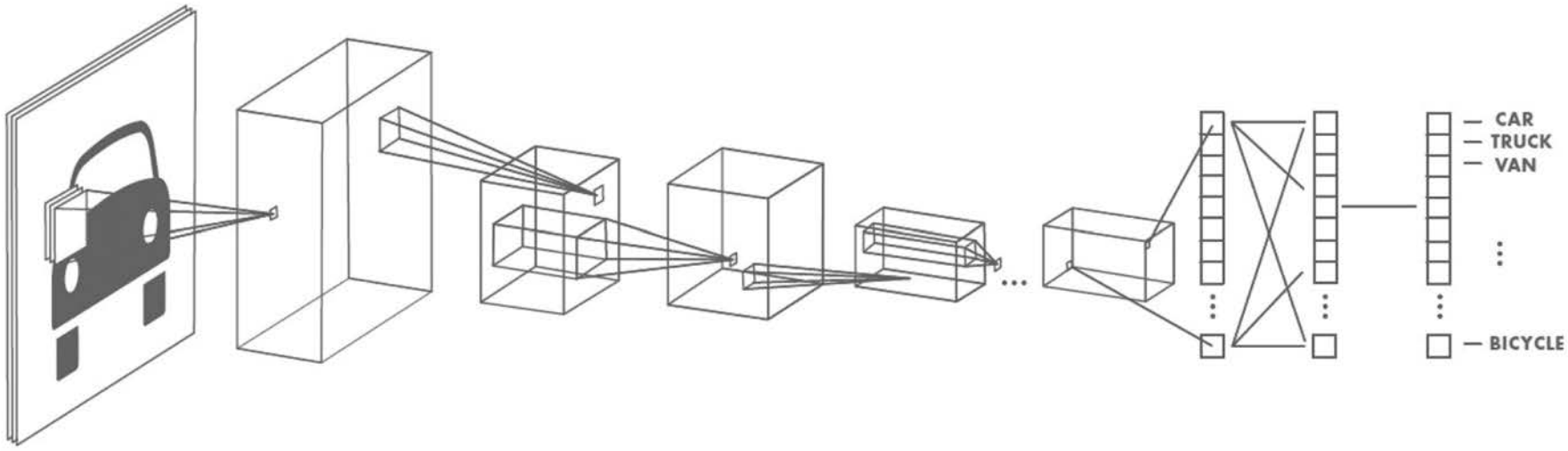
Take advantage of structure!



Hubel, Wiesel 1962;  1981

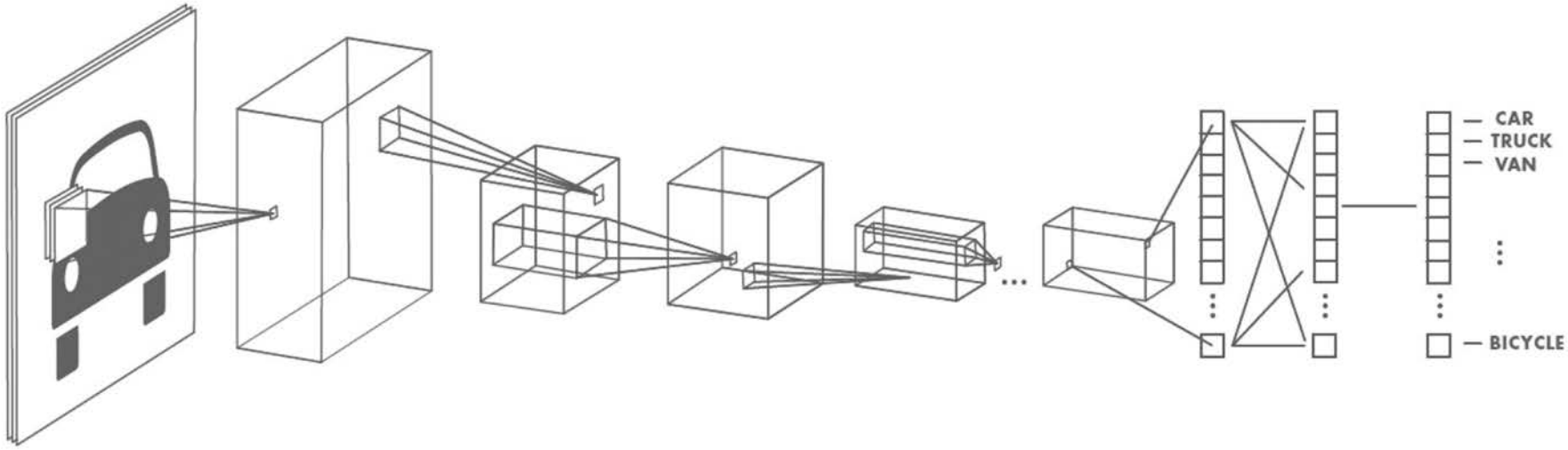


Fukushima 1980



LeCun et al. 1989

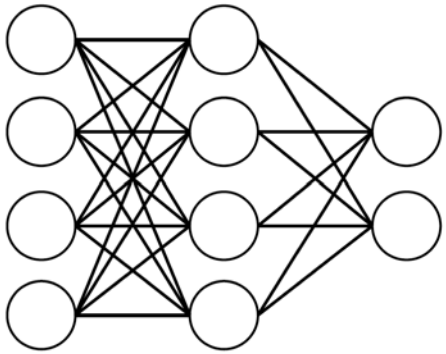
Convolutional Neural Networks



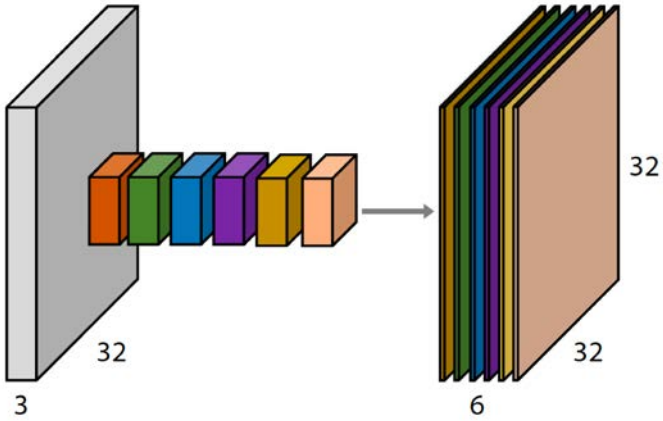
- Take advantage of self-similar structures at different scales
- **Local** operations with **shared** weights
- Shift-equivariant **convolutional filters** + **pooling** = shift invariance
- $O(1)$ **parameters per filter**
- $O(n)$ **complexity per layer**

LeCun et al. 1989

Inductive biases

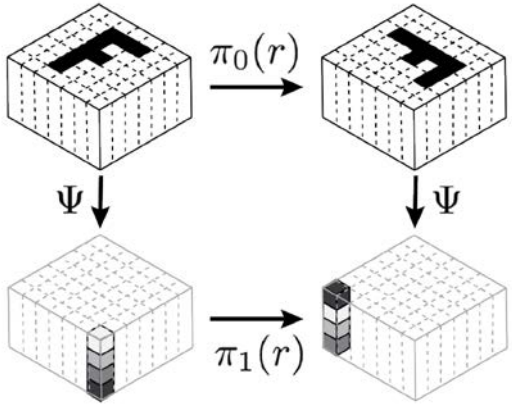


Multilayer perceptrons
"Universal approximators"



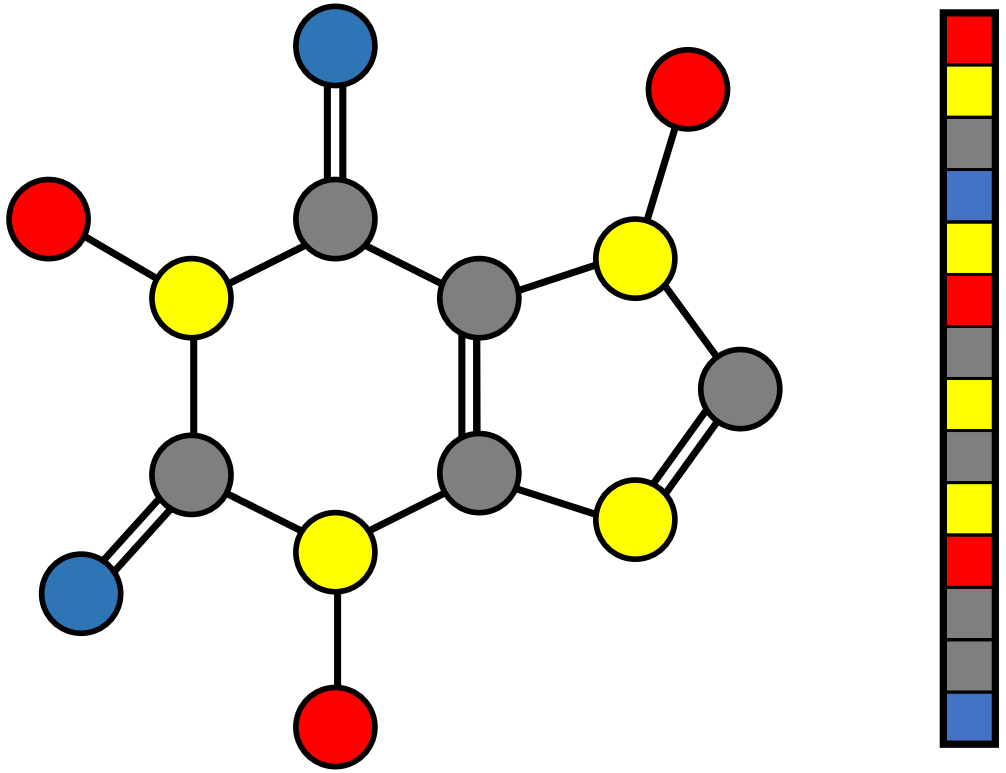
CNNs
Translation equivariance

LeCun et al. 1989

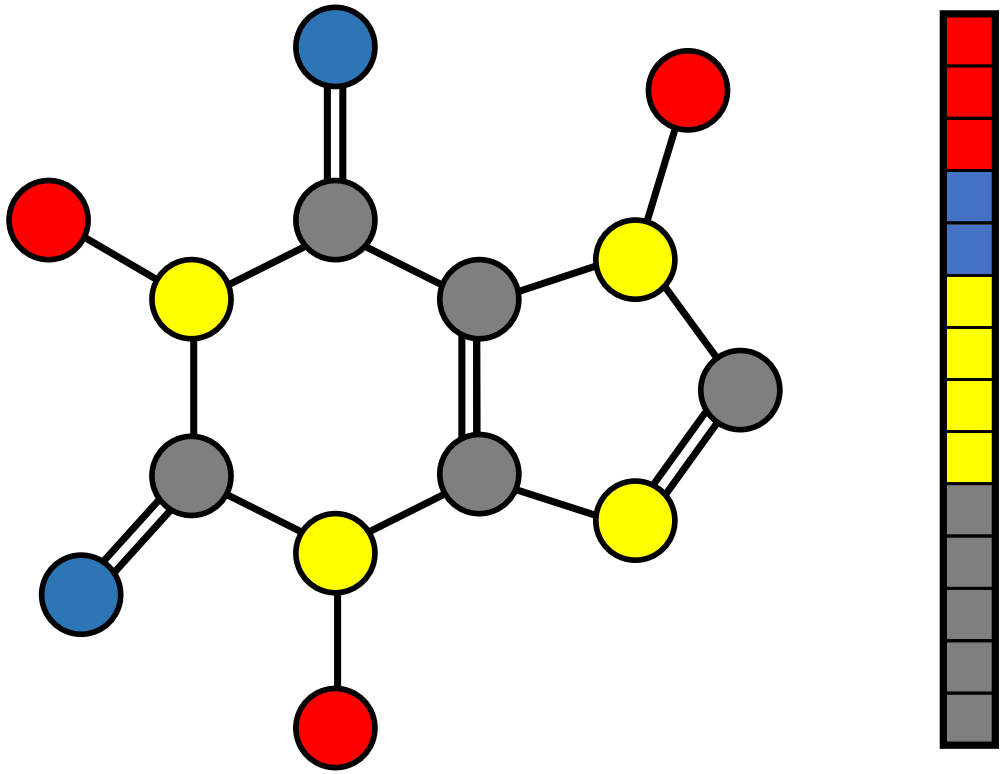


G-CNNs
Group equivariance

Cohen, Welling 2016

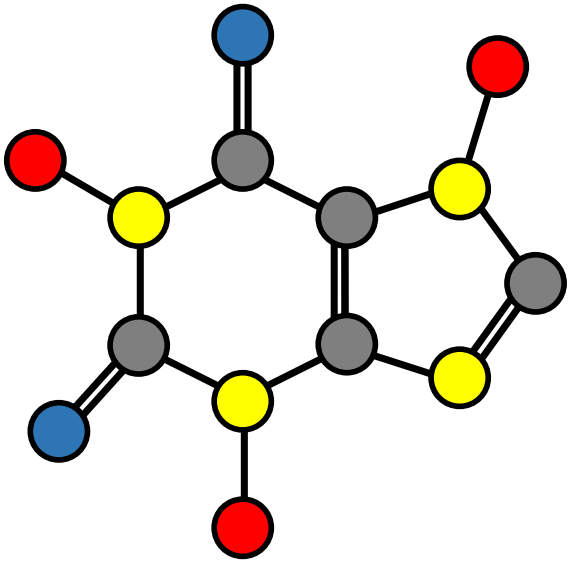


energy U = ?

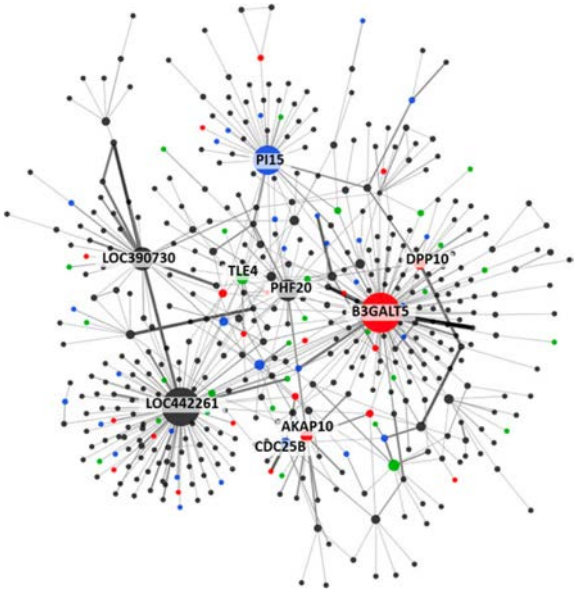


energy U = ?

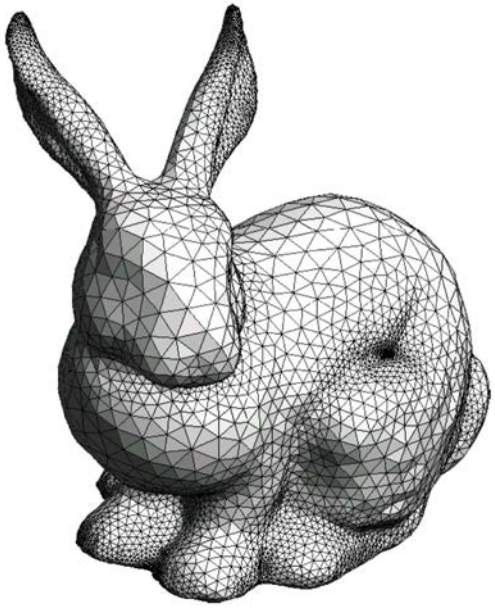
Social networks



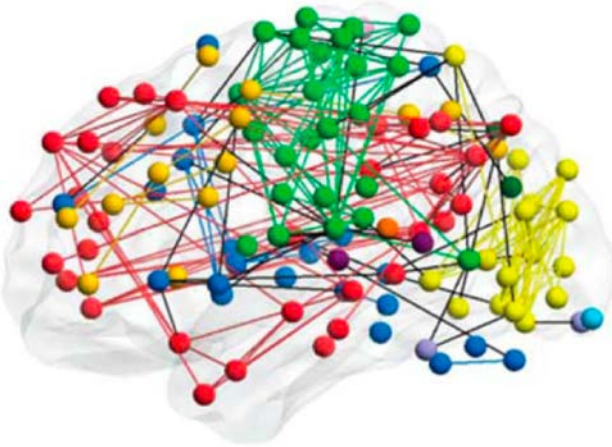
Molecules



Interaction networks



Meshes

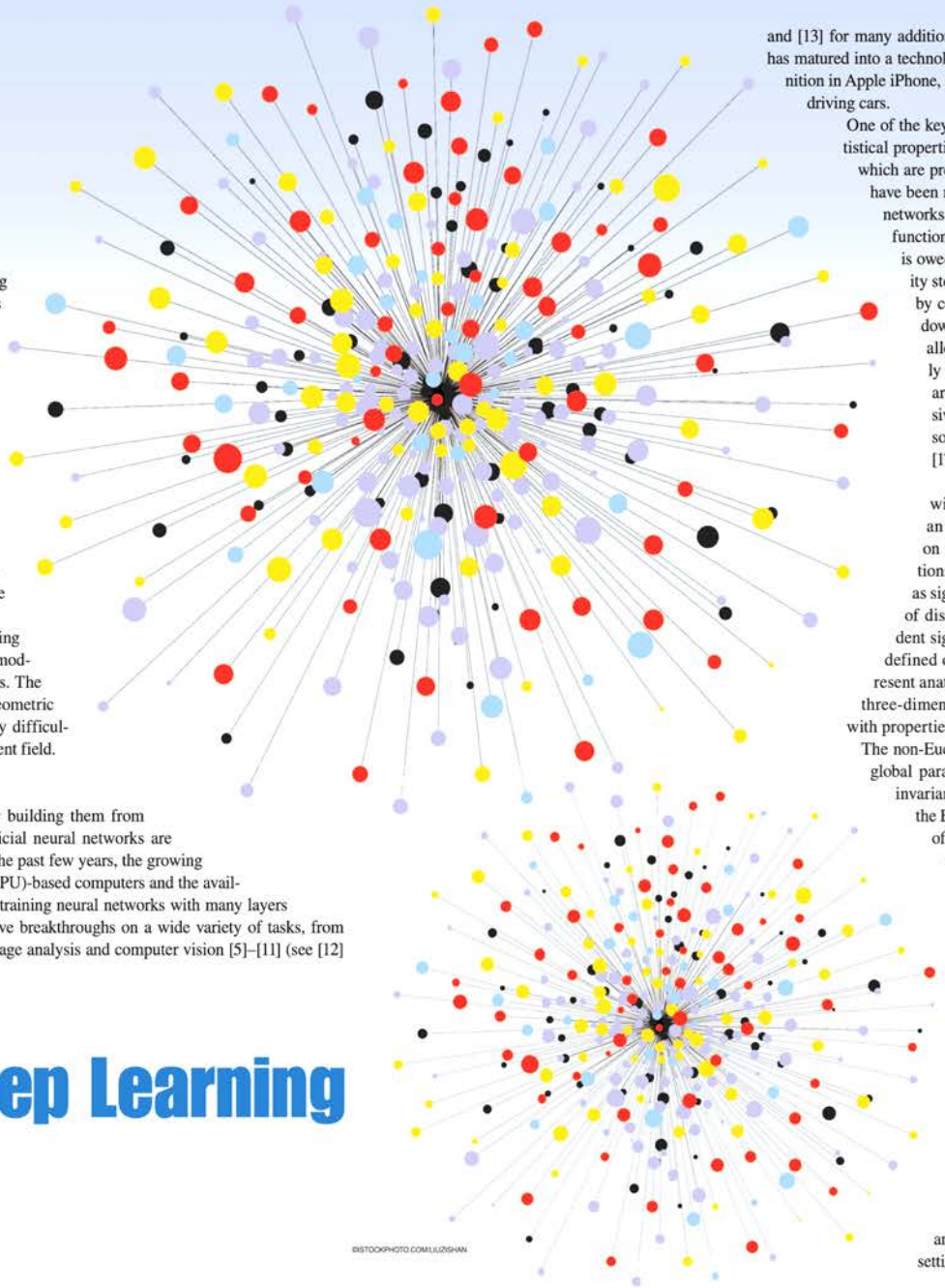


Functional networks

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst

Geometric Deep Learning

Going beyond Euclidean data



Many scientific fields study data with an underlying structure that is non-Euclidean. Some examples include social networks in computational social sciences, sensor networks in communications, functional networks in brain imaging, regulatory networks in genetics, and meshed surfaces in computer graphics. In many applications, such geometric data are large and complex (in the case of social networks, on the scale of billions) and are natural targets for machine-learning techniques. In particular, we would like to use deep neural networks, which have recently proven to be powerful tools for a broad range of problems from computer vision, natural-language processing, and audio analysis. However, these tools have been most successful on data with an underlying Euclidean or grid-like structure and in cases where the invariances of these structures are built into networks used to model them.

Geometric deep learning is an umbrella term for emerging techniques attempting to generalize (structured) deep neural models to non-Euclidean domains, such as graphs and manifolds. The purpose of this article is to overview different examples of geometric deep-learning problems and present available solutions, key difficulties, applications, and future research directions in this nascent field.

Overview of deep learning

Deep learning refers to learning complicated concepts by building them from simpler ones in a hierarchical or multilayer manner. Artificial neural networks are popular realizations of such deep multilayer hierarchies. In the past few years, the growing computational power of modern graphics processing unit (GPU)-based computers and the availability of large training data sets have allowed successfully training neural networks with many layers and degrees of freedom (DoF) [1]. This has led to qualitative breakthroughs on a wide variety of tasks, from speech recognition [2], [3] and machine translation [4] to image analysis and computer vision [5]–[11] (see [12] and [13] for many additional examples of successful applications of deep learning). Today, deep learning has matured into a technology that is widely used in commercial applications, including Siri speech recognition in Apple iPhone, Google text translation, and Mobileye vision-based technology for autonomously driving cars.

One of the key reasons for the success of deep neural networks is their ability to leverage statistical properties of the data, such as stationarity and compositionality through local statistics, which are present in natural images, video, and speech [14], [15]. These statistical properties have been related to physics [16] and formalized in specific classes of convolutional neural networks (CNNs) [17]–[19]. In image analysis applications, one can consider images as functions on the Euclidean space (plane), sampled on a grid. In this setting, stationarity is owed to shift invariance, locality is due to the local connectivity, and compositionality stems from the multiresolution structure of the grid. These properties are exploited by convolutional architectures [20], which are built of alternating convolutional and downsampling (pooling) layers. The use of convolutions has a twofold effect. First, it allows extracting local features that are shared across the image domain and greatly reduces the number of parameters in the network with respect to generic deep architectures (and thus also the risk of overfitting), without sacrificing the expressive capacity of the network. Second, the convolutional architecture itself imposes some priors about the data, which appear very suitable especially for natural images [17]–[19], [21].

While deep-learning models have been particularly successful when dealing with speech, image, and video signals, in which there are an underlying Euclidean structure, recently there has been a growing interest in trying to apply learning on non-Euclidean geometric data. Such kinds of data arise in numerous applications. For instance, in social networks, the characteristics of users can be modeled as signals on the vertices of the social graph [22]. Sensor networks are graph models of distributed interconnected sensors, whose readings are modeled as time-dependent signals on the vertices. In genetics, gene expression data are modeled as signals defined on the regulatory network [23]. In neuroscience, graph models are used to represent anatomical and functional structures of the brain. In computer graphics and vision, three-dimensional (3-D) objects are modeled as Riemannian manifolds (surfaces) endowed with properties such as color texture.

The non-Euclidean nature of such data implies that there are no such familiar properties as global parameterization, common system of coordinates, vector space structure, or shift invariance. Consequently, basic operations like convolution that are taken for granted in the Euclidean case are even not well defined on non-Euclidean domains. The purpose of this article is to show different methods of translating the key ingredients of successful deep-learning methods, such as CNNs, to non-Euclidean data.

Geometric learning problems

Broadly speaking, we can distinguish between two classes of geometric learning problems. In the first class of problems, the goal is to characterize the structure of the data. The second class of problems deals with analyzing functions defined on a given non-Euclidean domain. These two classes are related, because understanding the properties of functions defined on a domain conveys certain information about the domain, and vice versa, the structure of the domain imposes certain properties on the functions on it.

Structure of the domain

As an example of the first class of problems, assume to be given a set of data points with some underlying low-dimensional structure embedded into a high-dimensional Euclidean space. Recovering that low-dimensional structure is often referred to as *manifold learning* or *nonlinear dimensionality reduction* and is an instance of unsupervised learning (note that the notion of manifold in this setting can be considerably more general than a classical smooth manifold; see, e.g.,

Digital Object Identifier 10.1109/MSP.2017.2693418
Date of publication: 11 July 2017

1053-5888/17©2017IEEE

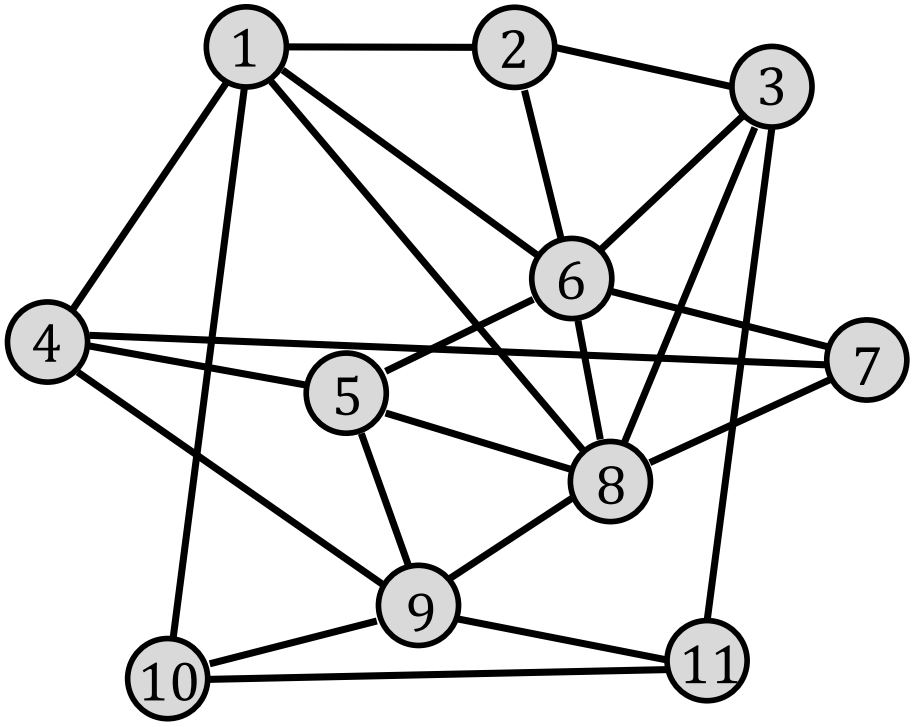
Geometric Deep Learning

Relational inductive biases **Graph neural networks**

Graph representation learning

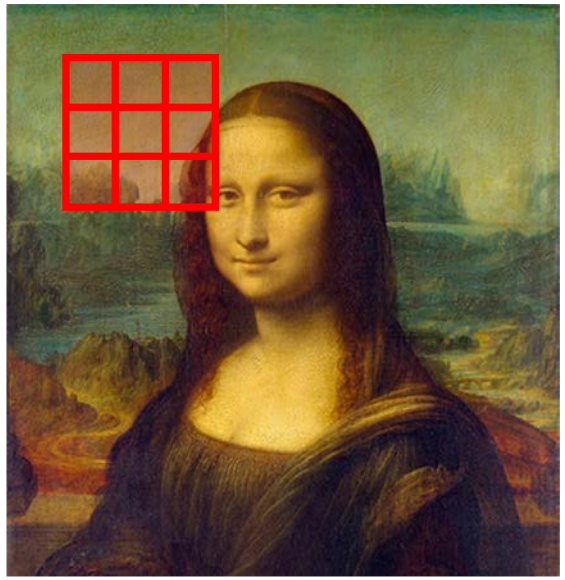
"**Differential geometry** and **graph theory** […] are insufficiently known in the signal processing community. One of our goals is to provide an accessible overview of these models"

Manifolds



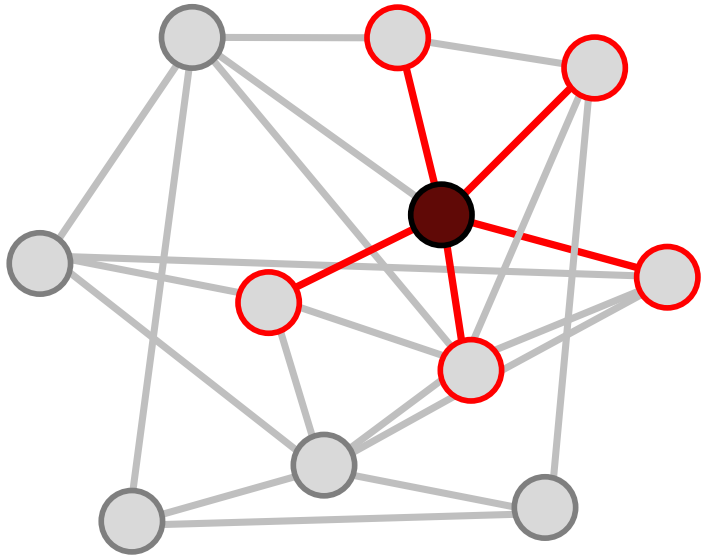
Graphs

Images



- Constant number of neighbors
- Fixed ordering of neighbors
- Shift invariance

Graphs



- Different number of neighbors
- No ordering of neighbors
- Permutation invariance

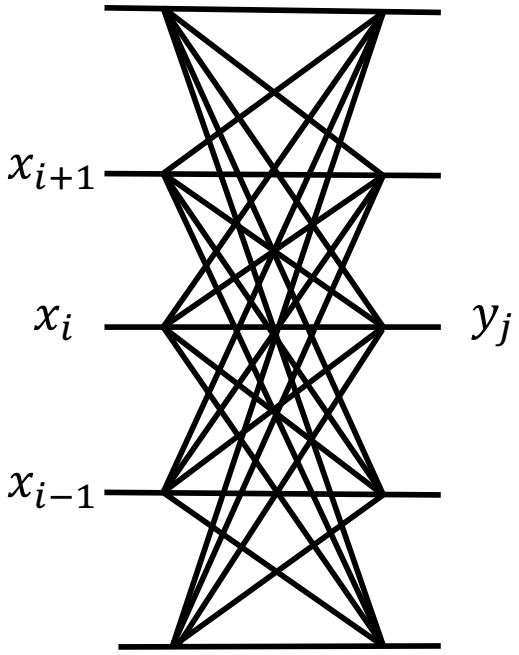
Convolution?

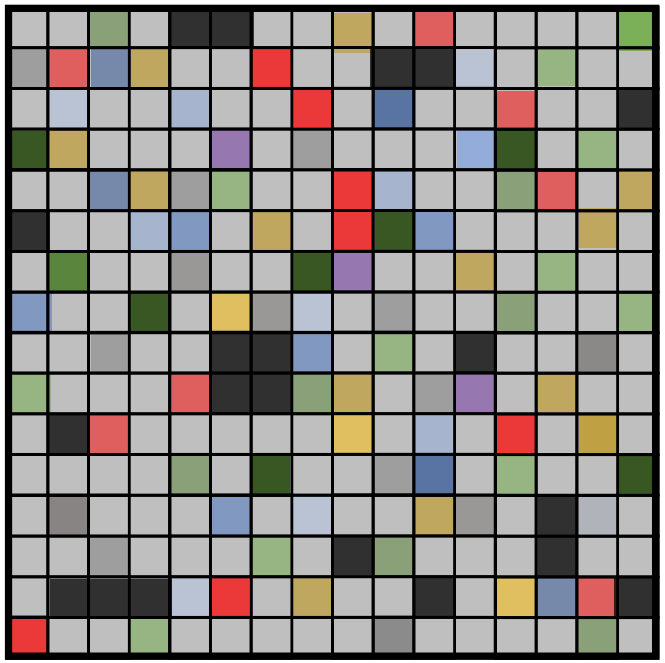
Pooling?

Efficient computation?

