Enhancing Graph Neural Networks with Geometric Structure Analysis

Maysam Behmanesh

behmanesh@lix.Polytechnique.fr

Nov. 25th - 2024

Outlines

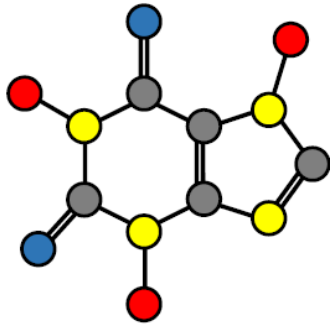
1. Machine learning on graphs
2. Graph Neural Networks (GNNs)
3. Challenges on GNNs
4. Innovative methods
5. Conclusions

Motivation

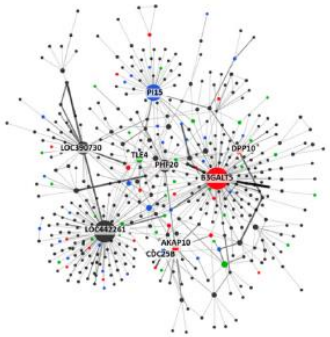
- **Graphs are Everywhere!**

Growth of ***diverse*** graph based data: Social networks, Molecules, Interaction networks, Bio-medical imaging ….

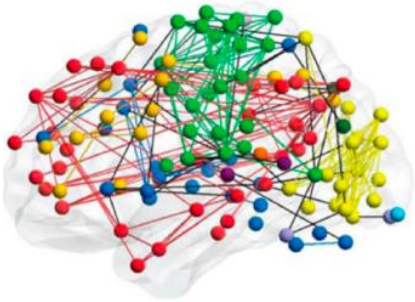
Social networks



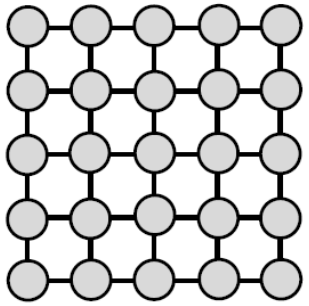
Molecules



Interaction networks



Functional networks

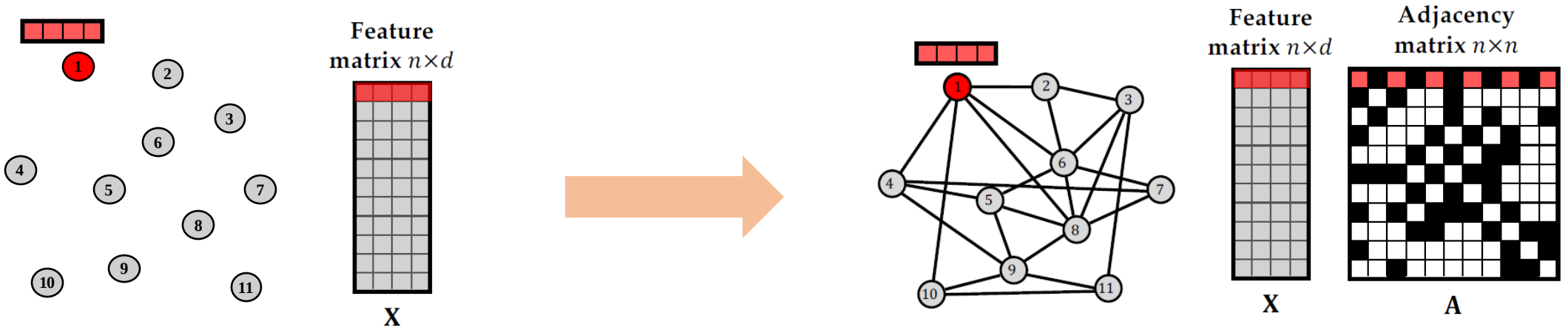


Grids

Motivation

- **Implicit graphs**

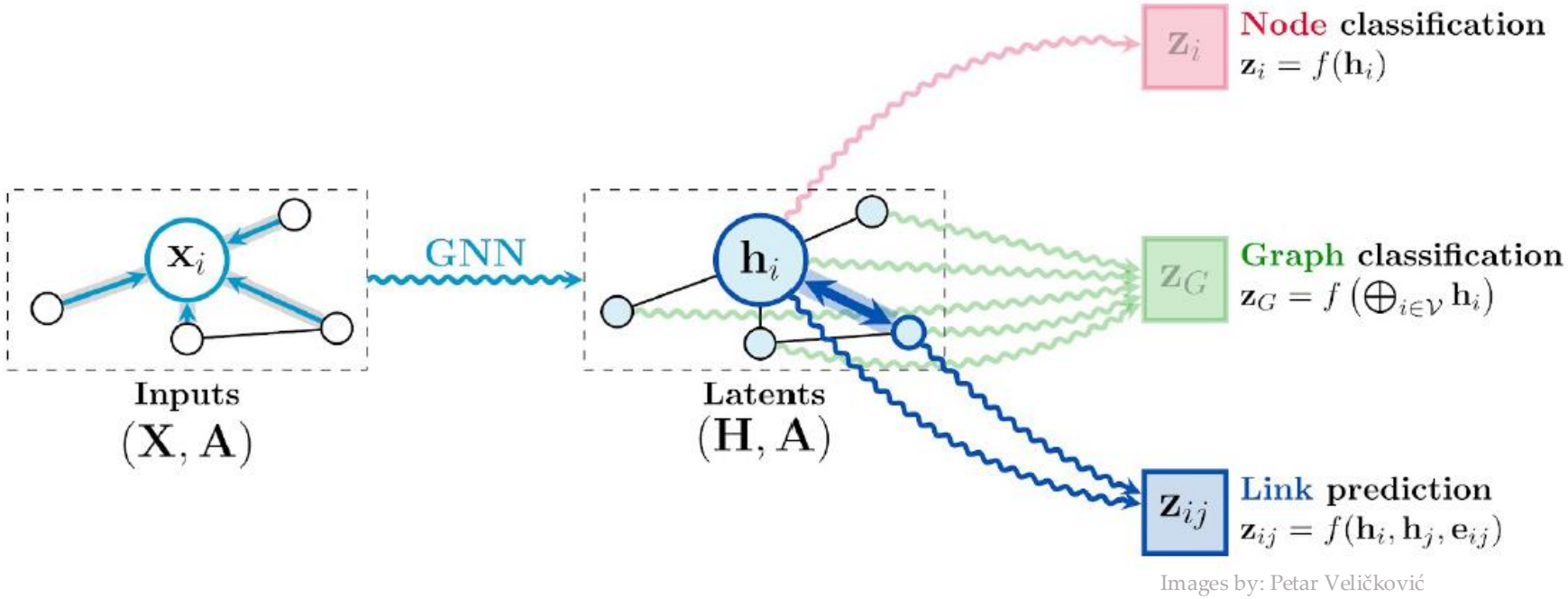
Inject geometric information into point cloud to form an ***implicit graph***



