GEOMETRIC DEEP LEARNING

Joseph Bullock

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

Geometric Deep Learning

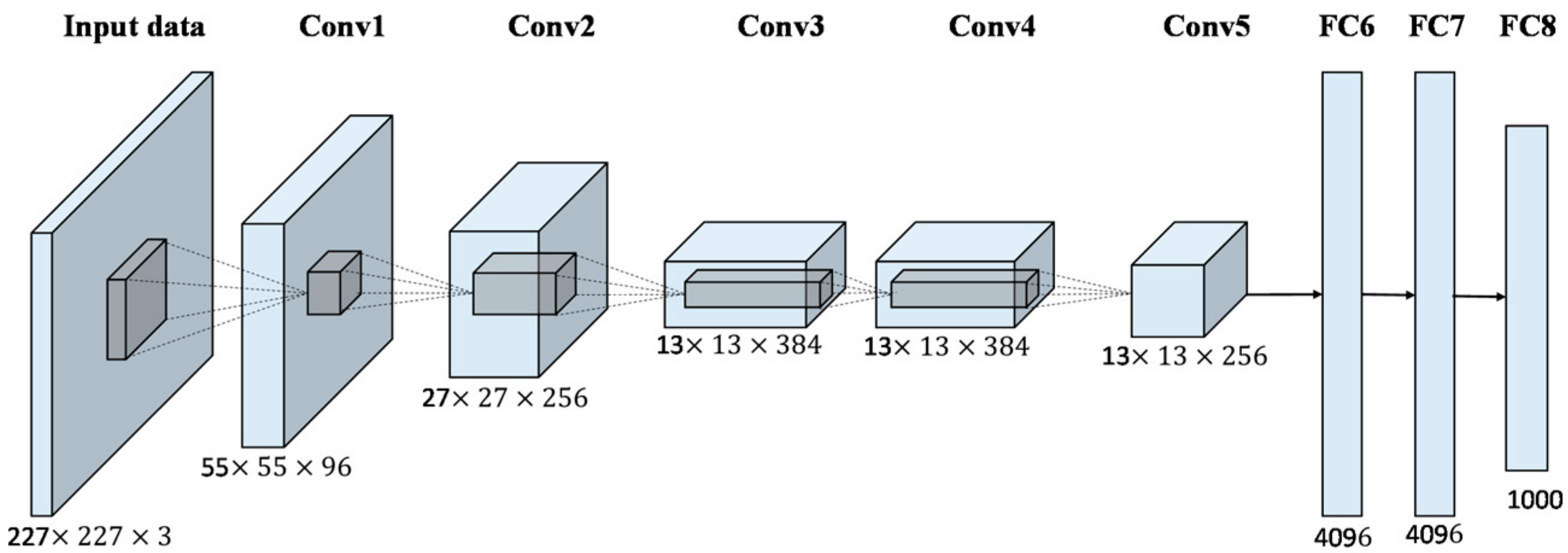
Going beyond Euclidean data

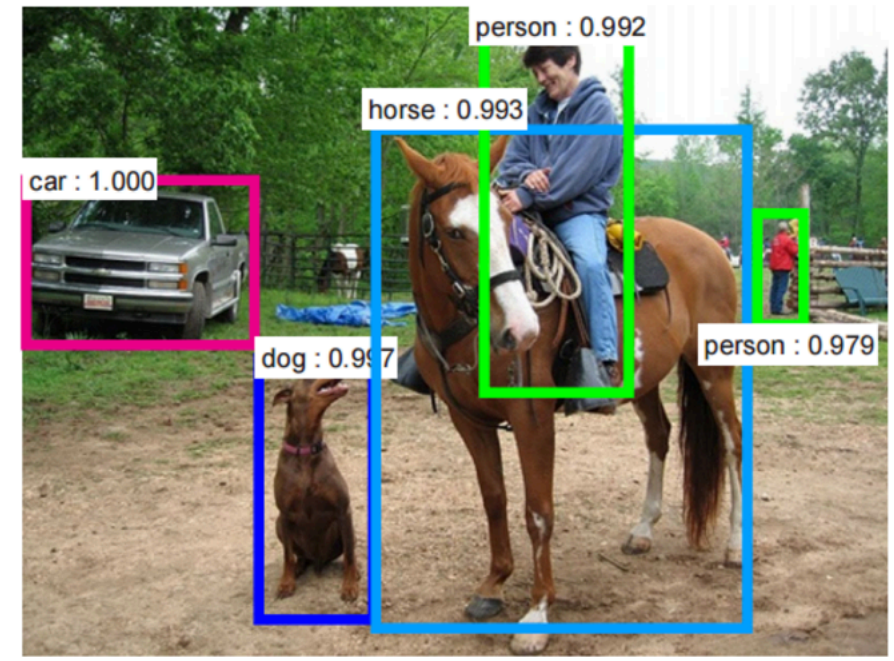
Many scientific fields study data with an underlying structure that is non-Euclidean. Some examples include social networks in computational social sciences, sensor networks in communications, functional networks in brain imaging, regulatory networks in genetics, and meshed surfaces in computer graphics. 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 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]

Overview of Convolutional Neural Networks (CNNs)

