Graph Neural Networks: Introduction, some theoretical properties

Nicolas Keriven

CNRS, IRISA

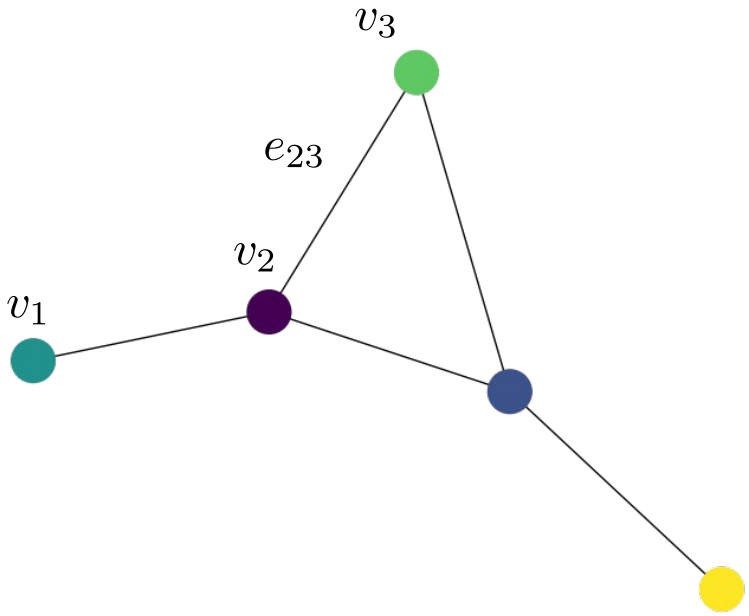




Graphs ?

A Graph $G = (V, E)$ is formed by:

- Nodes (or vertices) $V = \{v_1, \ldots, v_n\}$
- Edges $E = \{e_{i_1 j_1}, \ldots, e_{i_m j_m}\}$



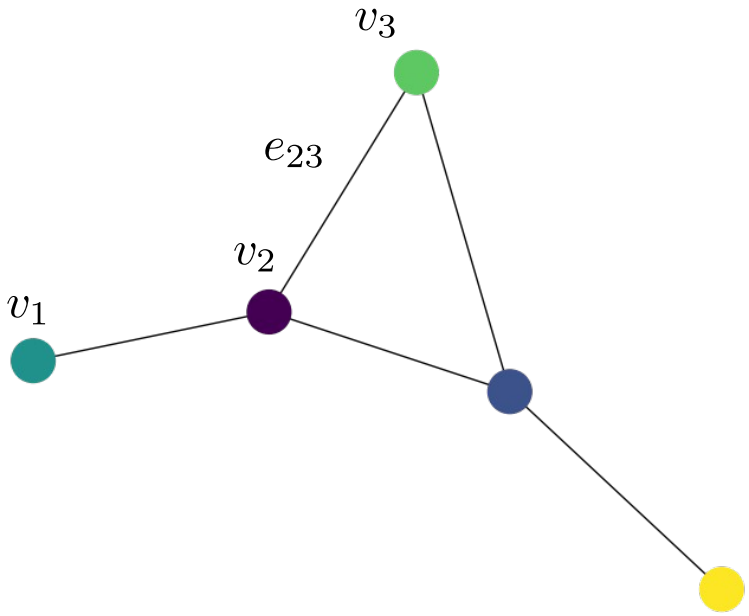
Graphs ?

A Graph $G = (V, E)$ is formed by:

- Nodes (or vertices) $V = \{v_1, \ldots, v_n\}$
- Edges $E = \{e_{i_1 j_1}, \ldots, e_{i_m j_m}\}$

Can include:

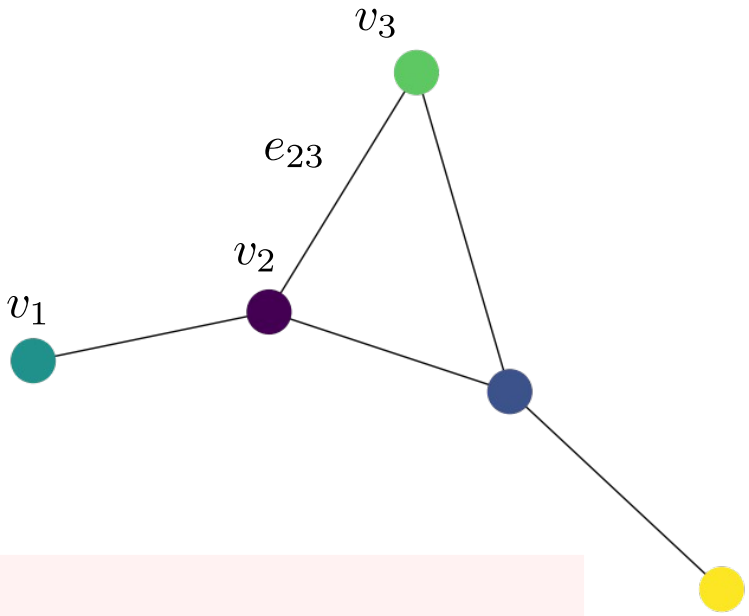
- Node features
- Edges features
- etc



Graphs ?

A Graph $G = (V, E)$ is formed by:

- Nodes (or vertices) $V = \{v_1, \ldots, v_n\}$
- Edges $E = \{e_{i_1 j_1}, \ldots, e_{i_m j_m}\}$



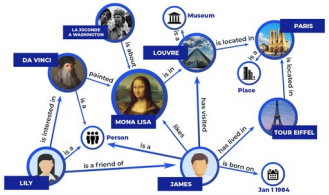
Can include:

- Node features
- Edges features
- etc

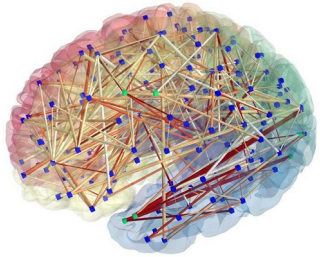
A graph is:

- A purely mathematical object
- A principled way to represent many types of complex data (eg. any type of network)

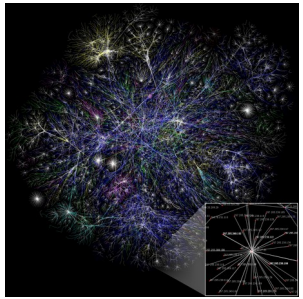
Graphs: examples



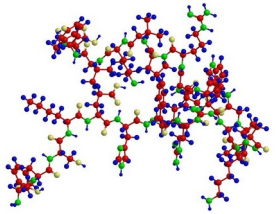
Knowledge graph



Brain connectivity network



Internet



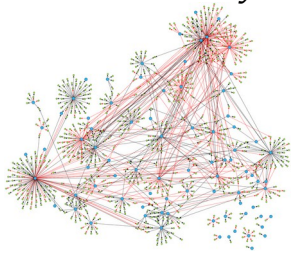
Molecule



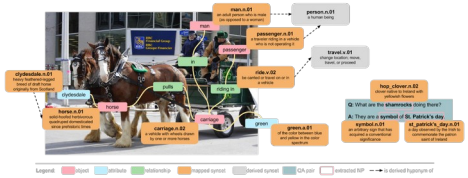
Social network



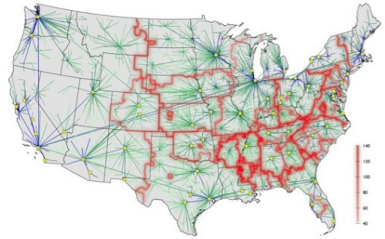
Computer network



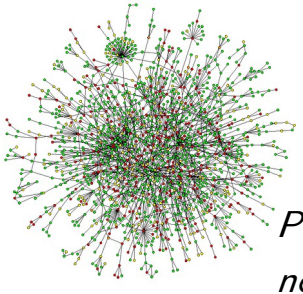
Gene regulatory network



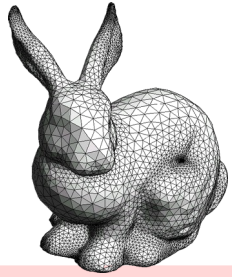
Scene understanding network



Transportation network

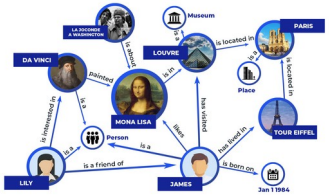


Protein interaction network

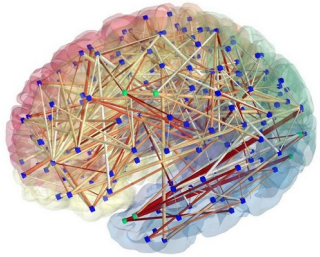


3D mesh

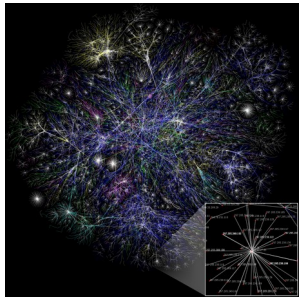
Graphs: examples



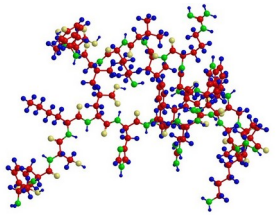
Knowledge graph



Brain connectivity network



Internet



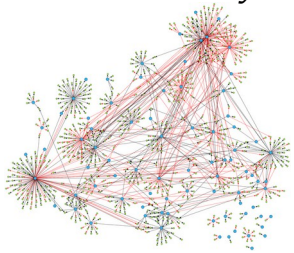
Molecule



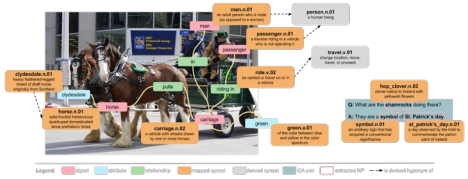
Social network



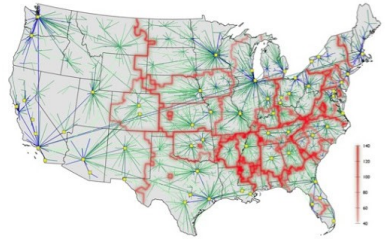
Computer network



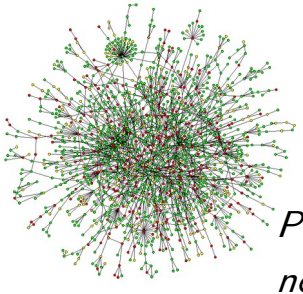
Gene regulatory network



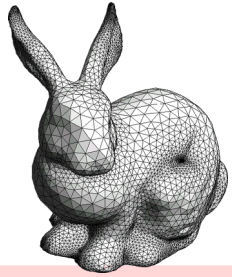
Scene understanding network



Transportation network



Protein interaction network



3D mesh

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

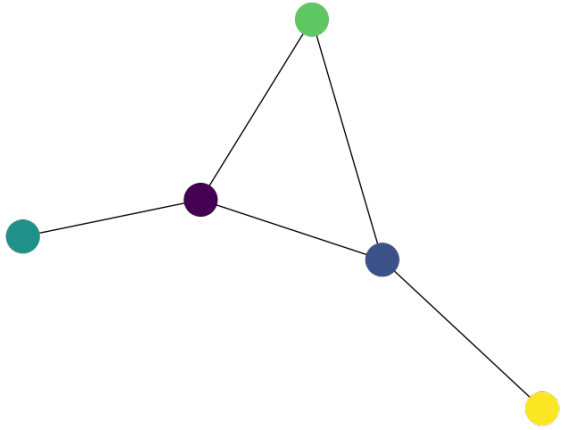
Graphs: notations

A graph is usually represented by

- An adjacency matrix $A \in \{0,1\}^{n \times n} : A_{ij} = \begin{cases} 1 \text{ if } (i,j) \in E \\ 0 \text{ otherwise.} \end{cases}$
- (Often!) node feature matrix $Z \in \mathbb{R}^{n \times d}$

A is usually *sparse*, (lots of 0s), so fast to handle with dedicated tools

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



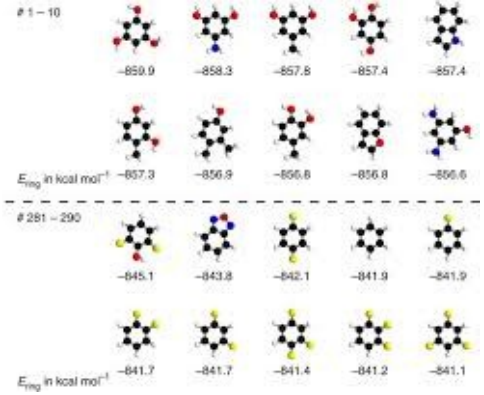
Machine learning... on graphs

Machine learning on graphs **comes in many flavors**

Machine learning... on graphs

Machine learning on graphs **comes in many flavors**

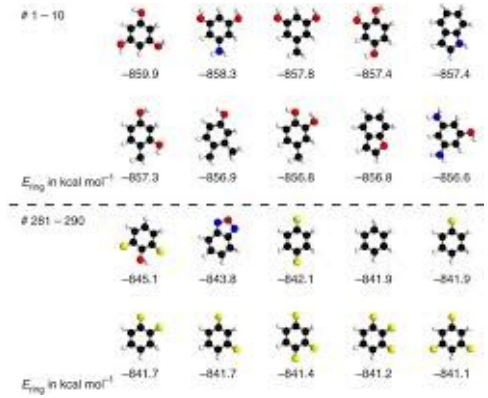
- Supervised/semi-supervised:
  - Graph prediction: labelled graphs -> label new graph
    - Molecule classification, drug efficiency prediction
  - Node (or edge) prediction: labelled nodes -> label other nodes
    - Recommender systems, protein interface prediction

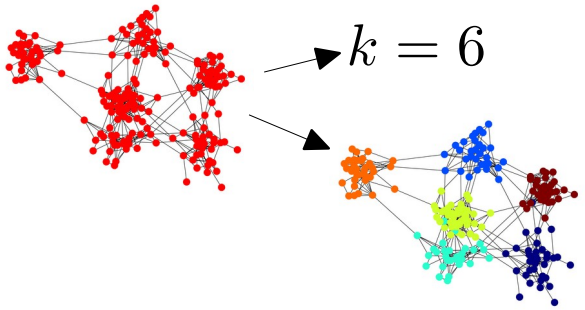


Machine learning... on graphs

Machine learning on graphs **comes in many flavors**

- Supervised/semi-supervised:
  - Graph prediction: labelled graphs -> label new graph
    - Molecule classification, drug efficiency prediction
  - Node (or edge) prediction: labelled nodes -> label other nodes
    - Recommender systems, protein interface prediction
- Unsupervised (... also semi-supervised):
  - Community detection: one graph -> group nodes
    - Social network analysis
  - Link prediction: one graph -> potential new edge?
    - Recommender systems

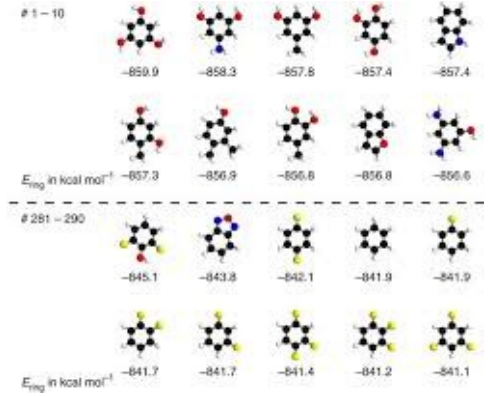


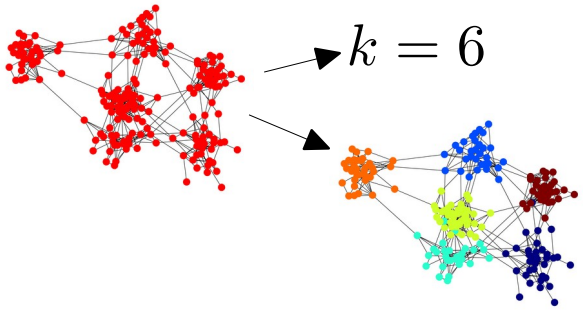


Machine learning... on graphs

Machine learning on graphs **comes in many flavors**

- Supervised/semi-supervised:
  - Graph prediction: labelled graphs -> label new graph
    - Molecule classification, drug efficiency prediction
  - Node (or edge) prediction: labelled nodes -> label other nodes
    - Recommender systems, protein interface prediction
- Unsupervised (... also semi-supervised):
  - Community detection: one graph -> group nodes
    - Social network analysis
  - Link prediction: one graph -> potential new edge?
    - Recommender systems
- Generative (diffusion, flow-matching...)
  - drug design, data augmentation, conditioning for other modalities...