Images by: Michael Bronstein

Graph Neural Networks

- Input: graph-structured data
- Output: function that map graph to *learned representation*



Images by: Petar Veličković

- How to learn representation?
- *Message passing mechanism* aggregates information from neighbors to update the representation

Graph Neural Networks

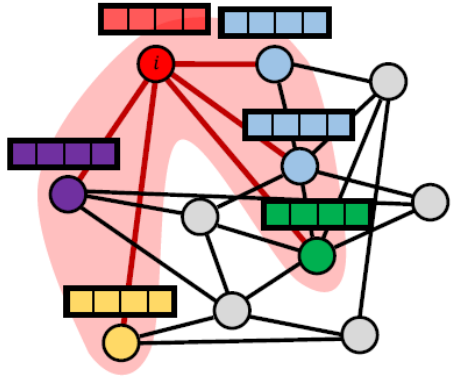
- **Message Passing mechanism**

1. Node $\mathbf{x}_i$ gathers messages from its neighbors $\mathcal{N}(\mathbf{x}_i)$

$$\mathbf{m}_i^{(l)} = \text{AGGREGATE}(\{\mathbf{h}_j^{(l-1)} : j \in \mathcal{N}(i)\})$$

2. The representation of $\mathbf{x}_i$ is updated

$$\mathbf{h}_i^{(l)} = \text{UPDATE}(\mathbf{h}_i^{(l-1)}, \mathbf{m}_i^{(l)})$$

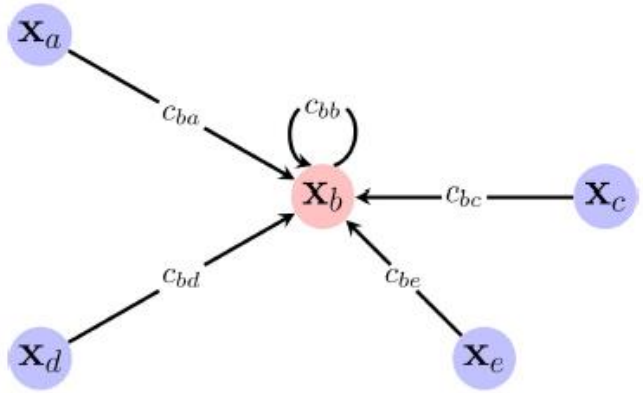


$$\mathbf{X}_{\mathcal{N}_i} = \{\mathbf{x}_{j \in \mathcal{N}_i}\}$$

Graph Neural Networks

- **Graph Convolutional Network (GCN)**
- Node features aggregate using $\widehat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$

$$\mathbf{H}^{(l+1)} = ReLU\left(\widehat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$



- Aggregation from more distant nodes:

Iteratively performing the message-passing and update node features (3-layer GCN)

$$\mathbf{H}^{(4)} = ReLU\left(\widehat{\mathbf{A}}\, ReLU\left(\widehat{\mathbf{A}}\, ReLU\left(\widehat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(1)}\right)\mathbf{W}^{(2)}\right)\mathbf{W}^{(3)}\right)$$

Useful for **homophilic** graphs !

Thomas N. Kipf & M. Welling, "Semi-supervised classification with graph convolutional networks," ICLR, 2017

Challenges…

Questions:
1- How to apply GCN on heterophilic graphs?
2- How to have long-distance communication between nodes with GCN?

GCN with higher-order layers
The models is prone to *oversmoothing*

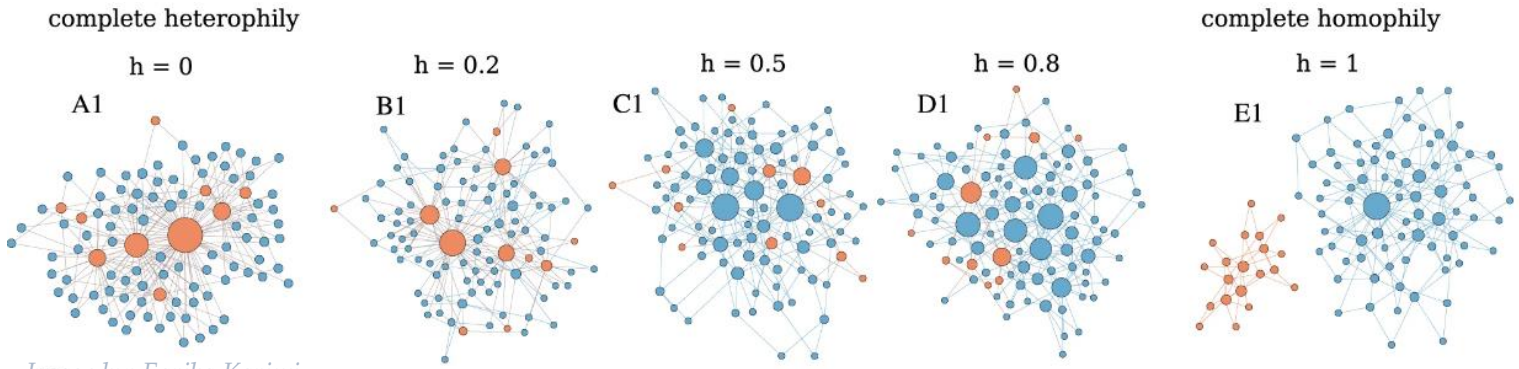
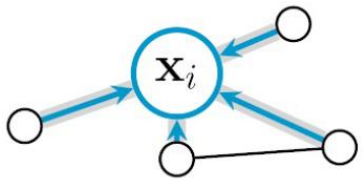