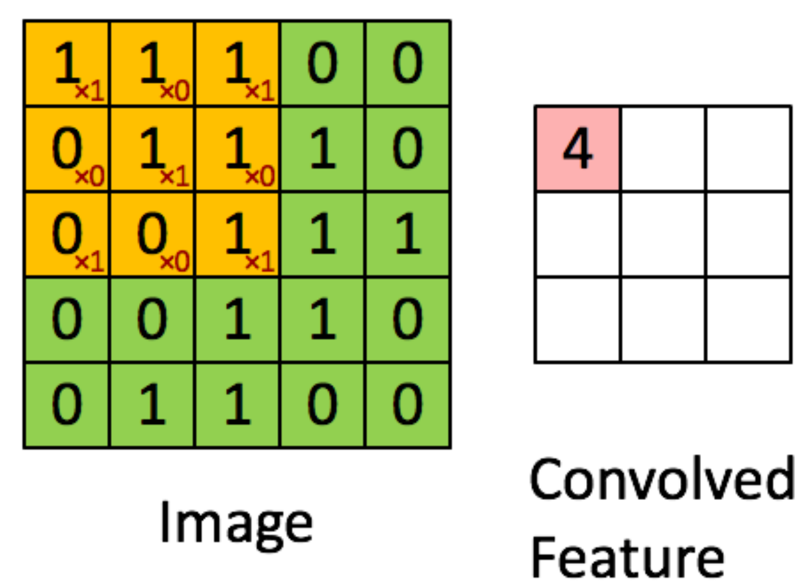




Good at extracting information from complex images

- **Locality**
- **Compositionality and hierarchy**
- **Filters have compact support**

Can think of data (images) as functions on **Euclidean plane**, sampled on a grid



Top left image: Han et al. '17; Top right image: Data Science Blog; Bottom image: Reppel '17; Properties: LeCun et al. '89, Mallat '12 [arXiv:11012286]

Overview of Convolutional Neural Networks (CNNs)

Convolution

$$\vec{g} = C_\Gamma(\vec{f}\,), \quad \vec{f} = (f_1(x), \ldots, f_p(x)) \qquad l' = 1,\ldots,p$$

$$\Gamma = (\gamma_{l,l'}) \qquad l = 1,\ldots,q$$

Activation function

$$g_l(x) = \xi\left(\sum_{l'=1}^{p}\left(f_{l'} \star \gamma_{l,l'}\right)(x)\right), \quad \left(f \star \gamma\right)(x) = \int_\Omega f(x - x')\gamma(x')dx'$$

$$\vec{g}(x) = (g_1(x), \ldots, g_q(x))$$

Euclidean domain

Pooling

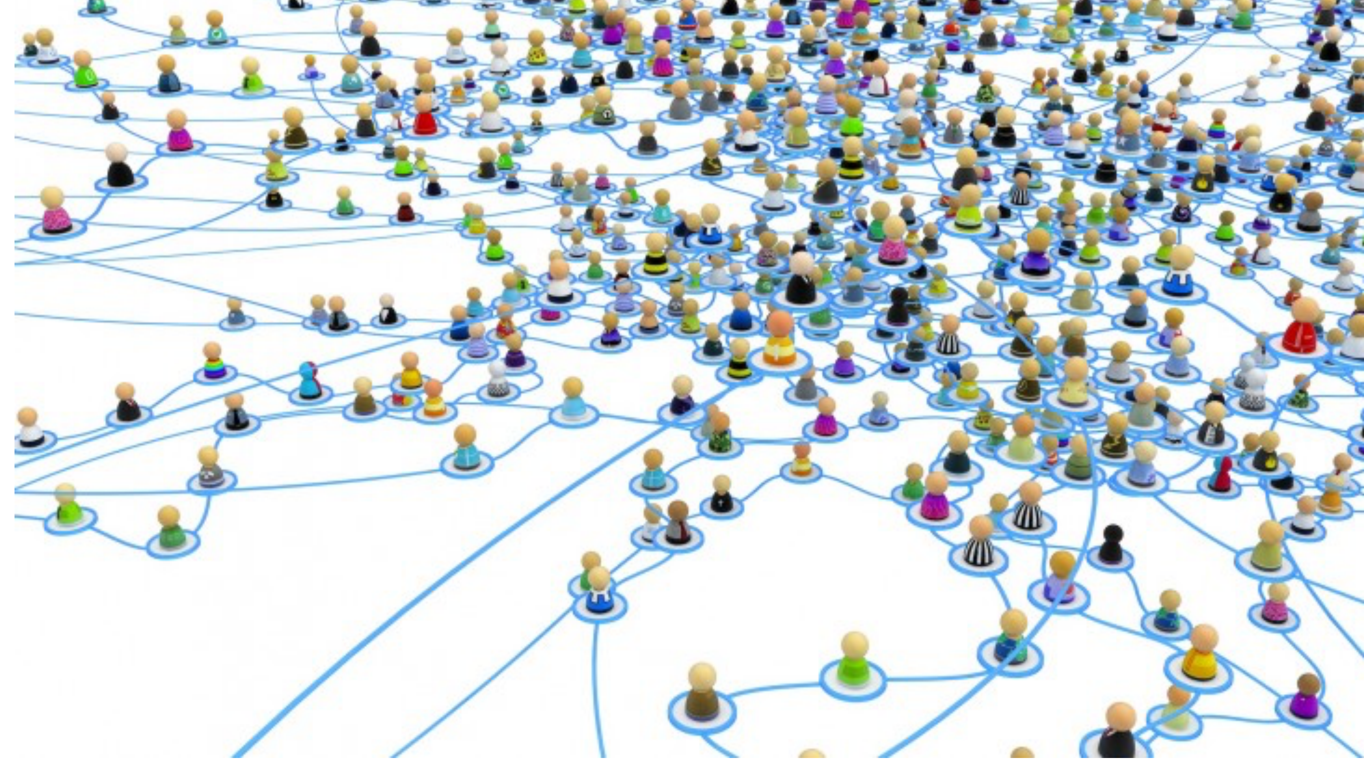
$$\vec{g} = P(\vec{f}\,)$$

$$g_l(x) = P\left(\left\{f_l(x') : \; x' \in \mathcal{N}(x)\right\}\right)$$

CNN

$$U_\Theta(f) = \left(C_{\Gamma^{(K)}} \ldots P \ldots \circ C_{\Gamma^{(2)}} \circ C_{\Gamma^{(1)}}\right)$$

