Graph Neural Networks on Large Random Graphs

joint works with

Nicolas KERIVEN

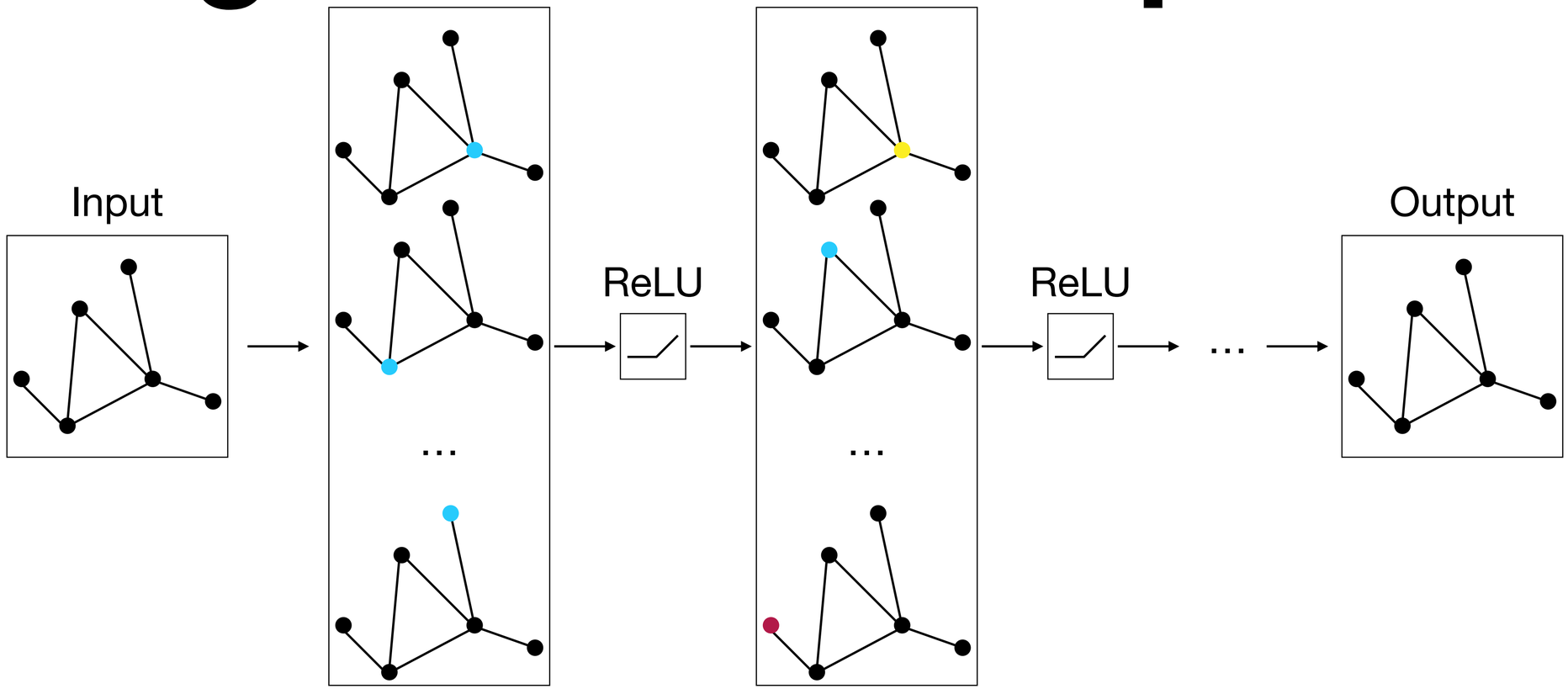
Alberto BIETTI

Samuel VAITER
CNRS & LABO J.A. DIEUDONNÉ

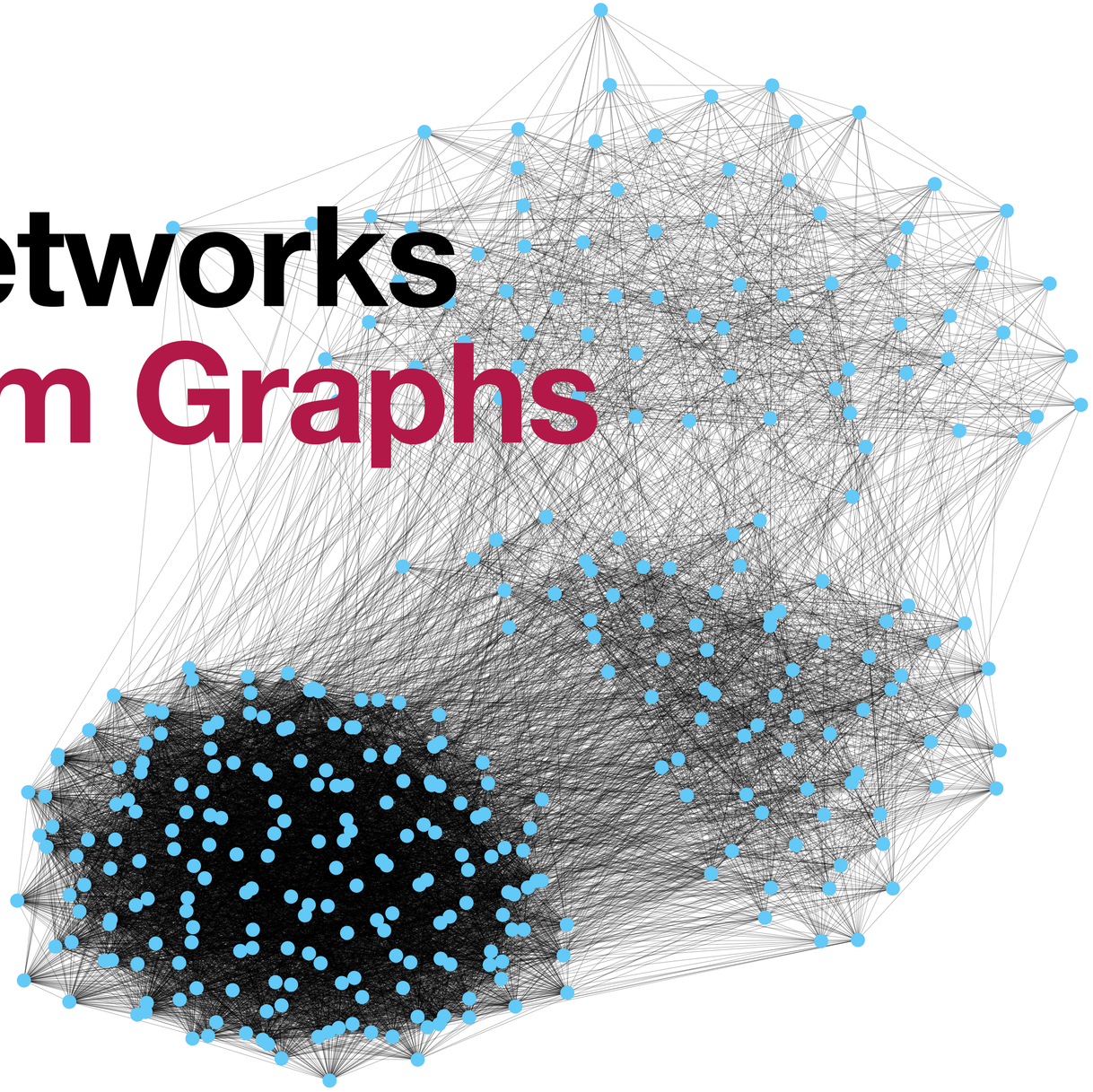
UNIVERSITÉ CÔTE D'AZUR

cnrs

Graph Neural Networks
on Large Random Graphs



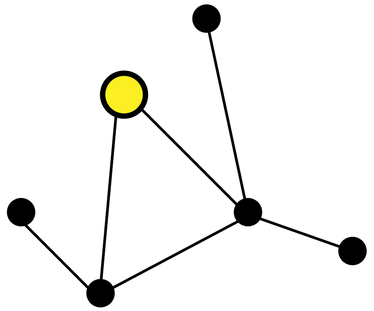
Graph Neural Networks on Large Random Graphs