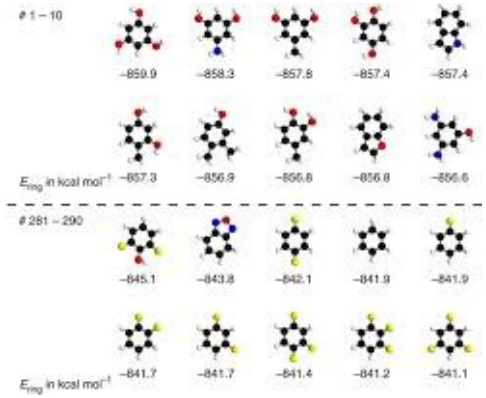


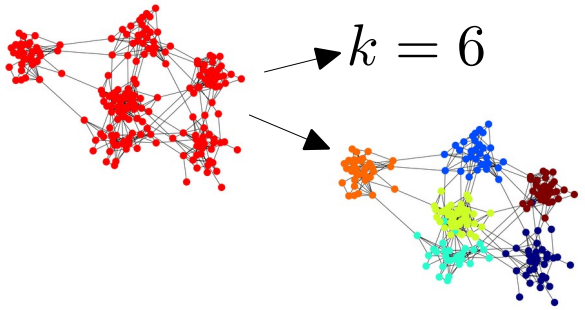


Machine learning... on graphs

Machine learning on graphs **comes in many flavors**

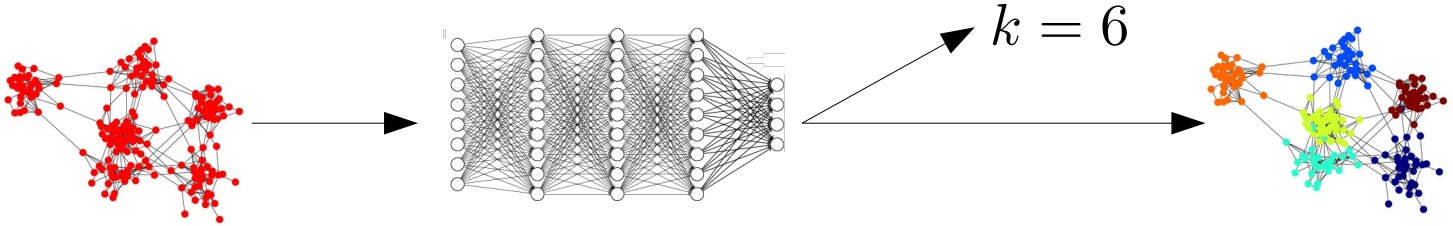
- Supervised/semi-supervised:
  - Graph prediction: labelled graphs -> label new graph
    - Molecule classification, drug efficiency prediction
  - Node (or edge) prediction: labelled nodes -> label other nodes
    - Recommender systems, protein interface prediction
- Unsupervised (... also semi-supervised):
  - Community detection: one graph -> group nodes
    - Social network analysis
  - Link prediction: one graph -> potential new edge?
    - Recommender systems
- Generative (diffusion, flow-matching...)
  - drug design, data augmentation, conditioning for other modalities...





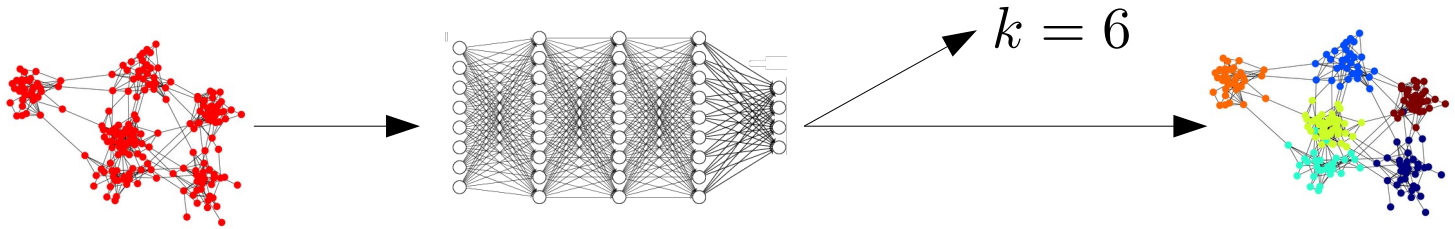
Need to be permutation invariant/equivariant

ML on graphs: Graph Neural Networks



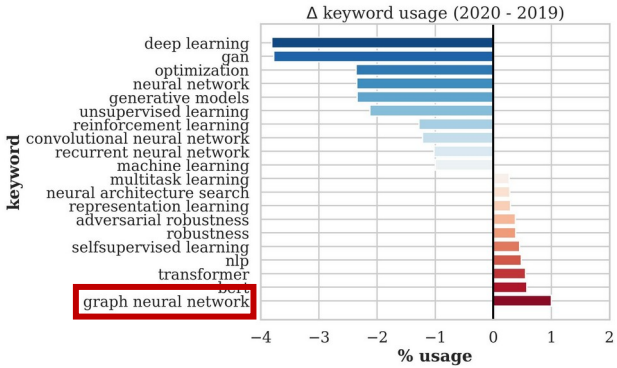
Graph Neural Networks (GNN) are "deep architectures" on graphs.

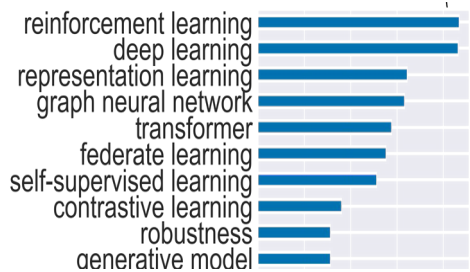
ML on graphs: Graph Neural Networks



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

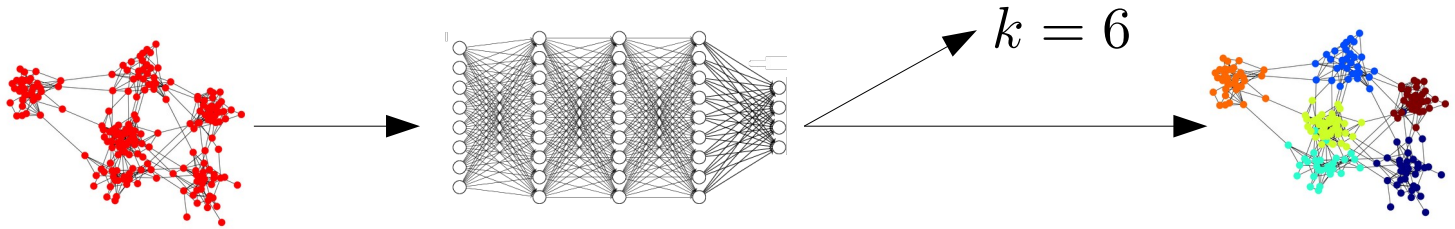
- (Used to be?) very trendy
  - A lot of good papers, a lot of noise
  - Somewhat a a more mature field, even if very active community still





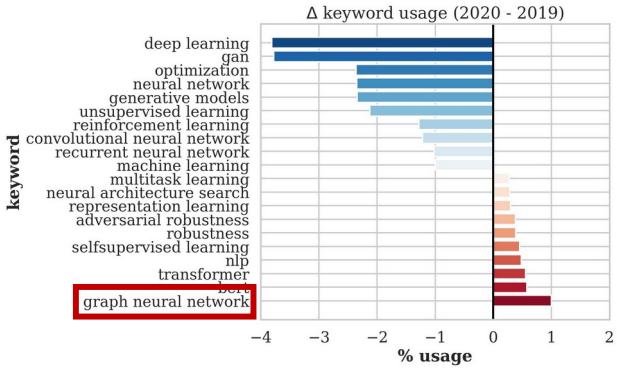
ICLR 2023

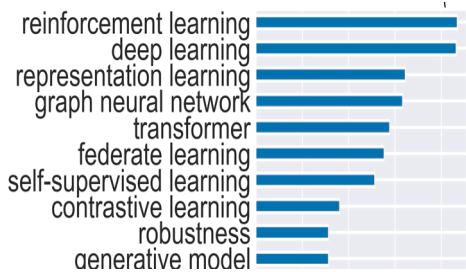
ML on graphs: Graph Neural Networks



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

- (Used to be?) very trendy
  - A lot of good papers, a lot of noise
  - Somewhat a a more mature field, even if very active community still
- Does not work that well(?) (compared to other "deep learning")
  - Simple methods may perform better, people might not test them...
  - Room for improvement (even after all this time)
  - Still, they ***teach us*** a lot of useful intuitions on ***graph ML problems***





ICLR 2023

ML on graphs: some material

- (Some) GNN reviews
  - Wu et al. *A Comprehensive Survey on Graph Neural Networks.* (2020)
  - Dwivedi et al. *Benchmarking Graph Neural Networks.* (2020)
  - Bronstein et al. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges* (2021)
  - Zhang et al. *A Survey on Graph Neural Networks* (2026)
  - Tanis et al. *Introduction to Graph Neural Networks for Machine Learning Engineers* (2026)
  - Etc.
- Datasets
  - Stanford Large Network Dataset Collection. snap.stanford.edu/data
  - Hu et al. *Open Graph Benchmark* (2020)
  - Pytorch Geometric datasets
- Python Libraries
  - Networkx (medium-sized graph manipulation, visualization)
  - Pytorch Geometric
  - Deep Graph Library (Tensorflow-based GNN)

ML on graphs: some material

- (Some) GNN reviews
  - Wu et al. *A Comprehensive Survey on Graph Neural Networks.* (2020)
  - Dwivedi et al. *Benchmarking Graph Neural Networks* (2020)
  - Bronstein et al. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges* (2021)
  - Zhang et al. *A Survey on Graph Neural Networks* (2026)
  - Tanis et al. *Introduction to Graph Neural Networks for Machine Learni*
  - Etc.
- Datasets
  - Stanford Large Network Dataset Collection. snap.stanford.edu/data
  - Hu et al. *Open Graph Benchmark* (2020)
  - Pytorch Geometric datasets
- Python Libraries
  - Networkx (medium-sized graph manipulation, visualization)
  - Pytorch Geometric
  - Deep Graph Library (Tensorflow-based GNN)

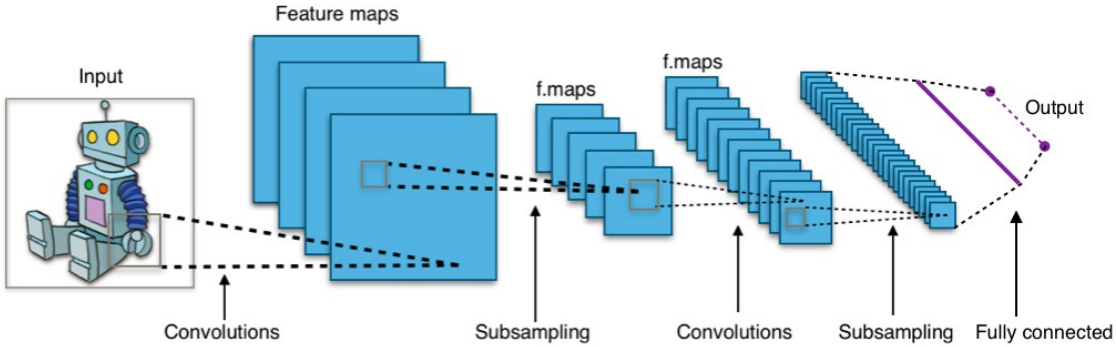
Just download Pytorch Geometric

Outline

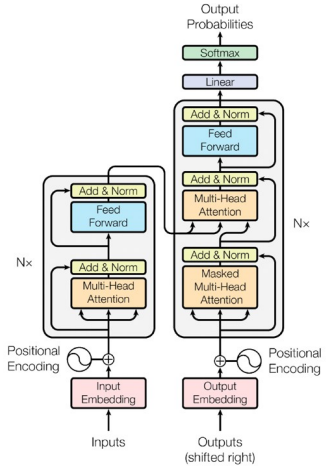
1 From Deep Convolutional Networks to GNNs

2 Some (theoretical) properties of GNNs

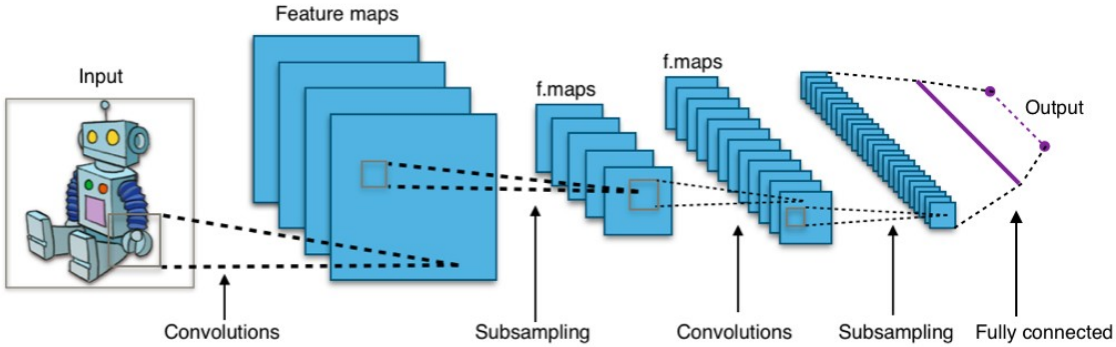
Deep Neural Networks



"Deep" learning: alternates between linearities and (differentiable) non-linearities



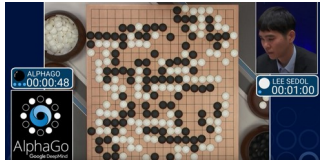
Deep Neural Networks

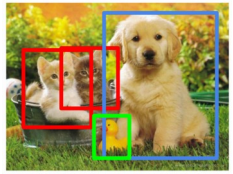


"Deep" learning: alternates between linearities and (differentiable) non-linearities

State-of-the-art in everything

- Computer vision
- Speech recognition
- Natural Language Processing
- Reinforcement learning
- Generative AI
- ...