Going Non-Euclidean

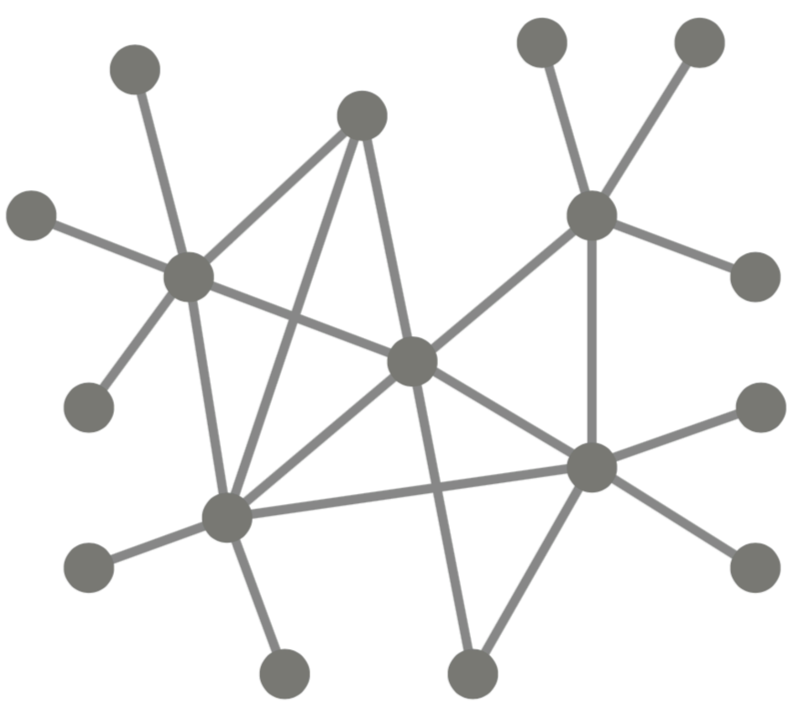


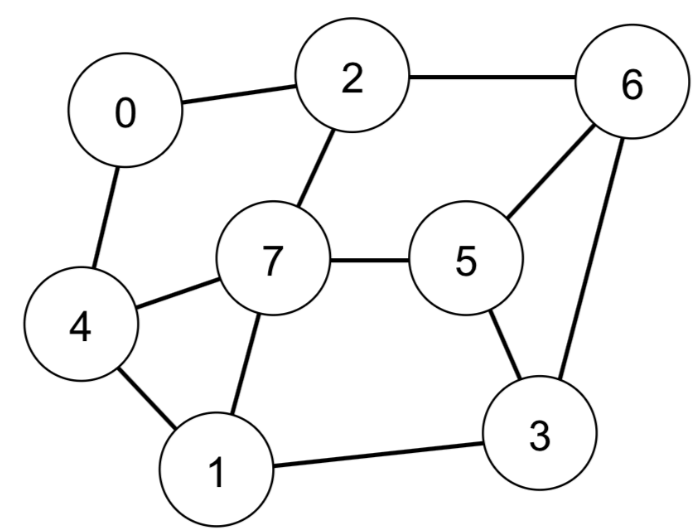
Many examples:
Social networks, meshed surfaces, sensor networks, word embeddings

e.g. LHC detector array

Problem: Convolution and pooling not well define on non-regular grids

Solution: Fourier transform on spectral (graph) domain



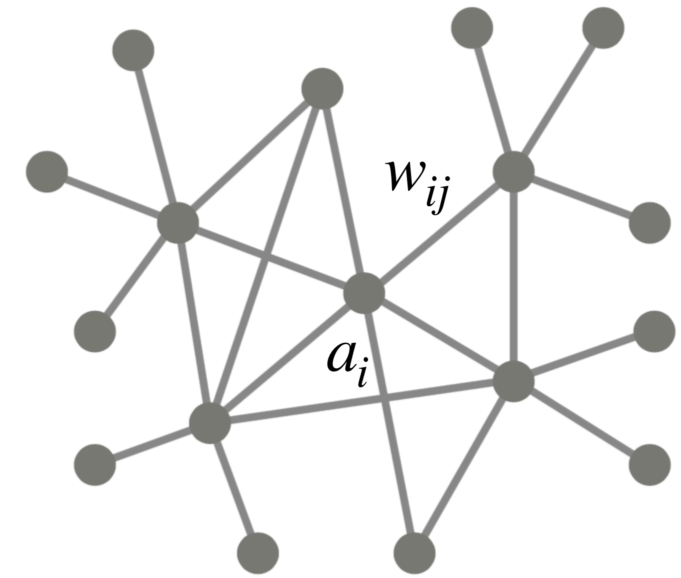


Top image: Saragossi ' 13

What is a graph?

Properties

Consider *weighted undirected* graphs



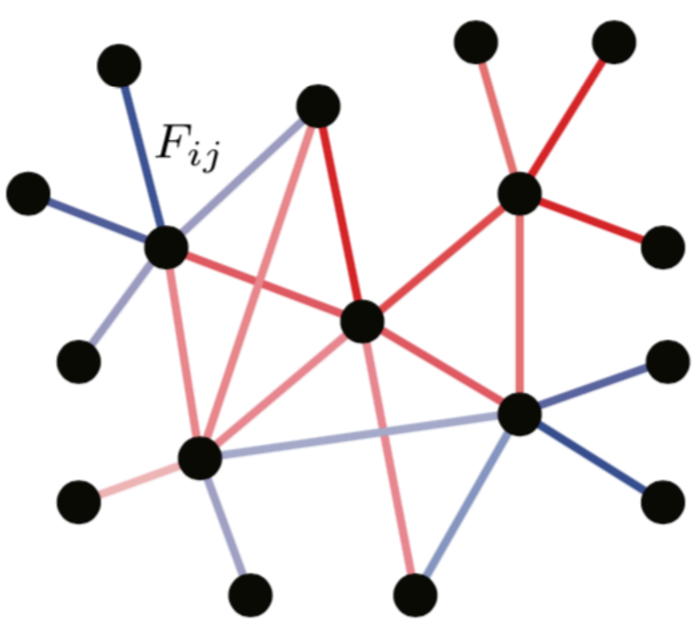
$(\mathscr{V}, \mathscr{E})$, $\quad \mathscr{V} = \{1, \ldots, n\}$ set of vertices