Graphs and Invariants

Graphs

Graph $G = (V, E)$

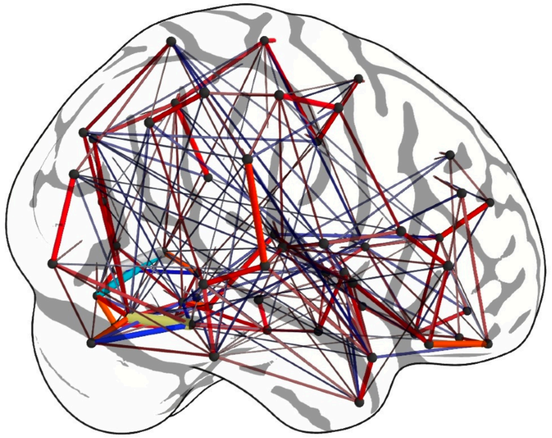


unordered

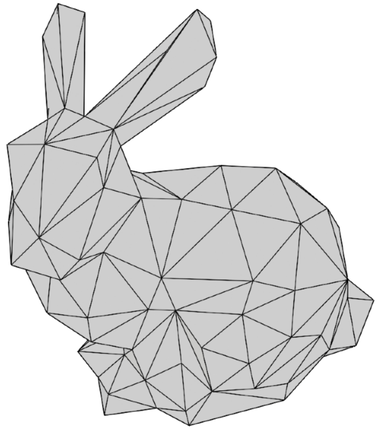
Adjacency matrix

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \in \{0,1\}^{n \times n}$$

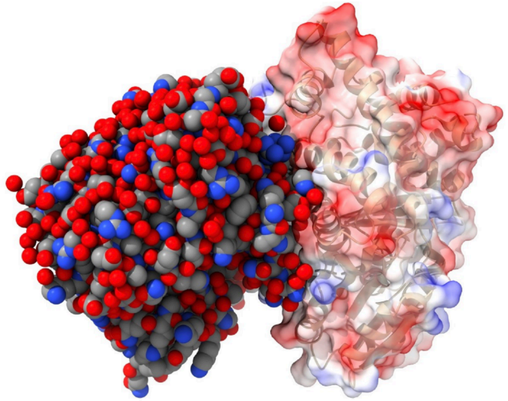
sparse



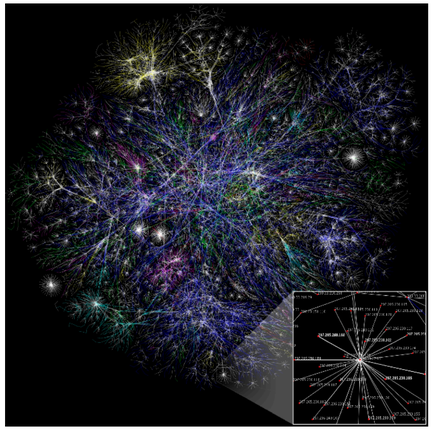
brain connectivity
[Varoquaux et al. ’12]



mesh
[Turk-Levoy ’96]



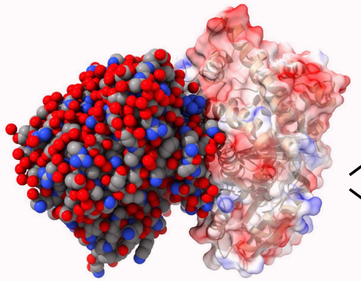
protein
[Sverrisson et al. ’21]



virtual network
[Lyon ’05]

Tasks in machine learning on graphs

Graph classification

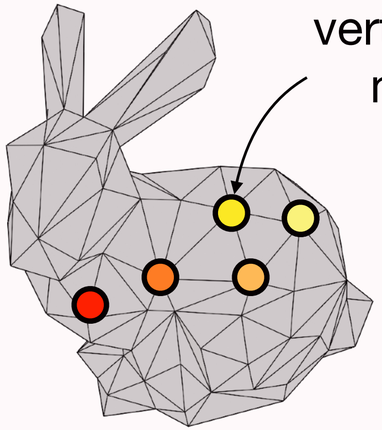


useful

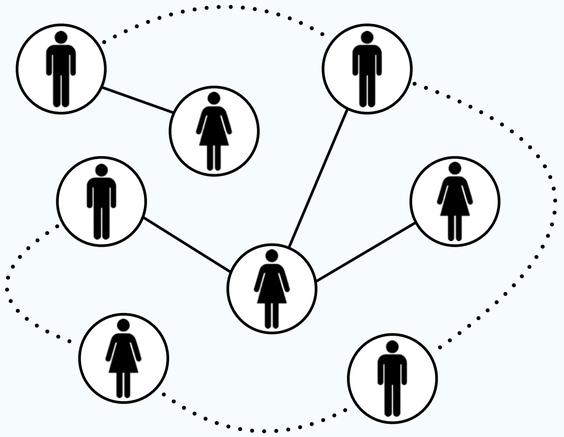
not useful

supervised learning

Node regression

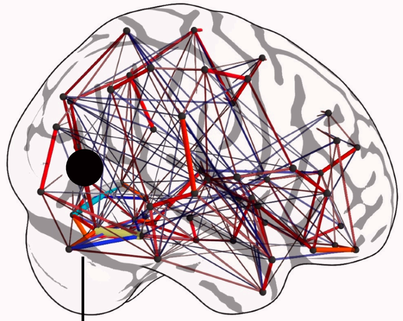


Link prediction



(un/semi)supervised learning

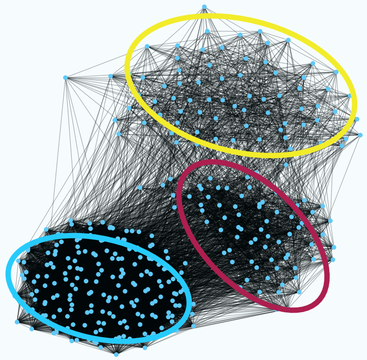
Node classification



neuron involved

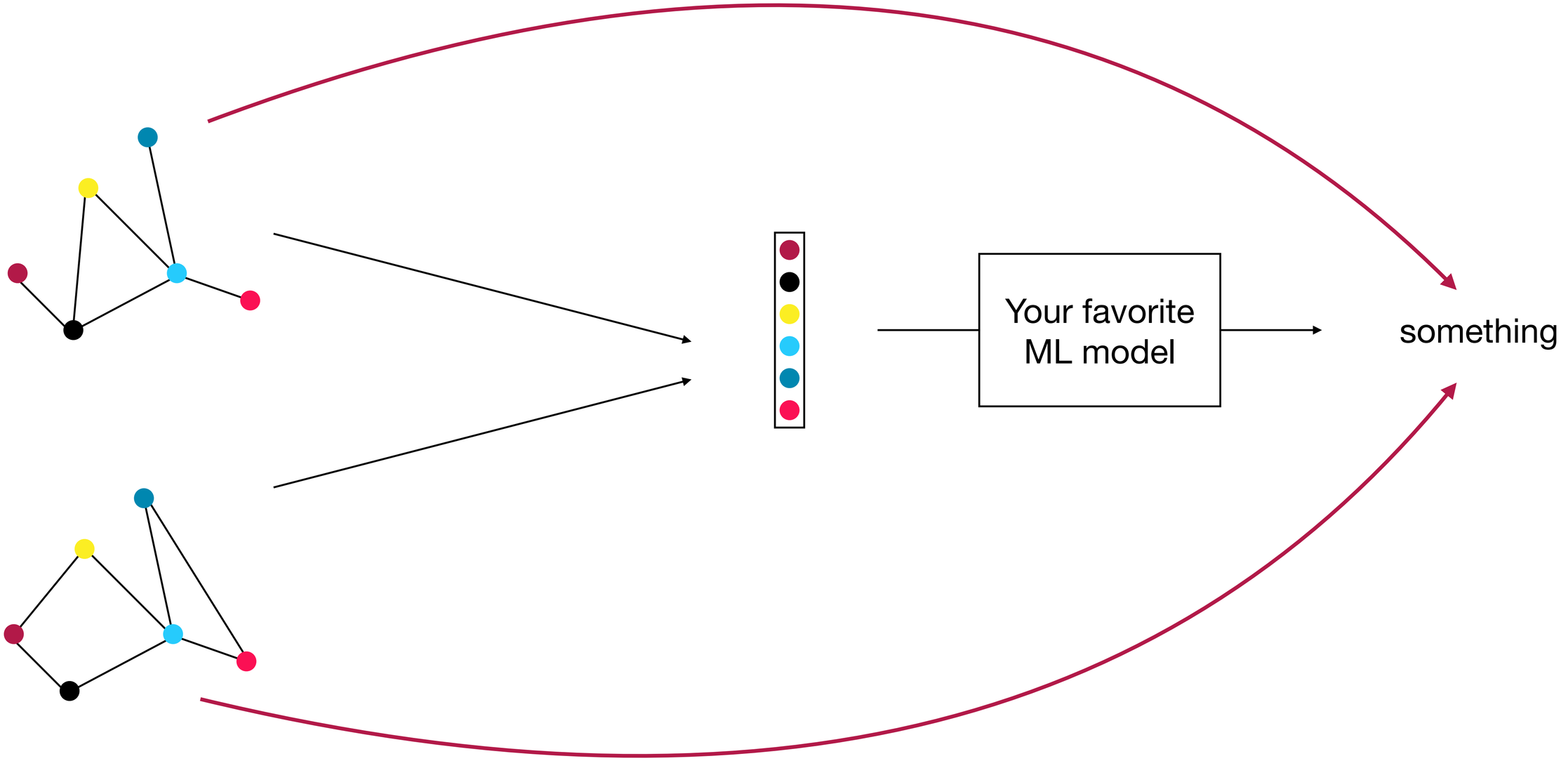
neuron not involved

Community detection

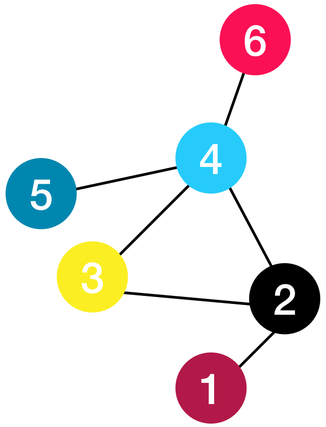


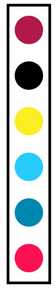
and more!

Tasks in machine learning ~~on graphs~~ ?



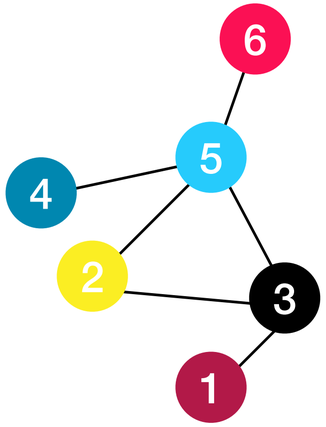
Tasks in machine learning ~~on graphs~~ ?