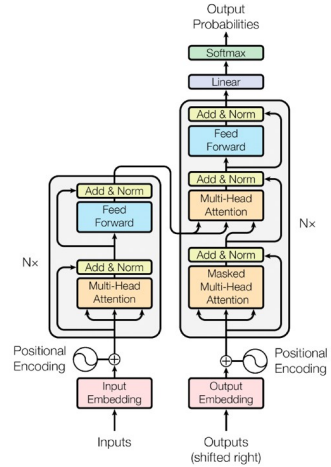




CAT, DOG, DUCK

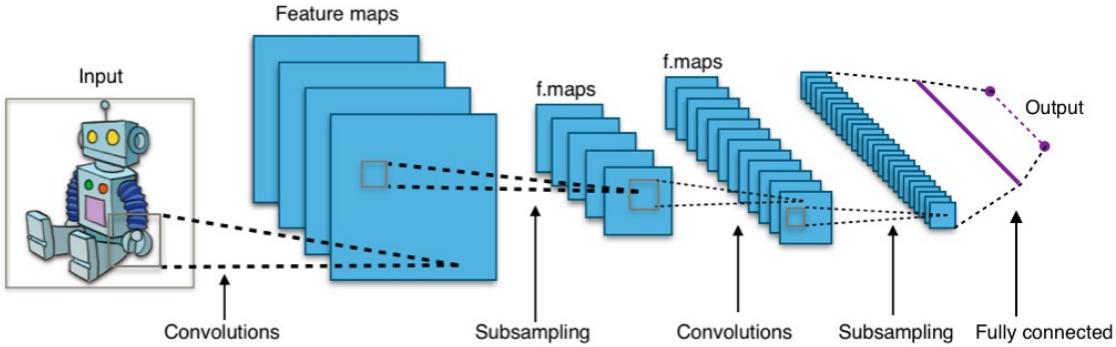








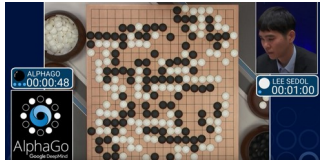
Deep Neural Networks

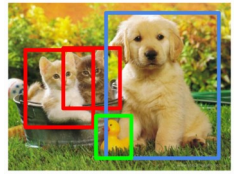


"Deep" learning: alternates between linearities and (differentiable) non-linearities

State-of-the-art in everything

- Computer vision
- Speech recognition
- Natural Language Processing
- Reinforcement learning
- Generative AI
- ...

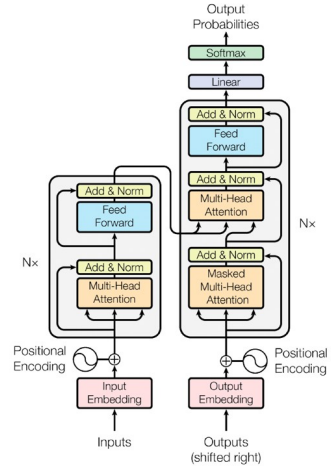




CAT, DOG, DUCK









How do we extend them to Graphs?
No node ordering: must be permutation-invariant (invariant to graph isomorphism)



ConvNets: convolution?

- The building blocks of Deep (Convolutional) Neural Networks are convolutions.

ConvNets: convolution?

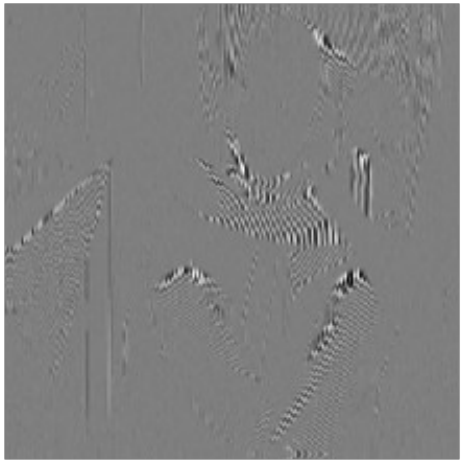
- The building blocks of Deep (Convolutional) Neural Networks are convolutions.



$\star$



$=$



$$(x \star h)[n] = \sum_r h[r]x[n-r]$$

Convolutions are local pattern-matching linear operators.

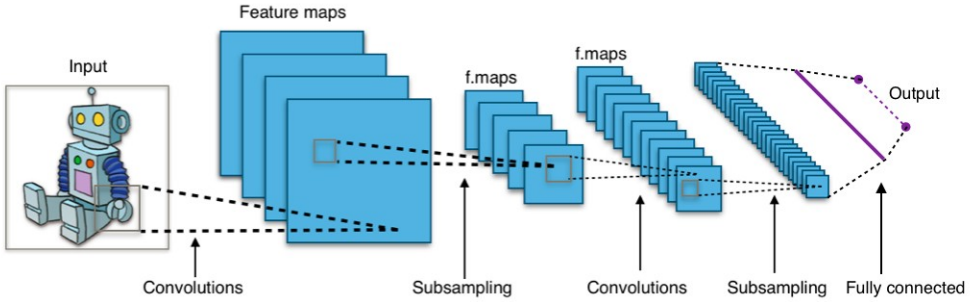
Usual filter banks (wavelets) use fixed filters, in ML the filters are (usually) learned.

ConvNets: multiscale

- How to detect complex "high-level" shapes?

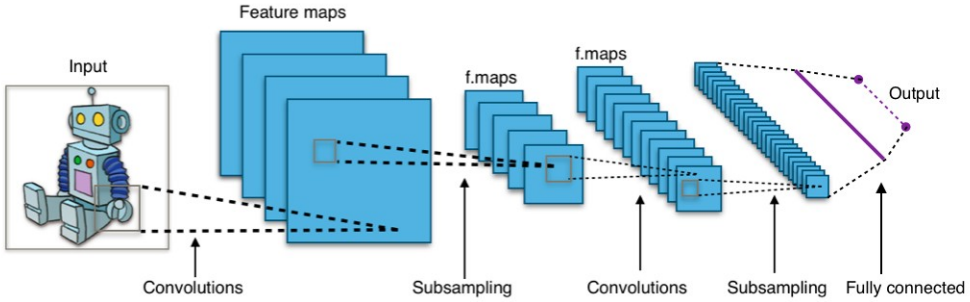
ConvNets: multiscale

- How to detect complex "high-level" shapes?
  - Trying every pattern is impossible! -> stack filters (and subsampling) to make it hierarchical.



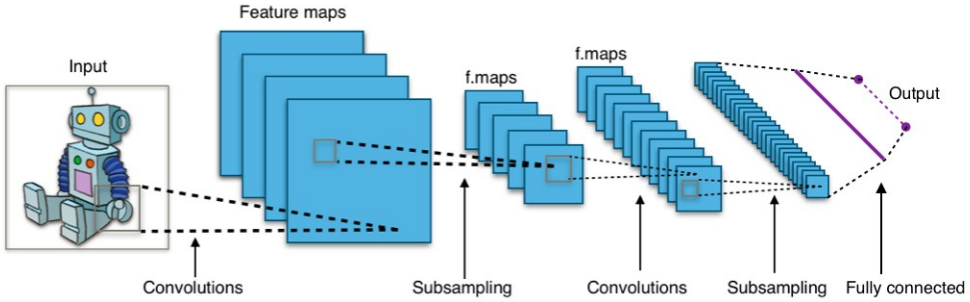
ConvNets: multiscale

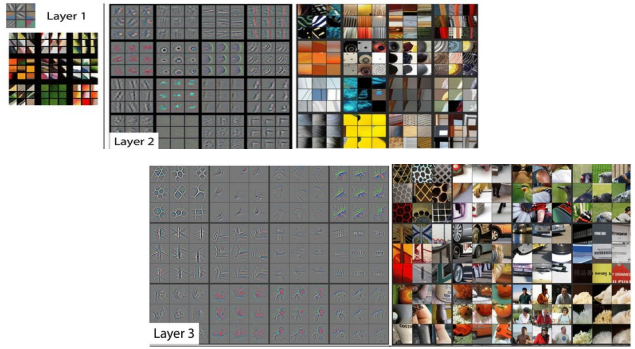
- How to detect complex "high-level" shapes?
  - Trying every pattern is impossible! -> stack filters (and subsampling) to make it hierarchical.
  - Naively stacking convolutions is still linear... -> add non-linear functions between each scale (layer)

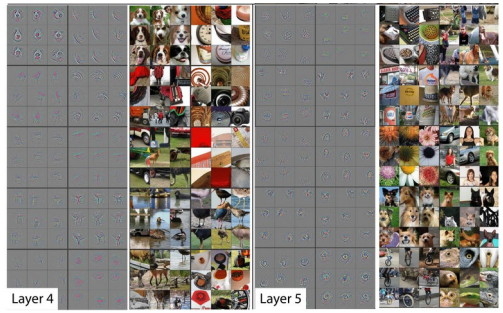


ConvNets: multiscale

- How to detect complex "high-level" shapes?
  - Trying every pattern is impossible! -> stack filters (and subsampling) to make it hierarchical.
  - Naively stacking convolutions is still linear... -> add non-linear functions between each scale (layer)





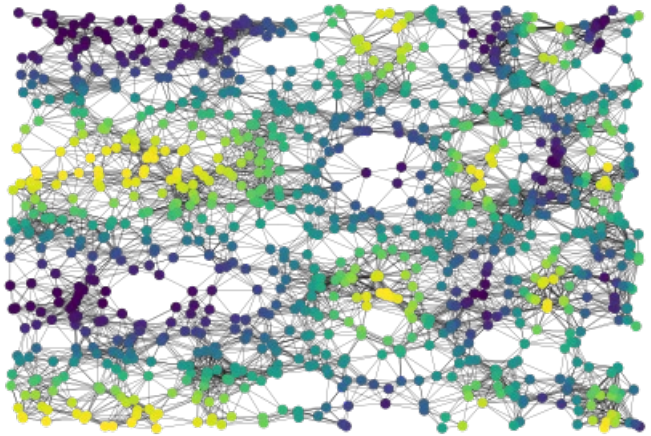


- Deeper layers are indeed activated by "higher-level" patterns.

Zeiler and Fergus. *Visualizing and Understanding Convolutional Networks* (2013)

Convolution on graphs?

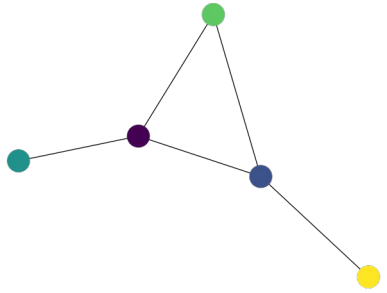
How to perform convolution of graphs?



$z \in \mathbb{R}^n$ on G

$?$

$\star$



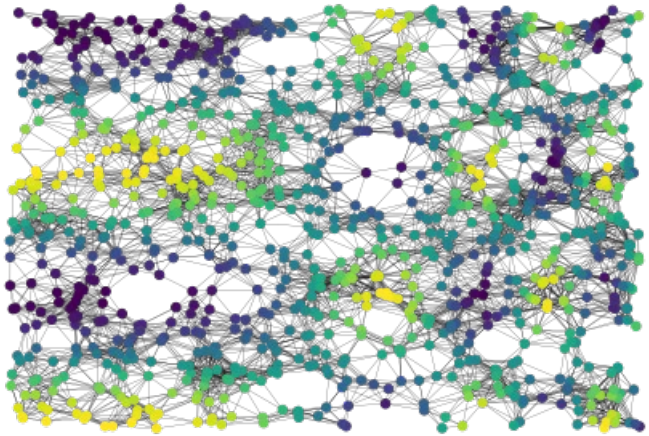
$h \in \mathbb{R}^p$ on H

Convolution on graphs?

How to perform convolution of graphs?

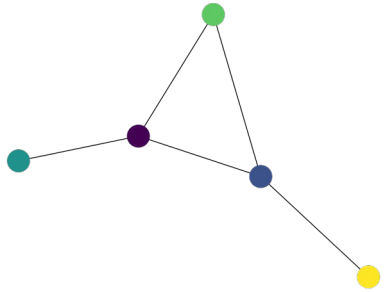
Two (main) problems to "pattern-matching" on graphs:

- No inherent node ordering
- No fixed neighborhood size



$z \in \mathbb{R}^n$ on G

$\overset{?}{\star}$



$h \in \mathbb{R}^p$ on H

Convolution and Fourier

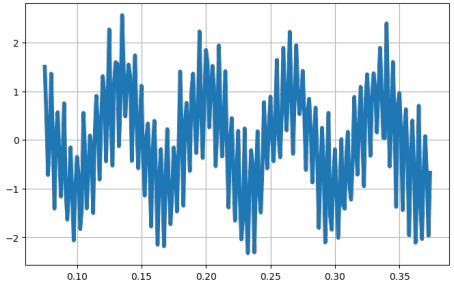
The "convolution theorem": convolution is multiplication in the Fourier domain.

$$\mathcal{F}(x \star h) = \mathcal{F}(x) \cdot \mathcal{F}(h)$$

Convolution and Fourier

The "convolution theorem": convolution is multiplication in the Fourier domain.