Convolution

Fully connected layer

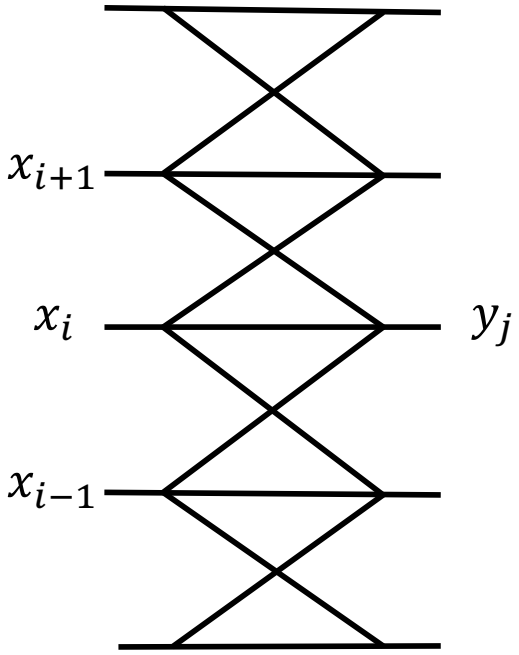


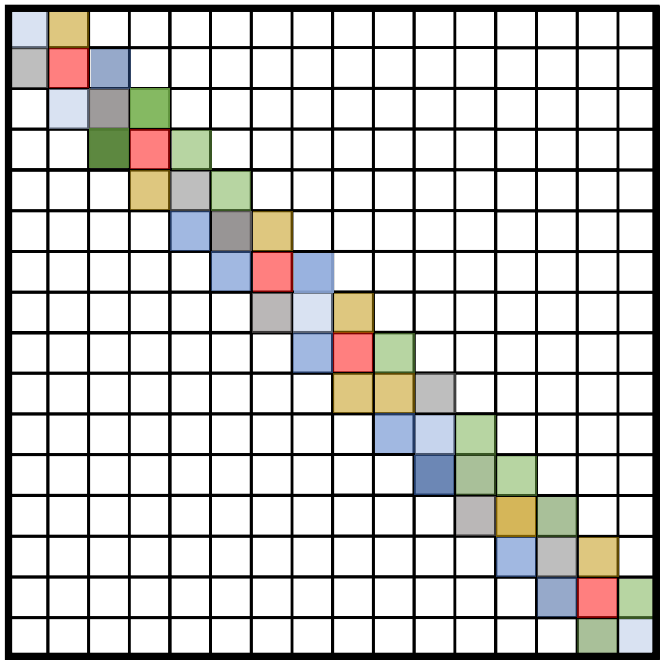


$$y_j = w_{j,1}x_1 + \cdots + w_{j,n}x_n$$

dense weights: n^2 parameters

Sparsely connected layer

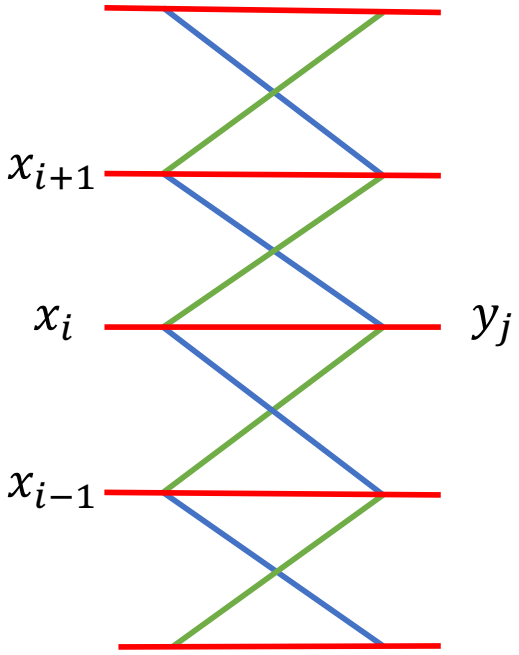


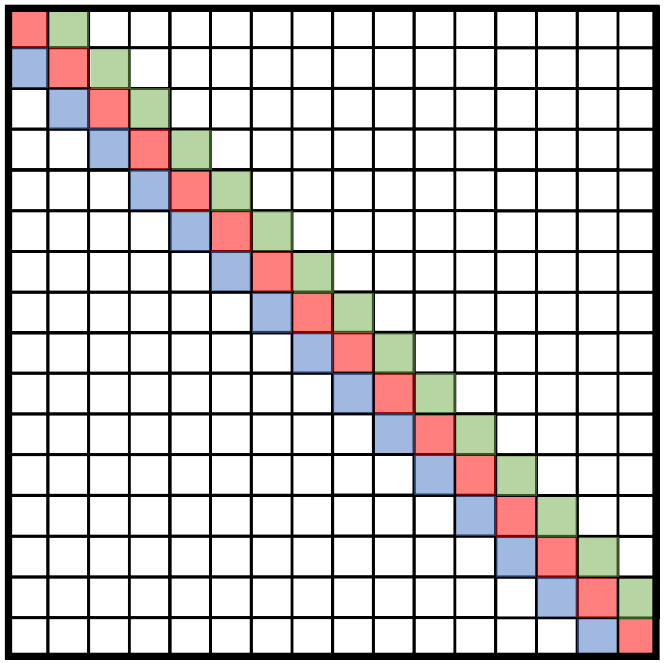


$$y_i = w_{j,i-1}x_{i-1} + w_{j,i}x_i + w_{j,i+1}x_{i+1}$$

position-dependent weights: 3n parameters

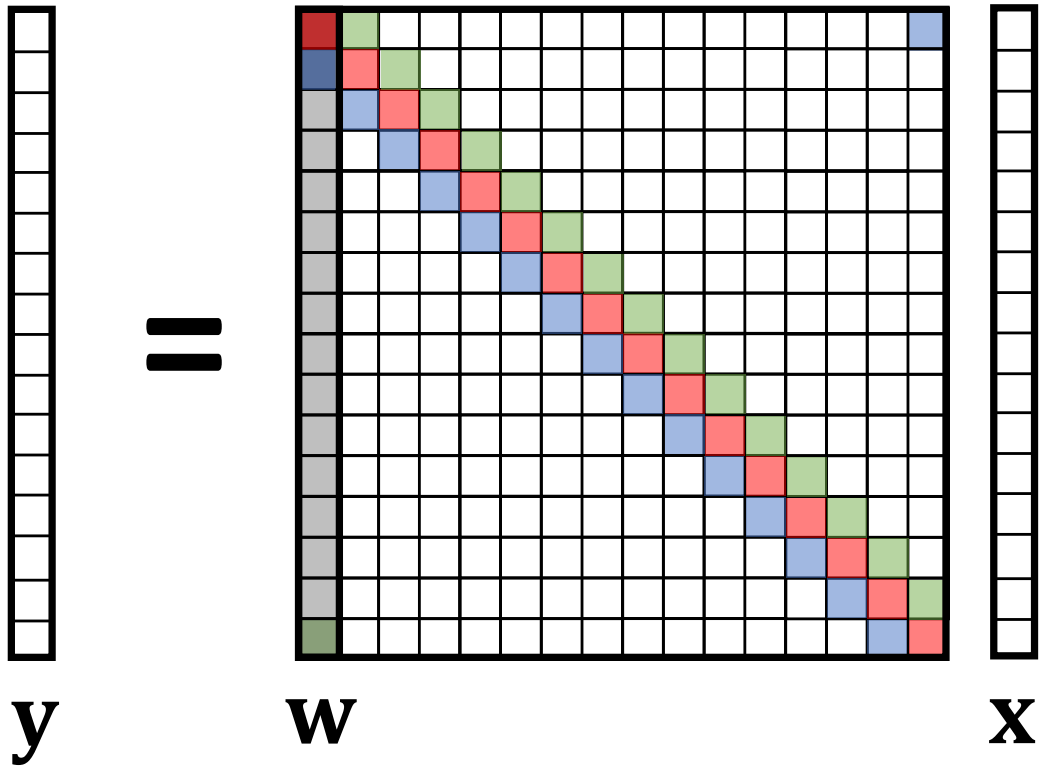
Convolutional layer

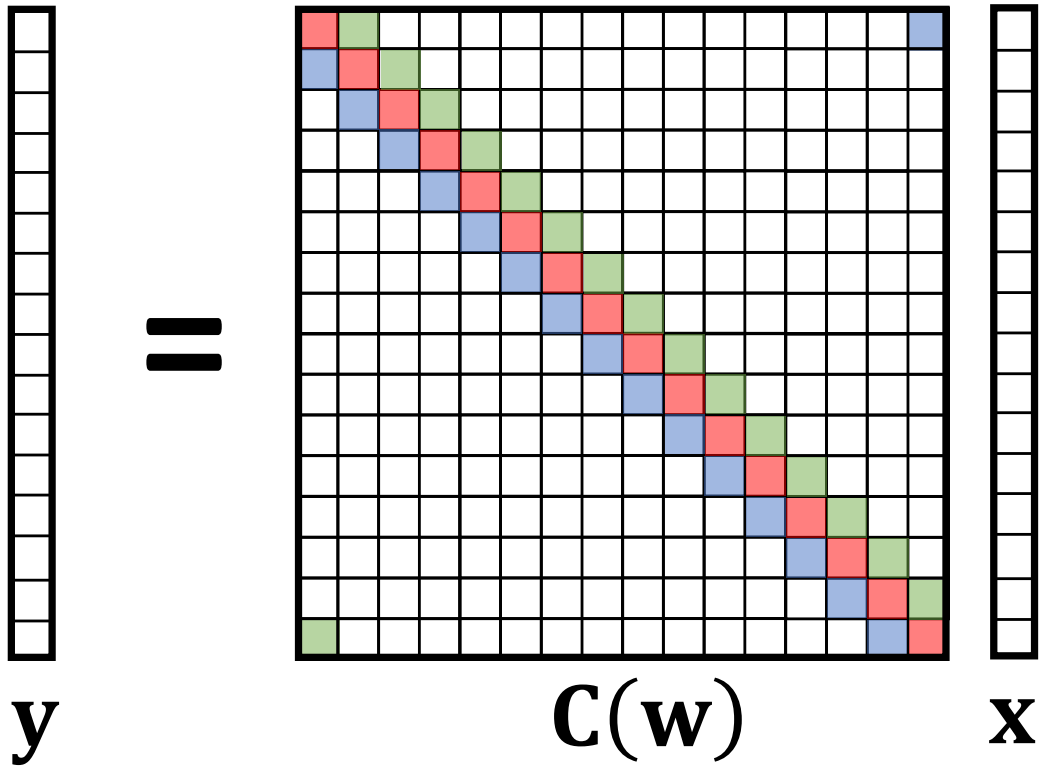




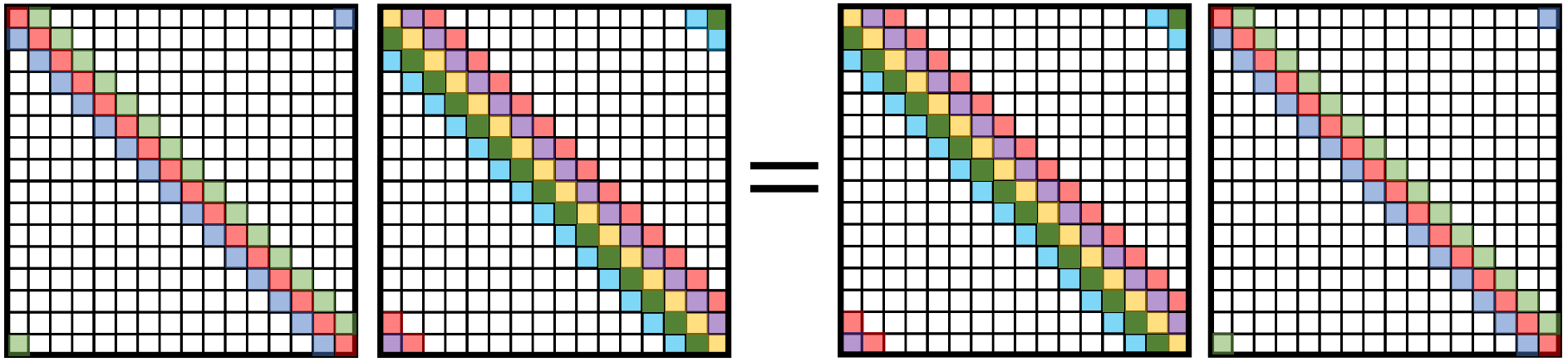
$$y_j = w_{-1}x_{i-1} + w_0 x_i + w_{+1}x_{i+1}$$

shared weights: 3 parameters


y
=
w
x

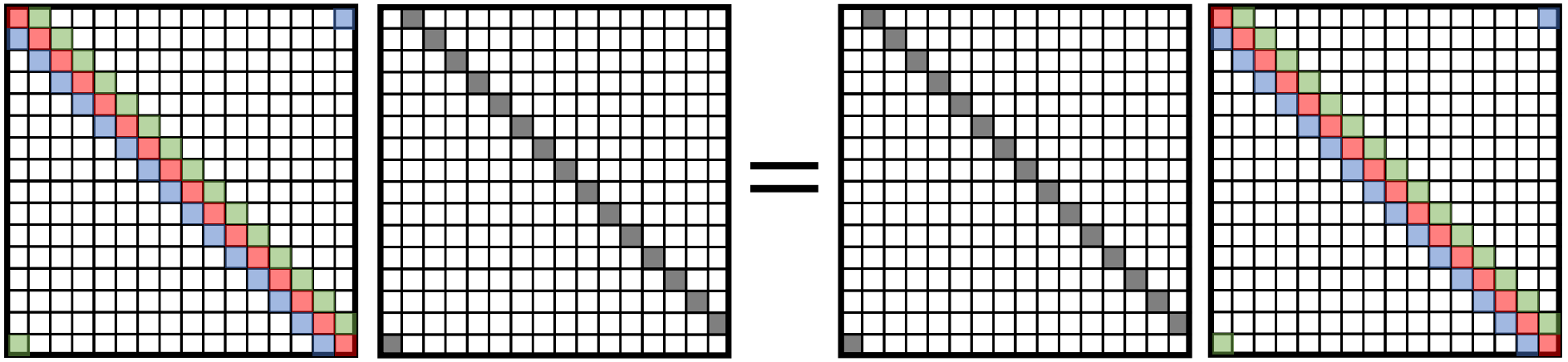


Convolution = circulant matrix



circulant matrices commute

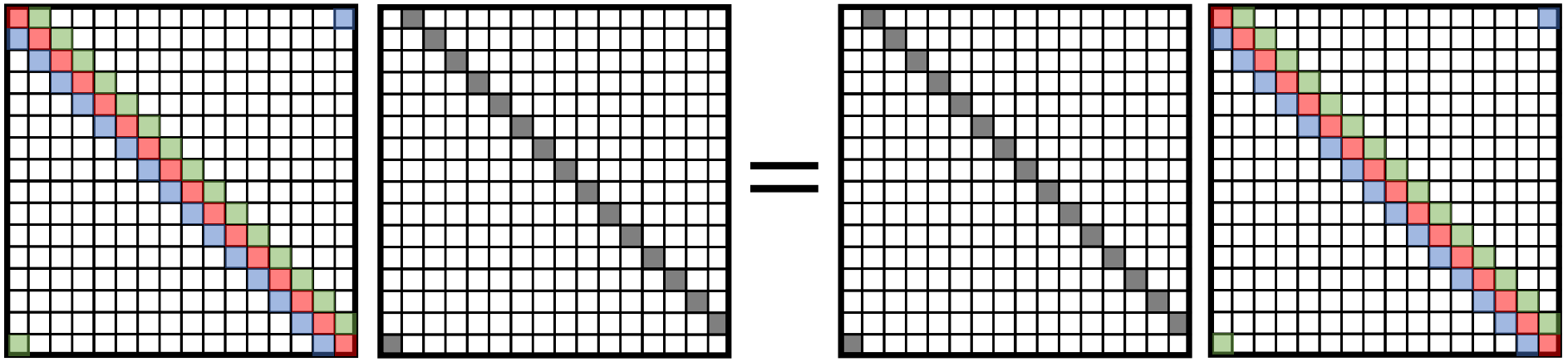
Shift-equivariance



circulant matrices commute

convolution commutes with shift

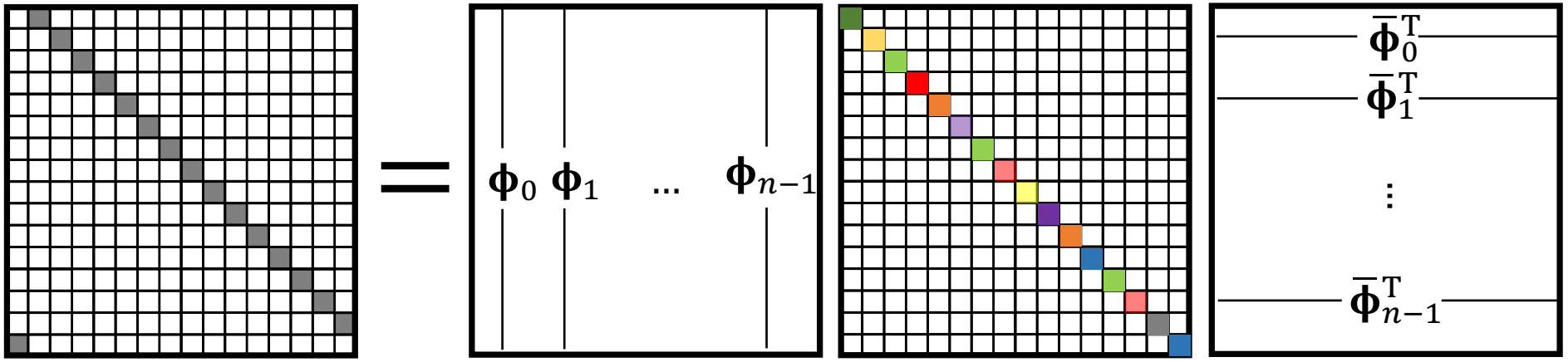
Joint diagonalization



commuting matrices are jointly diagonalizable

convolution is diagonalized by eigenvectors of S^{T}

Eigenvectors of $\mathbf{S}^{\mathrm{T}}$ = Fourier transform

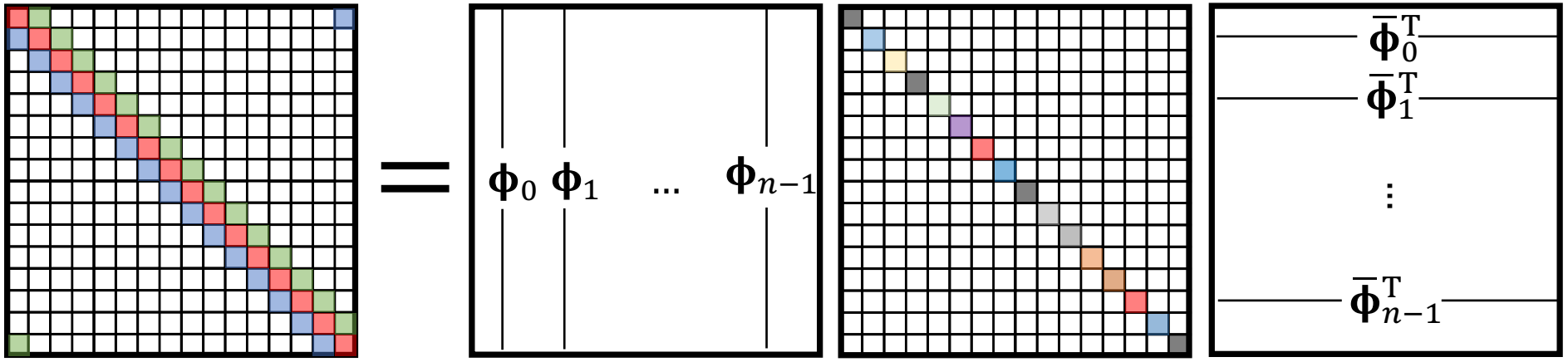


$\boldsymbol{\Phi}$ inverse DFT $\quad \boldsymbol{\Lambda} \quad \boldsymbol{\Phi}^*$ forward DFT

$$\boldsymbol{\Lambda} = \begin{bmatrix} e^{\mathrm{i}\frac{2\pi}{n}0} & & \\ & \ddots & \\ & & e^{\mathrm{i}\frac{2\pi}{n}(n-1)} \end{bmatrix} \qquad \boldsymbol{\phi}_k = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 \\ e^{\mathrm{i}\frac{2\pi}{n}k} \\ \vdots \\ e^{\mathrm{i}\frac{2\pi}{n}(n-1)k} \end{bmatrix}$$

Shift operator is diagonalized by the Fourier transform

Convolution in spectral domain

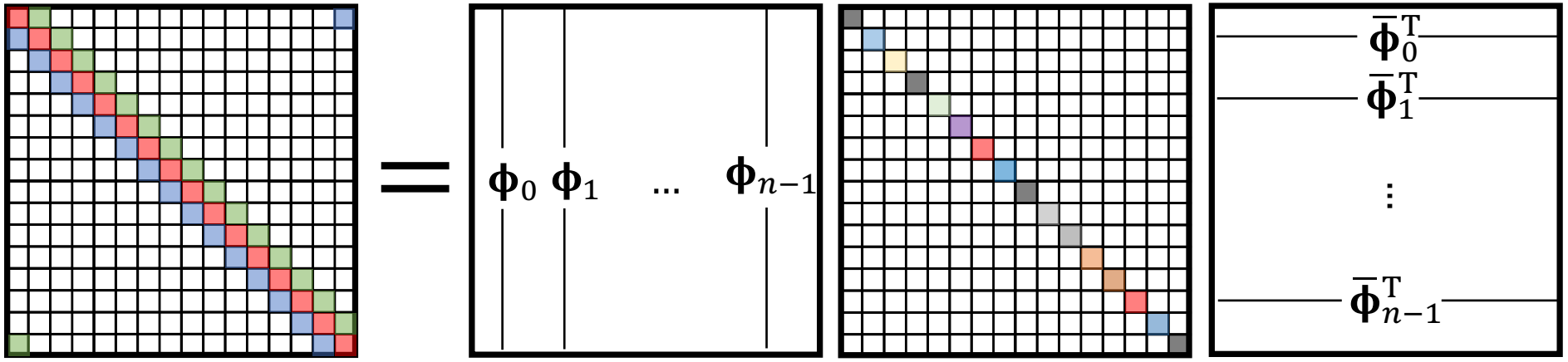


$\mathbf{C}(\mathbf{w})$ $\boldsymbol{\Phi}$ $\boldsymbol{\Lambda}_{\mathbf{C}(\mathbf{w})}$ $\boldsymbol{\Phi}^*$

inverse DFT

forward DFT

Convolution in spectral domain

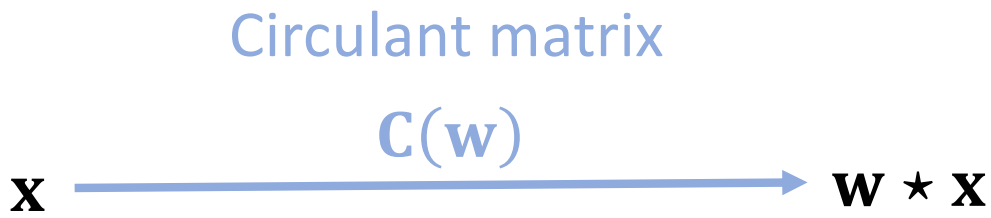


$\mathbf{C}(\mathbf{w})$ $\quad$ $\boldsymbol{\Phi}$ inverse DFT $\quad$ $\boldsymbol{\Lambda}_{\mathbf{C}(\mathbf{w})} = \mathrm{diag}(\hat{w}_1 \ldots \hat{w}_n)$ $\quad$ $\boldsymbol{\Phi}^*$ forward DFT

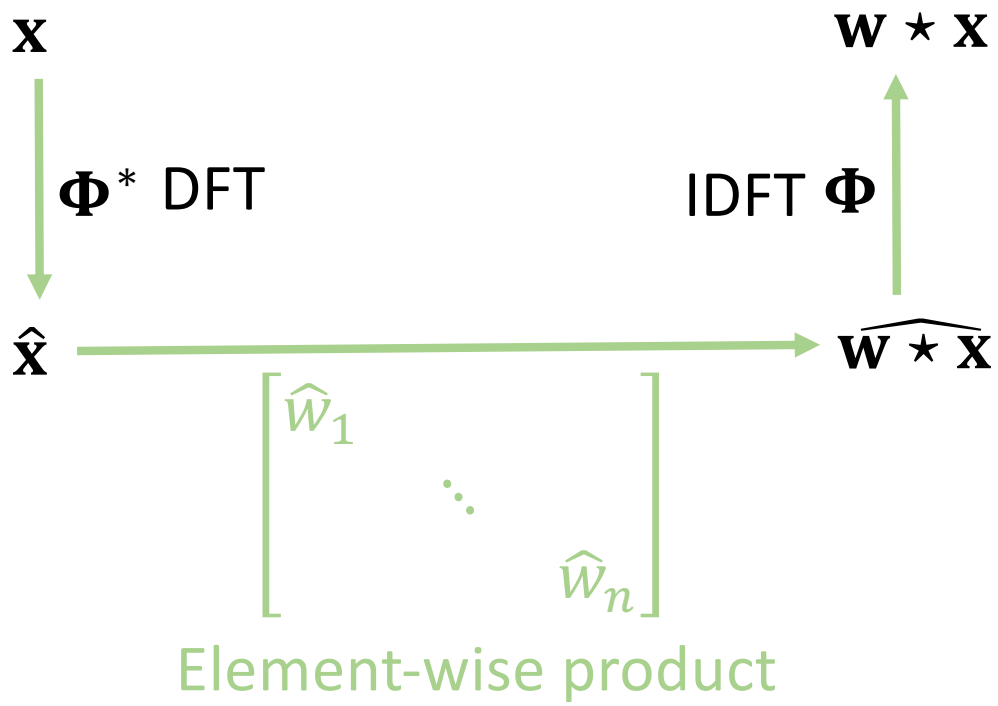
Eigenvalues of $\mathbf{C}(\mathbf{w})$ are the Fourier transform

$$\hat{\mathbf{w}} = \boldsymbol{\Phi}^* \mathbf{w}$$

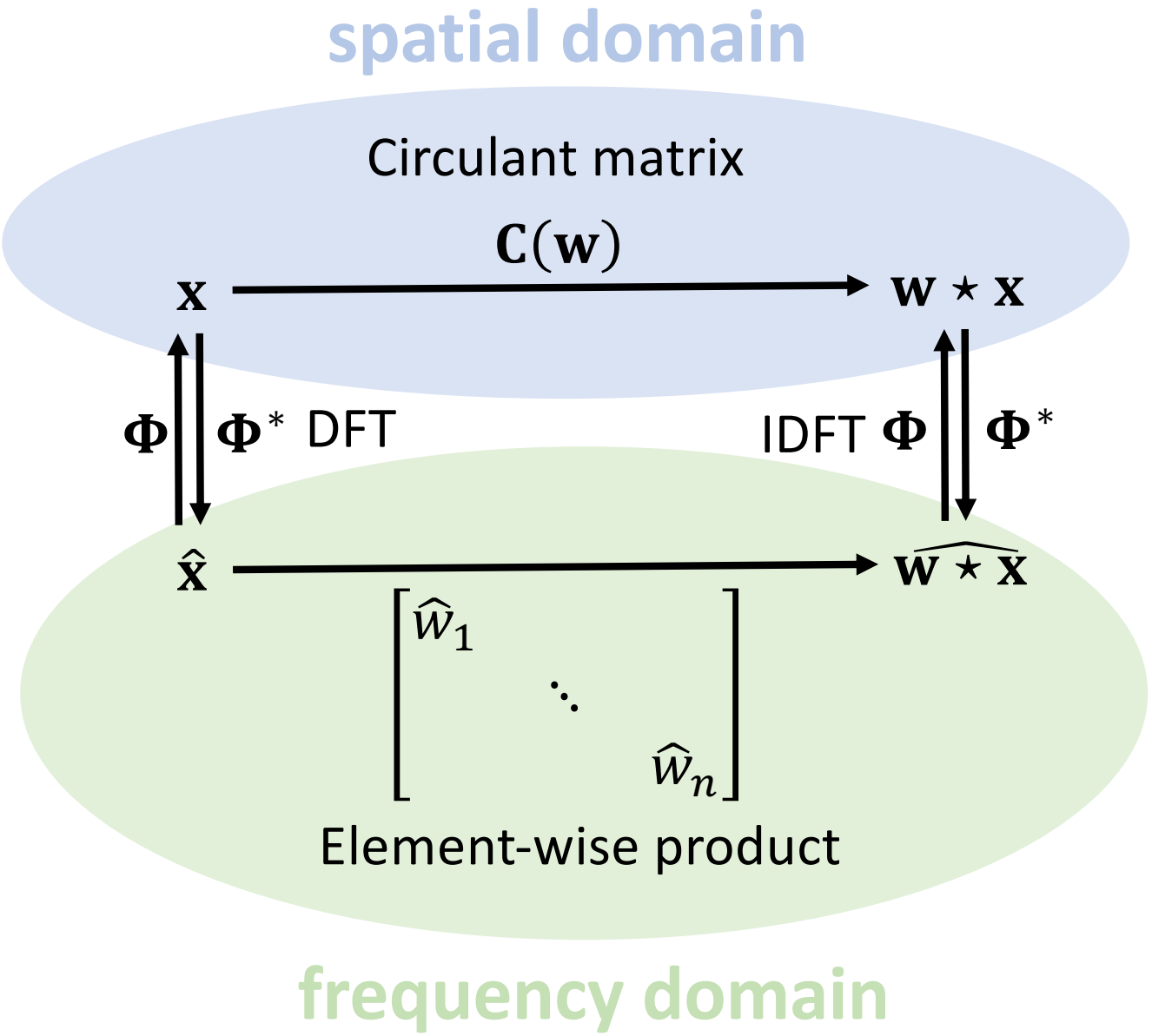
Convolution theorem



Convolution theorem



Convolution theorem

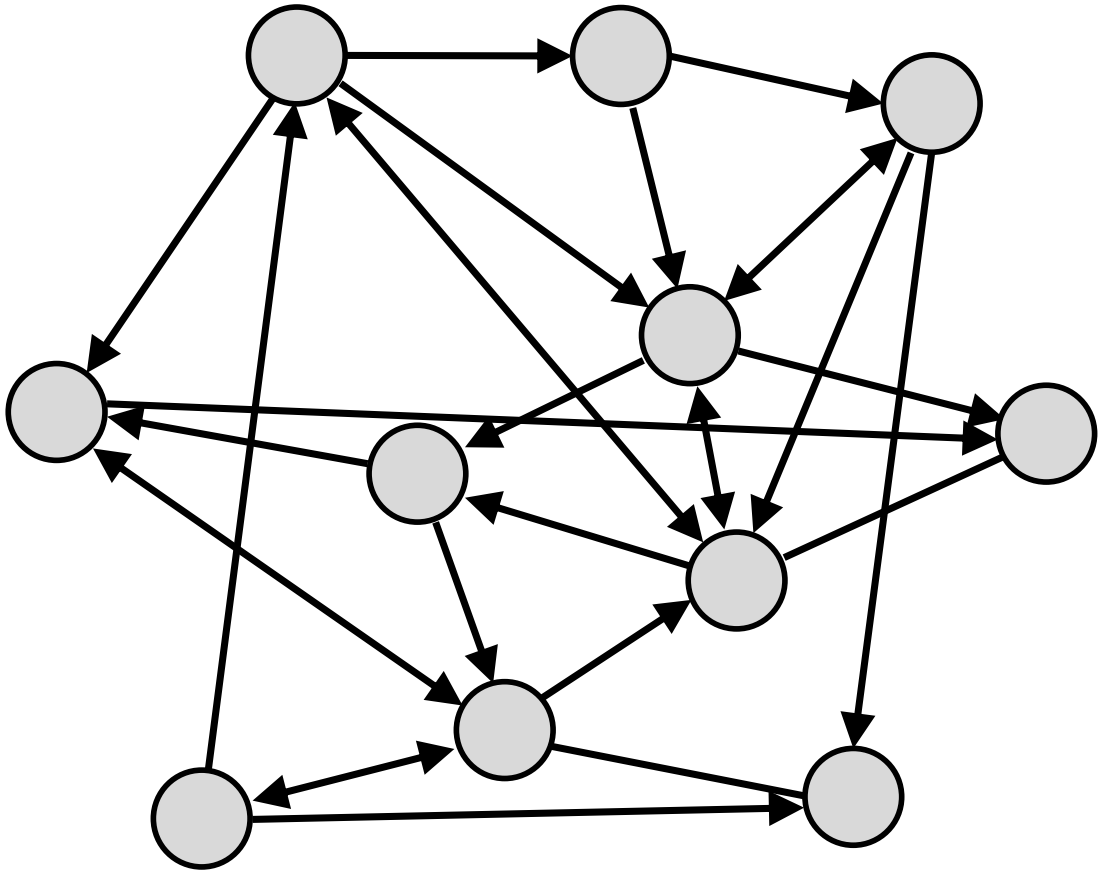


Key insights

- **Convolution in spatial domain:** circulant matrix
- **Local aggregation on adjacent nodes** with shared parameters
- Special structure due to underlying grid
- All circulant matrices diagonalized by DFT (eigenvectors of shift)
- **Convolution in frequency domain:** apply DFT, apply element-wise product, apply inverse DFT
- Efficient computation: $O(n)$ with small filter
  $O(n \log n)$ using FFT

