Graph Neural Networks on Large Random Graphs: Convergence, Stability, Universality

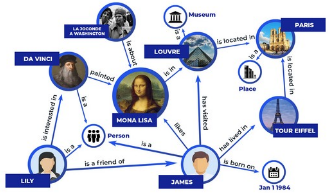
Nicolas Keriven

CNRS, GIPSA-lab

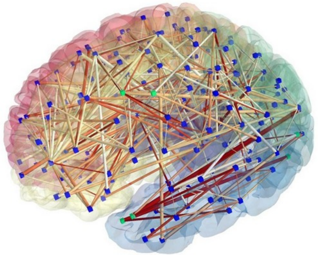
Joint work with Alberto Bietti (NYU) and Samuel Vaiter (CNRS, LJAD)

Graph Machine Learning

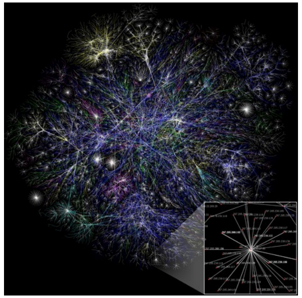
(Relatively) recent popularity of ML on graphs...



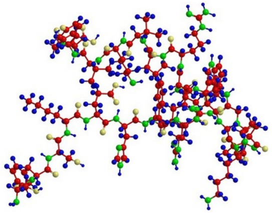
Knowledge graph



Brain connectivity network

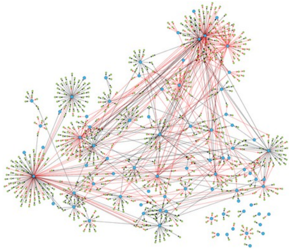


Internet

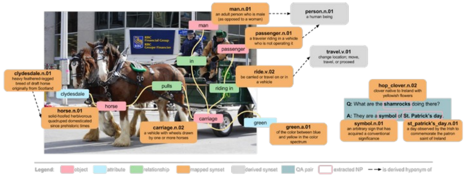


Molecule

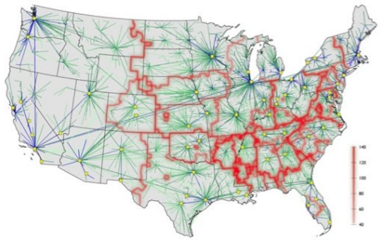
Social network



Gene regulatory network

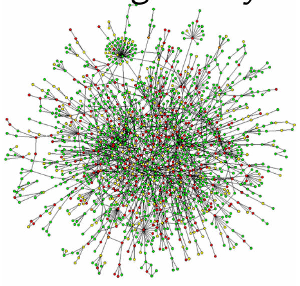


Scene understanding network

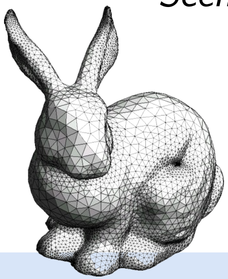


Transportation network

Computer network



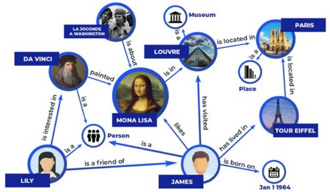
Protein interaction network



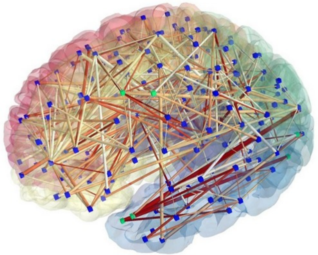
3D mesh

Graph Machine Learning

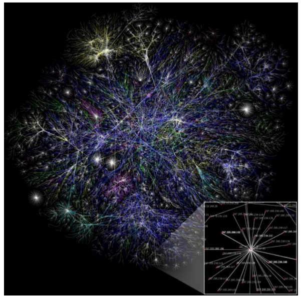
(Relatively) recent **popularity** of ML on graphs...



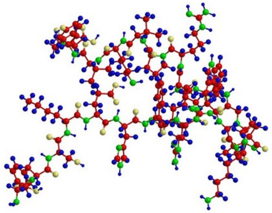
Knowledge graph



Brain connectivity network

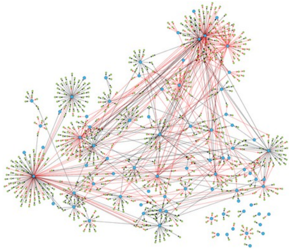


Internet

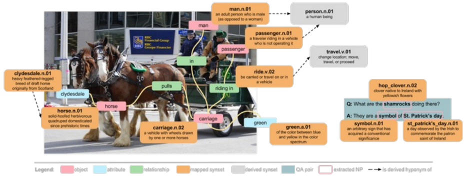


Molecule

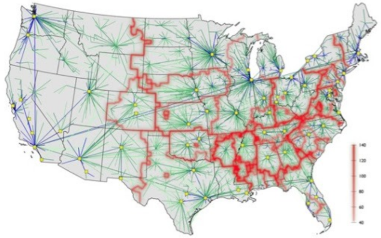
Social network



Gene regulatory network

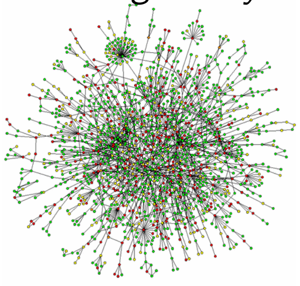


Scene understanding network

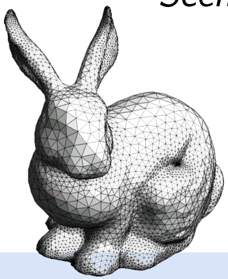


Transportation network

Computer network



Protein interaction network



3D mesh

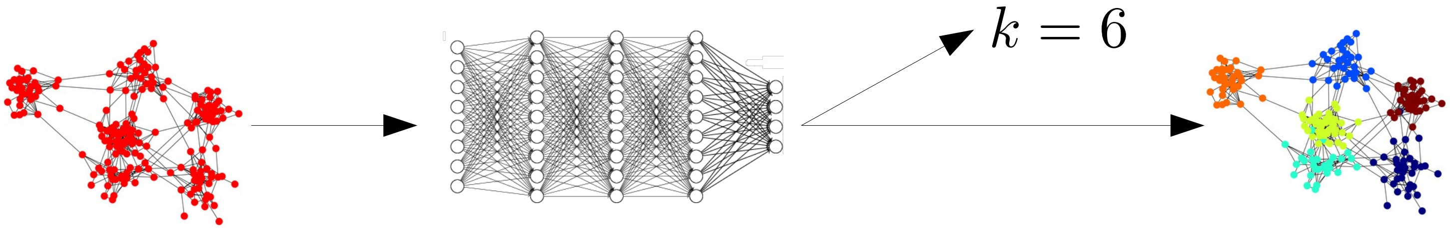
“if all you have is a hammer, everything looks like a nail”

ML on graphs: Graph Neural Networks

This talk: some **theoretical** properties of **Graph Neural Networks** on **large graphs.**

ML on graphs: Graph Neural Networks

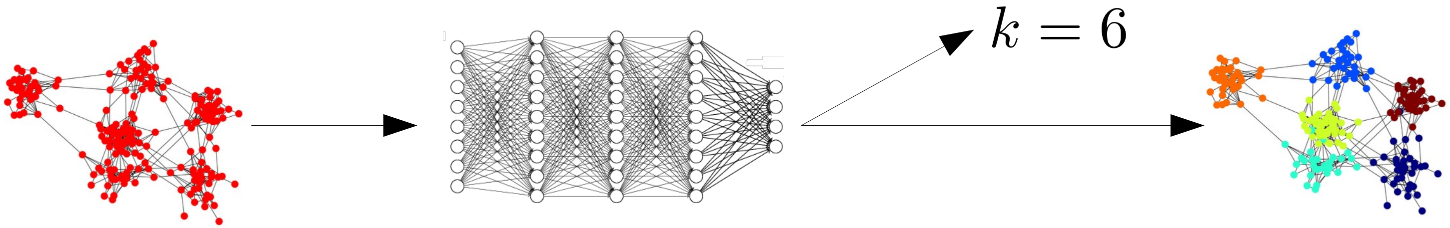
This talk: some **theoretical** properties of **Graph Neural Networks** on **large graphs.**



Graph Neural Networks (GNN) are "deep architectures" to do ML on graphs.

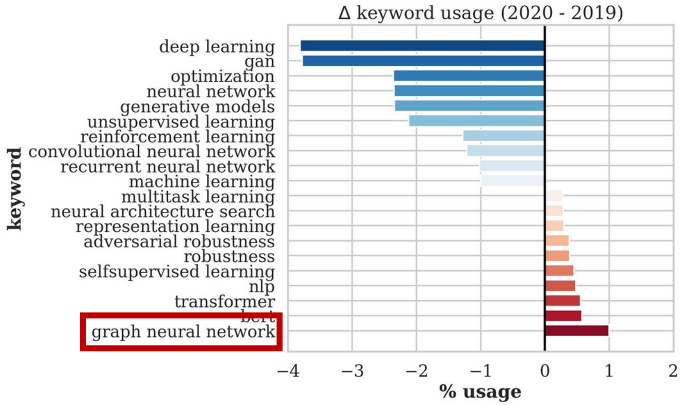
ML on graphs: Graph Neural Networks

This talk: some **theoretical** properties of **Graph Neural Networks** on **large graphs.**



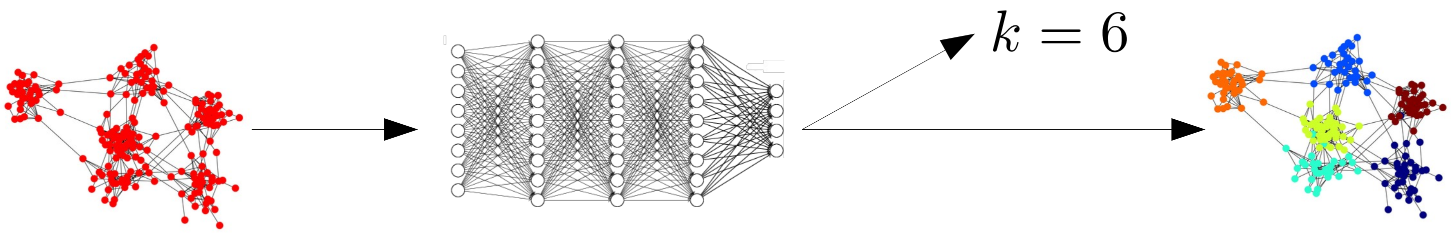
Graph Neural Networks (GNN) are "deep architectures" to do ML on graphs.

- Very (very) trendy right now!