Your favorite ML model

something



Your favorite ML model

something different

We need to have “some” **invariance**!

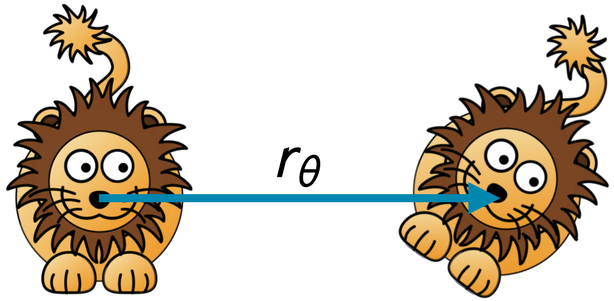
(In/Equi)variance

Group action

group, set

$$\begin{aligned} \mathcal{G} \times \mathcal{X} &\to \mathcal{X} \\ (g, x) &\mapsto g \cdot x \end{aligned}$$

$\mathcal{G} = \mathrm{SO}(2), \mathcal{X} = \mathbb{R}^2$



$\mathcal{G} = \mathfrak{S}_n, \mathcal{X} = \mathbb{R}^n$

$$(\sigma, x) \mapsto \begin{pmatrix} x_{\sigma(1)} \\ \vdots \\ x_{\sigma(n)} \end{pmatrix}$$

Invariance

$$f(g \cdot x) = f(x)$$

$\mathcal{G} = \mathrm{SO}(2), \mathcal{X} = \mathbb{R}^2, f = \|\cdot\|$

$$\|r_\theta(x)\| = \|x\|$$

Equivariance

$$f(g \cdot x) = g \cdot f(x)$$

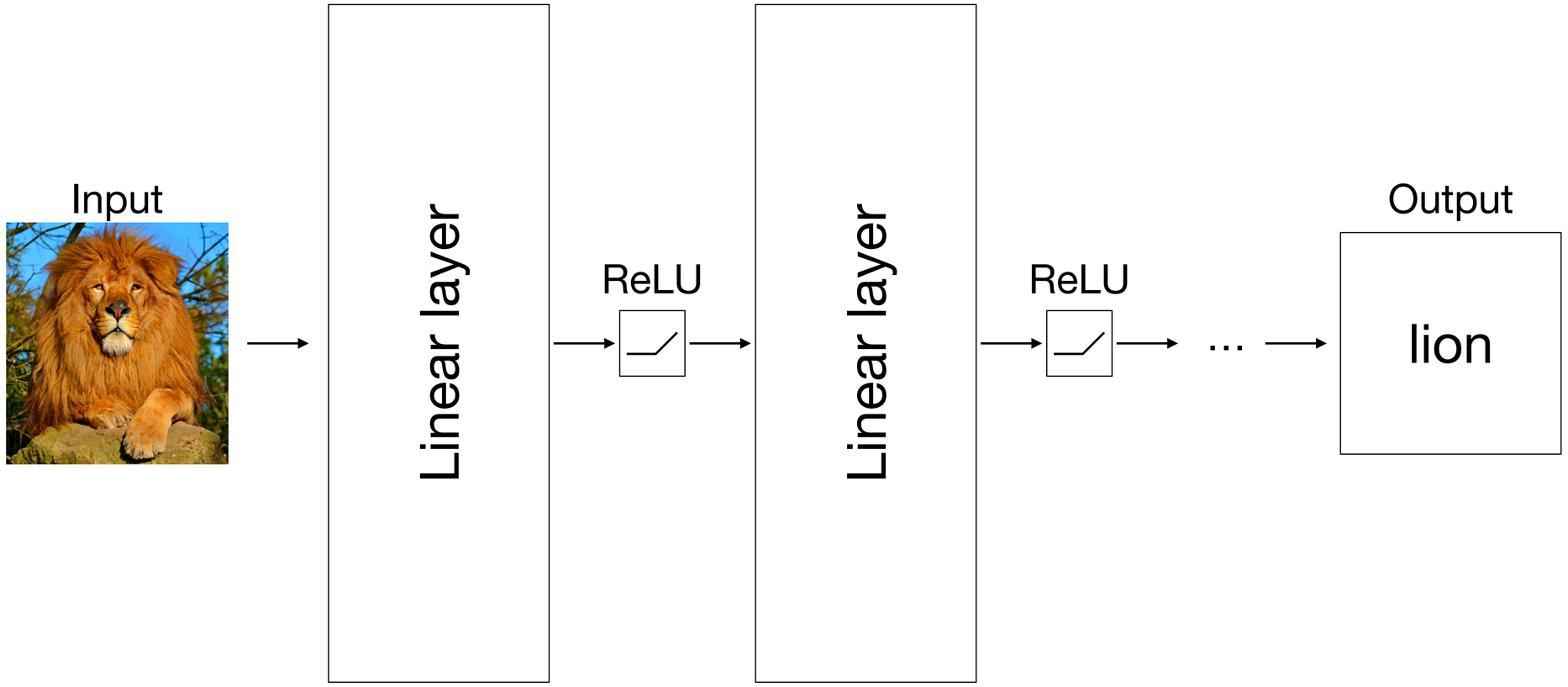
could be another action

$\mathcal{G} = \mathfrak{S}_n, \mathcal{X} = \mathbb{R}^n, f(x) = \arg\min_i x_i$

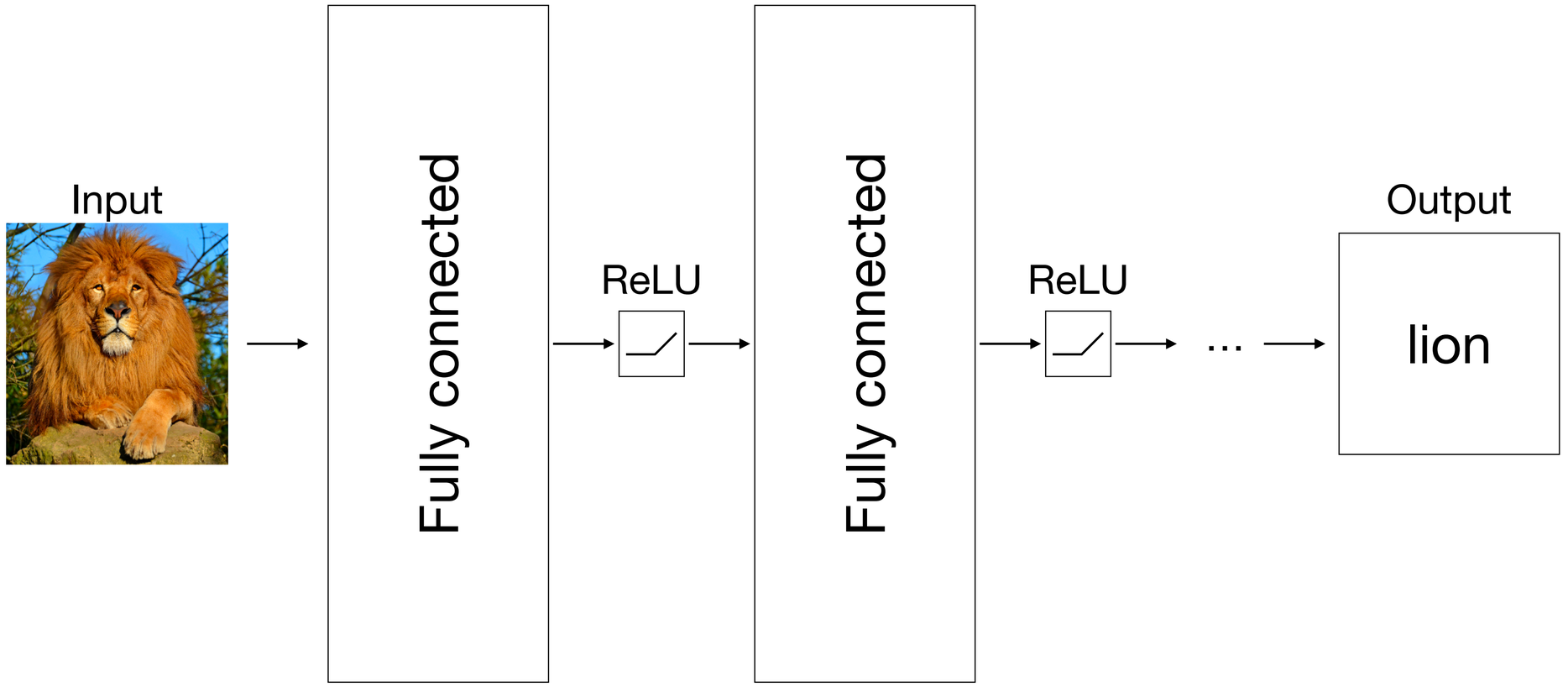
$$f(\sigma \cdot x) = \sigma(\arg\min_i x_i)$$


Your favorite
(in/equi)variant
ML model?

