Graph Signal Processing for Machine Learning

A Review and New Perspectives

Xiaowen Dong, Dorina Thanou, Laura Toni,
Michael Bronstein, Pascal Frossard

ICASSP Tutorial, June 2021

Part II
GSP for ML: Additional challenges

ICASSP Tutorial, June 2021

Outline

- Brief introduction to graph signal processing (GSP)
- Key GSP tools for machine learning
- Challenge I: GSP for exploiting data structure
- Challenge II: GSP for improving efficiency and robustness
- Challenge III: GSP for enhancing model interpretability
- Applications
- Summary, open challenges, and new perspectives

Outline

- Brief introduction to graph signal processing (GSP)
- Key GSP tools for machine learning
- Challenge I: GSP for exploiting data structure
- Challenge II: GSP for improving efficiency and robustness
- Challenge III: GSP for enhancing model interpretability
- Applications
- Summary, open challenges, and new perspectives

Outline

- Brief introduction to graph signal processing (GSP)
- Key GSP tools for machine learning
- Challenge I: GSP for exploiting data structure
- Challenge II: GSP for improving efficiency and robustness
- Challenge III: GSP for enhancing model interpretability
- Applications
- Summary, open challenges, and new perspectives

GSP for improving efficiency and robustness

A. Improvement on data efficiency and robustness

B. Robustness against topological noise

C. Improvement on computational efficiency

GSP for improving efficiency and robustness

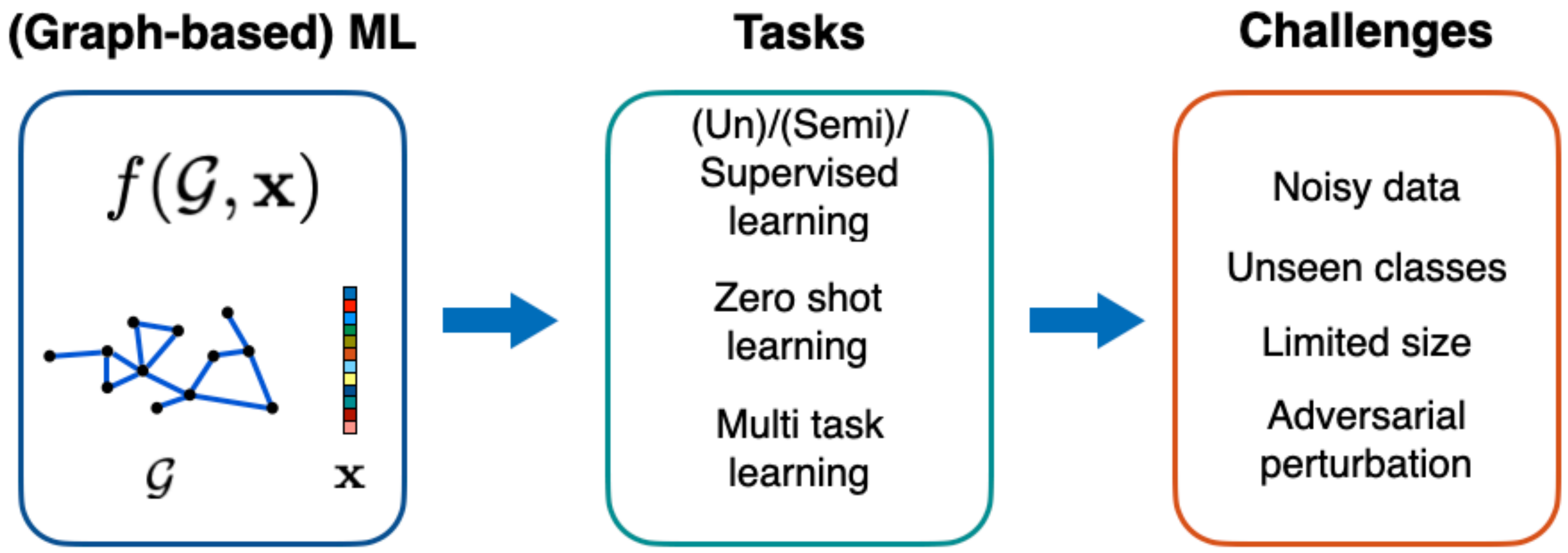
A. Improvement on data efficiency and robustness

B. Robustness against topological noise

C. Improvement on computational efficiency

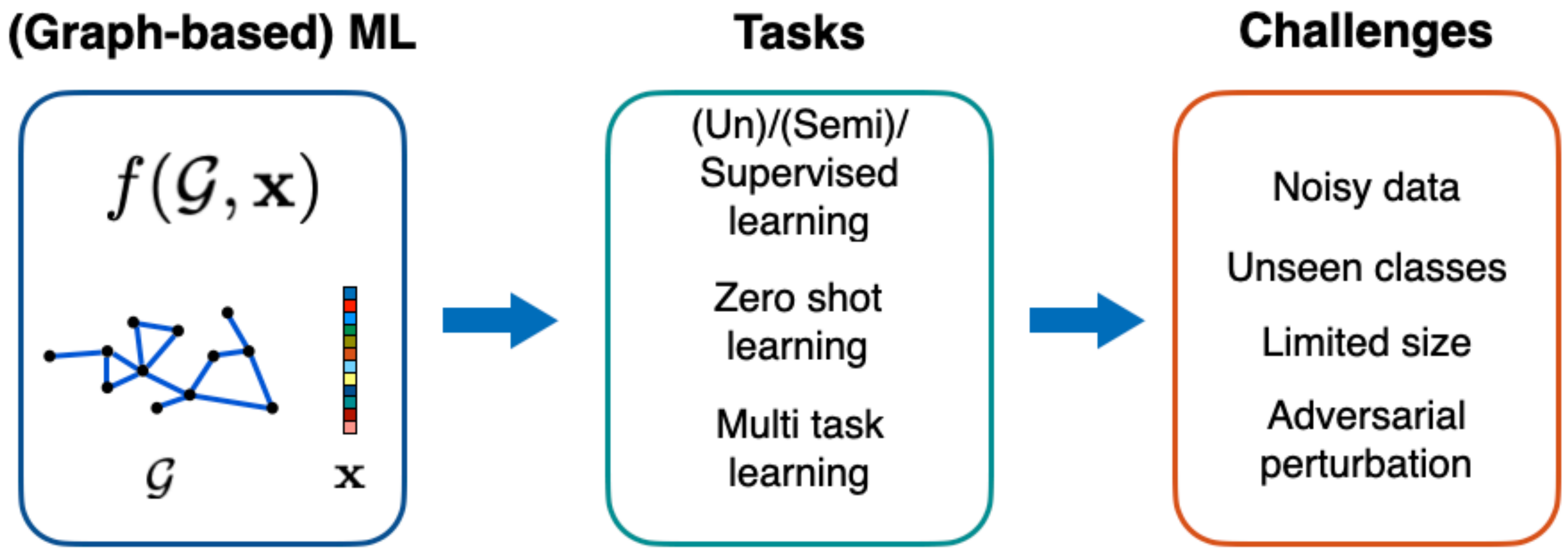
A. Improvement on data efficiency and robustness

- In many classical ML applications, data is scarce or noisy



A. Improvement on data efficiency and robustness

- In many classical ML applications, data is scarce or noisy



- Need for prior knowledge to improve performance:
  - Graph-based regularization could help in that direction

A.1. Graph signal smoothness as loss function

- Cross entropy loss is widely used in (semi-)supervised learning frameworks. However:
  - Underlying manifold structure is not preserved: Inputs of the same class tend to be mapped to the same output
  - One-hot-bit encoding is independent of the input distribution and network initialization: slow training process

A.1. Graph signal smoothness as loss function

- Cross entropy loss is widely used in (semi-)supervised learning frameworks. However:
  - Underlying manifold structure is not preserved: Inputs of the same class tend to be mapped to the same output
  - One-hot-bit encoding is independent of the input distribution and network initialization: slow training process

- An alternative GSP inspired loss function:
  - Maximizes the distance between images of distinct classes
  - Minimizes a loss based on the smoothness of the label signals on a similarity graph

A.1. Graph signal smoothness as loss function

- Cross entropy loss is widely used in (semi-)supervised learning frameworks. However:
  - Underlying manifold structure is not preserved: Inputs of the same class tend to be mapped to the same output
  - One-hot-bit encoding is independent of the input distribution and network initialization: slow training process

- An alternative GSP inspired loss function:
  - Maximizes the distance between images of distinct classes
  - Minimizes a loss based on the smoothness of the label signals on a similarity graph

$$S_G(s_c) = s_c^T L s_c = \sum_{u,v \in V} W_{v,u}\big(s_c(v) - s_c(u)\big)^2$$

Smoothness of label signal for class c

A.1. Graph signal smoothness as loss function

- Cross entropy loss is widely used in (semi-)supervised learning frameworks. However:
  - Underlying manifold structure is not preserved: Inputs of the same class tend to be mapped to the same output
  - One-hot-bit encoding is independent of the input distribution and network initialization: slow training process

- An alternative GSP inspired loss function:
  - Maximizes the distance between images of distinct classes
  - Minimizes a loss based on the smoothness of the label signals on a similarity graph

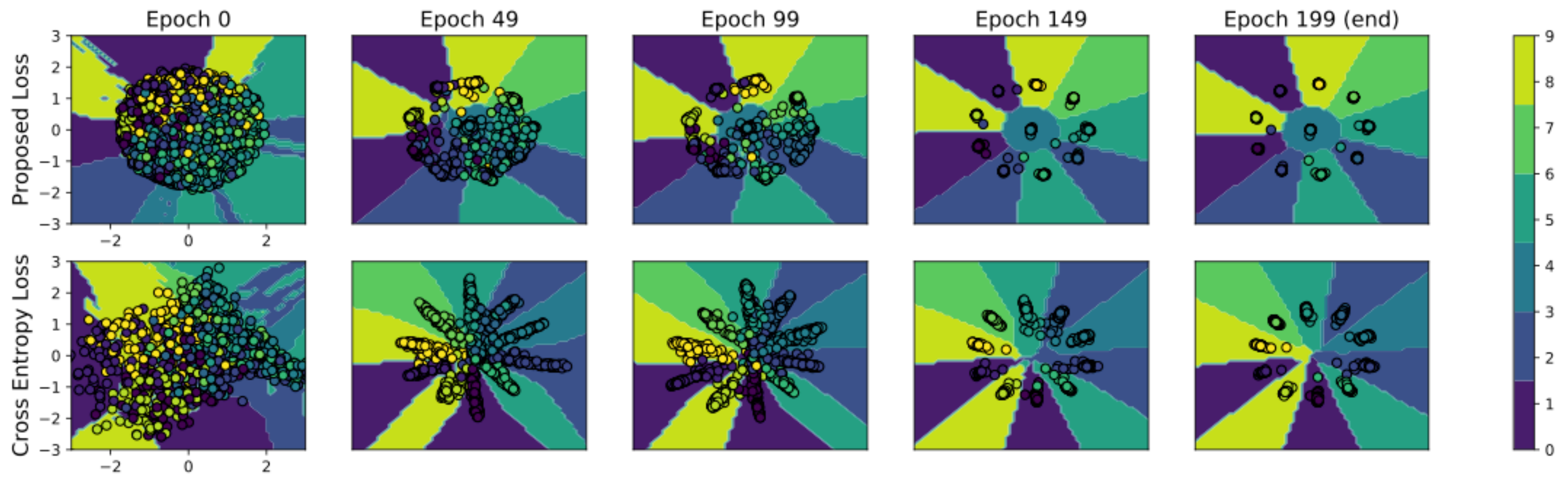
$$S_G(s_c) = s_c^T L s_c = \sum_{u,v \in V} W_{v,u}\big(s_c(v) - s_c(u)\big)^2$$

Smoothness of label signal for class c

$$\mathcal{L}_{G_k}(f) = \sum_{c=1}^{C} S_G(s_c) = \sum_{s_c(v)s_c(u)=0, \forall c} exp(-\alpha \| f(x_v) - f(x_u) \|)$$

Experimental validation

- Clustered like embeddings for CIFAR-10



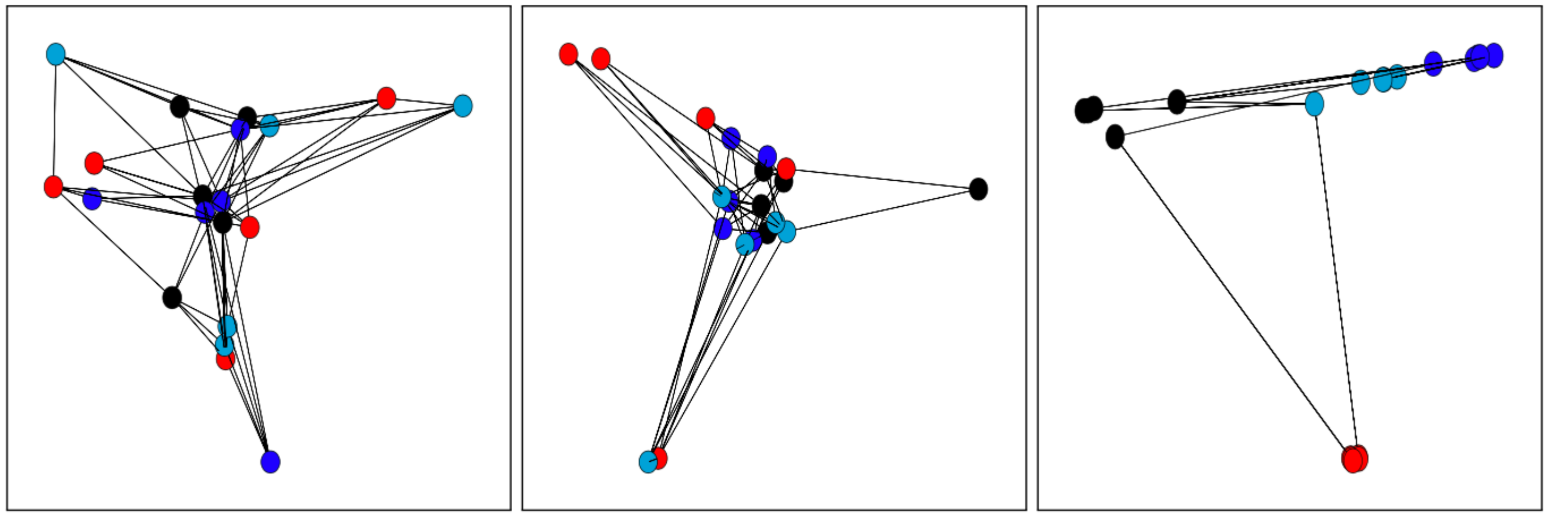
- Robustness to cluster deviations

Method	Clean test error	MCE	relative MCE
Cross-entropy	**5.06%**	100	100
Proposed	5.60%	**95.28**	**90.33**

Bontonou et al., “Introducing graph smoothness loss for training deep learning architectures,” IEEE Data Science Workshop 2019

A.2. Latent geometry graphs

- Capture the latent geometry of intermediate layers of deep networks using graphs
- Quantify the expressive power of each layer by measuring the label variation



(a) Input σ : 0.48 (b) Middle σ : 0.17 (c) Penultimate σ : 0.05

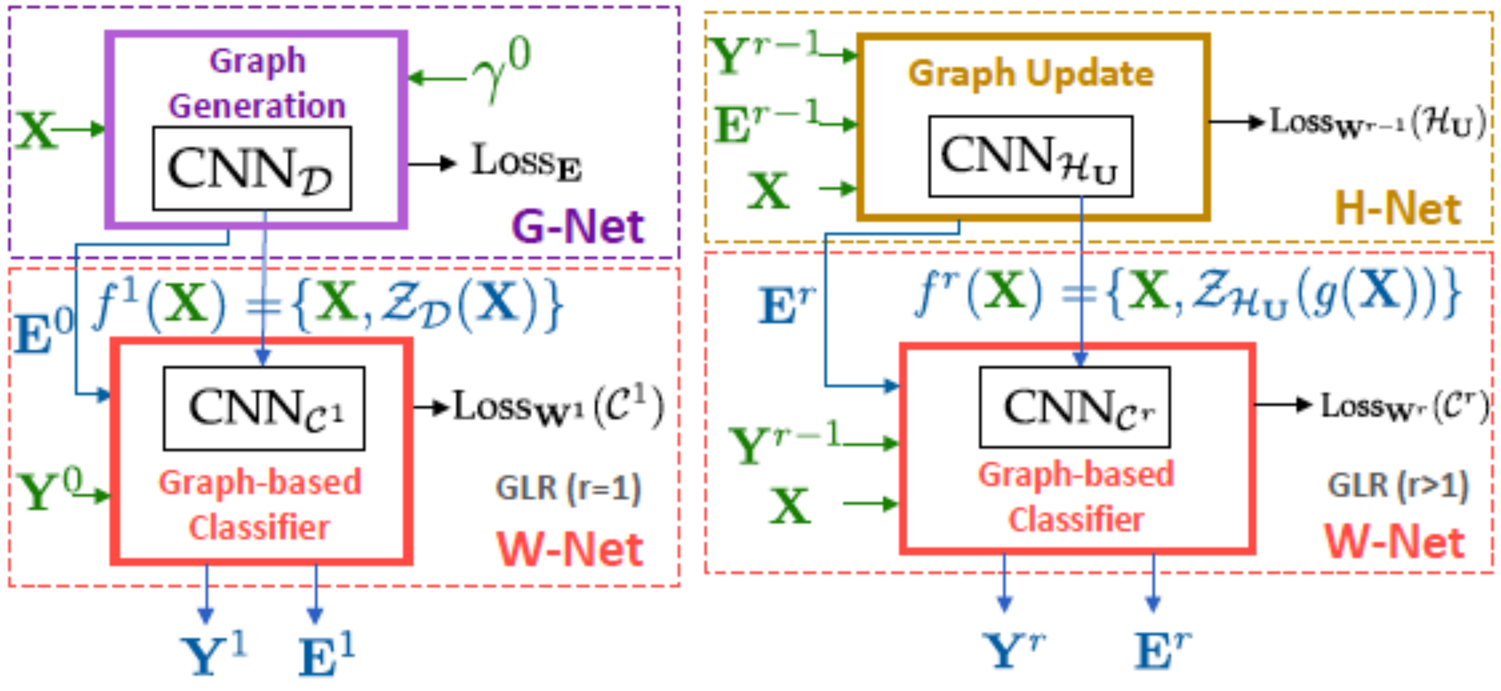
- Impose desirable properties into these representations by applying GSP constraints
  - Robustness: Enforce smooth label variations between consecutive layers

Lassance et al., "Representing Deep Neural Networks Latent Space Geometries with Graphs ", Algorithms, 2021
Lassance et al., "Laplacian networks: bounding indicator function smoothness for neural networks robustness", APSIPA Trans. on Sign. and Infor. Process., 2021

A.3. Graph regularization for dealing with noisy labels

- Main challenge: Binary classification with noisy labels

- Proposed approach (DynGLR): A two step learning process
  - **Graph learning:** extract deep feature maps and learn a graph that maximizes/ minimizes similarity between two nodes that have same/opposite labels (G-Net)
  - **Classifier learning:** alternate between refining the graph (H-Net) and restoring the corrupted classifier signal (W-Net) through graph Laplacian regularization (GLR)

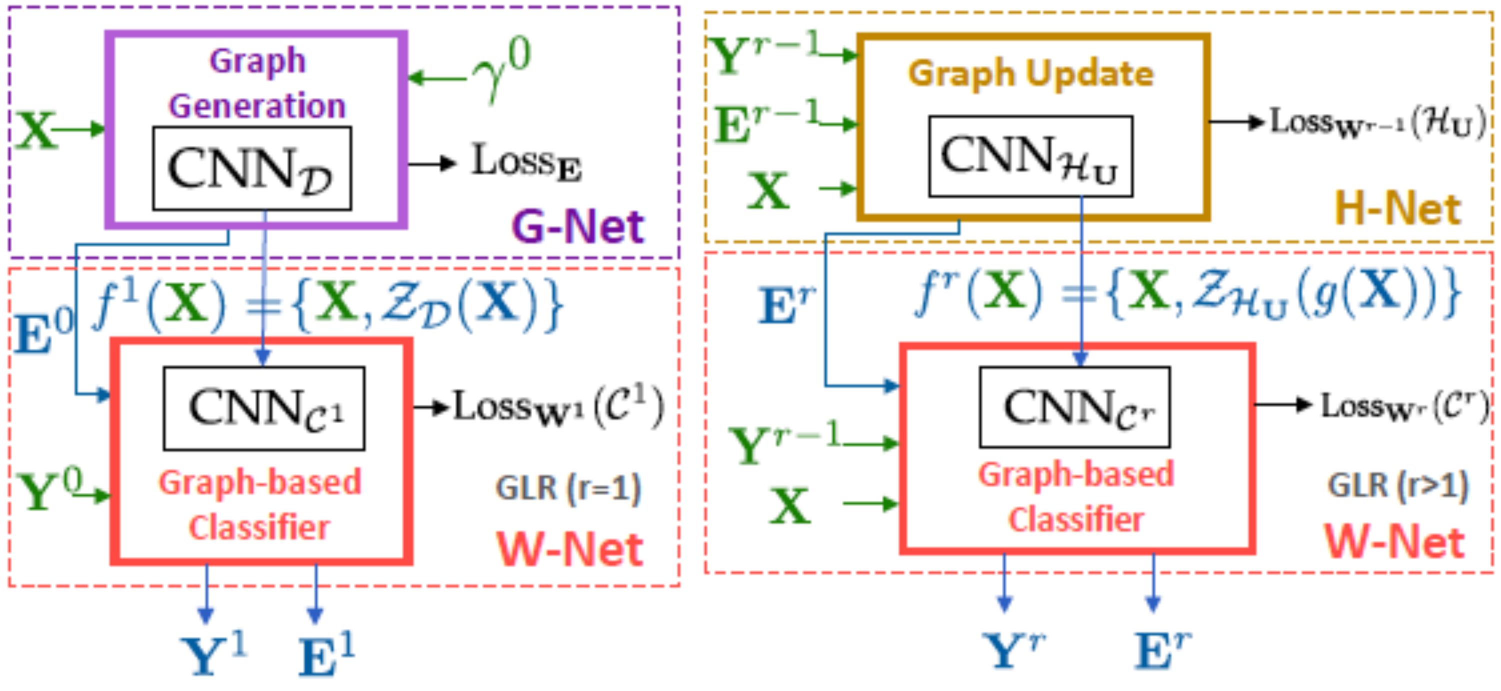


Graph Generation
$\mathbf{X}$
γ^0
$\mathrm{CNN}_{\mathcal{D}}$
$\mathrm{Loss}_{\mathbf{E}}$
G-Net
$\mathbf{E}^0$
$f^1(\mathbf{X}) = \{\mathbf{X}, \mathcal{Z}_{\mathcal{D}}(\mathbf{X})\}$
$\mathrm{CNN}_{\mathcal{C}^1}$
$\mathrm{Loss}_{\mathbf{W}^1}(\mathcal{C}^1)$
$\mathbf{Y}^0$
Graph-based Classifier
GLR (r=1)
W-Net
$\mathbf{Y}^1$
$\mathbf{E}^1$
$\mathbf{Y}^{r-1}$
$\mathbf{E}^{r-1}$
$\mathbf{X}$
Graph Update
$\mathrm{CNN}_{\mathcal{H}_{\mathbf{U}}}$
$\mathrm{Loss}_{\mathbf{W}^{r-1}}(\mathcal{H}_{\mathbf{U}})$
H-Net
$\mathbf{E}^r$
$f^r(\mathbf{X}) = \{\mathbf{X}, \mathcal{Z}_{\mathcal{H}_{\mathbf{U}}}(g(\mathbf{X}))\}$
$\mathrm{CNN}_{\mathcal{C}^r}$
$\mathrm{Loss}_{\mathbf{W}^r}(\mathcal{C}^r)$
$\mathbf{Y}^{r-1}$
$\mathbf{X}$
Graph-based Classifier
GLR (r>1)
W-Net
$\mathbf{Y}^r$
$\mathbf{E}^r$