Image by: Fariba Karimi

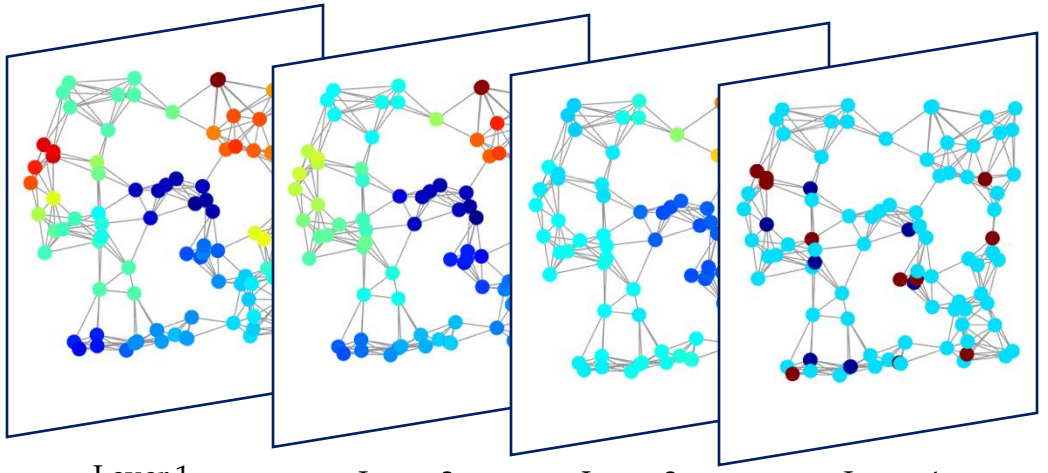
Challenge

- Message-passing based approaches are prone to ***oversmoothing***
- GNNs have **structural limitations** to apply on *large-scale* or *heterophilic* graphs!



Message-passing

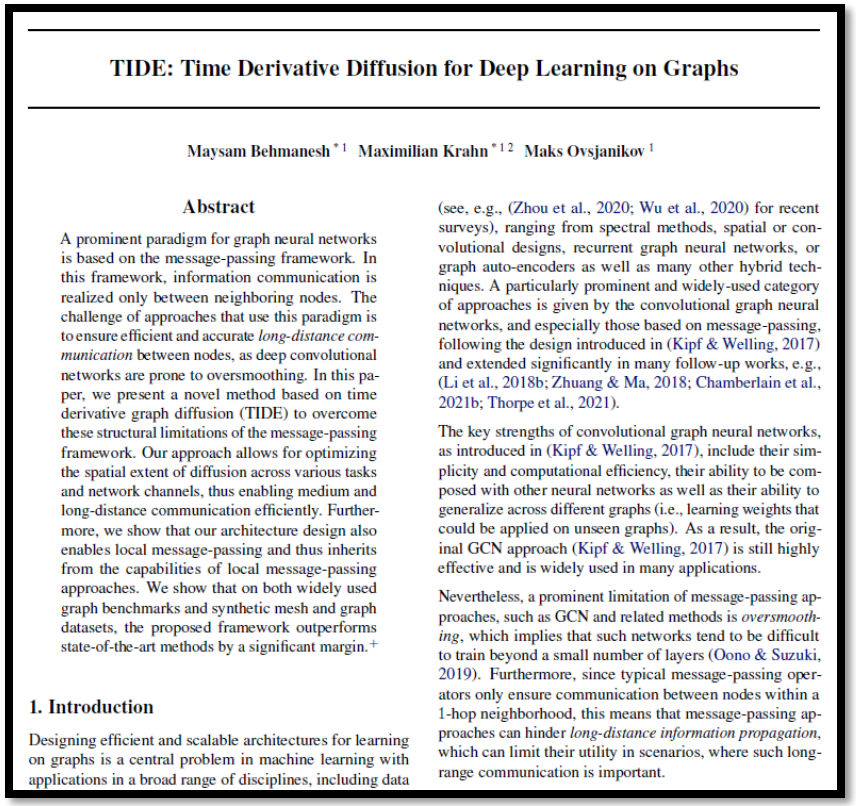
$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$



Our Goals:

- Avoid structural limitations of the message-passing frameworks
- Facilitate information propagation by using the diffusion equation
- Ensure long-distance communication between nodes

TIDE: Time Derivative Diffusion for Deep Learning on Graphs



TIDE: Time Derivative Diffusion for Deep Learning on Graphs

Maysam Behmanesh [*1] Maximilian Krahn [*1 2] Maks Ovsjanikov [1]

Abstract

A prominent paradigm for graph neural networks is based on the message-passing framework. In this framework, information communication is realized only between neighboring nodes. The challenge of approaches that use this paradigm is to ensure efficient and accurate *long-distance communication* between nodes, as deep convolutional networks are prone to oversmoothing. In this paper, we present a novel method based on time derivative graph diffusion (TIDE) to overcome these structural limitations of the message-passing framework. Our approach allows for optimizing the spatial extent of diffusion across various tasks and network channels, thus enabling medium and long-distance communication efficiently. Furthermore, we show that our architecture design also enables local message-passing and thus inherits from the capabilities of local message-passing approaches. We show that on both widely used graph benchmarks and synthetic mesh and graph datasets, the proposed framework outperforms state-of-the-art methods by a significant margin.+

1. Introduction

Designing efficient and scalable architectures for learning on graphs is a central problem in machine learning with applications in a broad range of disciplines, including data (see, e.g., (Zhou et al., 2020; Wu et al., 2020) for recent surveys), ranging from spectral methods, spatial or convolutional designs, recurrent graph neural networks, or graph auto-encoders as well as many other hybrid techniques. A particularly prominent and widely-used category of approaches is given by the convolutional graph neural networks, and especially those based on message-passing, following the design introduced in (Kipf & Welling, 2017) and extended significantly in many follow-up works, e.g., (Li et al., 2018b; Zhuang & Ma, 2018; Chamberlain et al., 2021b; Thorpe et al., 2021).