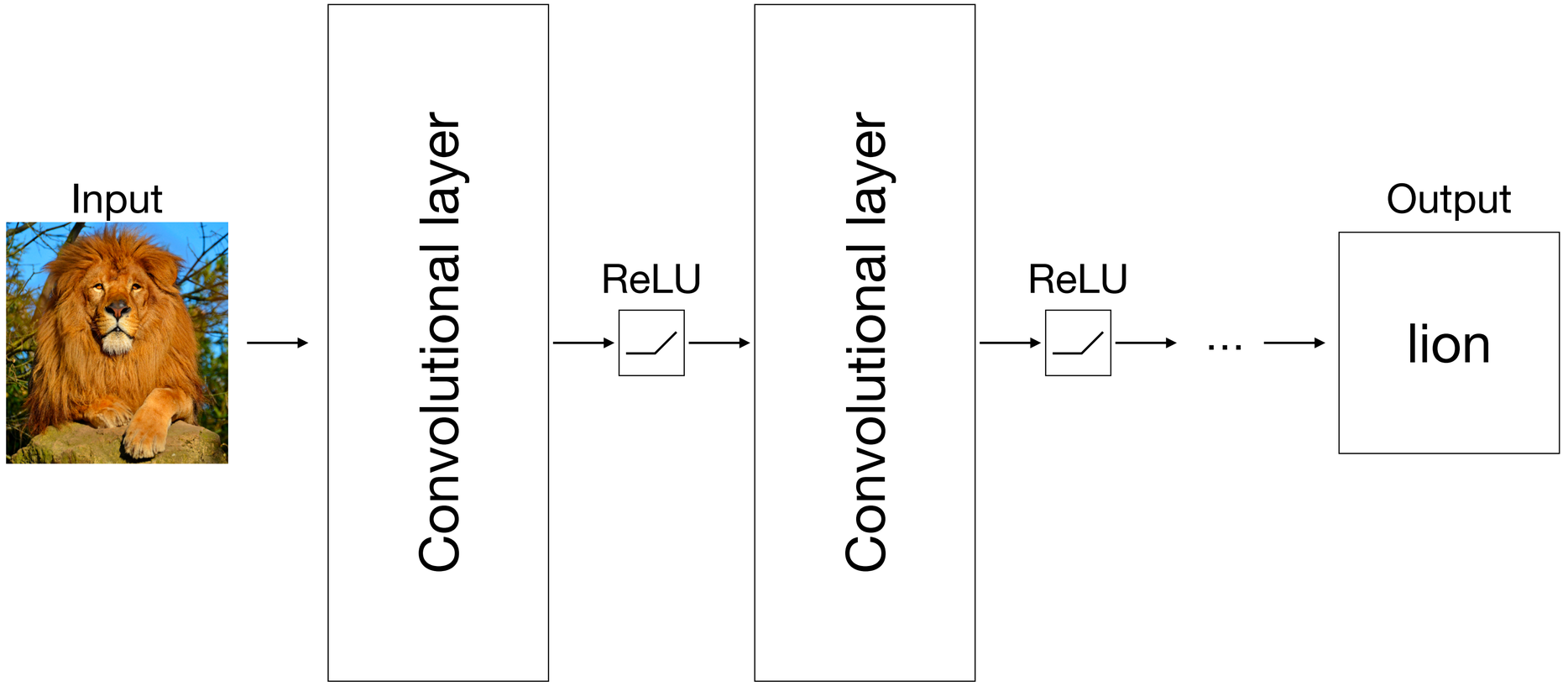
Deep Neural Networks



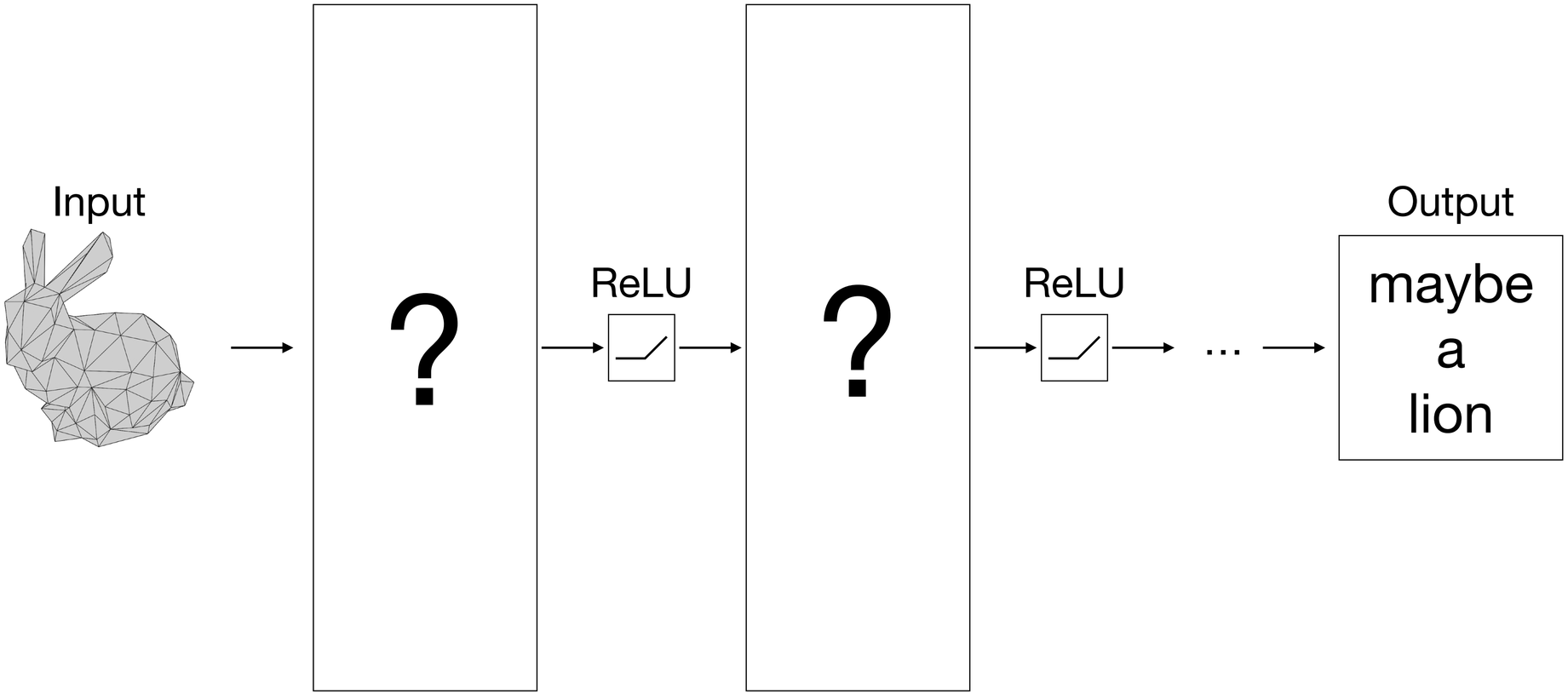
DNN: Multilayer Perceptron



DNN: Convolutional Neural Network (CNN)



DNN: Graph Neural Network?



Graph Neural Network

The Fourier Way

Convolution

Convolution (1D continuous)