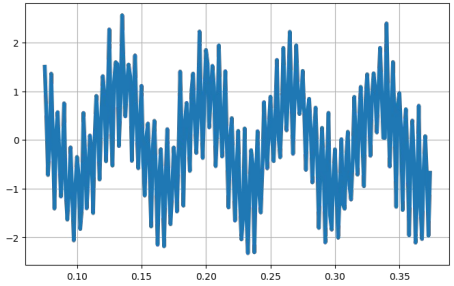
$$\mathcal{F}(x \star h) = \mathcal{F}(x) \cdot \mathcal{F}(h)$$

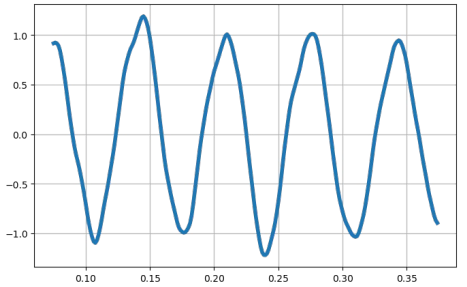


Convolution and Fourier

The "convolution theorem": convolution is multiplication in the Fourier domain.

$$\mathcal{F}(x \star h) = \mathcal{F}(x) \cdot \mathcal{F}(h)$$

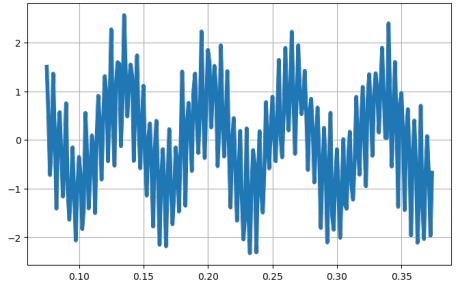




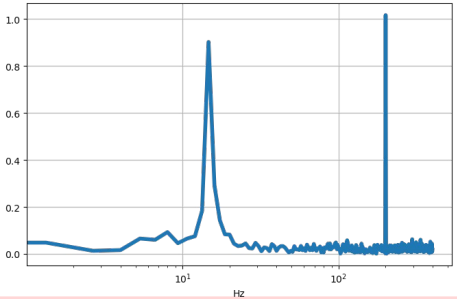
Convolution and Fourier

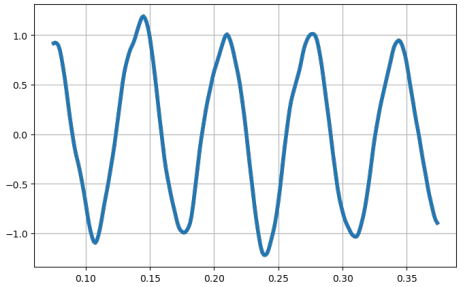
The "convolution theorem": convolution is multiplication in the Fourier domain.

$$\mathcal{F}(x \star h) = \mathcal{F}(x) \cdot \mathcal{F}(h)$$



$\mathcal{F}$

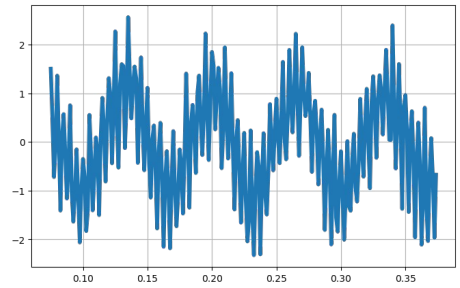


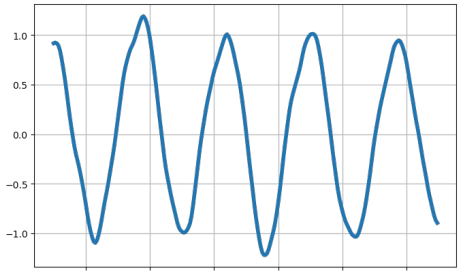


Convolution and Fourier

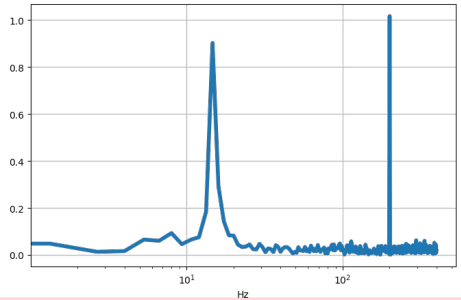
The "convolution theorem": convolution is multiplication in the Fourier domain.

$$\mathcal{F}(x \star h) = \mathcal{F}(x) \cdot \mathcal{F}(h)$$

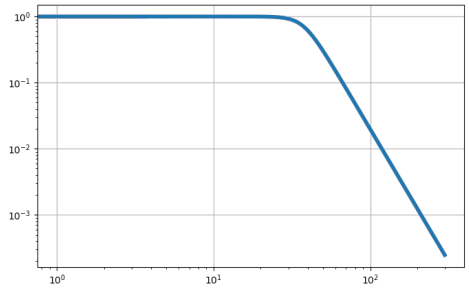




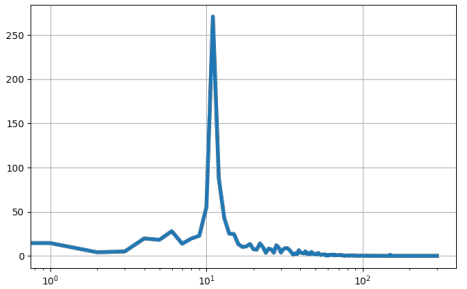
$\mathcal{F}$



$\times$



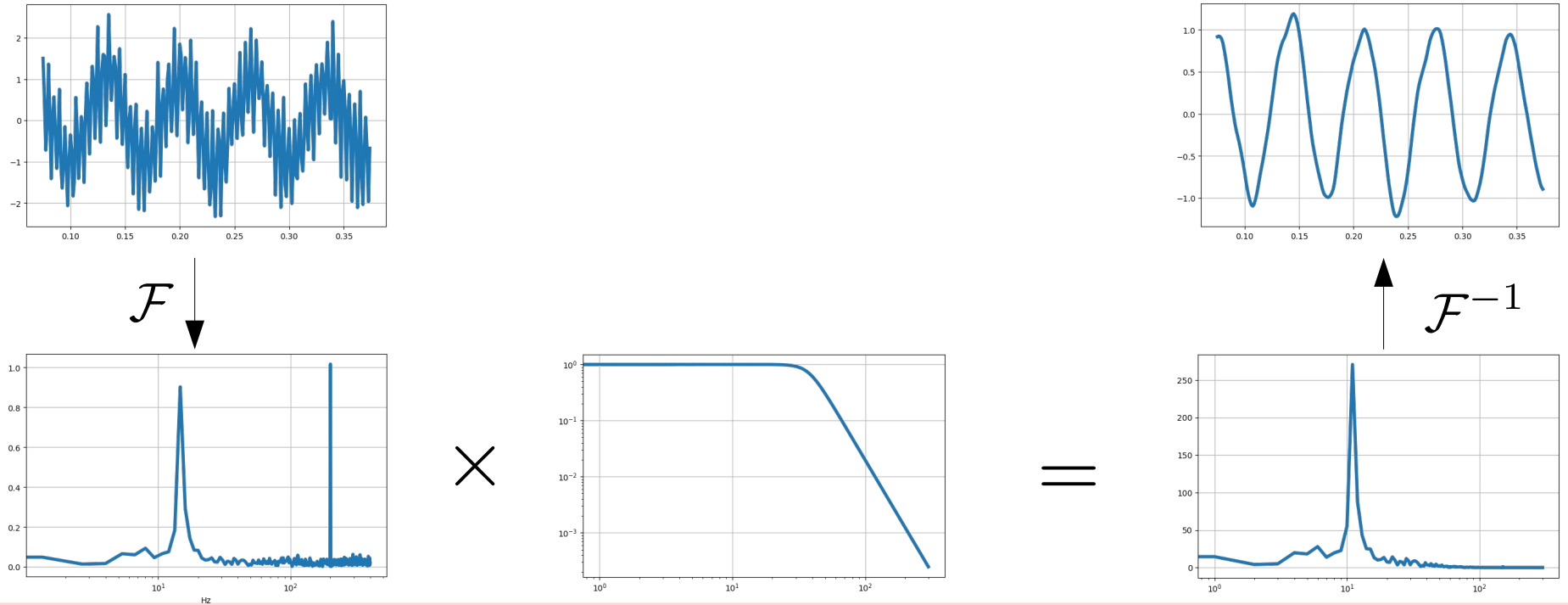
$=$



Convolution and Fourier

The "convolution theorem": convolution is multiplication in the Fourier domain.

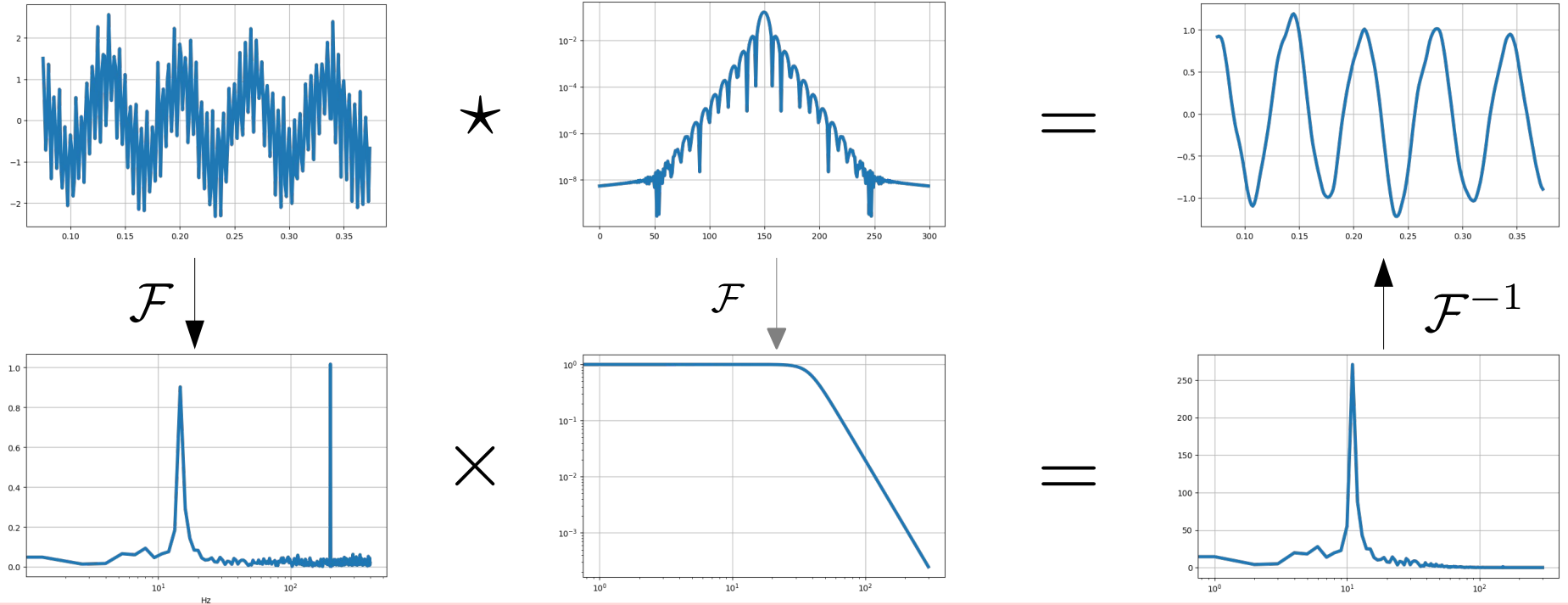
$$\mathcal{F}(x \star h) = \mathcal{F}(x) \cdot \mathcal{F}(h)$$



Convolution and Fourier

The "convolution theorem": convolution is multiplication in the Fourier domain.

$$\mathcal{F}(x \star h) = \mathcal{F}(x) \cdot \mathcal{F}(h)$$



Fourier transform on graphs: Laplacian

How to define the Fourier transform on graphs? $\mathcal{F}f(\omega) = \int f(t)e^{-2i\pi\omega t}dt = \langle f, e^{-2i\pi\omega\cdot}\rangle_{L^2}$

Fourier transform on graphs: Laplacian

How to define the Fourier transform on graphs? $\mathcal{F}f(\omega) = \int f(t)e^{-2i\pi\omega t}dt = \langle f, e^{-2i\pi\omega\cdot}\rangle_{L^2}$

- (On the torus) complex exponentials are eigenfunctions of the Laplacian $\Delta = \sum_i \frac{\partial^2}{\partial x_i^2}$

Fourier transform on graphs: Laplacian

How to define the Fourier transform on graphs? $\mathcal{F}f(\omega) = \int f(t)e^{-2i\pi\omega t}dt = \langle f, e^{-2i\pi\omega\cdot}\rangle_{L^2}$

- (On the torus) complex exponentials are eigenfunctions of the Laplacian $\Delta = \sum_i \frac{\partial^2}{\partial x_i^2}$
- Laplacian operators (matrix) on graphs can be defined!

$$L = D - A \qquad D = diag(d_i) \text{ with degrees } d_i = (A1_n)_i$$

Fourier transform on graphs: Laplacian

How to define the Fourier transform on graphs? $\mathcal{F}f(\omega) = \int f(t)e^{-2i\pi\omega t}dt = \langle f, e^{-2i\pi\omega\cdot}\rangle_{L^2}$

- (On the torus) complex exponentials are eigenfunctions of the Laplacian $\Delta = \sum_i \frac{\partial^2}{\partial x_i^2}$
- Laplacian operators (matrix) on graphs can be defined!

$$L = D - A \qquad D = diag(d_i) \text{ with degrees } d_i = (A1_n)_i$$

- Dirichlet energy $z^\top Lz = \sum_{e_{ij}\in E}(z_i - z_j)^2$, and $L = \nabla^\top\nabla$ where $(\nabla z)(e_{ij}) = z_i - z_j$

Fourier transform on graphs: Laplacian

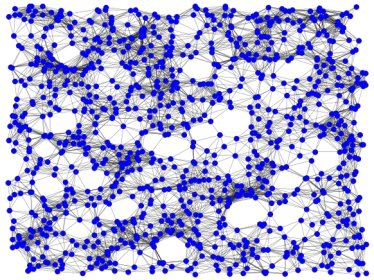
How to define the Fourier transform on graphs? $\mathcal{F}f(\omega) = \int f(t)e^{-2i\pi\omega t}dt = \langle f, e^{-2i\pi\omega\cdot}\rangle_{L^2}$

- (On the torus) complex exponentials are eigenfunctions of the Laplacian $\Delta = \sum_i \frac{\partial^2}{\partial x_i^2}$
- Laplacian operators (matrix) on graphs can be defined!