A.3. Graph regularization for dealing with noisy labels

- Main challenge: Binary classification with noisy labels

- Proposed approach (DynGLR): A two step learning process
  - **Graph learning:** extract deep feature maps and learn a graph that maximizes/minimizes similarity between two nodes that have same/opposite labels (G-Net)
  - **Classifier learning:** alternate between refining the graph (H-Net) and restoring the corrupted classifier signal (W-Net) through graph Laplacian regularization (GLR)

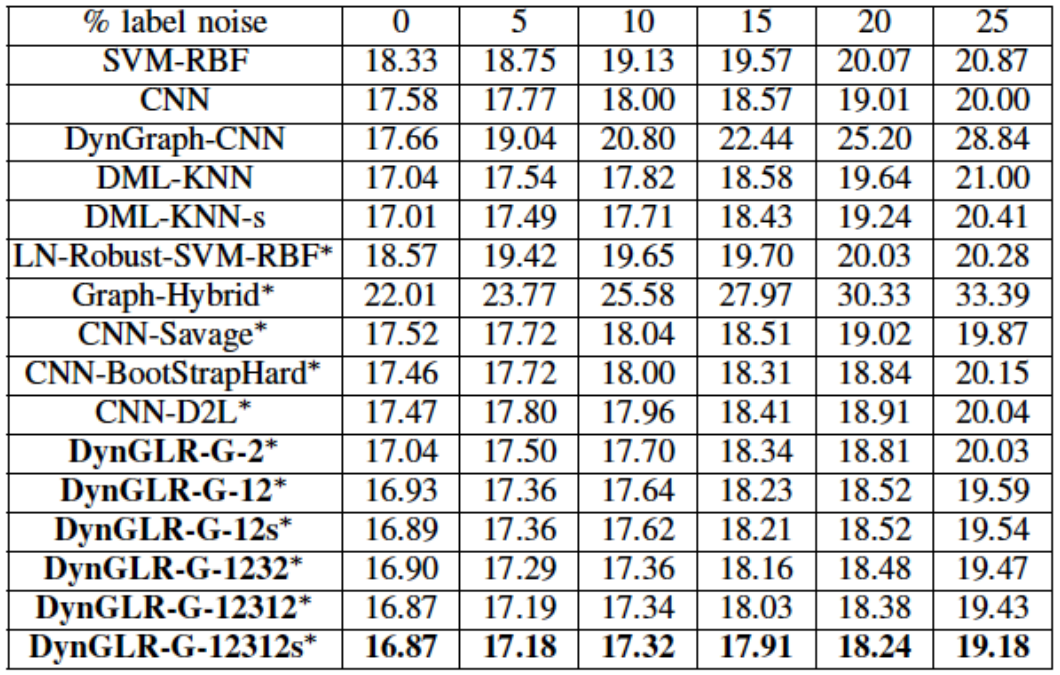


Graph Generation
$\mathrm{CNN}_{\mathcal{D}}$
$\mathbf{X}$
γ^0
$\mathrm{Loss}_{\mathbf{E}}$
G-Net
$\mathbf{E}^0$
$f^1(\mathbf{X}) = \{\mathbf{X}, \mathcal{Z}_{\mathcal{D}}(\mathbf{X})\}$
$\mathrm{CNN}_{\mathcal{C}^1}$
Graph-based Classifier
$\mathbf{Y}^0$
$\mathrm{Loss}_{\mathbf{W}^1}(\mathcal{C}^1)$
GLR (r=1)
W-Net
$\mathbf{Y}^1$
$\mathbf{E}^1$
Graph Update
$\mathrm{CNN}_{\mathcal{H}_\mathrm{U}}$
$\mathbf{Y}^{r-1}$
$\mathbf{E}^{r-1}$
$\mathbf{X}$
$\mathrm{Loss}_{\mathbf{W}^{r-1}}(\mathcal{H}_\mathrm{U})$
H-Net
$\mathbf{E}^r$
$f^r(\mathbf{X}) = \{\mathbf{X}, \mathcal{Z}_{\mathcal{H}_\mathrm{U}}(g(\mathbf{X}))\}$
$\mathrm{CNN}_{\mathcal{C}^r}$
Graph-based Classifier
$\mathbf{Y}^{r-1}$
$\mathbf{X}$
$\mathrm{Loss}_{\mathbf{W}^r}(\mathcal{C}^r)$
GLR (r>1)
W-Net
$\mathbf{Y}^r$
$\mathbf{E}^r$

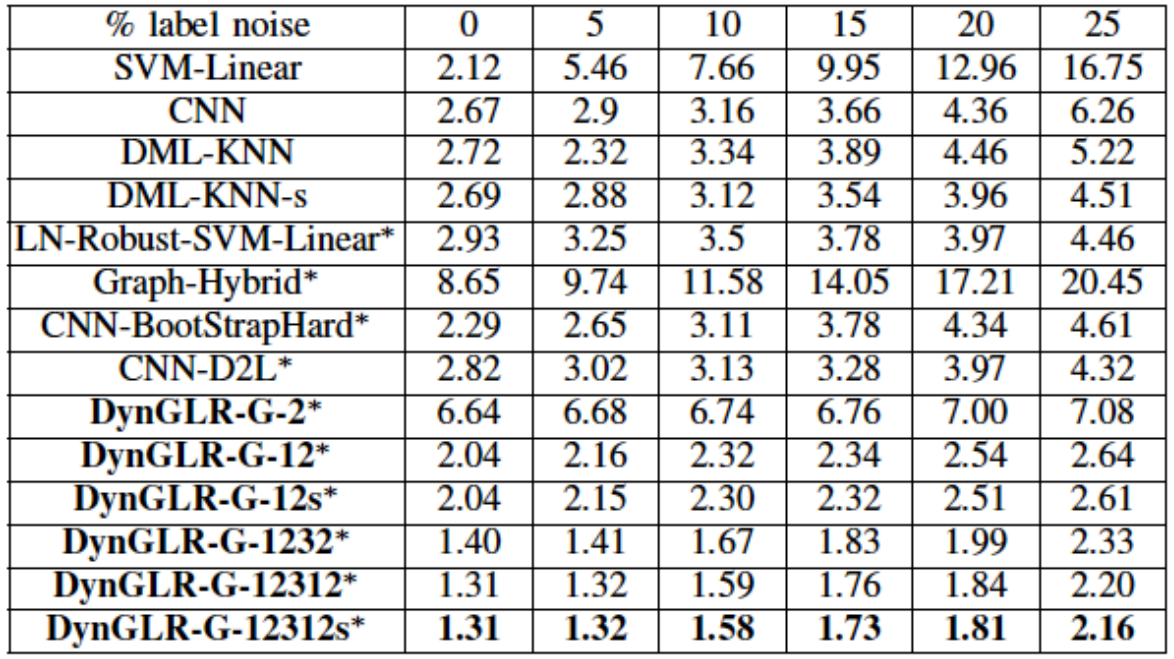
$$Y^r = \arg\min_B \left(\|Y^{r-1} - B\|_2^2 + \mu^r B^T L^r B \right)$$ GLR

Experimental validation

- Classification error curves for different levels of noisy labels
  - Phoneme: contains nasal and oral sounds
  - CIFAR-10: subselection of two classes, airplane and ship



% label noise	0	5	10	15	20	25
SVM-RBF	18.33	18.75	19.13	19.57	20.07	20.87
CNN	17.58	17.77	18.00	18.57	19.01	20.00
DynGraph-CNN	17.66	19.04	20.80	22.44	25.20	28.84
DML-KNN	17.04	17.54	17.82	18.58	19.64	21.00
DML-KNN-s	17.01	17.49	17.71	18.43	19.24	20.41
LN-Robust-SVM-RBF*	18.57	19.42	19.65	19.70	20.03	20.28
Graph-Hybrid*	22.01	23.77	25.58	27.97	30.33	33.39
CNN-Savage*	17.52	17.72	18.04	18.51	19.02	19.87
CNN-BootStrapHard*	17.46	17.72	18.00	18.31	18.84	20.15
CNN-D2L*	17.47	17.80	17.96	18.41	18.91	20.04
DynGLR-G-2*	17.04	17.50	17.70	18.34	18.81	20.03
DynGLR-G-12*	16.93	17.36	17.64	18.23	18.52	19.59
DynGLR-G-12s*	16.89	17.36	17.62	18.21	18.52	19.54
DynGLR-G-1232*	16.90	17.29	17.36	18.16	18.48	19.47
DynGLR-G-12312*	16.87	17.19	17.34	18.03	18.38	19.43
DynGLR-G-12312s*	**16.87**	**17.18**	**17.32**	**17.91**	**18.24**	**19.18**



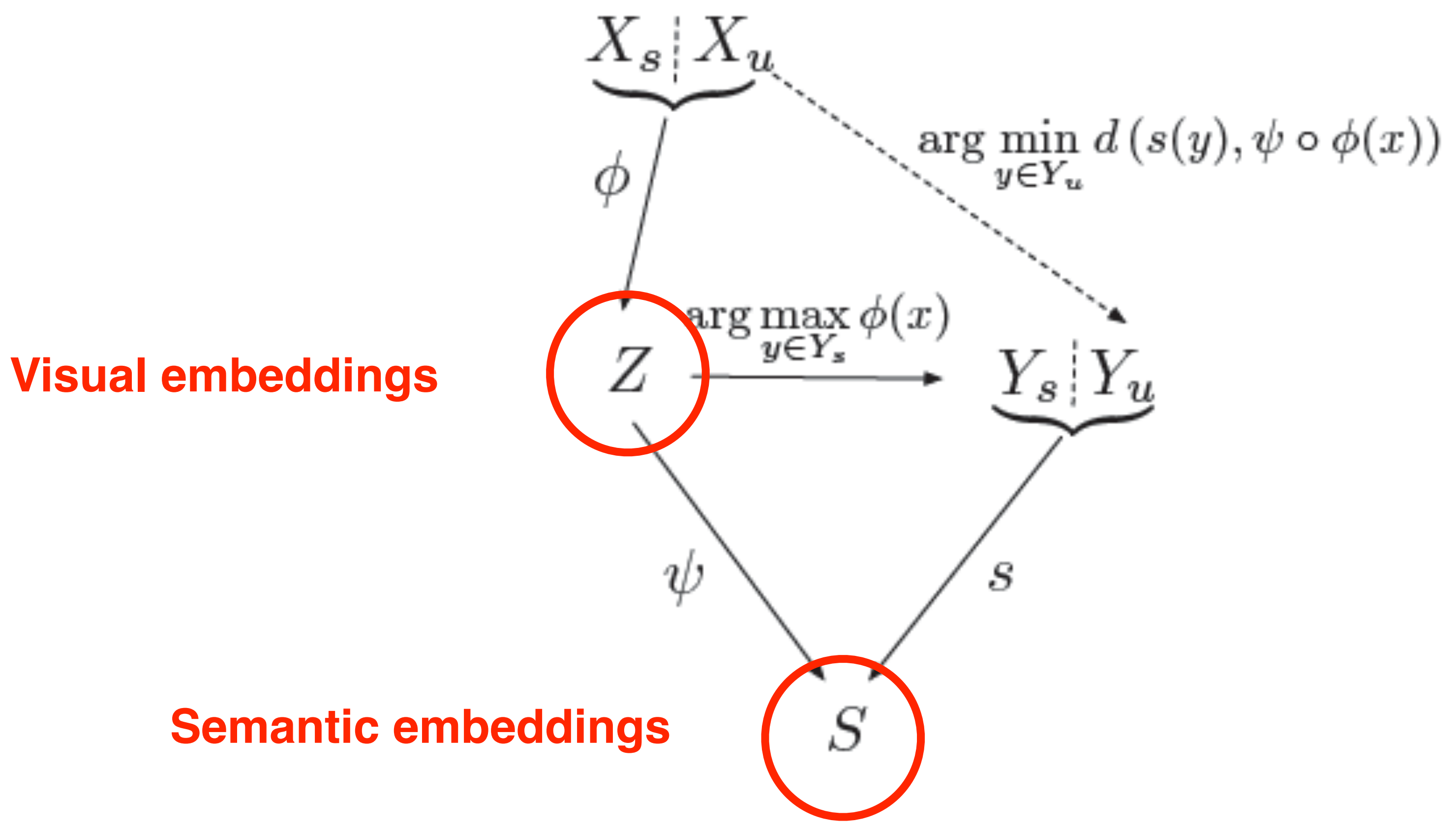
% label noise	0	5	10	15	20	25
SVM-Linear	2.12	5.46	7.66	9.95	12.96	16.75
CNN	2.67	2.9	3.16	3.66	4.36	6.26
DML-KNN	2.72	2.32	3.34	3.89	4.46	5.22
DML-KNN-s	2.69	2.88	3.12	3.54	3.96	4.51
LN-Robust-SVM-Linear*	2.93	3.25	3.5	3.78	3.97	4.46
Graph-Hybrid*	8.65	9.74	11.58	14.05	17.21	20.45
CNN-BootStrapHard*	2.29	2.65	3.11	3.78	4.34	4.61
CNN-D2L*	2.82	3.02	3.13	3.28	3.97	4.32
DynGLR-G-2*	6.64	6.68	6.74	6.76	7.00	7.08
DynGLR-G-12*	2.04	2.16	2.32	2.34	2.54	2.64
DynGLR-G-12s*	2.04	2.15	2.30	2.32	2.51	2.61
DynGLR-G-1232*	1.40	1.41	1.67	1.83	1.99	2.33
DynGLR-G-12312*	1.31	1.32	1.59	1.76	1.84	2.20
DynGLR-G-12312s*	**1.31**	**1.32**	**1.58**	**1.73**	**1.81**	**2.16**

- Introducing GLR is beneficial especially in the high noise regime

Ye et al., “Robust Deep Graph Based Learning for Binary Classification” IEEE Tans. on Sign. and Inform. Process. over Net. 2020

A.4. Graph regularization for zero shot learning

- **Zero shot learning:** Exploit models trained with supervision, and some mapping to a semantic space, to recognise objects as belonging to classes with unseen examples during training



$\underbrace{X_s \mid X_u}$
ϕ
$\arg\min_{y \in Y_u} d\left(s(y), \psi \circ \phi(x)\right)$
$\arg\max_{y \in Y_s} \phi(x)$
Visual embeddings
Z
$\underbrace{Y_s \mid Y_u}$
ψ
s
Semantic embeddings
S

- **Key assumption:** Regularize the space of the continuous manifolds Z, S and the map between them by minimising the isoperimetric loss (IPL)

Graph-based approximation of IPL

- **Isoperimetric loss (IPL):** measures the flow through a closed neighborhood relative to the area of its boundary

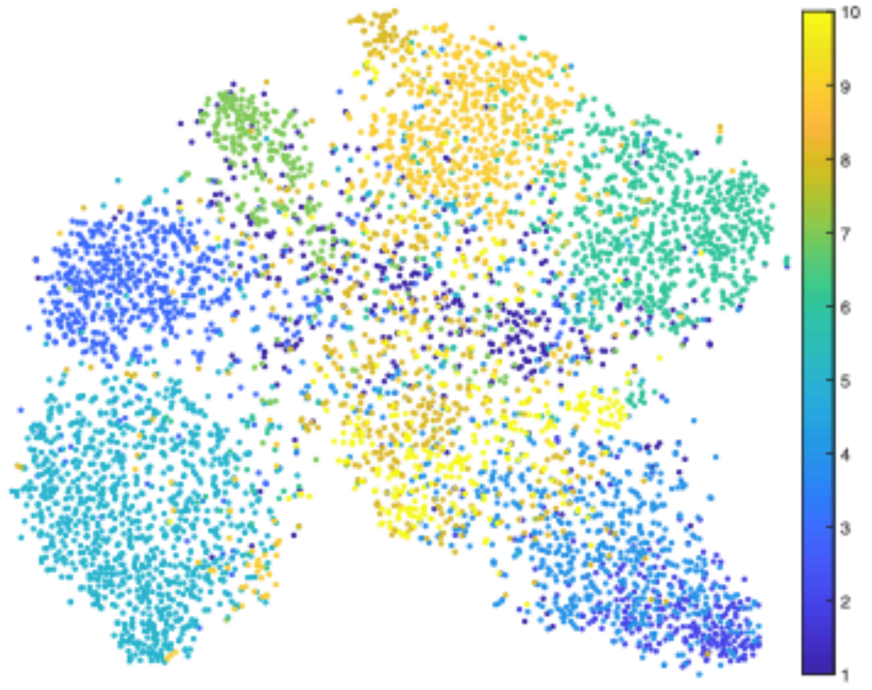
Graph-based approximation of IPL

- **Isoperimetric loss (IPL):** measures the flow through a closed neighborhood relative to the area of its boundary

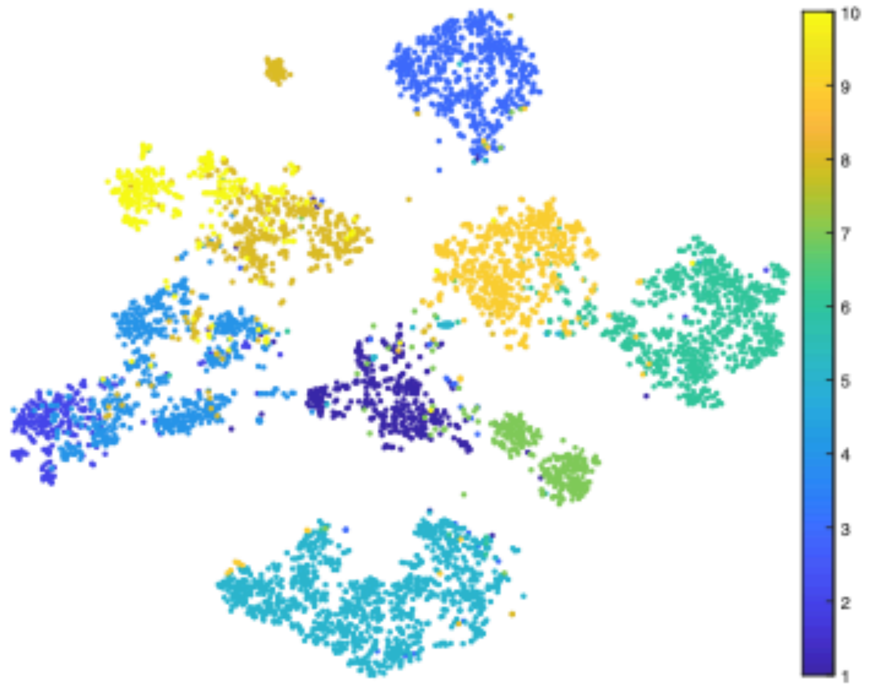
- **A graph-based approximation of IPL:**
  - Treat visual embeddings as vertices in a graph, with affinities as edges, $G = (Z, W)$ and visual-to-embedding map as a linear function on the graph $\psi : G \rightarrow S$
  - Seek a non-parametric deformation represented by changes in the connectivity matrix, that minimises the IPL
  - Approximate IPL loss by using spectral reduction: Spectral graph wavelets of the semantic embedding space $\hat{S} = g(L)S$
  - Construct a new graph based on the spectral embeddings $G^{'} = (\hat{S}, W')$
  - Perform clustering to map clusters to labels

Experimental validation

- **Embeddings in AWA1**



2D t-SNE, no IPL



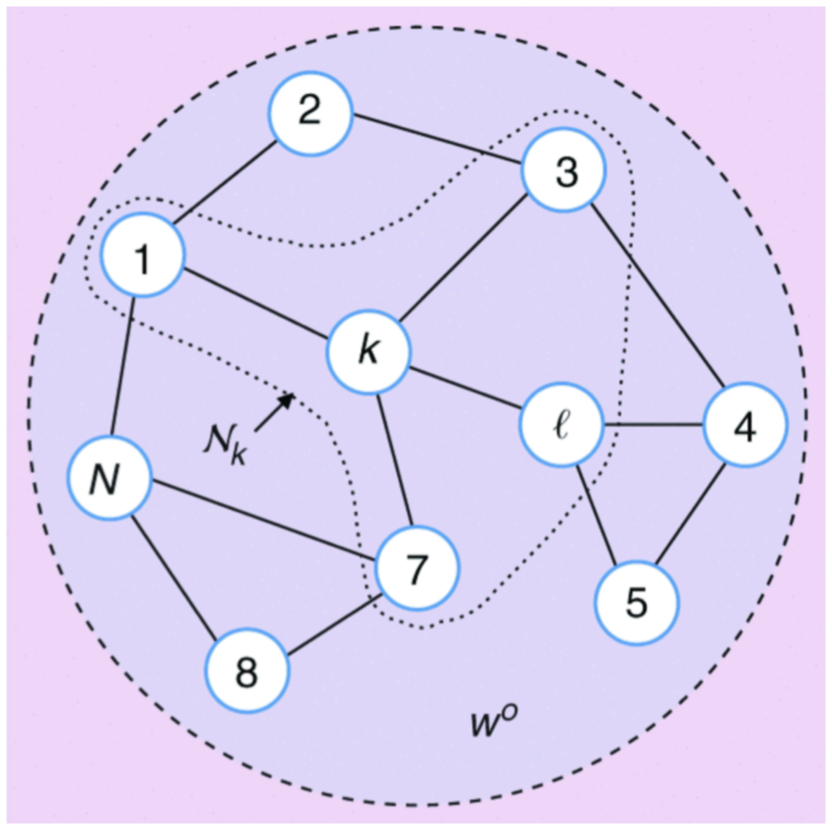
2D t-SNE, with IPL

- IPL regularization generates more compact embeddings
- Error decrease of 9.8% and 44% for AWA1 and CUB dataset