$\mathscr{E} \subseteq \mathscr{V} \times \mathscr{V}$ set of edges

$a_i > 0,\ i \in \mathscr{V}$ vertex weights

$w_{ij} \geq 0,\ (i,j) \in \mathscr{E}$ edge weights

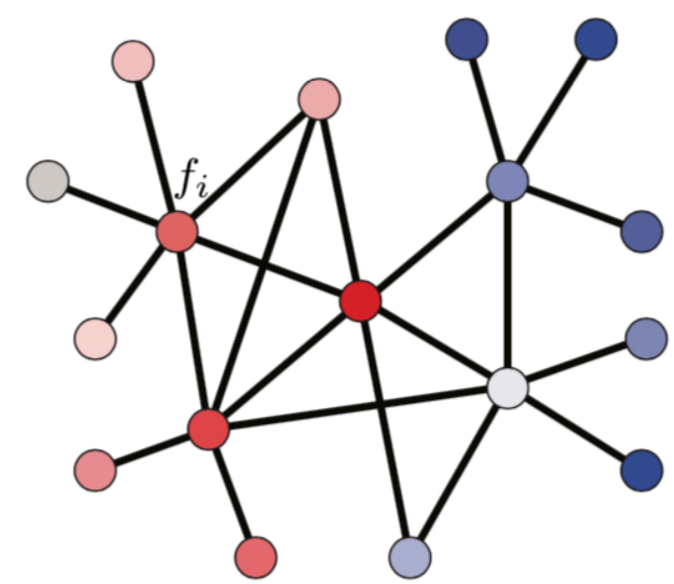
Define a *Hilbert space*:



$$f : \mathscr{V} \to \mathbb{R},\ \ F : \mathscr{E} \to \mathbb{R}$$

$$\langle f, g \rangle_{L^2(\mathscr{V})} = \sum_{i \in \mathscr{V}} a_i f_i g_i$$

$$\langle F, G \rangle_{L^2(\mathscr{E})} = \sum_{i \in \mathscr{E}} w_{ij} F_{ij} G_{ij}$$



Fourier Transform and the Laplacian (Euclidean case)

1D Laplacian: $\frac{d^2}{dx^2}$

On the real line: $\hat{f}(x) = \frac{1}{2\pi}\int f(\omega)\, e^{-i\omega x} d\omega$

Note: $\frac{d^2}{dx^2} e^{-i\omega x} = (-\omega^2)\, e^{-i\omega x} \implies e^{-i\omega x}$ eigenvectors of the 1D Laplacian operator

Generalise: $\Delta\Phi = \Phi\Lambda, \qquad \Phi = (\phi_1 \ldots, \phi_n)$

$$\Lambda = diag\left(\lambda_1, \ldots, \lambda_n\right)$$

Δ is symmetric so λ_l are real

Hammond et al. '09 [arXiv:0912.3848]

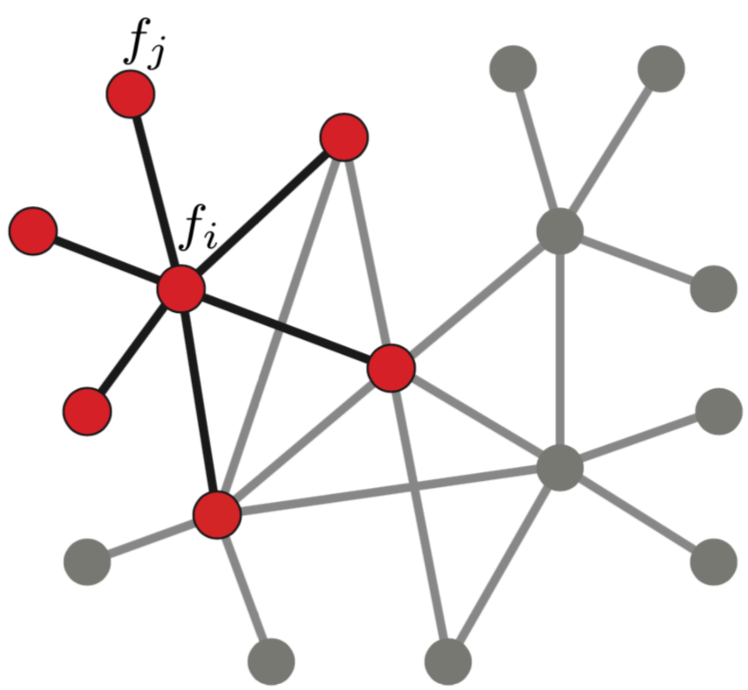
Fourier Transform and the Laplacian (Non-Euclidean case)

Assume analogous structure: $\Delta\Phi = \Phi\Lambda,$ $\Phi = (\phi_1 \ldots, \phi_n)$ Fourier basis on a graph

$$\Lambda = diag\left(\lambda_1, \ldots, \lambda_n\right)$$

Generalised Fourier Transform: $\hat{f}(\omega) = \Phi^{\dagger} f(\omega)$

Using graph Hilbert space:



Define: $\left(\nabla f\right)_{ij} = f_i - f_j$

$$\left(\mathbf{div} F\right)_i = \frac{1}{a_i} \sum_{j:(i,j)\in\mathcal{E}} w_{ij} F_{ij}$$

$$\Longrightarrow \quad \Delta = -\,\mathbf{div}\,\nabla$$

$$\left(\Delta f\right)_i = \frac{1}{a_i} \sum_{(i,j)\in\mathcal{E}} w_{ij}(f_i - f_j)$$