The key strengths of convolutional graph neural networks, as introduced in (Kipf & Welling, 2017), include their simplicity and computational efficiency, their ability to be composed with other neural networks as well as their ability to generalize across different graphs (i.e., learning weights that could be applied on unseen graphs). As a result, the original GCN approach (Kipf & Welling, 2017) is still highly effective and is widely used in many applications.

Nevertheless, a prominent limitation of message-passing approaches, such as GCN and related methods is *oversmoothing*, which implies that such networks tend to be difficult to train beyond a small number of layers (Oono & Suzuki, 2019). Furthermore, since typical message-passing operators only ensure communication between nodes within a 1-hop neighborhood, this means that message-passing approaches can hinder *long-distance information propagation*, which can limit their utility in scenarios, where such long-range communication is important.

Maysam Behmanesh

Maximilian Krahn

Maks Ovsjanikov

https://github.com/maysambehmanesh/TIDE

"TIDE: Time Derivative Diffusion for Deep Learning on Graphs," M. Behmanesh, M. Krahn, and M. Ovsjanikov, ICML, 2023

TIDE

Key idea: ensure information propagation using the ***diffusion equation***

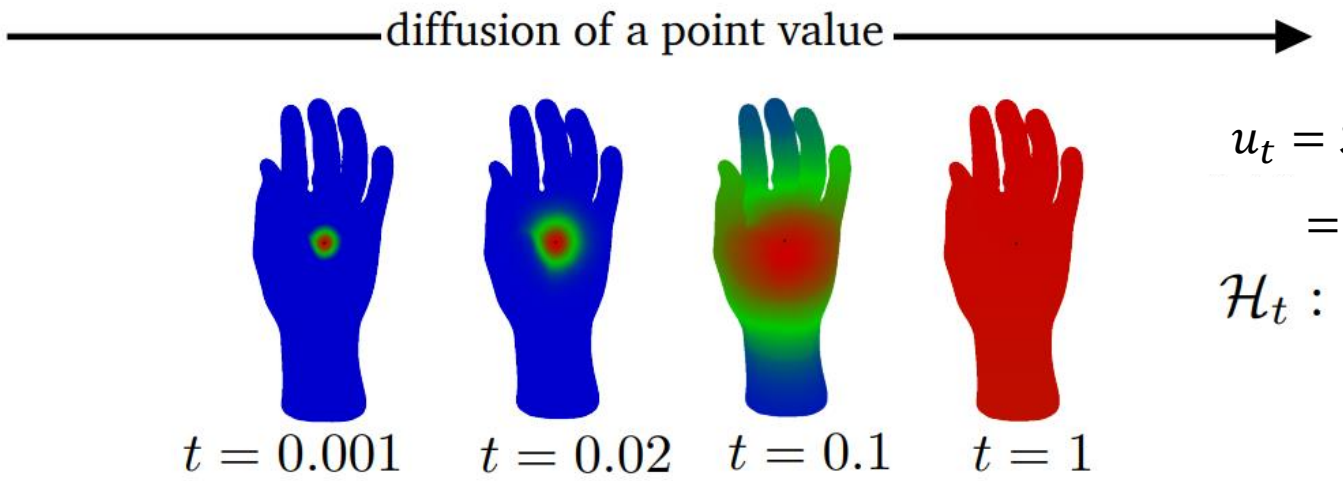
In the continuous setting, the diffusion process is described as the solution of *the heat equation*

$$\frac{\partial u}{\partial t} = -\Delta u$$

↳ Basic linear PDE

↳ Defined on graph via the Laplace-Beltrami operator Δ

↳ Implemented & well-studied on many domains



$$u_t = \mathcal{H}_t(u_0)$$
$$= \exp(-t\Delta)\, u_0$$

$\mathcal{H}_t$: Heat operator

Image from GDL-course, Ovsjanikov

TIDE

Time-derivative diffusion

Main goal:
Combine the *local accuracy* with the *global information propagation* (without oversmoothing)

We propose time-derivative diffusion as a communication mechanism:

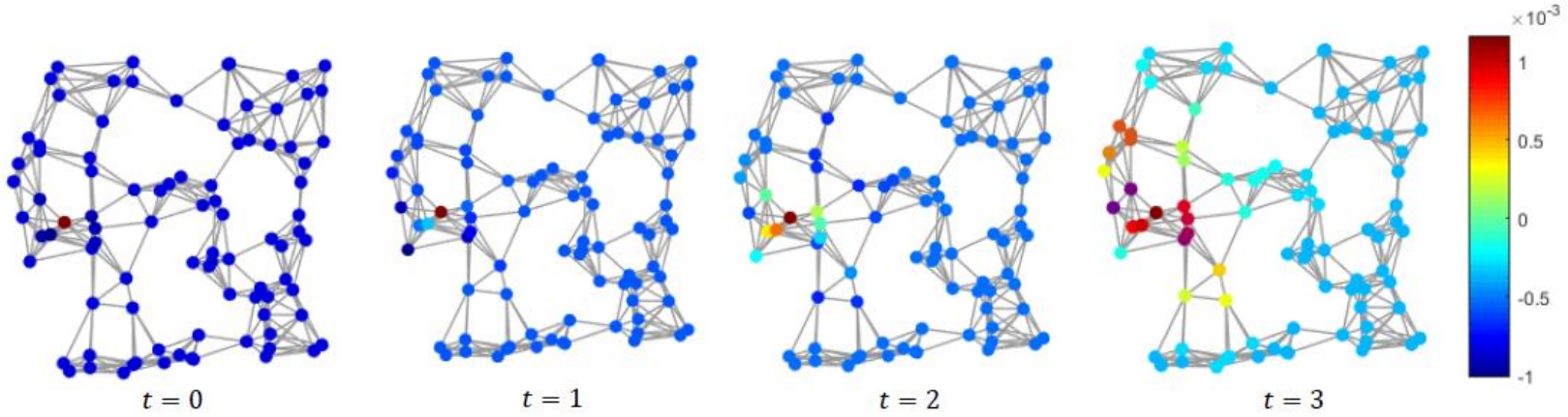
$$\frac{\partial u}{\partial t} = -\Delta u \quad \rightarrow \quad u_t = \mathcal{H}_t(u_0) = \exp(-t\mathbf{L})\, u_0 \quad \rightarrow \quad -\frac{\partial u_t}{\partial t} = \mathbf{L}u_t = \mathbf{L}\exp(-t\mathbf{L})\, u_0$$