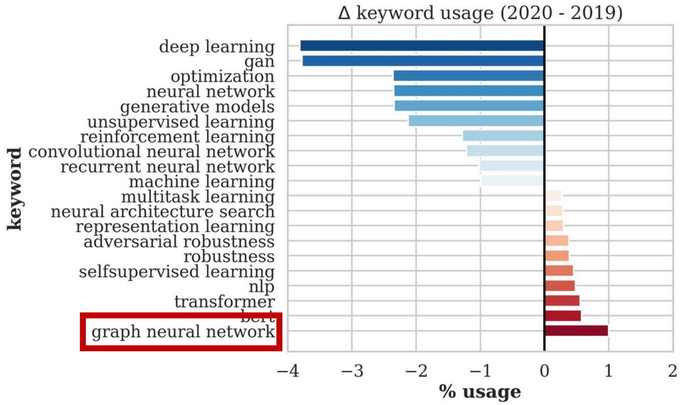
ML on graphs: Graph Neural Networks

This talk: some **theoretical** properties of **Graph Neural Networks** on large graphs.



Graph Neural Networks (GNN) are "deep architectures" to do ML on graphs.

- Very (very) trendy right now!
- Work quite well, but...
  - Room for improvement! (compared to other "deep learning")
  - No "ImageNet moment" yet for GNNs (see *Open Graph Benchmark*)
  - The theory might be **actually useful** to design new architectures



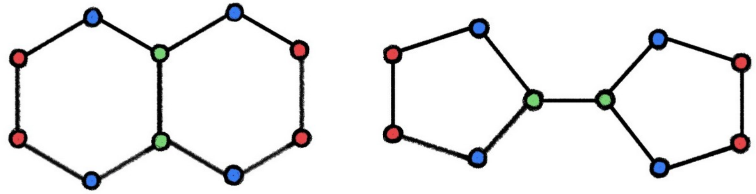
Large graphs?

- (Even) compared to regular NNs, many properties of GNNs are still quite mysterious.
  - Eg: **universality** of NNs is known since the 90s, for GNNs it is still a very active field.

Large graphs?

- (Even) compared to regular NNs, many properties of GNNs are still quite mysterious.
  - Eg: **universality** of NNs is known since the 90s, for GNNs it is still a very active field.

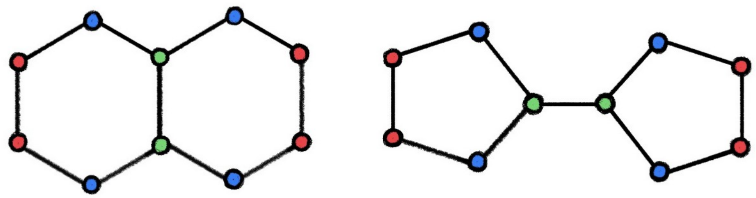
- Most analyses of GNNs are discrete in nature.
  - *Can a GNN distinguish two non-isomorphic graphs?*
  - *Can a GNN count triangles? compute the diameter of a graph? etc.*



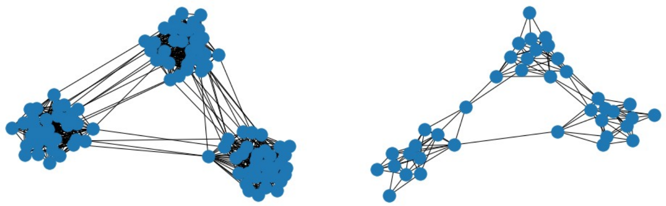
Large graphs?

- (Even) compared to regular NNs, many properties of GNNs are still quite mysterious.
  - Eg: **universality** of NNs is known since the 90s, for GNNs it is still a very active field.

- Most analyses of GNNs are discrete in nature.
  - *Can a GNN distinguish two non-isomorphic graphs?*
  - *Can a GNN count triangles? compute the diameter of a graph? etc.*



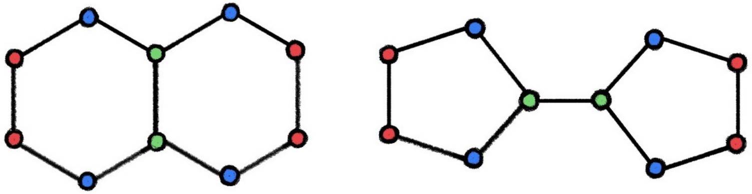
- Large graphs may "look the same", but are never isomorphic, of different size, etc.



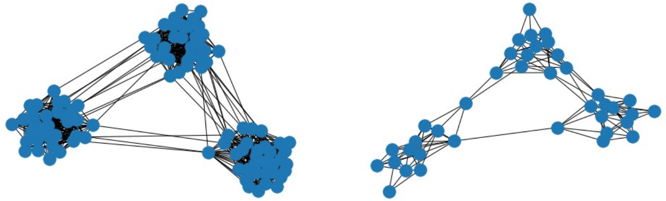
Large graphs?

- (Even) compared to regular NNs, many properties of GNNs are still quite mysterious.
  - Eg: **universality** of NNs is known since the 90s, for GNNs it is still a very active field.

- Most analyses of GNNs are discrete in nature.
  - *Can a GNN distinguish two non-isomorphic graphs?*
  - *Can a GNN count triangles? compute the diameter of a graph? etc.*



- Large graphs may "look the same", but are never isomorphic, of different size, etc.



This talk: use random graph models to analyze GNN properties on large graphs

Outline

1. Convergence of GNNs
2. Stability of c-GNNs
3. Universality of c-GNNs

Random graphs models

Long history of modelling large graphs with **random generative models**

Chung and Lu. *Complex Graphs and Networks* (2004)
Penrose. *Random Geometric Graphs* (2008)
Lovasz. *Large networks and graph limits* (2012)
Frieze and Karonski. *Introduction to random graphs* (2016)

Random graphs models

Long history of modelling large graphs with **random generative models**

Chung and Lu. *Complex Graphs and Networks* (2004)
Penrose. *Random Geometric Graphs* (2008)
Lovasz. *Large networks and graph limits* (2012)
Frieze and Karonski. *Introduction to random graphs* (2016)

Latent position models *(W-random graphs, kernel random graphs...)*

$x_i \overset{iid}{\sim} P \in \mathbb{R}^d \qquad a_{ij} \sim \mathrm{Ber}(\alpha_n W(x_i, x_j))$

Unknown *latent variables* — *Connectivity kernel*

Random graphs models

Long history of modelling large graphs with **random generative models**

Chung and Lu. *Complex Graphs and Networks* (2004)
Penrose. *Random Geometric Graphs* (2008)
Lovasz. *Large networks and graph limits* (2012)
Frieze and Karonski. *Introduction to random graphs* (2016)

Latent position models *(W-random graphs, kernel random graphs...)*

$$x_i \overset{iid}{\sim} P \in \mathbb{R}^d \qquad a_{ij} \sim \mathrm{Ber}(\alpha_n W(x_i, x_j))$$

Unknown *latent variables* *Connectivity kernel*

Dense $\alpha_n \sim 1$ *Sparse* $\alpha_n \sim 1/n$

Relatively sparse $\alpha_n \sim (\log n)/n$

Random graphs models

Long history of modelling large graphs with **random generative models**

Chung and Lu. *Complex Graphs and Networks* (2004)
Penrose. *Random Geometric Graphs* (2008)
Lovasz. *Large networks and graph limits* (2012)
Frieze and Karonski. *Introduction to random graphs* (2016)

Latent position models *(W-random graphs, kernel random graphs...)*

$x_i \overset{iid}{\sim} P \in \mathbb{R}^d$ — ***Unknown*** *latent variables*

$a_{ij} \sim \mathrm{Ber}(\alpha_n W(x_i, x_j))$ — *Connectivity kernel*

$z_i = f(x_i)$ — *Node features (opt.)*

Dense $\alpha_n \sim 1$ *Sparse* $\alpha_n \sim 1/n$

Relatively sparse $\alpha_n \sim (\log n)/n$

Random graphs models

Long history of modelling large graphs with **random generative models**

Chung and Lu. *Complex Graphs and Networks* (2004)
Penrose. *Random Geometric Graphs* (2008)
Lovasz. *Large networks and graph limits* (2012)
Frieze and Karonski. *Introduction to random graphs* (2016)

Latent position models *(W-random graphs, kernel random graphs...)*

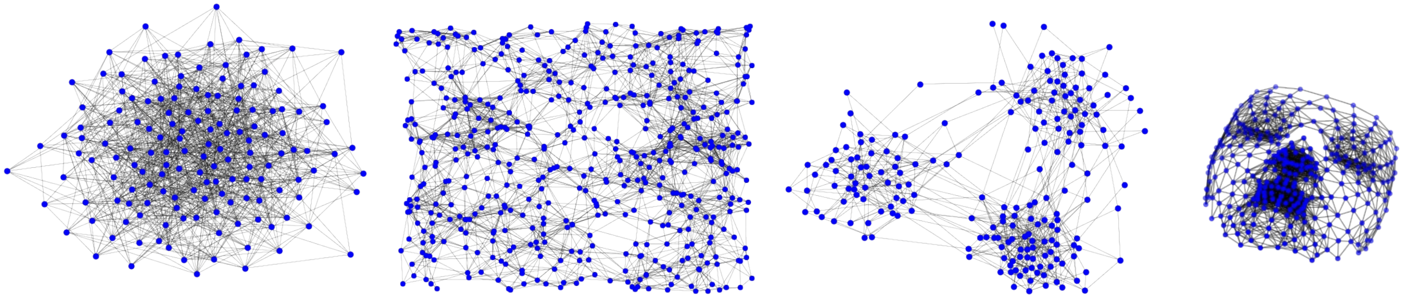
$x_i \overset{iid}{\sim} P \in \mathbb{R}^d$ — ***Unknown*** *latent variables*

$a_{ij} \sim \mathrm{Ber}(\alpha_n W(x_i, x_j))$ — *Connectivity kernel*

$z_i = f(x_i)$ — *Node features (opt.)*

Dense $\alpha_n \sim 1$ *Sparse* $\alpha_n \sim 1/n$

Relatively sparse $\alpha_n \sim (\log n)/n$

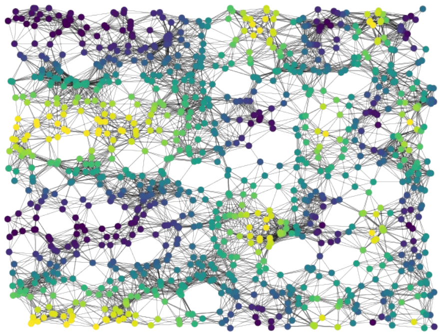


Includes Erdös-Rényi, Stochastic Block Models, Gaussian kernel, epsilon-graphs...