Convolution Theorem

Generally (Euclidean) :

$$\widehat{(f \star g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$$

$$= \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx \int_{-\infty}^{\infty} g(x) e^{-i\omega x} dx$$

$$= \sum_{i \geq 0} \langle f, \phi_i \rangle_{L^2(\chi)} \langle g, \phi_i \rangle_{L^2(\chi)} (\omega)$$

Discrete case (Euclidean):

$$\vec{f} = (f_1, \ldots, f_n)^{\mathbf{T}}, \quad \vec{g} = (g_1, \ldots, g_n)^{\mathbf{T}}$$

$$\widehat{f \star g} = \begin{bmatrix} \hat{g}_1 & & \\ & \ddots & \\ & & \hat{g}_n \end{bmatrix} \cdot \begin{bmatrix} \hat{f}_1 \\ \vdots \\ \hat{f}_n \end{bmatrix}$$

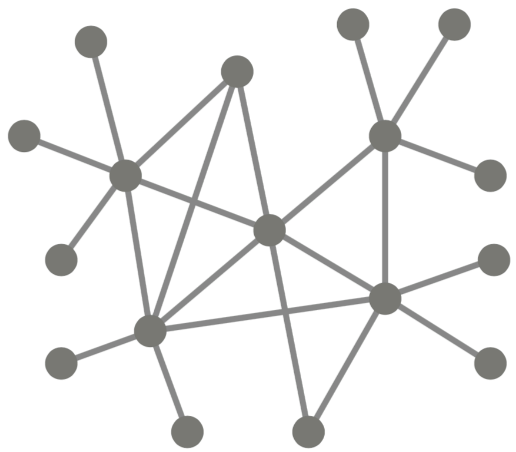
Fourier basis diagonalises the matrix

Problem: Cannot define $x - x'$ on graph

Solution: Take Convolution Theorem as a definition $\widehat{(f \star g)}(x) = \sum_{i \leq 0} \langle f, \phi_i \rangle_{L^2(\chi)} \langle g, \phi_i \rangle_{L^2(\chi)} (x)$

The Process

$\vec{f}$



$\vec{g}$

$\Phi^\dagger \vec{f}$

FT

FT

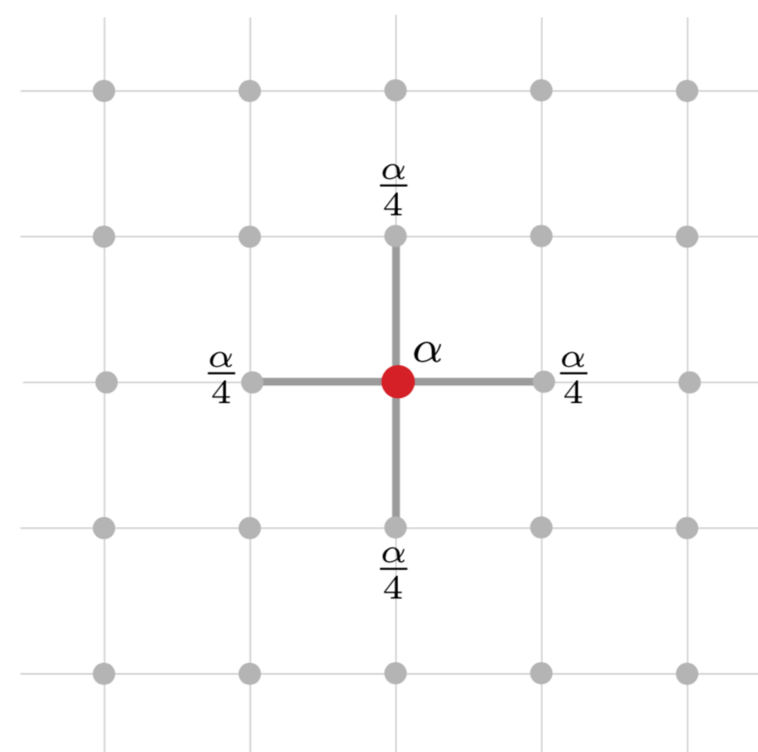
$\Phi^\dagger \vec{g}$

Convolution Theorem

$$\Phi^\dagger \vec{g}\, \Phi^\dagger \vec{f} = \mathbf{diag}(\hat{g}) \Phi^\dagger \vec{f}$$

IFT

$$\Phi(\mathbf{diag}(\hat{g}) \Phi^\dagger f)$$



Convolution on a graph

$$(f \star g)(x) = \sum_{i \leq 0} \underbrace{\underbrace{\langle f, \phi_i \rangle_{L^2(\chi)} \langle g, \phi_i \rangle_{L^2(\chi)}}_{\textbf{Fourier domain}} \phi_i(x)}_{\textbf{Inverse Fourier Transform}}$$