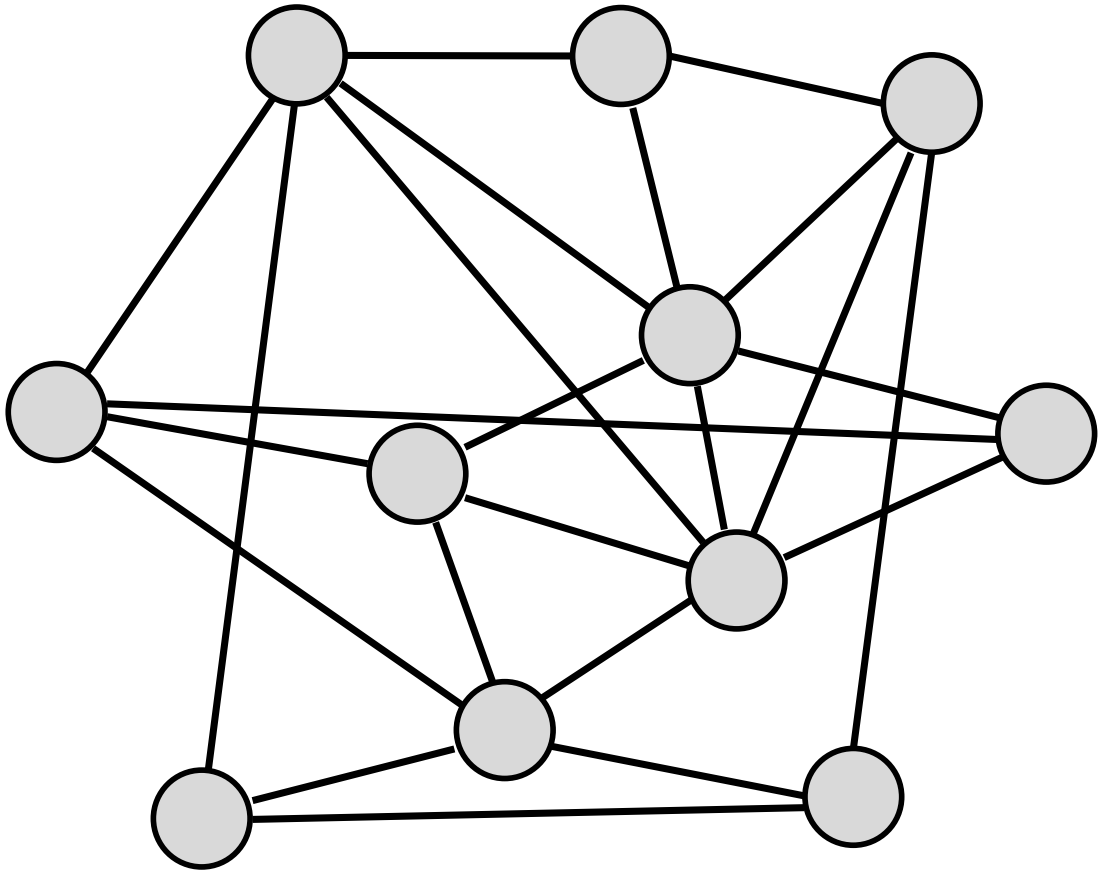
From grids to graphs

Graphs



- **Vertices** or **nodes** $\mathcal{V} = \{1, \ldots, n\}$
- **Edges** $\mathcal{E} = \{(i, j) : i, j \in \mathcal{V}\} \subseteq \mathcal{V} \times \mathcal{V}$ (directed)

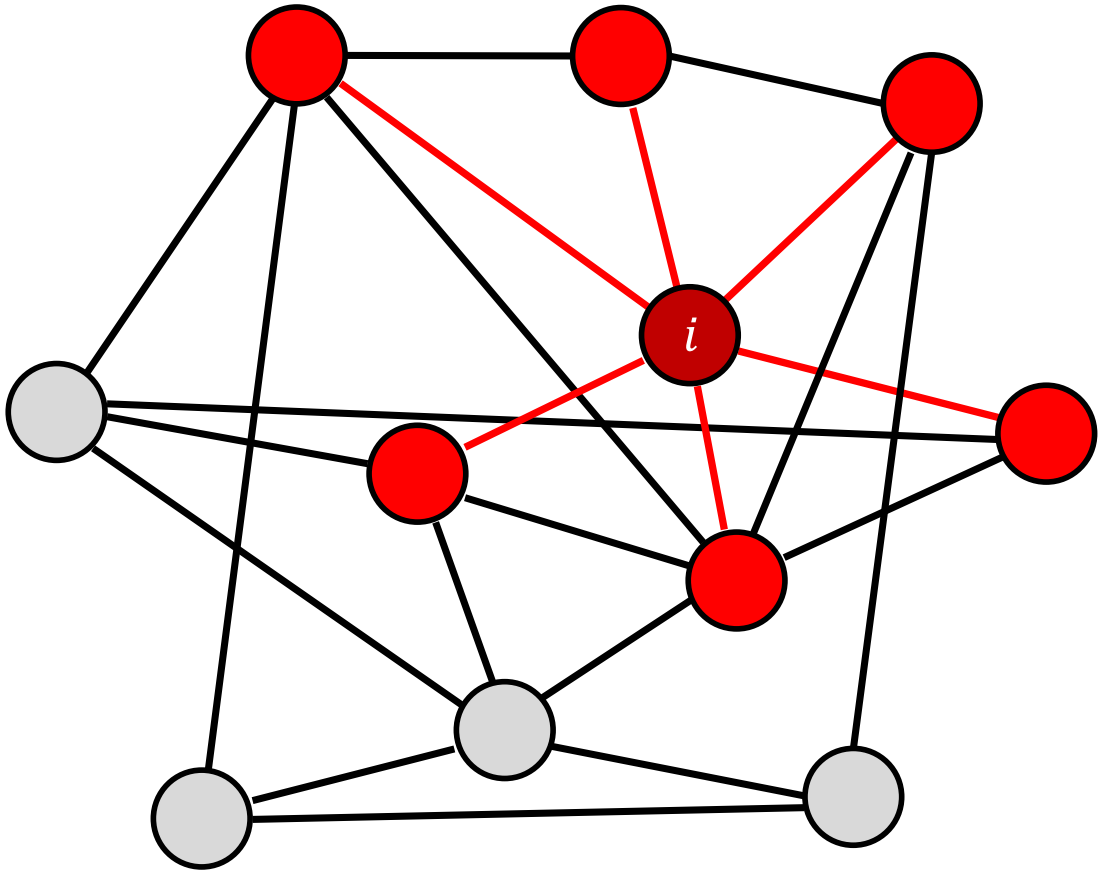
Graphs



- **Vertices** or **nodes** $\mathcal{V} = \{1, \ldots, n\}$
- **Edges** $\mathcal{E} = \{(i,j) : i,j \in \mathcal{V}\} \subseteq \mathcal{V} \times \mathcal{V}$ (directed)

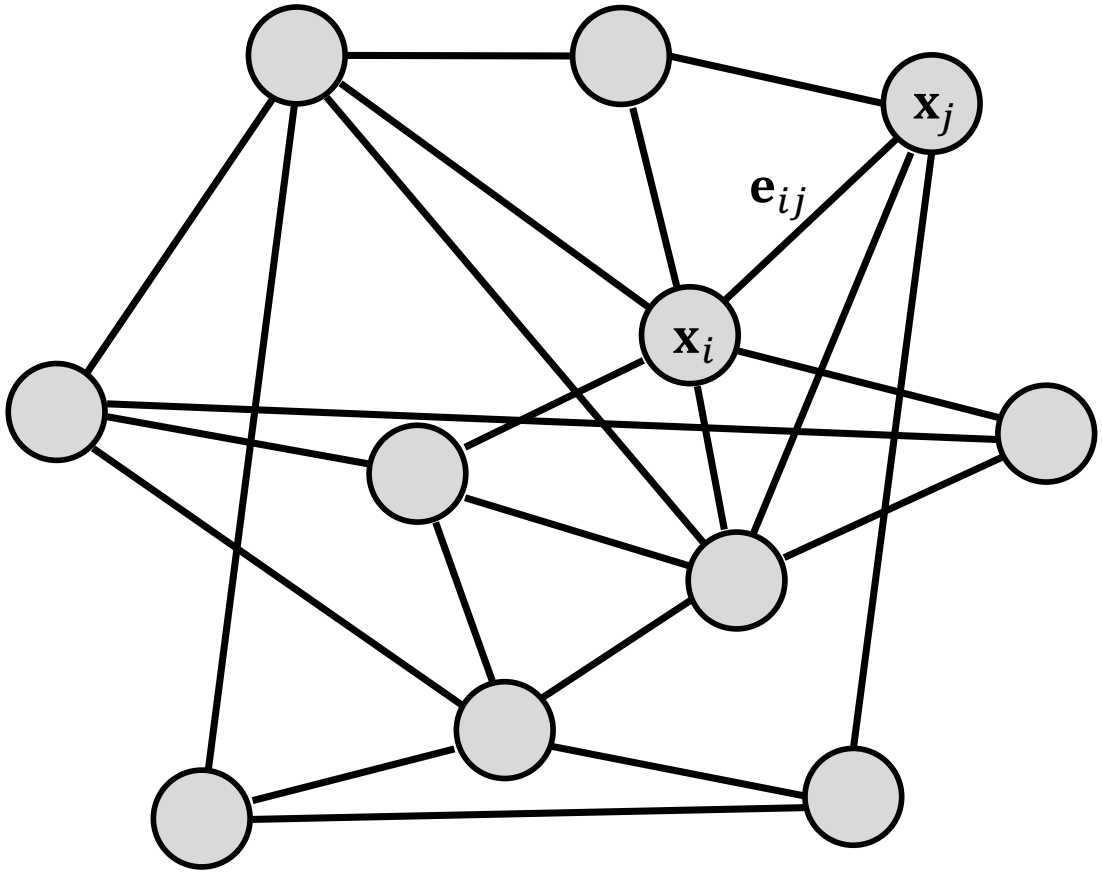
  $\mathcal{E} = \{\{i,j\} : i,j \in \mathcal{V}\} \subseteq \mathcal{V} \times \mathcal{V}$ (undirected)

Graphs



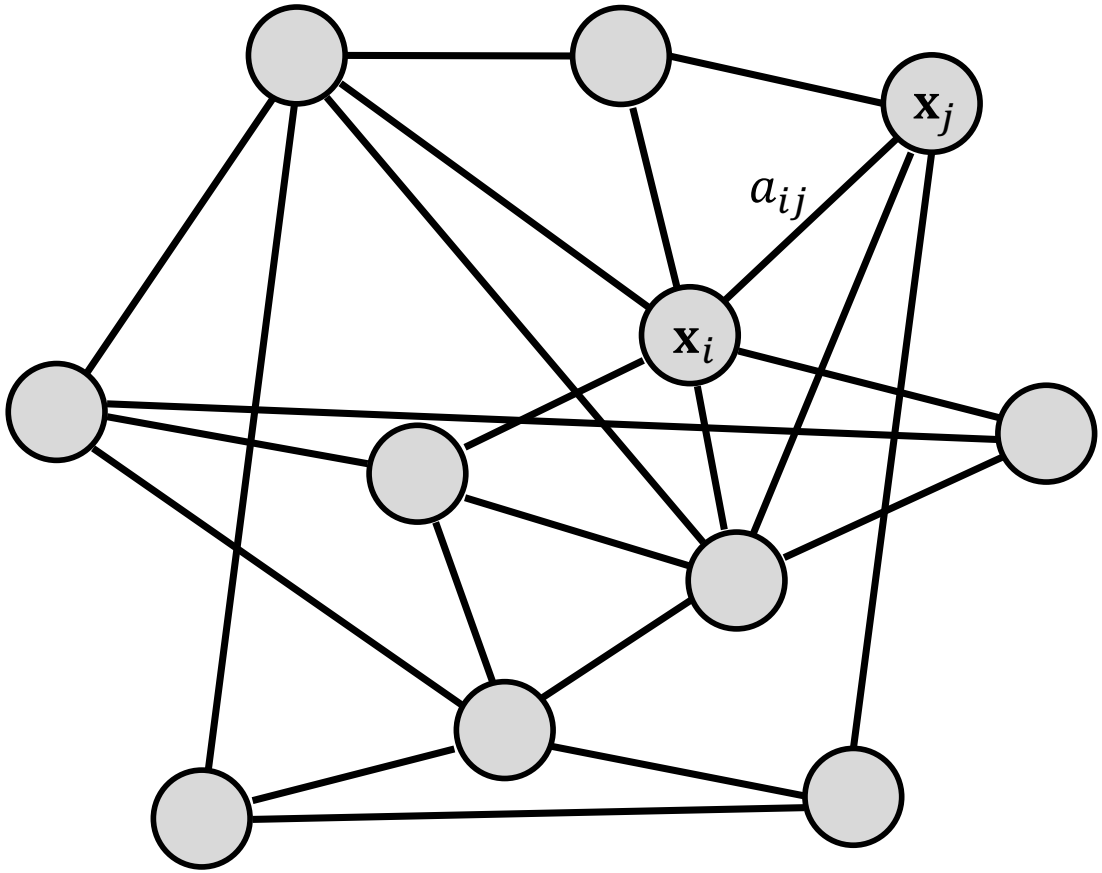
- **Neighbourhood** $\mathcal{N}(i) = \{j : (i,j) \in \mathcal{E}\}$
- **Degree** $d_i = |\mathcal{N}(i)|$

Attributes



- **Node features** $\mathbf{x} : \mathcal{V} \rightarrow \mathbb{R}^d$ $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^{\mathrm{T}}$
- **Edge features** $\mathbf{e} : \mathcal{E} \rightarrow \mathbb{R}^{d'}$

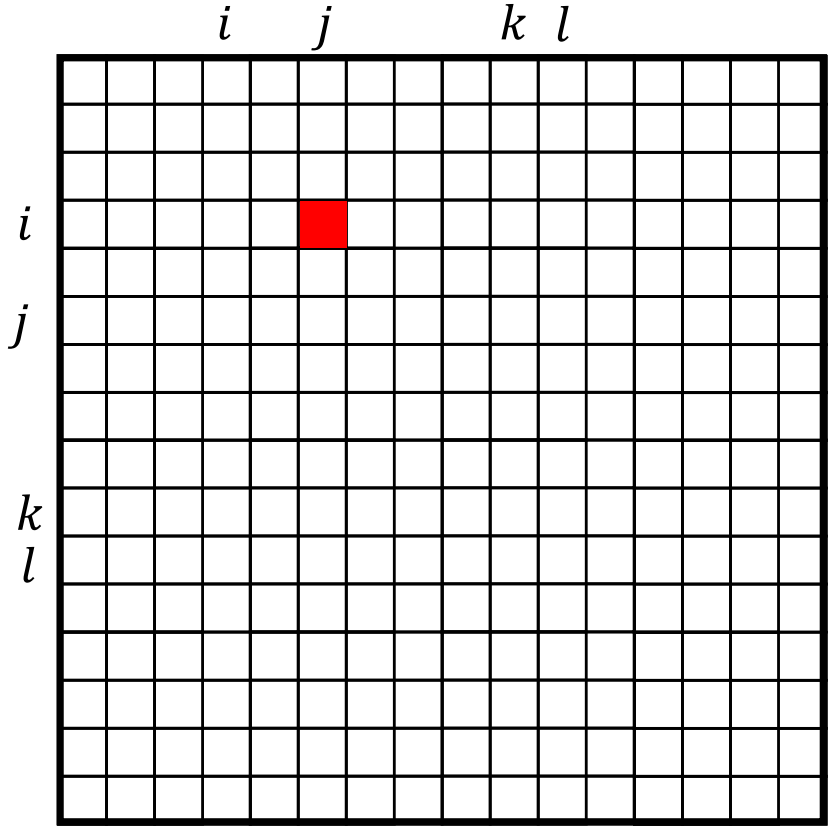
Attributes

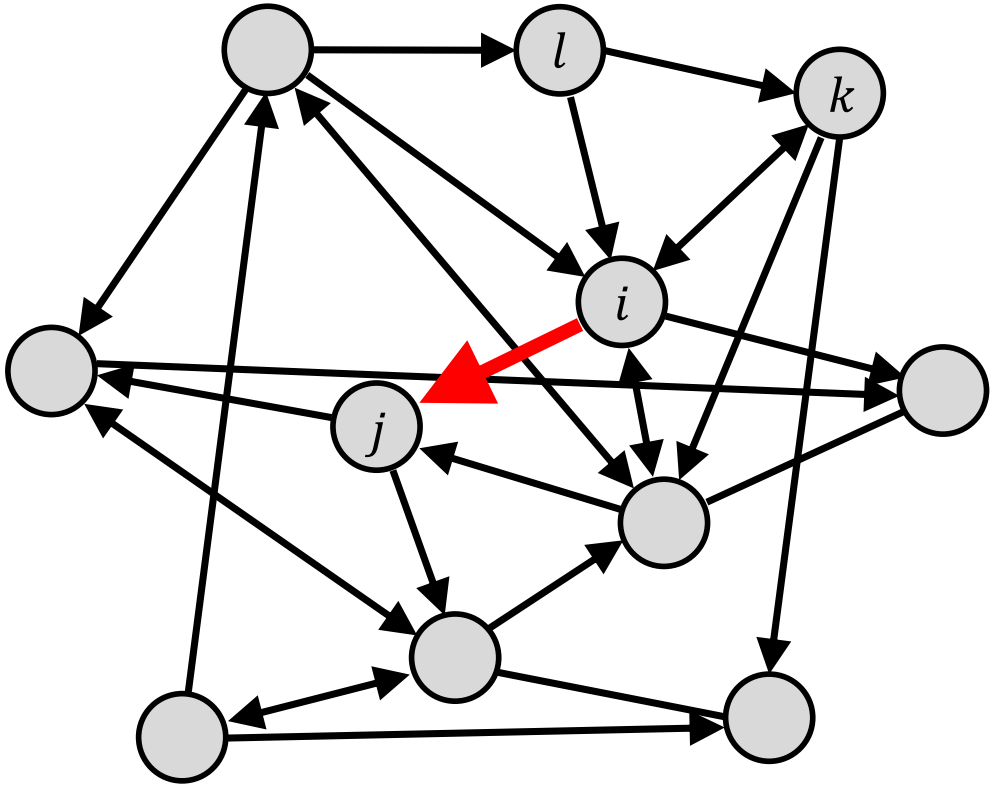


- **Node features** $\mathbf{x} : \mathcal{V} \to \mathbb{R}^d$ $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^{\mathrm{T}}$
- **Edge features** $\mathbf{e} : \mathcal{E} \to \mathbb{R}^{d'}$

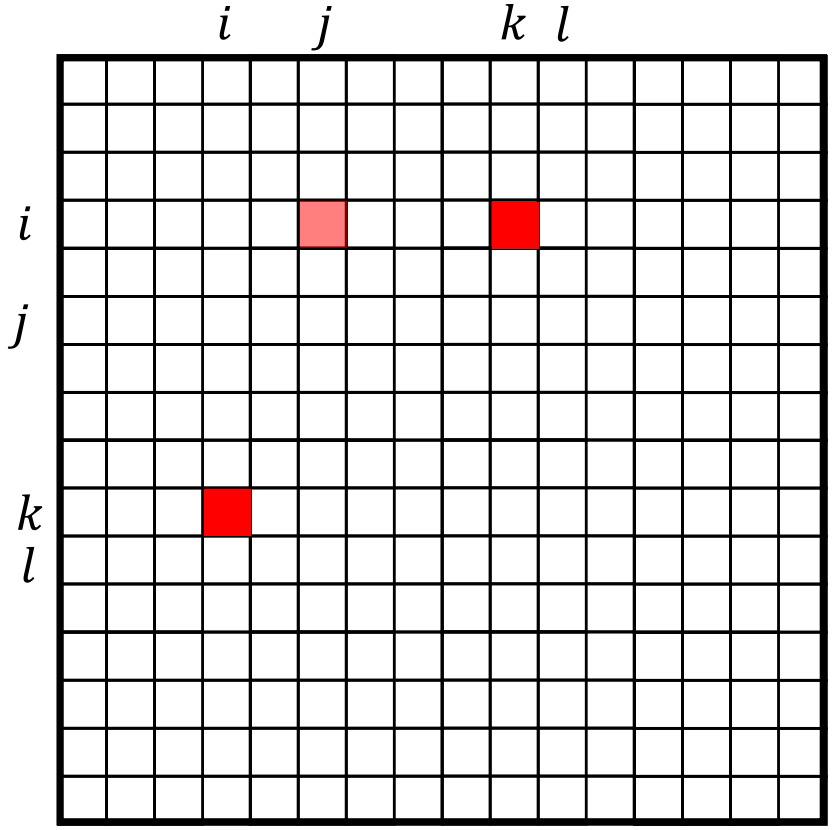
  particular case $a: \mathcal{E} \to \mathbb{R}_+$ (weighted graph)

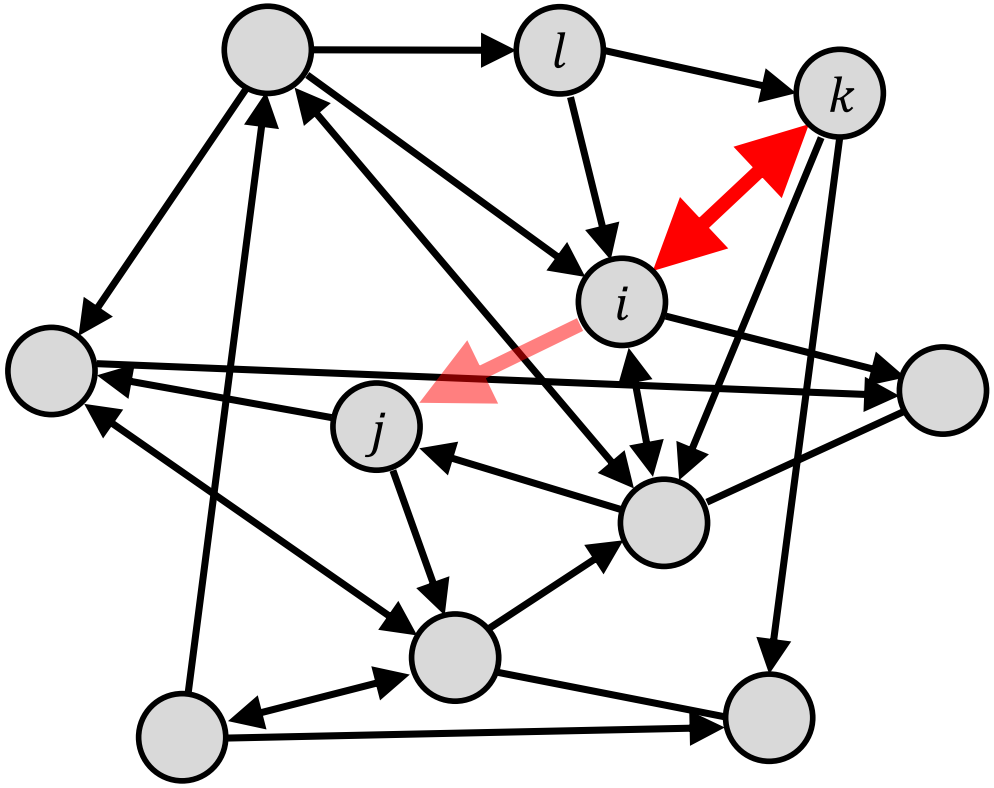
Adjacency matrix



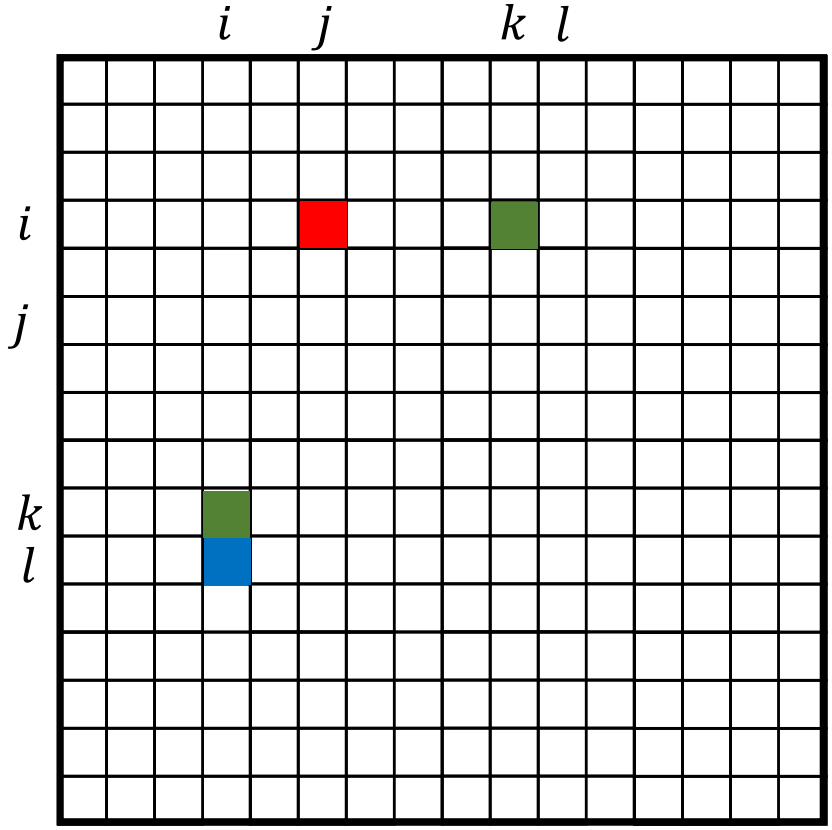


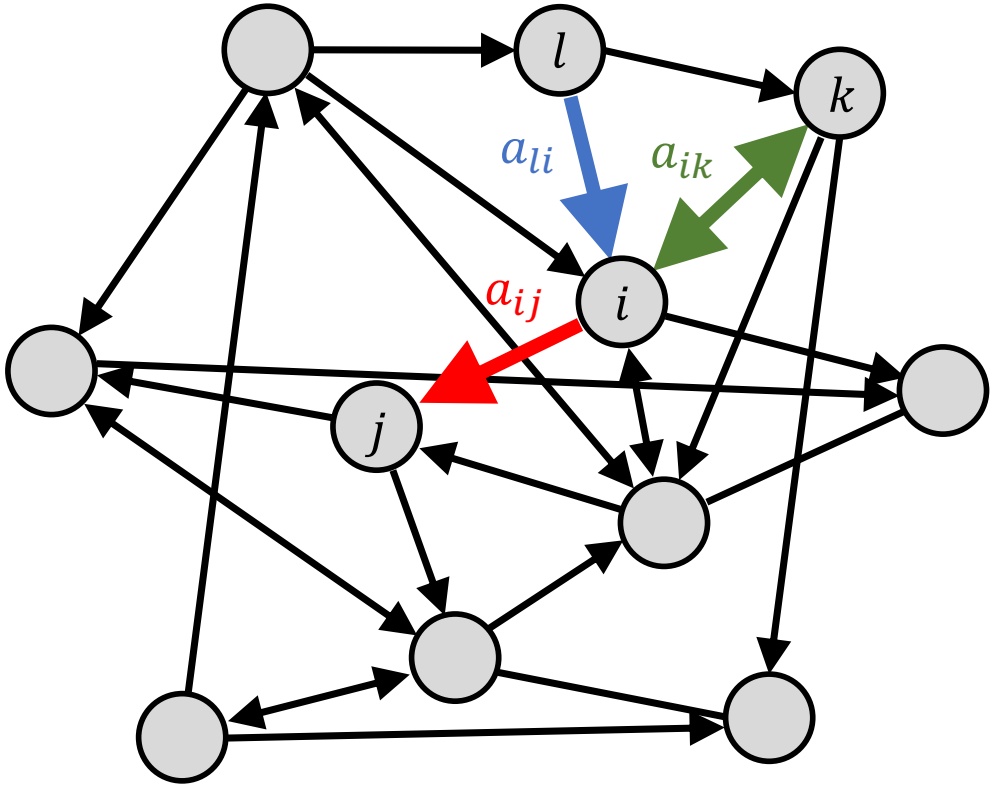
Adjacency matrix



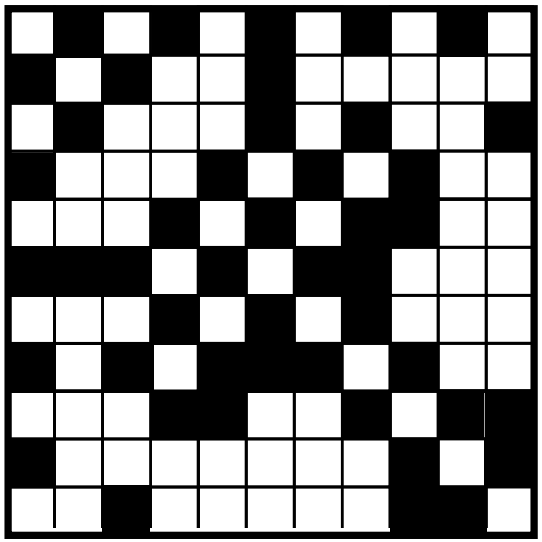


Weighted adjacency matrix

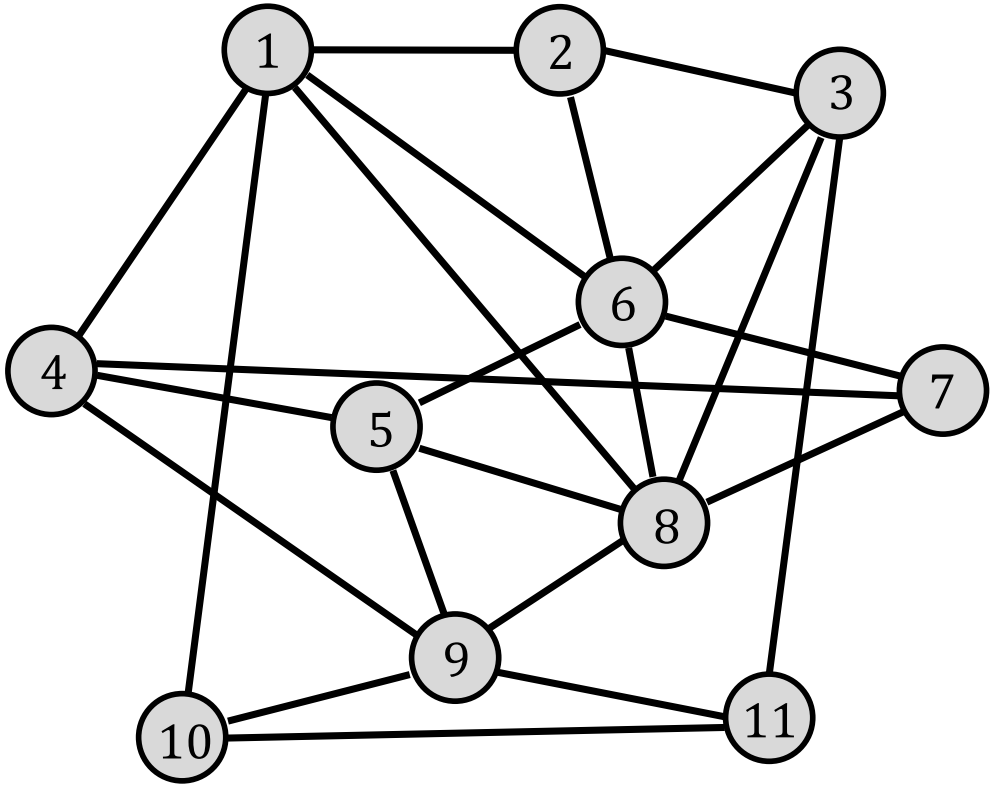




Adjacency matrix



$$a_{ij} = 1 \Leftrightarrow (i,j) \in \mathcal{E}$$

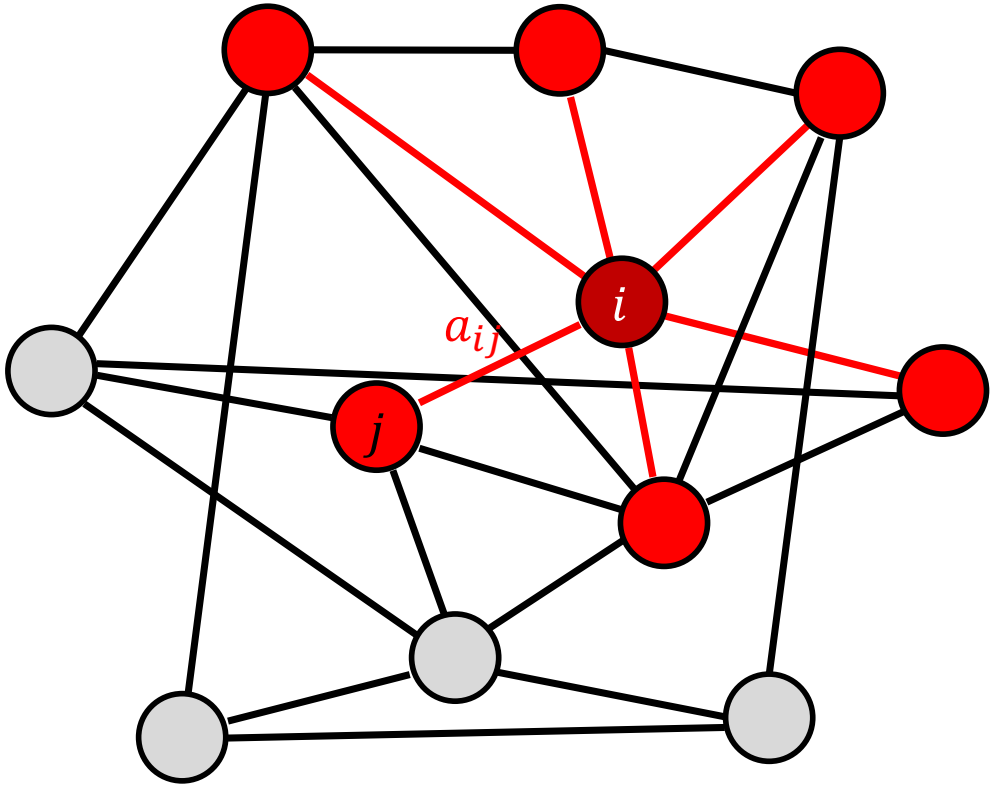


adjacency matrix symmetric for undirected graphs

Graph Laplacian

$$(\mathbf{\Delta x})_i = \mathbf{x}_i - \frac{1}{d_i} \sum_{j \in \mathcal{N}(i)} a_{ij} \mathbf{x}_j$$