Filering on graphs

(Early-days) GNNs are based on **graph-convolutions** (filtering)

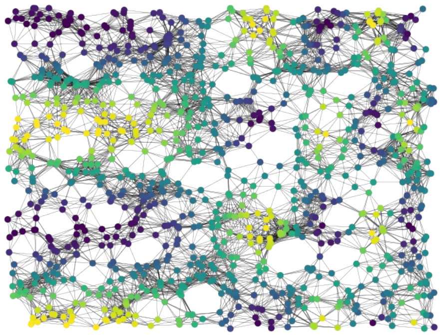
Bruna et al. *Spectral Networks and Locally Connected Networks on Graphs* (2013)
Bronstein et al. *Geometric Deep Learning* (2017)



Filtering on graphs

(Early-days) GNNs are based on **graph-convolutions** (filtering)

Bruna et al. *Spectral Networks and Locally Connected Networks on Graphs* (2013)
Bronstein et al. *Geometric Deep Learning* (2017)

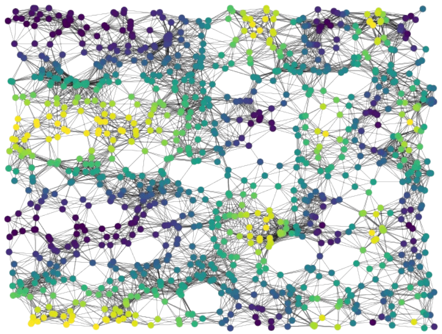


- Based on graph Fourier transform

Filtering on graphs

(Early-days) GNNs are based on **graph-convolutions** (filtering)

Bruna et al. *Spectral Networks and Locally Connected Networks on Graphs* (2013)
Bronstein et al. *Geometric Deep Learning* (2017)



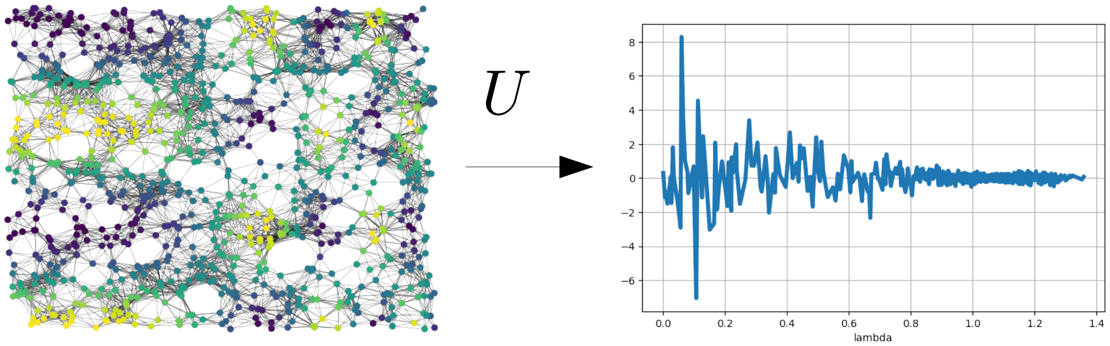
- Based on graph Fourier transform
- Defined by diagonalizing the graph Laplacian $L = Id - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = U^\top \Lambda U$

Filtering on graphs

(Early-days) GNNs are based on **graph-convolutions** (filtering)

Bruna et al. *Spectral Networks and Locally Connected Networks on Graphs* (2013)
Bronstein et al. *Geometric Deep Learning* (2017)

- Based on graph Fourier transform
- Defined by diagonalizing the graph Laplacian $L = Id - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U^\top \Lambda U$

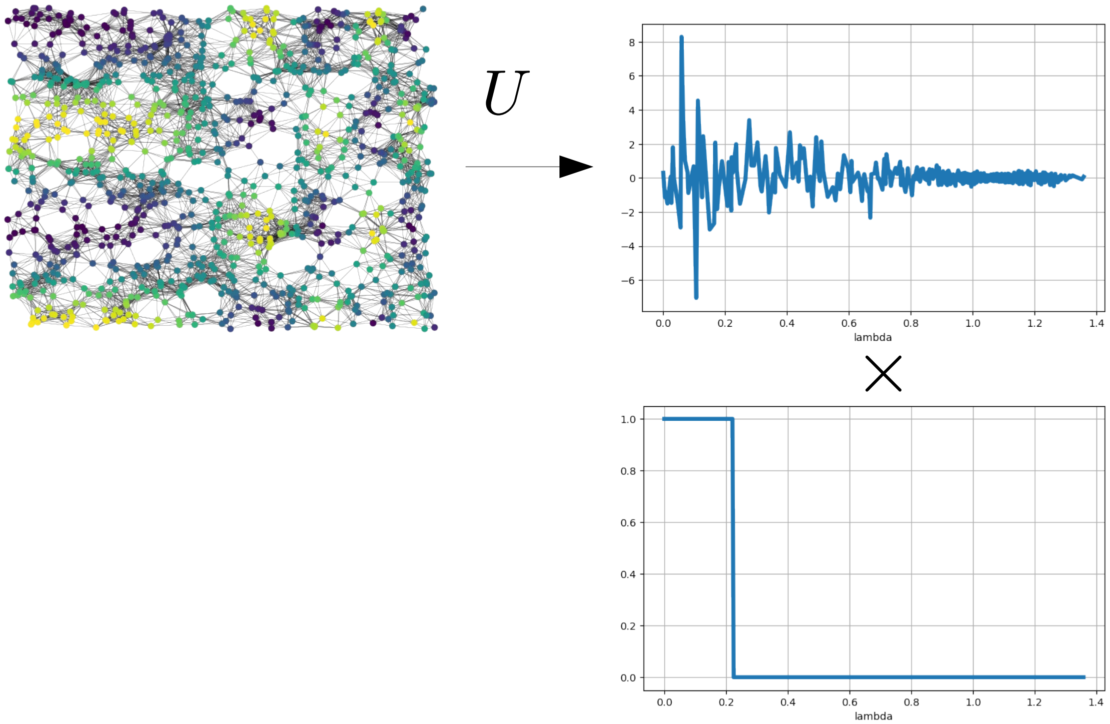


Filtering on graphs

(Early-days) GNNs are based on **graph-convolutions** (filtering)

Bruna et al. *Spectral Networks and Locally Connected Networks on Graphs* (2013)
Bronstein et al. *Geometric Deep Learning* (2017)

- Based on graph Fourier transform
- Defined by diagonalizing the graph Laplacian $L = Id - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U^\top \Lambda U$

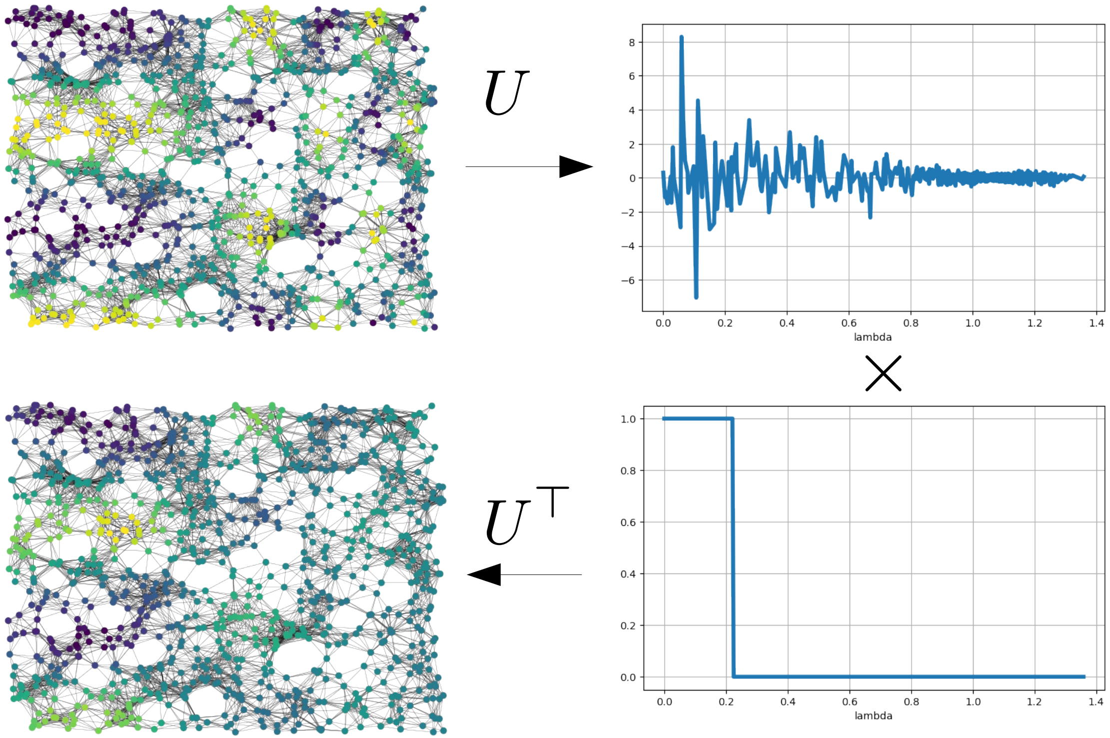


Filtering on graphs

(Early-days) GNNs are based on **graph-convolutions** (filtering)

Bruna et al. *Spectral Networks and Locally Connected Networks on Graphs* (2013)
Bronstein et al. *Geometric Deep Learning* (2017)

- Based on graph Fourier transform
- Defined by diagonalizing the graph Laplacian $L = Id - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U^\top \Lambda U$



Chung. *Spectral Graph Theory.* (1999)
Shuman et al. *The Emerging Field of Signal Processing on Graphs.* (2013)
Defferrard et al. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering* (2016)

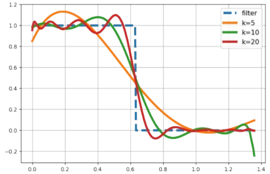
Filtering on graphs

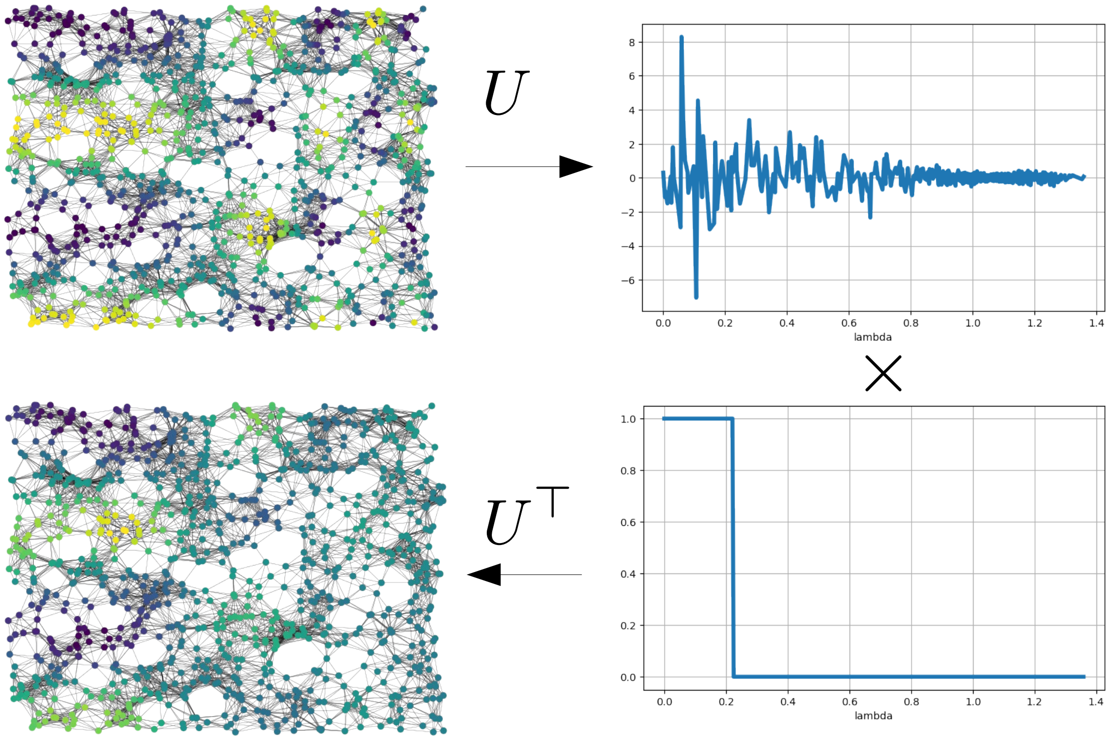
(Early-days) GNNs are based on **graph-convolutions** (filtering)

