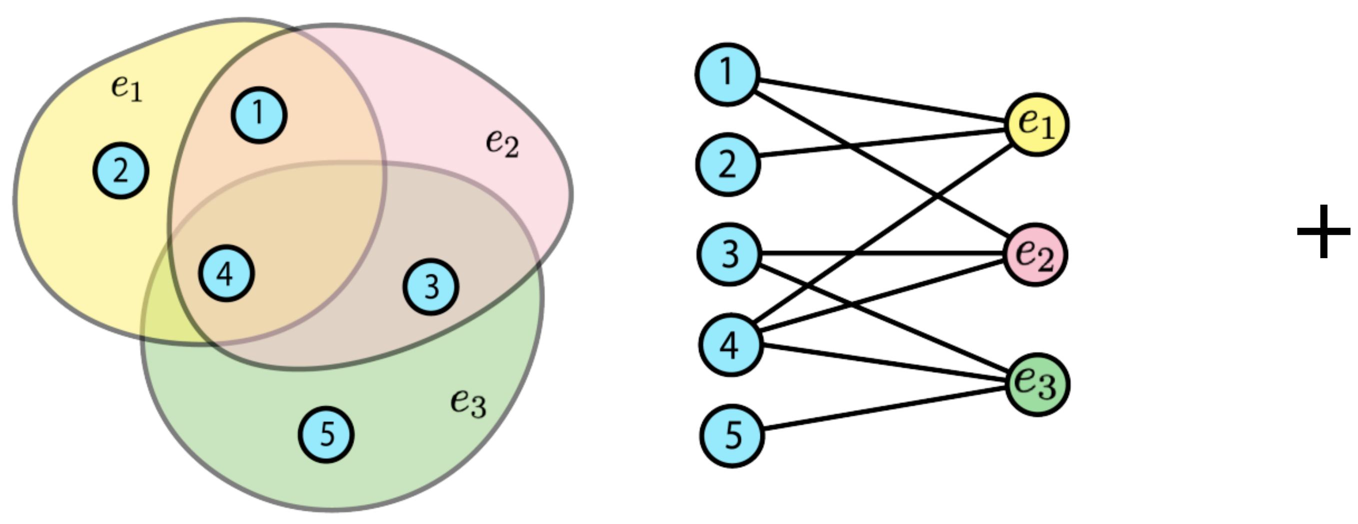
Higher-Order Expander Graph Propagation

Thomas Christie*, Yu He*,[1]

* Equal Contribution. Work done at University of Cambridge. [1] Stanford University

TLDR: We propose a framework to construct **bipartite expanders** that capture **higher-order interactions** while leveraging **expander properties**, in order to mitigate the **over-squashing** problem for GNNs.