$$(g * h)(t) = \int_{\mathbb{R}} g(t-\tau)h(\tau)d\tau$$

signal, filter

local averaging

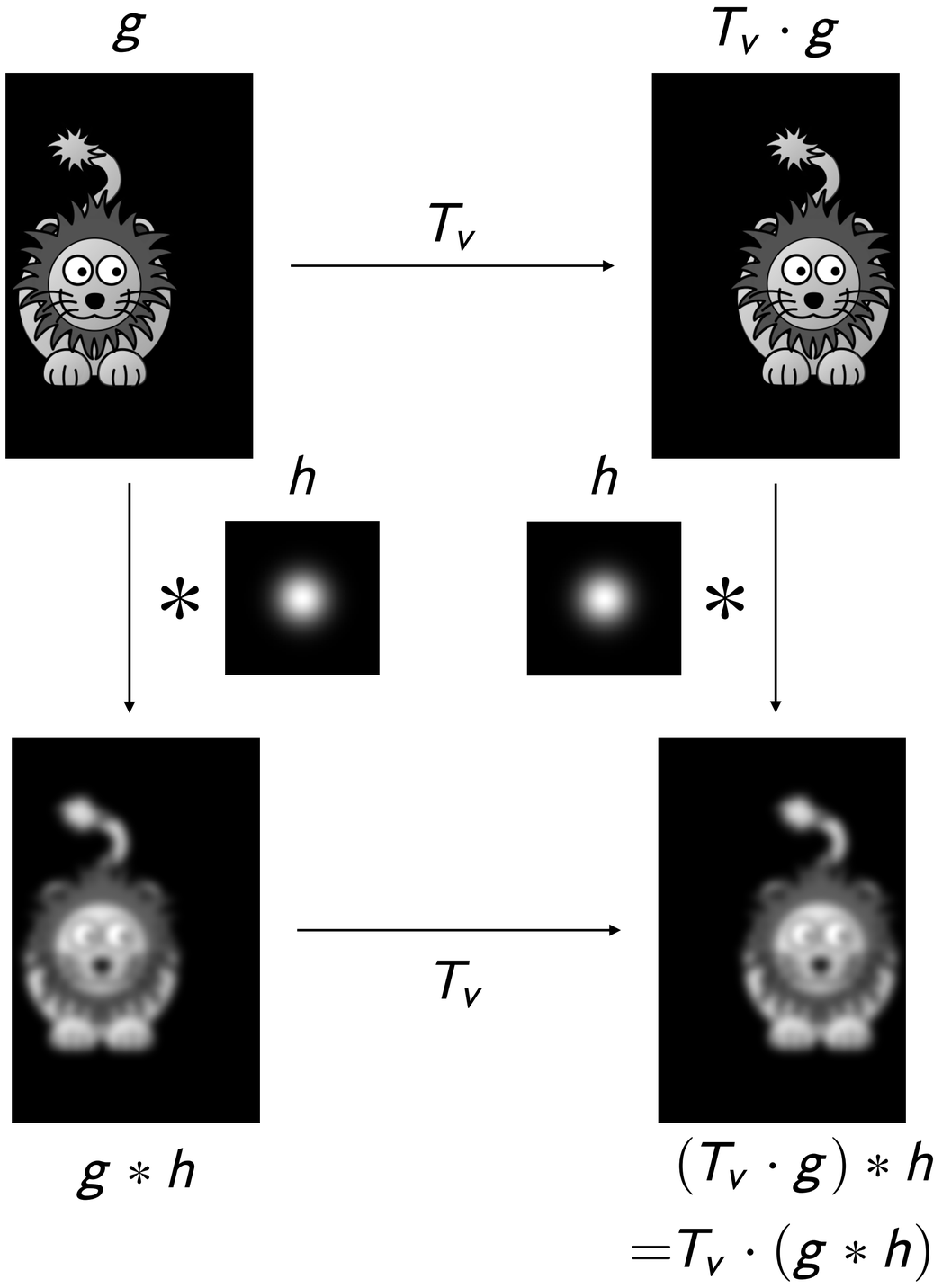
Convolution is *shift-equivariant*

$$(T_v \cdot g) * h = T_v \cdot (g * h)$$

shift operator

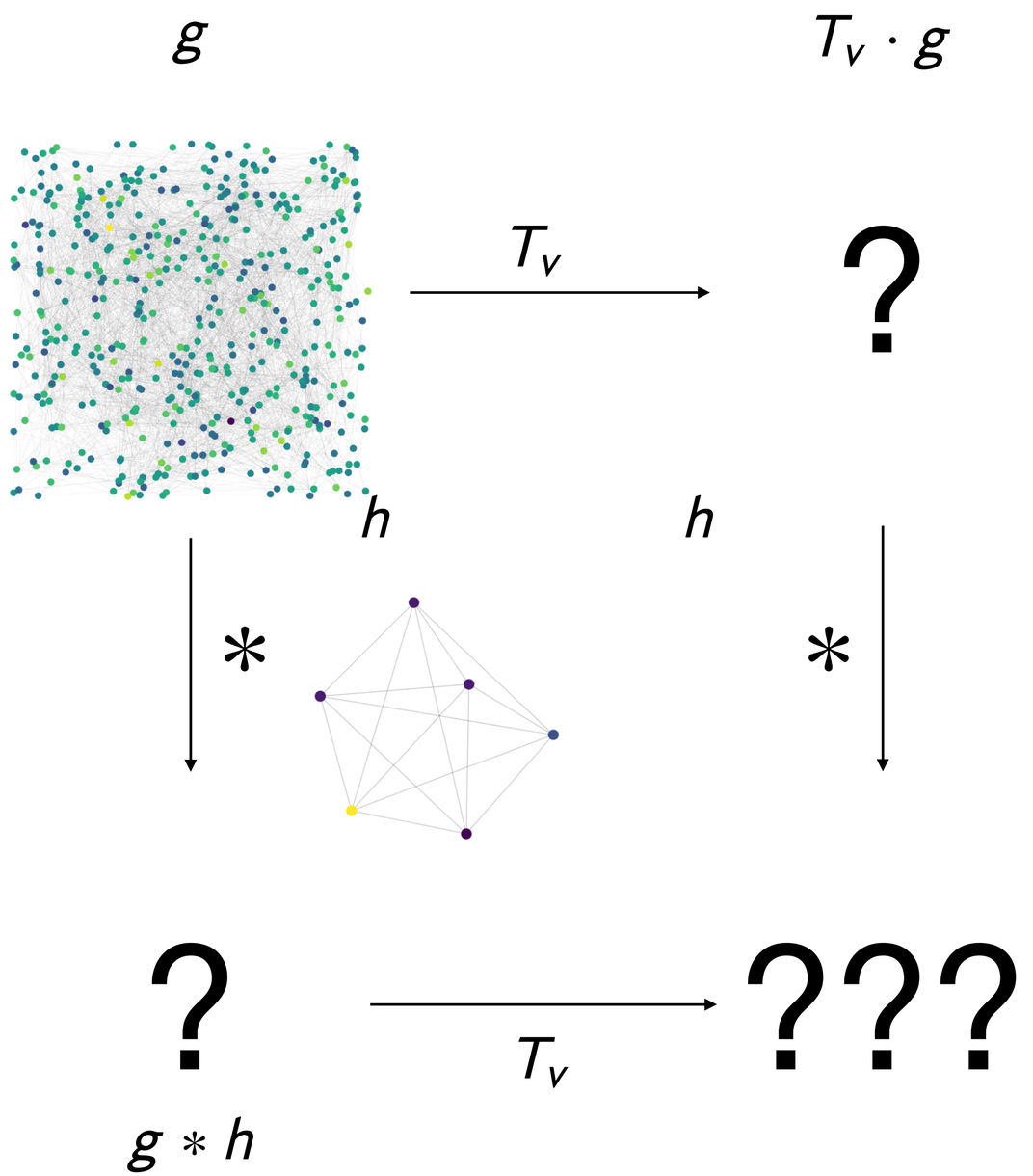
Traditional signal processing: fixed (e.g., wavelets)

ML: **learned**



Convolution *on a graph*

convolution?
shift-equivariant?



Convolution and Fourier

Convolution (1D continuous)

$$(g * h)(t) = \int_{\mathbb{R}} g(t-\tau)h(\tau)d\tau$$

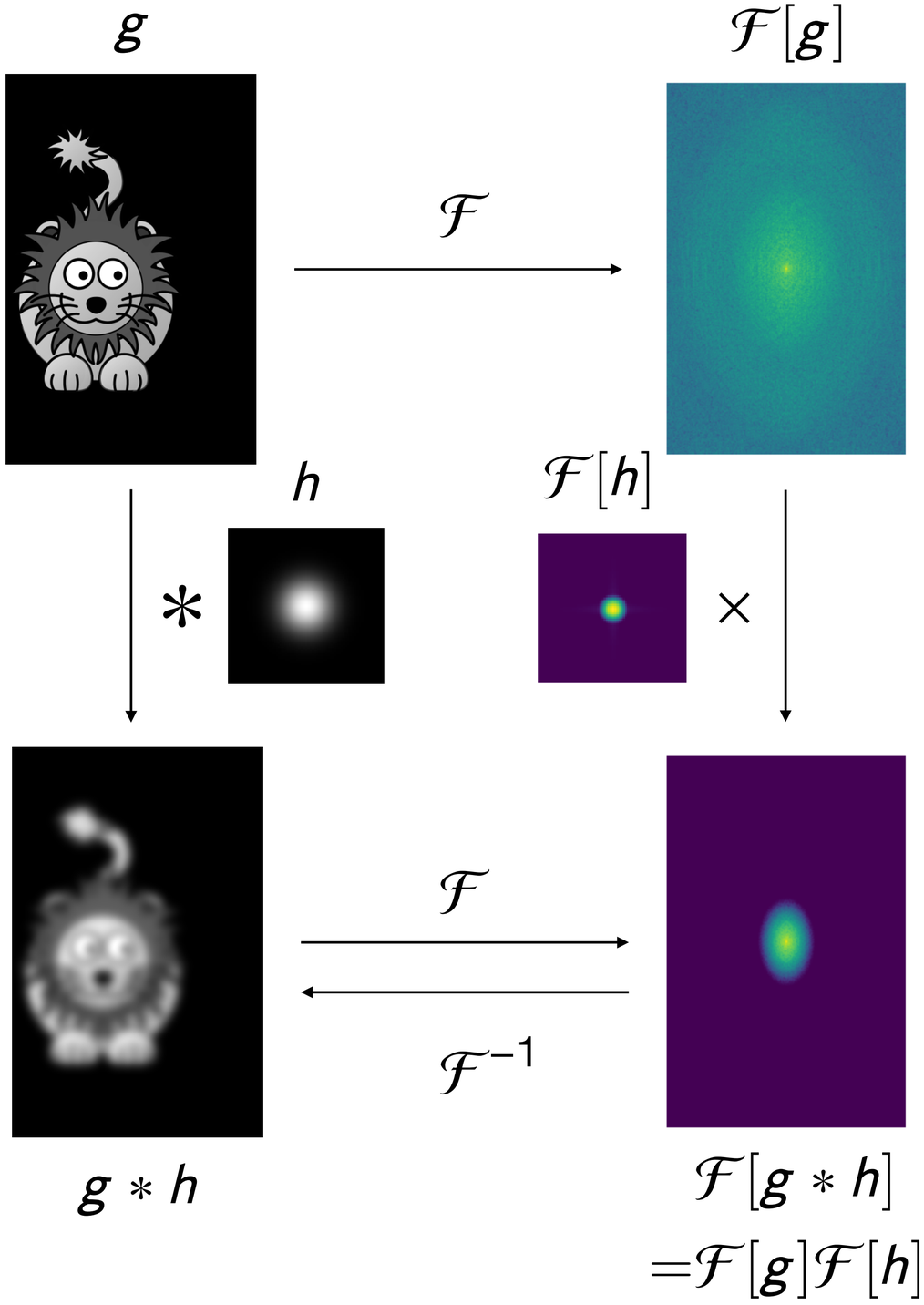
signal, filter

local averaging

Convolution *is* product in Fourier

$$\mathcal{F}[g * h] = \mathcal{F}[g]\mathcal{F}[h]$$

$$\mathcal{F}[g](\omega) = \int g(t)e^{-2i\pi\omega t}\mathrm{d}t$$
$$= \langle g, \exp_{-2i\pi\omega}\rangle_{L^2(\mathbb{R})}$$



Fourier on graphs

Eigenfunctions of the Laplacian

$$-\Delta f = \lambda f \xleftrightarrow{\text{on the torus}} f = \exp_{-2i\pi\omega}$$