Bruna et al. *Spectral Networks and Locally Connected Networks on Graphs* (2013)
Bronstein et al. *Geometric Deep Learning* (2017)

- Based on graph Fourier transform
- Defined by diagonalizing the graph Laplacian $L = Id - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U^\top \Lambda U$
- Popular filters are polynomial filters

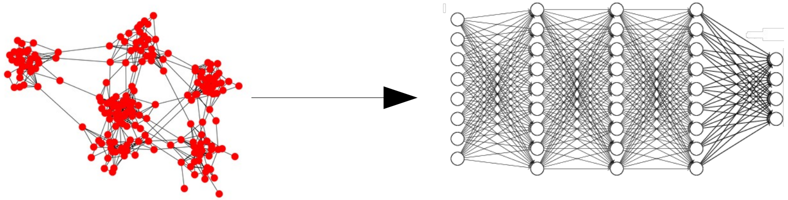
$$h \star z = \left(\sum_k \beta_k L^k\right) z$$





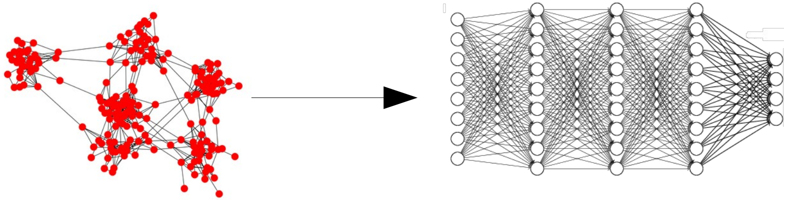
Chung. *Spectral Graph Theory.* (1999)
Shuman et al. *The Emerging Field of Signal Processing on Graphs.* (2013)
Defferrard et al. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering* (2016)

Discrete vs. continuous



(Spectral) Graph Neural Networks

Discrete vs. continuous

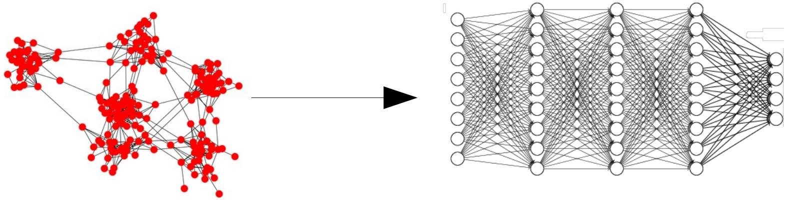


(Spectral) Graph Neural Networks

- Propagate **signal over nodes**

$$z_j^{(\ell+1)} = \rho\left(\sum\nolimits_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

Discrete vs. continuous



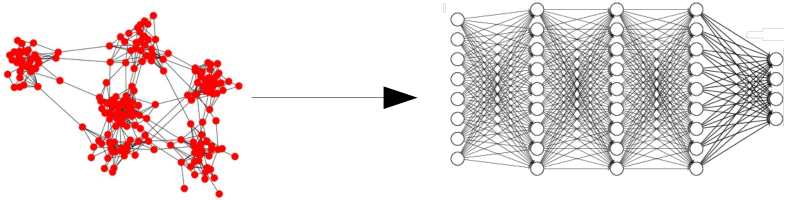
(Spectral) Graph Neural Networks

- Propagate **signal over nodes**

$$z_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

Trainable polynomial graph filters with normalized Laplacian $h(L) = \sum_k \beta_k L^k$, $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

Discrete vs. continuous



(Spectral) Graph Neural Networks

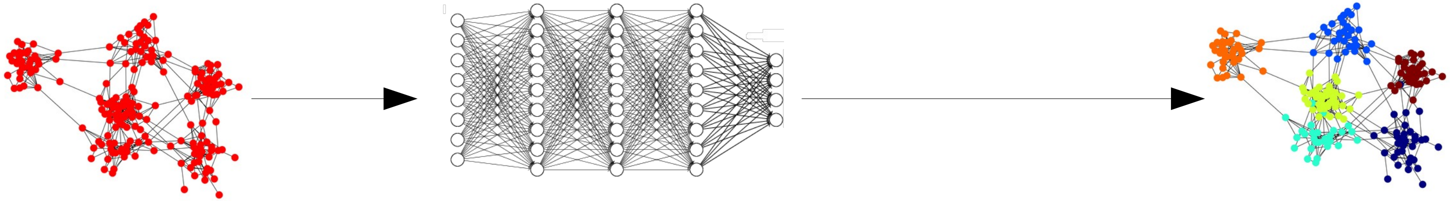
- Propagate **signal over nodes**

$$z_j^{(\ell+1)} = \rho\left(\sum\nolimits_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

ReLU

Trainable polynomial graph filters with normalized Laplacian $h(L) = \sum_k \beta_k L^k$, $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

Discrete vs. continuous



(Spectral) Graph Neural Networks

- Propagate **signal over nodes**

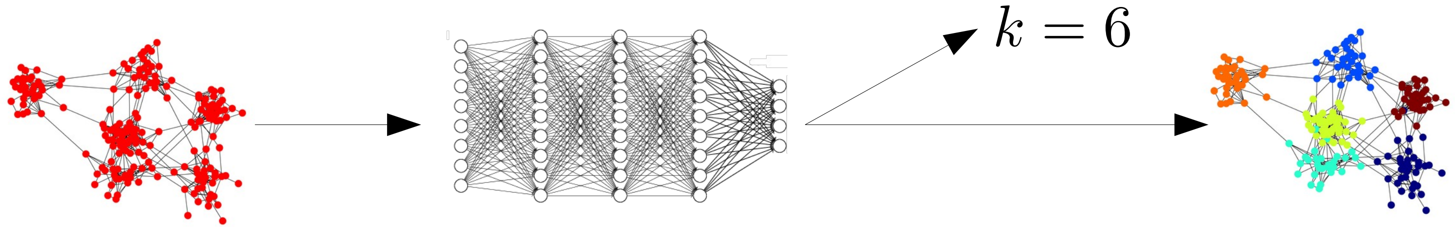
$$z_j^{(\ell+1)} = \underset{\text{ReLU}}{\rho} \left(\sum_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n \right)$$

Trainable polynomial graph filters with normalized Laplacian $h(L) = \sum_k \beta_k L^k$ $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

Output

- Signal over nodes (permutation-equivariant)

Discrete vs. continuous



(Spectral) Graph Neural Networks

- Propagate **signal over nodes**

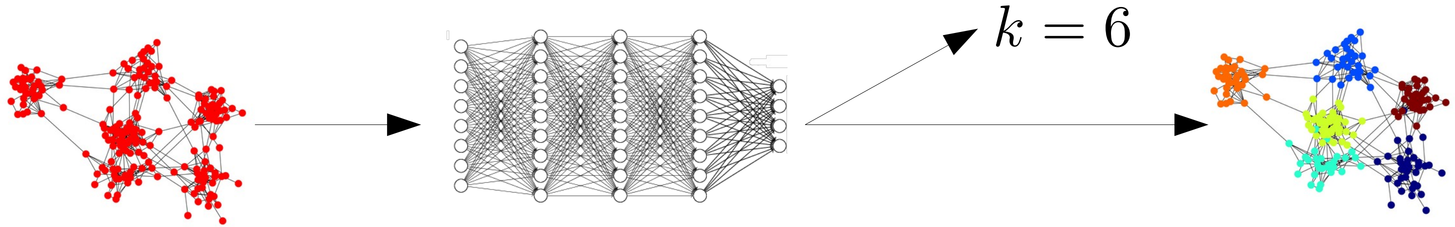
$$z_j^{(\ell+1)} = \underset{\text{ReLU}}{\rho}\left(\sum_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

Trainable polynomial graph filters with normalized Laplacian $h(L) = \sum_k \beta_k L^k$ $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

Output

- Signal over nodes (permutation-equivariant)
- Single vector with pooling (permutation-invariant)

Discrete vs. continuous



(Spectral) Graph Neural Networks

- Propagate **signal over nodes**

$$z_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(L)\, z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

ReLU

Trainable polynomial graph filters with normalized Laplacian $h(L) = \sum_k \beta_k L^k$, $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

Output

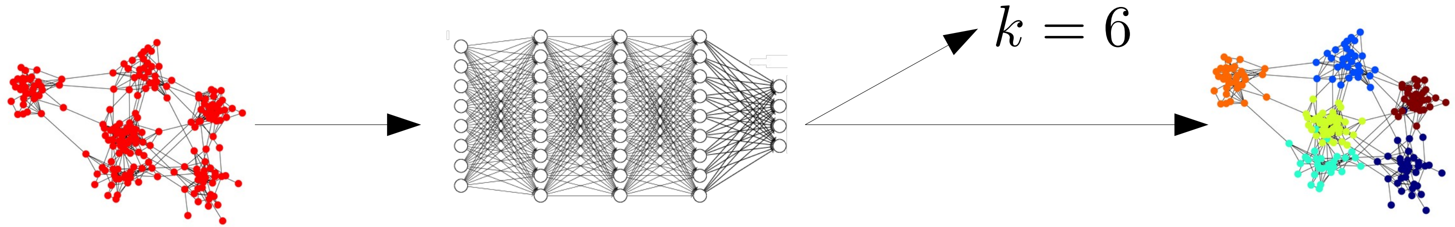
- Signal over nodes (permutation-equivariant)
- Single vector with pooling (permutation-invariant)

Continuous Graph Neural Networks

- Propagate **function over latent space**

$$f_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(\mathcal{L})\, f_i^{(\ell)} + b_j^{(\ell)}\right)$$