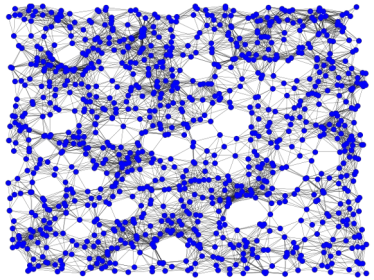
$$L = D - A \qquad D = diag(d_i) \text{ with degrees } d_i = (A1_n)_i$$

- Dirichlet energy $z^\top Lz = \sum_{e_{ij}\in E}(z_i - z_j)^2$, and $L = \nabla^\top\nabla$ where $(\nabla z)(e_{ij}) = z_i - z_j$
- Normalized Laplacian (eigenvalues between 0 and 2) $L = Id - D^{-1/2}AD^{-1/2}$

Fourier transform on graphs



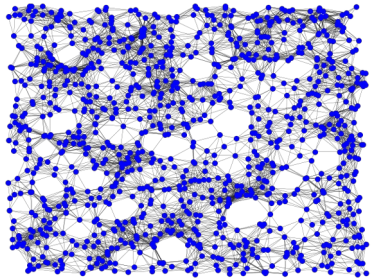
Fourier transform on graphs



Diagonalize the Laplacian:

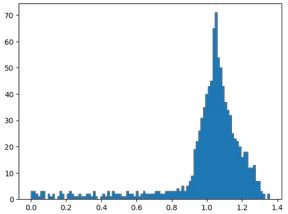
$$L = U \Lambda U^\top$$

Fourier transform on graphs



Diagonalize the Laplacian:

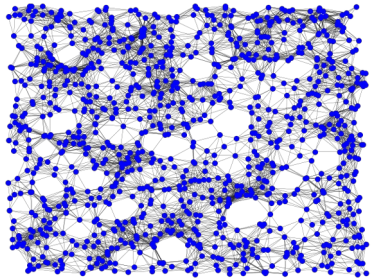
$$L = U \Lambda U^\top$$



$$\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

Eigenvalues: “frequencies” $0 = \lambda_1 \leq \ldots \leq \lambda_n \leq 2$

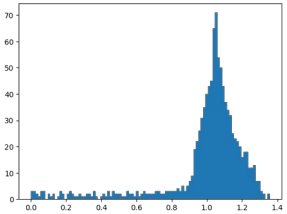
Fourier transform on graphs



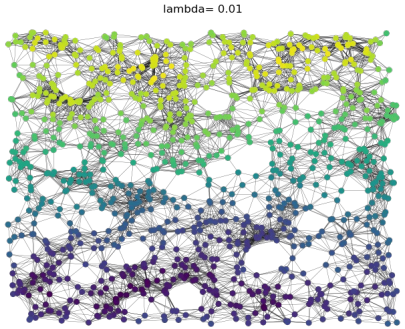
Diagonalize the Laplacian:

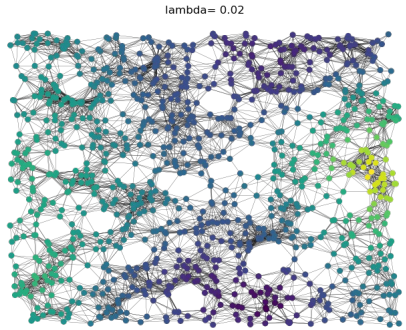
$$L = U \Lambda U^\top$$

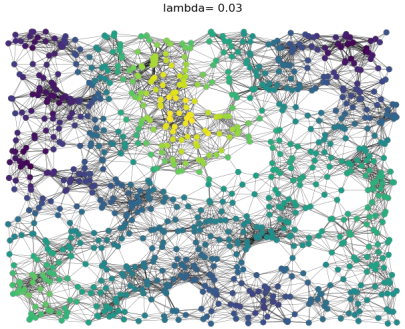
$[u_1, \ldots, u_n]$ Eigenvectors: "Fourier modes"

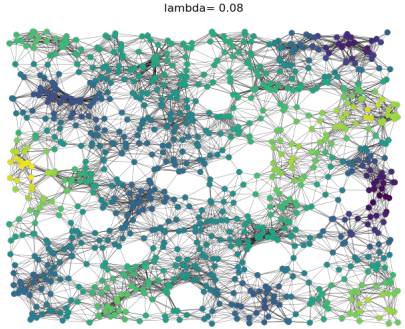


$$\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$



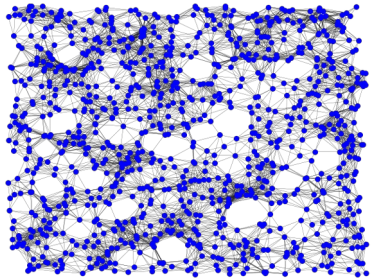






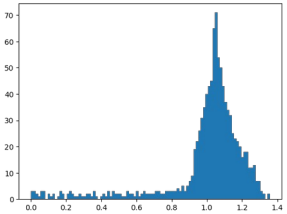
Eigenvalues: "frequencies" $0 = \lambda_1 \leq \ldots \leq \lambda_n \leq 2$

Fourier transform on graphs

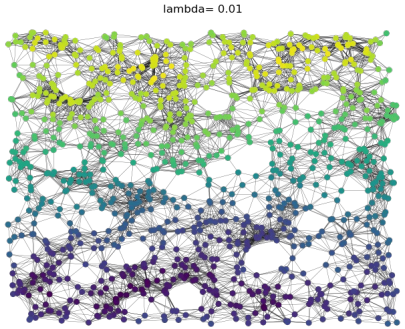


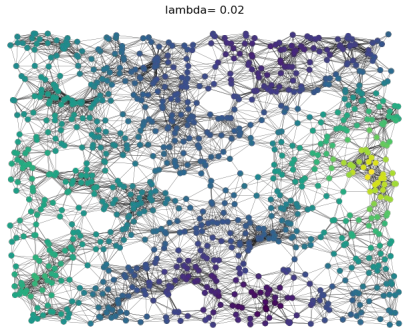
Diagonalize the Laplacian: $[u_1, \ldots, u_n]$ Eigenvectors: "Fourier modes"

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

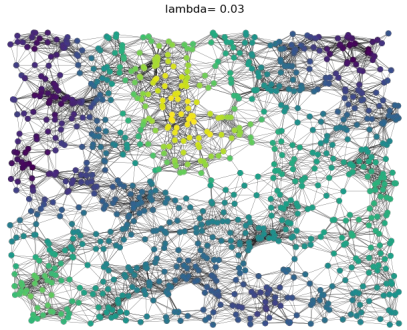


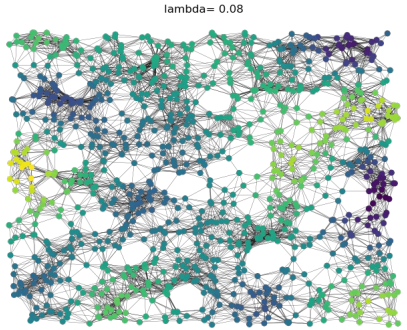
$$\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$





Eigenvalues: "frequencies" $0 = \lambda_1 \leq \ldots \leq \lambda_n \leq 2$

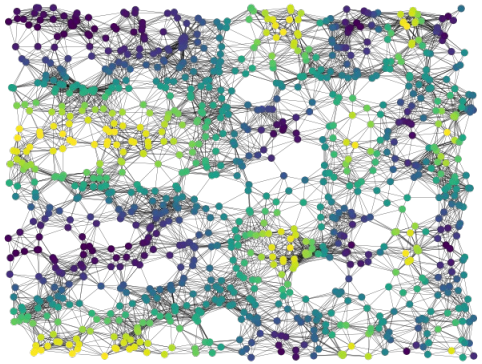




- Fourier transform $\mathcal{F} z = U^\top z$
- Inverse Fourier transform $\mathcal{F}^{-1} \tilde{z} = U \tilde{z}$

Filtering on graphs

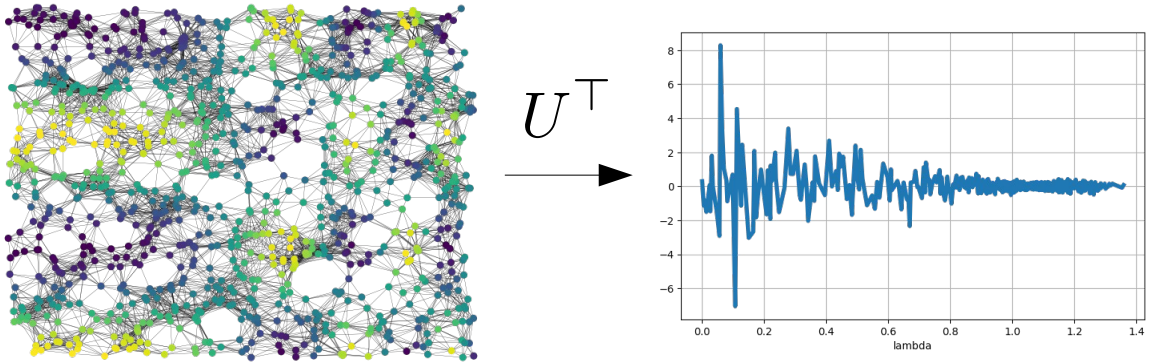
How to filter a signal z ?



Filtering on graphs

How to filter a signal z ?

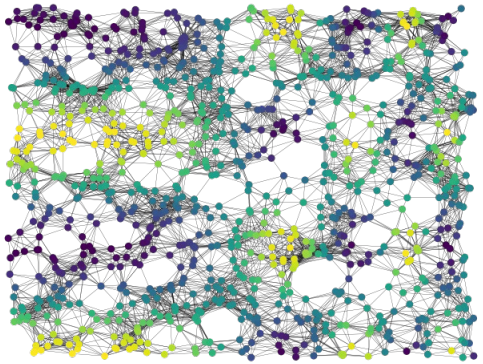
- Compute Fourier transform



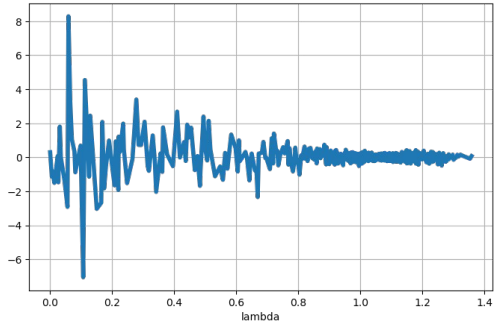
Filtering on graphs

How to filter a signal z ?

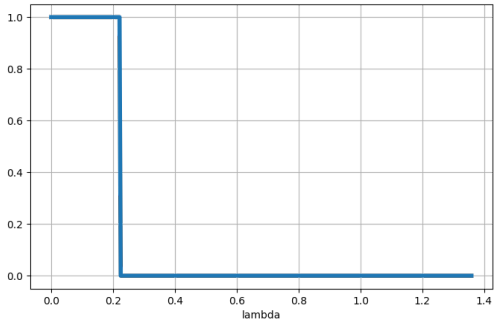
- Compute Fourier transform
- Multiply by filter $h \in \mathbb{R}^n$



$U^\top$



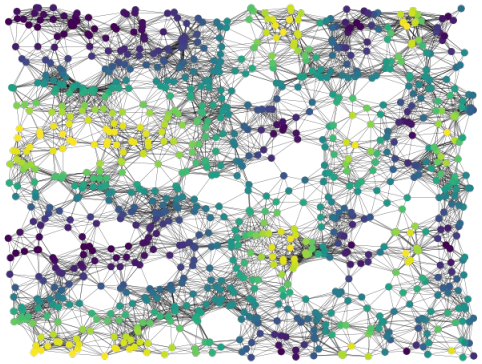
×



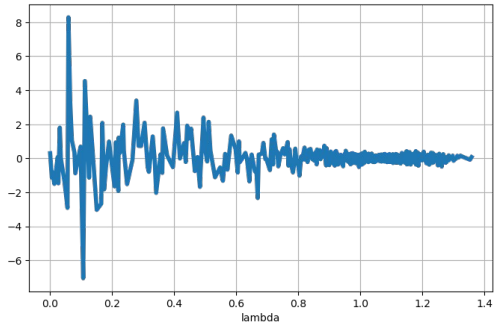
Filtering on graphs

How to filter a signal z ?

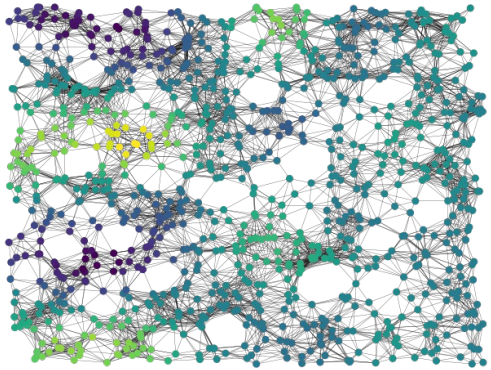
- Compute Fourier transform
- Multiply by filter $h \in \mathbb{R}^n$
- Compute inv. Fourier transform



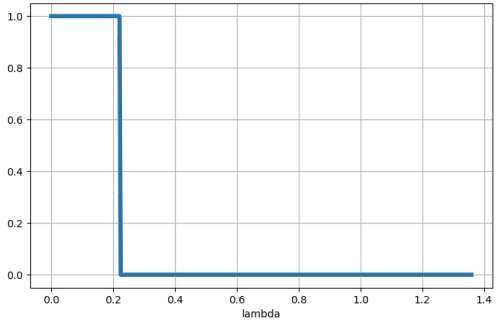
$U^\top$



×



U

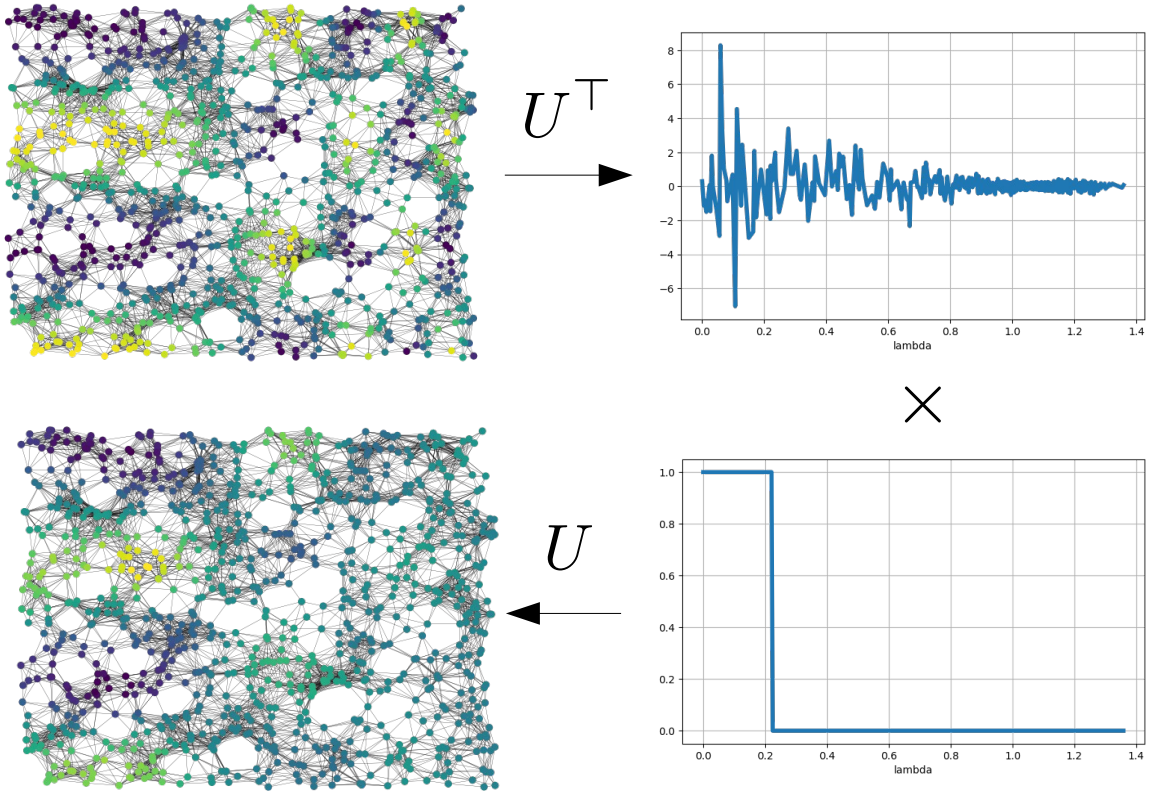


Filtering on graphs

How to filter a signal z ?