"Local difference operator"



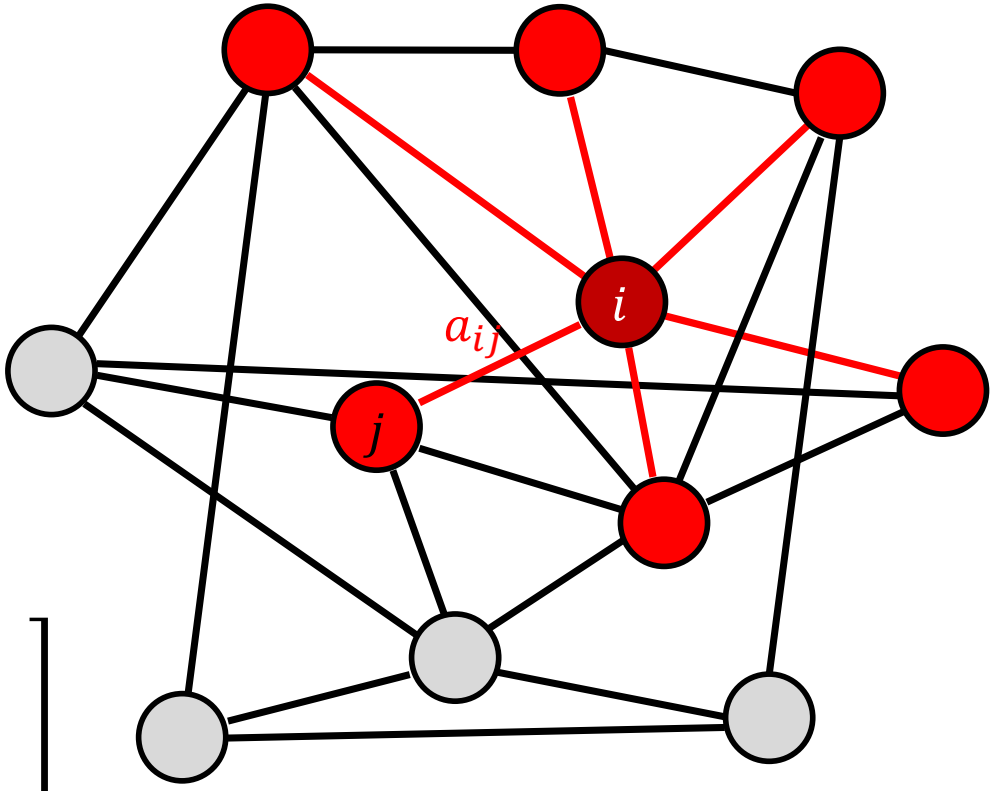
Graph Laplacian

$$(\mathbf{\Delta x})_i = \frac{1}{d_i} \sum_{j \in \mathcal{N}(i)} a_{ij} (\mathbf{x}_i - \mathbf{x}_j)$$

(normalized) Laplacian matrix

$$\mathbf{\Delta} = \mathbf{D}^{-1}(\mathbf{D} - \mathbf{A}) = \mathbf{I} - \mathbf{D}^{-1}\,\mathbf{A}$$

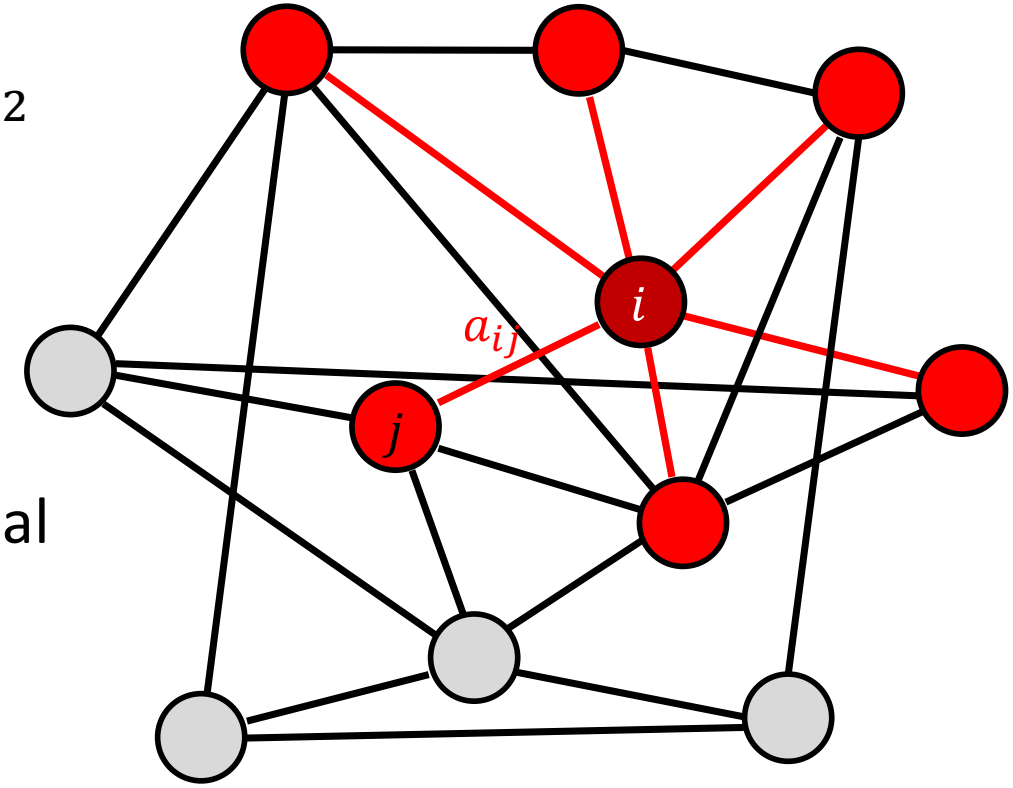
Degree matrix $\mathbf{D} = \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{bmatrix}$



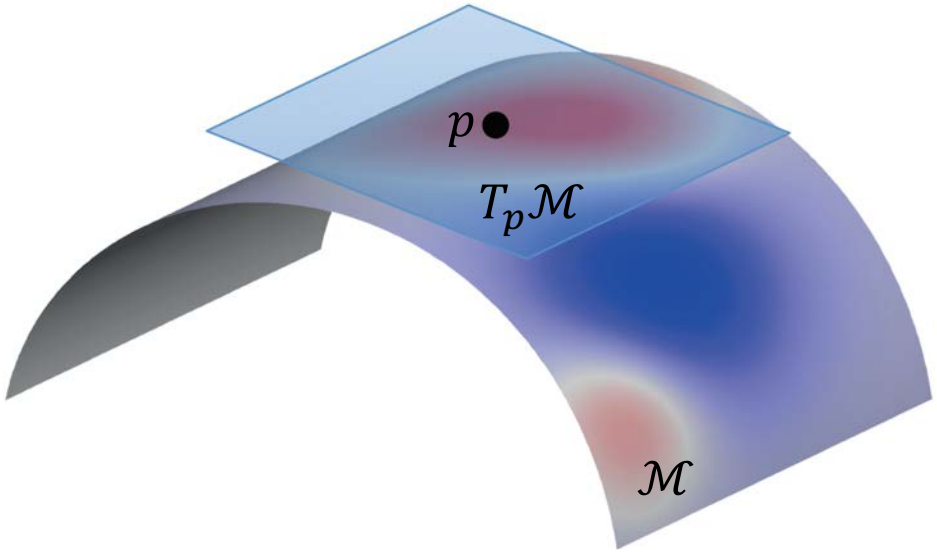
Dirichlet energy

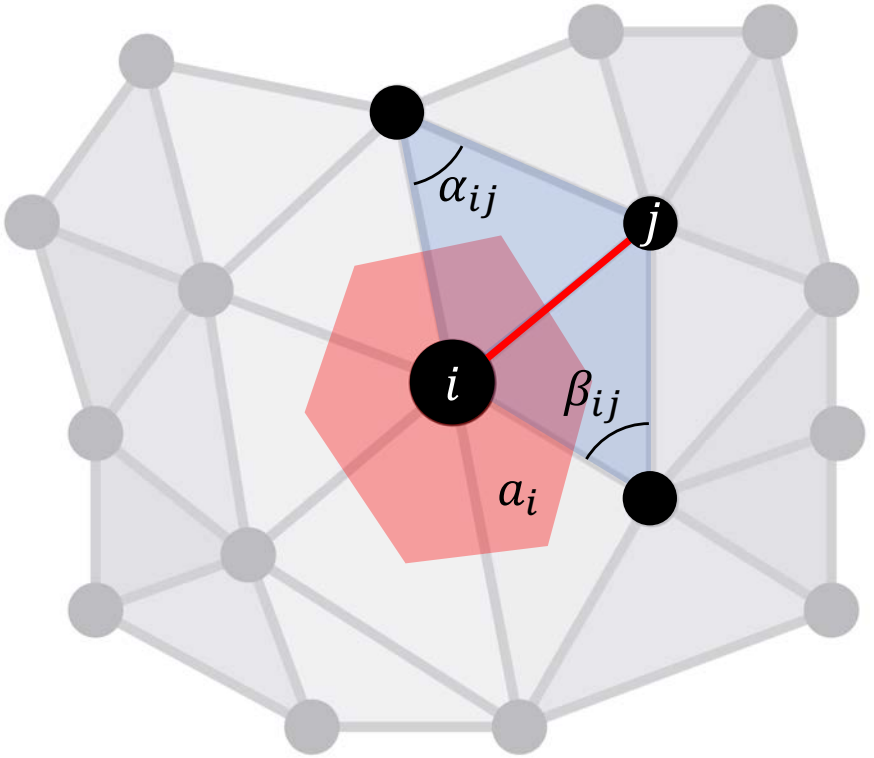
$$E(\mathbf{X}) = \frac{1}{d_i} \sum_{j \in \mathcal{N}(i)} a_{ij} \left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2$$
$$= \operatorname{trace}\left(\mathbf{X}\boldsymbol{\Delta}\mathbf{X}^{\mathrm{T}}\right)$$

measures how **smooth** the signal is on the graph



Laplacian on manifolds and meshes





Laplace-Beltrami operator

$$\Delta f = \mathrm{div}(\nabla f)$$

Laplace 1787; Beltrami 1902

Cotangent Laplacian

$$(\mathbf{\Delta x})_i = \frac{1}{a_i} \sum_{j \in \mathcal{N}(i)} \frac{\cot\alpha_{ij} + \cot\beta_{ij}}{2} \left(\mathbf{x}_i - \mathbf{x}_j\right)$$

Duffin 1959; Pinkal, Polthier 1993; Desbrun et al. 1999

Convolution on graphs?

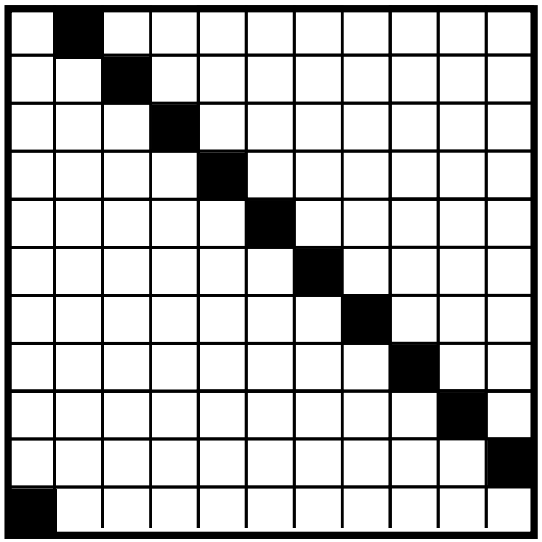
Spatial domain

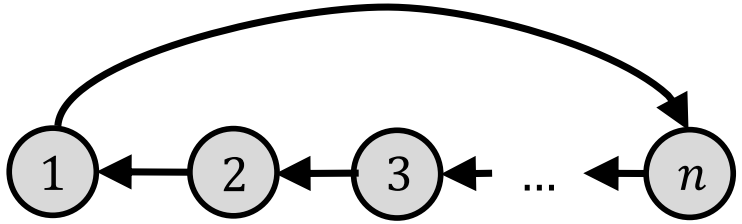
local aggregation on adjacent nodes with shared parameters

Frequency domain

Graph Fourier transform

1D grid = ring graph





adjacency matrix A of ring graph = shift operator S^T

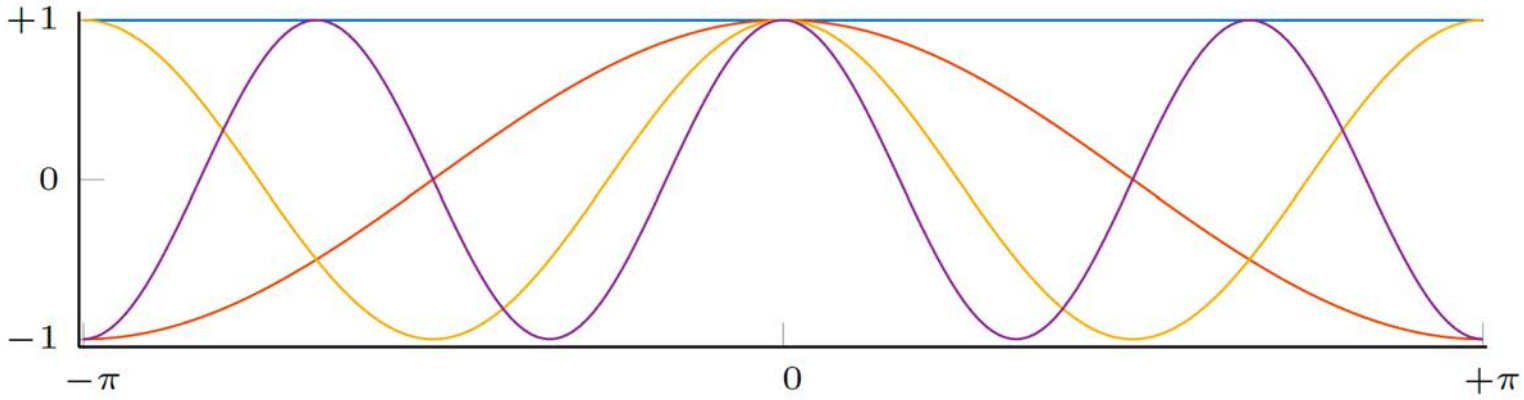
Ortega et al. 2017

From Grid to Graph Fourier Transform

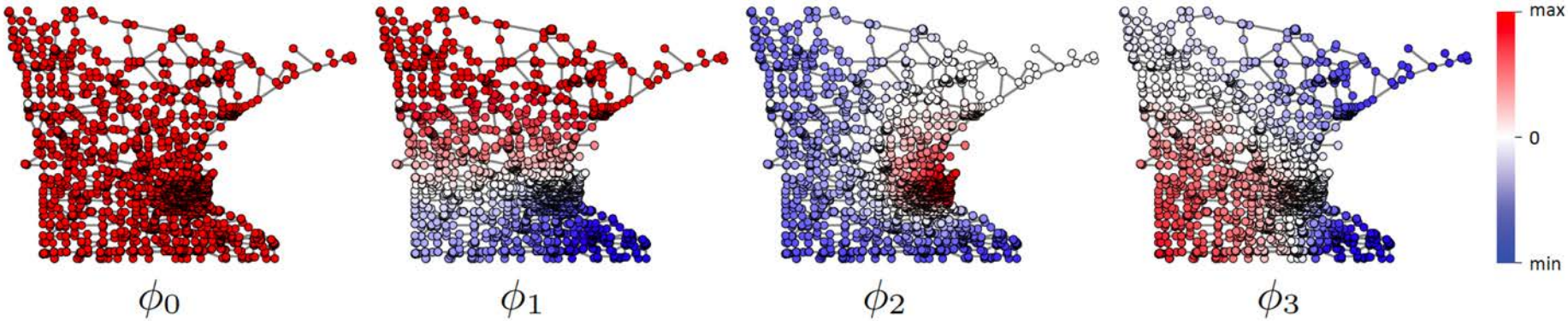
- **Key idea:** use eigenvectors of the **adjacency matrix $\mathbf{A}$** or **graph Laplacian $\boldsymbol{\Delta}$** as the analogy of Fourier transform
- The two are equivalent on grids (all circulant matrices are jointly diagonalizable) but different on graphs
- **Undirected graphs** (with **symmetric $\mathbf{A}$** and **$\boldsymbol{\Delta}$**) have **orthogonal eigenvectors**
- **Directed graphs** (with **asymmetric $\mathbf{A}$** and **$\boldsymbol{\Delta}$**) require more elaborate analysis using Jordan decomposition/generalized eigenvectors

Taubin 1995; Karni, Gotsman 2000; Levy 2006; B et al. 2010-2014
Shuman et al. 2013; Sandryhaila, Moura 2013

Graph Fourier Transform



First eigenvectors of ring graph Laplacian = classical Fourier basis



First eigenvectors of the Minnesota road network Laplacian

Spectral graph convolution

$$\mathbf{x} \star \mathbf{w} = \mathbf{\Phi} \begin{bmatrix} \widehat{w}_1 & & \\ & \ddots & \\ & & \widehat{w}_n \end{bmatrix} \mathbf{\Phi}^{\mathrm{T}} \mathbf{x}$$

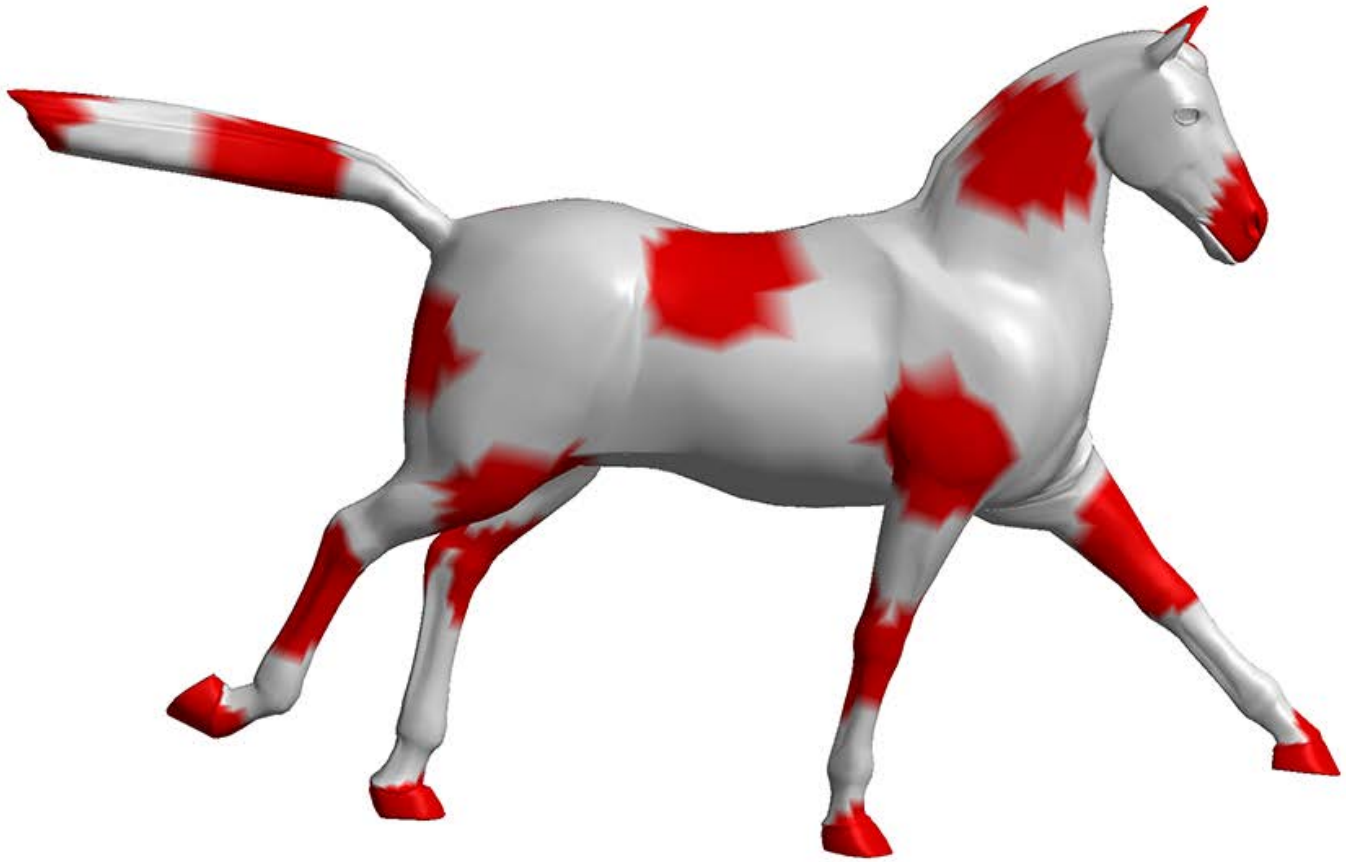
In order to compute convolution $\mathbf{x} \star \mathbf{w}$

- **Graph FT:** $\hat{\mathbf{x}} = \mathbf{\Phi}^{\mathrm{T}} \mathbf{x}$ $O(n^2)+O(n^3)$
- **Apply filter:** $\hat{\mathbf{x}} \circ \widehat{\mathbf{w}} = \begin{bmatrix} \widehat{w}_1 & & \\ & \ddots & \\ & & \widehat{w}_n \end{bmatrix} \hat{\mathbf{x}}$ $O(n)$
- **Inverse graph FT:** $\mathbf{x} \star \mathbf{w} = \mathbf{\Phi}(\hat{\mathbf{x}} \circ \widehat{\mathbf{w}})$ $O(n^2)$

Spectral graph convolution: drawbacks

- **Computational complexity** at least $O(n^2)$ vs $O(n)$ or $O(n\log n)$
- **Number of parameters** $O(n)$ vs $O(1)$
- **No guarantee of spatial localization**
- **Isotropic filters**
- **Filters are basis-dependent and do not generalize across graphs**

Basis dependence

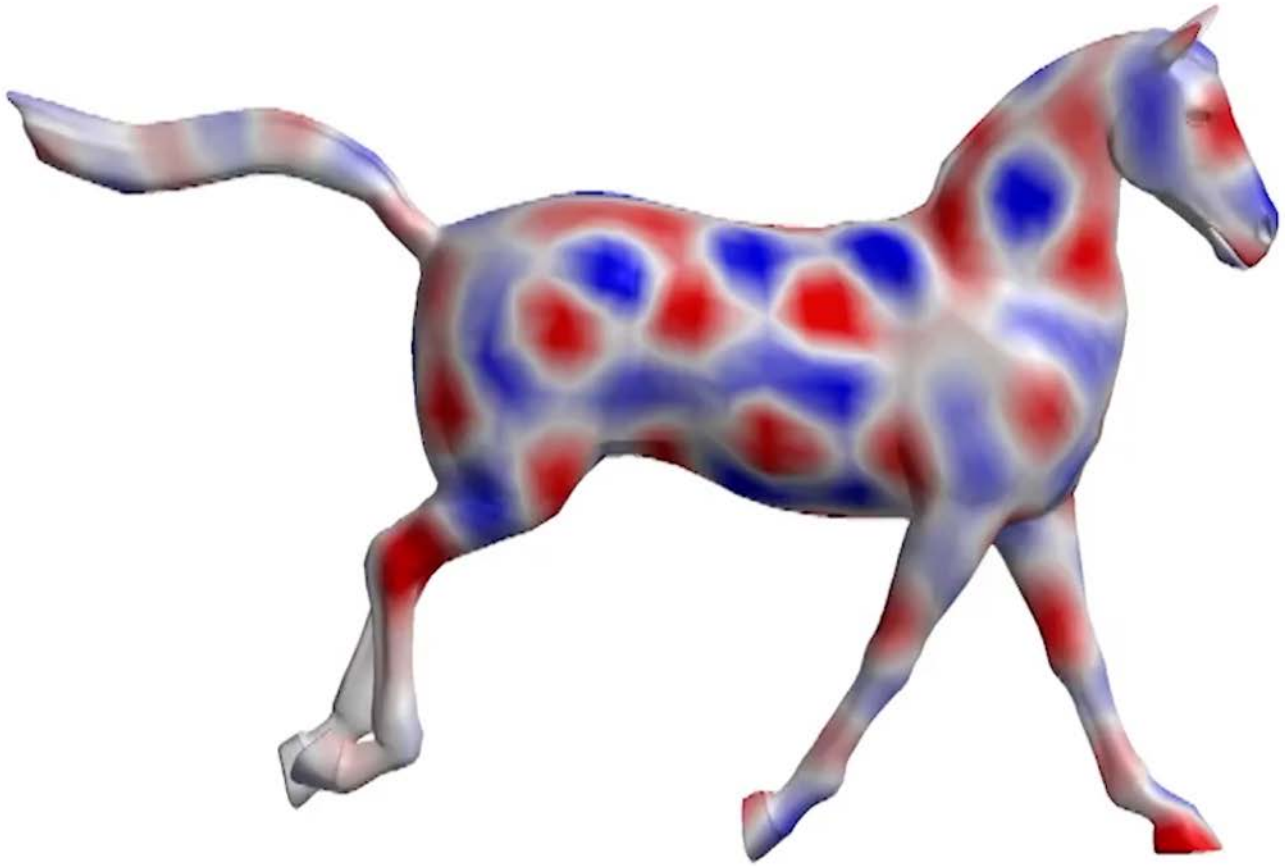


Original signal x

Basis dependence

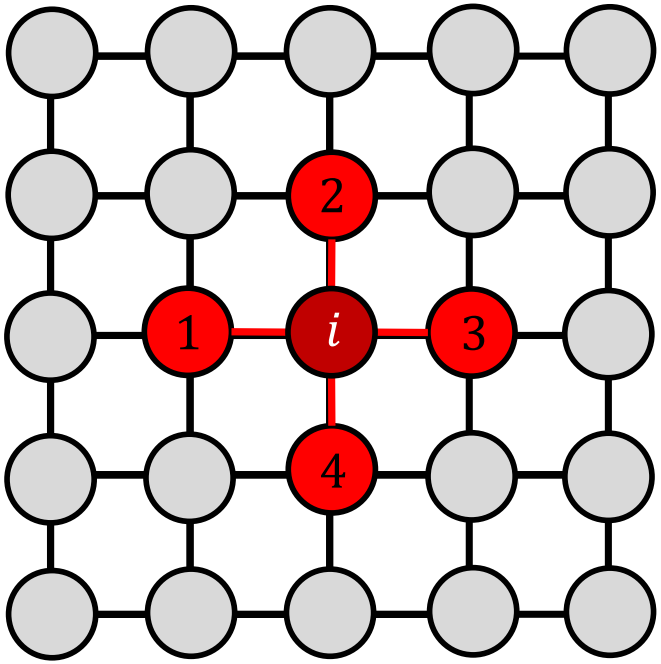
$$\textbf{Filter output } \mathbf{x} \star \mathbf{w} = \boldsymbol{\Phi} \begin{bmatrix} \widehat{w}_1 & & \\ & \ddots & \\ & & \widehat{w}_n \end{bmatrix} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{x}$$

Basis dependence

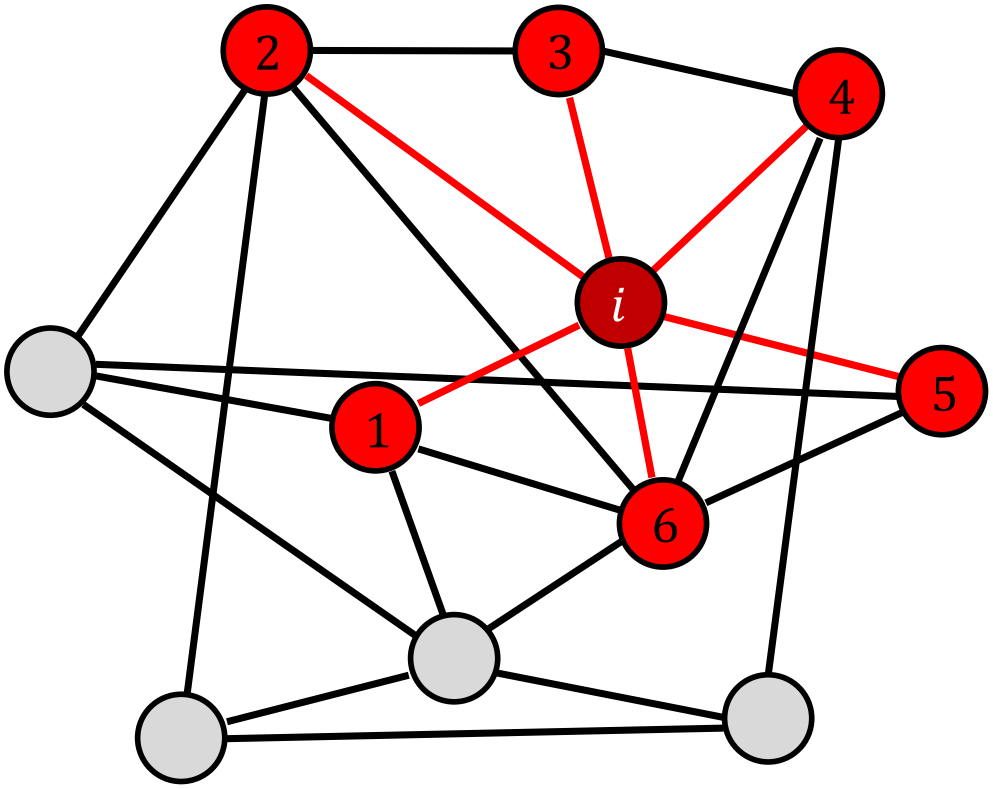


$$\textbf{Filter output } \mathbf{x} \star \mathbf{w} = \boldsymbol{\Psi} \begin{bmatrix} \widehat{w}_1 & & \\ & \ddots & \\ & & \widehat{w}_n \end{bmatrix} \boldsymbol{\Psi}^{\mathrm{T}} \mathbf{x}$$