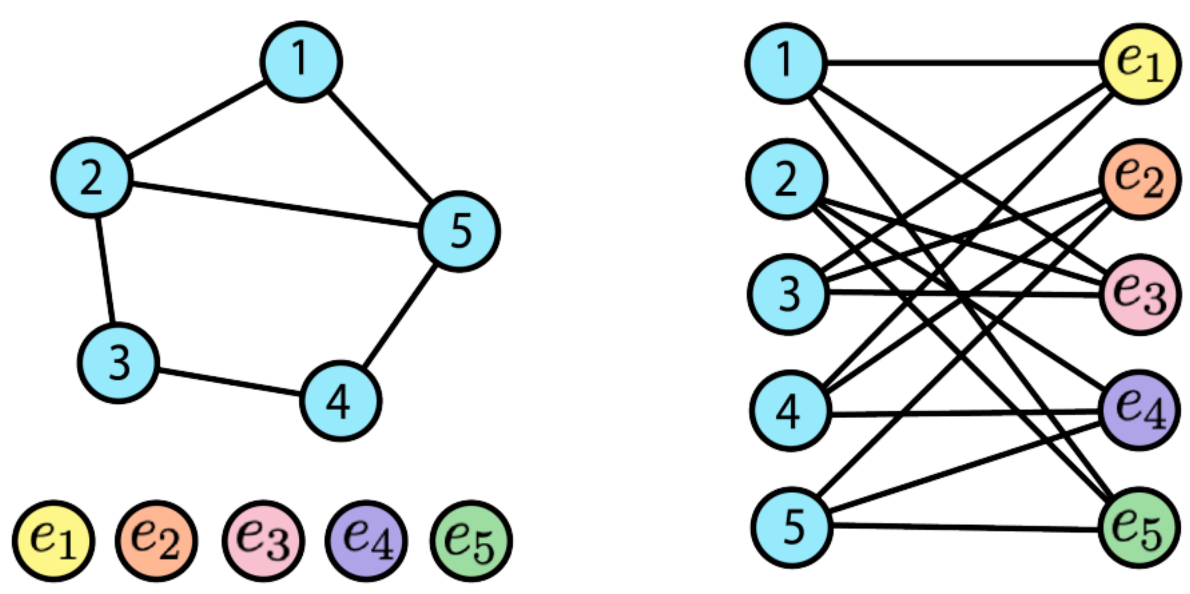
Hypergraphs as bipartite graphs



A hypergraph (left) can be represented as a bipartite graph (right), where nodes are at the left-hand side and hyperedges at the right-hand side.

Bipartite expanders to capture higher-order interactions.

Higher-Order Expander Graph Propagation



Original graph (with hyperedge nodes)

Bipartite expander

Construction of bipartite expanders:

(i) Perfect matchings

A matching on a graph is defined as a set of edges without common vertices, and a perfect matching is a matching which contains all vertices of the graph.

We construct bipartite expanders by taking the union of k disjoint perfect matchings, making them k-regular.

(ii) Ramanujan condition

A k-regular graph G is said to be Ramanujan if it satisfies the property $\lambda(\mathcal{G}) \leq 2\sqrt{k-1}$. Here, $\lambda(\mathcal{G})$ is the largest magnitude non-trivial eigenvalue.

We additionally impose Ramanujan condition that gives low diameters and high expander constants.

Message passing framework:

1. Augment the original graph with hyperedge nodes.
2. Construct bipartite expanders using perfect matchings or Ramanujan bipartite graphs.
3. Perform message-passing on the original graph.
4. Perform bi-directional message-passing on the bipartite expander graph.
5. Interleave two message-passing layers, with the original graph as the first and last layers.

Expander graphs

A k-regular graph $G = (V, E)$ is said to be a c-expander graph if

$$\frac{|\partial_{out}(\mathcal{A})|}{|\mathcal{A}|} \geq c$$

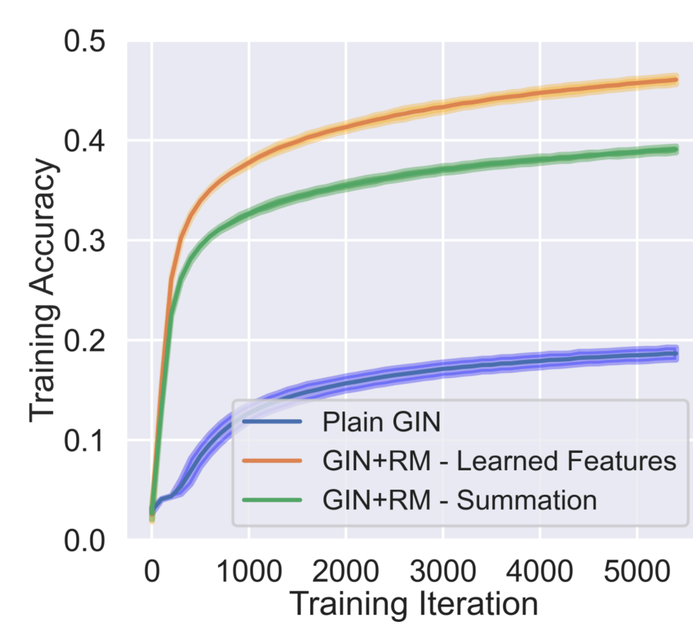
for all subsets $\mathcal{A} \subset \mathcal{V}$ with $|\mathcal{A}| \leq \frac{|\mathcal{V}|}{2}$.