(Torus) Laplacian $\Delta = \sum_i \frac{\partial^2}{\partial x_i^2}$

(Graph) Laplacian L: $\langle z, Lz \rangle = \sum_{i \sim j} (x_i - x_j)^2$

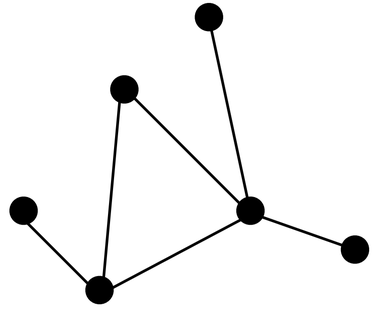
$$L = D - A$$

degree matrix $D = \mathrm{diag}(d_i)$

adjacency matrix

Normalized Laplacian

$$L = \mathrm{Id} - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$



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

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

Fourier on graphs

Normalised Laplacian

$$L = \mathrm{Id} - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

Diagonalization

$$L = U \Lambda U^{\top}$$

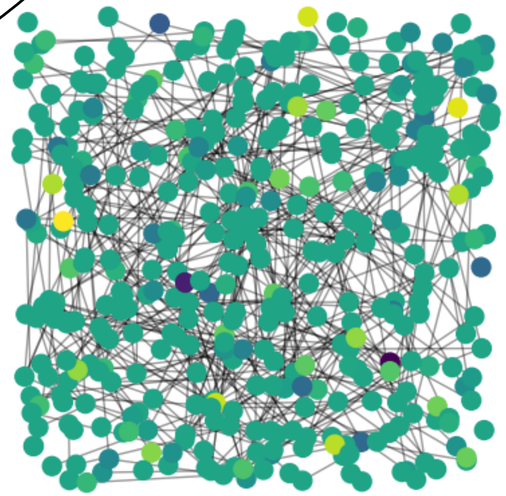
Fourier transform

$$\mathcal{F} z = U^{\top} z$$

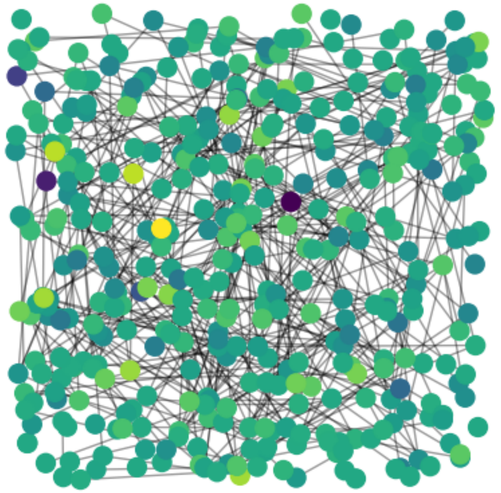
Inverse Fourier transform

$$\mathcal{F}^{-1} \theta = U \theta$$

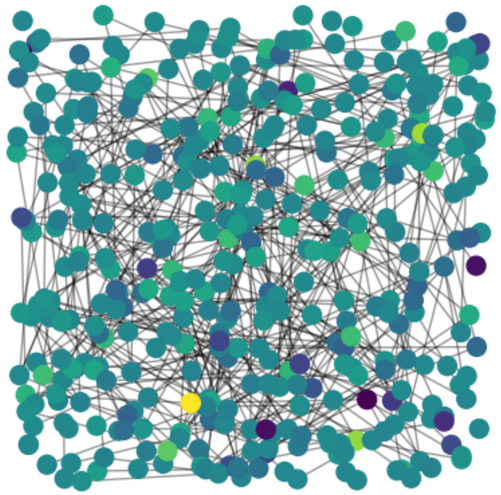
λ = 1.0000000000000009



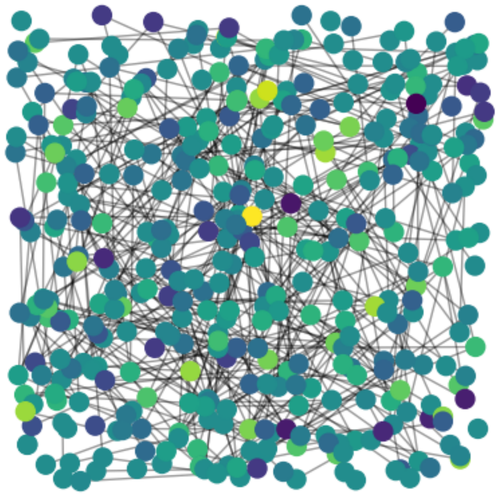
λ = 0.5250670287796952



λ = 1.190716231138907



λ = 1.8125226032632884



Filtering on graphs

Normalised Laplacian

$$L = \mathrm{Id} - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

Diagonalization

$$L = U \Lambda U^\top$$

Costly in practice 😥

Fourier transform

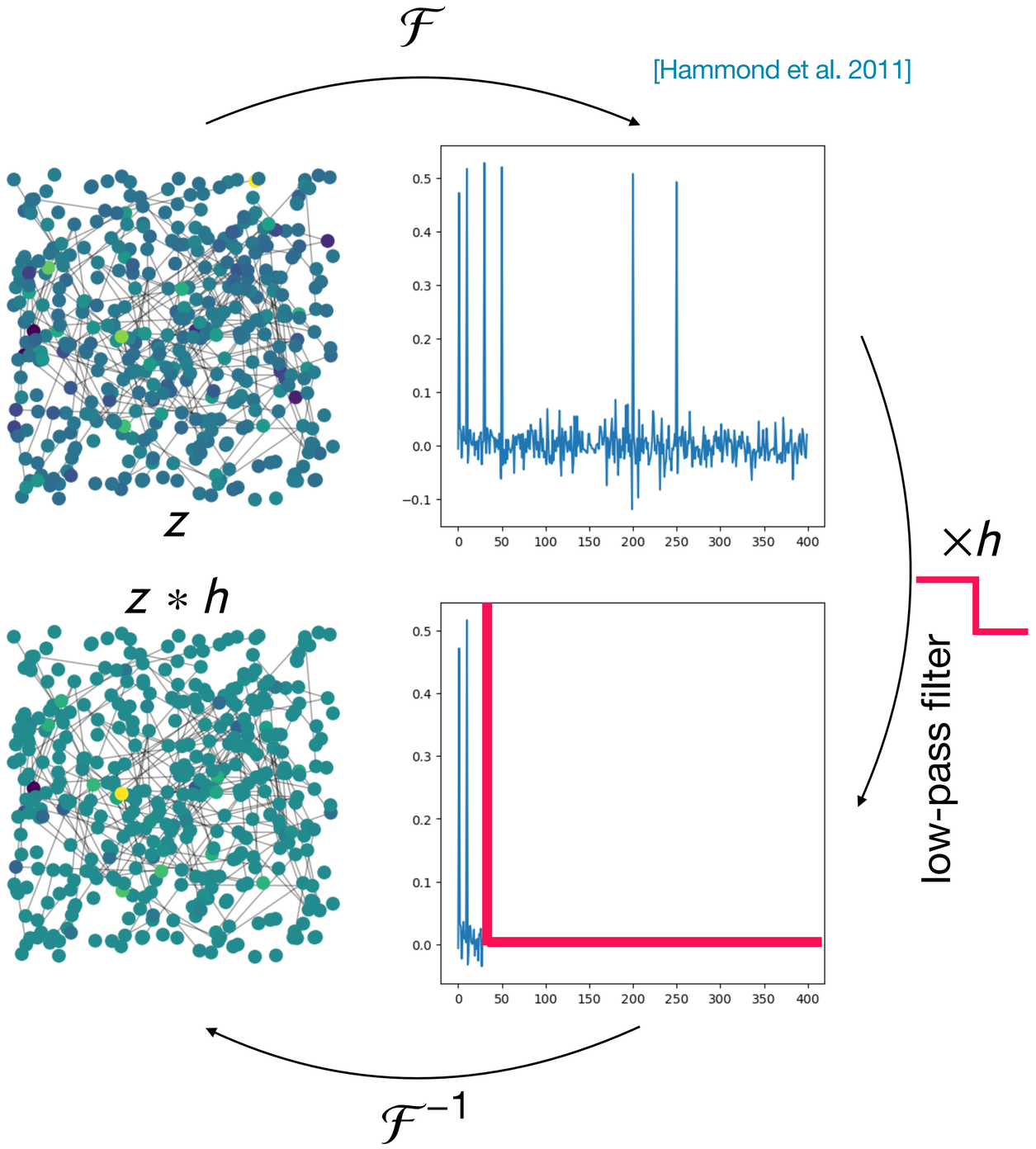
$$\mathcal{F} z = U^\top z$$

Inverse Fourier transform

$$\mathcal{F}^{-1} \theta = U \theta$$

Polynomial filters

$$z * h = \left(\sum_m \beta_m L^m \right) z$$



(Spectral) Graph Neural Network

[Henaff et al. '15, Defferrard et al. '16]