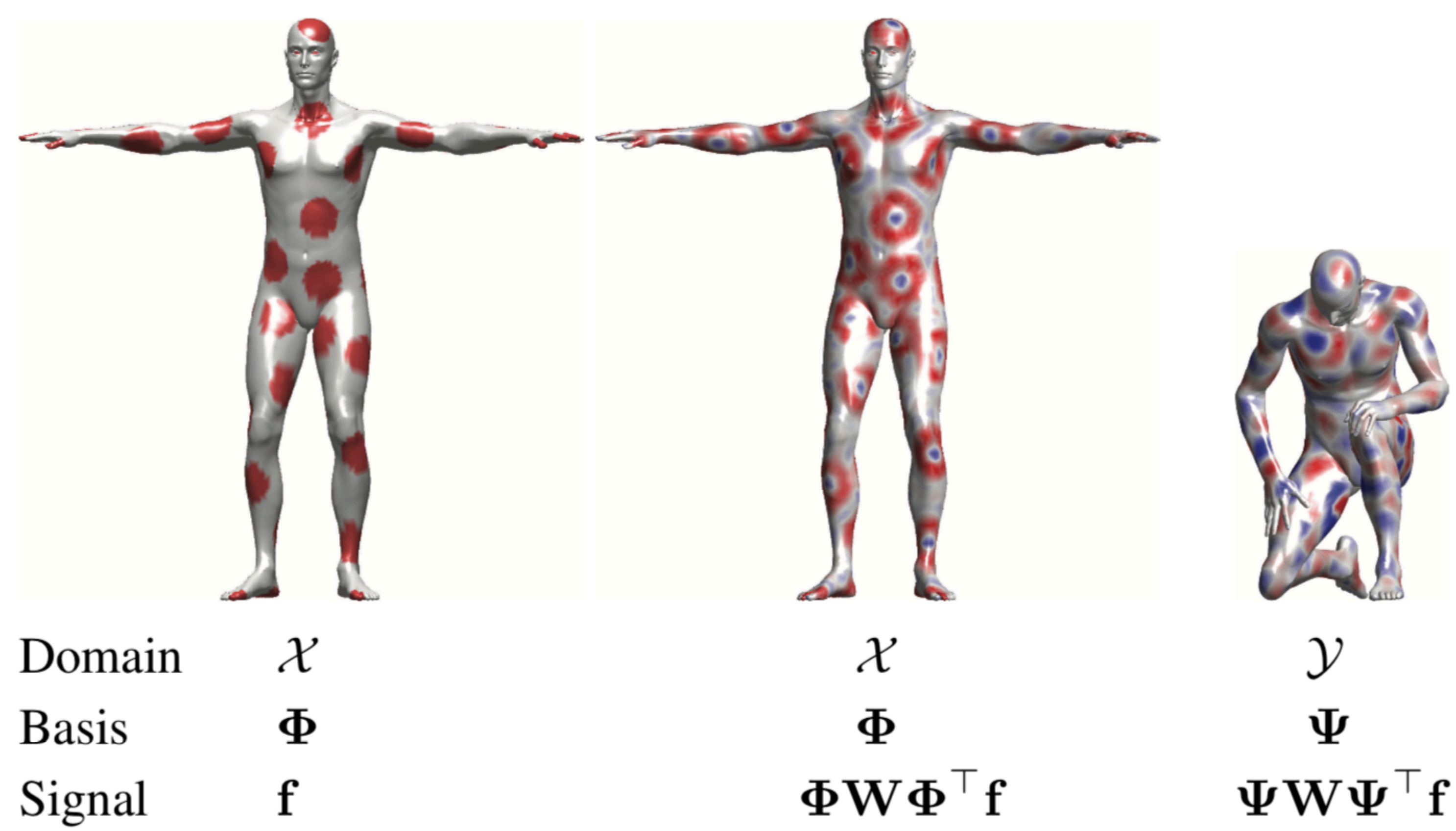
$$\vec{G}\vec{f} = \Phi\, \mathbf{diag}(\hat{g})\, \Phi^{\dagger}\, \vec{f}$$

Define: $$g_l = \xi\left(\sum_{l'=1}^{q} \Phi \Gamma_{l,l'} \Phi^{\mathbf{T}} f_{l'}\right),$$

$F = f_1, \ldots, f_p$ input signal

$G = g_1, \ldots, g_q$ output signal

Problems:
- Computationally inefficient to compute FT and IFT
- Filters are basis dependent



Bruna et al. '13 [arXiv:1312.6203]

How can we define the filters?

Problem: Want localised filters

Solution: Can be shown that spectral filters can be parametrised as: **diag**$(\Gamma_{l,l'}) = \beta_j(\lambda_i)\,\alpha_{l,l'}$

$$g_\alpha(\Delta) = \Phi\, g_\alpha(\Lambda)\, \Phi^{\mathbf{T}} \rightarrow g_\alpha(\lambda) = \sum_{j=0}^{r-1} \alpha_j\, \lambda^j$$

Problem: Filter coefficients are unstable under perturbation

Solution: e.g. use Chebyshev polynomials

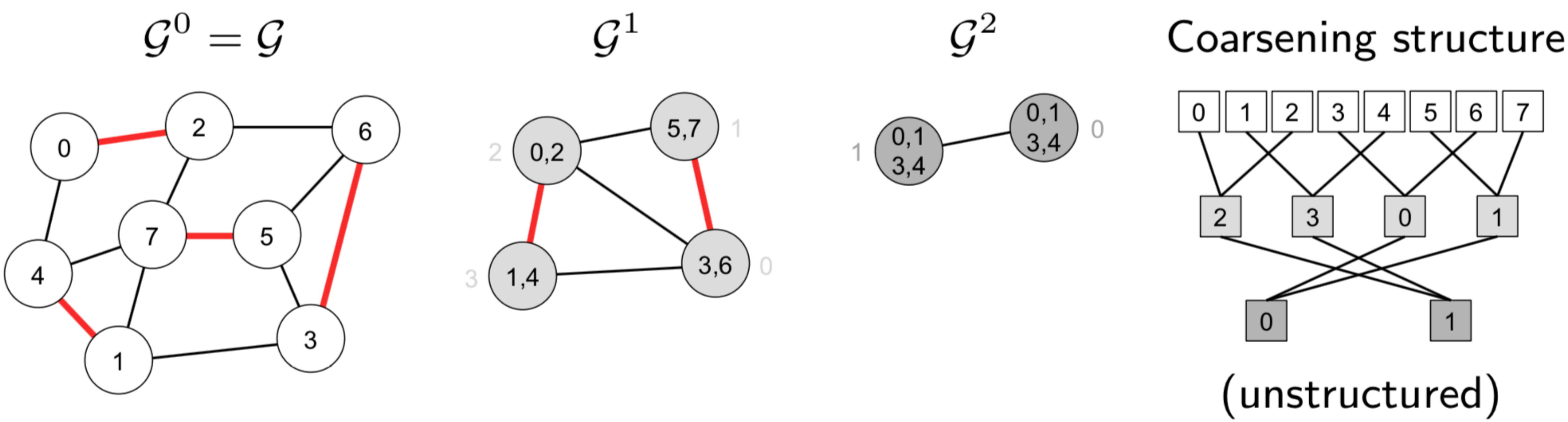
$$T_0(\lambda) = 1$$
$$T_1(\lambda) = \lambda$$
$$T_j(\lambda) = 2\lambda T_{j-1}(\lambda) - T_{j-2}(\lambda)$$