Properties: highly connected, sparse graph, low diameter

Previous works [1, 2, 3] apply expander graphs in GNNs to overcome the **over-squashing problem** - where information from an exponential number of neighbors gets compressed into a fixed-size vector, leading to potential information loss .

Experimental results

(i) Tree Neighbors Match



(ii) OGBG-molhiv

Model	Test ROC-AUC
Plain GIN [40]	0.7558 ± 0.0140
EGP [20]	0.7934 ± 0.0035
GIN+PM+Learned Features	0.7742 ± 0.0104
GIN+PM+Summation	0.7751 ± 0.0138
GIN+RM+Learned Features	0.7628 ± 0.0132
GIN+RM+Summation	0.7737 ± 0.0138

Mean ± STD test ROC-AUC score. Best, Second Best and Third Best results are colored.

To deal with the hyperedge node features, we propose two methods: learn the features end-to-end (Learned Features) or perform summation during left-to-right message passing on the bipartite expander (Summation).

(iii) OGBG-code2

Model	Test F1 Score
Plain GIN [40]	0.1495 ± 0.0023
EGP [20]	0.1497 ± 0.0015
GIN + 3-Regular Bipartite Expander + Learned Features	0.1519 ± 0.0020
GIN + 3-Regular Bipartite Expander + Summation	0.1254 ± 0.0029

Mean ± STD test F1 score. Best, Second Best and Third Best results are colored.

We compare our model with GIN [4] and EGP [1], aggregating the results over 10 seeds with the same setup.

Conclusion & Future work

- We show bipartite expanders can help to alleviate over-squashing problem in GNNs by additionally capturing higher-order interactions.
- Datasets: long-range dependencies.
- Bipartite expanders: explicit construction methods.
- Bipartite message passing: hypergraph neural networks.

Bibliography

[1] Andreea Deac, Marc Lackenby, and Petar Veličković. *Expander graph propagation,* 2022.
[2] Pradeep Kr. Banerjee, Kedar Karhadkar, Yu Guang Wang, Uri Alon, and Guido Montúfar. *Oversquashing in gnns through the lens of information contraction and graph expansion,* 2022.
[3] Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J. Sutherland, Ali Kemal Sinop. *Exphormer: Sparse Transformers for Graphs,* 2023.
[4] Keyulu Xu, Weihua Hu, Jure Leskovec, Stefanie Jegelka. *How powerful are Graph Neural Networks?* 2018.

Sheaf-based Positional Encodings for Graph Neural Networks

Yu He[1], Cristian Bodnar[2], Pietro Liò[3]

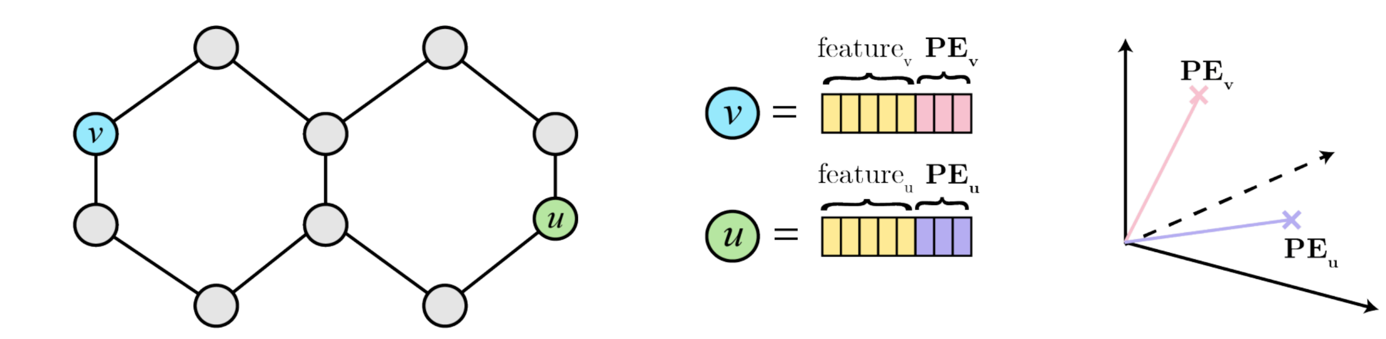
[1] Stanford University. Work done at University of Cambridge.
[2] Microsoft Research AI4Science [3] University of Cambridge