$z_j^{(k)} \xrightarrow[\text{signal over nodes}]{\text{propagate}} z_j^{(k+1)}$

Normalized Laplacian

$L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

$$z_j^{(k+1)} = \rho\left(\Sigma_i h_{ij}^{(k)}(L) z_i^{(k)}\right)$$

could include bias term

non-linearity
e.g. ReLU

Trained polynomial filter $h(L) = \sum_m \beta_m L^m$

output

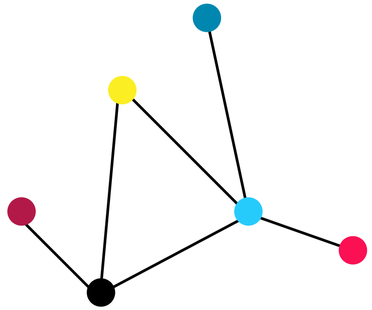
node classification
$\mathfrak{S}$-equivariant
node-signal $\Phi_G(z) = z^{(K)}\theta$

last (linear) layer of the GNN

graph classification
$\mathfrak{S}$-invariant
pooling $\bar{\Phi}_G(z) = n^{-1} \sum_i \Phi_G(z)_i$

Large (Random) Graphs

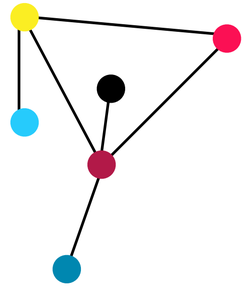
Graph isomorphism

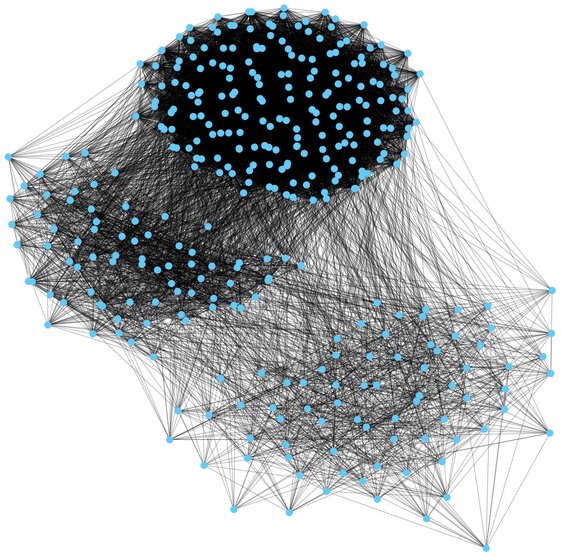


$\mathfrak{S}_n$

$=$

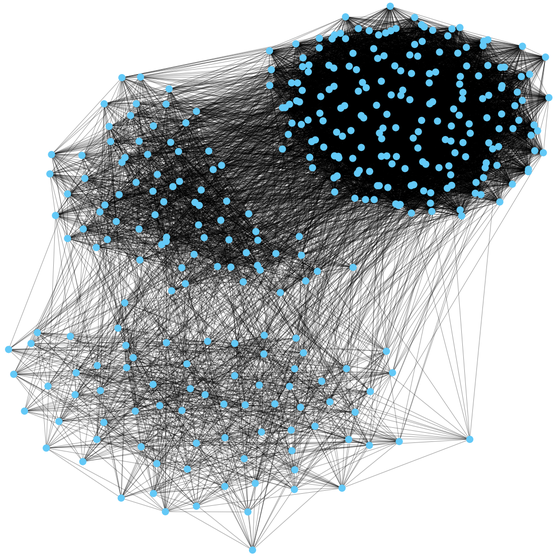
?





$=$

?



Random graph models

Latent position models $\Gamma = (P, W, f)$

$x_i \overset{\text{i.i.d}}{\sim} P \in \mathbb{R}^d$ — unknown latent variables

$a_{ij} \sim \text{Ber}(\alpha_n W(x_i, x_j)) \in \{0, 1\}$ — connectivity kernel

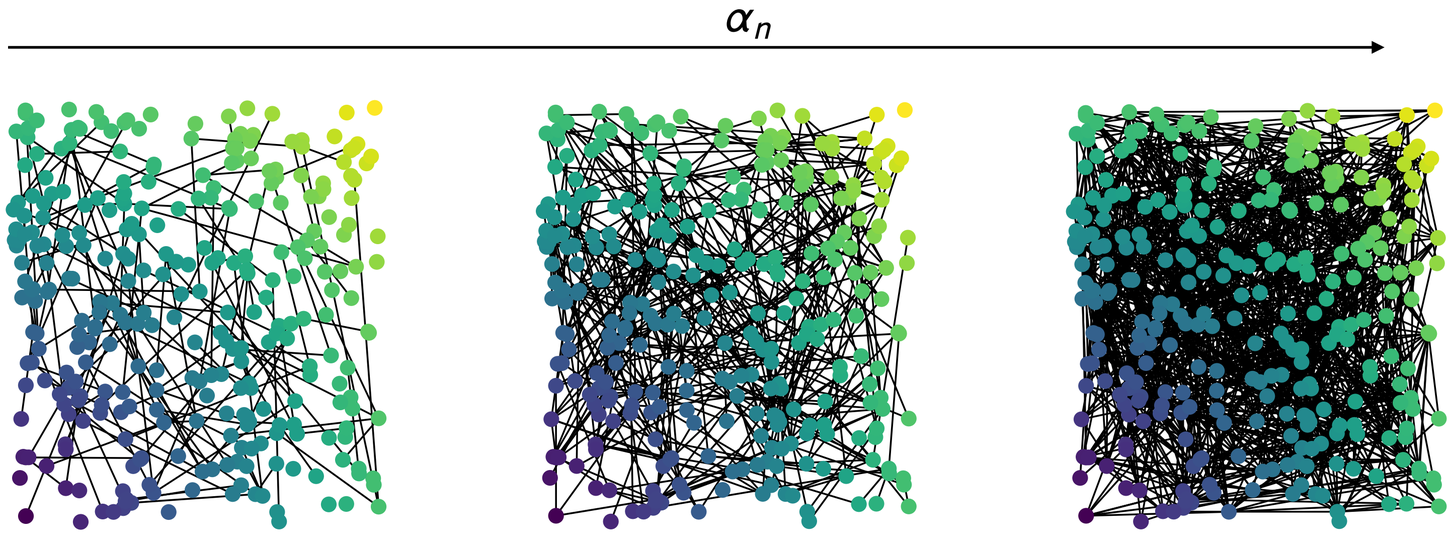
$z_i = f(x_i)$ — optional node features

Sparsity

$\alpha_n \sim 1$	dense
$\alpha_n \sim (\log n)/n$	**relatively sparse**
$\alpha_n \sim 1/n$	sparse

ex:
- Stochastic Block Model
- Epsilon graph

Example: Uniform + Gaussian + Norm



$P = \mathcal{U}([0,1]^2)$ $W(x, x') = \exp(-\|x - x'\|^2)$ $f(x) = \|x\|$

[Chung&Lu '04, Lovasz '12]

Discrete VS Continuous

Graph Neural Network

$z_j^{(k)} \xrightarrow[\text{signal over nodes}]{\text{propagate}} z_j^{(k+1)}$

$$z_j^{(k+1)} = \rho\left(\Sigma_i h_{ij}^{(k)}(L) z_i^{(k)}\right)$$

non-linearity

Trained polynomial filter $h(L) = \sum_m \beta_m L^m$

Normalized Laplacian

$$L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

literature

[von Luxburg et al. '04]
[Rosasco et al '11]
[Lei—Rinaldo '15]
[KV '22]
...

output

$\mathfrak{S}$-equivariant
node-signal $\Phi_G(z) = z^{(K)}\theta$
$\mathfrak{S}$-invariant
pooling $\bar{\Phi}_G(z) = n^{-1}\sum_i \Phi_G(z)_i$

Continuous Graph Neural Network

$f_j^{(k)} \xrightarrow[\text{functions over latent space}]{\text{propagate}} f_j^{(k+1)}$

$$f_j^{(k+1)} = \rho\left(\Sigma_i h_{ij}^{(k)}(\mathcal{L}) f_i^{(k)}\right)$$

Laplacian operator

$$\mathcal{L}f = \int \frac{W(\cdot, x)}{\sqrt{d(\cdot)d(x)}} f(x) dP(x)$$

output

GL-equivariant
function $\Phi_{W,P}(f) = \theta^\top f^{(K)}$
GL-invariant
vector $\bar{\Phi}_{W,P}(f) = \int \Phi_{W,P}(f)(x) \mathrm{d}P(x)$

Non-asymptotic convergence of GNN

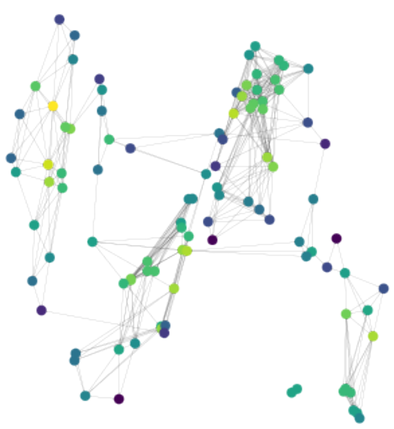
Theorem *(Continuous limit of GNNs towards c-GNNs)* [KBV '20]

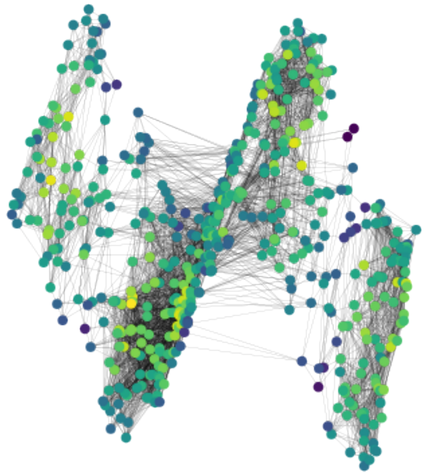
In the **relatively sparse** regime $\alpha_n \lesssim \frac{\log n}{n}$, with overwhelming probability,