- Compute Fourier transform
- Multiply by filter $h \in \mathbb{R}^n$
- Compute inv. Fourier transform

$$(h \star z) = U \mathrm{diag}(h) U^\top z$$



Chung. *Spectral Graph Theory.* (1999)

Shuman et al. *The Emerging Field of Signal Processing on Graphs.* (2013)

Filtering on graphs

- Frequencies and Fourier modes are graph-dependent

Filtering on graphs

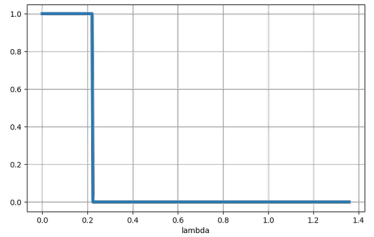
- Frequencies and Fourier modes are graph-dependent
  - Use a function of the frequencies

Filtering on graphs

- Frequencies and Fourier modes are graph-dependent
  - Use a function of the frequencies

$h : \mathbb{R} \to \mathbb{R}$

$$z \star h = U \mathrm{diag}(h(\lambda_i)) U^\top z$$

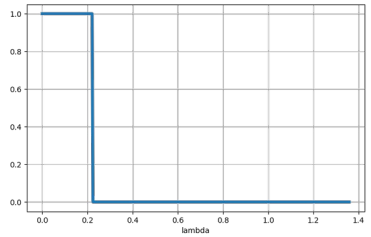


Ex: Low-pass $h(\lambda) = 1_{\lambda \leq \lambda_0}$

Filtering on graphs

- Frequencies and Fourier modes are graph-dependent
  - Use a function of the frequencies

$h : \mathbb{R} \to \mathbb{R}$ $\qquad$ $z \star h = U \mathrm{diag}(h(\lambda_i)) U^\top z$



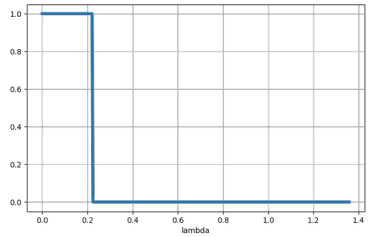
Ex: Low-pass $h(\lambda) = 1_{\lambda \leq \lambda_0}$

- Diagonalization is (very) costly on large graphs!

Filtering on graphs

- Frequencies and Fourier modes are graph-dependent
  - Use a function of the frequencies

$h : \mathbb{R} \to \mathbb{R}$ $\qquad z \star h = U \mathrm{diag}(h(\lambda_i)) U^\top z$



Ex: Low-pass $h(\lambda) = 1_{\lambda \leq \lambda_0}$

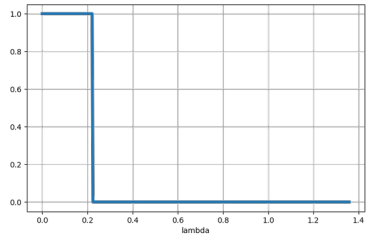
- Diagonalization is (very) costly on large graphs!
  - Use polynomial filters

$h(\lambda) = \sum_k \beta_k \lambda^k$ $\qquad z \star h = h(L) z = \sum_k \beta_k L^k z$

Filtering on graphs

- Frequencies and Fourier modes are graph-dependent
  - Use a function of the frequencies

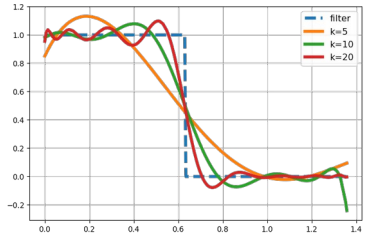
$$h : \mathbb{R} \to \mathbb{R} \qquad z \star h = U \mathrm{diag}(h(\lambda_i)) U^\top z$$



Ex: Low-pass $h(\lambda) = 1_{\lambda \leq \lambda_0}$

- Diagonalization is (very) costly on large graphs!
  - Use polynomial filters

$$h(\lambda) = \sum_k \beta_k \lambda^k \qquad z \star h = h(L) z = \sum_k \beta_k L^k z$$

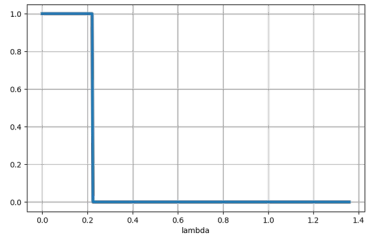


Poly. Approx. of low-pass

Filtering on graphs

- Frequencies and Fourier modes are graph-dependent
  - Use a function of the frequencies

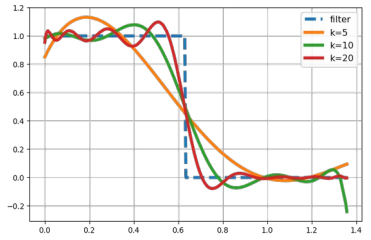
$$h : \mathbb{R} \to \mathbb{R} \qquad z \star h = U \mathrm{diag}(h(\lambda_i)) U^\top z$$



Ex: Low-pass $h(\lambda) = 1_{\lambda \leq \lambda_0}$

- Diagonalization is (very) costly on large graphs!
  - Use polynomial filters

$$h(\lambda) = \sum_k \beta_k \lambda^k \qquad z \star h = h(L) z = \sum_k \beta_k L^k z$$



Poly. Approx. of low-pass

- Filter are "localized"
- Can make use of efficient sparse matrix-vector multiplication

Hammond et al. *Wavelets on Graphs via Spectral Graph Theory.* (2011)

(Spectral) GNNs

Spectral GNN

Henaff et al. *Deep Convolutional Networks on Graph-Structured Data* (2015)

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

(Spectral) GNNs

Spectral GNN

Henaff et al. *Deep Convolutional Networks on Graph-Structured Data* (2015)

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

Non-lin. function (eg ReLU)

Trainable (Polynomial) Filters

Defferrard et al. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering* (2016)

Trainable Bias

(Spectral) GNNs

Spectral GNN

Henaff et al. *Deep Convolutional Networks on Graph-Structured Data* (2015)

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

Non-lin. function (eg ReLU)

Trainable (Polynomial) Filters

Defferrard et al. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering* (2016)

Trainable Bias

- Final layer: output signal over nodes (for node classif) or perform pooling (for graph classif)
  - Respectively permutation equivariant/invariant ***by design***

(Spectral) GNNs

Spectral GNN

Henaff et al. *Deep Convolutional Networks on Graph-Structured Data* (2015)

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

Non-lin. function (eg ReLU)

Trainable (Polynomial) Filters

Defferrard et al. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering* (2016)

Trainable Bias

- Final layer: output signal over nodes (for node classif) or perform pooling (for graph classif)
  - Respectively permutation equivariant/invariant ***by design***
- Early architectures include "graph coarsening" (subsampling) but difficult problem

(Spectral) GNNs

Spectral GNN

Henaff et al. *Deep Convolutional Networks on Graph-Structured Data* (2015)

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

Non-lin. function (eg ReLU)

Trainable (Polynomial) Filters

Defferrard et al. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering* (2016)

Trainable Bias

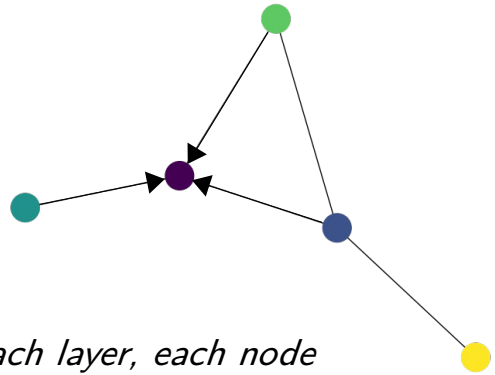
- Final layer: output signal over nodes (for node classif) or perform pooling (for graph classif)
  - Respectively permutation equivariant/invariant ***by design***
- Early architectures include "graph coarsening" (subsampling) but difficult problem
- Need input node feature $Z^{(0)}$. No real solution otherwise...

Duong et al. *On Node Features for Graph Neural Networks* (2019)

Wijesinghe et al. *Graph Self-Supervised Learning with Learnable Structural and Positional Encodings* (2025)

The message-passing paradigm

"Modern" GNNs are often built around a message-passing interpretation.



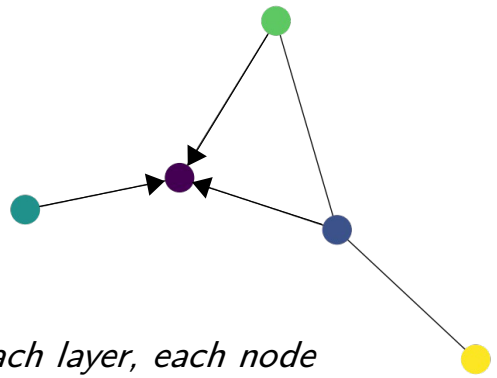
At each layer, each node receives "messages" from its neighbors.

Gilmer et al. *Neural Message Passing for Quantum Chemistry.* (2017)

$$x_i^{(\ell)} = \text{AGGREGATE}\left(x_i^{(\ell-1)}, \{x_j^{(\ell-1)}, e_{ij} \in E\}\right)$$

The message-passing paradigm

"Modern" GNNs are often built around a message-passing interpretation.



At each layer, each node receives "messages" from its neighbors.

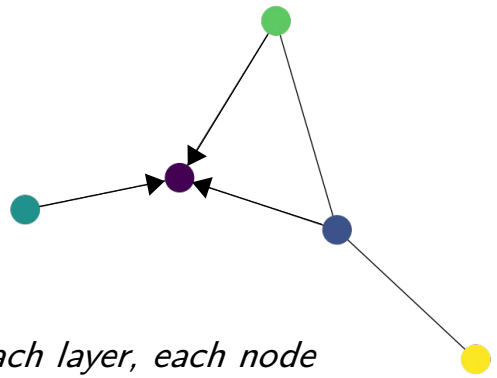
Gilmer et al. *Neural Message Passing for Quantum Chemistry.* (2017)

$$x_i^{(\ell)} = \text{AGGREGATE}\left(x_i^{(\ell-1)}, \{x_j^{(\ell-1)}, e_{ij} \in E\}\right)$$

- Messages are treated as a **set**: no node ordering!

The message-passing paradigm

"Modern" GNNs are often built around a message-passing interpretation.

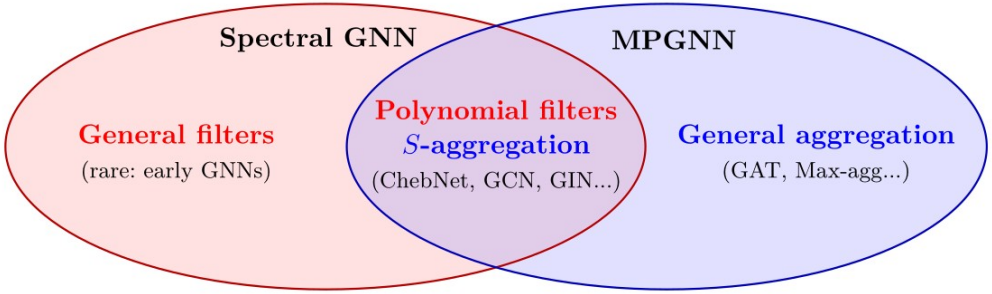


At each layer, each node receives "messages" from its neighbors.

Gilmer et al. *Neural Message Passing for Quantum Chemistry.* (2017)

$$x_i^{(\ell)} = \text{AGGREGATE}\left(x_i^{(\ell-1)}, \{x_j^{(\ell-1)}, e_{ij} \in E\}\right)$$

- Messages are treated as a **set**: no node ordering!
- When AGG is SUM/MEAN: order-1 polynomial filter!!
- Popular: **attention-based** (GAT)



Outline

1. From Deep Convolutional Networks to GNNs

2. Some (theoretical) properties of GNNs

Expressive power of GNN

- Classical DNN are "universal": as the number of neurons grow, they can approximate any continuous function. What about GNNs?

Hornik et al. *Multilayer Feedforward Networks are Universal Approximators* (1989)
Cybenko. *Approximation by superpositions of a sigmoidal function* (1989)

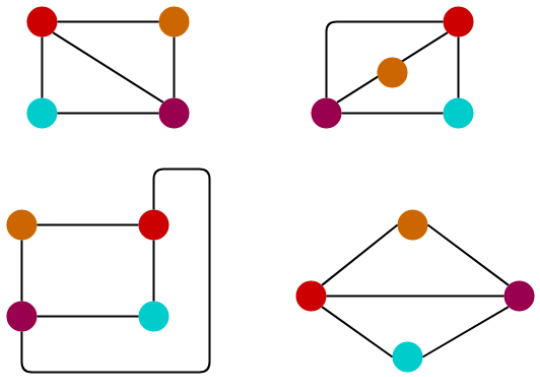
Expressive power of GNN

- Classical DNN are "universal": as the number of neurons grow, they can approximate any continuous function. What about GNNs?

  Hornik et al. *Multilayer Feedforward Networks are Universal Approximators* (1989)
  Cybenko. *Approximation by superpositions of a sigmoidal function* (1989)

- "Graph-classif" GNN are **permutation-invariant**, aka insensitive to graph isomorphism
  - "Node-classif" GNN are ***permutation-equivariant***

$$G \sim G' \quad \Leftrightarrow \quad \exists \sigma \in \mathcal{P}_n : A = \sigma^\top A' \sigma$$



Graph Isomorphism Example

Expressive power of GNN

- Classical DNN are "universal": as the number of neurons grow, they can approximate any continuous function. What about GNNs?