Isotropic filters on graphs

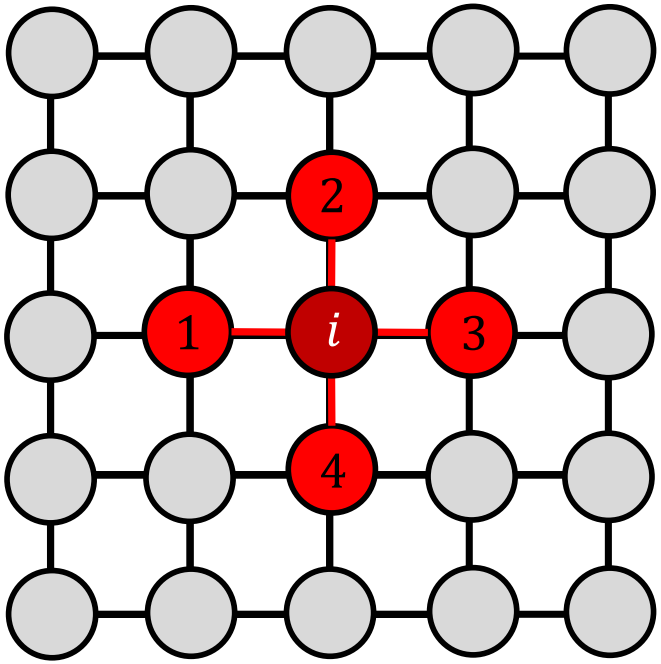


Grid

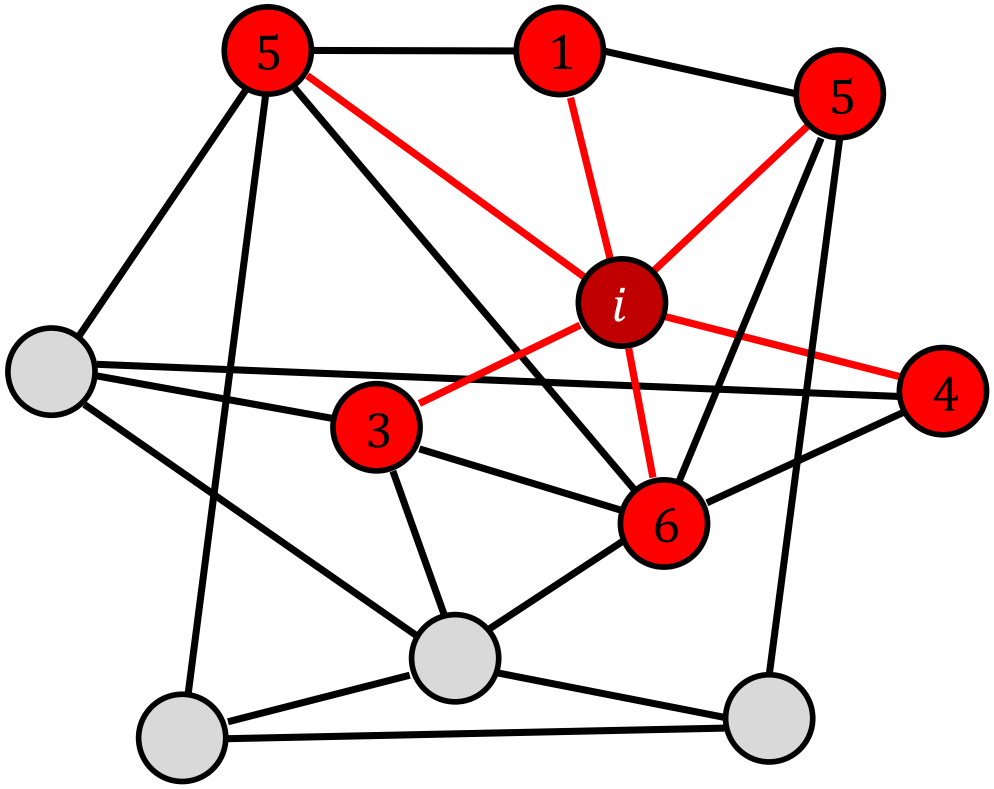


Graph

Isotropic filters on graphs



Grid



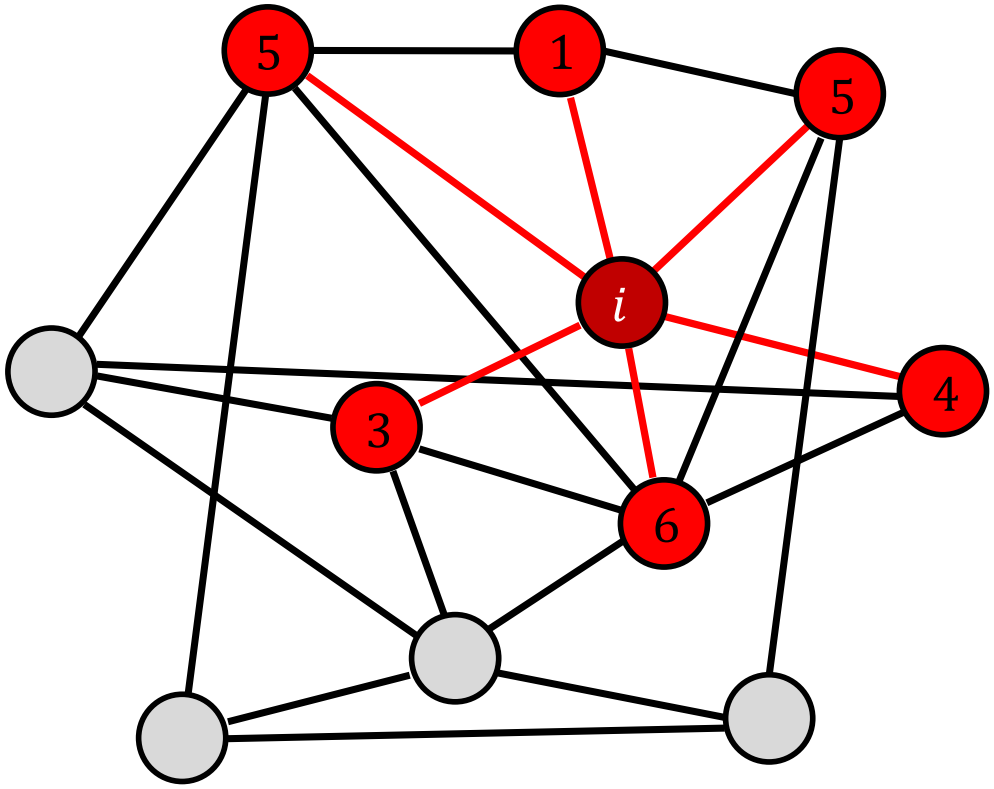
Graph

local permutation invariance

(no “directions” on graphs)

Isotropic filters on graphs

$$(\mathbf{\Delta x})_i = \frac{1}{d_i} \sum_{j \in \mathcal{N}(i)} a_{ij}(\mathbf{x}_i - \mathbf{x}_j)$$

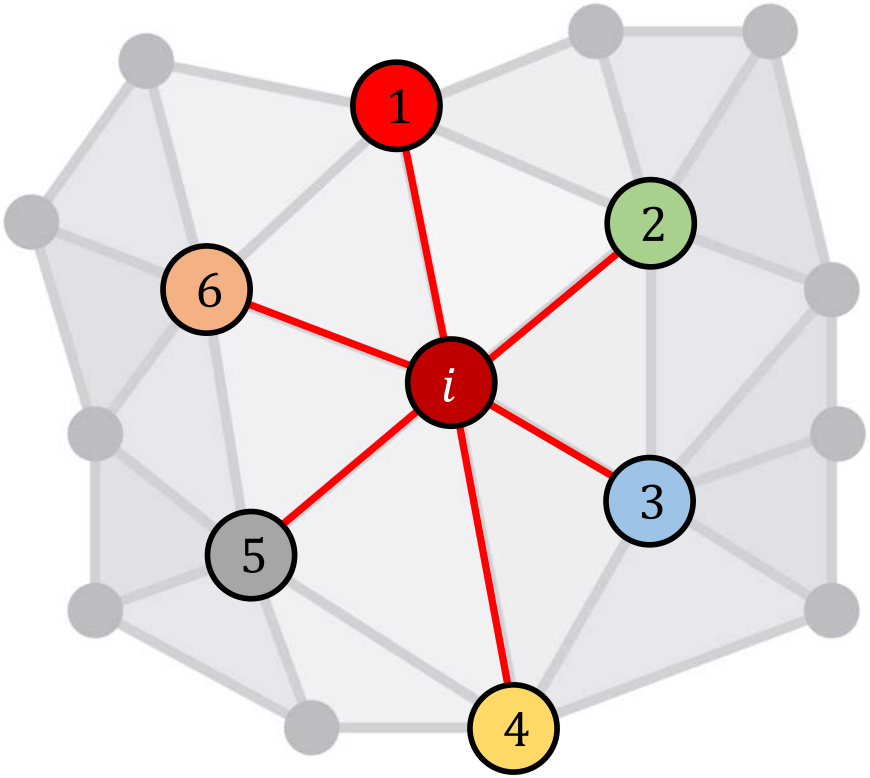


Graph

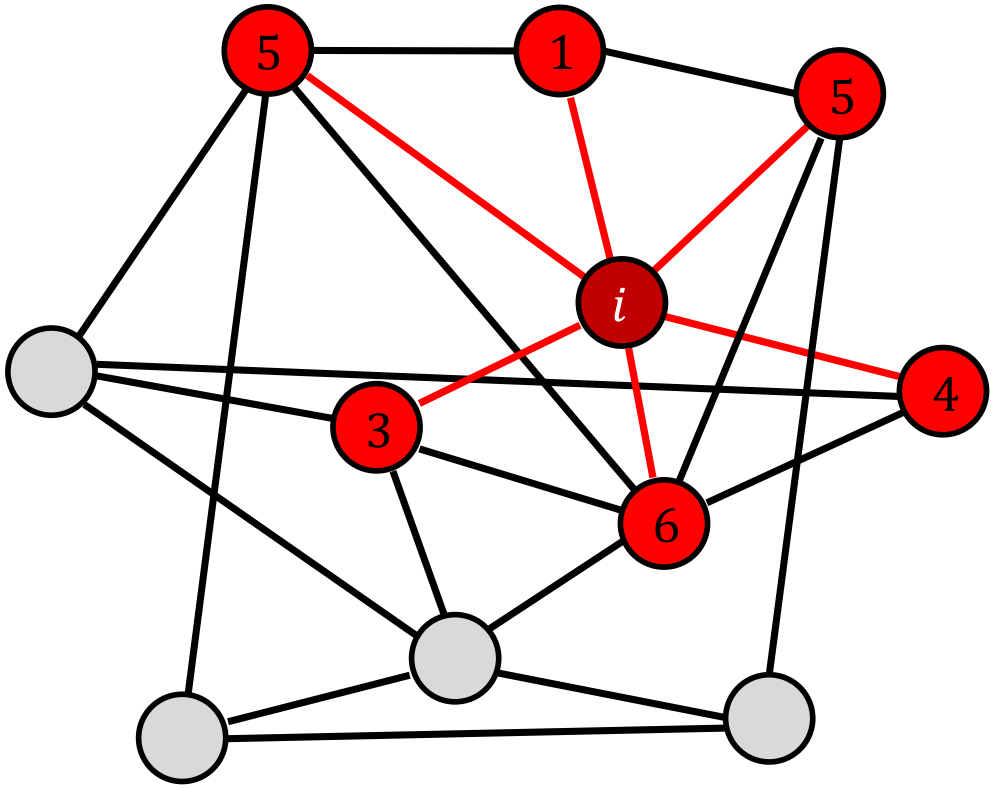
graph Laplacian is permutation-invariant

hence isotropic

Anisotropic filters on meshes

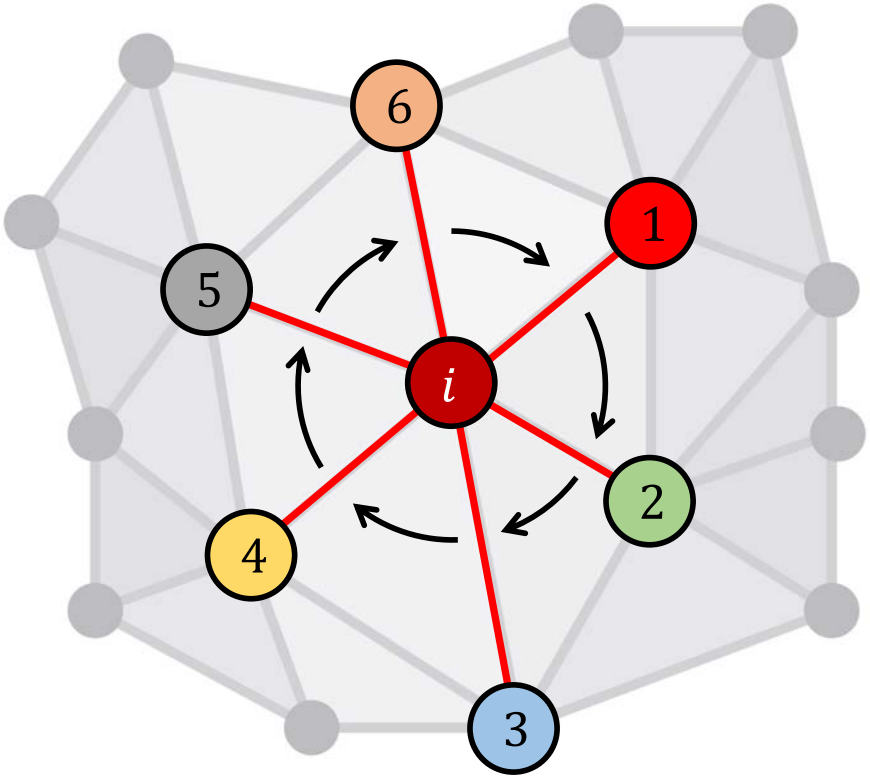


Mesh



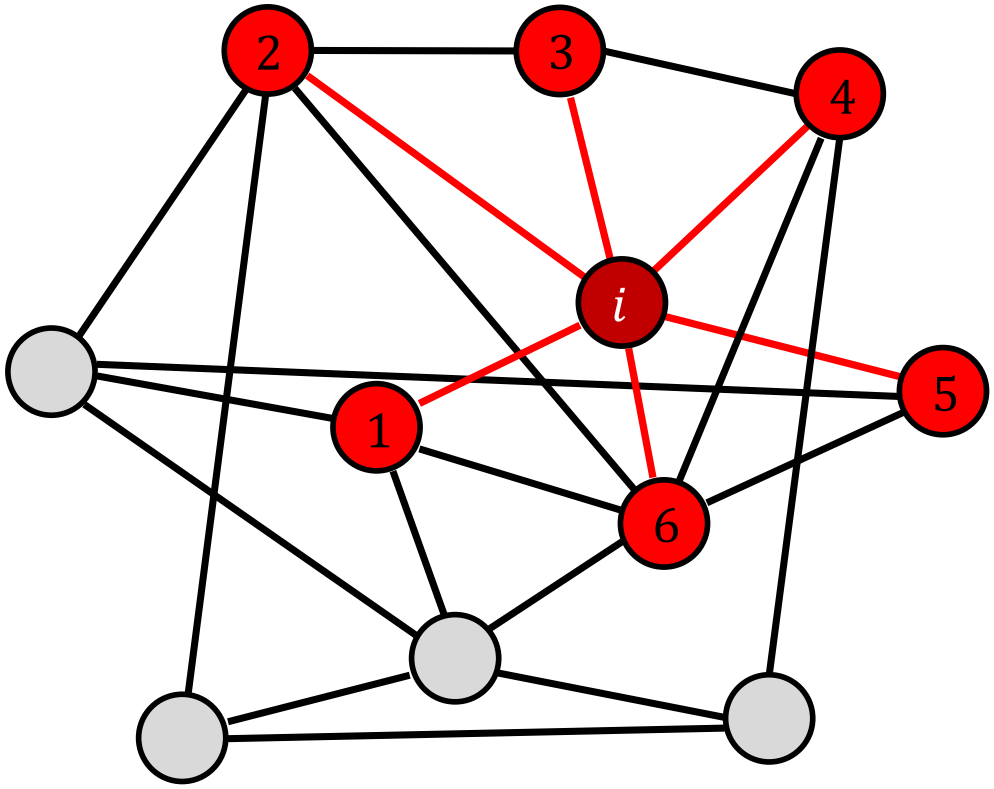
Graph

Anisotropic filters on meshes



Mesh

rotation



Graph

permutation

Anisotropic filters on meshes

Anisotropic spectral filters on meshes

Boscaini et B 2016

Spectral graph convolution, take 2

$$p(\mathbf{\Delta})\mathbf{x} = \mathbf{\Phi} \begin{bmatrix} p(\lambda_1) & & \\ & \ddots & \\ & & p(\lambda_n) \end{bmatrix} \mathbf{\Phi}^{\mathrm{T}}\mathbf{x}$$

- Matrix function applied to $\mathbf{\Delta}$
- Interpret eigenvalues $\lambda_1, \dots, \lambda_n$ as **frequencies** and $p(\lambda)$ as **spectral transfer function**
- Make $p(\lambda)$ parametric with $O(1)$ **parameters**
- Make $p(\lambda)$ expressible in terms of simple matrix operations, to avoid explicit computation of $\mathbf{\Phi}$
- Possible to guarantee **stability** under graph perturbations
- Possible to guarantee **localization**

Levie et B 2019; Levie, Monti et B 2018

Polynomial filter (ChebNet)

$$p(\lambda) = w_0 + w_1\lambda^1 + \cdots + w_p\lambda^p$$

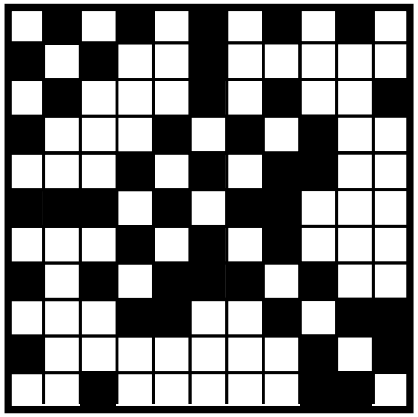
- **Number of learnable parameters $O(1)$**

Defferard et al. 2016

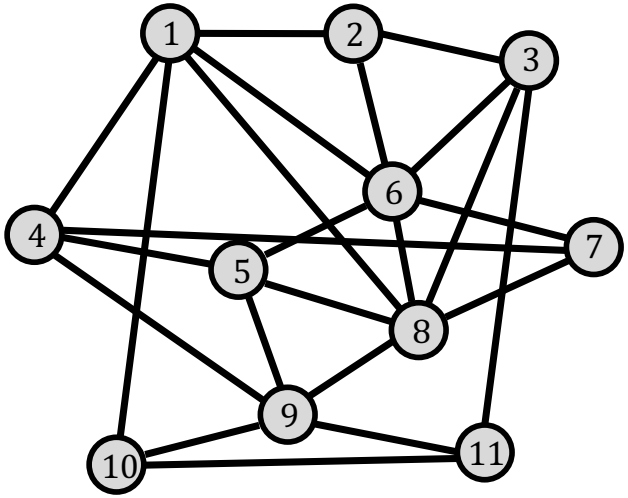
Polynomial filter (ChebNet)

$$p(\boldsymbol{\Delta}) = w_0\mathbf{I} + w_1\boldsymbol{\Delta}^1 + \cdots + w_p\boldsymbol{\Delta}^p$$

- **Number of learnable parameters** $O(1)$
- **Efficient computation** $O(|\mathcal{E}|) \sim O(n)$ avoiding graph FT altogether



$O(|\mathcal{E}|)$ non-zeros

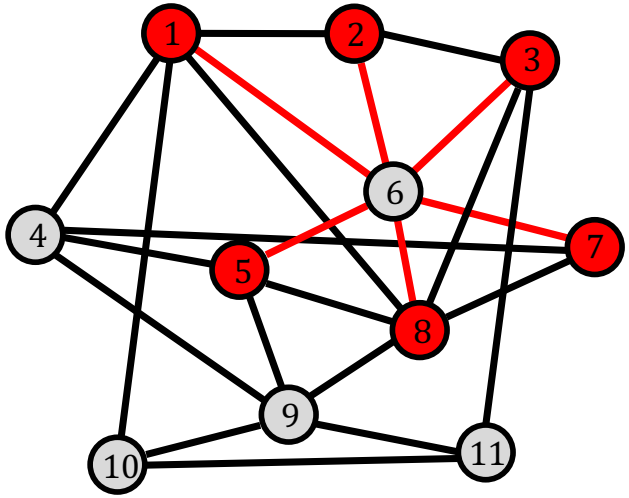


Defferard et al. 2016

Polynomial filter (ChebNet)

$$p(\boldsymbol{\Delta}) = w_0\mathbf{I} + w_1\boldsymbol{\Delta}^1 + \cdots + w_p\boldsymbol{\Delta}^p$$

- **Number of learnable parameters** $O(1)$
- **Efficient computation** $O(|\mathcal{E}|) \sim O(n)$ avoiding graph FT altogether
- **Localization to *p*-hops** since $\boldsymbol{\Delta}^p$ is localized to *p*-hops



Defferard et al. 2016

Polynomial filter (ChebNet)

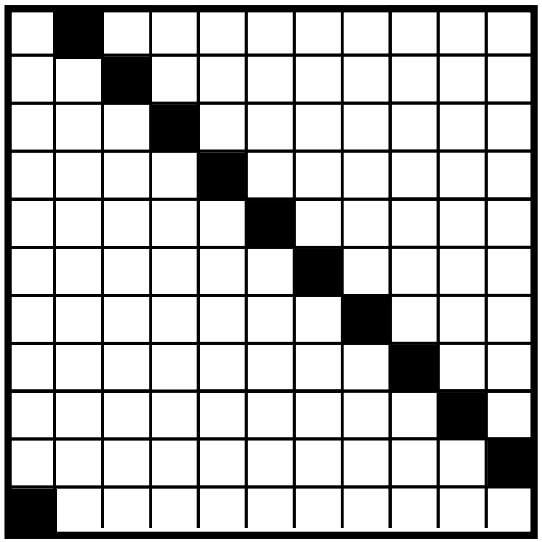
$$p(\boldsymbol{\Delta}) = w_0\mathbf{I} + w_1\boldsymbol{\Delta}^1 + \cdots + w_p\boldsymbol{\Delta}^p$$

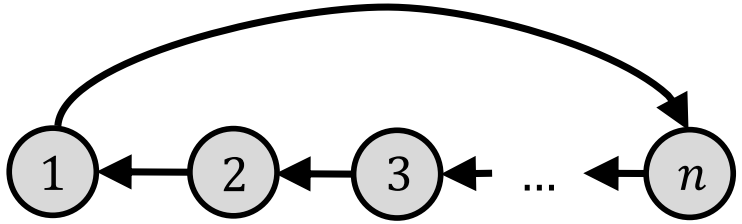
- **Number of learnable parameters** $O(1)$
- **Efficient computation** $O(|\mathcal{E}|) \sim O(n)$ avoiding graph FT altogether
- **Localization to *p*-hops** since $\boldsymbol{\Delta}^p$ is localized to *p*-hops
- **Generalization** across graphs (stability under graph perturbation)
- Can be used with other operators, e.g. $\mathbf{A}$
- Can be used with directed graphs

Defferard et al. 2016

Spatial graph convolution

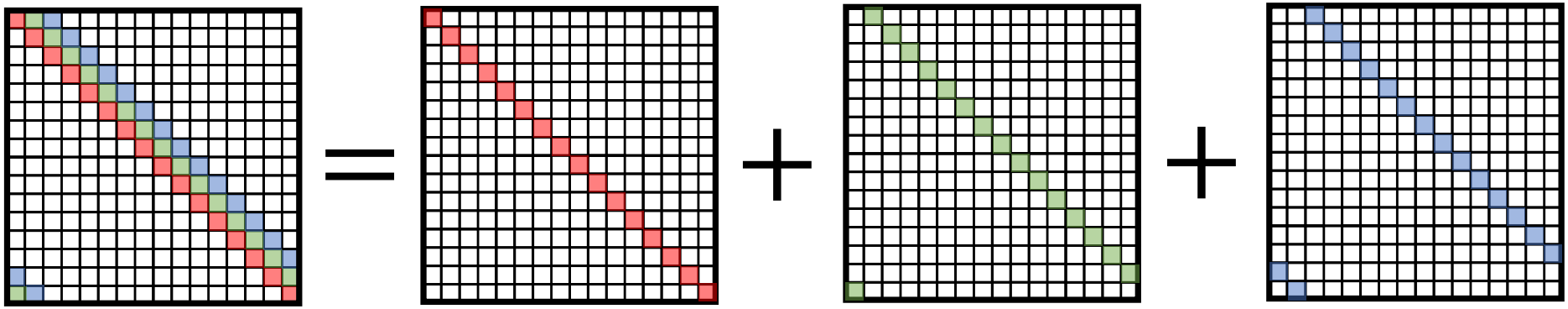
1D grid = ring graph





adjacency matrix = Shift operator

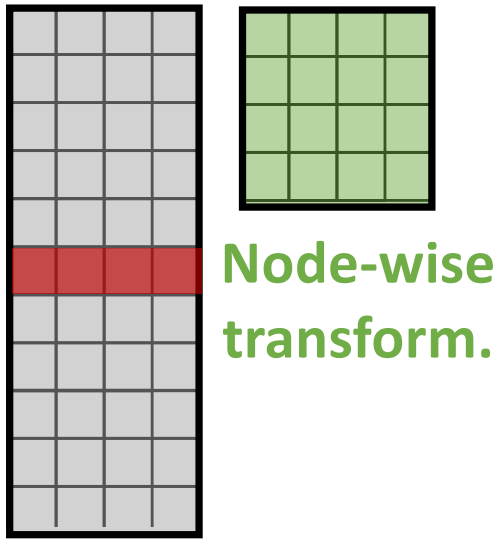
Convolution, revisited



$$p(\mathbf{A}) = w_0\mathbf{I} + w_1\mathbf{A}^1 + w_2\mathbf{A}^2$$

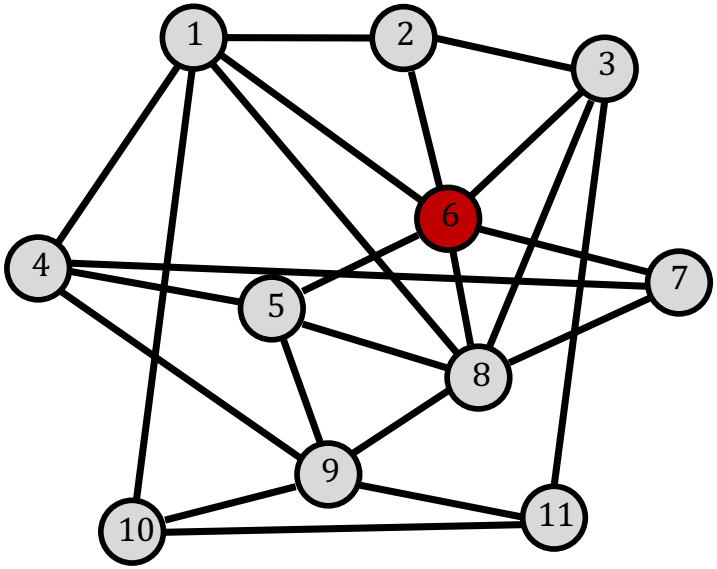
Sandryhaila, Moura 2013

Graph Convolutional Networks (GCN)



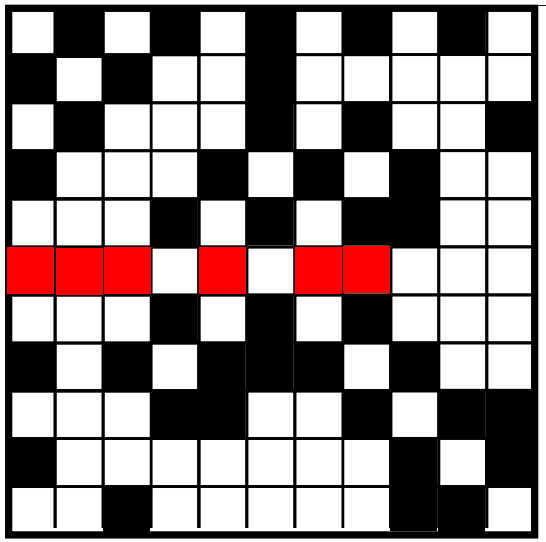
Node-wise transform.

Node-wise features

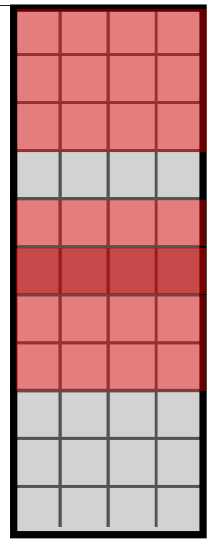


Kipf, Welling 2016

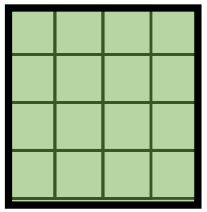
Graph Convolutional Networks (GCN)



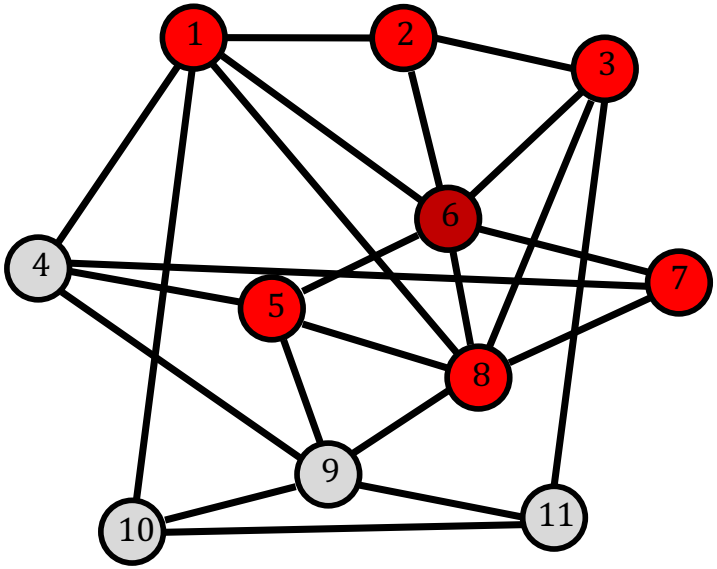
Graph diffusion



Node-wise features



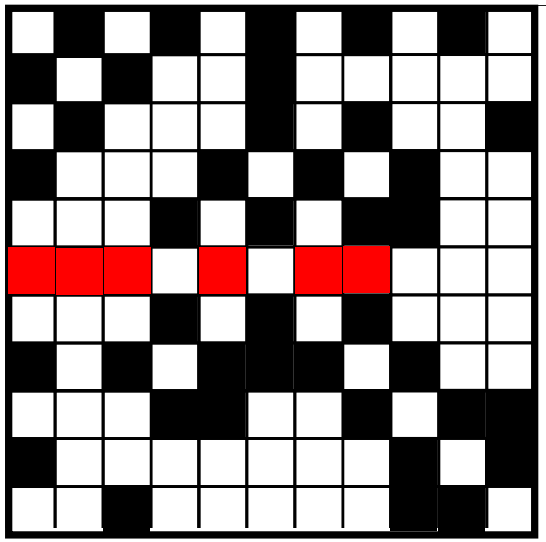
Node-wise transform.



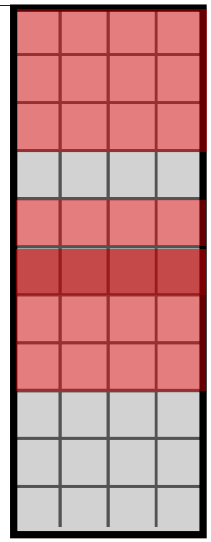
$$\mathbf{Y} = \mathrm{ReLU}(\mathbf{AXW})$$

Kipf, Welling 2016

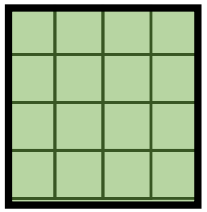
Graph Convolutional Networks (GCN)



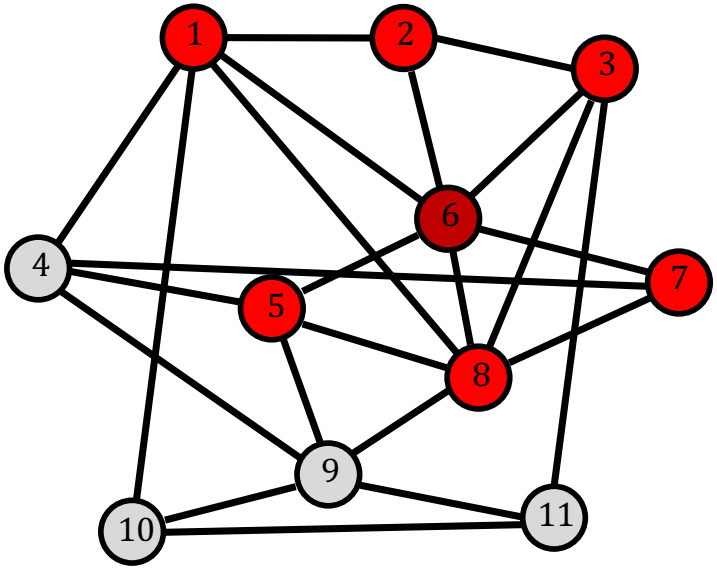
Graph diffusion



Node-wise features



Node-wise transform.