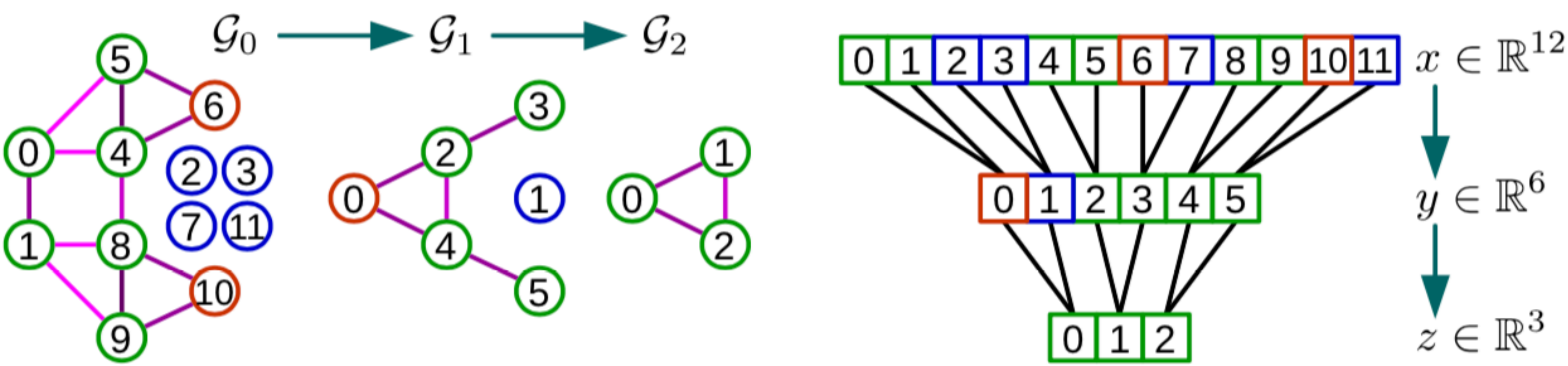
$$g_\alpha(\tilde{\Delta}) = \sum_{j=0}^{r-1} \alpha_j \Phi\, T_j(\tilde{\Lambda})\, \Phi^{\mathbf{T}} = \sum_{j=0}^{r-1} \alpha_j T_j(\tilde{\Lambda}),$$

$$\tilde{\Delta} = 2\lambda_n^{-1}\Delta - \mathbf{I}$$
$$\tilde{\Lambda} = 2\lambda_n^{-1}\Lambda - \mathbf{I}$$

Defferrard et al. '16 [arXiv:1606.09375]; Hammond et al. '09 [arXiv:0912.3848]

Pooling on a graph





Dhillon et al. '07; Defferrard et al. '16 [arXiv:1606.09375]