NEURAL INFORMATION PROCESSING SYSTEMS

NeurReps Workshop

TLDR: We propose to construct **positional encodings** for GNNs using **the sheaf Laplacian**, in the aim to encode both the **structural** and **semantic** information from the graph and its node data.

Positional Encodings for GNNs

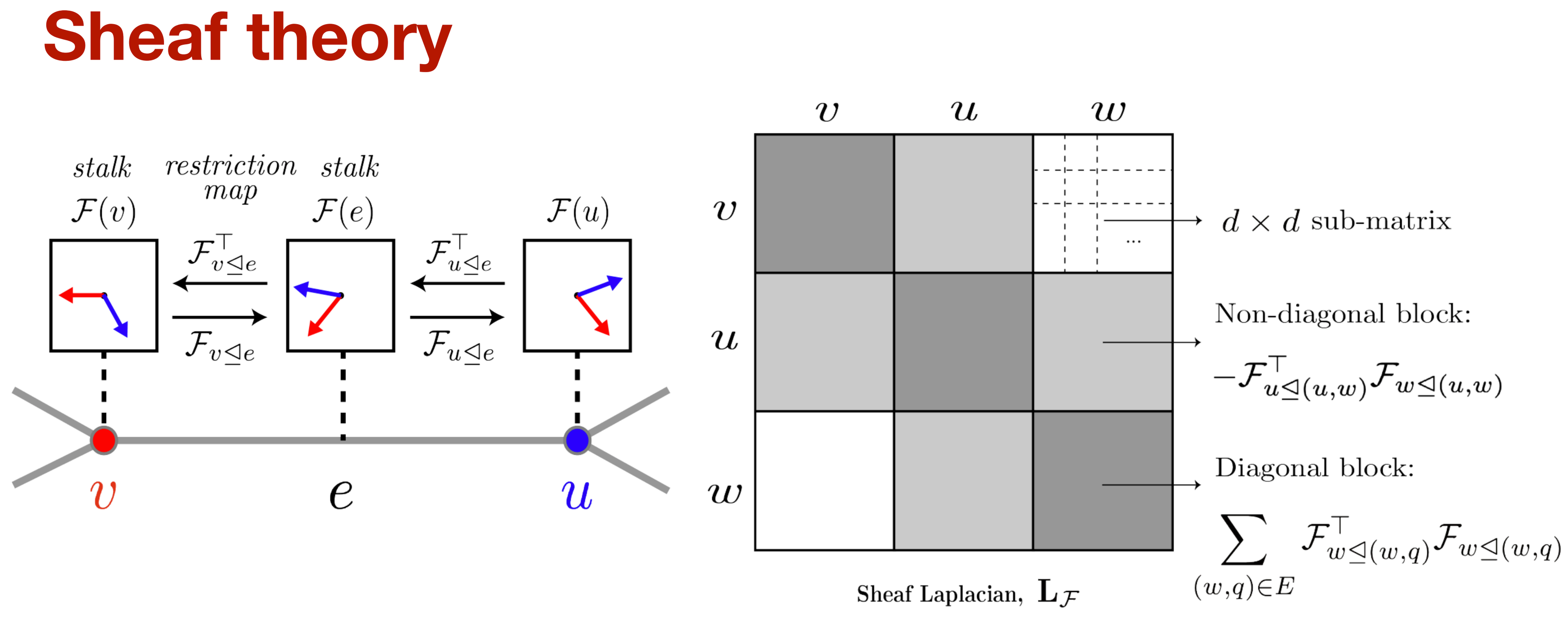


Positional encodings inform the nodes of their position in the graph, which helps to break the locality constraint from message-passing.