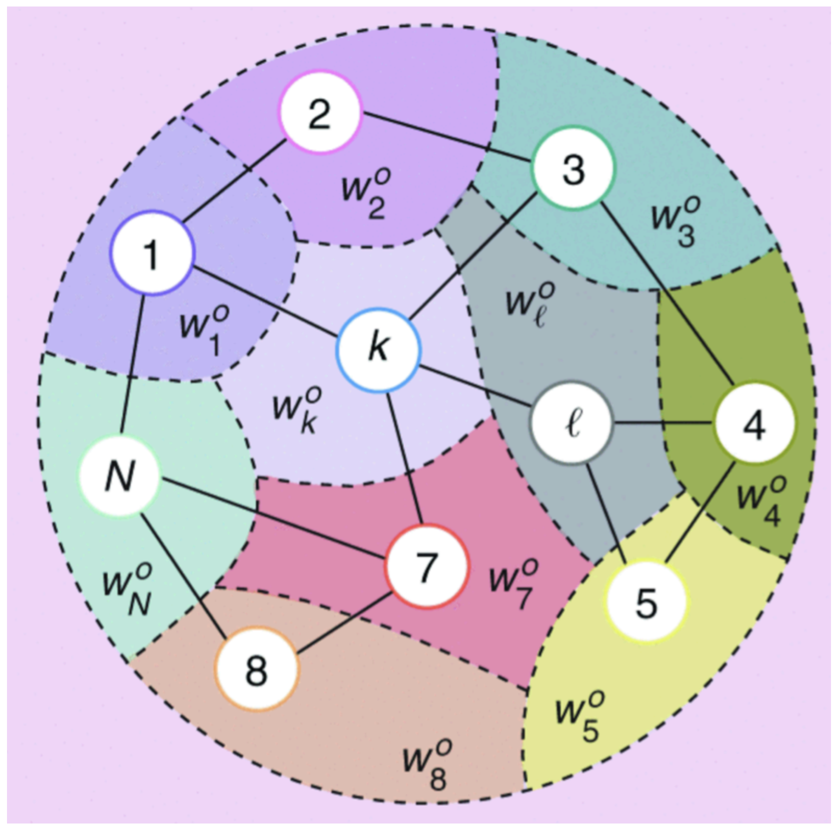
Deutch et al., "Zero shot learning with the isoperimetric loss", AAAI, 2020

A.5. GSP for multi-task learning

- **Multi-task learning:** Learn simultaneously several related tasks
  - Helps improve generalisation performance
- Often data are collected in a distributed fashion
  - Each node can communicate only with local neighbors to solve a task



a) single task learning



b) multi-task learning

Nassif et al., “Multi-task learning over graphs”, IEEE Signal Process. Mag., 2020 14

Spectral regularization for multi-task learning

- The graph captures the correlation between the tasks, i.e., nodes of the graph

- The goal of each node k is to compute the parameters that minimise an objective function

$$w_k^0 = \arg\min_{w_k} J_k(w_k)$$

- The relationship between the tasks can be exploited by imposing a regularization of the cost function on the task graph

$$W^* = \arg\min_{W} J^{glob}(W) = \sum_{k=1}^{N} J_k(w_k) + \frac{\eta}{2} R(W, G)$$

- Regularization examples:
  - Smoothness of tasks in the graph $R(W, G) = W^T L W$
  - Graph spectral regularization $R(W, G) = W^T g(L) W$

Summary so far

- Graphs can be used to capture hidden dependencies of any type of data

- The domain of the graph is application specific
    - Latent space of features
    - Manifold approximation
    - Distributed agents

- GSP tools, and in particular, graph regularization help towards imposing desirable properties
    - Better node embeddings
    - Cleaner signals
    - More robust models

GSP for improving efficiency and robustness

A. Improvement on data efficiency and robustness

B. Robustness against topological noise

C. Improvement on computational efficiency

GSP for improving efficiency and robustness

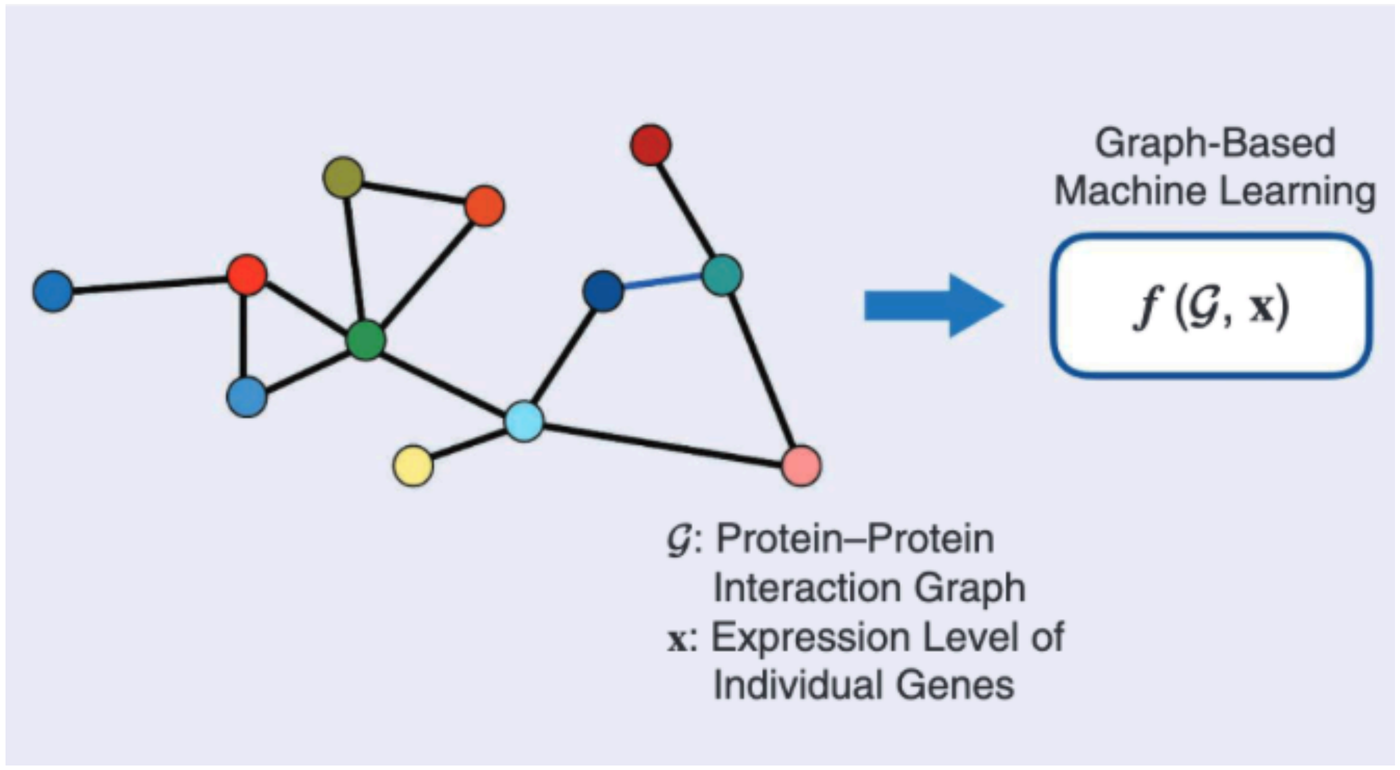
A. Improvement on data efficiency and robustness

B. Robustness against topological noise

C. Improvement on computational efficiency

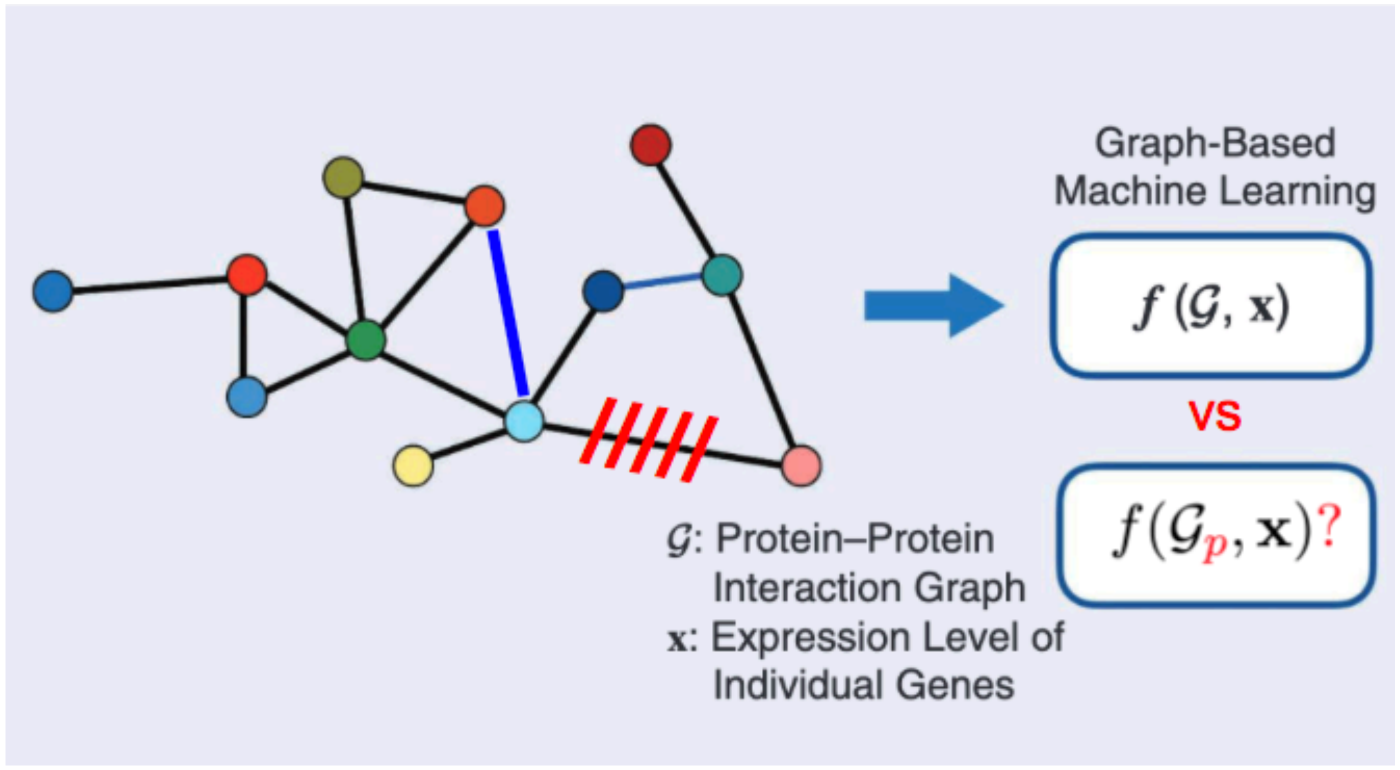
B. Robustness against topological noise

- In the context of graph-structured data, stability can be defined with respect to perturbation to the underlying topology


Graph-Based Machine Learning
$f(\mathcal{G}, \mathbf{x})$
$\mathcal{G}$: Protein–Protein Interaction Graph
$\mathbf{x}$: Expression Level of Individual Genes

B. Robustness against topological noise

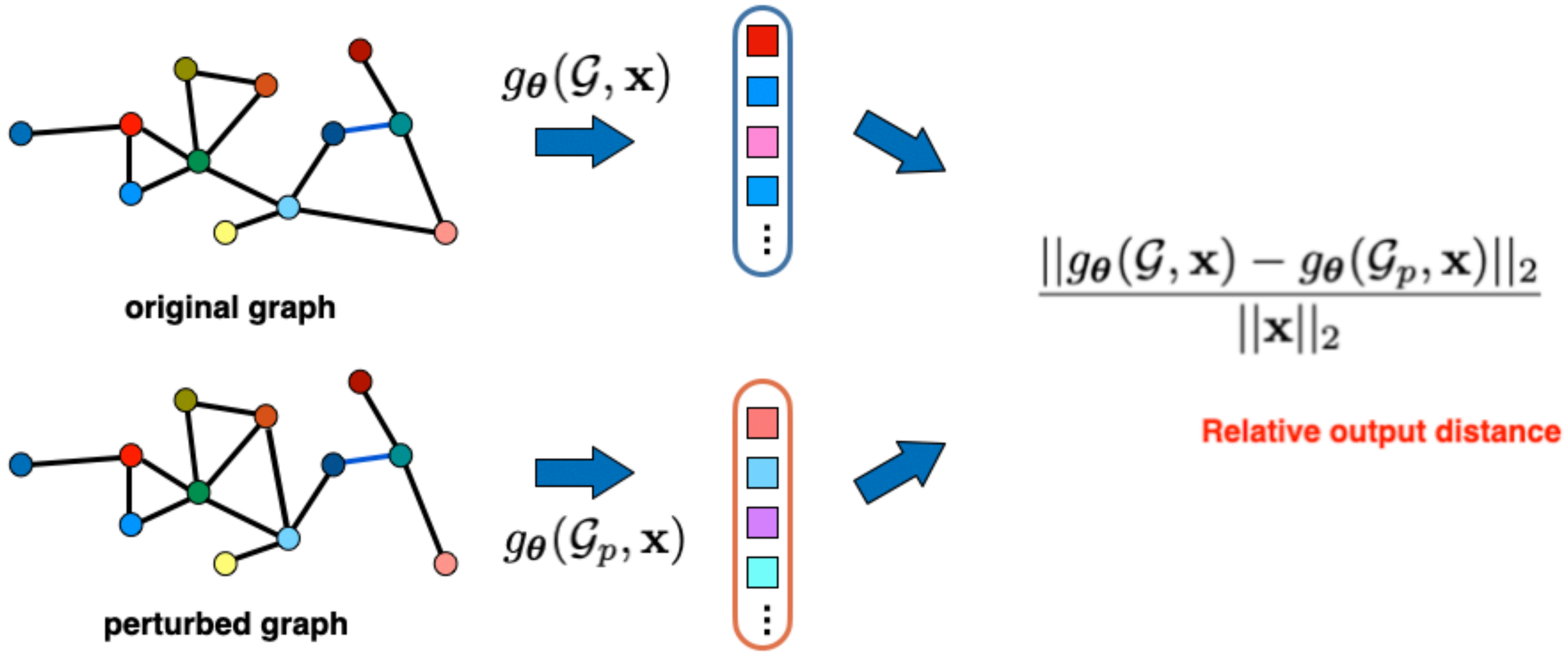
- In the context of graph-structured data, stability can be defined with respect to perturbation to the underlying topology


Graph-Based Machine Learning
$f(\mathcal{G}, \mathbf{x})$
VS
$f(\mathcal{G}_p, \mathbf{x})$?
$\mathcal{G}$: Protein–Protein Interaction Graph
$\mathbf{x}$: Expression Level of Individual Genes

Why is it important?

- noisy/unreliable graph
- adversaries
- transferability
- changes through time

B.1. Stability of graph filters



$g_{\boldsymbol{\theta}}(\mathcal{G}, \mathbf{x})$
original graph
$g_{\boldsymbol{\theta}}(\mathcal{G}_p, \mathbf{x})$
perturbed graph
$\frac{\|g_{\boldsymbol{\theta}}(\mathcal{G}, \mathbf{x}) - g_{\boldsymbol{\theta}}(\mathcal{G}_p, \mathbf{x})\|_2}{\|\mathbf{x}\|_2}$
Relative output distance

- spectral graph filters in the Cayley smoothness space are stable

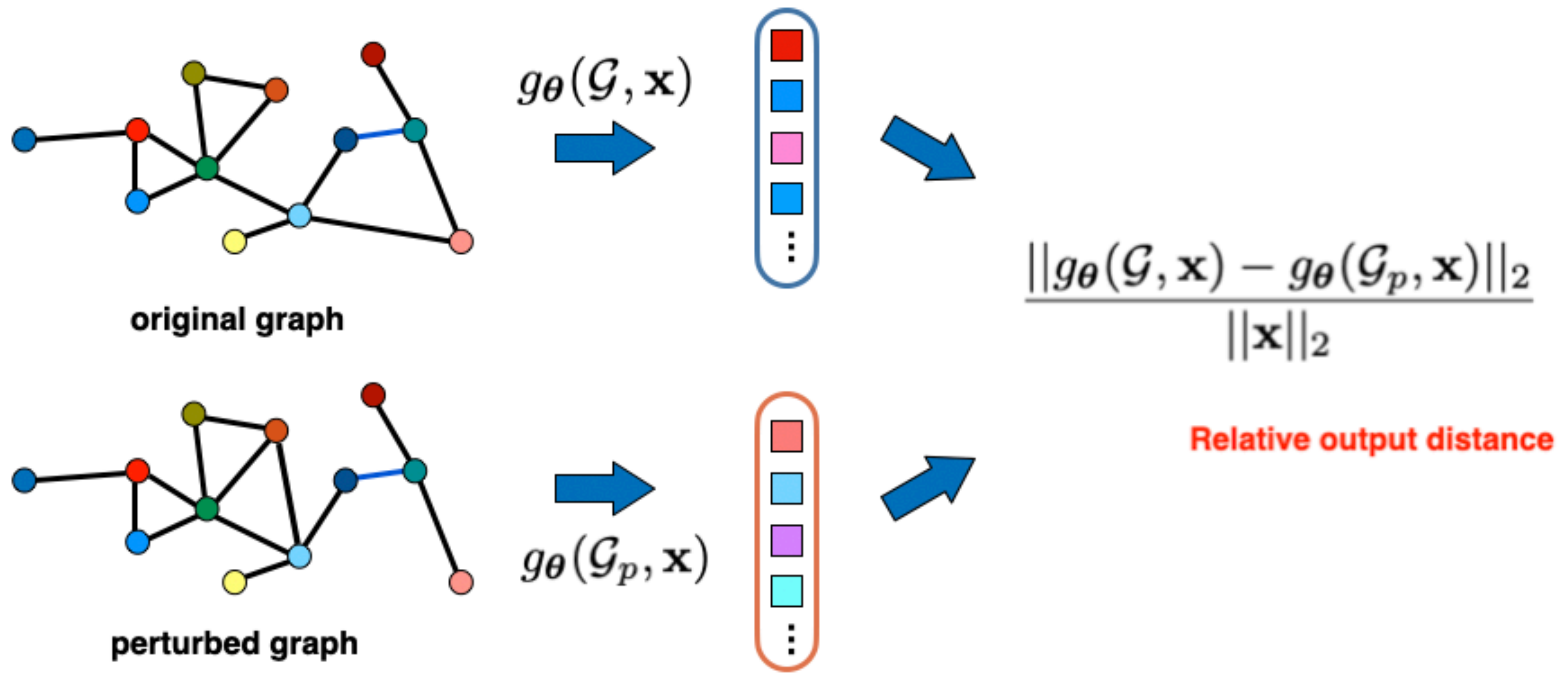
$$\|g(\boldsymbol{\Delta}) - g(\boldsymbol{\Delta}')\| \leq \|g\|_{\mathcal{C}} \left((\|\boldsymbol{\Delta}\| + 1) \frac{\|\mathbf{E}\|}{1 - \|\mathbf{E}\|} + \|\mathbf{E}\| \right)$$
$$= O(\|\boldsymbol{\Delta} - \boldsymbol{\Delta}'\|).$$

$\boldsymbol{\Delta}$: graph shift operator

$\mathbf{E}$: error matrix

Levie et al., “On the transferability of spectral graph filters,” SampTA, 2019.

Stability of graph filters


$g_{\boldsymbol{\theta}}(\mathcal{G}, \mathbf{x})$
$g_{\boldsymbol{\theta}}(\mathcal{G}_p, \mathbf{x})$
original graph
perturbed graph
$\frac{||g_{\boldsymbol{\theta}}(\mathcal{G}, \mathbf{x}) - g_{\boldsymbol{\theta}}(\mathcal{G}_p, \mathbf{x})||_2}{||\mathbf{x}||_2}$
Relative output distance

- polynomial spectral graph filters are stable

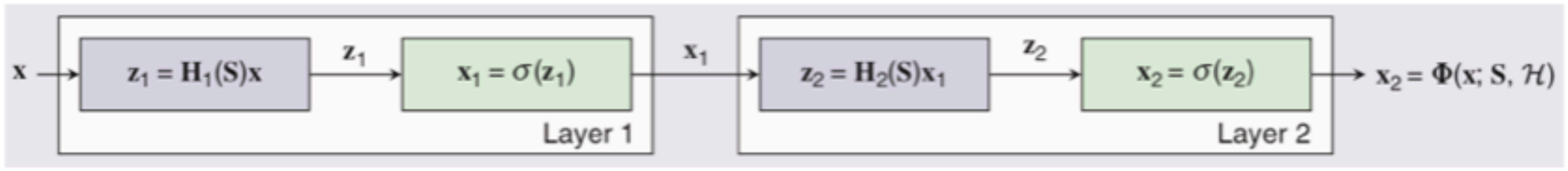
$$||g_{\theta}(\mathbf{L}) - g_{\theta}(\mathbf{L_p})||_2 \leq \frac{1}{4}||\Theta_{-0}||_1(K^2 - 1)\left(\frac{K+1}{K-1}\right)^K ||\mathbf{L} - \mathbf{L_p}||_2$$

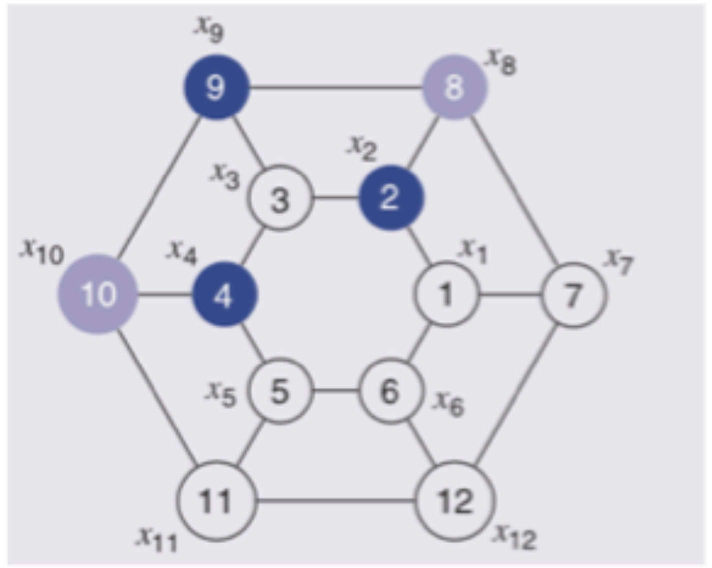
$\mathbf{L}$: graph Laplacian

Θ_{-0} : polynomial coefficient (except for constant)

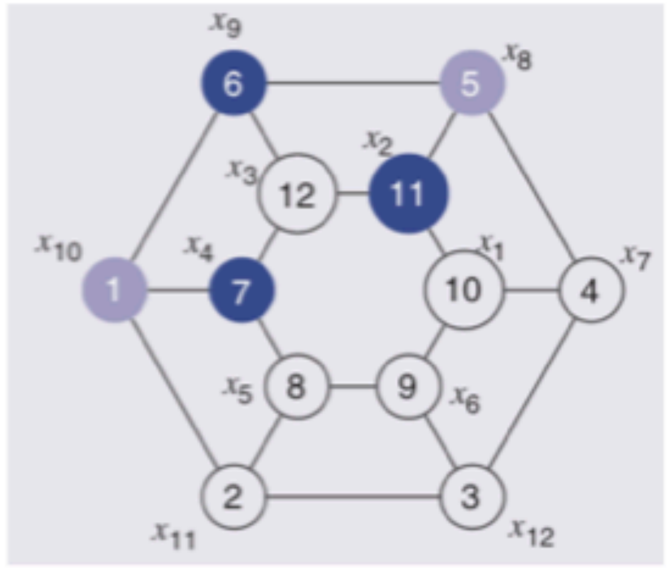
Kenlay et al., “On the stability of polynomial spectral graph filters,” ICASSP, 2020.