Discrete vs. continuous



(Spectral) Graph Neural Networks

- Propagate **signal over nodes**

$$z_j^{(\ell+1)} = \rho\left(\sum\nolimits_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

ReLU

Trainable polynomial graph filters with normalized Laplacian $h(L) = \sum_k \beta_k L^k$ $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

Output

- Signal over nodes (permutation-equivariant)
- Single vector with pooling (permutation-invariant)

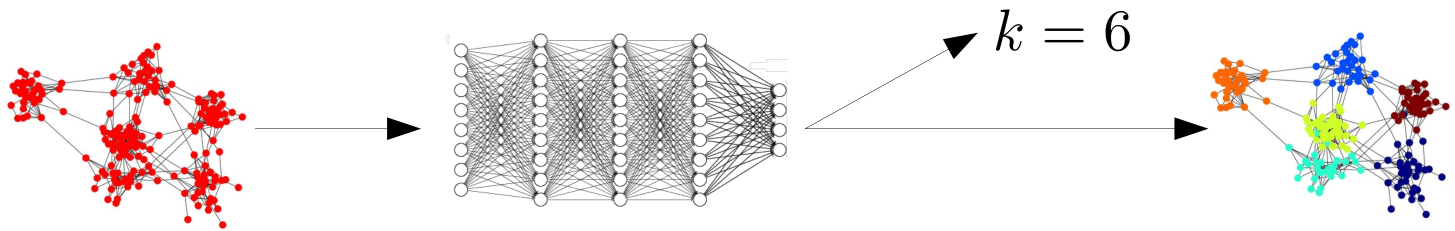
Continuous Graph Neural Networks

- Propagate **function over latent space**

$$f_j^{(\ell+1)} = \rho\left(\sum\nolimits_i h_{ij}^{(\ell)}(\mathcal{L}) f_i^{(\ell)} + b_j^{(\ell)}\right)$$

Filters with normalized Laplacian **operator** $\mathcal{L} f = \int \frac{W(\cdot, x)}{\sqrt{d(\cdot) d(x)}} f(x) dP(x)$

Discrete vs. continuous



(Spectral) Graph Neural Networks

- Propagate **signal over nodes**

$$z_j^{(\ell+1)} = \rho\left(\sum\nolimits_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

ReLU

Trainable polynomial graph filters with normalized Laplacian $h(L) = \sum_k \beta_k L^k$, $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

Output

- Signal over nodes (permutation-equivariant)
- Single vector with pooling (permutation-invariant)

Continuous Graph Neural Networks

- Propagate **function over latent space**

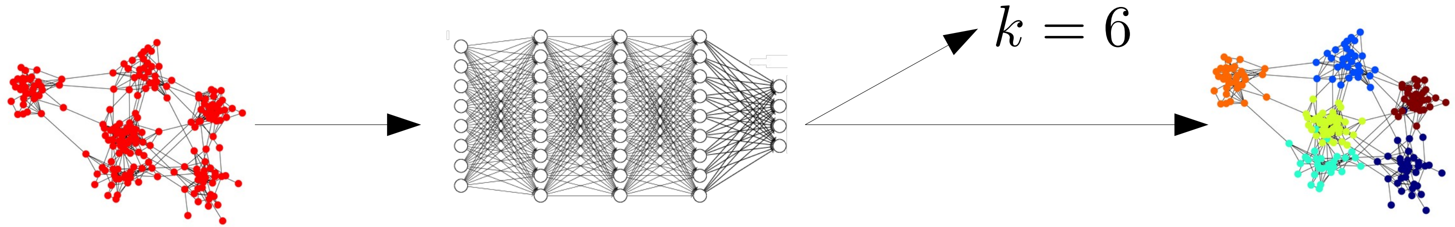
$$f_j^{(\ell+1)} = \rho\left(\sum\nolimits_i h_{ij}^{(\ell)}(\mathcal{L}) f_i^{(\ell)} + b_j^{(\ell)}\right)$$

Filters with normalized Laplacian **operator** $\mathcal{L}f = \int \frac{W(\cdot, x)}{\sqrt{d(\cdot)d(x)}} f(x) dP(x)$

Output

- Function ("continuous" permutation-equivariant)
- Vector ("continuous" permutation-invariant)

Discrete vs. continuous



Same parameters!

(Spectral) Graph Neural Networks

- Propagate **signal over nodes**

$$z_j^{(\ell+1)} = \rho\left(\sum\nolimits_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

ReLU

Trainable polynomial graph filters with normalized Laplacian $h(L) = \sum_k \beta_k L^k$ $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

Output

- Signal over nodes (permutation-equivariant)
- Single vector with pooling (permutation-invariant)

Continuous Graph Neural Networks

- Propagate **function over latent space**

$$f_j^{(\ell+1)} = \rho\left(\sum\nolimits_i h_{ij}^{(\ell)}(\mathcal{L}) f_i^{(\ell)} + b_j^{(\ell)}\right)$$

Filters with normalized Laplacian **operator** $\mathcal{L} f = \int \frac{W(\cdot, x)}{\sqrt{d(\cdot) d(x)}} f(x) dP(x)$

Output

- Function ("continuous" permutation-equivariant)
- Vector ("continuous" permutation-invariant)

Continuous limit of GNNs

Thm (Non-asymptotic convergence)

If $\alpha_n \gtrsim (\log n)/n$ *, with probability* $1 - n^{-r}$*, the deviation between the outputs of the discrete GNN and the continuous GNN is*

Perm-inv	$\|\Phi_G(Z) - \Phi_{W,P}(f)\|$	$\boxed{\lesssim \frac{d}{\sqrt{n}} + \frac{1}{\sqrt{\alpha_n n}}}$
Perm-equi	$\left(\frac{1}{n}\sum_i \|\Phi_G(Z)_i - \Phi_{W,P}(f)(x_i)\|^2\right)^{1/2}$	

Continuous limit of GNNs

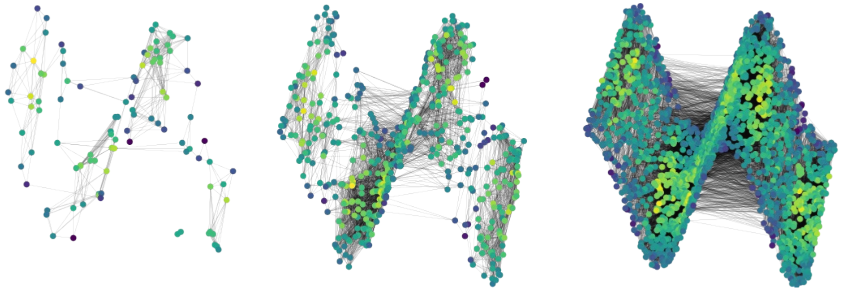
Thm (Non-asymptotic convergence)

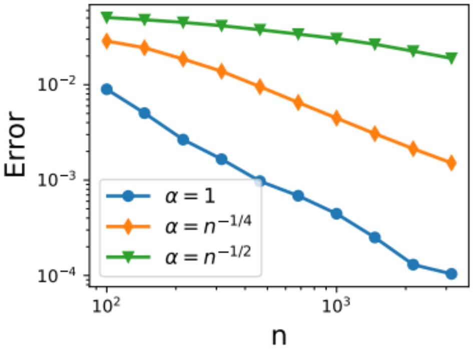
If $\alpha_n \gtrsim (\log n)/n$ *, with probability* $1 - n^{-r}$*, the deviation between the outputs of the discrete GNN and the continuous GNN is*

Perm-inv $\quad \|\Phi_G(Z) - \Phi_{W,P}(f)\|$

Perm-equi $\quad \left(\frac{1}{n}\sum_i \|\Phi_G(Z)_i - \Phi_{W,P}(f)(x_i)\|^2\right)^{1/2}$

$$\boxed{\lesssim \frac{d}{\sqrt{n}} + \frac{1}{\sqrt{\alpha_n n}}}$$





Continuous limit of GNNs

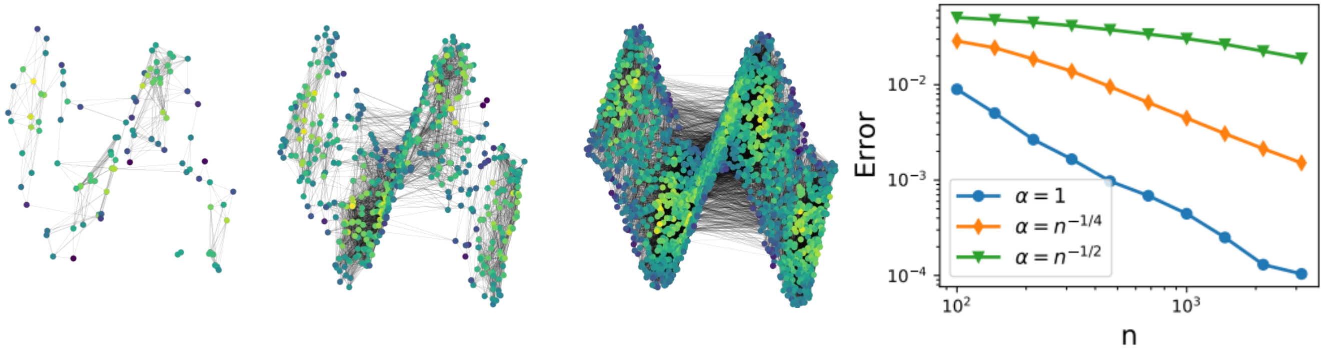
Thm (Non-asymptotic convergence)

If $\alpha_n \gtrsim (\log n)/n$ *, with probability* $1 - n^{-r}$*, the deviation between the outputs of the discrete GNN and the continuous GNN is*

Perm-inv $\quad \|\Phi_G(Z) - \Phi_{W,P}(f)\|$

Perm-equi $\quad \left(\frac{1}{n}\sum_i \|\Phi_G(Z)_i - \Phi_{W,P}(f)(x_i)\|^2\right)^{1/2}$

$$\lesssim \frac{d}{\sqrt{n}} + \frac{1}{\sqrt{\alpha_n n}}$$



- Thanks to normalized Laplacian, the limit does ***not*** depend on α_n but the rate of convergence does...
- Could have used normalized adjacency $A/(n\alpha_n)$ with operator $\int W(x,y)f(y)dP(y)$

Outline

1. Convergence of GNNs

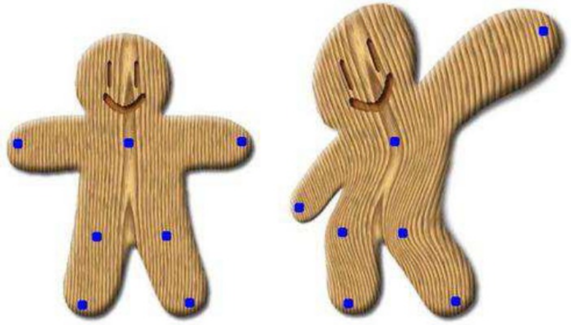
2. Stability of c-GNNs

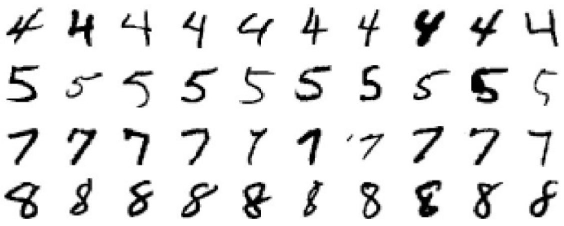
3. Universality of c-GNNs

Large graphs?