Example: citation networks

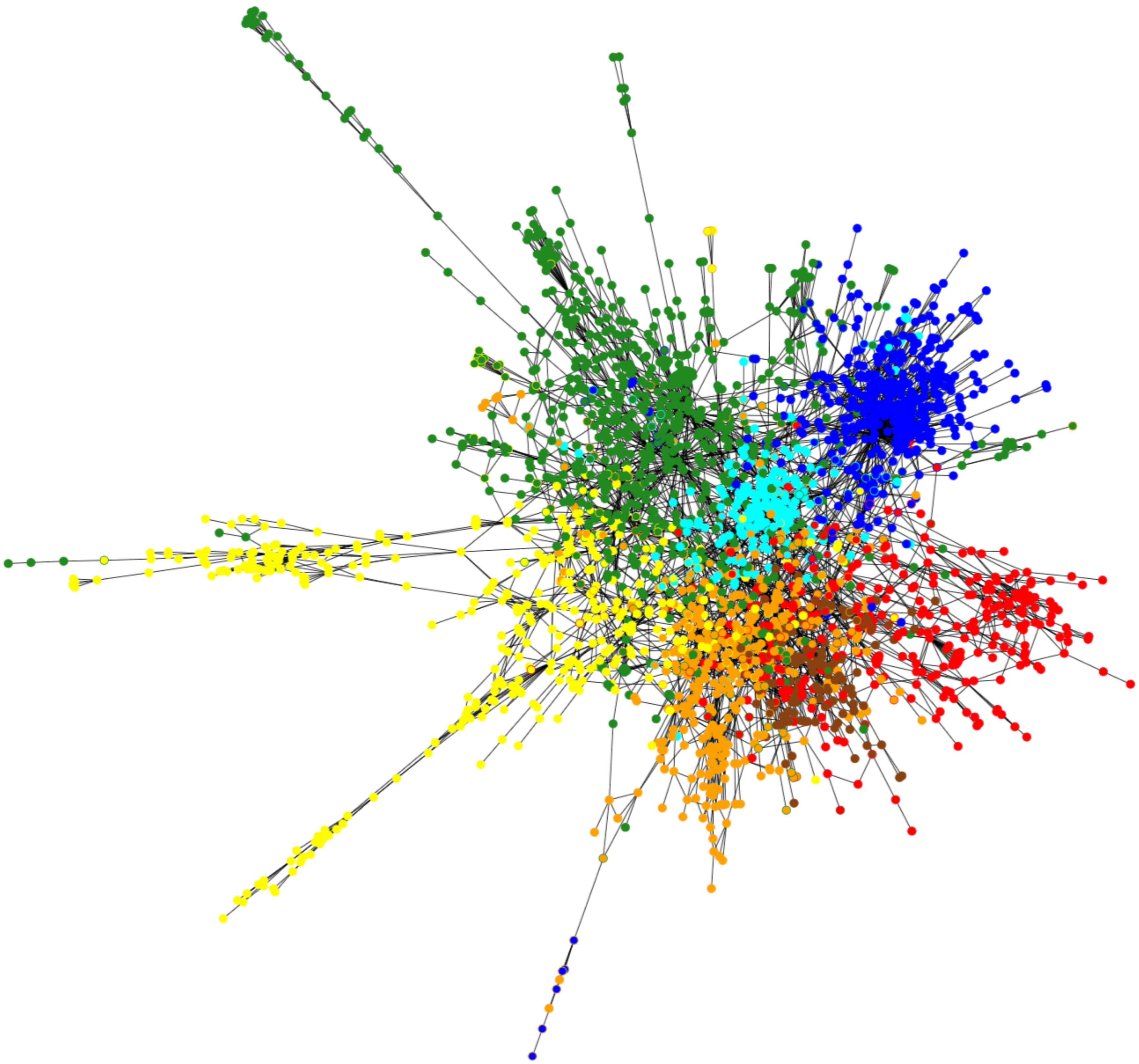


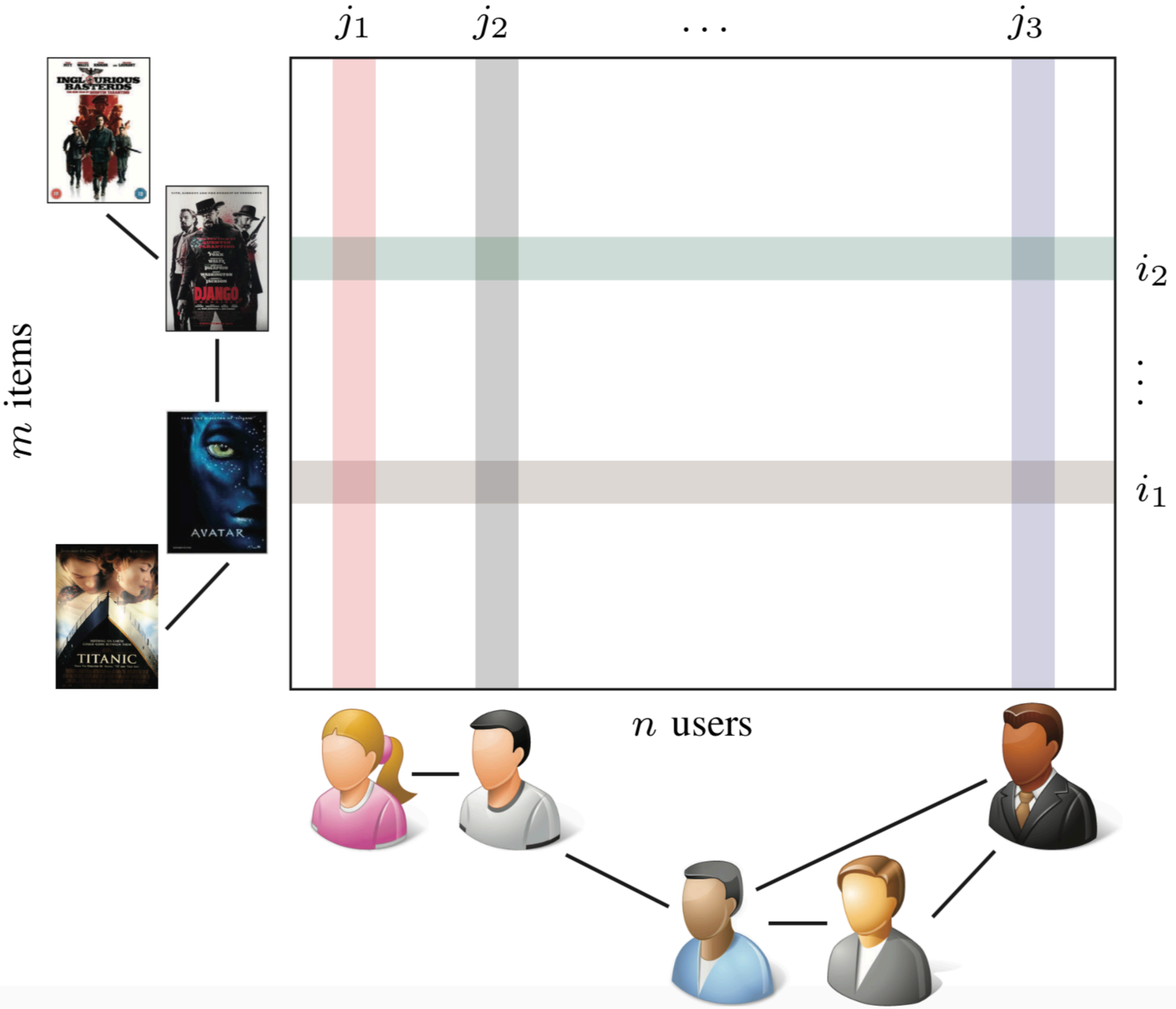
Figure: Monti et al. 2016; data: Sen et al. 2008

Example: citation networks

Method	**Cora**[1]	**PubMed**[2]
Manifold Regularization[3]	59.5%	70.7%
Semidefinite Embedding[4]	59.0%	71.1%
Label Propagation[5]	68.0%	63.0%
DeepWalk[6]	67.2%	65.3%
Planetoid[7]	75.7%	77.2%
Graph Convolutional Net[8]	**81.59%**	**78.72%**

Classification accuracy of different methods on citation network datasets

Monti et al. 2016; data: [1,2]Sen et al. 2008; methods: [3]Belkin et al. 2006; [4]Weston et al. 2012; [5]Zhu et al. 2003; [6]Perozzi et al. 2014; [7]Yang et al. 2016; [8]Kipf, Welling 2016



j_1
j_2
$\cdots$
j_3
i_2
i_1
m items
n users