B.2. Stability of graph neural networks





$\mathbf{S}, \mathbf{x}$



$\hat{\mathbf{S}} = \mathbf{P}^T\mathbf{S}\mathbf{P}, \hat{\mathbf{x}} = \mathbf{P}^T\mathbf{x}$

- equivariance of graph neural networks under permutation **P**

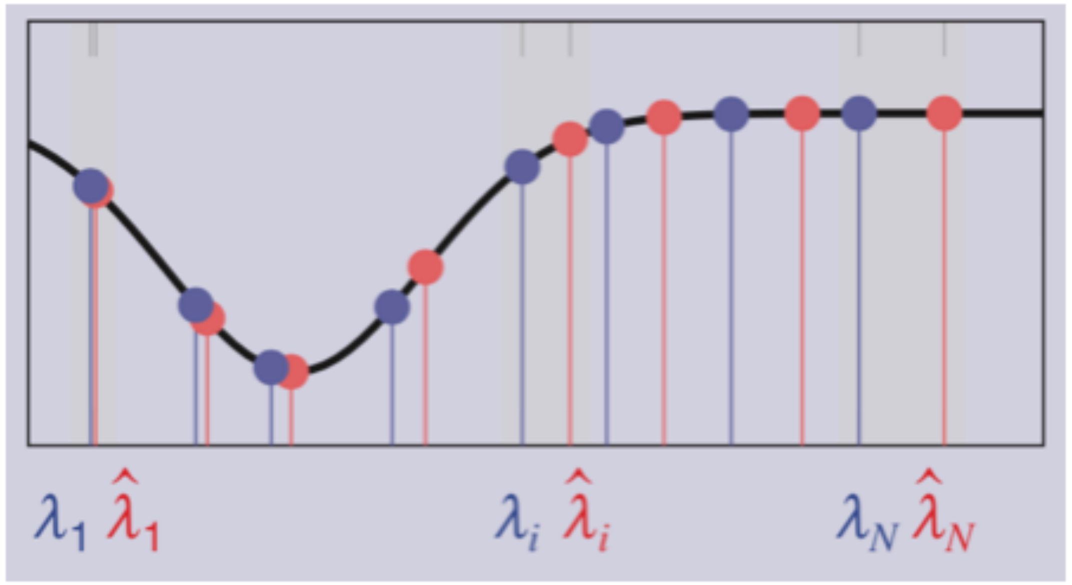
$$\mathbf{\Phi}(\hat{\mathbf{x}}; \hat{\mathbf{S}}, \mathcal{H}) = \mathbf{P}^T \mathbf{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})$$

- GSP with GNNs is independent of node relabelings!

Gama et al., “Stability properties of graph neural networks,” IEEE TSP, 2020.

Ruiz et al., “Graph neural networks: Architectures, stability, and transferability,” Proceedings of the IEEE, 2021.

Stability of graph neural networks



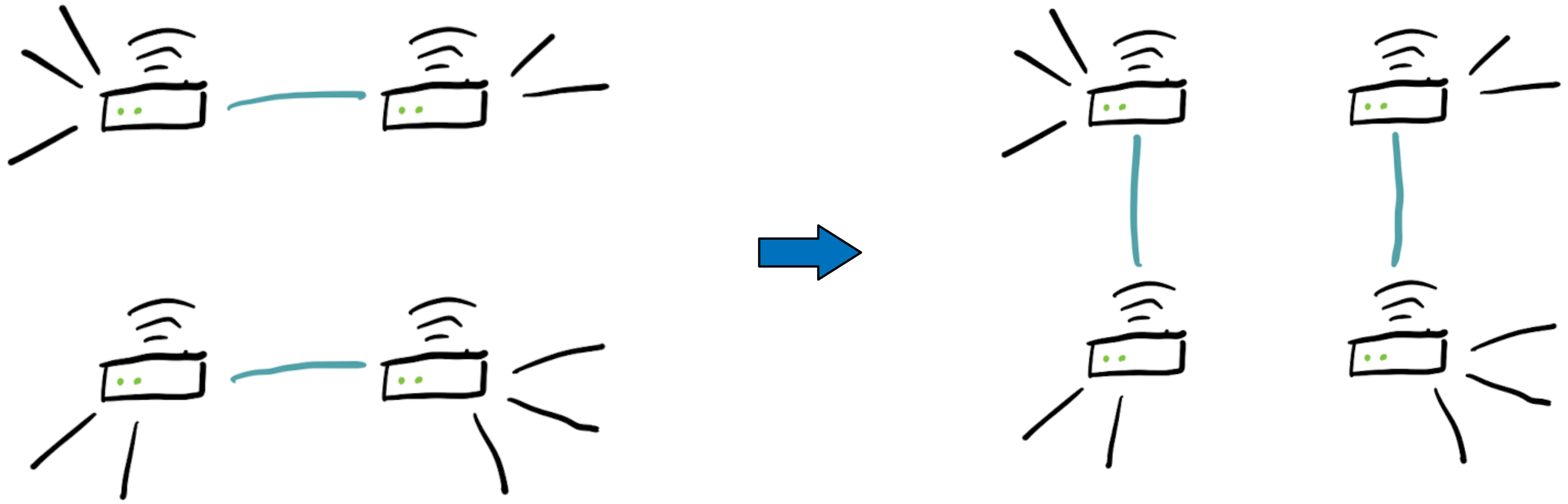
- graph neural networks with integral Lipschitz filters are stable

$$||\Phi(\mathbf{x};\mathbf{S},\mathcal{H}) - \Phi(\underbrace{\mathbf{P}^{*T}}_{\text{operator minimising operator distance}}\mathbf{x};\mathbf{P}^{*T}\mathbf{S}\mathbf{P}^{*},\mathcal{H})|| \le 2\underbrace{C}_{\text{Lipschitz constant}}(1+\underbrace{\delta\sqrt{N}}_{\text{eigenvector misalignment}})\underbrace{L}_{\text{number of layers}}\underbrace{\epsilon}_{\text{operator distance}}||\mathbf{x}|| + \mathcal{O}(\epsilon^2)$$

Gama et al., "Stability properties of graph neural networks," IEEE TSP, 2020.
Ruiz et al., "Graph neural networks: Architectures, stability, and transferability," Proceedings of the IEEE, 2021.

Stability of graph neural networks

- Perturbations that do not modify the degree distribution of the graph



- stability under double-edge rewiring

$$\underbrace{\|\mathbf{X}^{(L)} - \mathbf{X}_p^{(L)}\|_F}_{\text{output change}} \leq \underbrace{\sqrt{d}}_{\text{data}} \underbrace{L \prod_{l=1}^{L} \|\boldsymbol{\Theta}^{(l)}\|_2}_{\text{model}} \underbrace{\|\mathbf{E}\|_2}_{\text{structural change}}$$

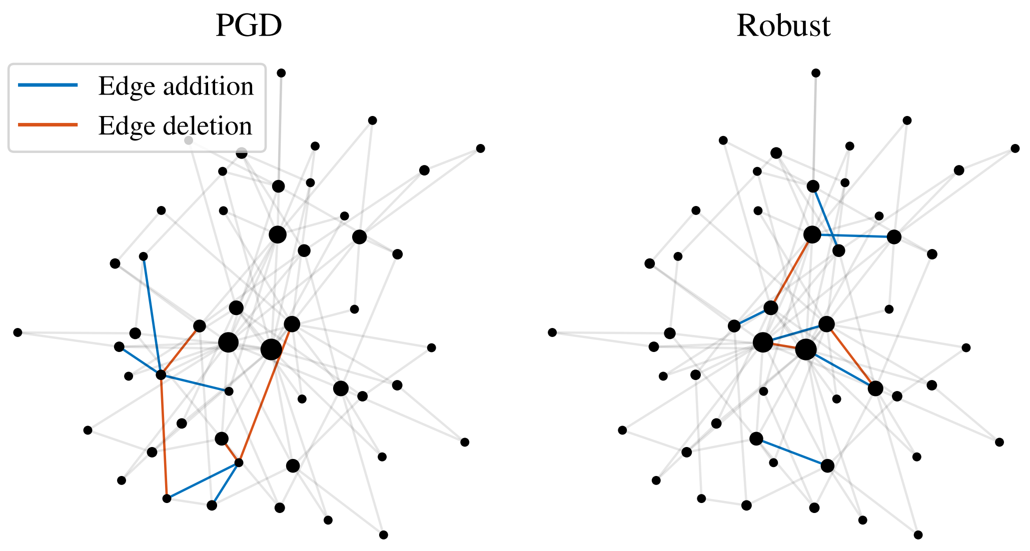
$d =$ Node feature dimension

$L =$ Number of layers

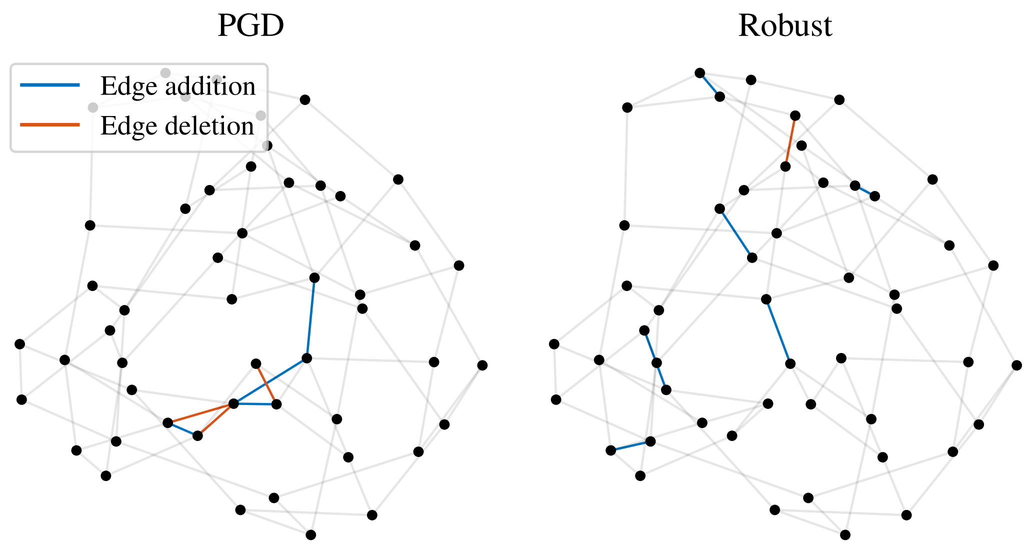
$\mathbf{E} = \boldsymbol{\Delta}_\mathbf{p} - \boldsymbol{\Delta}$

Kenlay et al., "On the stability of graph convolutional neural networks under edge rewiring," ICASSP, 2021.

B.3. Interpretable stability bounds



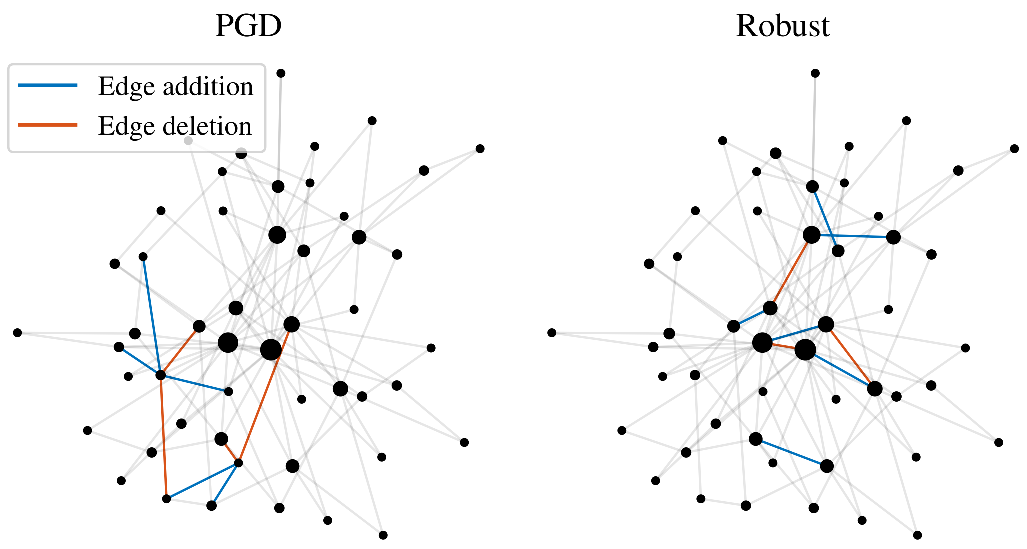
BA graph



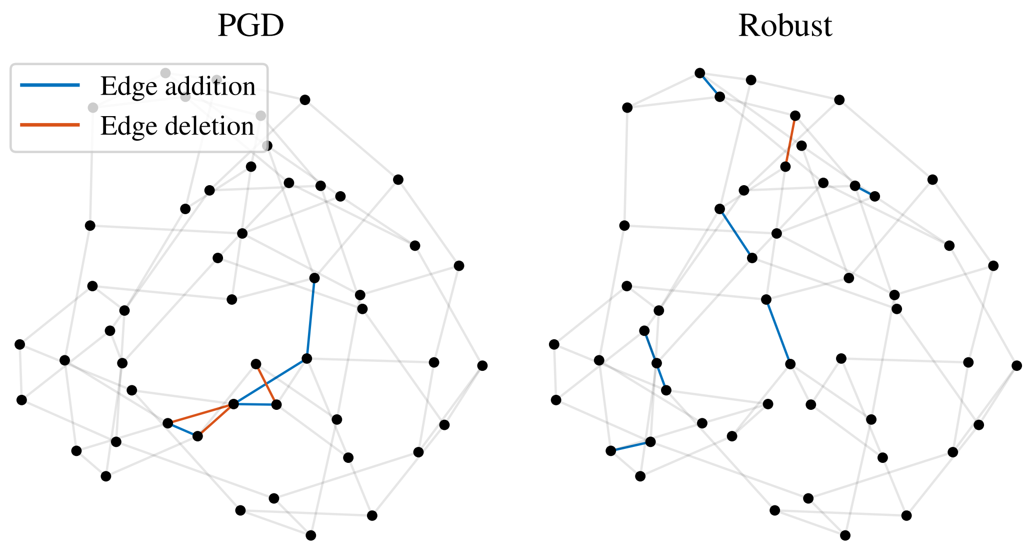
3-regular graph

- how does stability depend on topological properties of perturbation?

B.3. Interpretable stability bounds



BA graph

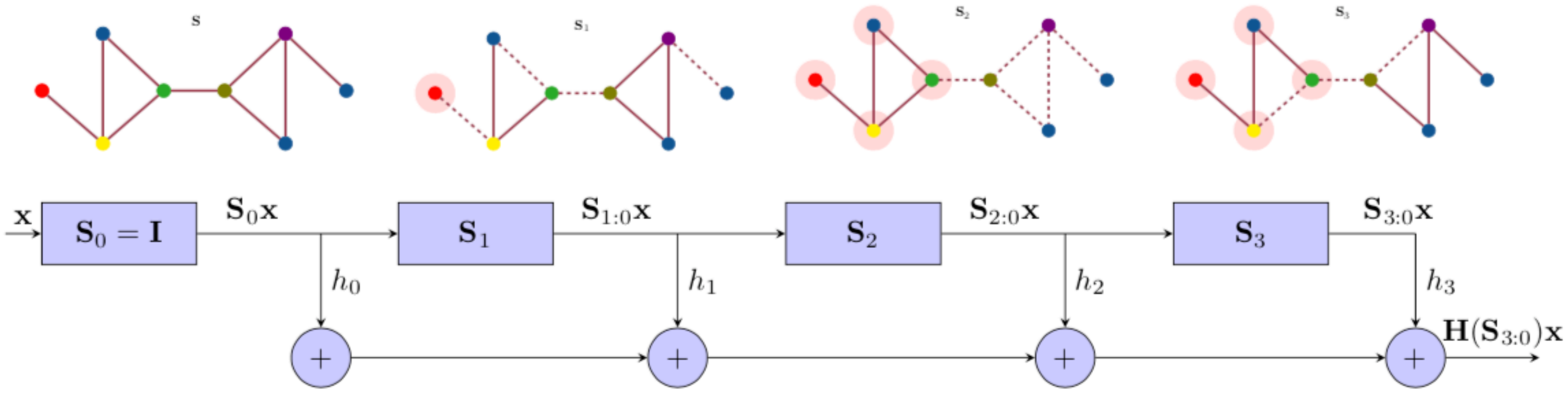


3-regular graph

- how does stability depend on topological properties of perturbation?
- spectral graph filters are most stable if
  - adding or deleting edges between high degree nodes
  - not perturbing too much around any one node

B.4. Robustness beyond stability

- GSP in the presence of topological uncertainty (more in part III) [2, 4]
- Filters on stochastic time-evolving graphs [1]
- Stochastic graph filters built on sequence of randomly perturbed graphs [3]



[1] Isufi et al., "Filtering random graph processes over random time-varying graphs," IEEE TSP, 2017.
[2] Ceci and Barbarossa, "Graph signal processing in the presence of topology uncertainties," IEEE TSP, 2020.
[3] Gao et al., "Stochastic graph neural networks," ICASSP, 2020.
[4] Miettinen et al., "Modelling graph errors: Towards robust graph signal processing," arXiv, 2020.

GSP for improving efficiency and robustness

A. Improvement on data efficiency and robustness

B. Robustness against topological noise

C. Improvement on computational efficiency

GSP for improving efficiency and robustness

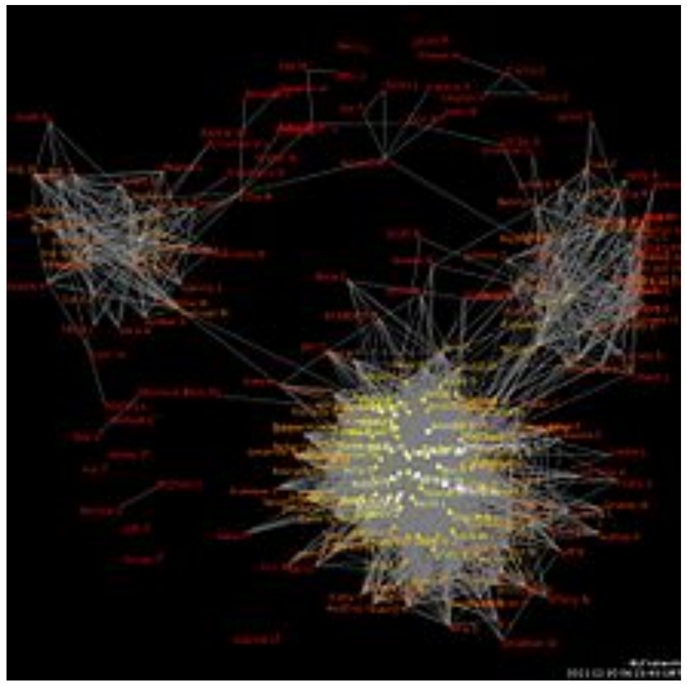
A. Improvement on data efficiency and robustness

B. Robustness against topological noise

C. Improvement on computational efficiency

C. Improvement on computational complexity

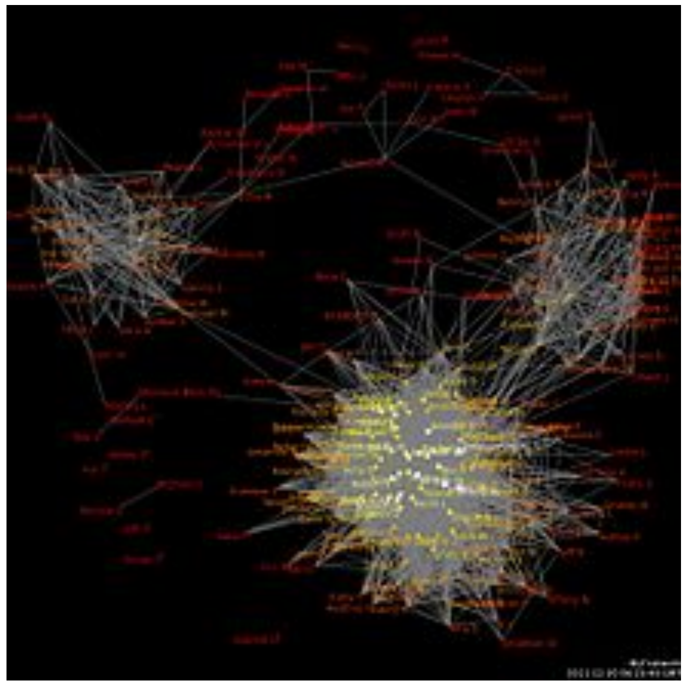
- Many ML algorithms require a large amount of resources (i.e., space, runtime, communication) for their computation
  - Eigendecomposition of a large matrix is expensive
  - Training and deploying GNNs remains challenging due to high memory consumption and inference latency



https://en.wikipedia.org/wiki/Social_network_analysis

C. Improvement on computational complexity

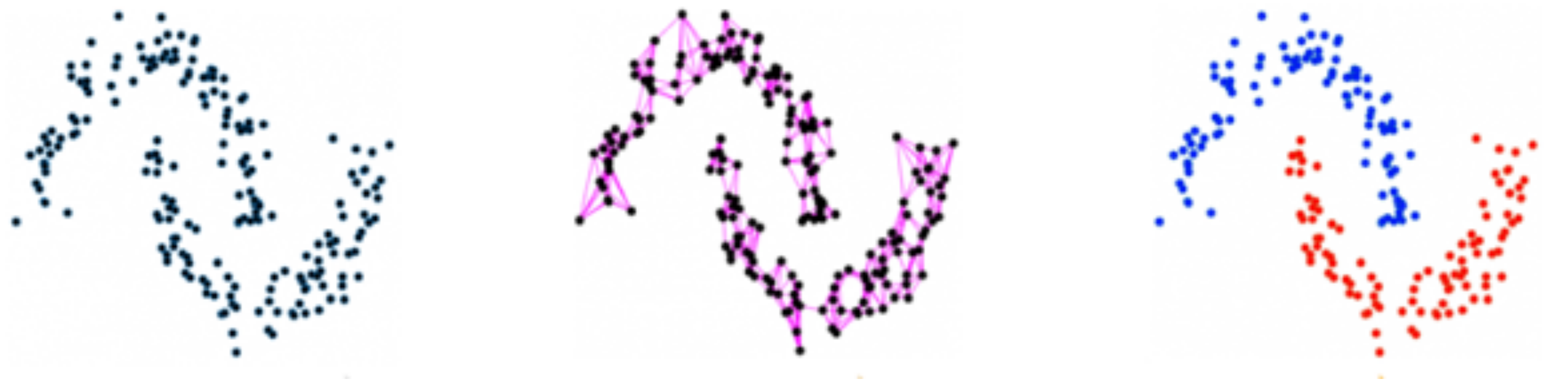
- Many ML algorithms require a large amount of resources (i.e., space, runtime, communication) for their computation
  - Eigendecomposition of a large matrix is expensive
  - Training and deploying GNNs remains challenging due to high memory consumption and inference latency


https://en.wikipedia.org/wiki/Social_network_analysis

Can we use the graph structure and classical GSP operators to alleviate some of these issues?