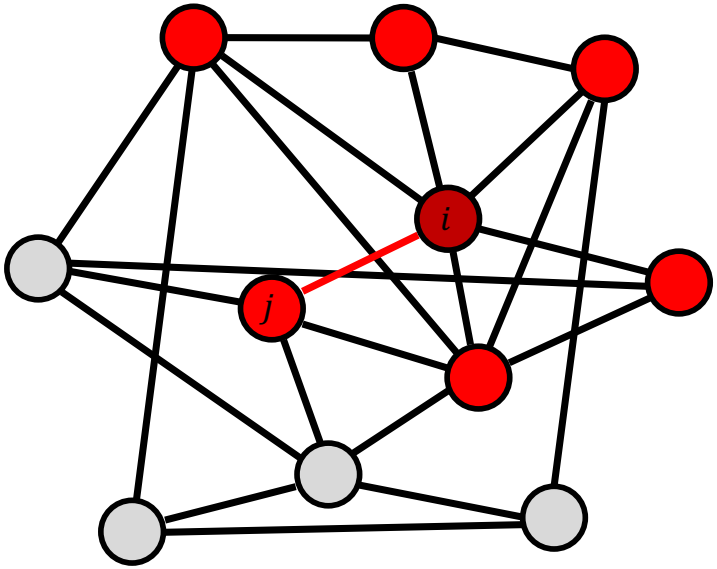


$$\mathbf{Y} = \mathrm{softmax}(\mathbf{A}\mathrm{ReLU}(\mathbf{A}\mathbf{X}\mathbf{W}_1)\mathbf{W}_2)$$

Kipf, Welling 2016

Graph Attention Networks (GAT)

$$\mathbf{y}_i = \sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{x}_j$$



attention score

$$\alpha_{ij} = \frac{\exp\left(\xi\left(\left[\mathbf{x}_i \mathbf{W}, \mathbf{x}_j \mathbf{W}\right]\mathbf{a}\right)\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(\xi\left(\left[\mathbf{x}_i \mathbf{W}, \mathbf{x}_k \mathbf{W}\right]\mathbf{a}\right)\right)}$$

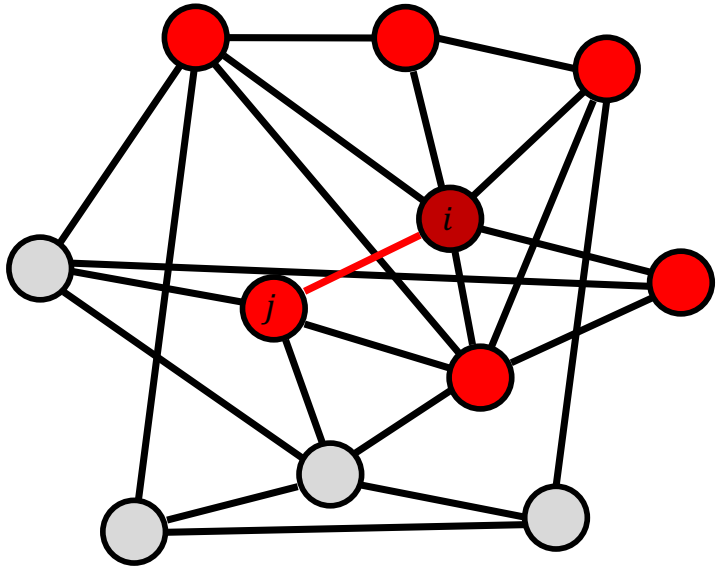
$$\mathbf{Y} = \mathbf{A}(\mathbf{X}, \mathbf{W}, \mathbf{a})\mathbf{X}$$

Monti et B 2017; Veličković et al. 2018

Message Passing Neural Network (MPNN)

General aggregation function

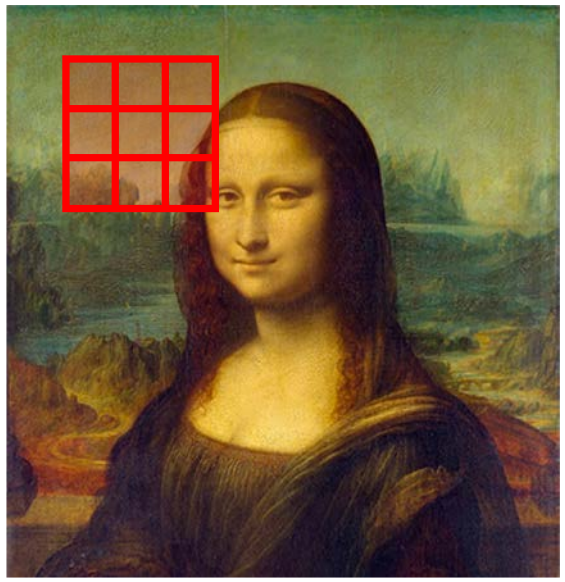
$$\mathbf{y}_i = g\left(\sum_{j \in \mathcal{N}(i)} \boldsymbol{h}\left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{ij}, \mathbf{W}\right)\right)$$



$$\mathbf{Y} = \boldsymbol{\mathcal{A}}(\mathbf{X}, \mathbf{E}, \mathbf{W})$$

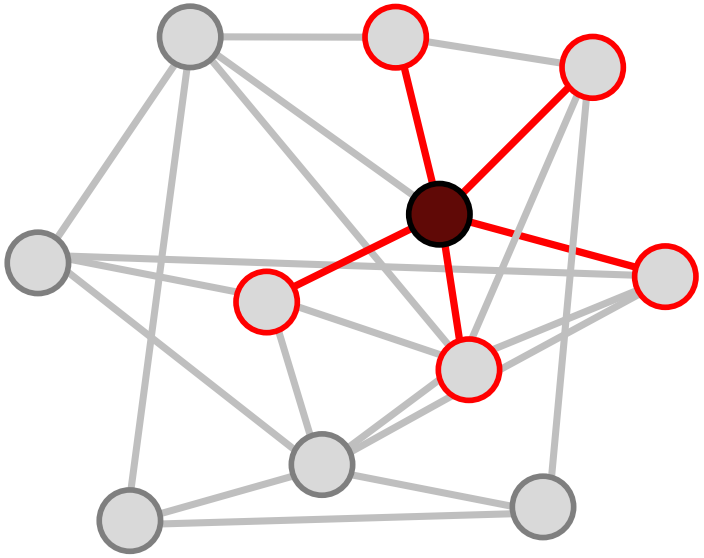
Gilmer et al. 2017 (MPNN); Wang et B 2018 (EdgeConv)

Images



- Convolution
- Local operations (window)
- Constant number of neighbours
- Fixed ordering of neighbours
- Shift equivariance
- O(n) complexity

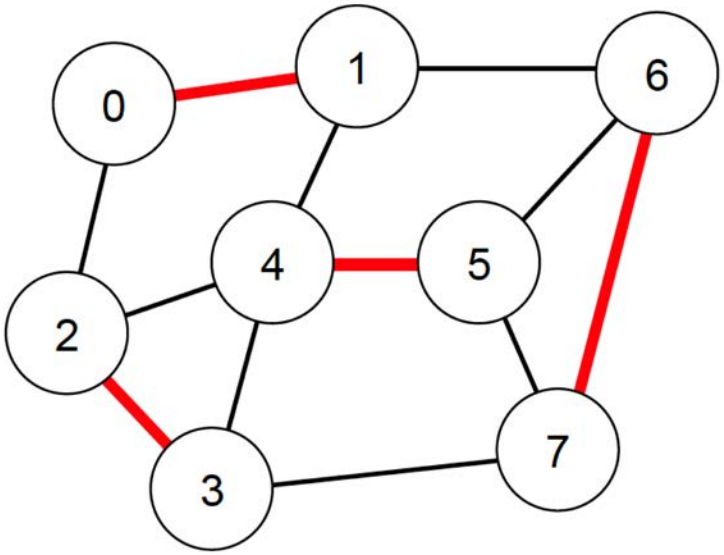
Graphs



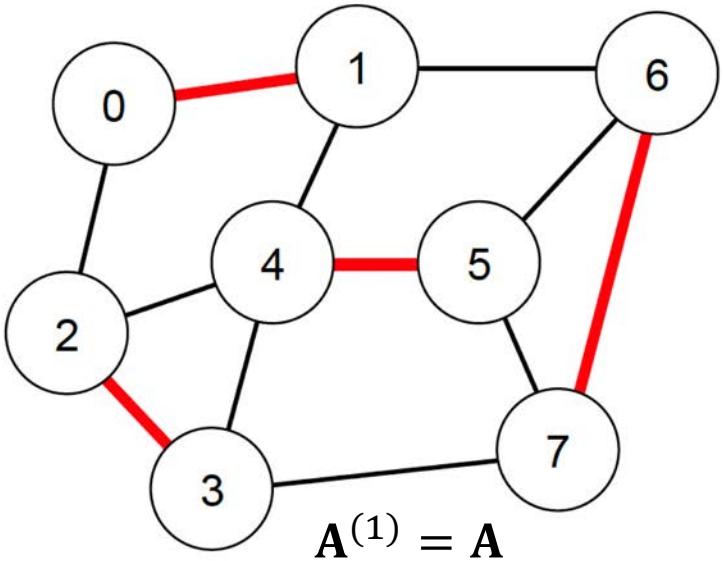
- Message passing
- Local operations (1-hop)
- Different number of neighbours
- No ordering of neighbours
- Permutation invariance
- O(n) complexity

Pooling

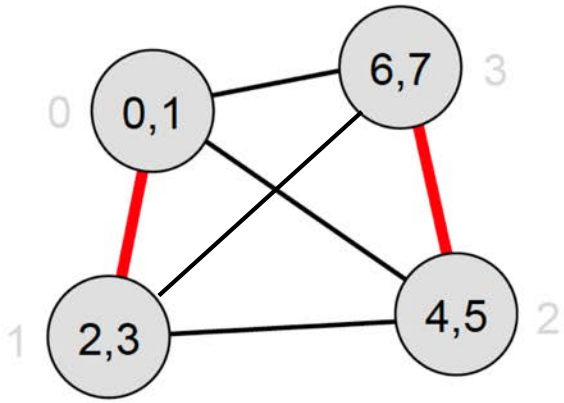
Graph coarsening



Graph coarsening



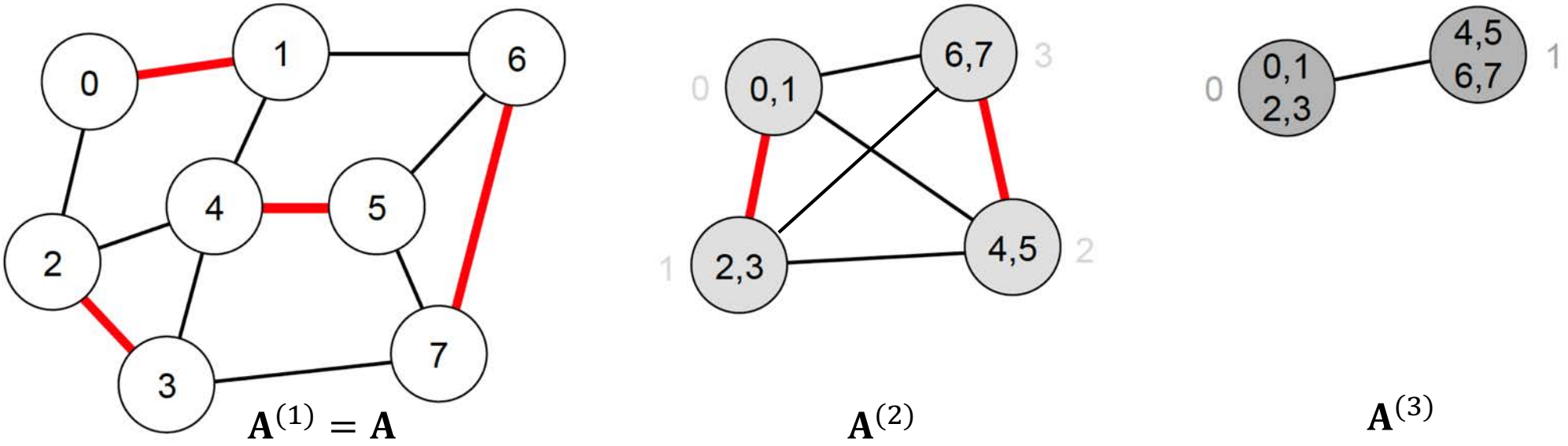
$\mathbf{A}^{(1)} = \mathbf{A}$



$\mathbf{A}^{(2)}$

- Sequence of coarser graphs with adjacency matrices $\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots$

Graph coarsening



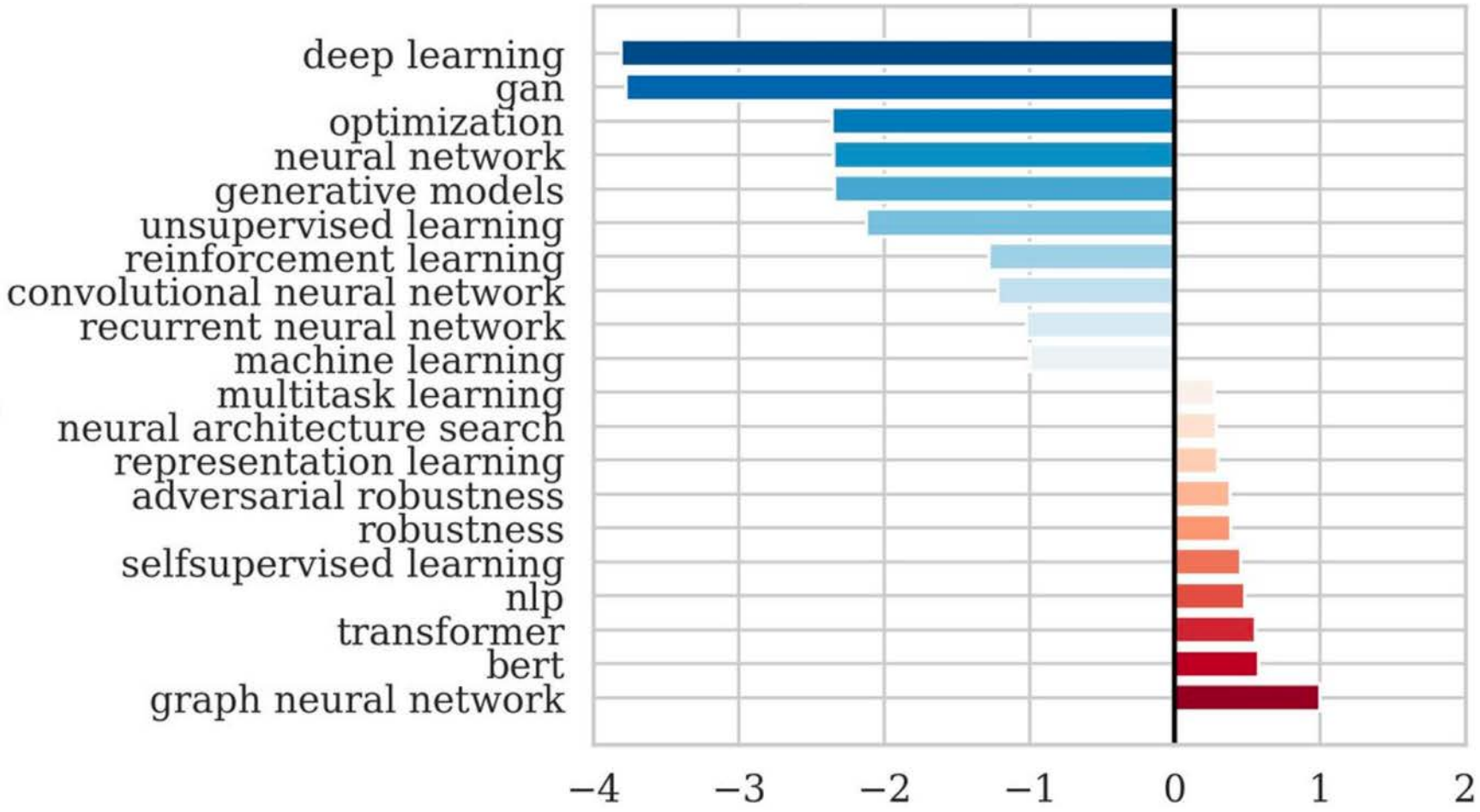
$\mathbf{A}^{(1)} = \mathbf{A}$ $\mathbf{A}^{(2)}$ $\mathbf{A}^{(3)}$

- Sequence of coarser graphs with adjacency matrices $\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots$
- Pooling of features on collapsed vertices $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots$
- Interleave convolutional / pooling layers
- Learnable pooling

Ying et al. 2018 (DiffPool)

“…we might be witnessing a new field being born.”

ICLR 2020 submissions keyword statistics



Δ between 2020 and 2019 in %

Plot: Pau Rodríguez López

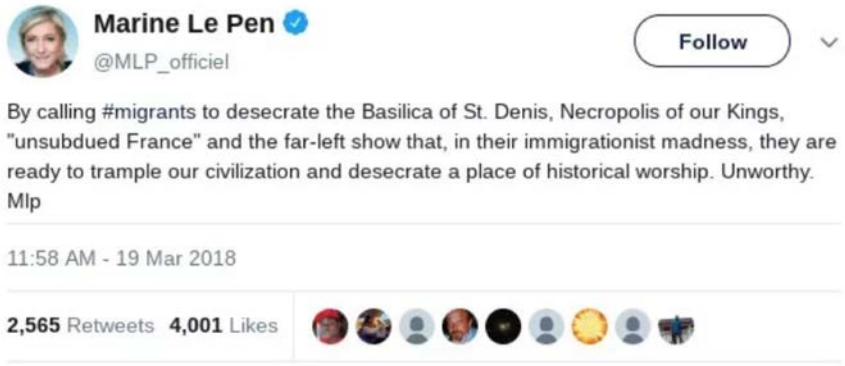
“We expect the following years to bring exciting new methods and applications”

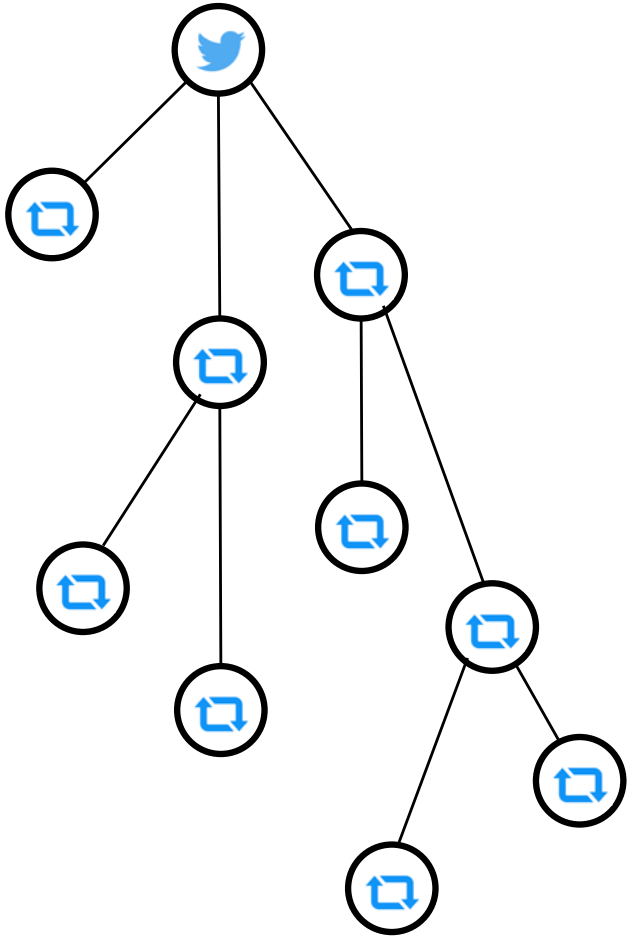
Vosoghi et el. 2018

POLITIFACT

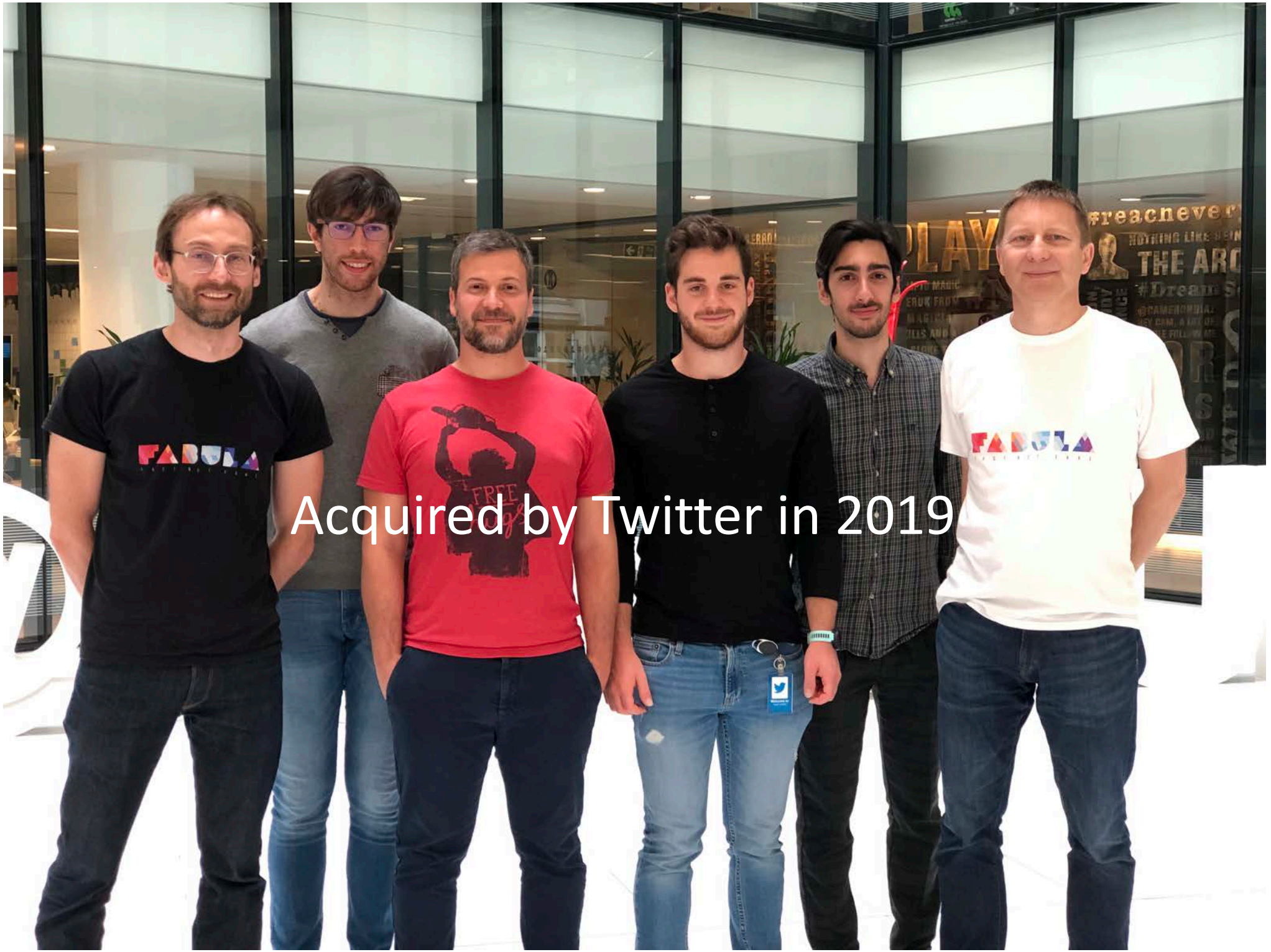
Did 'Muslim Migra[illegible]' Attac[illegible] Catholic Church?

A video of pro-migrant p[illegible] being removed from the Basilica of Saint-Denis in [illegible]nce was shared with the inflammatory and incor[illegible] c[illegible] that it shows Muslim immigrants attacking a[illegible]urch.

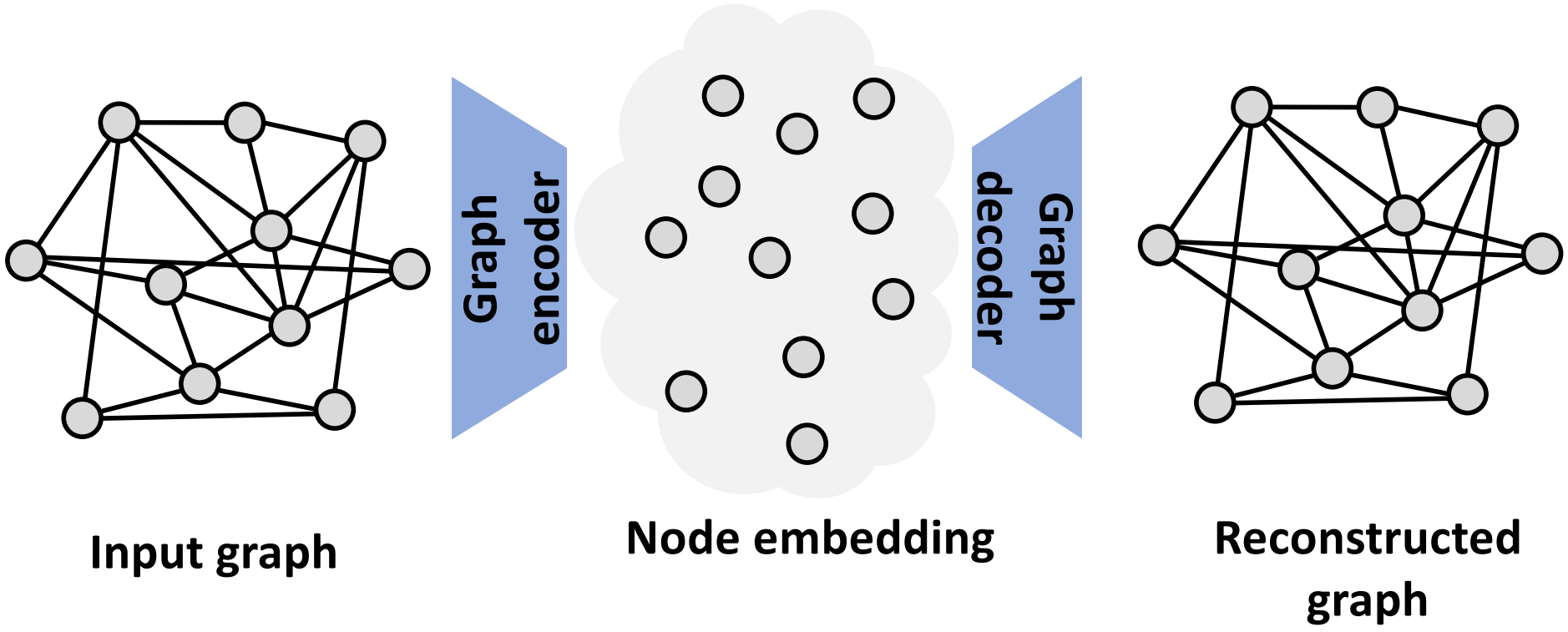




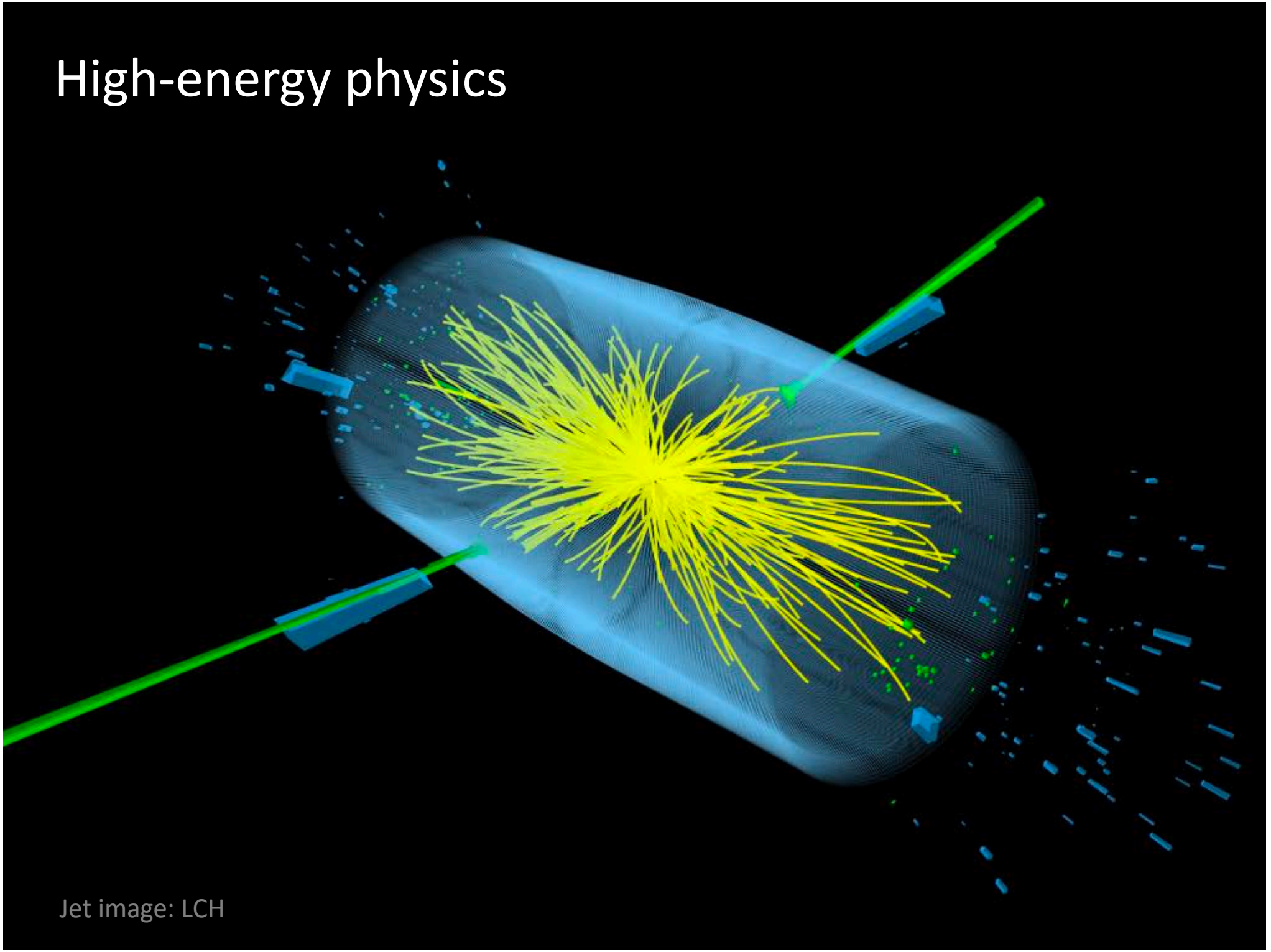
Monti et B 2019


Acquired by Twitter in 2019

Recommender systems and link prediction

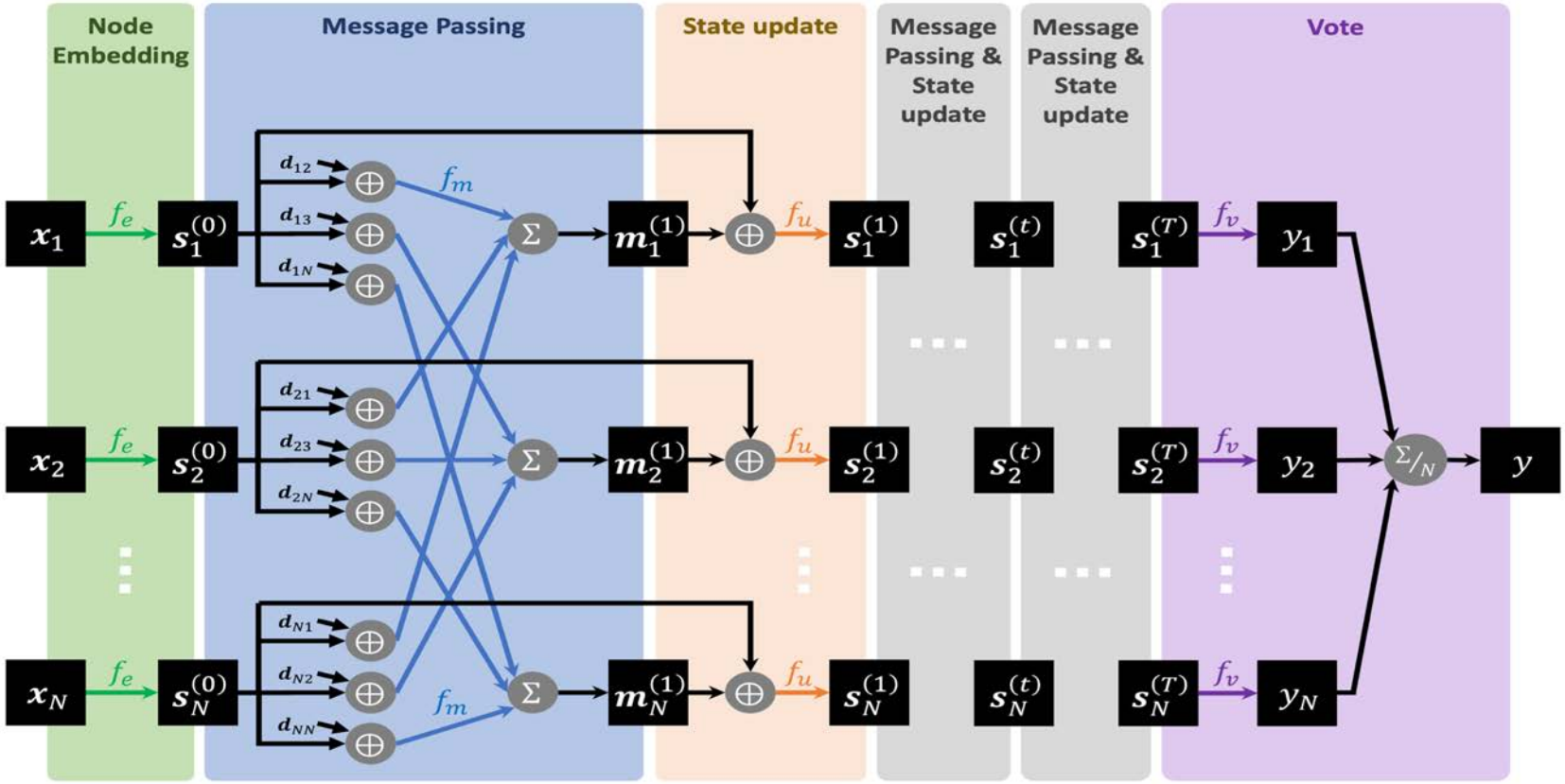


High-energy physics



Jet image: LCH

LHC: stop pair production



GNN architecture for event graph classification

Abdughani et al. 2018

LHC: Particle reconstruction

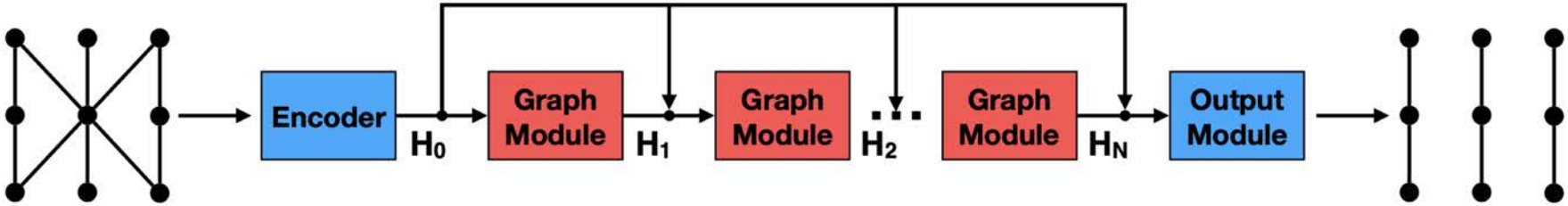


Figure 1: The Graph Neural Network architecture used for tracking.