- **CNN** (translation-invariant) are robust to spatial deformations

$$\|\Phi(f) - \Phi(f \circ (Id - \tau))\| \leq \|\nabla\tau\|_\infty$$



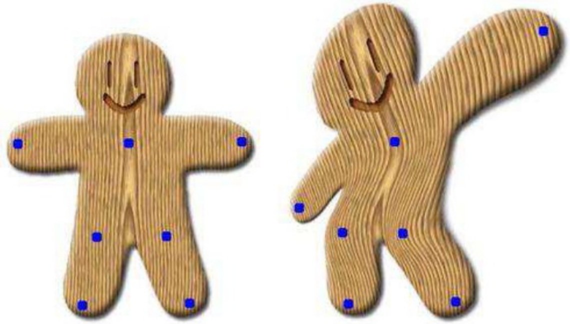


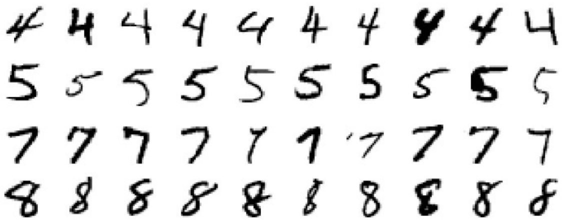
Mallat. *Group Invariant Scattering* (2012)
Bruna and Mallat. *Classification with scattering operators* (2013)
Bietti and Mairal. *Group invariance, stability to deformations, and complexity of deep convolutional representations* (2019)

Large graphs?

- **CNN** (translation-invariant) are robust to spatial deformations

$$\|\Phi(f) - \Phi(f \circ (Id - \tau))\| \leq \|\nabla \tau\|_\infty$$





Mallat. *Group Invariant Scattering* (2012)
Bruna and Mallat. *Classification with scattering operators* (2013)
Bietti and Mairal. *Group invariance, stability to deformations, and complexity of deep convolutional representations* (2019)

- **GNN**: stability to discrete graph metrics

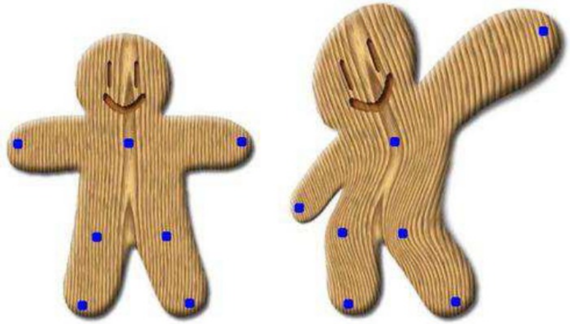
$$\|\Phi_G(Z) - \Phi_{G'}(Z)\| \leq d(G, G')$$

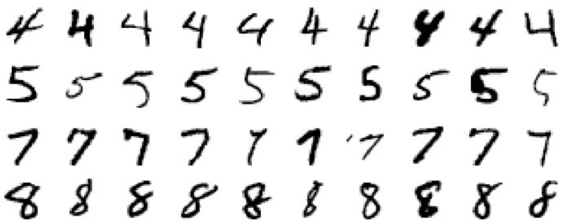
Gama et al. *Stability Properties of Graph Neural Networks* (2020)

Large graphs?

- **CNN** (translation-invariant) are robust to spatial deformations

$$\|\Phi(f) - \Phi(f \circ (Id - \tau))\| \leq \|\nabla \tau\|_\infty$$





Mallat. *Group Invariant Scattering* (2012)
Bruna and Mallat. *Classification with scattering operators* (2013)
Bietti and Mairal. *Group invariance, stability to deformations, and complexity of deep convolutional representations* (2019)

- **GNN**: stability to discrete graph metrics

$$\|\Phi_G(Z) - \Phi_{G'}(Z)\| \leq d(G, G')$$

Gama et al. *Stability Properties of Graph Neural Networks* (2020)

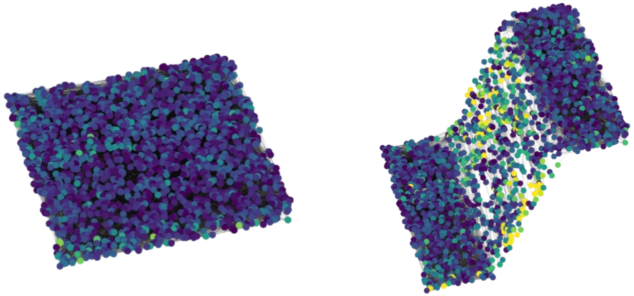
  - *Difficult to interpret, difficult to define for different-sized graphs*
  - *What's a meaningful notion of deformation for a graph?*

Stability of continuous GNNs

Continuous domain allows to define **intuitive geometric deformations**

Stability of continuous GNNs

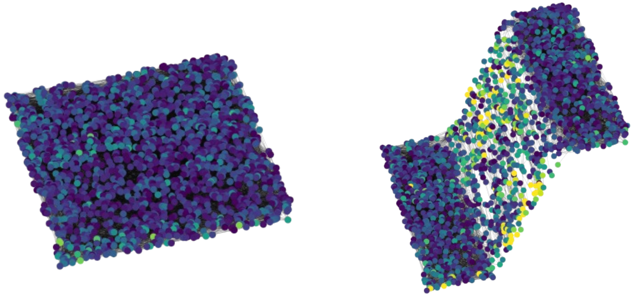
Continuous domain allows to define **intuitive geometric deformations**



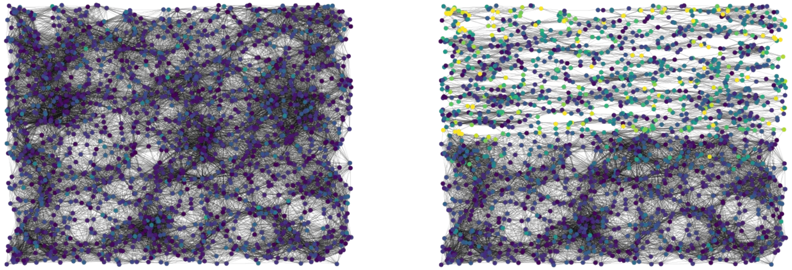
Deformation of distribution

Stability of continuous GNNs

Continuous domain allows to define **intuitive geometric deformations**



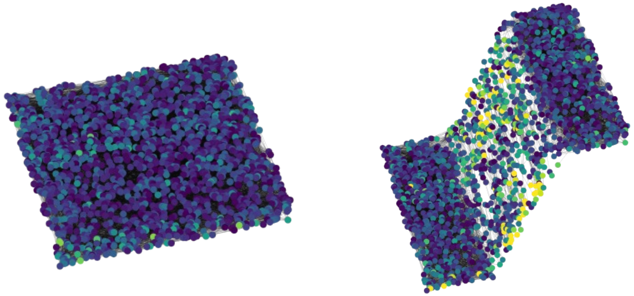
Deformation of distribution



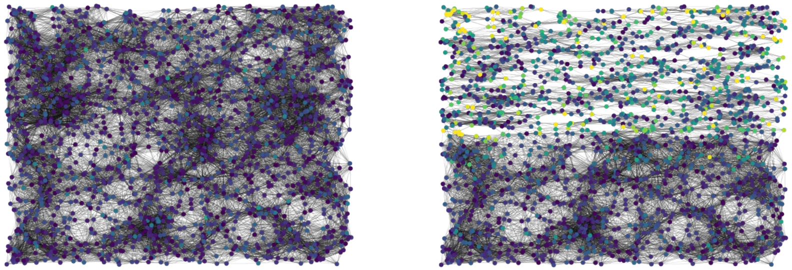
Deformation of kernel

Stability of continuous GNNs

Continuous domain allows to define **intuitive geometric deformations**



Deformation of distribution



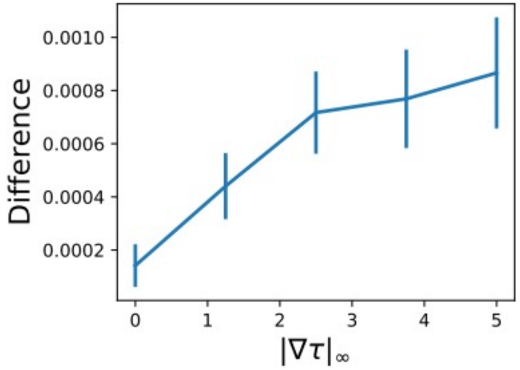
Deformation of kernel

Thm (Stability, simplified)

For *translation-invariant* kernels, if:

- W is replaced by $W(x-\tau(x), x'-\tau(x'))$
- P is replaced by $(Id-\tau)\sharp P$ (and f is translated)
- f is replaced by $f\circ(Id-\tau)$

Then, the deviation of c-GNN is bounded by $\|\nabla\tau\|_\infty$



Outline

1. Convergence of GNNs

2. Stability of c-GNNs

3. Universality of c-GNNs

GNN vs. WL

- Assume no node feature. Basic strategy: input constant $\Phi_G(1)$

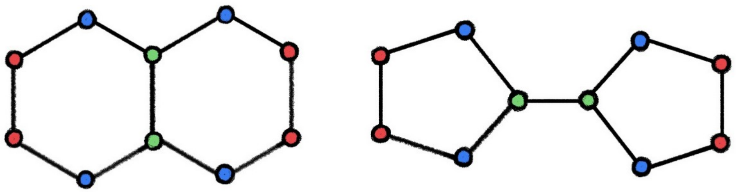
GNN vs. WL

- Assume no node feature. Basic strategy: input constant $\Phi_G(1)$
- Are GNNs universal on graph structures? Can they distinguish *non-isomorphic graphs*?

GNN vs. WL

- Assume no node feature. Basic strategy: input constant $\Phi_G(1)$
- Are GNNs universal on graph structures? Can they distinguish *non-isomorphic graphs*?
- A classical algorithm for graph isomorphism is the Weisfeiler-Lehman test.
  - Works by propagating labels with injective message-passing function
  - Sometimes yields false positive

WL fails here...