C.1. Complexity of spectral clustering

- **Spectral clustering** : Given a graph G capturing the structure of N data points, find the partition of the nodes into K clusters



- **Algorithm:** Given the graph Laplacian matrix L
  - Compute the first K eigenvectors $V_K = [V_1, V_2, ..., V_K]$
  - Compute the embedding of each node n in V_K : $\phi_n = U_K^T \delta_n$
  - Run K-Means with Euclidean distance on the embeddings $D_{n,m} = \|\phi_n - \phi_m\|_2$ to compute the clusters

- **Bottlenecks:** When N, K are large
  - Computing the eigendecomposition is expensive $O(NK^2)$
  - The complexity of K-means is high $O(NK^2)$

von Luxburg, “A tutorial on spectral clustering”, Statistics and Computing, 2007

Compressive spectral clustering

- **A GSP perspective**
  - Computation of eigenvectors is bypassed by filtering random graph signals
  - K-means is performed on a subset of randomly selected nodes $m \approx K \log K$
  - The cluster centers of the whole graph are interpolated

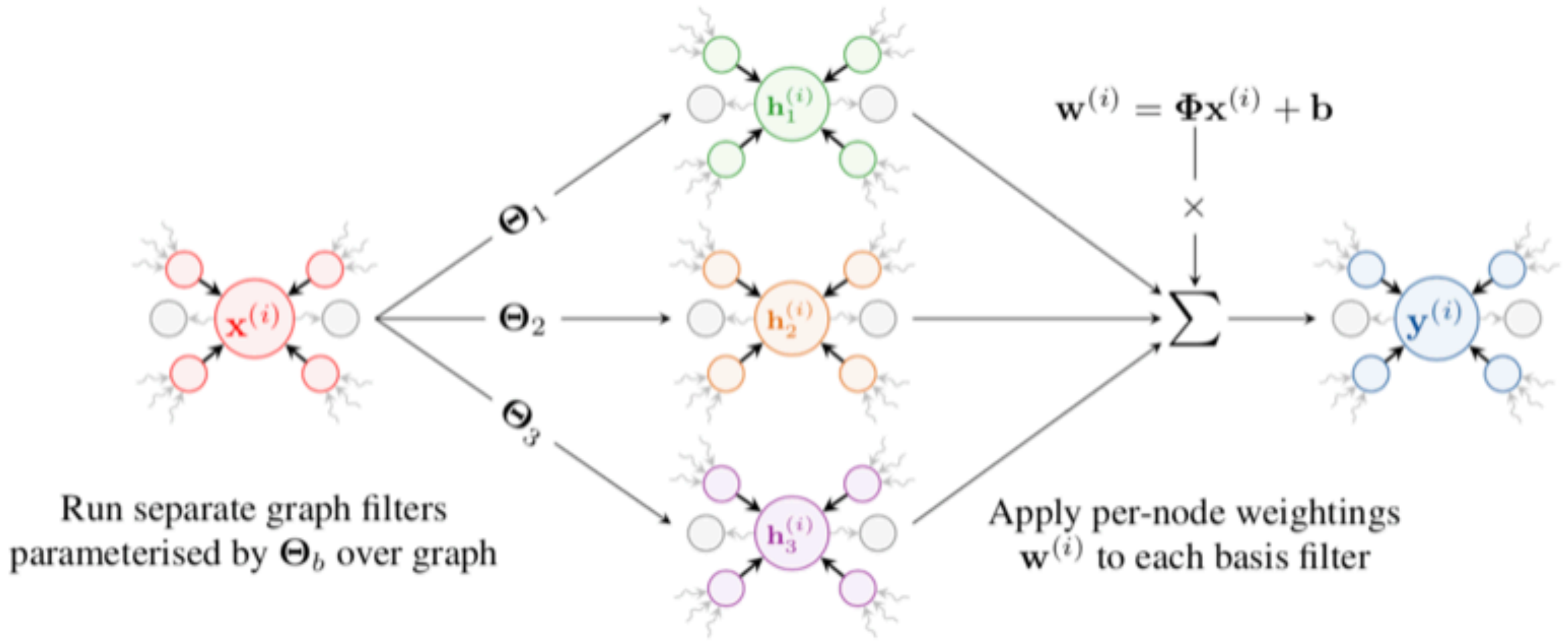
- **Algorithm:**
  - Estimate λ_k from L
  - Generate d random graph signals in matrix $R \in \mathbf{R}^{N \times d}$
  - Filter them with a polynomial filter H_{λ_k}, i.e., $\tilde{\phi}_n = (H_{\lambda_k} R)^T \delta_i$
  - If $d \sim \log N : \tilde{D}_{n,m} = \|\tilde{\phi}_n - \tilde{\phi}_m\|_2 \approx D_{n,m}$
  - Draw randomly $m \approx K \log K$ sample nodes, and run K-means to obtain cluster centers $c^r = [c_1^r, ..., c_K^r]$
  - Interpolate the centers by exploiting the sampling theory of bandlimited graph signals

$$\tilde{c}_j = \arg \min_{c_j \in \mathbb{R}^N} \|M c_j - c_j^r\|_2^2 + \gamma c_j^T g(L) c_j$$

Trembley et al., “Compressing spectral clustering”, ICML, 2016
Trembley et al., “Approximating spectral clustering by sampling: A review”, Sampling Techniques for Supervised or Unsupervised Tasks, Springer 2020

C.2. Adaptive filters and per-node weighting in GNN

- A (graph) signal processing approach for efficient hardware usage


$\mathbf{x}^{(i)}$
Θ_1
Θ_2
Θ_3
$\mathbf{h}_1^{(i)}$
$\mathbf{h}_2^{(i)}$
$\mathbf{h}_3^{(i)}$
$\mathbf{w}^{(i)} = \Phi\mathbf{x}^{(i)} + \mathbf{b}$
$\times$
$\sum$
$\mathbf{y}^{(i)}$
Run separate graph filters parameterised by Θ_b over graph
Apply per-node weightings $\mathbf{w}^{(i)}$ to each basis filter

- Adaptive graph filters for each node
- Node-specific weighted combination of adaptive filters Θ
- For 1-order Chebyshev polynomials:

$$Y = \sum_{b=1}^{B} w_b \odot (D^{-1/2} A D^{-1/2}) X \Theta_b$$

Tailor et al., “Adaptive filters and aggregator fusion for efficient graph convolutions,” arXiv, 2021.

Adaptive filters and per-node weighting in GNN

$$\begin{aligned}\mathbf{y}^{(i)} &= \Big\Vert_{h=1}^{H} \alpha_{h,i,i}\mathbf{\Theta}\mathbf{x}^{(i)} + \sum_{j\in\mathcal{N}(i)} \alpha_{h,i,j}\mathbf{\Theta}\mathbf{x}^{(j)} \\ &= \Big\Vert_{h=1}^{H} \underbrace{\mathbf{\Theta}}_{\text{Shared Weights}} \underbrace{\left(\sum_{j\in\mathcal{N}(i)\cup\{i\}} \alpha_{h,i,j}\mathbf{x}^{(j)}\right)}_{\text{Message Materialization}}\end{aligned}$$

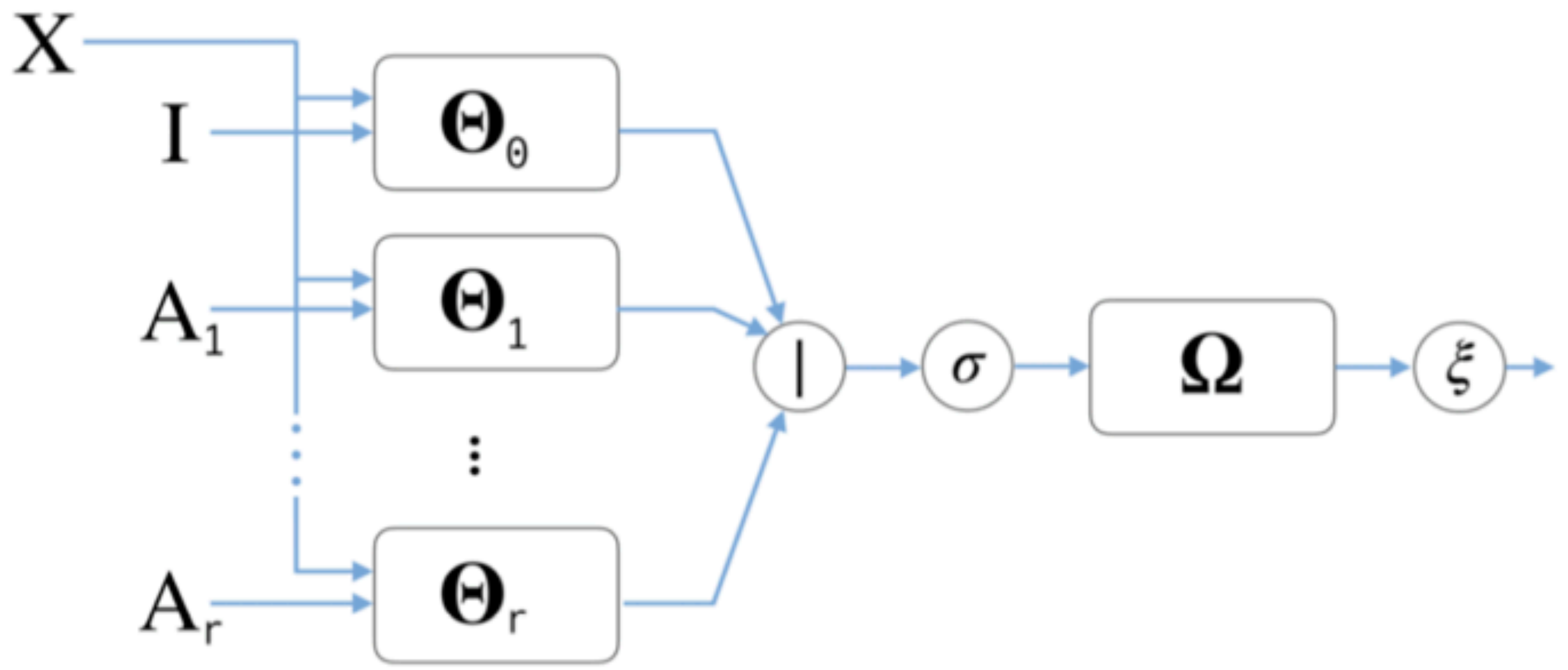
Graph attention network

$$\begin{aligned}\mathbf{y}^{(i)} &= \Big\Vert_{h=1}^{H} \left(\sum_{b=1}^{B} w_{h,b}^{(i)}\mathbf{\Theta}_b\right)\left(\sum_{j\in\mathcal{N}(i)\cup\{i\}} \frac{\mathbf{x}^{(j)}}{\sqrt{\deg(i)\deg(j)}}\right) \\ &= \Big\Vert_{h=1}^{H} \underbrace{\mathbf{\Theta}_h^{(i)}}_{\text{Varying per Node}} \underbrace{\left(\sum_{j\in\mathcal{N}(i)\cup\{i\}} \frac{\mathbf{x}^{(j)}}{\sqrt{\deg(i)\deg(j)}}\right)}_{\text{Computable via SpMM}}\end{aligned}$$

Efficient graph convolution

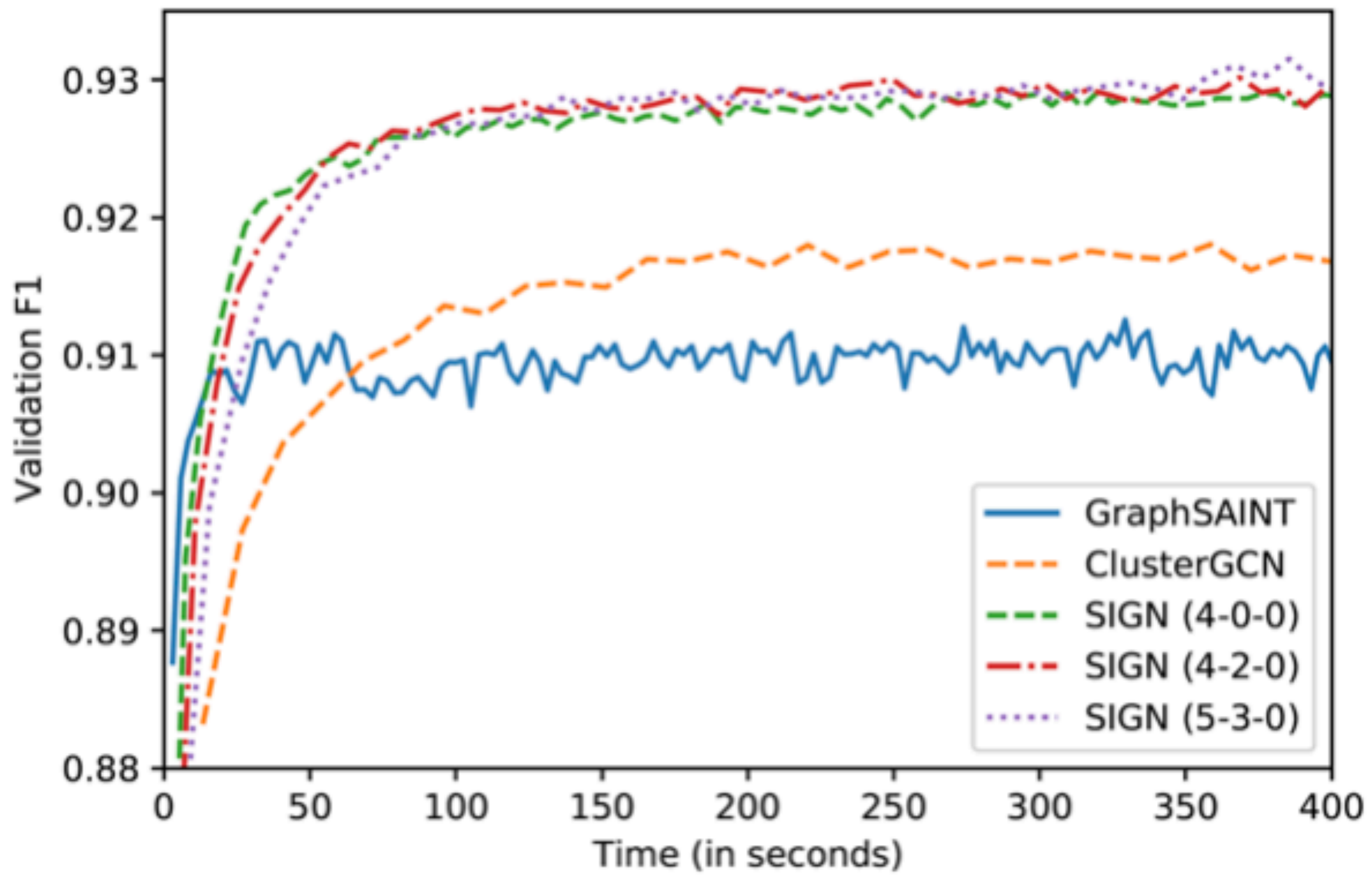
- Operations can be computed with the standard compressed sparse row (CSR) algorithm (SpMM)
- Lower memory consumption (no message materialisation)
- Requires only $O(N)$ memory

C.3. Scalable inception graph network



- diffusion operators **A** can be chosen as different powers of a graph shift operator (shallow network with wider receptive field)
- r=0 (**I**) corresponds to 1x1 convolution in inception module
- matrix products **AX** do not depend on Θ and can be pre-computed

Scalable inception graph network



Take home message:
GSP for robustness and efficiency

GSP Tools …

Graph signal regularization

Graph based transforms

Graph interpolation

Graph filtering

Diffusion operators

… for ML

Better embeddings

Higher classification accuracy in noisy settings

Stability with respect to topological noise

Faster approximation of eigendecomposition

Less complex GNNs

Outline

- Brief introduction to graph signal processing (GSP)
- Key GSP tools for machine learning
- Challenge I: GSP for exploiting data structure
- Challenge II: GSP for improving efficiency and robustness
- Challenge III: GSP for enhancing model interpretability
- Applications
- Summary, open challenges, and new perspectives

Outline

- Brief introduction to graph signal processing (GSP)
- Key GSP tools for machine learning
- Challenge I: GSP for exploiting data structure
- Challenge II: GSP for improving efficiency and robustness
- Challenge III: GSP for enhancing model interpretability
- Applications
- Summary, open challenges, and new perspectives

GSP for enhancing interpretability

A. **Domain specific:** Extract relevant knowledge from data
   - A.1. Signal analysis: Use GSP to reveal interpretable features
   - A.2. Structure inference: Use GSP to learn interpretable structures

B. **Model specific:** Improve our understanding of ML models
   - B.1. Understanding the expressive power of GNNs
   - B.2. A posteriori interpretation of DNN

GSP for enhancing interpretability

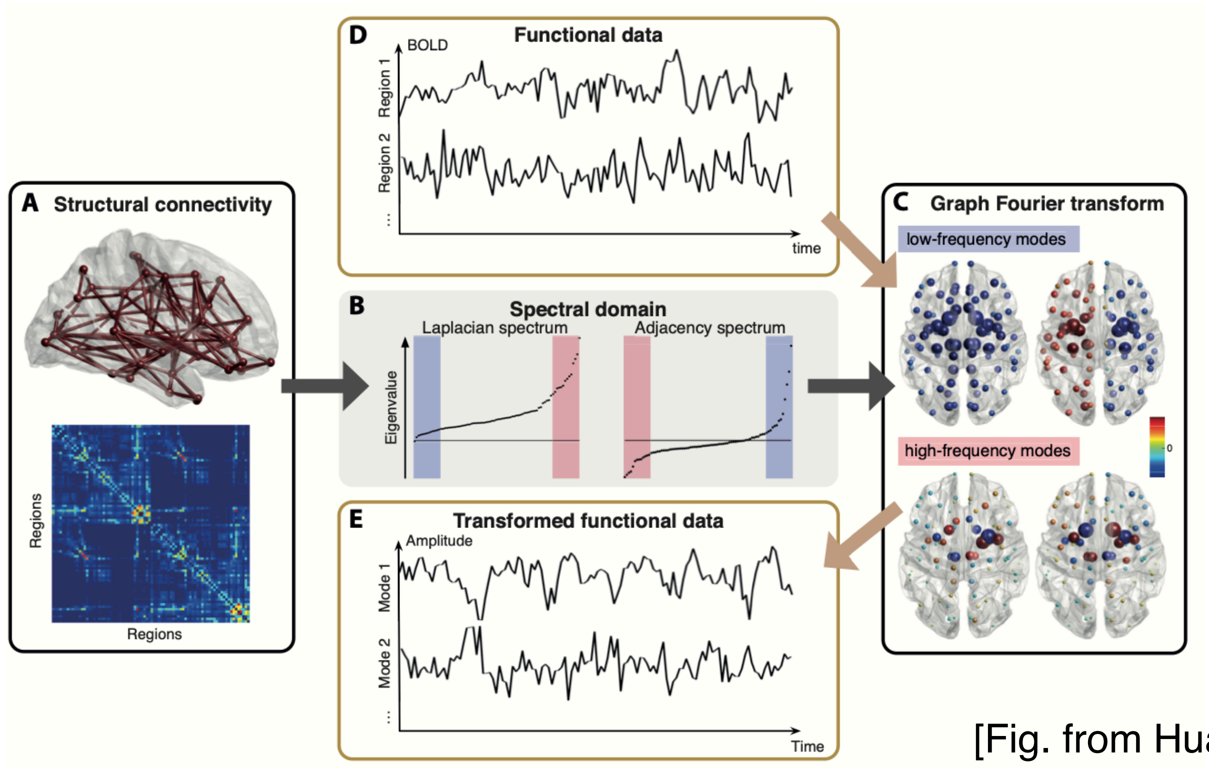
A. **Domain specific:** Extract relevant knowledge from data
  - A.1. Signal analysis: Use GSP to reveal interpretable features
  - A.2. Structure inference: Use GSP to learn interpretable structures

B. **Model specific:** Improve our understanding of ML models
  - B.1. Understanding the expressive power of GNNs
  - B.2. A posteriori interpretation of DNN

A.1. Extracting domain knowledge

- Graph based transforms have been successful in domain specific knowledge discovery

- In neuroscience, GSP tools have been used to improve our understanding of the biological mechanisms underlying human cognition and brain disorders

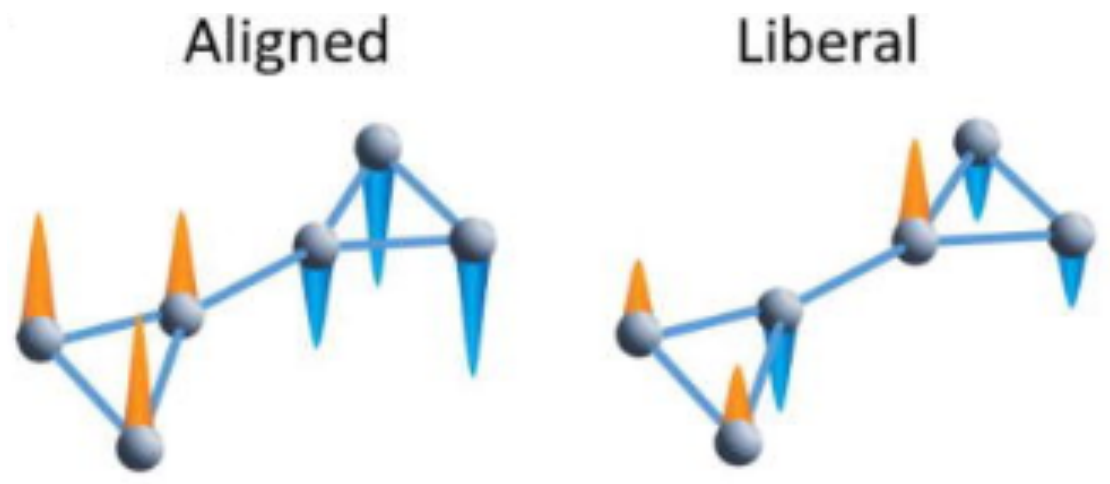


[Fig. from Huang'18]