TIDE combines local accuracy with global information propagation by:

$$\mathcal{L}_k^{TIDE}(\mathbf{U}) = \sigma\left(T_t(\mathbf{U})\mathbf{W}^{(k)}\right) = \sigma\left(\mathbf{L}\exp(-t_k\mathbf{L})\mathbf{U}\,\mathbf{W}^k\right)$$

TIDE

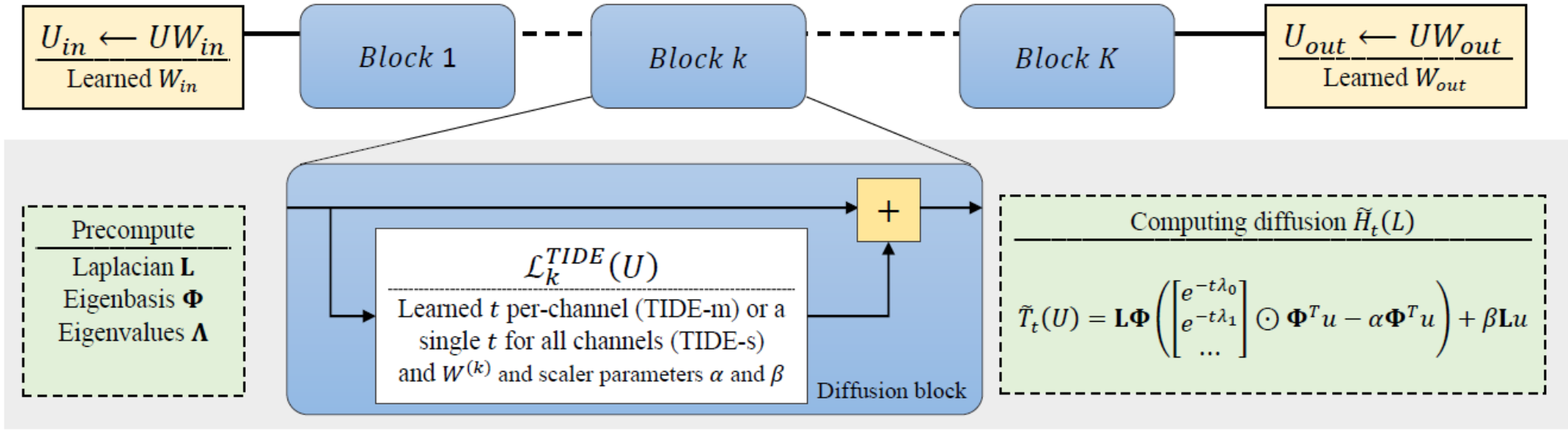
Key idea:
Using *learnable time diffusion* which allows information propagation on the graph



↳ small t, the support is local and the output is concentrated at the center node
↳ larger t, the spatial support increases, facilitating distant communication between nodes

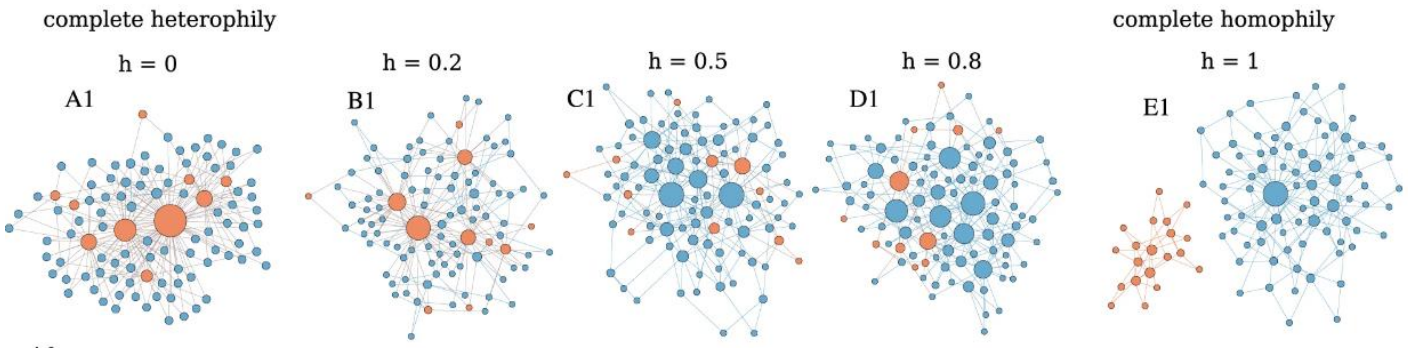
TIDE

TIDE Architecture



Two variants of TIDE:
TIDE-m: learn t per-channel
TIDE-s: learn a single t for all channels