The graph Laplacian is a popular candidate for designing positional encodings for graphs, but it encodes purely the graph structure, without taking the node data into account.

However, **heterophilic graphs** have dissimilar nodes connected, reflected by their node features.

Sheaf theory



Cellular sheaf

A cellular sheaf $(G, \mathcal{F})$ on an undirected graph $G = (V, E)$ consists of:

- *A vector space $\mathcal{F}(v)$ for each vertex $v \in V$.*
- *A vector space $\mathcal{F}(e)$ for each edge $e \in E$.*
- *A linear map $\mathcal{F}_{v\trianglelefteq e} : \mathcal{F}(v) \to \mathcal{F}(e)$ for each incident node-edge pair $v \trianglelefteq e$.*

We call the vector spaces of the nodes and edges as **stalks**, and the linear maps as **restriction maps**.

0-cochain

The space of 0-cochains $C^0(G;\mathcal{F}) := \bigoplus_{v\in V} \mathcal{F}(v)$ is the space formed by all the stalks associated with the nodes of the graph, where $\bigoplus$ denotes the direct sum of vector spaces.

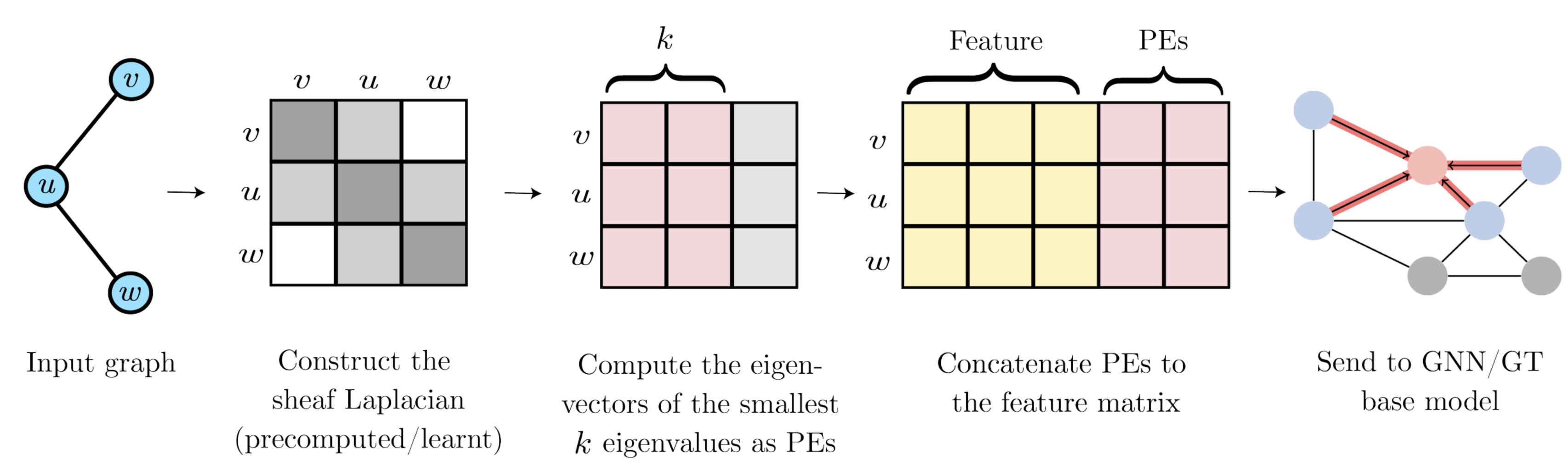
The sheaf Laplacian operator for a given cellular sheaf measures the aggregated "disagreement of opinions" at each node.