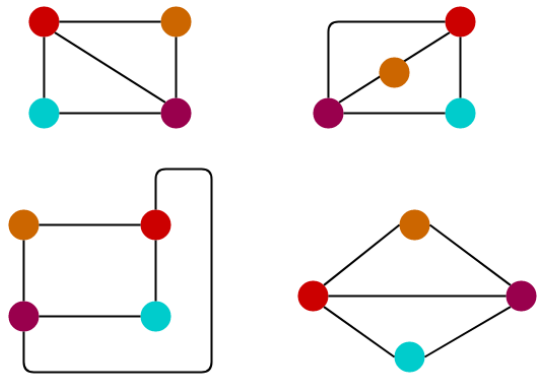
Hornik et al. *Multilayer Feedforward Networks are Universal Approximators* (1989)
Cybenko. *Approximation by superpositions of a sigmoidal function* (1989)

- "Graph-classif" GNN are **permutation-invariant**, aka insensitive to graph isomorphism
  - "Node-classif" GNN are ***permutation-equivariant***

$$G \sim G' \quad \Leftrightarrow \quad \exists \sigma \in \mathcal{P}_n : A = \sigma^\top A' \sigma$$

Graph isomorphism problem:

- *No known polynomial algorithm. Best:* $O\left(e^{(\log n)^{O(1)}}\right)$
- *Not known if NP-complete*
- *Might be a class of complexity on its own!*



Graph Isomorphism Example

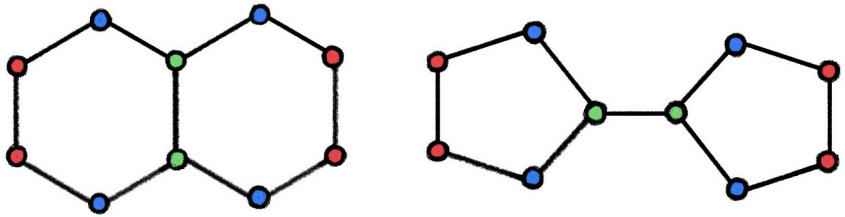
Babai. *Graph Isomorphism in Quasipolynomial Time* (2015)

GNN vs. WL

- A classical algorithm for graph isomorphism is the Weisfeiler-Lehman test.
  - Starts with arbitrary labelling of nodes (among discrete set)
  - Propagate labels with injective agg. function, repeat n times, and compares final sets of labels.

GNN vs. WL

- A classical algorithm for graph isomorphism is the Weisfeiler-Lehman test.
  - Starts with arbitrary labelling of nodes (among discrete set)
  - Propagate labels with injective agg. function, repeat n times, and compares final sets of labels.
  - Can distinguish a "large-class" of non-isomorphic graphs (but not all!)

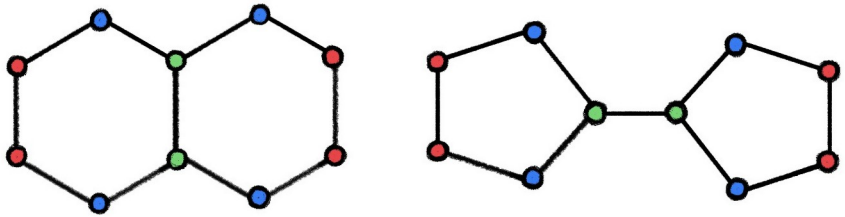


WL fails here...

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

- A classical algorithm for graph isomorphism is the Weisfeiler-Lehman test.
  - Starts with arbitrary labelling of nodes (among discrete set)
  - Propagate labels with injective agg. function, repeat n times, and compares final sets of labels.
  - Can distinguish a "large-class" of non-isomorphic graphs (but not all!)



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)

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

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

Beyond WL...

- GNN expressivity can be improved...

Beyond WL...

- GNN expressivity can be improved...
  - to be as powerful as "higher-order WL"

Maron et al. *Provably Powerful Graph Networks* (2019)
Chen et al. *On the equivalence between graph isomorphism testing and function approximation with GNNs* (2019)

Beyond WL...

- GNN expressivity can be improved...
  - to be as powerful as "higher-order WL"
  - by giving nodes unique/random identifiers

Maron et al. *Provably Powerful Graph Networks* (2019)
Chen et al. *On the equivalence between graph isomorphism testing and function approximation with GNNs* (2019)

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

Beyond WL...

- GNN expressivity can be improved...
  - to be as powerful as "higher-order WL"
  - by giving nodes unique/random identifiers
  - by counting/sampling substructures or other "Positional Encodings"

Maron et al. *Provably Powerful Graph Networks* (2019)
Chen et al. *On the equivalence between graph isomorphism testing and function approximation with GNNs* (2019)

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

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

Beyond WL...

- GNN expressivity can be improved...
  - to be as powerful as "higher-order WL"
  - by giving nodes unique/random identifiers
  - by counting/sampling substructures or other "Positional Encodings"
- True universality can be attained by allowing unbounded "tensorization" order
  - Far too expensive to implement in practice...

Maron et al. *Provably Powerful Graph Networks* (2019)
Chen et al. *On the equivalence between graph isomorphism testing and function approximation with GNNs* (2019)

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

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

Maron et al. *On the Universality of Invariant Networks* (2019)
Keriven and Peyré. *Universal Invariant and Equivariant Graph Neural Network.* (2019)

Beyond WL...

- GNN expressivity can be improved...
  - to be as powerful as "higher-order WL"
  - by giving nodes unique/random identifiers
  - by counting/sampling substructures or other "Positional Encodings"
- True universality can be attained by allowing unbounded "tensorization" order
  - Far too expensive to implement in practice...
- GI test improved to different notions of "expressivity"
  - But combinatorial "expressivity" is still a difficult topic

Maron et al. *Provably Powerful Graph Networks* (2019)
Chen et al. *On the equivalence between graph isomorphism testing and function approximation with GNNs* (2019)

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

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

Maron et al. *On the Universality of Invariant Networks* (2019)
Keriven and Peyré. *Universal Invariant and Equivariant Graph Neural Network.* (2019)

Boker et al. *Fine-grained Expressivity of Graph Neural Networks* (2023)
Loukas. *What graph neural networks cannot learn: depth vs width* (2020)

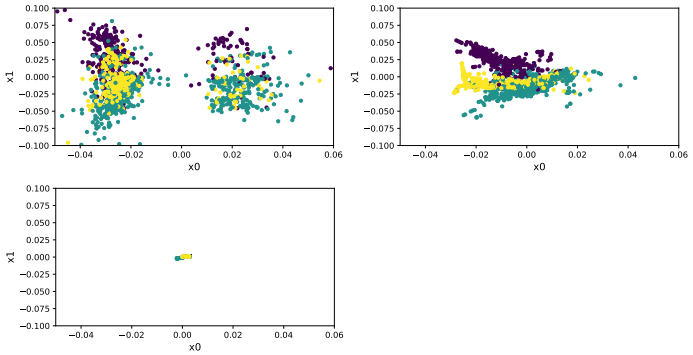
Depth and oversmoothing

- Most GNNs are quite shallow: 4 to 10 layers

Depth and oversmoothing

- Most GNNs are quite shallow: 4 to 10 layers
- Often explained by "oversmoothing":
  - Each MP layer tends to be "low-pass" in frequency
  - Depth **smoothes out** the node representations

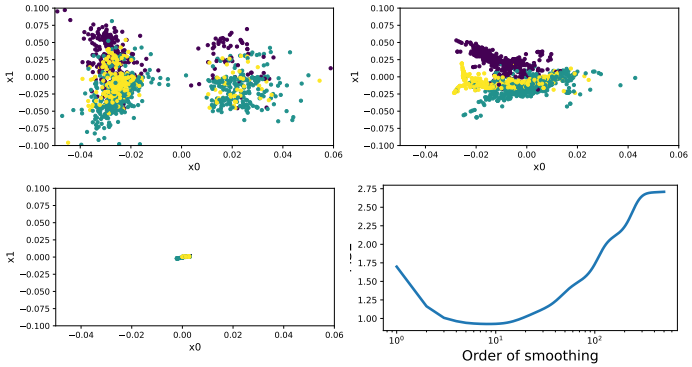
Li et al. *Deeper insights into graph convolutional networks for semi-supervised learning* (2018)

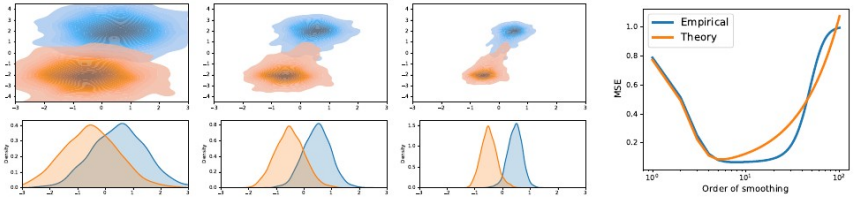


Depth and oversmoothing

- Most GNNs are quite shallow: 4 to 10 layers
- Often explained by "oversmoothing":
  - Each MP layer tends to be "low-pass" in frequency
  - Depth **smoothes out** the node representations
  - We still need depth!

Li et al. *Deeper insights into graph convolutional networks for semi-supervised learning* (2018)





Keriven. *Not too little, not too much: a theoretical analysis of graph (over)smoothing* (2022)

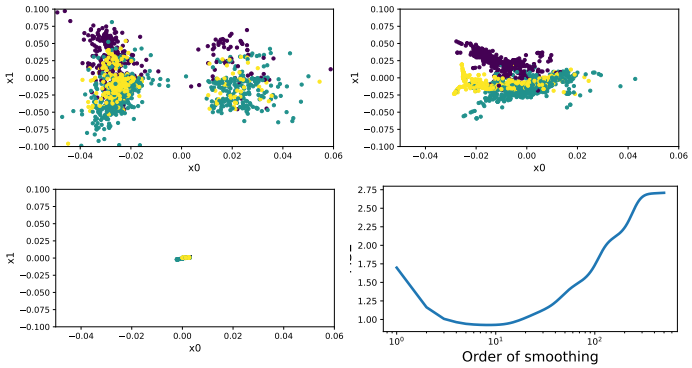
Depth and oversmoothing

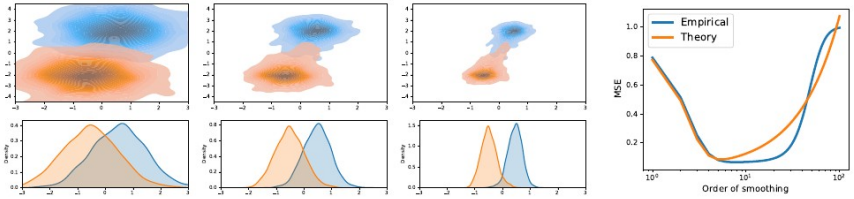
- Most GNNs are quite shallow: 4 to 10 layers
- Often explained by "oversmoothing":
  - Each MP layer tends to be "low-pass" in frequency
  - Depth **smoothes out** the node representations
  - We still need depth!

- Mitigation: normalization, skip-connection...

Zhao et al. *Pairnorm: Tackling Oversmoothing in GNNs* (2020)
Li et al. *Training Graph Neural Networks with 1000 Layers* (2021)
Rusch et al. *A Survey on Oversmoothing in Graph Neural Networks* (2023)

Li et al. *Deeper insights into graph convolutional networks for semi-supervised learning* (2018)





Keriven. *Not too little, not too much: a theoretical analysis of graph (over)smoothing* (2022)

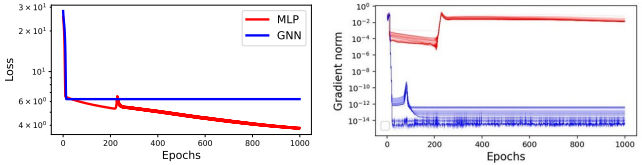
Depth and oversquashing

- We moved "beyond" vanilla oversmoothing, but depth remains an issue!

Kormann et al. *Position: Don't be Afraid of Over-Smoothing And Over-Squashing* (2026)

Depth and oversquashing

- We moved "beyond" vanilla oversmoothing, but depth remains an issue!
  - **Optimization** issue?

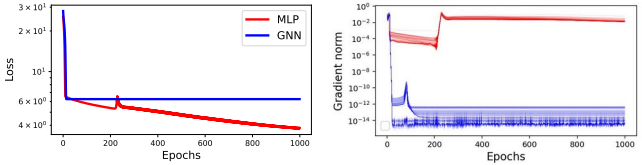


Kormann et al. *Position: Don't be Afraid of Over-Smoothing And Over-Squashing* (2026)

Keriven. *Backward Oversmoothing: why is it hard to train deep Graph Neural Networks?* (2026)

Depth and oversquashing

- We moved "beyond" vanilla oversmoothing, but depth remains an issue!
  - **Optimization** issue?
  - Other? (do we need depth?)
  - Long/short range, homophily/heterophily...



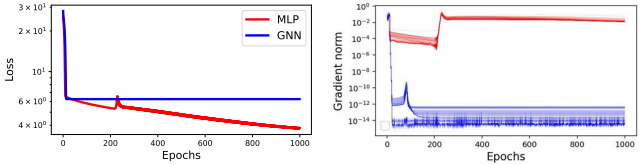
Kormann et al. *Position: Don't be Afraid of Over-Smoothing And Over-Squashing* (2026)

Keriven. *Backward Oversmoothing: why is it hard to train deep Graph Neural Networks?* (2026)

Zheng et al. *Graph Neural Networks for Graphs with Heterophily: A Survey* (2022)

Depth and oversquashing

- We moved "beyond" vanilla oversmoothing, but depth remains an issue!
  - **Optimization** issue?
  - Other? (do we need depth?)
  - Long/short range, homophily/heterophily...