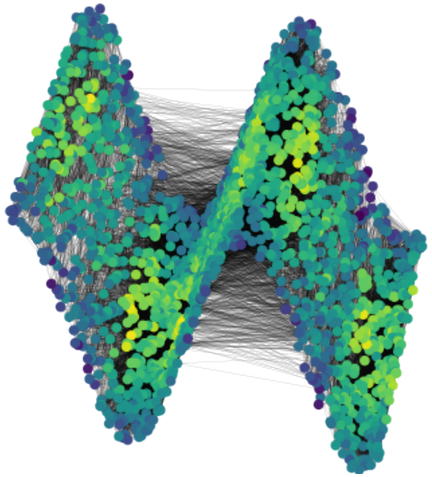
$$d(\Phi_G(z), \Phi_\Gamma(f)) \lesssim dn^{-1/2} + (\alpha_n n)^{-1/2}$$

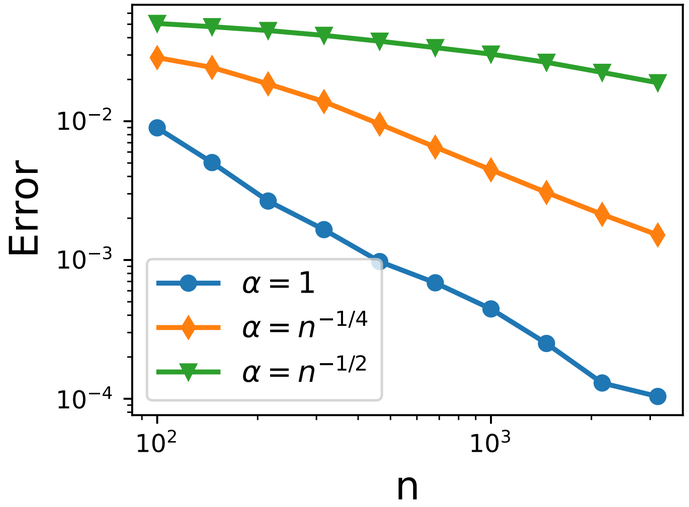
$\mathfrak{S}$-invariant: $d(\Phi_G(z), \Phi_\Gamma(f)) = \|\Phi_G(z) - \Phi_\Gamma(f)\|$

$\mathfrak{S}$-equivariant: $d(\Phi_G(z), \Phi_\Gamma(f)) = \left(\frac{1}{n}\sum_i \|\Phi_G(z)_i - \Phi_\Gamma(f)(x_i)\|^2\right)^{1/2}$









same limit,
different rate of convergence

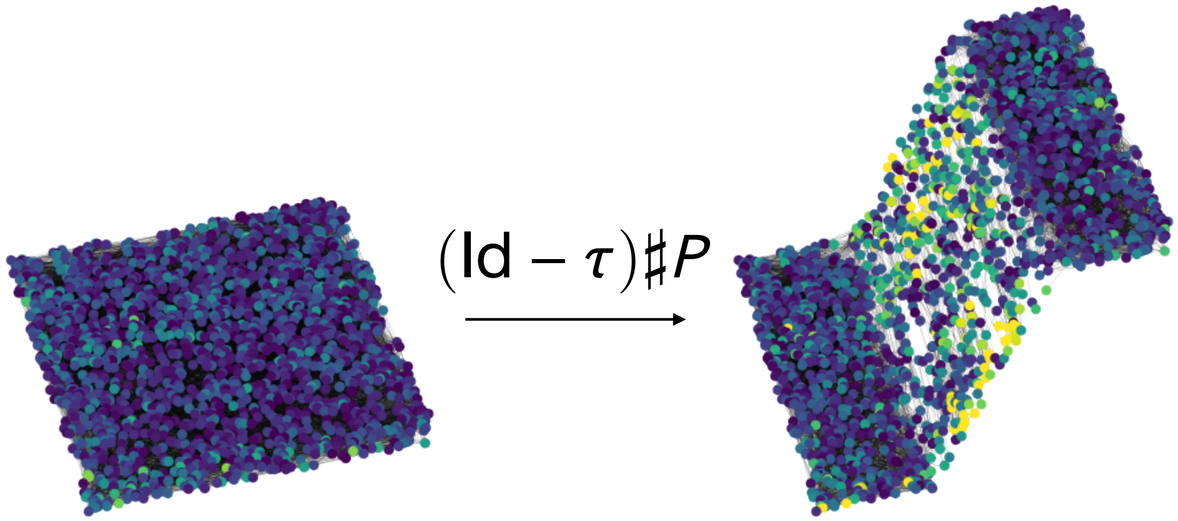
Stability to deformation

Theorem *(Deformation of a random graph model).* [KBV '20]

For translation-invariant kernels, if

- W is replaced by $W(x - \tau(x), x' - \tau(x'))$
- P is replaced by $(\mathrm{Id} - \tau)\sharp P$
- f is replaced by $f \circ (\mathrm{Id} - \tau)$

Then, the deviation of the cGNN is bounded by $\|\nabla\tau\|_\infty$



Conclusion

GNNs ❤️ Random graphs = large-scale properties

Not covered today:

— Universality

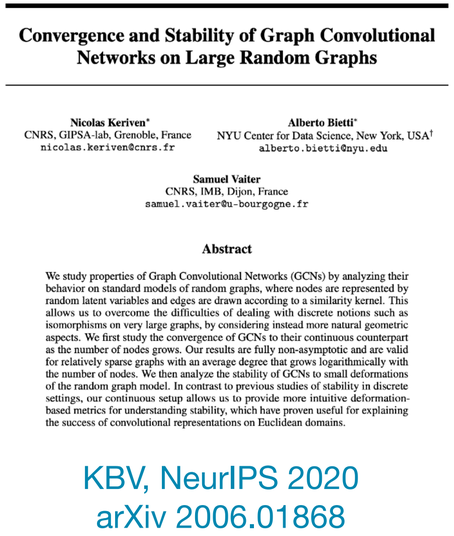
— Role of node features

Perspectives:

— Message passing cGNNs

— Sparse case

— Optimization/Generalization properties



Convergence and Stability of Graph Convolutional Networks on Large Random Graphs

Nicolas Keriven[*]
CNRS, GIPSA-lab, Grenoble, France
nicolas.keriven@cnrs.fr

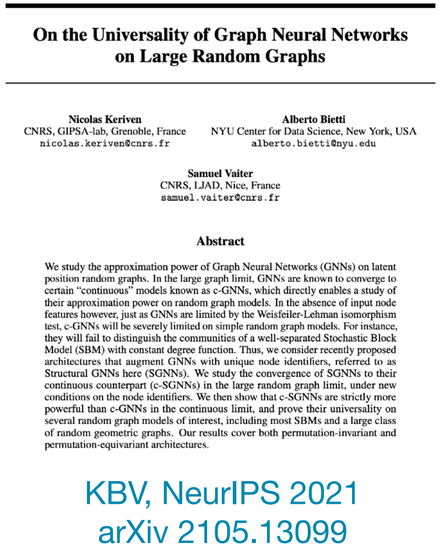
Alberto Bietti[*]
NYU Center for Data Science, New York, USA[†]
alberto.bietti@nyu.edu

Samuel Vaiter
CNRS, IMB, Dijon, France
samuel.vaiter@u-bourgogne.fr

Abstract

We study properties of Graph Convolutional Networks (GCNs) by analyzing their behavior on standard models of random graphs, where nodes are represented by random latent variables and edges are drawn according to a similarity kernel. This allows us to overcome the difficulties of dealing with discrete notions such as isomorphisms on very large graphs, by considering instead more natural geometric aspects. We first study the convergence of GCNs to their continuous counterpart as the number of nodes grows. Our results are fully non-asymptotic and are valid for relatively sparse graphs with an average degree that grows logarithmically with the number of nodes. We then analyze the stability of GCNs to small deformations of the random graph model. In contrast to previous studies of stability in discrete settings, our continuous setup allows us to provide more intuitive deformation-based metrics for understanding stability, which have proven useful for explaining the success of convolutional representations on Euclidean domains.

KBV, NeurIPS 2020
arXiv 2006.01868



On the Universality of Graph Neural Networks on Large Random Graphs

Nicolas Keriven
CNRS, GIPSA-lab, Grenoble, France
nicolas.keriven@cnrs.fr

Alberto Bietti
NYU Center for Data Science, New York, USA
alberto.bietti@nyu.edu

Samuel Vaiter
CNRS, LJAD, Nice, France
samuel.vaiter@cnrs.fr

Abstract

We study the approximation power of Graph Neural Networks (GNNs) on latent position random graphs. In the large graph limit, GNNs are known to converge to certain "continuous" models known as c-GNNs, which directly enables a study of their approximation power on random graph models. In the absence of input node features however, just as GNNs are limited by the Weisfeiler-Lehman isomorphism test, c-GNNs will be severely limited on simple random graph models. For instance, they will fail to distinguish the communities of a well-separated Stochastic Block Model (SBM) with constant degree function. Thus, we consider recently proposed architectures that augment GNNs with unique node identifiers, referred to as Structural GNNs here (SGNNs). We study the convergence of SGNNs to their continuous counterpart (c-SGNNs) in the large random graph limit, under new conditions on the node identifiers. We then show that c-SGNNs are strictly more powerful than c-GNNs in the continuous limit, and prove their universality on several random graph models of interest, including most SBMs and a large class of random geometric graphs. Our results cover both permutation-invariant and permutation-equivariant architectures.

KBV, NeurIPS 2021
arXiv 2105.13099