Sheaf Laplacian

Given a cellular sheaf $(G;\mathcal{F})$, the sheaf Laplacian is a linear map $\mathbf{L}_\mathcal{F} : C^0(G,\mathcal{F}) \to C^0(G,\mathcal{F})$, which can be defined node-wise as $\mathbf{L}_\mathcal{F}(\mathbf{x})_v = \sum_{v,u\trianglelefteq e}(\mathcal{F}_{v\trianglelefteq e}\mathbf{x}_v - \mathcal{F}_{u\trianglelefteq e}\mathbf{x}_u)$. Here, $\mathbf{x} \in C^0(G;\mathcal{F})$ is a 0-cochain, and $\mathbf{x}_v$ is the vector in $\mathcal{F}(v)$ of node v.

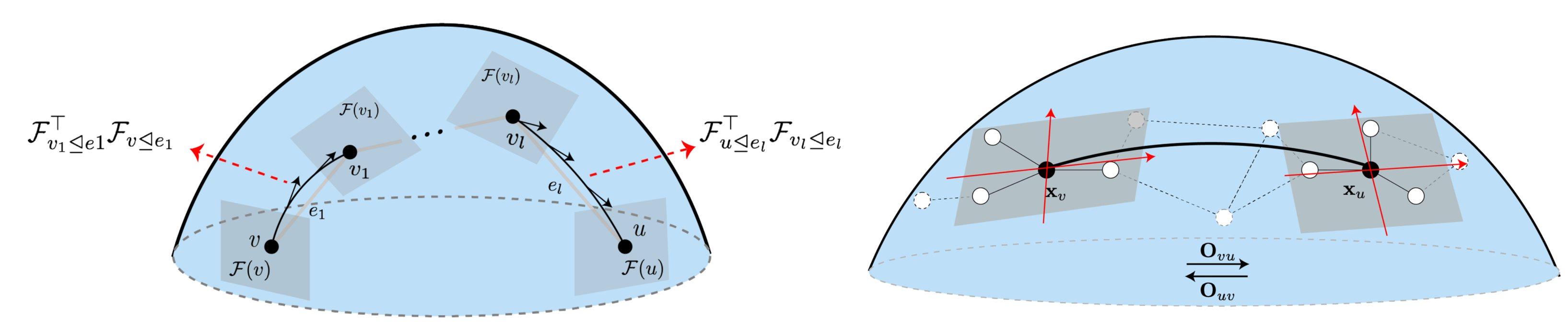
The sheaf Laplacian generalises the graph Laplacian: The graph Laplacian is a trivial sheaf, by setting all the stalks to scalars (d = 1, where d is the stalk dimension) and the restriction maps to identity functions.

Sheaf-based positional encodings



We propose to construct the sheaf-based positional encodings via precomputing or learning the sheaf Laplacian.

(i) Precomputed sheaf Laplacian (ConnLap)



The connection Laplacian is a special form of the sheaf Laplacian with an orthogonal matrix. It can be thought of as a discretised representation of the vector bundle, which draws an analogy to the concept of parallel transport on a manifold. We can compute the connection Laplacian by optimally aligning the orthonormal bases [1].