- Analysis in the spectral domain reveals the variation of signals on the anatomical network

A.1.1. GSP for understanding cognitive flexibility

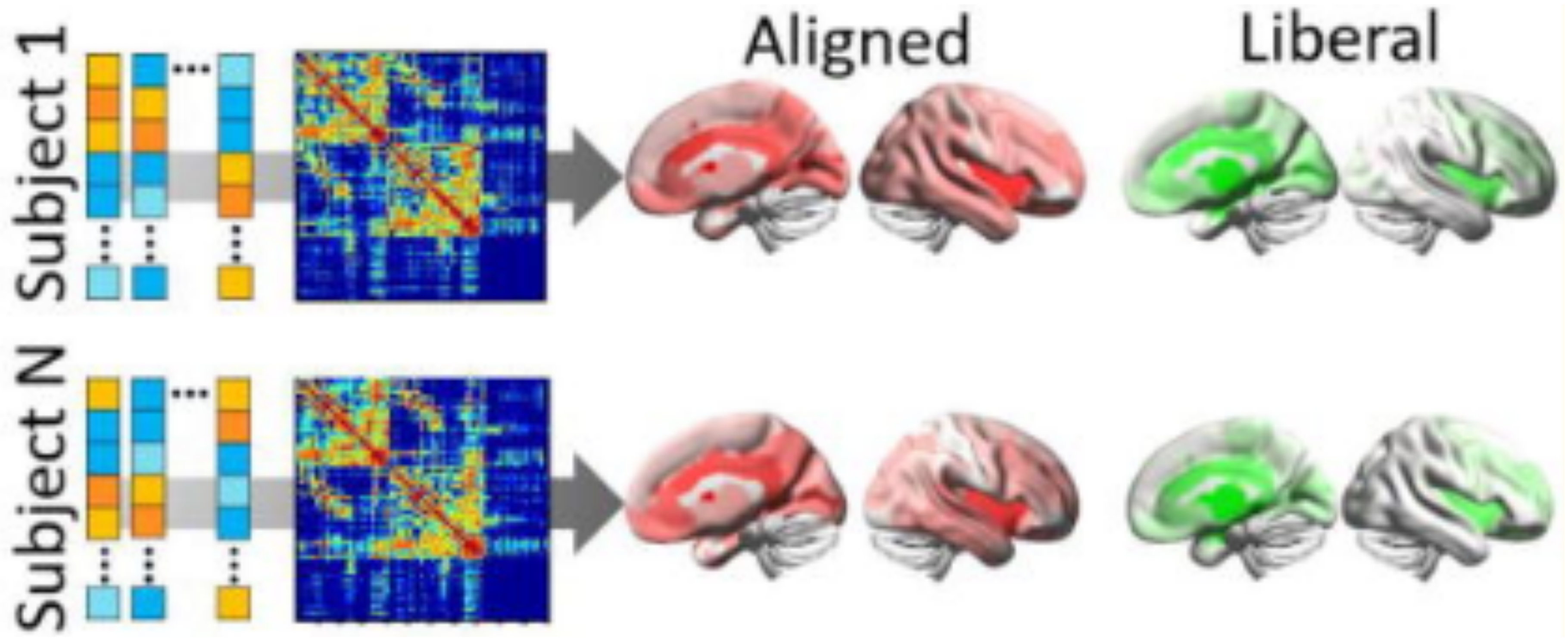
- Cognitive flexibility describes the human ability to switch between modes of mental function
- Clarifying the nature of cognitive flexibility is critical to understand the human mind
- GSP provides a framework for integrating brain network structure, function, and cognitive measures
- It allows to decompose each BOLD signal into aligned and liberal components



Medaglia et al., “Functional Alignment with Anatomical Networks is Associated with Cognitive Flexibility”, Nat. Hum. Behav., 2018

BOLD signal alignment across the brain

- Functional alignment with anatomical networks facilitates cognitive flexibility (lower switch costs)
  - Liberal signals are concentrated in subcortical regions and cingulate cortices
  - Aligned signals are concentrated in subcortical, default mode, fronto-patietal, and cingulo-opercular systems



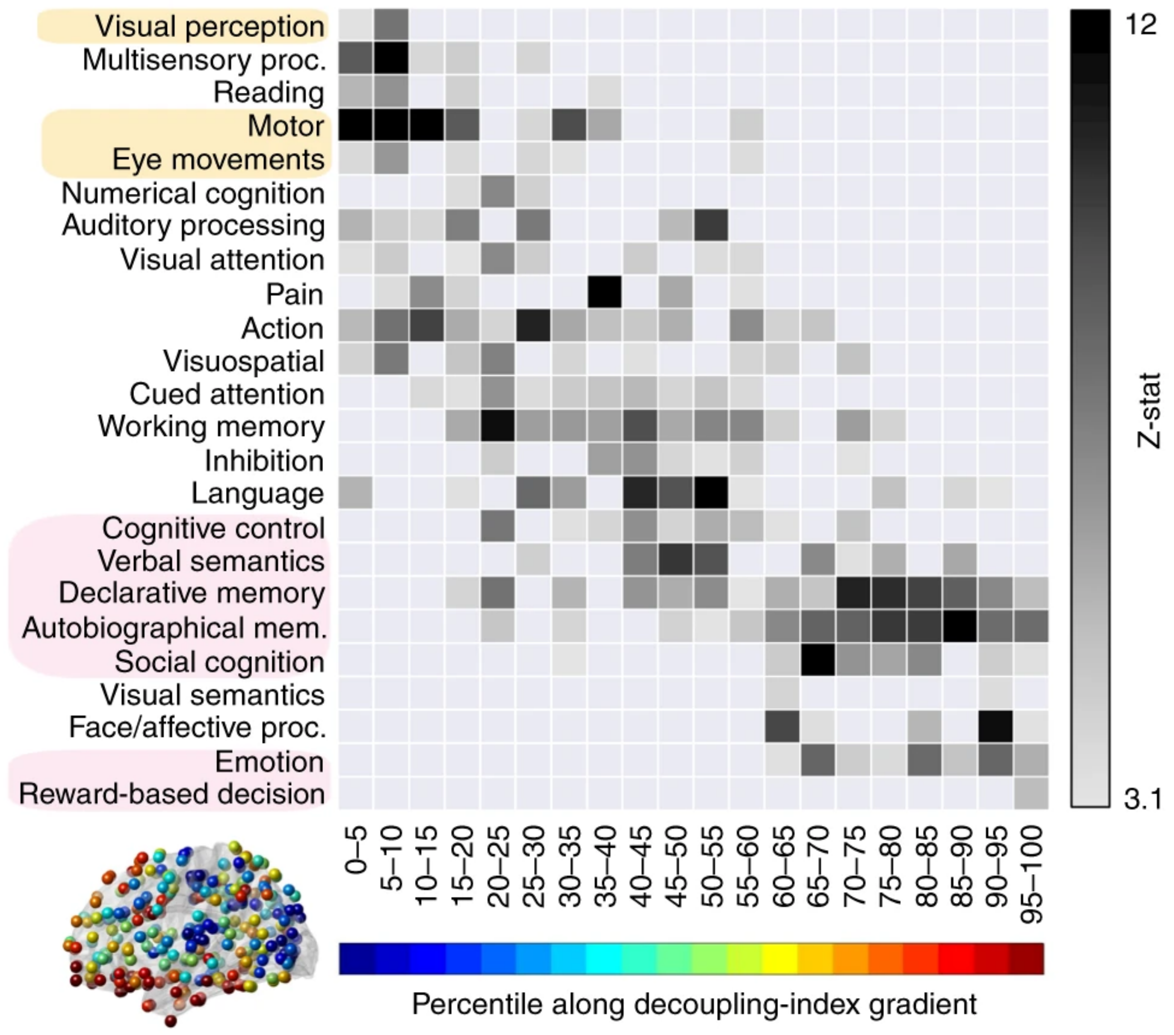
BOLD signals **White matter network** **Decomposition of the signals into aligned and liberal using GFT**

A.1.2. Structural decoupling index

- Ratio of liberal versus aligned energy in a specific node
- Spatial organization of regions according to decoupling index reveals behaviourally relevant gradient

Structurally coupled regions: Sensory regions

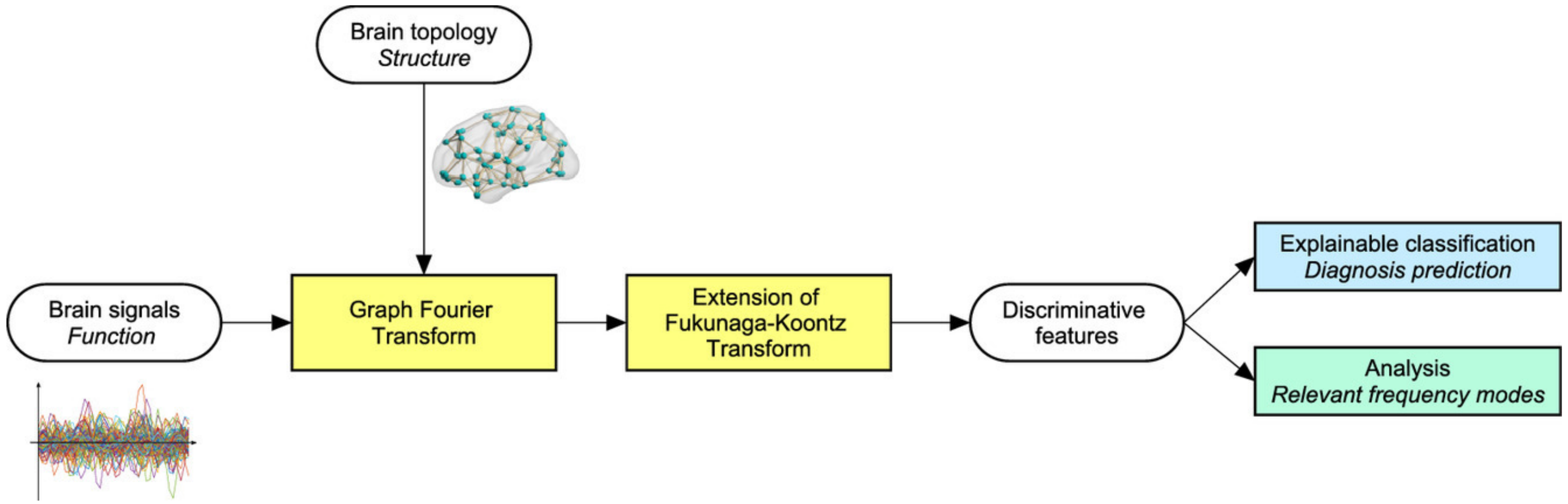
Structurally decoupled regions: Higher-level cognitive regions



Preti and Van De Ville., "Decoupling of brain function from structure reveals regional behavioral specialization in humans", Nat. Comm., 2019

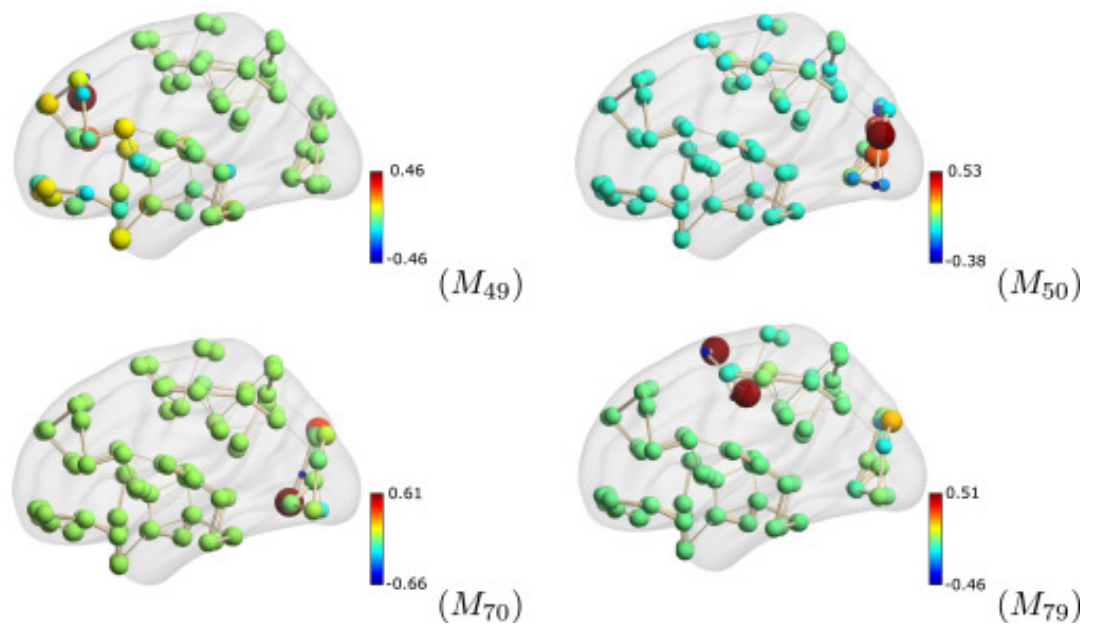
A.1.3. Understanding ASD through GSP

- Predict Autism Spectrum Disorder (ASD) by exploiting structural and functional information
- Discriminative patterns are extracted in the graph Fourier domain
- Frequency signatures are defined as the variability over time of graph Fourier modes

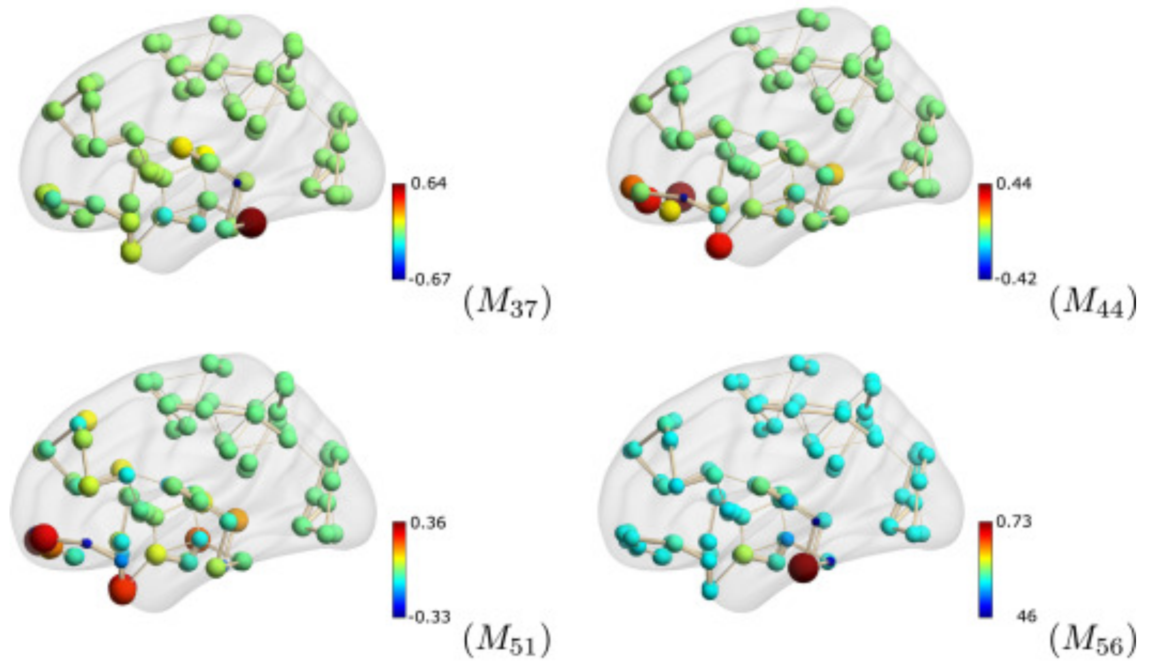


Itani and Thanou, “Combining anatomical and functional networks for neuropathology identification: A case study on autism spectrum disorder”, Medical Image Analysis, 2021

Interpretable and discriminative features



Neurotypical patients

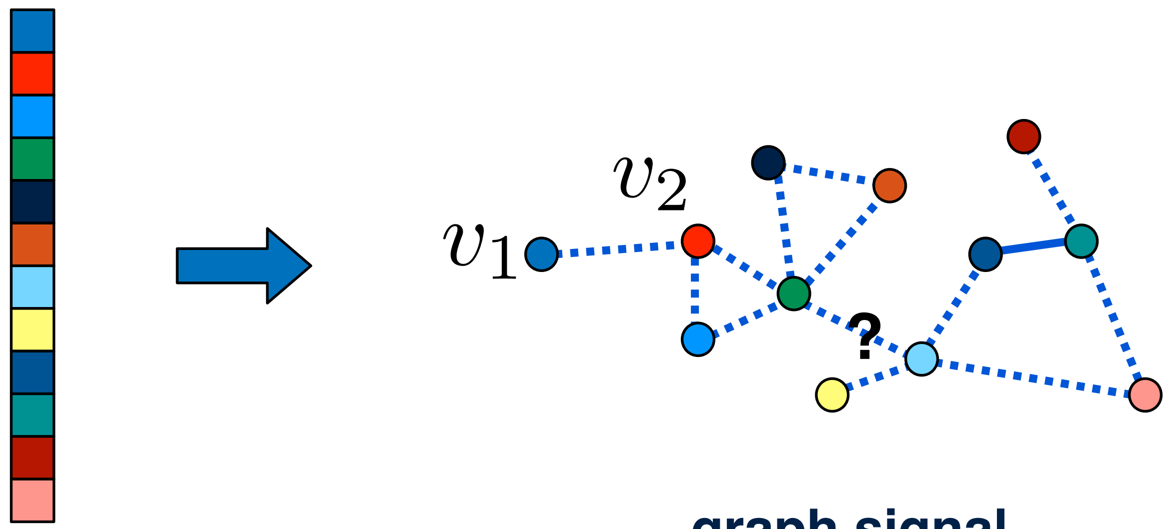


ASD patients

- Neurotypical patients express a predominant activity in the parieto-occipital regions
- ASD patients express high level of activity in the pronto-temporal areas

A.2. Structure inference

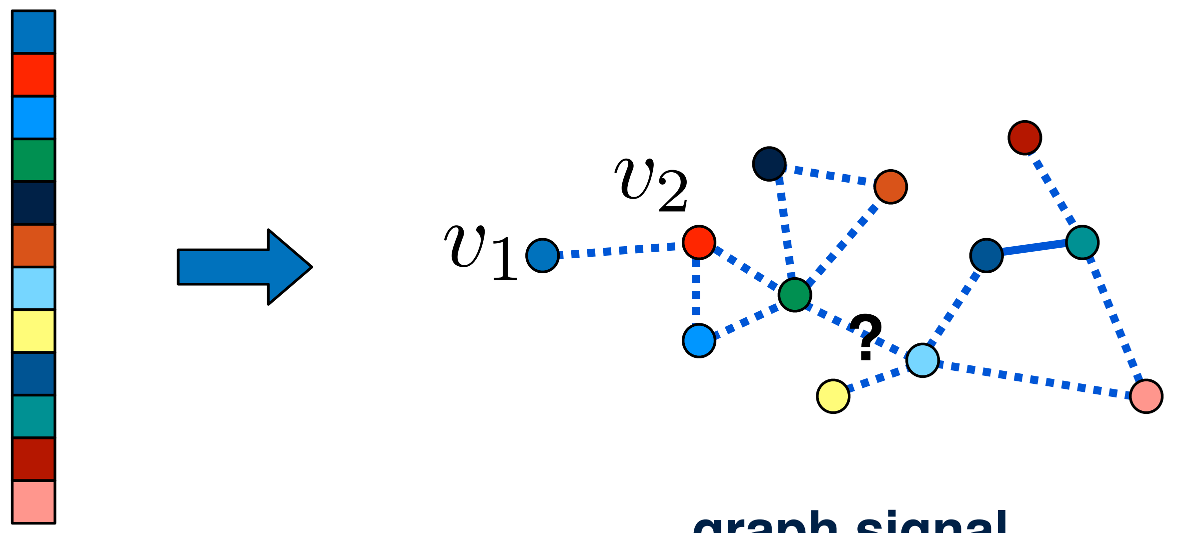
- Given observations on a number of variables, and some prior knowledge (distribution, model, constraints), learn a measure of relations between them



graph signal

A.2. Structure inference

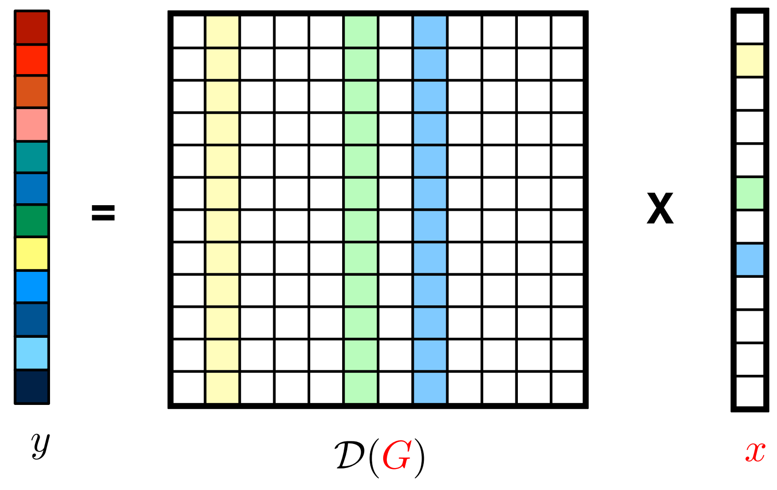
- Given observations on a number of variables, and some prior knowledge (distribution, model, constraints), learn a measure of relations between them



graph signal

How to infer interpretable structure from data?