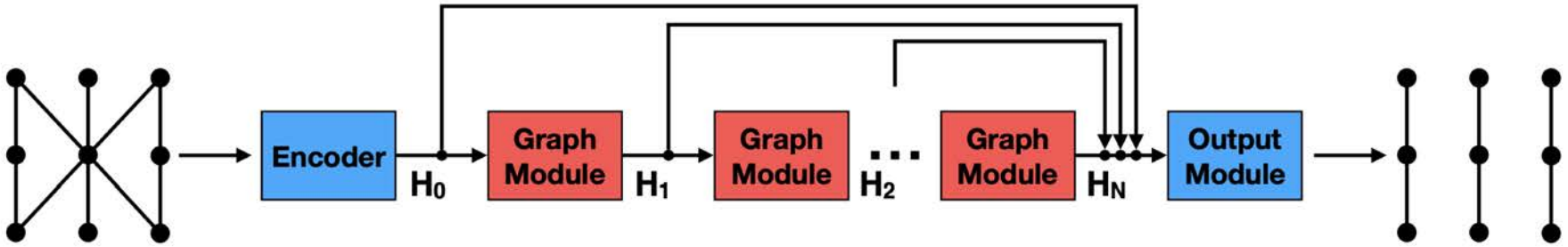
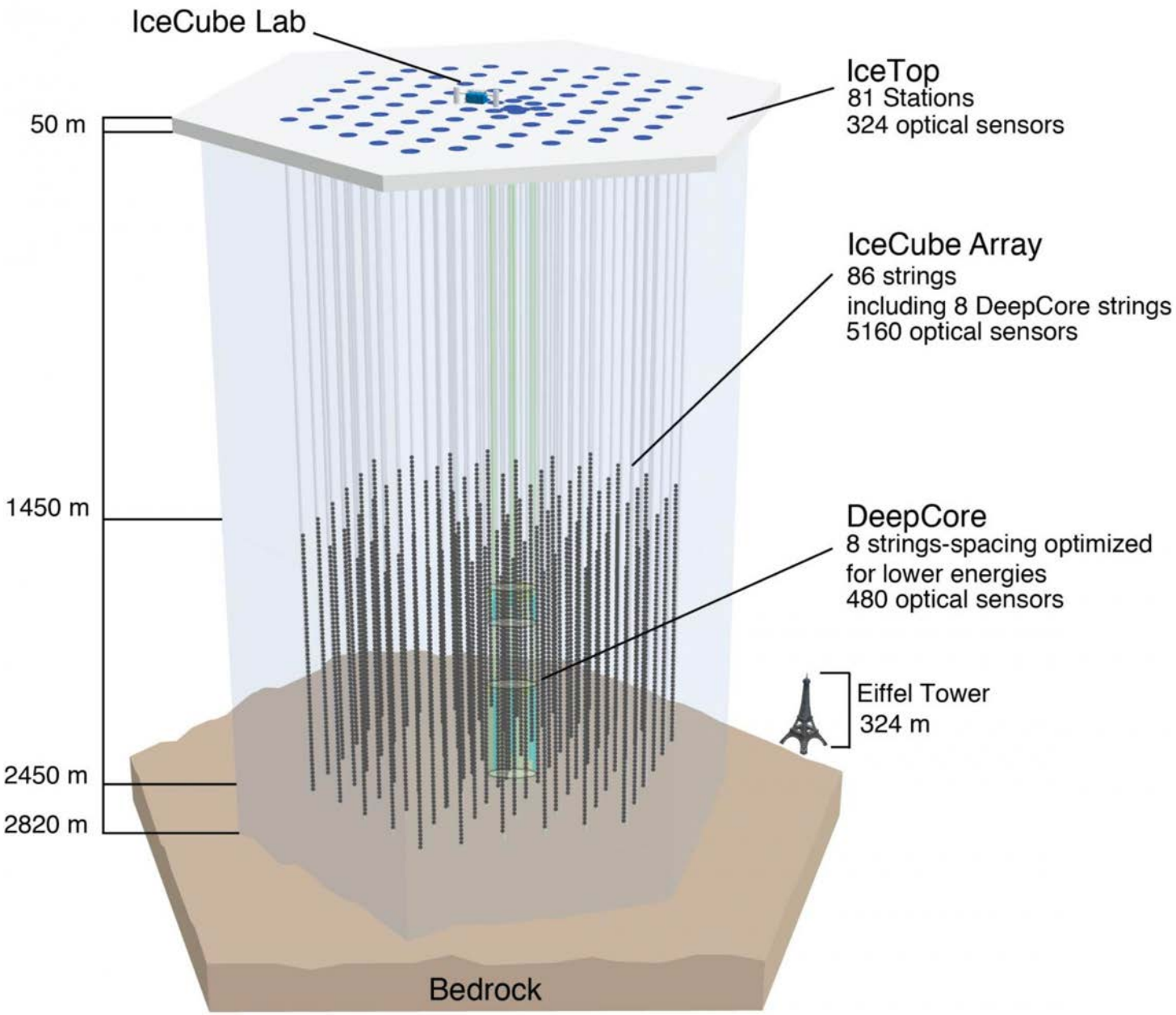


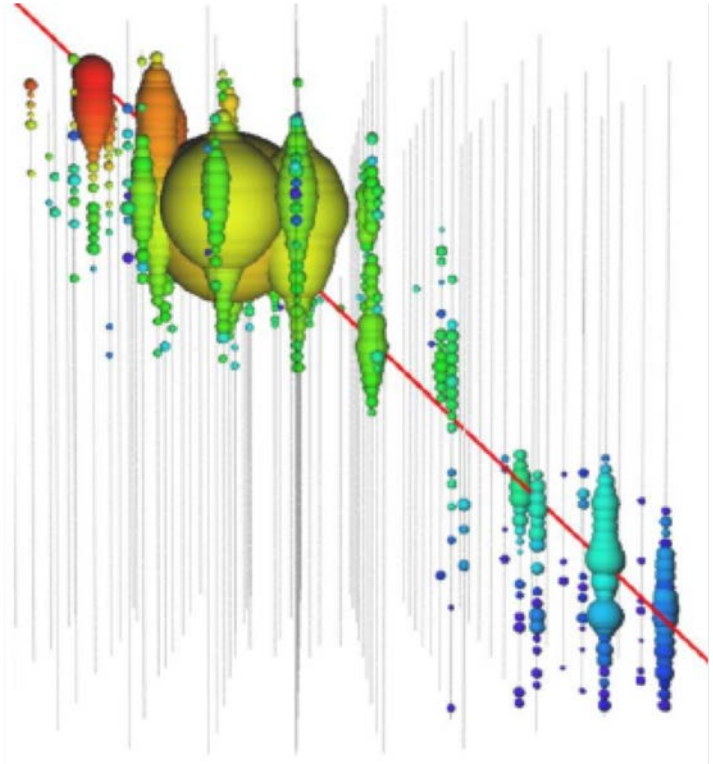
Figure 2: The Graph Neural Network architecture used for calorimeter clustering.

Ju et al. 2019

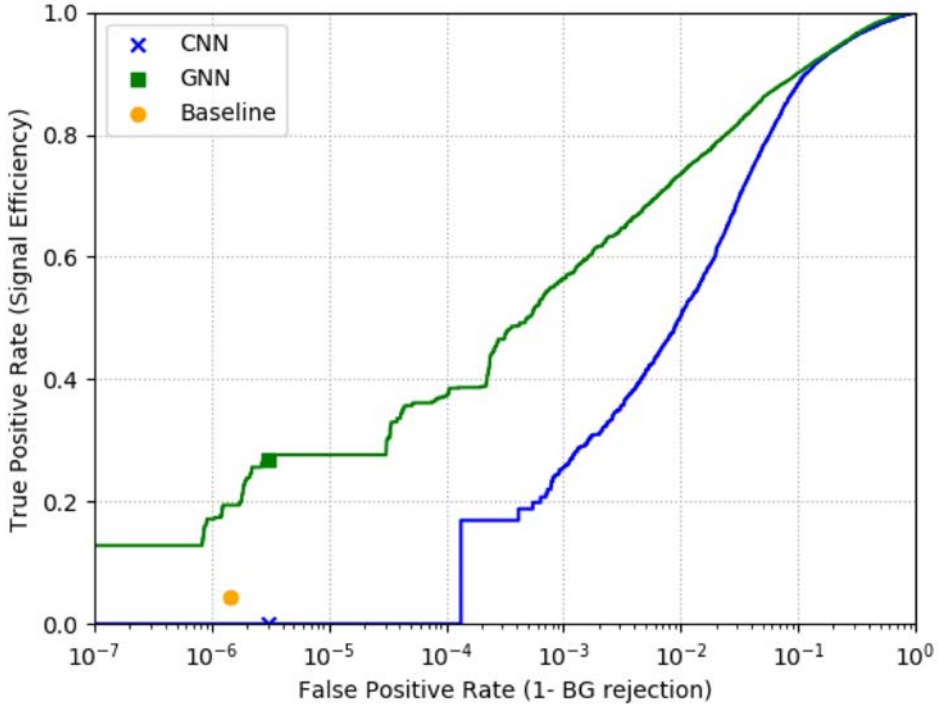
IceCube: neutrino detection



IceCube: neutrino detection



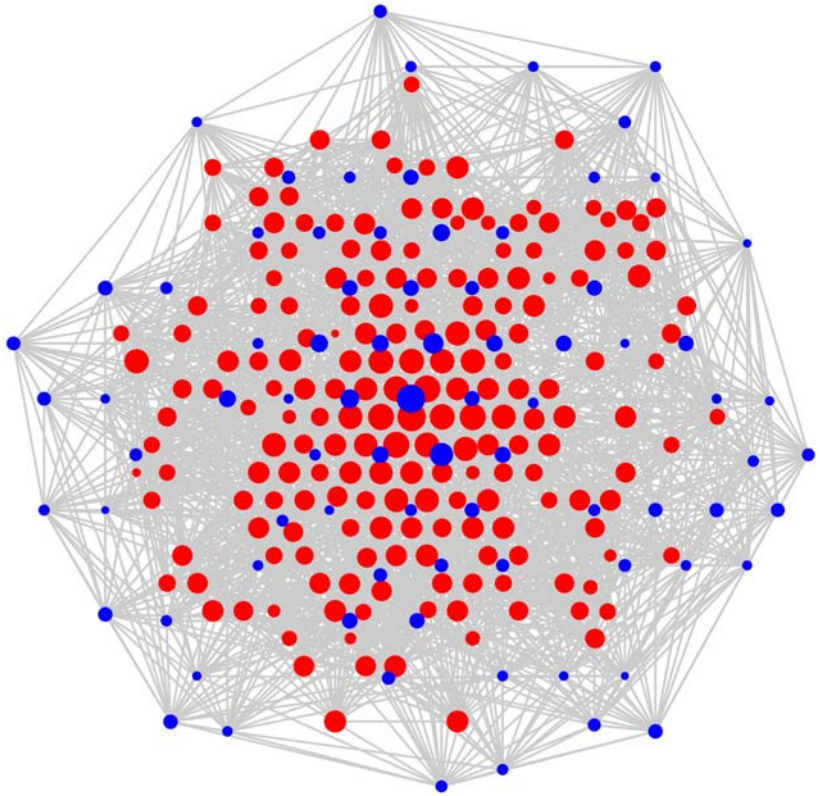
Light deposition for a high-energy single muon in IceCube detector



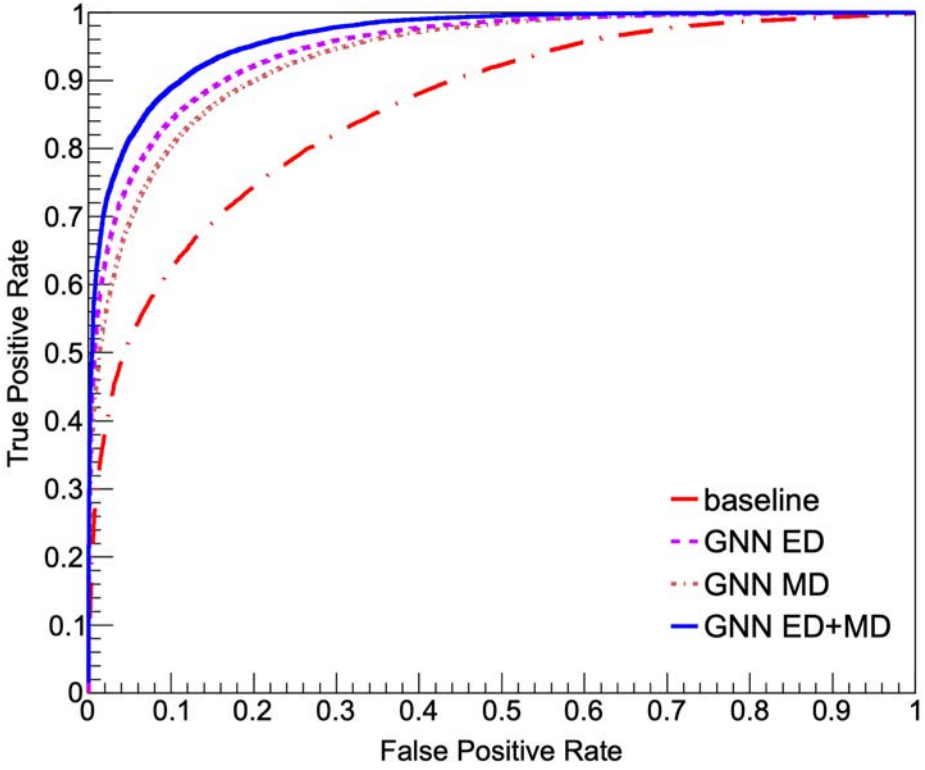
ROC curve comparing different methods for neutrino detection

Choma et B 2018

LHAASO CR experiment



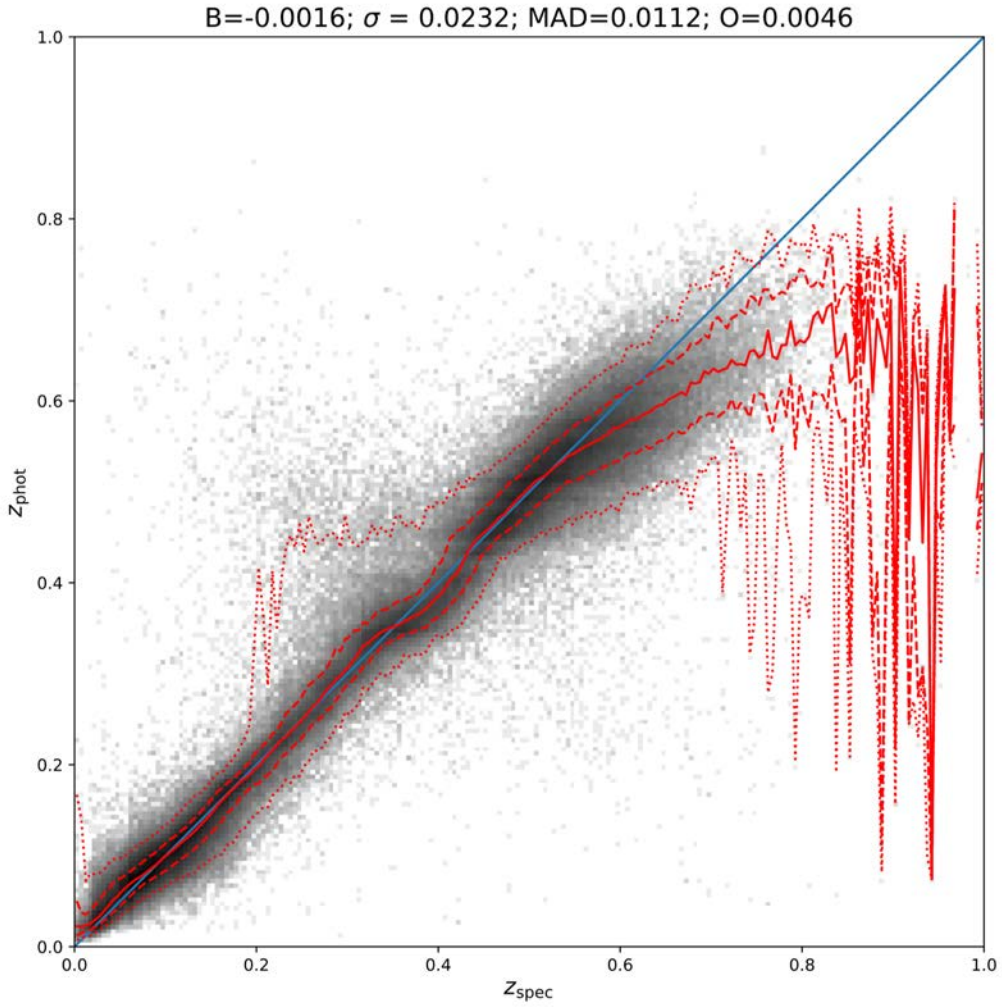
Graph-structured LHAASO-KM2A detectors activated by a 500-TeV EAS event (red=EM & blue=muon detectors)



ROC of classifying light component from the background

Jin, Chen, He 2019

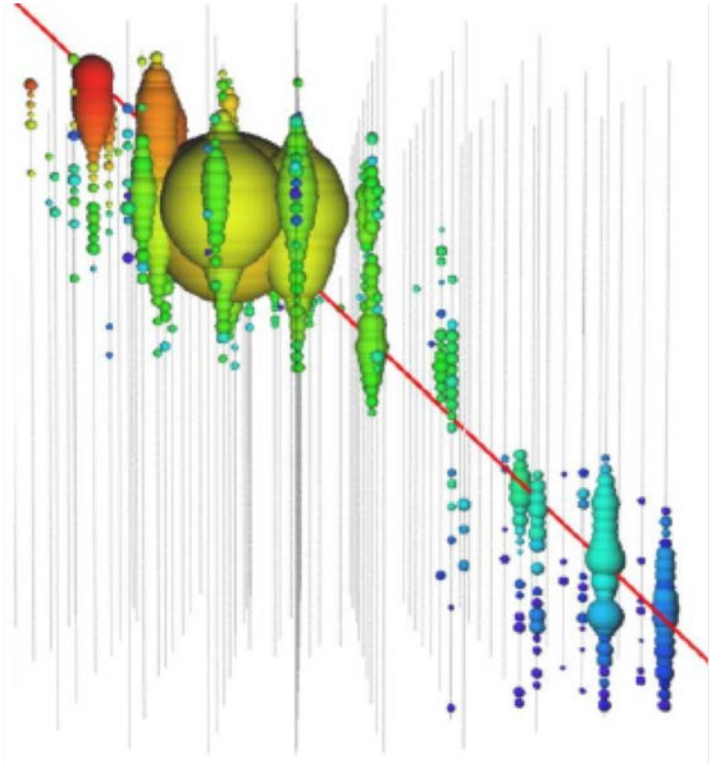
Astrophysics: Redshift regression



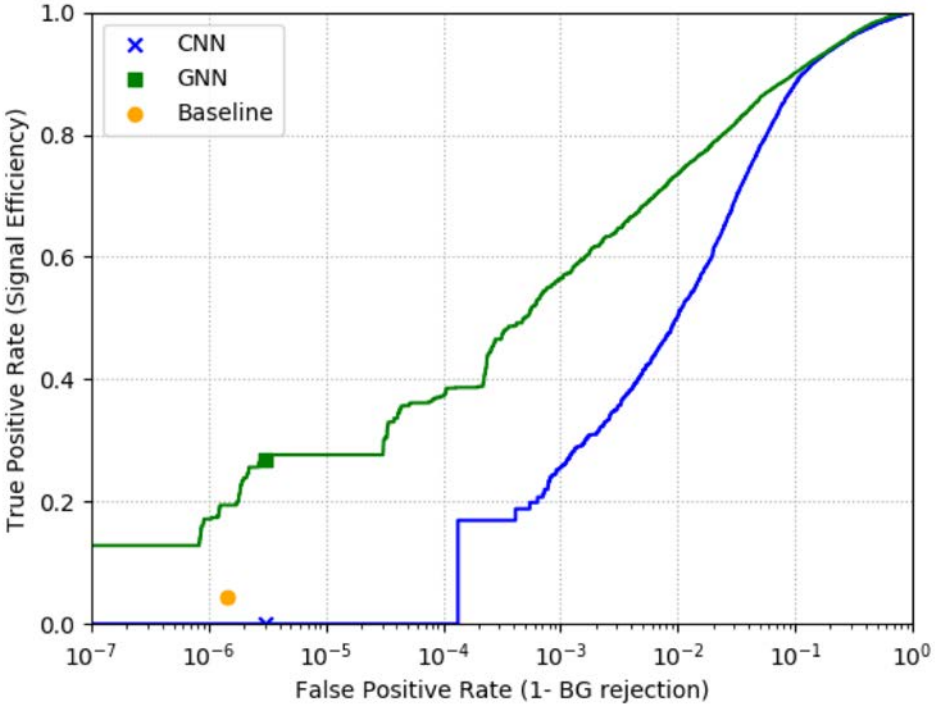
Predicted galaxy redshift from photometric observations using MoNet-style GNN vs groundtruth spectroscopic measurement

Beck et al 2019

Neutrino detection



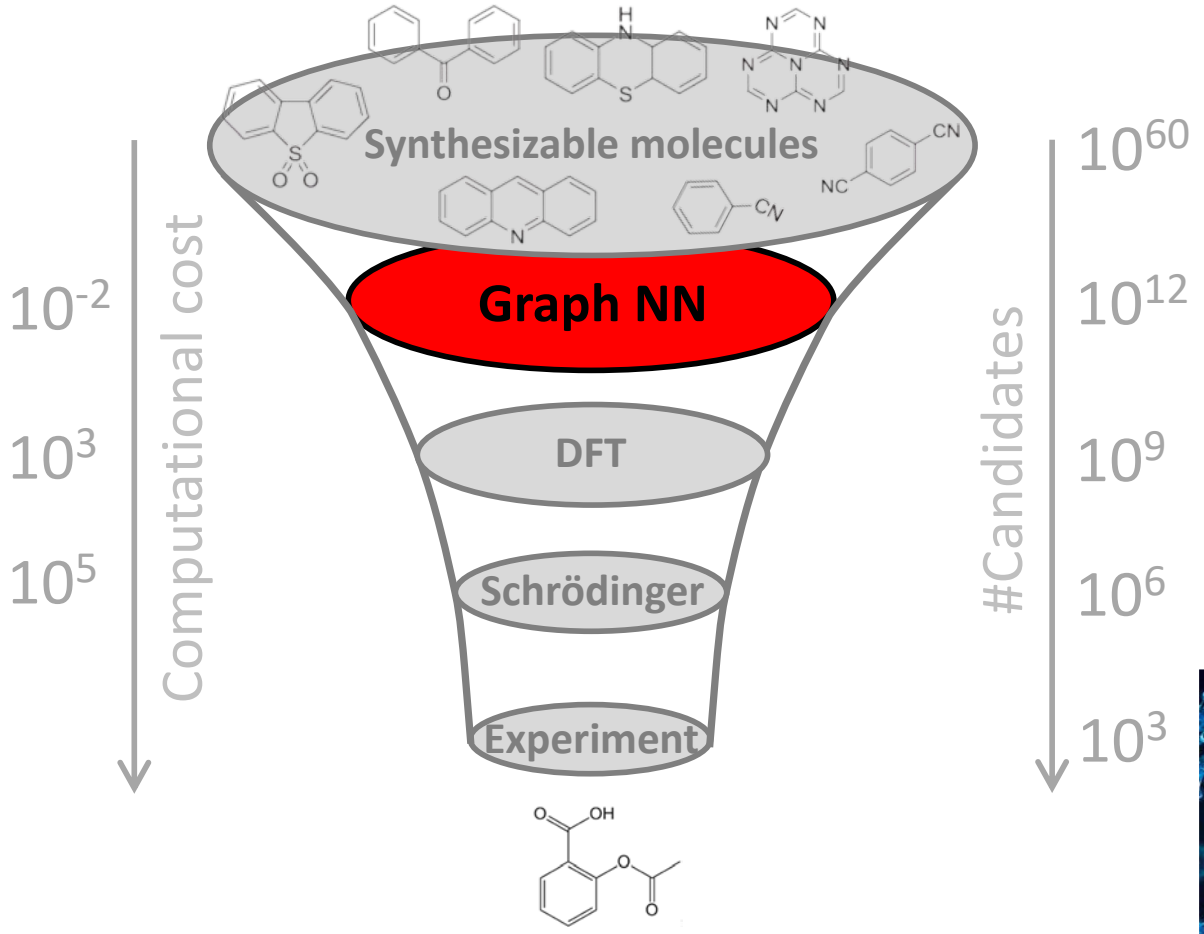
Light deposition for a high-energy single muon in IceCube detector



ROC curve comparing different methods for neutrino detection

Choma et B 2018

Computational chemistry and drug design



"Computational funnel"

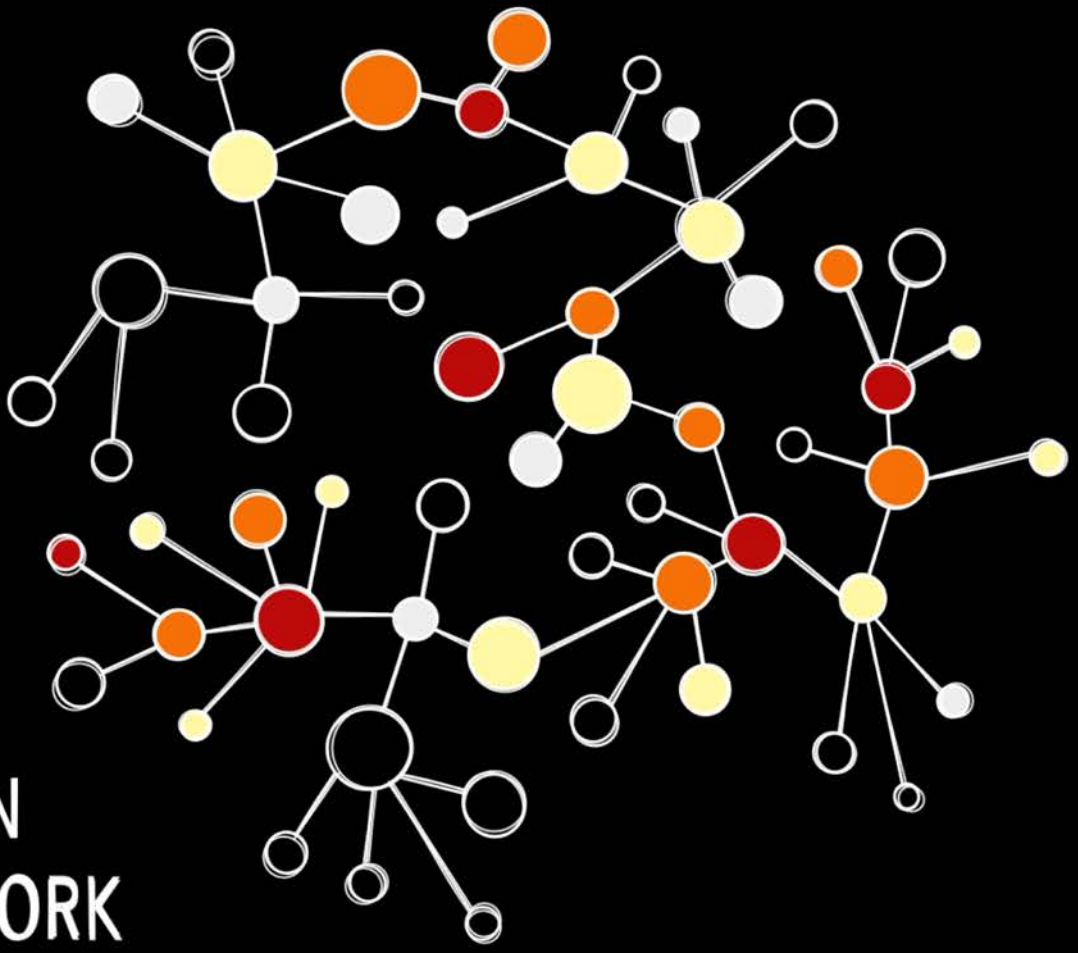
Duvenaud et al. 2015; Gilmer et al. 2017; Jin et al. 2020