Graph-level tasks:

GatedGCN	ZINC TestMAE (↓)	ZINC+LSPE TestMAE (↓)	MOLTOX21 TestAUC (↑)
No PE	0.251±0.009	N.A.	77.2±0.6
GraphLap	**0.202±0.006**	0.196±0.008	77.4±0.7
ConnLap	0.249±0.005	**0.193±0.014**	**77.9±0.2**

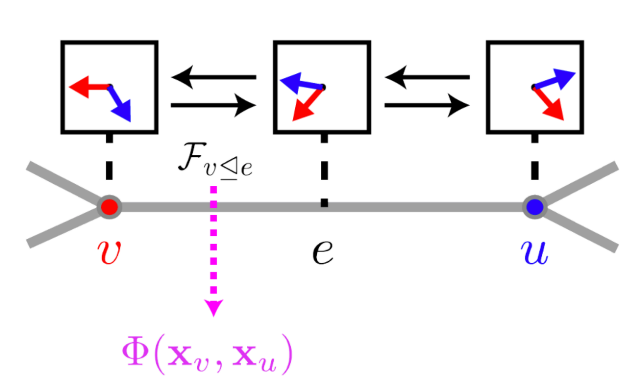
MOLTOX21

	GatedGCN	PNA	SAN
	TestAUC (↑)		
No PE	77.2±0.6	**75.5±0.8**	74.4±0.7
GraphLap	77.4±0.7	75.2±1.3	73.6±0.3
ConnLap	**77.9±0.2**	75.3±0.4	**74.5±0.4**

Mean±std MAE (↓) for ZINC and mean±std AUC (↑) for MOLTOX21.

We additionally allows ConnLap to evolve following LSPE [3].

(ii) Learnt sheaf Laplacian (SheafLap)



We approximate the restriction map using a learnable parametric function $\Phi : \mathbb{R}^{d\times 2} \to \mathbb{R}^{d\times d}$ [2]. That is, $\mathcal{F}_{v\trianglelefteq e:=(v,u)} = \Phi(\mathbf{x}_v, \mathbf{x}_u)$, where $\mathbf{x}_v$ and $\mathbf{x}_u$ are node features for nodes v and u.

Node-level tasks:

	Texas	Wisconsin	Film	Squirrel	Chameleon	Cornell	Citeseer	Pubmed	Cora
Hom level	0.11	0.21	0.22	0.22	0.23	0.30	0.74	0.80	0.81
#Nodes	183	251	7,600	5,201	2,277	183	3,327	18,717	2,708
#Edges	295	466	26,752	198,493	31,421	280	4,676	44,327	5,278
#Classes	5	5	5	5	5	5	7	3	6
No PE	57.30±5.51	49.80±6.80	**25.20±0.69**	46.62±3.62	63.97±3.10	45.95±6.84	72.34±1.41	**86.43±0.35**	84.71±1.23
GraphLap	58.22±7.03	**55.49±12.46**	25.13±0.99	47.56±3.03	64.28±3.00	**51.35±7.15**	73.83±2.07	86.43±0.36	85.05±1.47
ConnLap	**58.38±7.76**	**57.65±6.63**	**26.53±0.86**	**47.92±3.53**	**65.57±2.52**	**52.97±7.37**	**73.88±1.84**	**86.49±0.42**	**85.13±1.34**
SheafLap	**61.08±6.19**	54.51±7.22	23.80±1.10	**51.11±2.95**	**65.2±3.10**	48.38±5.05	**74.35±1.64**	85.84±0.65	**85.88±1.26**

Mean±std accuracy with decreasingly heterophilic graphs. Best and Second Best are coloured.

Conclusion & Future work

- The sheaf Laplacian outperforms the graph Laplacian in designing positional encodings by additionally taking the node data into account, especially for heterophilic graphs.
- What next? Learnt sheaf Laplacian on graph-level tasks; sign and basis invariance; theoretical proofs.

Bibliography

[1] Federico Barbero, Cristian Bodnar, Haitz Sáez de Ocáriz Borde, Michael Bronstein, Petar Veličković, and Pietro Liò. *Sheaf Neural Networks with Connection Laplacians, 2022.*

[2] Cristian Bodnar, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Liò, and Michael M. Bronstein. *Neural Sheaf Diffusion: A Topological Perspective on Heterophily and Oversmoothing in GNNs,* 2022.

[3] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. *Graph neural networks with learnable structural and positional representations,* 2021.