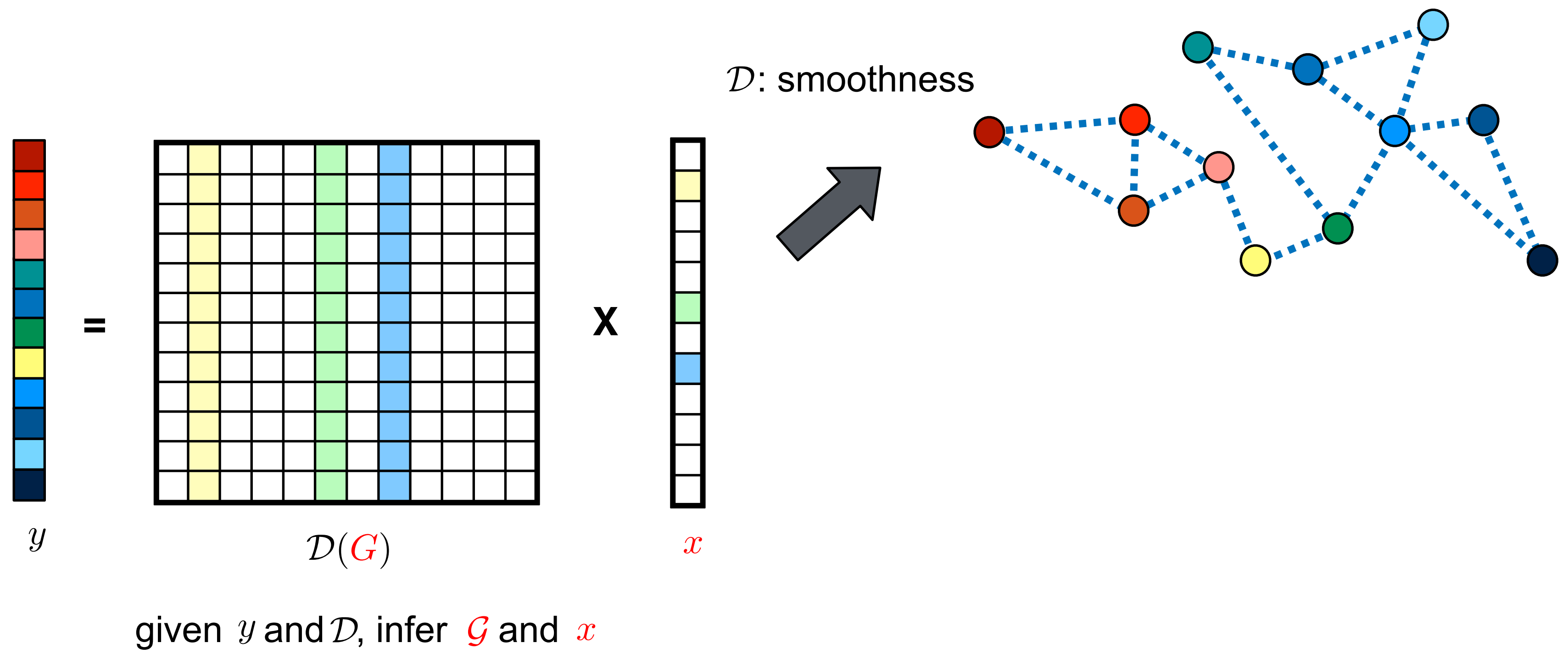
GSP for structure inference



given y and $\mathcal{D}$, infer $\mathcal{G}$ and x

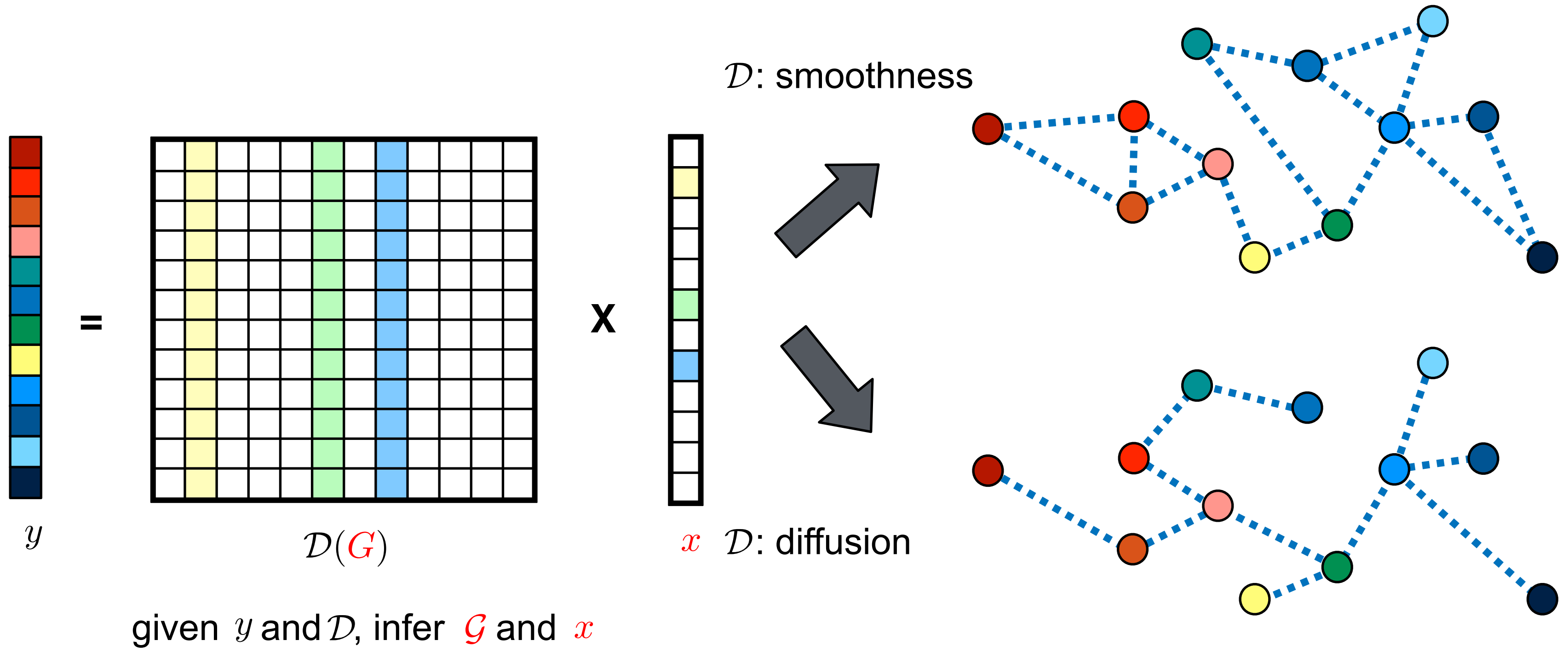
Dong et al., “Learning graphs from data: A signal representation perspective,” IEEE SPM, 2019.
Mateos et al., “Connecting the dots: Identifying network structure via graph signal processing,” IEEE SPM, 2019.

GSP for structure inference



$\mathcal{D}$: smoothness
=
X
y
$\mathcal{D}(G)$
x
given y and $\mathcal{D}$, infer $\mathcal{G}$ and x

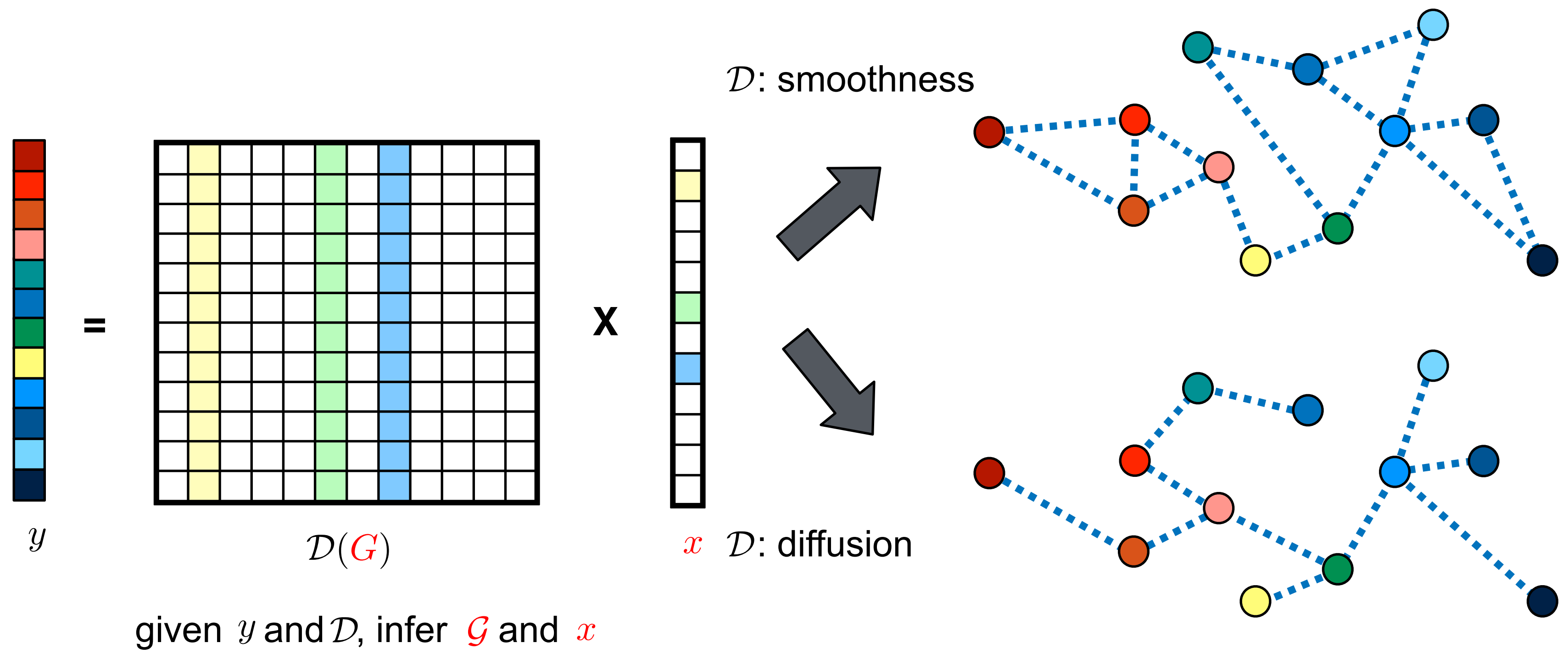
GSP for structure inference



=
x
y
$\mathcal{D}(G)$
x
$\mathcal{D}$: smoothness
$\mathcal{D}$: diffusion
given y and $\mathcal{D}$, infer $\mathcal{G}$ and x

Dong et al., “Learning graphs from data: A signal representation perspective,” IEEE SPM, 2019.
Mateos et al., “Connecting the dots: Identifying network structure via graph signal processing,” IEEE SPM, 2019.

GSP for structure inference



$\mathcal{D}$: smoothness
=
X
y
$\mathcal{D}(G)$
x
$\mathcal{D}$: diffusion
given y and $\mathcal{D}$, infer $\mathcal{G}$ and x

- Examples of signal graph models:
  - Smoothness: $\mathcal{D}(\mathcal{G}) = \chi$
  - Diffusion: $\mathcal{D}(\mathcal{G}) = e^{-\tau L}$

A.2.1 Imposing domain specific priors

- Leads to more interpretable structures
  - Genes are typically clustered into pathways
  - Bipartite graph structure is more probable for drug discovery

- Example of spatial and spectral constraints:
  - K-component graph

$$\mathcal{S}_\lambda = \{\{\lambda_j = 0\}_{j=1}^{k}, c_1 \le \lambda_{k+1} \le \cdots \le \lambda_p \le c_2\}$$

  - Connected sparse graph

$$\mathcal{S}_\lambda = \{\lambda_1 = 0, c_1 \le \lambda_2 \le \cdots \le \lambda_p \le c_2\}$$

  - K-connected d-regular graph

$$\mathcal{S}_\lambda = \{\{\lambda_j = 0\}_{j=1}^{k}, c_1 \le \lambda_{k+1} \le \cdots \le \lambda_p \le c_2\},\ diag(L) = d\mathbf{1}$$

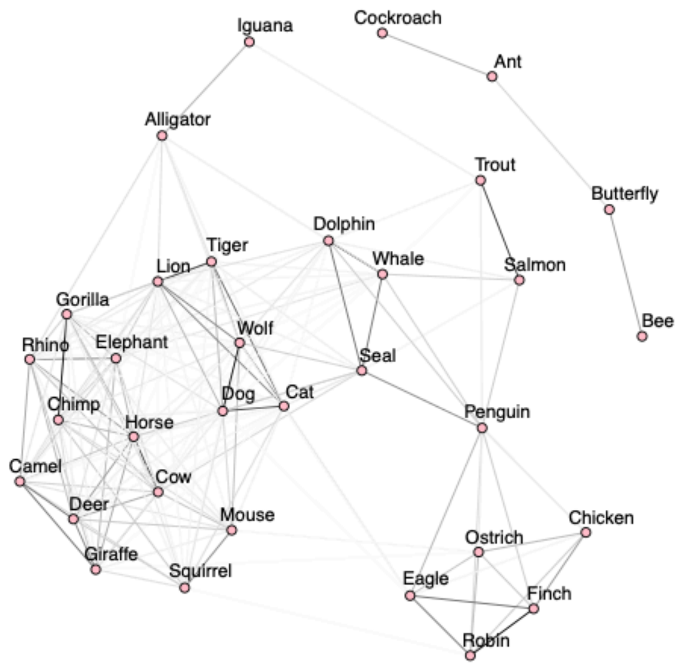
  - Cospectral graph

$$\mathcal{S}_\lambda = \{\lambda_i = f(\tilde{\lambda}_i),\ \forall i \in [1, p]\}$$

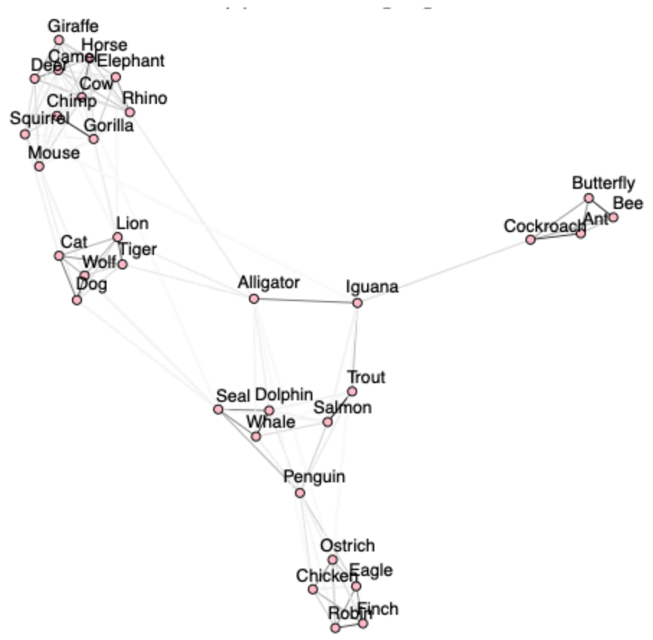
Kumar et al., “Structured Graph Learning via Laplacian Spectral Constraints”, NeurIPS, 2019

Illustrative example

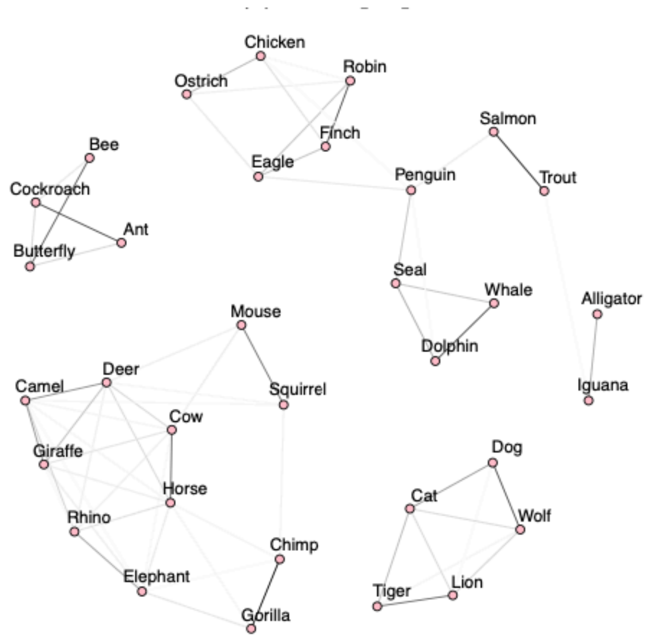
- Animals dataset



Graphical LASSO



1-component

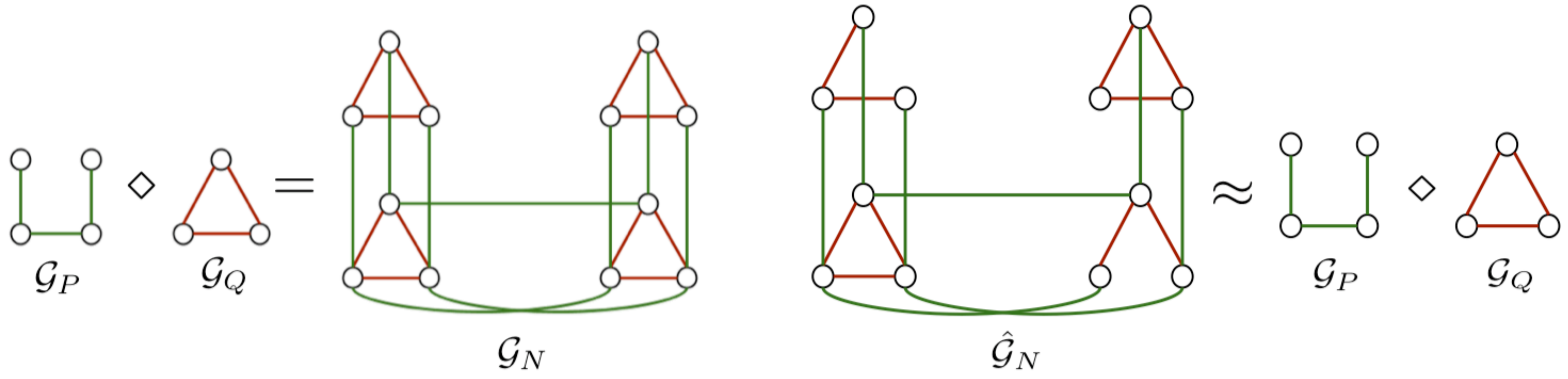


5-component

- Imposing components leads to more semantically meaningful graphs

A.2.2. Learning product graphs

- Cartesian product graphs are useful to explain complex relationships in multi-domain graph data



$\mathcal{G}_P \diamond \mathcal{G}_Q = \mathcal{G}_N$
$\hat{\mathcal{G}}_N \approx \mathcal{G}_P \diamond \mathcal{G}_Q$

exact decomposition **approximate decomposition**

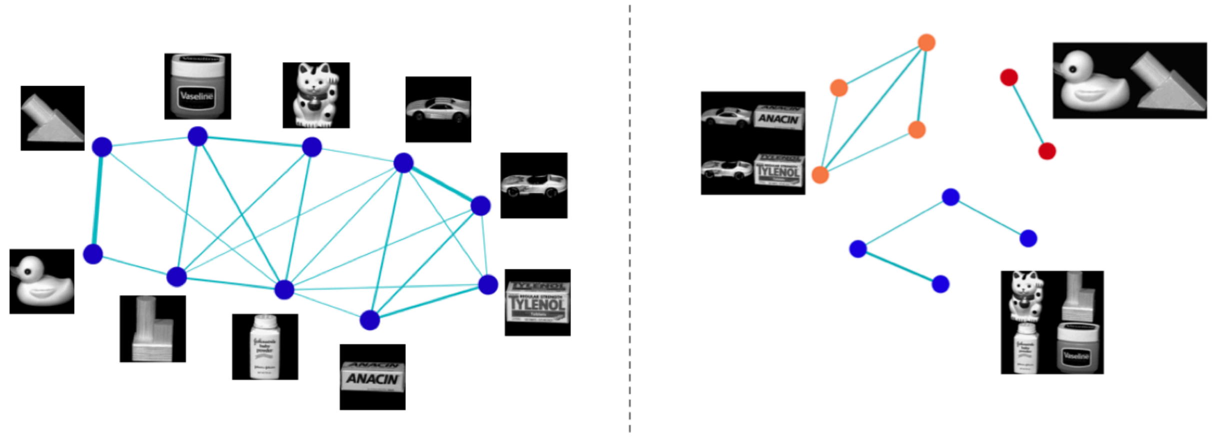
- learning graph factors with rank constraints

$$\begin{aligned}\underset{\mathbf{L}_P\in\mathcal{L}_P,\,\mathbf{L}_Q\in\mathcal{L}_Q}{\text{minimize}}\quad & \operatorname{tr}(\mathbf{L}_P\mathbf{S}_P)+\operatorname{tr}(\mathbf{L}_Q\mathbf{S}_Q)+h(\mathbf{L}_P,\mathbf{L}_Q)\\ \text{subject to}\quad & \operatorname{tr}(\mathbf{L}_P)=P,\ \operatorname{tr}(\mathbf{L}_Q)=Q,\\ & \operatorname{rank}(\mathbf{L}_P)=R_P\quad\text{and}\quad \operatorname{rank}(\mathbf{L}_Q)=R_Q\end{aligned}$$

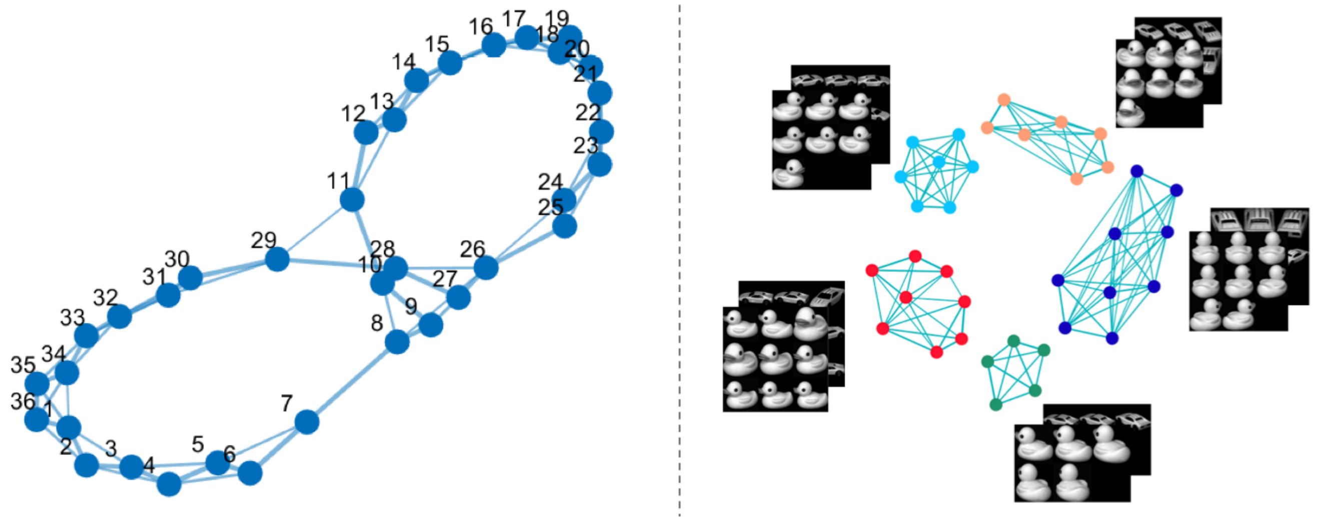
Kadambari and Chepuri, "Product graph learning from multi-domain data with sparsity and rank constraints," arXiv, 2020.