Weisfeiler and Lehman. *A reduction of a graph to a canonical form and an algebra arising during this reduction* (1968)

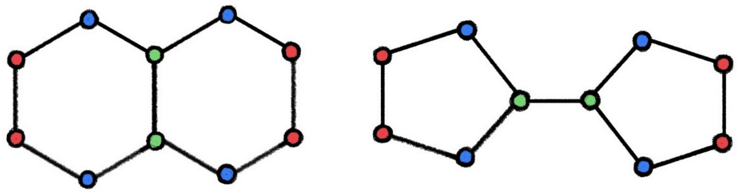
Babai and Kucera. *Canonical labelling of graphs in linear average time* (1979)

GNN vs. WL

- Assume no node feature. Basic strategy: input constant $\Phi_G(1)$
- Are GNNs universal on graph structures? Can they distinguish *non-isomorphic graphs*?
- A classical algorithm for graph isomorphism is the Weisfeiler-Lehman test.
  - Works by propagating labels with injective message-passing function
  - Sometimes yields false positive

Weisfeiler and Lehman. *A reduction of a graph to a canonical form and an algebra arising during this reduction* (1968)
Babai and Kucera. *Canonical labelling of graphs in linear average time* (1979)

WL fails here…



By construction, message-passing GNNs are not more powerful than WL test, and can be as powerful if the message-passing function is injective (sufficient number of neurons).

Xu et al. *How Powerful are Graph Neural Networks?* (2019)

Beyond WL

Going "beyond WL"...

Beyond WL

Going "beyond WL"...

- Using *higher-order tensors*
  - *Up until true universality!*

Maron et al. *Provably Powerful Graph Networks* (2019)

Maron et al. *On the Universality of Invariant Networks* (2019)

Keriven and Peyré. *Universal Invariant and Equivariant Graph Neural Networks* (2020)

Beyond WL

Going "beyond WL"...

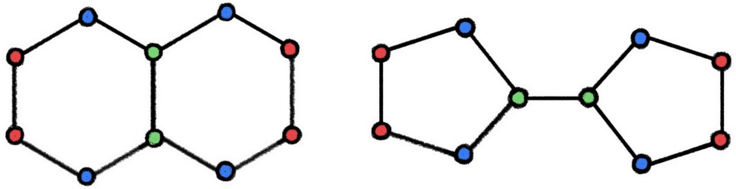
- Using *higher-order tensors*
  - *Up until true universality!*

Maron et al. *Provably Powerful Graph Networks* (2019)

Maron et al. *On the Universality of Invariant Networks* (2019)

Keriven and Peyré. *Universal Invariant and Equivariant Graph Neural Networks* (2020)

- Using higher-order *subgraph counting*



Morris et al. *Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks* (2019)

Bouritsas et al. *Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting* (2020)

Beyond WL

Going "beyond WL"...

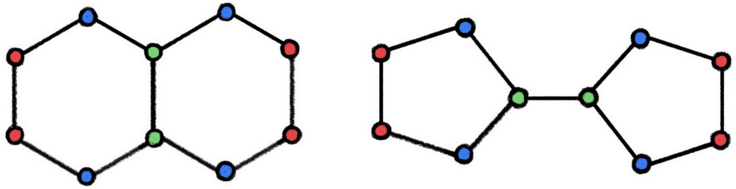
- Using *higher-order tensors*
  - *Up until true universality!*

Maron et al. *Provably Powerful Graph Networks* (2019)

Maron et al. *On the Universality of Invariant Networks* (2019)

Keriven and Peyré. *Universal Invariant and Equivariant Graph Neural Networks* (2020)

- Using higher-order *subgraph counting*



Morris et al. *Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks* (2019)

Bouritsas et al. *Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting* (2020)

- Using *node identifiers*: *Structured-GNN (SGNN)*

Vignac et al. *Building powerful and equivariant graph neural networks with message-passing* (2020)

Beyond WL

Going "beyond WL"...

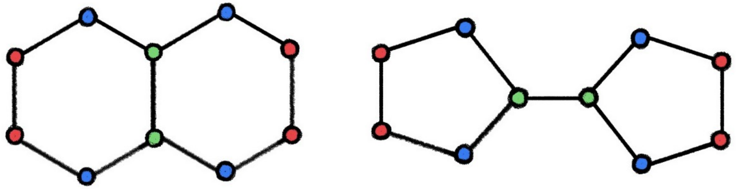
- Using *higher-order tensors*
  - *Up until true universality!*

Maron et al. *Provably Powerful Graph Networks* (2019)

Maron et al. *On the Universality of Invariant Networks* (2019)

Keriven and Peyré. *Universal Invariant and Equivariant Graph Neural Networks* (2020)

- Using higher-order *subgraph counting*



Morris et al. *Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks* (2019)

Bouritsas et al. *Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting* (2020)

- Using *node identifiers*: *Structured-GNN (SGNN)*

$$\Phi_G(1_n)$$

Vignac et al. *Building powerful and equivariant graph neural networks with message-passing* (2020)

Beyond WL

Going "beyond WL"...

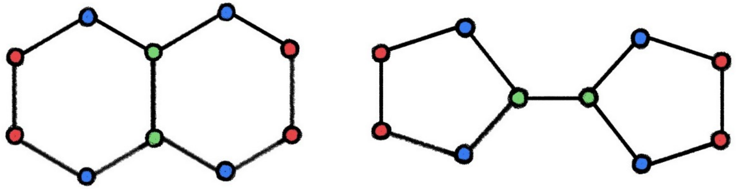
- Using *higher-order tensors*
  - *Up until true universality!*

Maron et al. *Provably Powerful Graph Networks* (2019)

Maron et al. *On the Universality of Invariant Networks* (2019)

Keriven and Peyré. *Universal Invariant and Equivariant Graph Neural Networks* (2020)

- Using higher-order *subgraph counting*



Morris et al. *Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks* (2019)

Bouritsas et al. *Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting* (2020)

- Using *node identifiers*: *Structured-GNN (SGNN)*

$$\Phi_G(1_n) = \Phi_G\left(\sum_i e_i\right)$$

Vignac et al. *Building powerful and equivariant graph neural networks with message-passing* (2020)

Beyond WL

Going "beyond WL"...

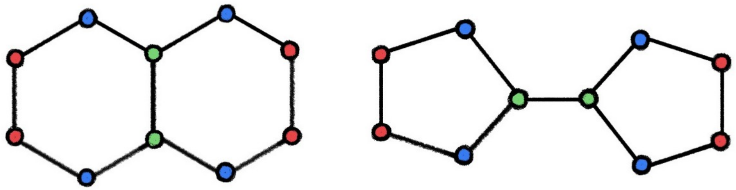
- Using *higher-order tensors*
  - *Up until true universality!*

Maron et al. *Provably Powerful Graph Networks* (2019)

Maron et al. *On the Universality of Invariant Networks* (2019)

Keriven and Peyré. *Universal Invariant and Equivariant Graph Neural Networks* (2020)

- Using higher-order *subgraph counting*



Morris et al. *Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks* (2019)

Bouritsas et al. *Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting* (2020)

- Using *node identifiers*: *Structured-GNN (SGNN)*

$$\Phi_G(1_n) = \Phi_G(\textstyle\sum_i e_i) \to \Phi_G(\textstyle\sum_i \Phi'_G(e_i))$$

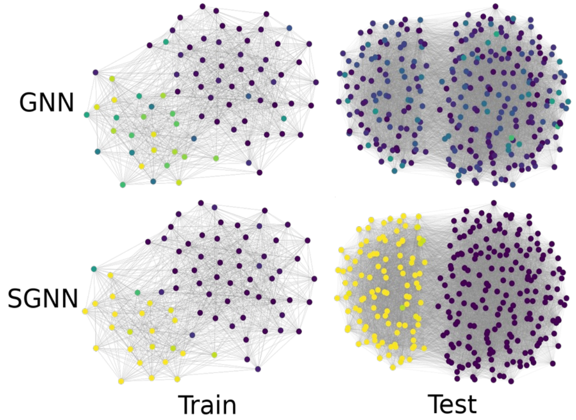
Vignac et al. *Building powerful and equivariant graph neural networks with message-passing* (2020)

Continuous SGNN and universality

- Thm. SGNNs converge toward c-SGNN.

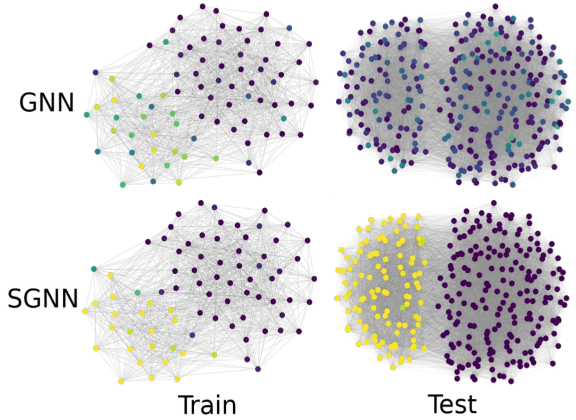
Continuous SGNN and universality

- Thm. SGNNs converge toward c-SGNN.
- Thm. c-SGNNs are *strictly more powerful* than c-GNNs



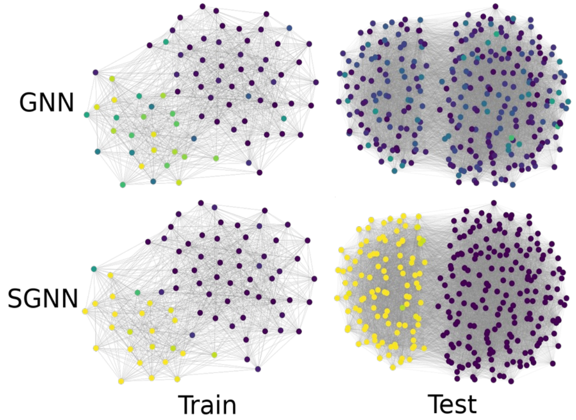
Continuous SGNN and universality

- Thm. SGNNs converge toward c-SGNN.
- Thm. c-SGNNs are *strictly more powerful* than c-GNNs
- Thm. Using Stone-Weierstrass theorem, c-SGNNs are universal (both permutation-invariant/equivariant):



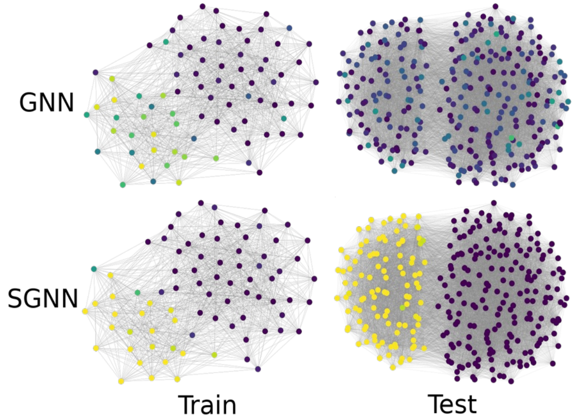
Continuous SGNN and universality

- Thm. SGNNs converge toward c-SGNN.
- Thm. c-SGNNs are *strictly more powerful* than c-GNNs
- Thm. Using Stone-Weierstrass theorem, c-SGNNs are universal (both permutation-invariant/equivariant):
  - On most SBMs