Results:
Long Range Communication



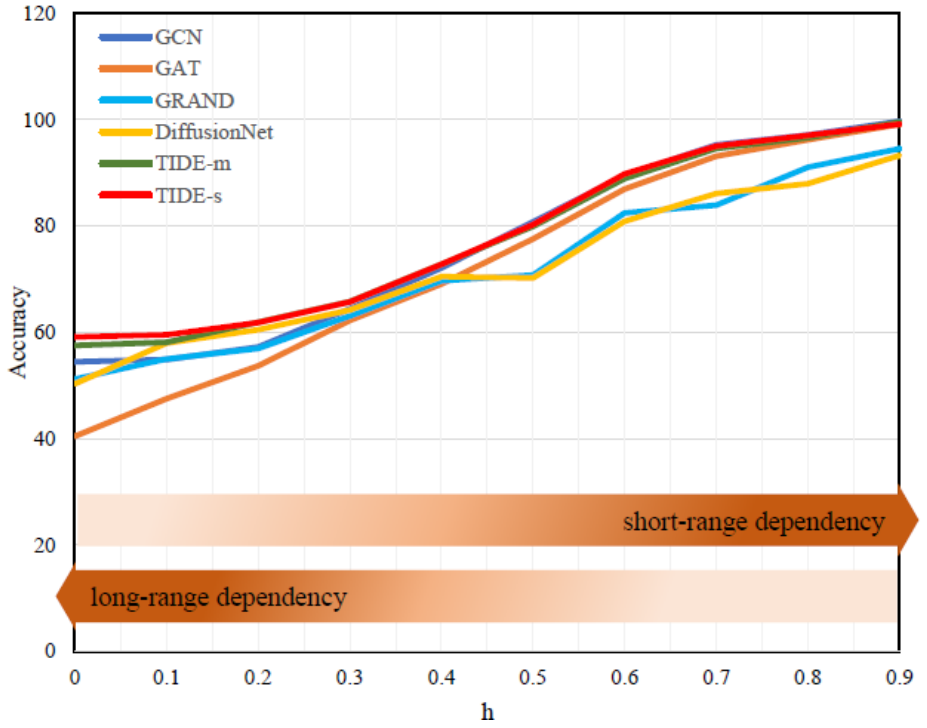


Image and dataset by: Homophily influences ranking of minorities in social networks, Fariba Karimi, Scientific Reports, 2018

TIDE

Results:
Node Classification

Scale	Dataset	#Nodes	#Edges	#Feature	#Class	#CC	h%	Avg. N.D.	Diameter
Small	Cora	2,708	5,429	1,433	7	78	80.4	4.08	19
	Citeseer	3,327	4,732	3,703	6	438	73.5	3.47	28
Medium	PubMed	19,717	44,324	500	3	1	80.2	4.5	18
	CoauthorCs	18,333	81,894	6,805	15	1	80	8.93	24
	Computers	13,381	245,778	767	10	314	77.7	36.74	10
	Photos	7,487	119,043	745	8	136	82.7	31.8	11
	ogbn-arxiv	169,343	1,166,243	128	40	1	65.4	13.67	23

#CC: Number of connected components, **h%**: Homophily rate, **Avg. N.D**: Average node degrees

Model	Cora	Citeseer	Pubmed	CoauthorCS	Computer	Photo	Ogbn-arxiv
GCN (Kipf & Welling, 2017)	83.30±0.36	68.23±0.91	76.78±0.31	90.17±0.50	81.01±0.65	91.71±0.67	65.91±0.12
GAT (Veličković et al., 2017)	81.83±0.42	69.19±0.53	75.49±0.43	90.15±0.35	80.25±0.52	91.57±0.41	54.23±0.22
GRAND (Chamberlain et al., 2021b)	80.71±0.86	68.06±0.18	74.61±0.25	90.59±0.21	72.96±0.49	84.17±0.34	59.29±0.12
GCNII (Chen et al., 2020)	79.94 ± 1.11	70.27±0.32	76.59±0.7	84.27±0.80	32.63±8.6	57.41±3.6	49.87±0.37
ACM (Luan et al., 2022)	81.83±0.12	69.03±0.02	73.3±0.63	**91.50±0.13**	77±0.65	**92.42±0.29**	66.23±0.42
DiffusionNet (Sharp et al., 2022)	80.96±0.50	70.00±0.91	73.09±0.15	89.52±0.22	74.72±0.66	87.17±0.26	54.79±0.16
TIDE-m	**84.47±0.43**	**70.32±0.68**	**77.59±0.04**	89.86±0.30	82.11±0.03	91.33±0.47	67.86±1.10
TIDE-s	84.31±0.36	70.24±0.80	77.24±0.62	90.21±0.12	**83.01±0.02**	92.06±0.51	**68.43±0.35**

TIDE

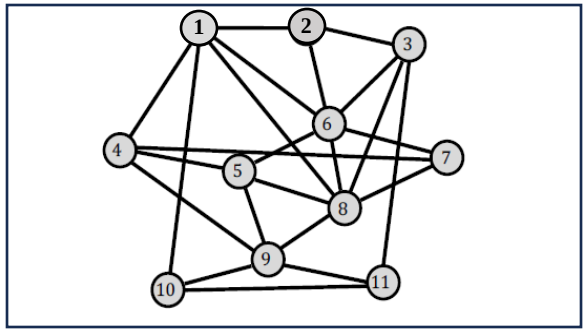
Results:
Node Classification on graphs with different homophily rates

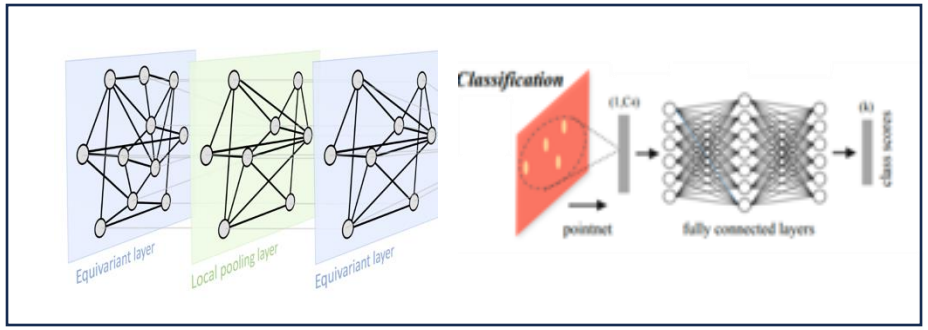
Graph	#Nodes	#Edges	#Node featues	#Class	Class type	h
Chameleon	2,277	36,101	2,325	5	Wiki pages	0.23
Actor	7,600	29,926	931	5	Actors in movies	0.22
Cornell	183	295	1,703	5	Web pages	0.3
Texas	183	309	1,703	5	Web pages	0.11
Wisconsin	251	499	1,703	5	Web pages	0.21
Genius	421,961	984,979	12	2	marked act.	0.618
Twitch-gamers	168,114	6,797,557	7	2	mature content	0.545
Snap-patents	2,923,922	13,975,788	269	5	time granted	0.073