Multi-view object clustering

- COIL-20 dataset:
  - 10 objects $\Rightarrow$ Object graph of 10 nodes
  - Rotation every 10 degrees: 36 views/image $\Rightarrow$ View graph with 36 nodes



Object graph and its connected components



View graph and its connected components

GSP for enhancing interpretability

A. **Domain specific:** Extract relevant knowledge from data
  - A.1. Signal analysis: Use GSP to reveal interpretable features
  - A.2. Structure inference: Use GSP to learn interpretable structures

B. **Model specific:** Improve our understanding of ML models
  - B.1. Understanding the expressive power of GNNs
  - B.2. A posteriori interpretation of DNNs

GSP for enhancing interpretability

A. **Domain specific:** Extract relevant knowledge from data
  - A.1. Signal analysis: Use GSP to reveal interpretable features
  - A.2. Structure inference: Use GSP to learn interpretable structures

B. **Model specific:** Improve our understanding of ML models
  - B.1. Understanding the expressive power of GNNs
  - B.2. A posteriori interpretation of DNNs

B.1. A GSP perspective on the expressive power of GNNs

- A spectral analysis of GNNs provides a complementary point of view to classical Weisfeiler-Lehman (WL) test

- One step further in explaining GNNs

- A common framework for spectral and spatial GNNs

$$H^{(l+1)} = \sigma(\sum_s \underset{\substack{\uparrow \\ \text{Convolution support}}}{C^{(s)}} H^{(l)} W^{(l,s)})$$

- The frequency profile is defined as:

$$\Phi_s(\lambda) = diag^{-1}(U^T C^{(s)} \underset{\substack{\uparrow \\ \text{Eigenvectors of the Laplacian}}}{U})$$

Frequency support of well known architectures

$$\Phi_s(\lambda) = diag^{-1}(U^T C^{(s)} U)$$

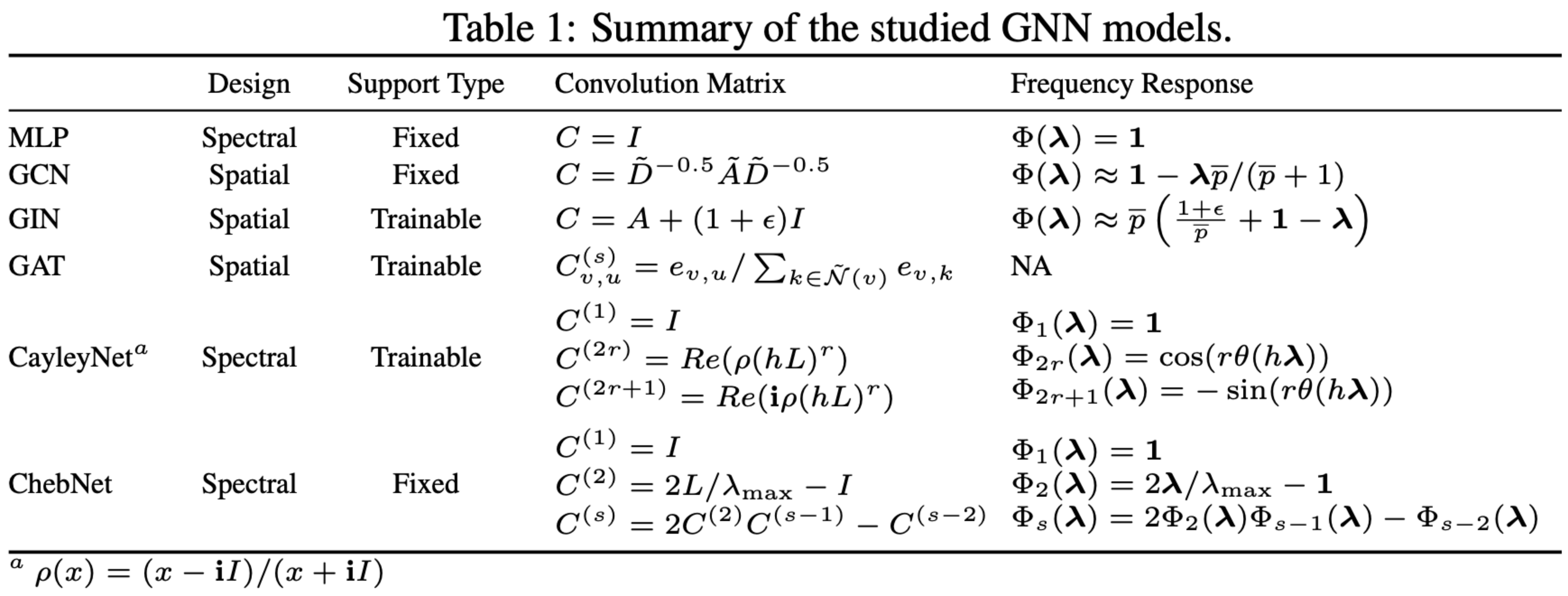


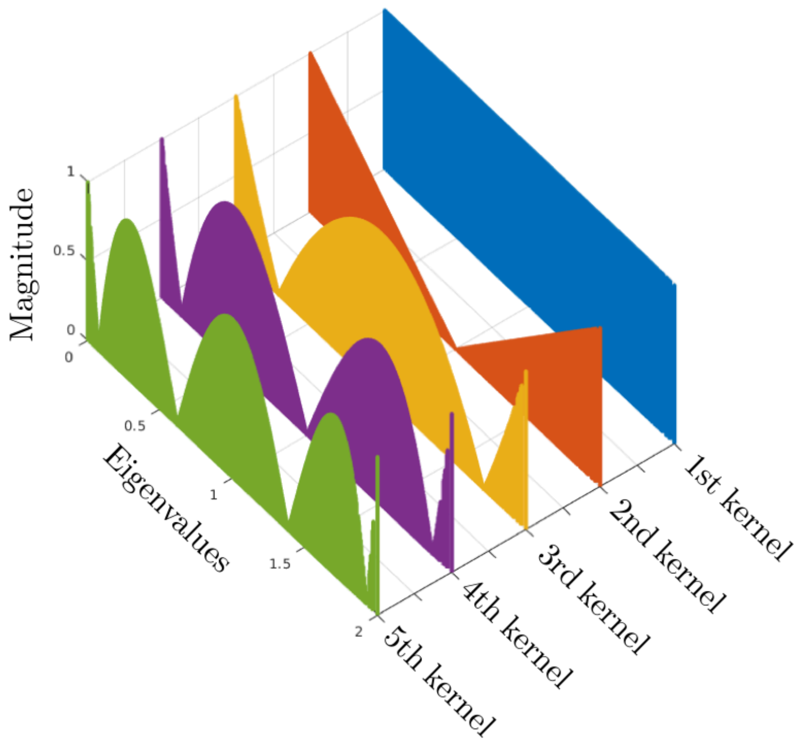
Table 1: Summary of the studied GNN models.

	Design	Support Type	Convolution Matrix	Frequency Response
MLP	Spectral	Fixed	$C = I$	$\Phi(\boldsymbol{\lambda}) = \mathbf{1}$
GCN	Spatial	Fixed	$C = \tilde{D}^{-0.5}\tilde{A}\tilde{D}^{-0.5}$	$\Phi(\boldsymbol{\lambda}) \approx \mathbf{1} - \boldsymbol{\lambda}\bar{p}/(\bar{p}+1)$
GIN	Spatial	Trainable	$C = A + (1+\epsilon)I$	$\Phi(\boldsymbol{\lambda}) \approx \bar{p}\left(\frac{1+\epsilon}{\bar{p}} + \mathbf{1} - \boldsymbol{\lambda}\right)$
GAT	Spatial	Trainable	$C^{(s)}_{v,u} = e_{v,u}/\sum_{k\in\tilde{\mathcal{N}}(v)} e_{v,k}$	NA
CayleyNet[a]	Spectral	Trainable	$C^{(1)} = I$ $C^{(2r)} = Re(\rho(hL)^r)$ $C^{(2r+1)} = Re(\mathbf{i}\rho(hL)^r)$	$\Phi_1(\boldsymbol{\lambda}) = \mathbf{1}$ $\Phi_{2r}(\boldsymbol{\lambda}) = \cos(r\theta(h\boldsymbol{\lambda}))$ $\Phi_{2r+1}(\boldsymbol{\lambda}) = -\sin(r\theta(h\boldsymbol{\lambda}))$
ChebNet	Spectral	Fixed	$C^{(1)} = I$ $C^{(2)} = 2L/\lambda_{\max} - I$ $C^{(s)} = 2C^{(2)}C^{(s-1)} - C^{(s-2)}$	$\Phi_1(\boldsymbol{\lambda}) = \mathbf{1}$ $\Phi_2(\boldsymbol{\lambda}) = 2\boldsymbol{\lambda}/\lambda_{\max} - \mathbf{1}$ $\Phi_s(\boldsymbol{\lambda}) = 2\Phi_2(\boldsymbol{\lambda})\Phi_{s-1}(\boldsymbol{\lambda}) - \Phi_{s-2}(\boldsymbol{\lambda})$

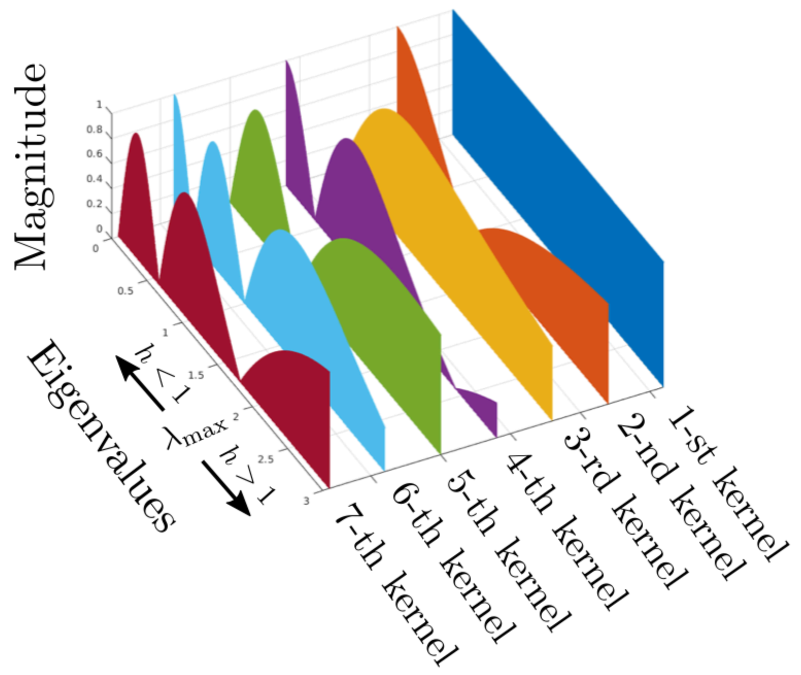
[a] $\rho(x) = (x - \mathbf{i}I)/(x + \mathbf{i}I)$

Balcilar et al., Analyzing the expressive power of graph neural networks in a spectral perspective, ICLR 2021

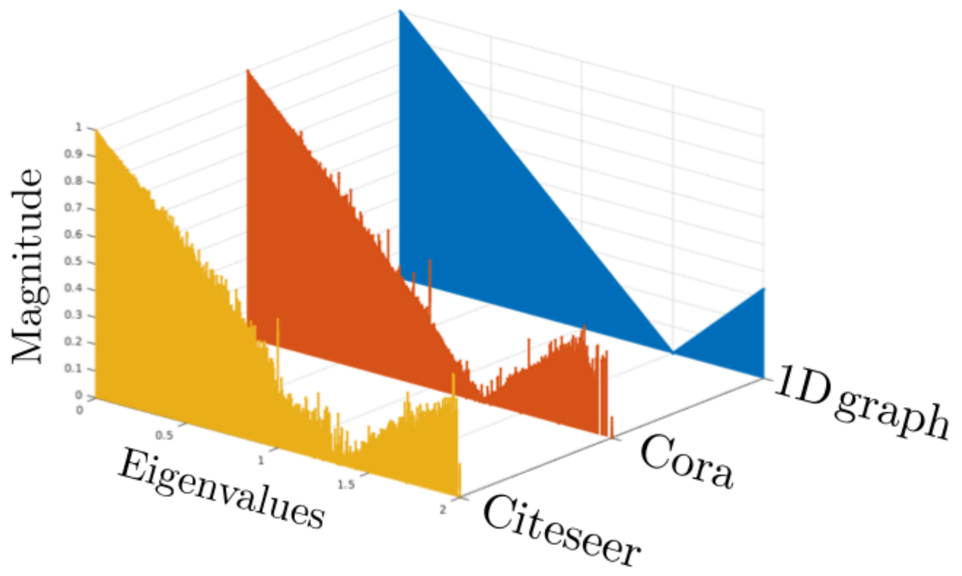
Frequency profiles of known GNNs



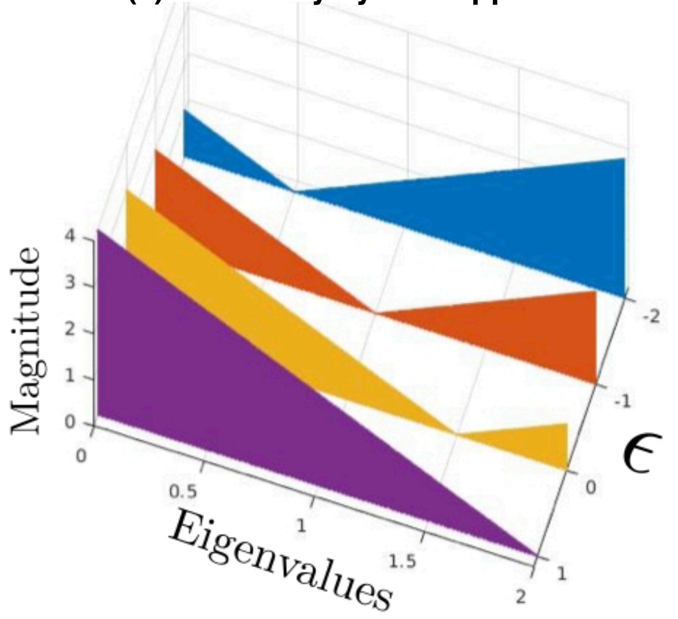
(i) first 5 ChebNet supports



(ii) first 7 CayleyNet supports



(iii) GCN frequency profiles

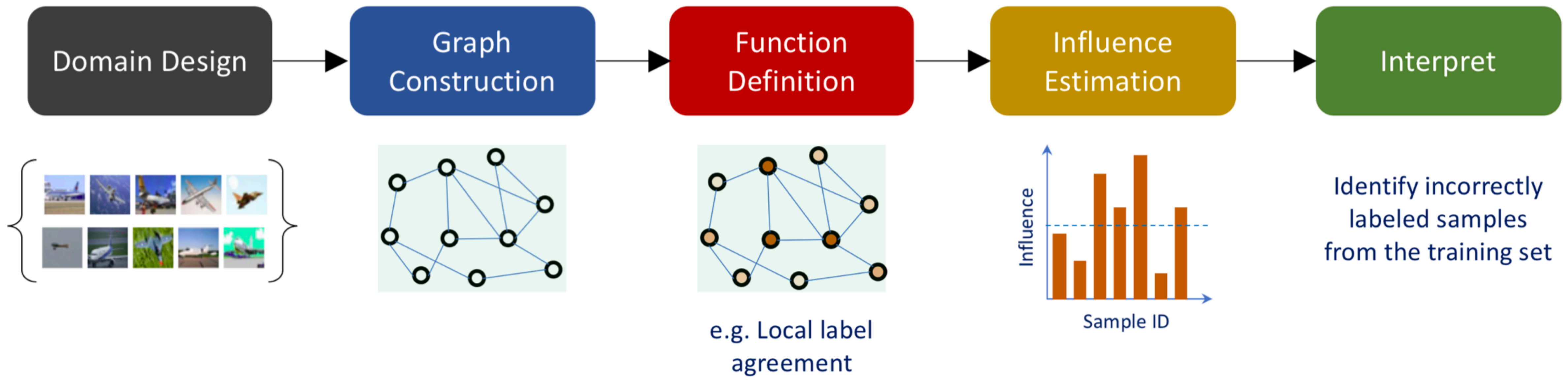


(iv) GIN on 1D

B.2. A posteriori interpretation of learning architectures

- Hypothesis: Identify relative change in model's prediction

- Model Analysis and Reasoning using Graph-based Interpretability:
  - Construct a domain (for interpretability) graph
  - Define an explanation function at the nodes of the graph
  - Choose the influential nodes by applying a high pass graph filtering
  - Generate explanations by determining influential nodes on the graph

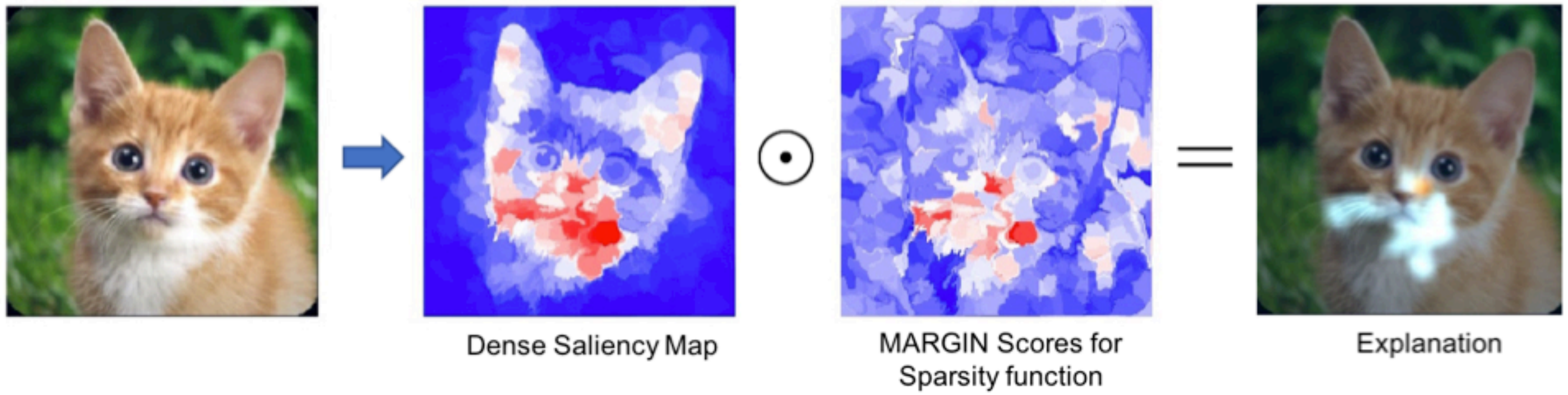


$G = (V, E, A)$ $\quad f(G)$ $\quad I(i) = \|f - Af\|_2^2, \forall i \in V$

Anirudh et al., MARGIN: Uncovering Deep Neural Networks Using Graph Signal Analysis, Frontiers in Big Data, 2021

Explanations for image classification (I)

- Nodes of the graph: superpixels from images
- Graph edges: relative importance of each superpixel
- Explanation: ratio between size of superpixel corresponding to the node and the size of the largest superpixel

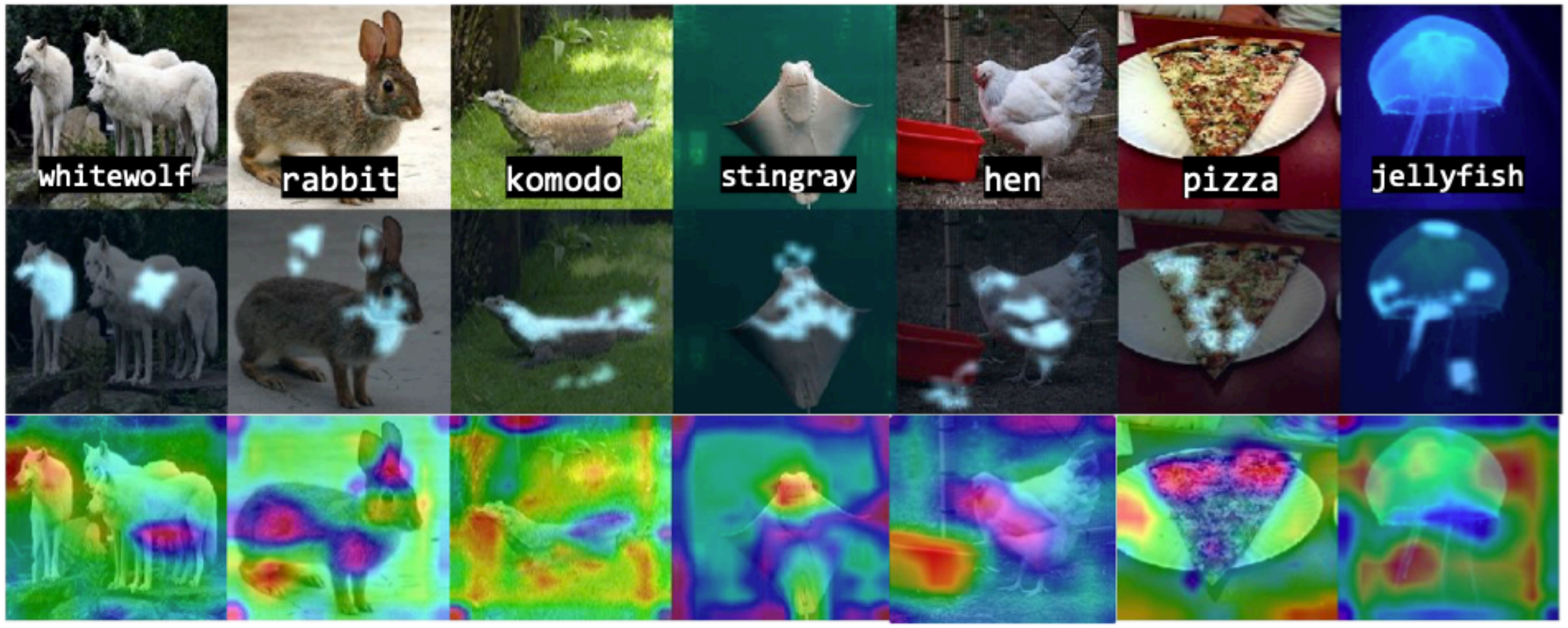


Explanations for image classification (II)

Original image

MARGIN

Grad-CAM



Interpreting decision boundaries

- Use MARGIN to identify samples that are likely to be misclassified
  - Nodes: embeddings of each sample
  - Edges: similarity between embeddings
  - Explanation function: local label agreement

(a) Most confusing samples for AlexNet pre-trained on ImageNet for the *Tabby Cat* and *Great Dane* classes

(b) Most confusing examples for a CNN trained on MNIST for the 0/6 digit classes

Take home message:
GSP for interpretability

- **Interpreting the data**
  - Graph-based transforms reveal new and interpretable features
  - Integrating them into a machine learning framework leads to more accurate and interpretable models
  - Imposing application-related constraints in topology inference algorithms generates interpretable structures
- **Interpreting the models**
  - Analysing the spectral behaviour of GNNs provides insights on their expressive power
  - GSP operators contribute towards the a posteriori interpretation of learning architectures

Summary

- GSP has shown promising results towards improving different aspects of classical ML algorithms
  - Robustness to noisy and limited data
  - Robustness to topological noise
  - Data and model interpretability

- Presented works are indicative only and non-exhaustive

- Plenty of room for exciting research!