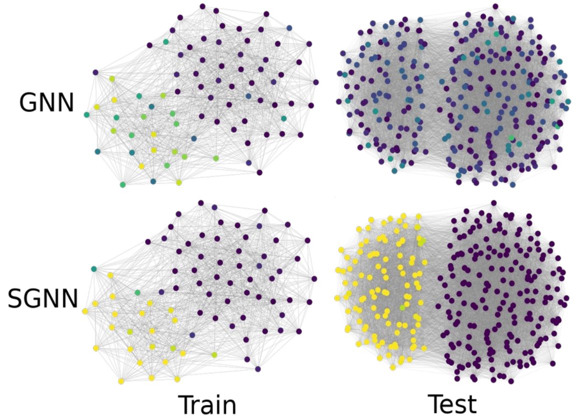
Continuous SGNN and universality

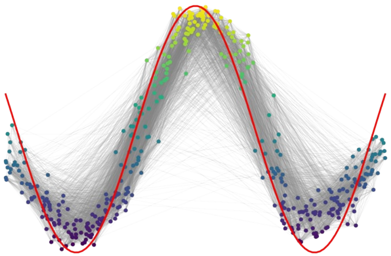
- Thm. SGNNs converge toward c-SGNN.
- Thm. c-SGNNs are *strictly more powerful* than c-GNNs
- Thm. Using Stone-Weierstrass theorem, c-SGNNs are universal (both permutation-invariant/equivariant):
  - On most SBMs
  - Many "additive" kernels $W(x, y) = w(v(x) + v(y))$



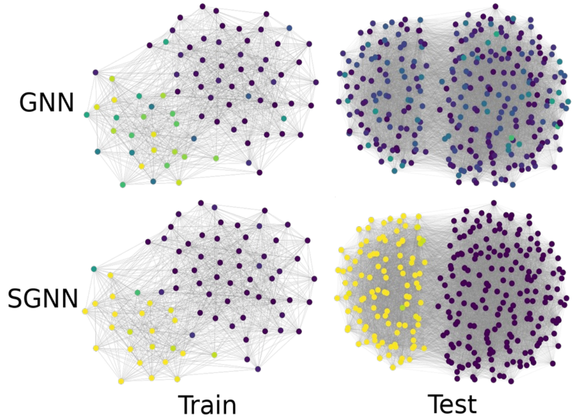
Continuous SGNN and universality

- Thm. SGNNs converge toward c-SGNN.
- Thm. c-SGNNs are *strictly more powerful* than c-GNNs
- Thm. Using Stone-Weierstrass theorem, c-SGNNs are universal (both permutation-invariant/equivariant):
  - On most SBMs
  - Many "additive" kernels $W(x, y) = w(v(x) + v(y))$
  - 1D radial kernels *(w/ symmetry)* $W(x, y) = w(|x - y|)$

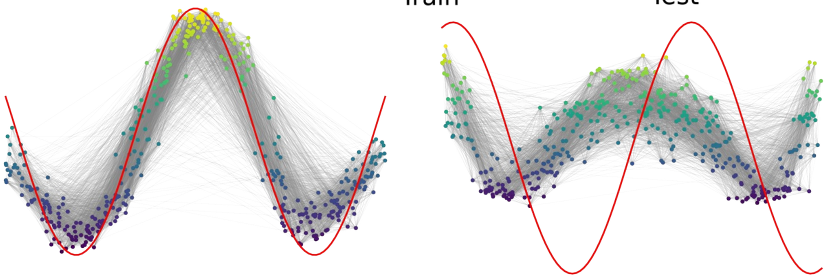




Continuous SGNN and universality

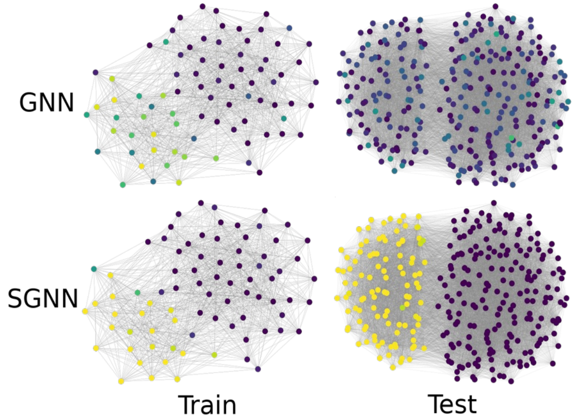


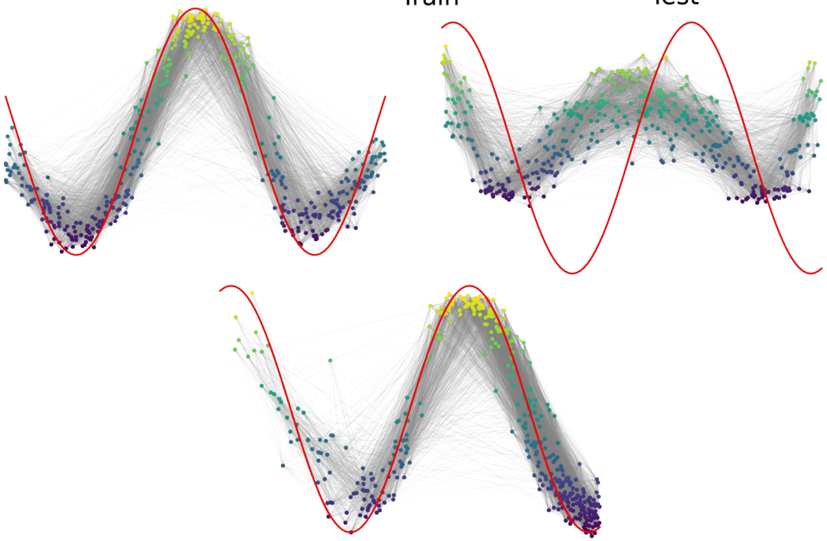
- Thm. SGNNs converge toward c-SGNN.
- Thm. c-SGNNs are *strictly more powerful* than c-GNNs
- Thm. Using Stone-Weierstrass theorem, c-SGNNs are universal (both permutation-invariant/equivariant):



  - On most SBMs
  - Many "additive" kernels $W(x,y) = w(v(x) + v(y))$
  - 1D radial kernels *(w/ symmetry)* $W(x,y) = w(|x-y|)$

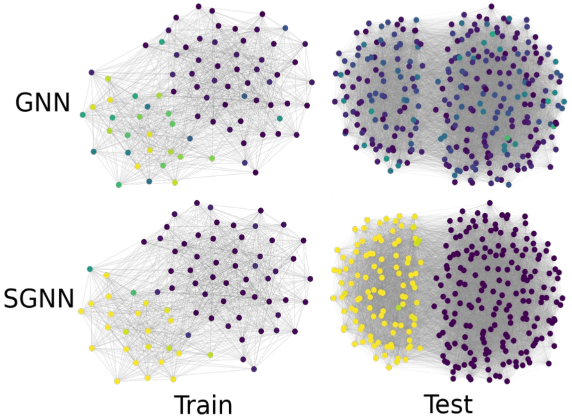
- Thm. SGNNs converge toward c-SGNN.
- Thm. c-SGNNs are *strictly more powerful* than c-GNNs
- Thm. Using Stone-Weierstrass theorem, c-SGNNs are universal (both permutation-invariant/equivariant):
  - On most SBMs
  - Many "additive" kernels $W(x,y) = w(v(x) + v(y))$
  - 1D radial kernels *(w/ symmetry)* $W(x,y) = w(|x-y|)$



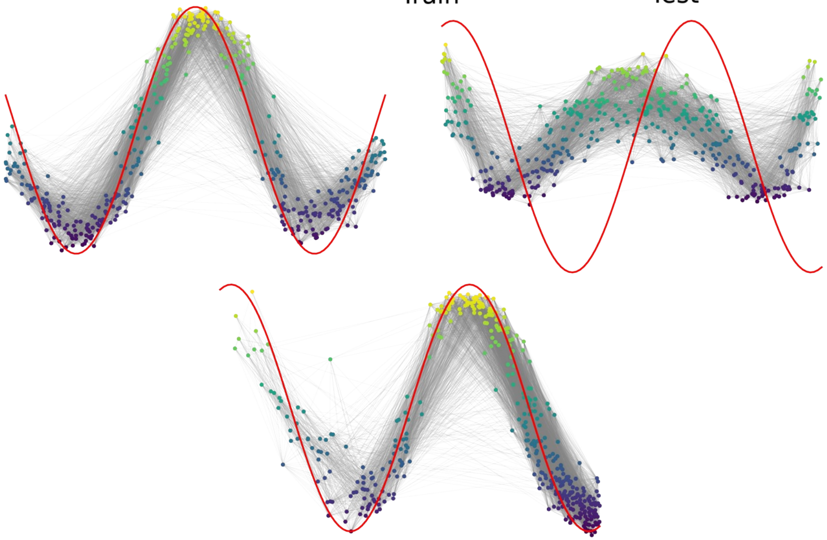


Continuous SGNN and universality

- Thm. SGNNs converge toward c-SGNN.
- Thm. c-SGNNs are *strictly more powerful* than c-GNNs



- Thm. Using Stone-Weierstrass theorem, c-SGNNs are universal (both permutation-invariant/equivariant):
  - On most SBMs
  - Many "additive" kernels $W(x,y) = w(v(x) + v(y))$
  - 1D radial kernels *(w/ symmetry)* $W(x,y) = w(|x-y|)$
  - Most dot-product kernels... $W(x,y) = w(x^\top y)$



Conclusion

- Many *complementary* approaches to GNN analysis

Conclusion

- Many *complementary* approaches to GNN analysis
- Random graphs help model large-scale properties

Conclusion

- Many *complementary* approaches to GNN analysis
- Random graphs help model large-scale properties

Outlooks:

- **Sparse graphs**, “differential” Laplacian operator, preferential attachement graphs...
- Message-passing GNNs, high-order tensors...
- Generalization, optimization...

Conclusion

- Many *complementary* approaches to GNN analysis
- Random graphs help model large-scale properties

Outlooks:

- **Sparse graphs**, "differential" Laplacian operator, preferential attachement graphs...
- Message-passing GNNs, high-order tensors...
- Generalization, optimization...

Keriven, Bietti, Vaiter. *Convergence and Stability of Graphs Convolutional Networks on Large Random Graphs.* NeurIPS (Spotlight) (2020)

Keriven, Bietti, Vaiter. *On the Universality of Graph Neural Networks on Large Random Graphs.* Submitted. (2021)

nkeriven.github.io