Model	Chameleon	Actor	Cornell	Texas	Wisconsin	Genius	Twitch-gamer	Snap-patents
GCN	45.18±0.62	29.38±0.5	43.24±1.3	63.51±1.9	54.92±9.7	80.87±0.13	60.60±0.19	36.84±0.37
GAT	44.96±6.2	28.88±1.0	54.05±1.1	62.16±0.08	55.88±1.4	79.83±0.23	53.08±0.16	38.76±0.75
GRAND	50.33±0.47	35.00±0.28	55.41±1.9	**67.62±1.9**	64.86±1.3	82.47±0.08	59.85±0.03	38.89±0.42
DiffusionNet	**53.84±1.1**	34.44±0.33	56.76±0.6	62.16±0.0	62.78±2.8	82.59±0.12	55.72±1.6	30.69±0.014
TIDE-m	52.08±1.1	36.18±0.47	58.11±1.9	64.86±1.5	**69.61±1.4**	**83.03±0.06**	**60.81±0.04**	40.56±1.7
TIDE-s	51.75±0.47	**36.64±0.47**	**59.46±1.9**	63.81±1.9	68.63±1.4	83.01±0.06	60.40±0.13	**40.75±0.58**

Challenge

1. GNNs are Limited to *data-rich applications,* not useful in generic tasks!
2. Lack of *generalizable* (transfer) learning on graph

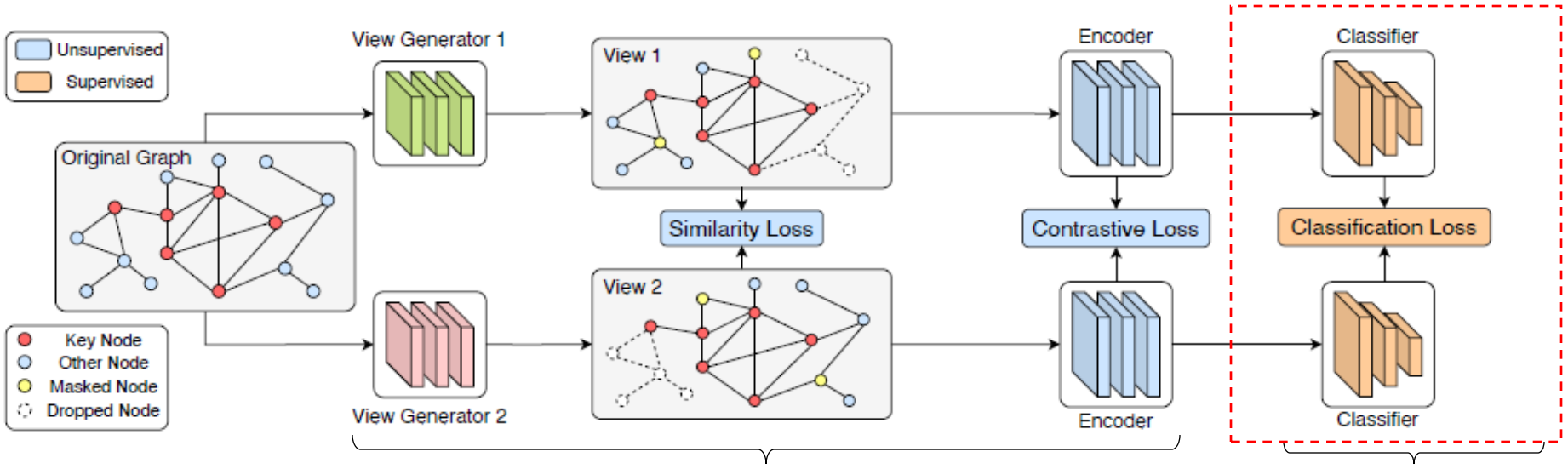




Research direction

Typical representation learning pipeline

Contrastive Learning: powerful feature learning without labels.