Stokes et al. 2020

Hyperfoods

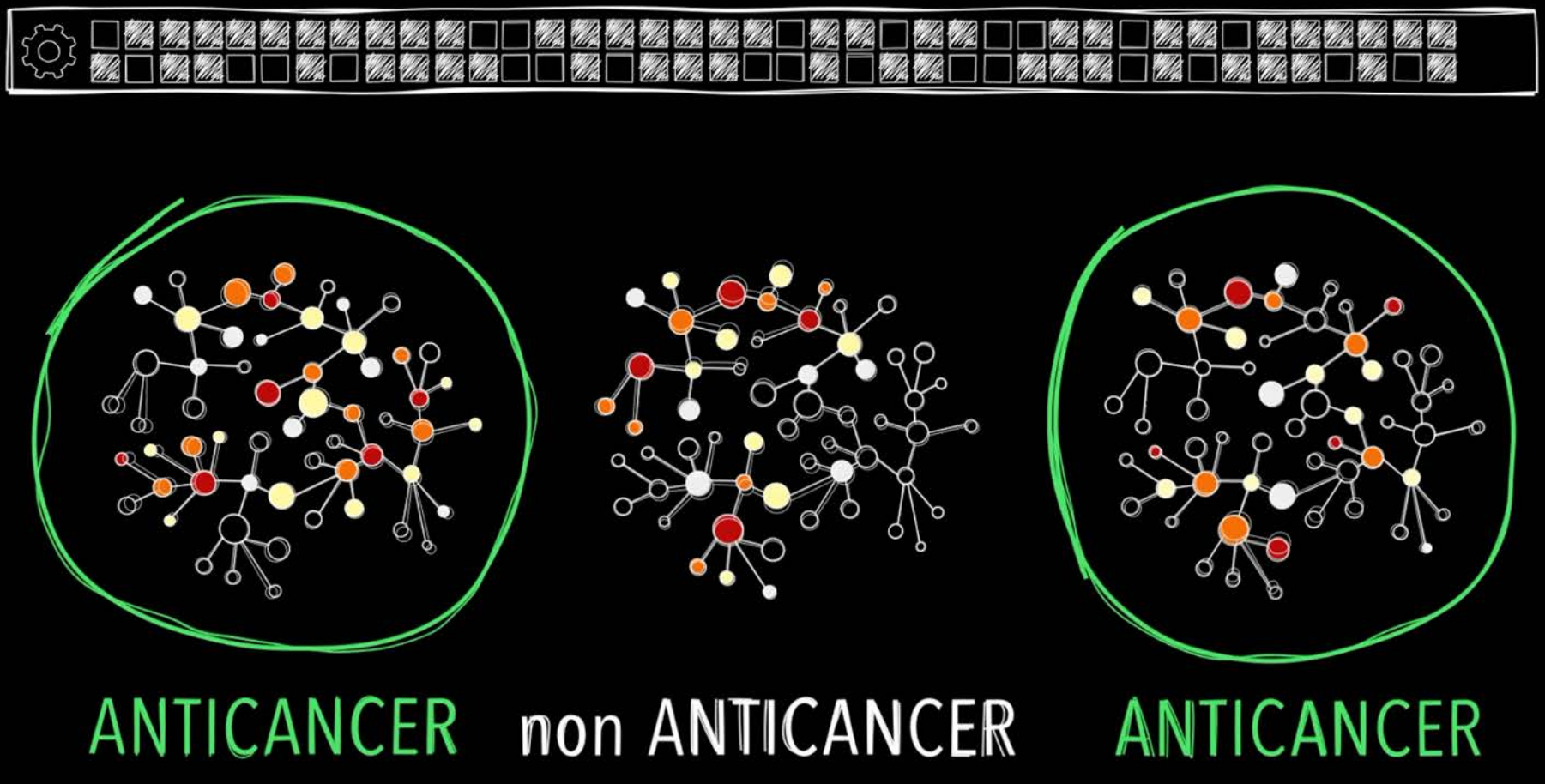
Veselkov et B 2019

Hyperfoods



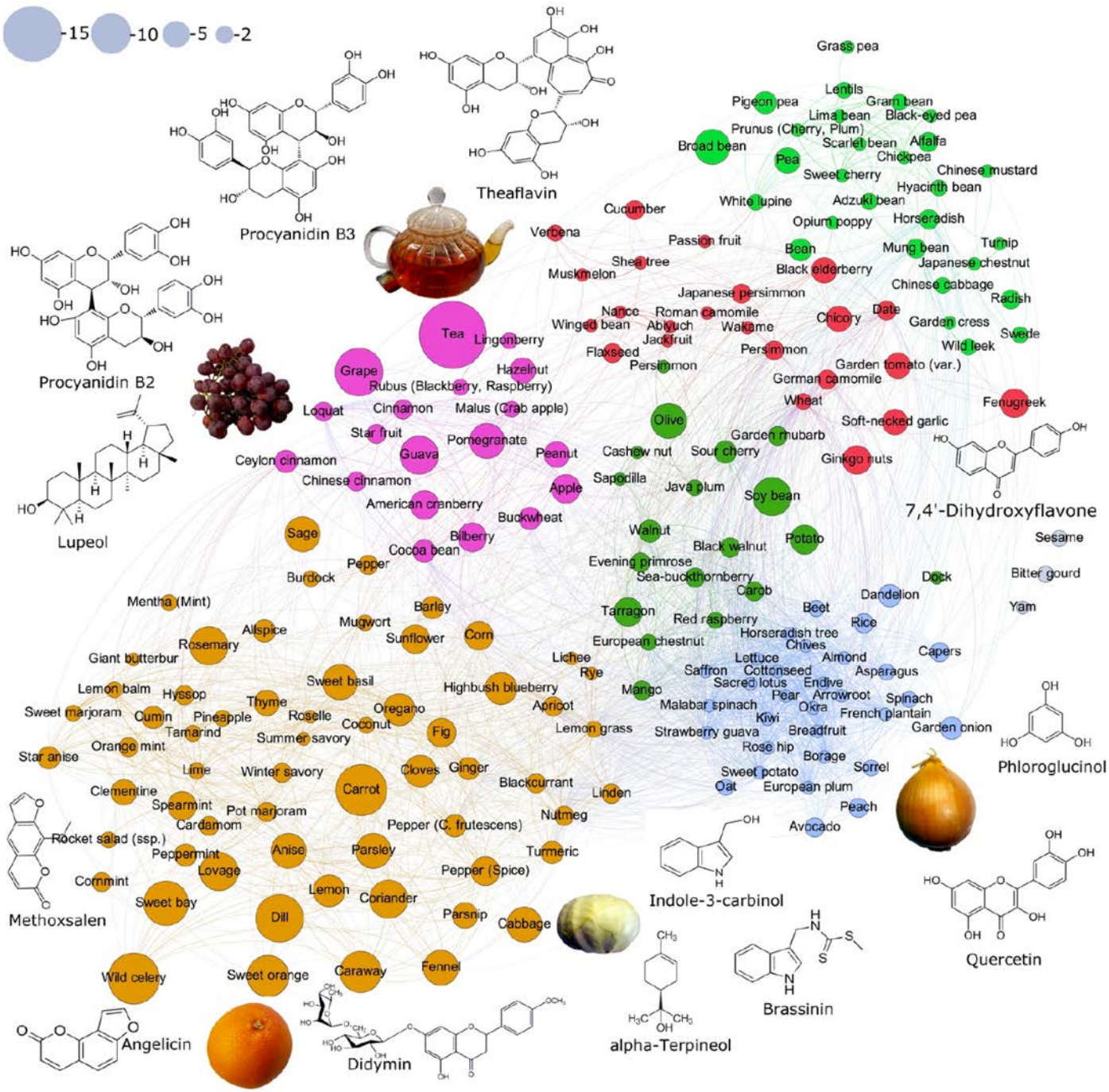
Veselkov et B 2019

Hyperfoods



Veselkov et B 2019

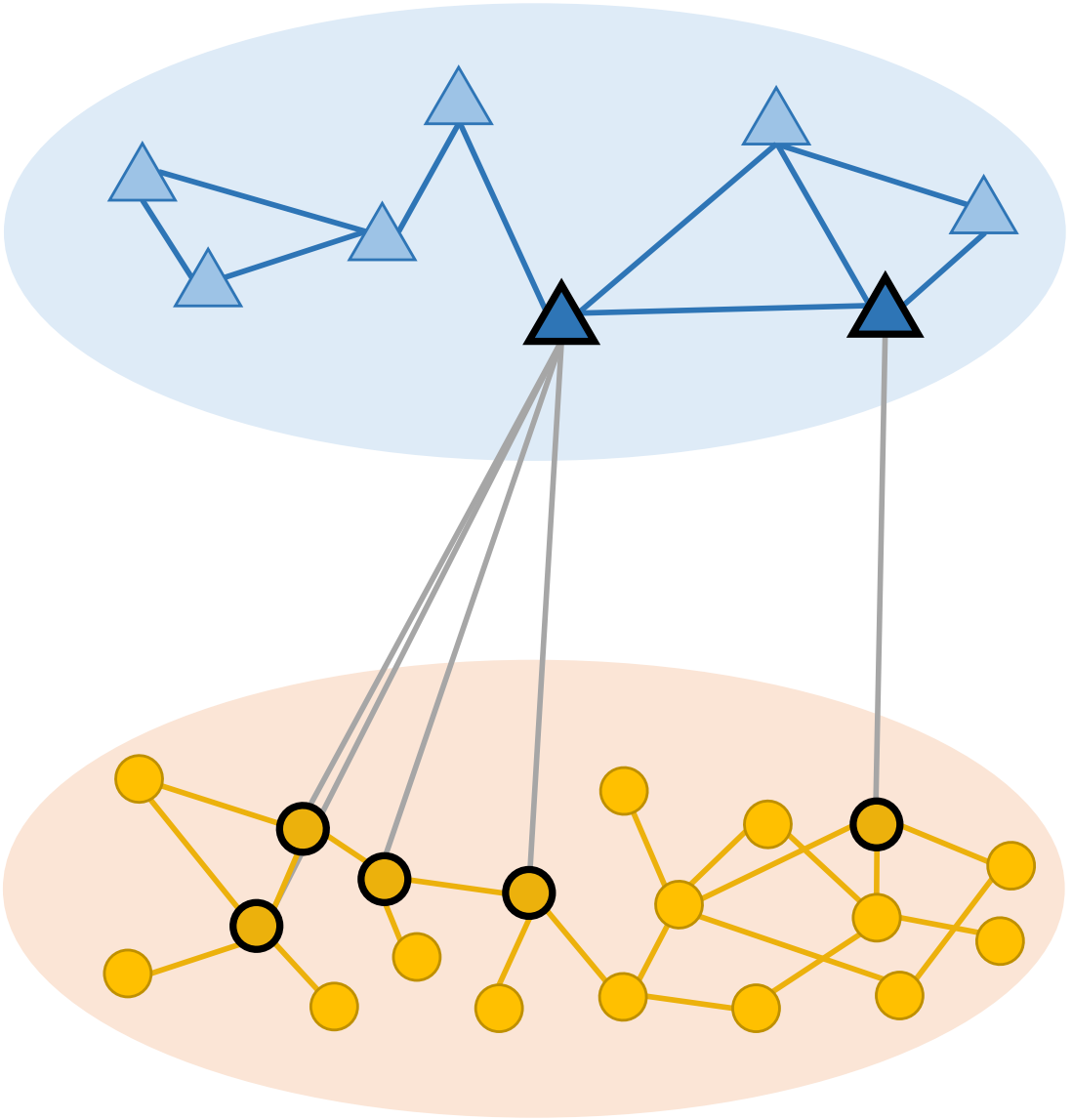
Hyperfoods



Veselkov et B 2019


DreamLab
KT.

Combinatorial drug therapy



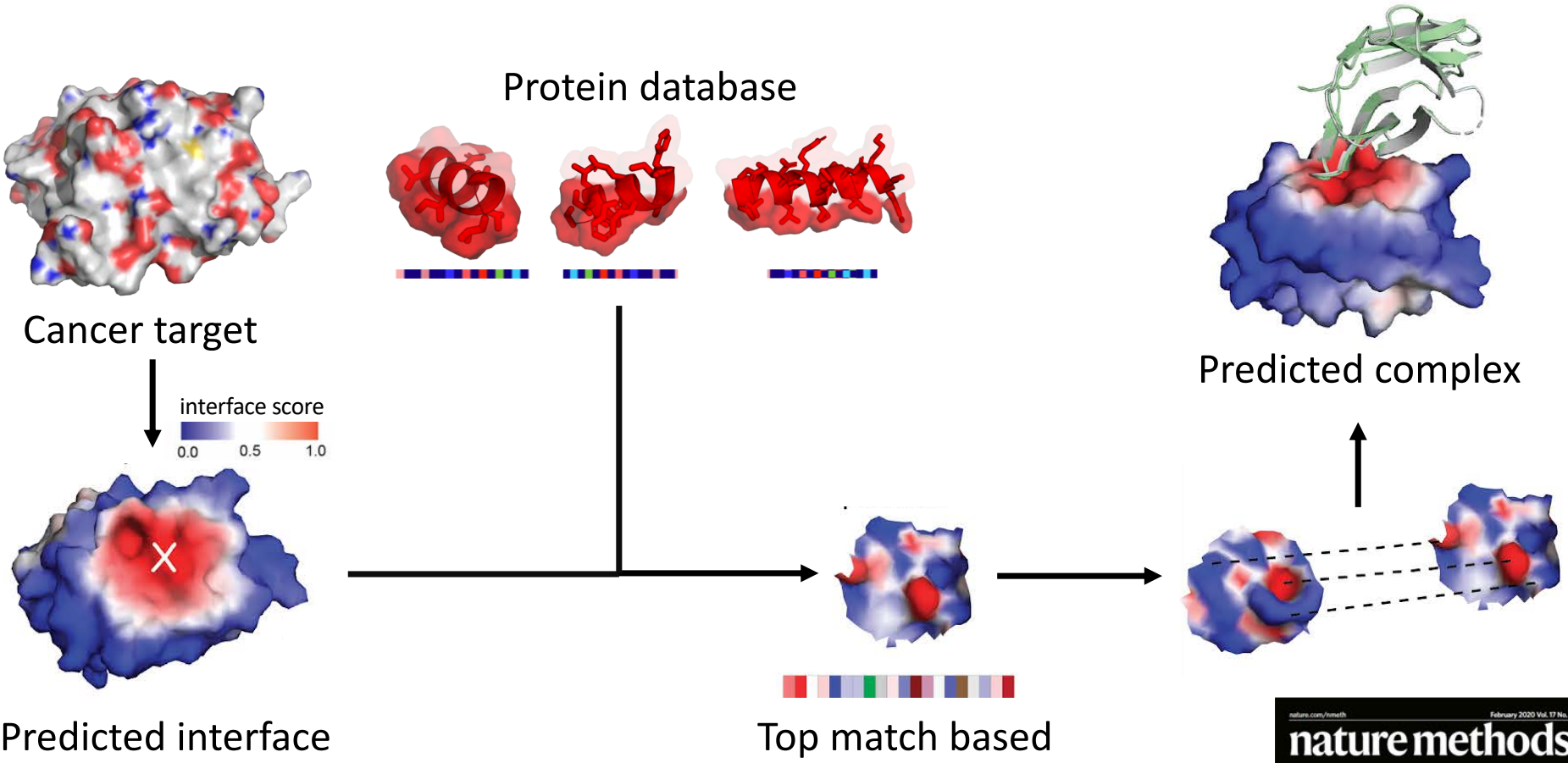
Zitnik et al 2018

Protein science and cancer immunotherapy

PD-1 **PD-L1**

The Nobel Prize in Physiology or Medicine 2018 was awarded jointly to James P. Allison and Tasuku Honjo "for their discovery of cancer therapy by inhibition of negative immune regulation."

De novo protein design

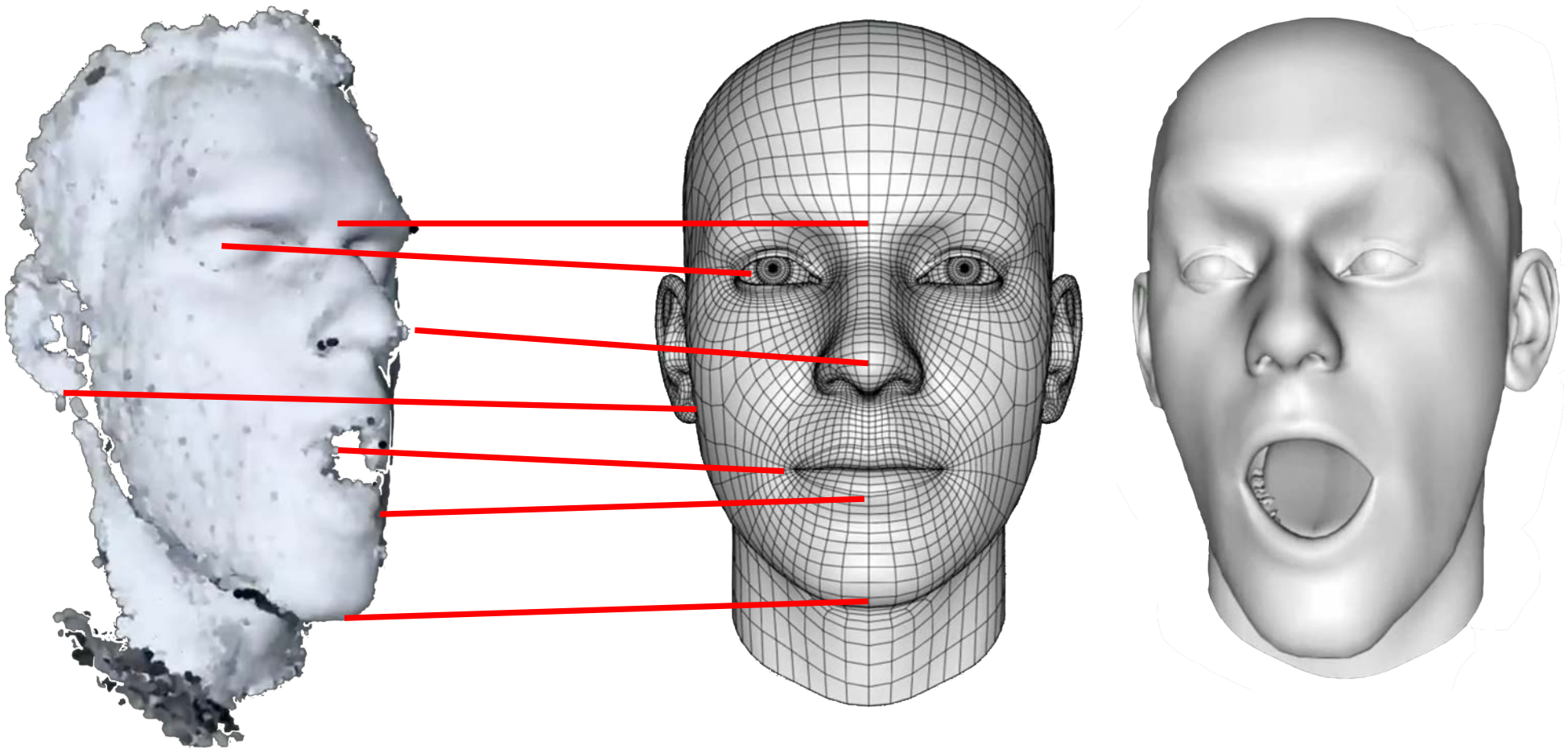


Gainza et B 2020

3D vision and graphics

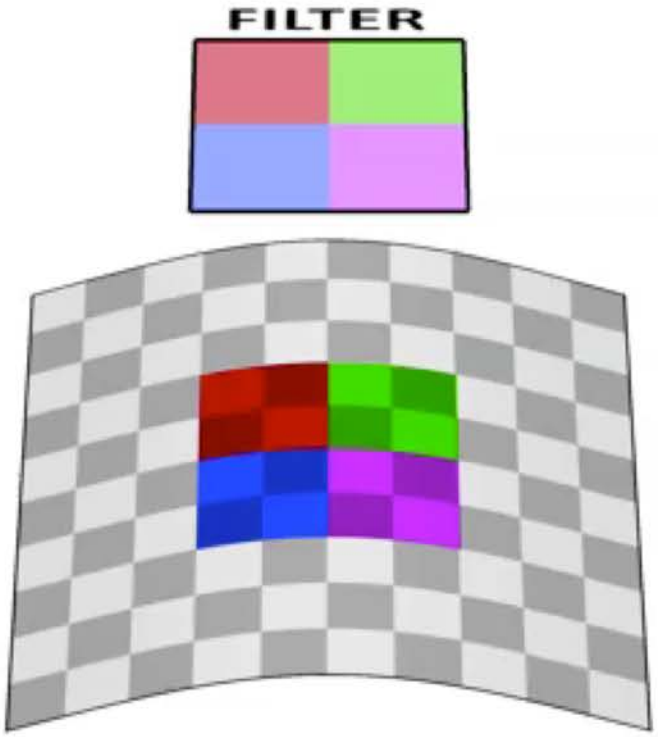
Video: FaceShift 2015

Shape analysis and synthesis

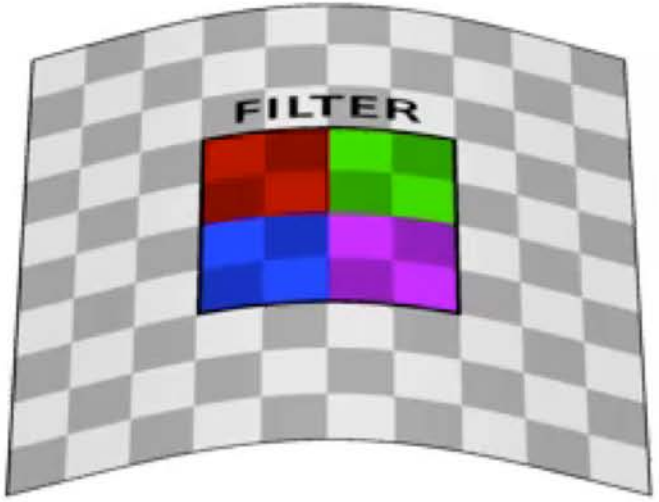


Analysis **Synthesis**

Images: Faceshift



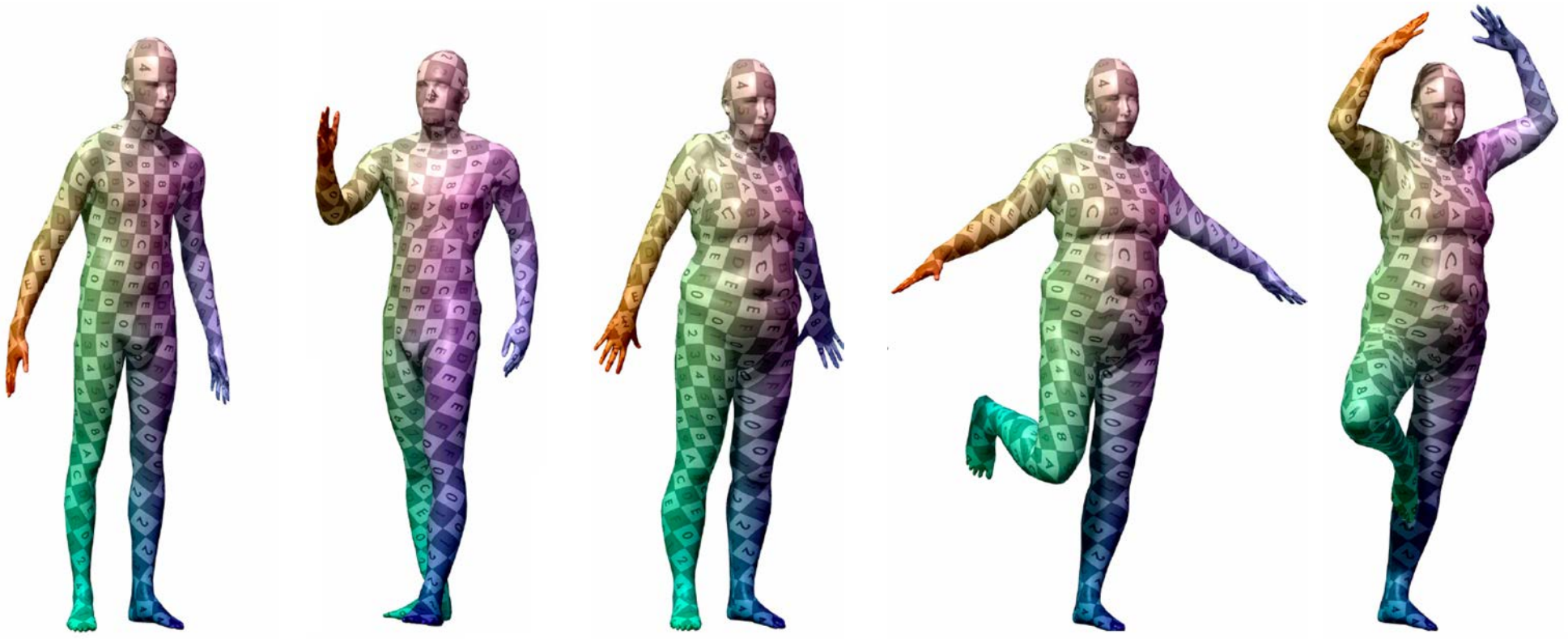
Classical (extrinsic) convolution



Geometric (intrinsic) convolution

Masci et B 2015

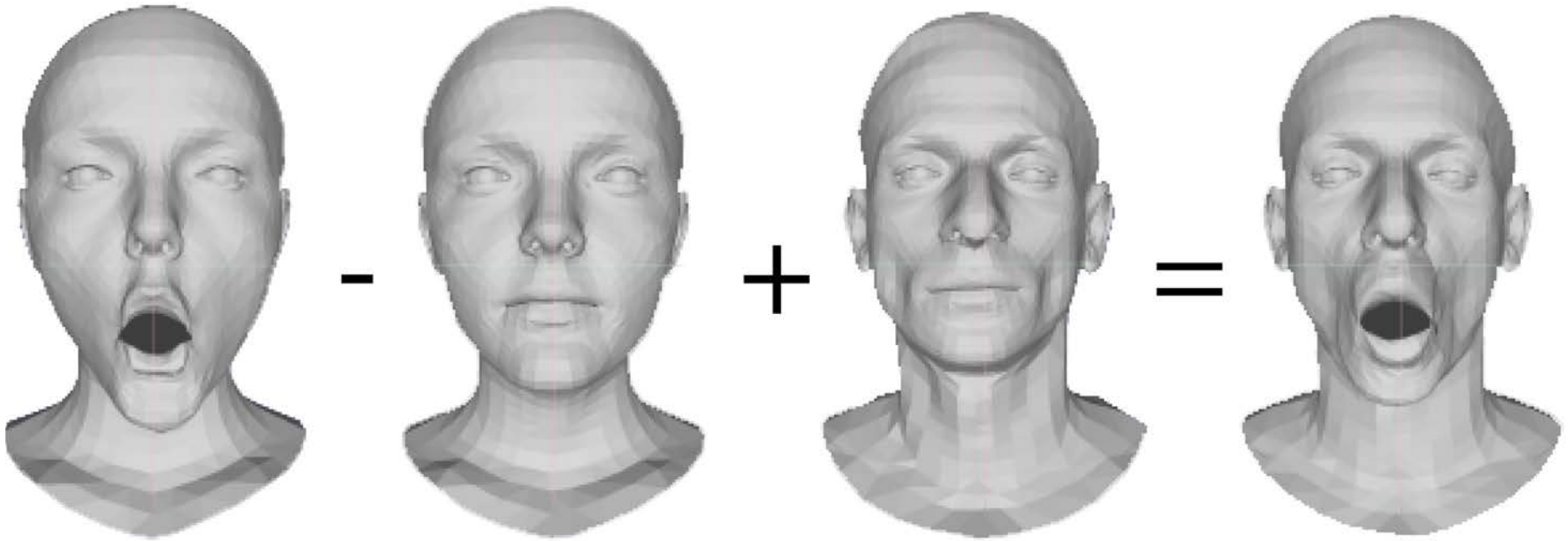
Deformable shape correspondence



Correspondence between 3D shapes

Masci et B 2015; Monti et B 2017; Litany et B 2017; Verma et al. 2017

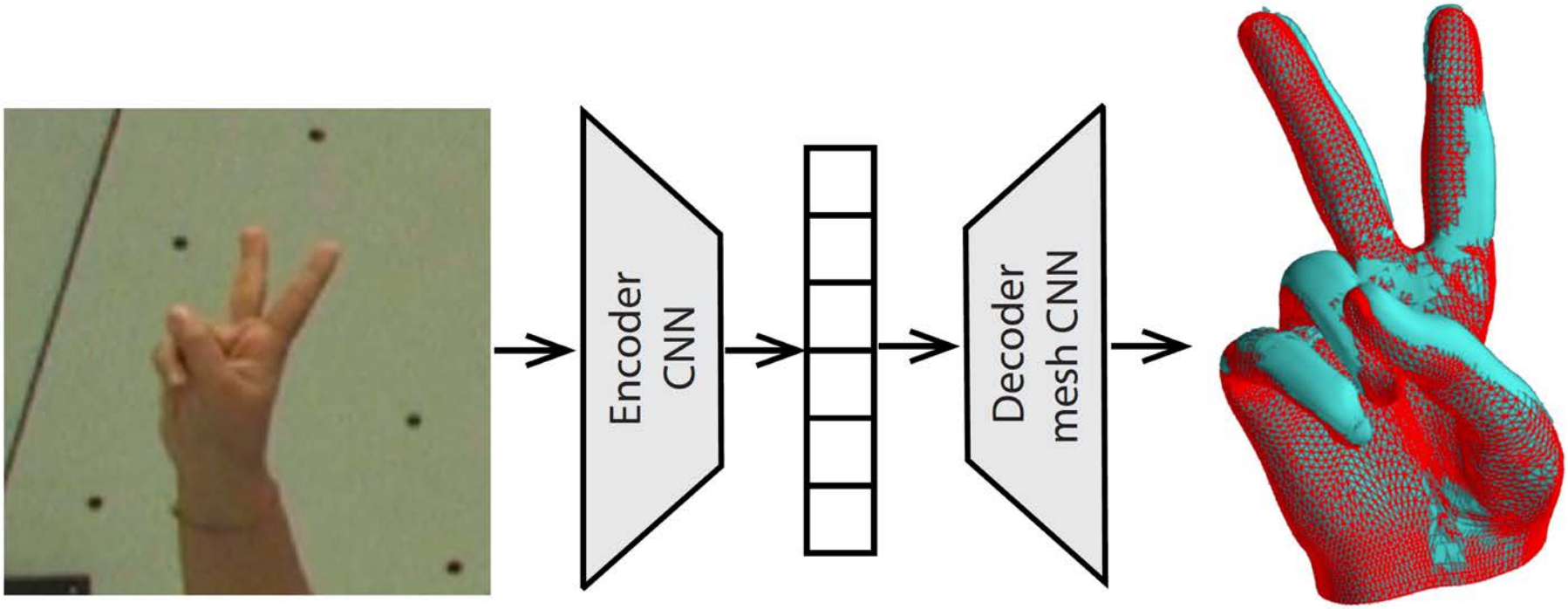
3D generative models



3D shape calculus in latent space

Litany et B 2018; Ranjan et al. 2018; Bouritsas et B 2019; Gong et B 2019; Gong, Basri et B 2020

3D hand reconstruction from 2D images



Mixed 2D/3D encoder/decoder model

Kulon et B 2019; Kulon et B 2020

3D hand reconstruction from 2D images

Examples of reconstructed 3D hands in the wild

Kulon et B 2019; Kulon et B 2020


Video: Ariel AI 2020

Conclusions

- Graphs are very general abstractions, useful everywhere
- Deep learning models capable of accounting for graph structures ("relational inductive bias")
- Cross-disciplinary research
- Very hot field
- Several success stories
- Some first industrial applications

Open problems

- Theoretical guarantees
- Standardized benchmarks
- Efficient computation and scaling to web-size graphs
- Higher order structures (e.g. motifs)
- Dynamic graphs
- Graph generating models
- Incorporating problem-specific knowledge


P. Lio'
E. Rossi
L. Cosmo
D. Eynard
F. Frasca
D. Mannion
F. Monti
J. Svoboda
D. Boscaini
E. Rodolà
J. Masci
V. Kim
UNIVERSITY OF CAMBRIDGE
FABULA
USI
SAPIENZA UNIVERSITÀ DI ROMA
nnaisense
Adobe


D. Kulon
G. Bouritsas
G. Gonzalez
S. Gong
S. Zafeiriou
K. Veselkov
L. Guibas
O. Litany
P. Claes
K. Otness
X. Bresson
D. Cremers
Imperial College London
Stanford University
KU LEUVEN
HARVARD


J. Solomon
Y. Wang
Y. Wang
Y. Sun
P. Vandergheynst
B. Correia
P. Gainza
R. Levie
A. Makadia
A. Bronstein
J. Bruna
N. Choma
MIT
EPFL
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
TEL AVIV UNIVERSITY
Technion
NYU


Fondation Dalle Molle
Pour la qualité de la vie
FNSNF
erc
Google
NVIDIA
amazon
THE ROYAL SOCIETY