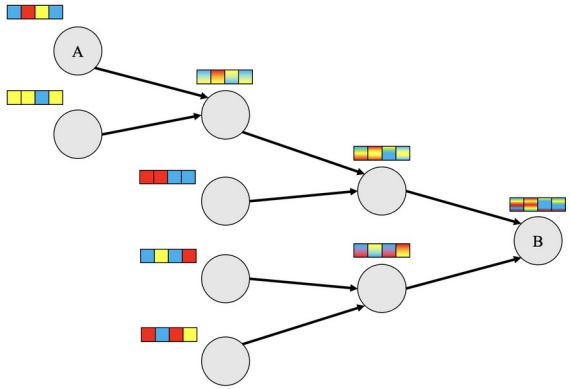
- Other depth issue: "over-squashing"
  - Node representations should **"represent" their neighborhood**
  - But it generally **grow exponentially** with depth!
  - A "large" quantity of (potentially irrelevant) information is "**squashed**" into a **fixed-size representation**
  - Mitigation: *graph rewiring...*

Kormann et al. *Position: Don't be Afraid of Over-Smoothing And Over-Squashing* (2026)

Keriven. *Backward Oversmoothing: why is it hard to train deep Graph Neural Networks?* (2026)

Zheng et al. *Graph Neural Networks for Graphs with Heterophily: A Survey* (2022)

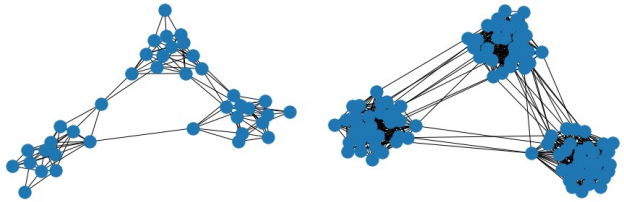
Topping et al. *Understanding Over-Squashing and Bottlenecks on Graphs Via Curvature* (2021)



Balla. *Over-squashing in Graph Neural Networks* (2021)

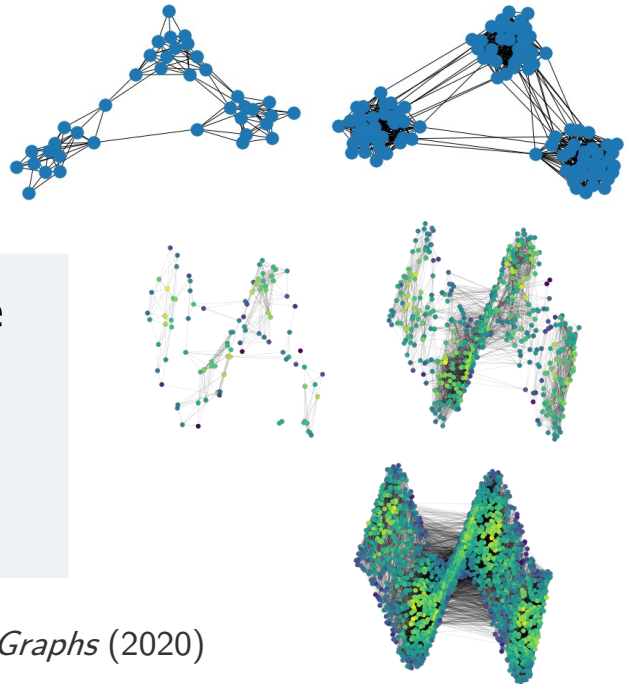
Behavior on large graphs

- The **"graph isomorphism"** point of view is very limited
- More generally, **"combinatorial" quantities"** (eg edit distance) are not really appropriate for large graphs that "look the same"



Behavior on large graphs

- The **"graph isomorphism"** point of view is very limited
- More generally, **"combinatorial" quantities"** (eg edit distance) are not really appropriate for large graphs that "look the same"

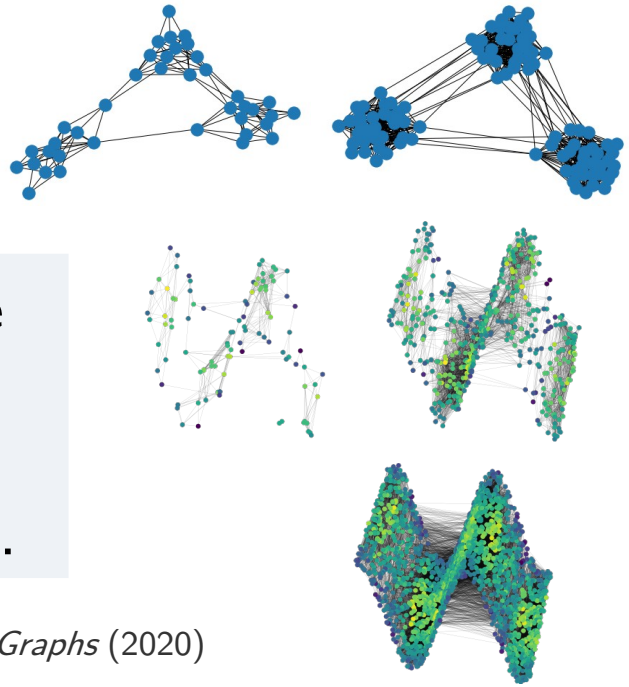


- GNNs on large (random) graphs may converges to some **"continuous" limit**

Keriven et al. *Convergence and Stability of Graph Convolutional Networks on Large Random Graphs* (2020)
Keriven et al. *On the Universality of Graph Neural Networks on Large Random Graphs* (2021)
Keriven, Vaiter. *What functions can Graph Neural Networks compute on random graphs? The role of Positional Encoding* (2023)
Levie. *A graphon-signal analysis of graph neural networks* (2023)
Ruiz et al. *Graphon neural networks and the transferability of graph neural networks* (2020)

Behavior on large graphs

- The **"graph isomorphism"** point of view is very limited
- More generally, **"combinatorial" quantities"** (eg edit distance) are not really appropriate for large graphs that "look the same"



- GNNs on large (random) graphs may converges to some **"continuous" limit**
  - **Known** notions: discrete Laplacian -> continuous Laplacian!
  - Allows to study notions of **transferability, stability, universality...**

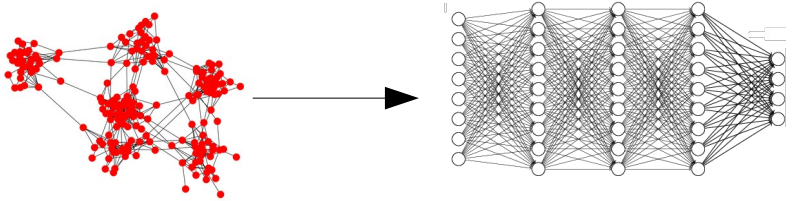
Keriven et al. *Convergence and Stability of Graph Convolutional Networks on Large Random Graphs* (2020)
Keriven et al. *On the Universality of Graph Neural Networks on Large Random Graphs* (2021)
Keriven, Vaiter. *What functions can Graph Neural Networks compute on random graphs? The role of Positional Encoding* (2023)
Levie. *A graphon-signal analysis of graph neural networks* (2023)
Ruiz et al. *Graphon neural networks and the transferability of graph neural networks* (2020)

Generalization

- In ML, generalization heavily (entirely) relies on ***i.i.d. samples***

Generalization

- In ML, generalization heavily (entirely) relies on ***i.i.d. samples***



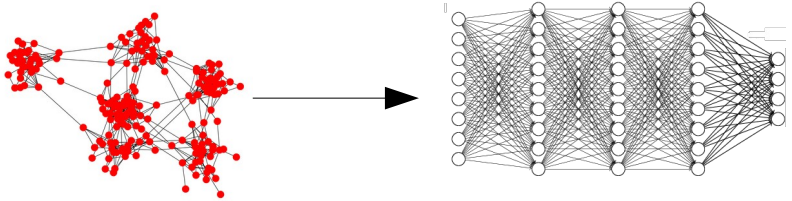
$k = 6$

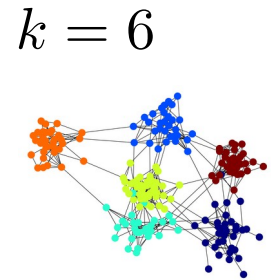
Graph-level: **OK**



Generalization

- In ML, generalization heavily (entirely) relies on ***i.i.d. samples***



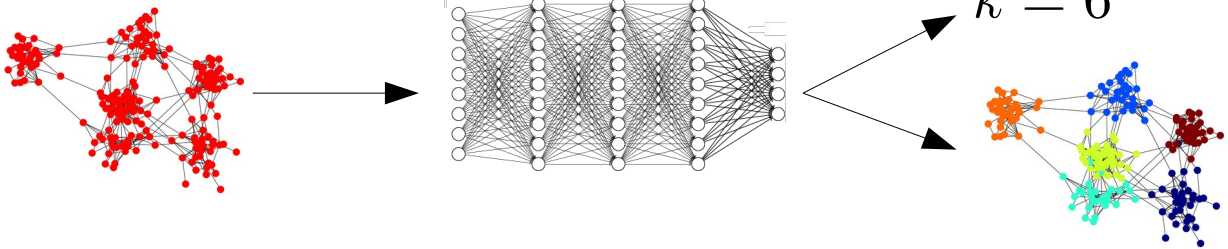


Graph-level: **OK** ✓

Node level: **Hmm** 🤔

Generalization

- In ML, generalization heavily (entirely) relies on ***i.i.d. samples***



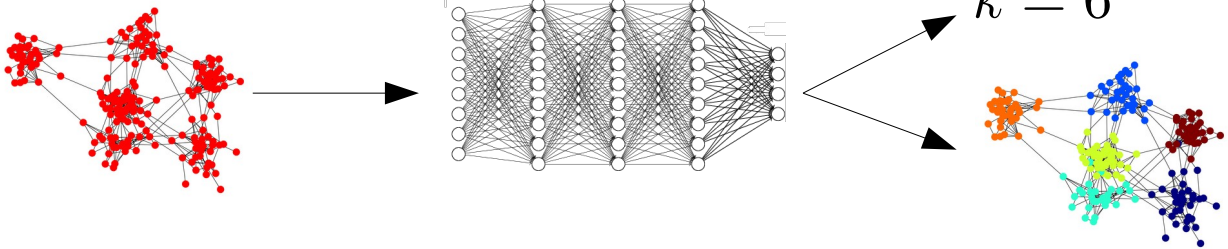
Graph-level: **OK** ✓

Node level: **Hmm** 🤔

- Most theoretical works are about **graph-level** generalization
  - *Rademacher, VC dimension...*

Generalization

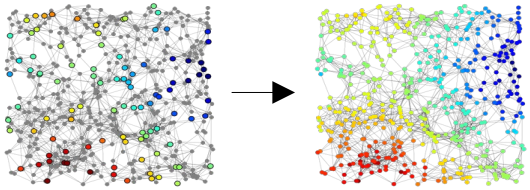
- In ML, generalization heavily (entirely) relies on ***i.i.d. samples***



Graph-level: **OK** ✓

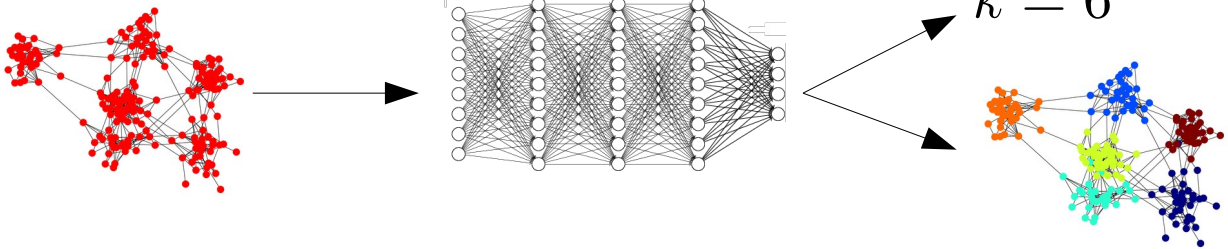
Node level: **Hmm** 🤔

- Most theoretical works are about **graph-level** generalization
  - *Rademacher, VC dimension...*
- Most node-level generalization are about **transductive** learning



Generalization

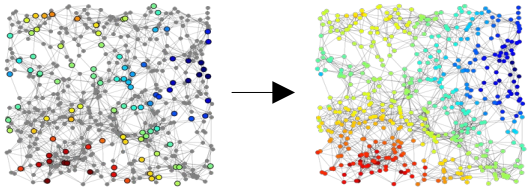
- In ML, generalization heavily (entirely) relies on ***i.i.d. samples***



Graph-level: **OK** ✓

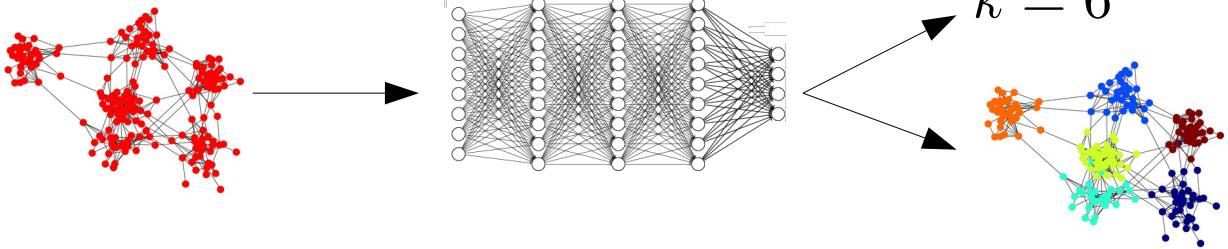
Node level: **Hmm** 🤔

- Most theoretical works are about **graph-level** generalization
  - *Rademacher, VC dimension...*
- Most node-level generalization are about **transductive** learning
- Some inductive node-level still assume **large number of graphs**



Generalization

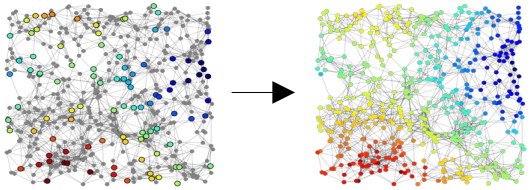
- In ML, generalization heavily (entirely) relies on ***i.i.d. samples***



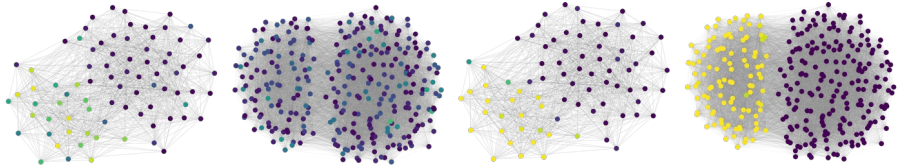
Graph-level: **OK** ✓

Node level: **Hmm** 🤔

- Most theoretical works are about **graph-level** generalization
  - *Rademacher, VC dimension...*
- Most node-level generalization are about **transductive** learning
- Some inductive node-level still assume **large number of graphs**
- What about *inductive from a single large graph?*



Keriven and Viegas. *A theory of inductive node-level generalization from a single large graph*. (Hopefully 2026)



Conclusion

- Graph ML and GNN are now well-established fields in ML
- Mostly "engineering/computer-science" driven, some blind spots (statistics, probability...)
- Still a lot to do! ("low-hanging fruits")
- Important to remain critical: **graph are often not the go-to solution!**

Conclusion

- Graph ML and GNN are now well-established fields in ML
- Mostly "engineering/computer-science" driven, some blind spots (statistics, probability...)
- Still a lot to do! ("low-hanging fruits")
- Important to remain critical: **graph are often not the go-to solution!**

- "Vanilla" GNNs are ***evolving***: in terms of **applications**, models...
- Still "human-sized" community and applications, still fun!
- Coming up with **real challenging applications/datasets** is still an active area of research
- Many open questions
  - including many in "non-deep" graph ML
  - Including very basic ones

Conclusion

- Graph ML and GNN are now well-established fields in ML
- Mostly "engineering/computer-science" driven, some blind spots (statistics, probability...)
- Still a lot to do! ("low-hanging fruits")
- Important to remain critical: **graph are often not the go-to solution!**

- "Vanilla" GNNs are ***evolving***: in terms of **applications**, models...
- Still "human-sized" community and applications, still fun!
- Coming up with **real challenging applications/datasets** is still an active area of research
- Many open questions
  - including many in "non-deep" graph ML
  - Including very basic ones



nkeriven.github.io