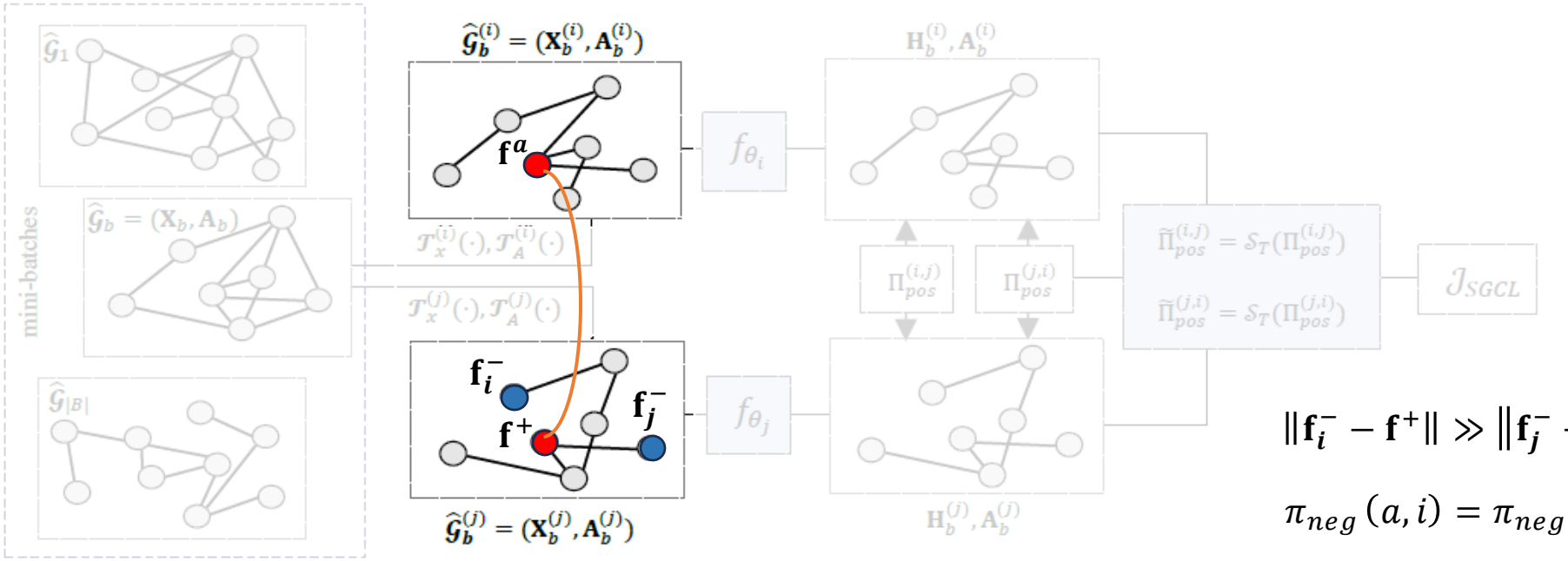


How to learn **informative features** on unlabeled graph (ideally, useful in **downstream applications**) ?

Challenge

Major challenge:

- GCL allocates negative pairs *uniformly*, regardless of their proximity to the true positive.



$$\|\mathbf{f}_i^- - \mathbf{f}^+\| \gg \|\mathbf{f}_j^- - \mathbf{f}^+\|$$

$$\pi_{neg}(a,i) = \pi_{neg}(a,j) = 1$$

We expect:

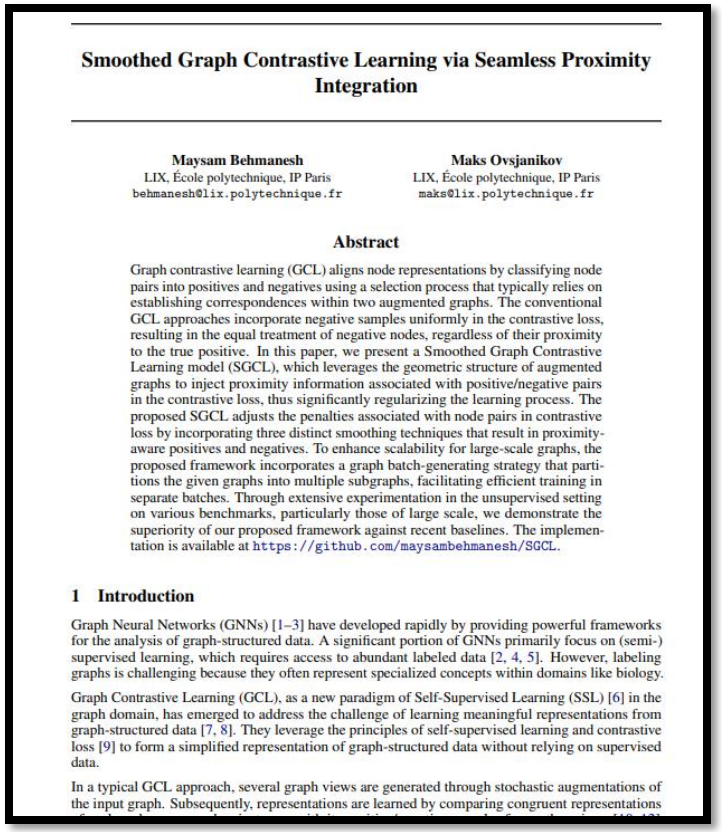
$$0 < \pi_{neg}(a,j) < \pi_{neg}(a,i) < 1$$

Our Goal:

- Integration proximity information in the contrastive loss.

SGCL

Smoothed Graph Contrastive Learning via Seamless Proximity Integration



Smoothed Graph Contrastive Learning via Seamless Proximity Integration

Maysam Behmanesh
LIX, École polytechnique, IP Paris
behmanesh@lix.polytechnique.fr

Maks Ovsjanikov
LIX, École polytechnique, IP Paris
maks@lix.polytechnique.fr

Abstract

Graph contrastive learning (GCL) aligns node representations by classifying node pairs into positives and negatives using a selection process that typically relies on establishing correspondences within two augmented graphs. The conventional GCL approaches incorporate negative samples uniformly in the contrastive loss, resulting in the equal treatment of negative nodes, regardless of their proximity to the true positive. In this paper, we present a Smoothed Graph Contrastive Learning model (SGCL), which leverages the geometric structure of augmented graphs to inject proximity information associated with positive/negative pairs in the contrastive loss, thus significantly regularizing the learning process. The proposed SGCL adjusts the penalties associated with node pairs in contrastive loss by incorporating three distinct smoothing techniques that result in proximity-aware positives and negatives. To enhance scalability for large-scale graphs, the proposed framework incorporates a graph batch-generating strategy that partitions the given graphs into multiple subgraphs, facilitating efficient training in separate batches. Through extensive experimentation in the unsupervised setting on various benchmarks, particularly those of large scale, we demonstrate the superiority of our proposed framework against recent baselines. The implementation is available at https://github.com/maysambehmanesh/SGCL.

1 Introduction

Graph Neural Networks (GNNs) [1–3] have developed rapidly by providing powerful frameworks for the analysis of graph-structured data. A significant portion of GNNs primarily focus on (semi-) supervised learning, which requires access to abundant labeled data [2, 4, 5]. However, labeling graphs is challenging because they often represent specialized concepts within domains like biology.

Graph Contrastive Learning (GCL), as a new paradigm of Self-Supervised Learning (SSL) [6] in the graph domain, has emerged to address the challenge of learning meaningful representations from graph-structured data [7, 8]. They leverage the principles of self-supervised learning and contrastive loss [9] to form a simplified representation of graph-structured data without relying on supervised data.

In a typical GCL approach, several graph views are generated through stochastic augmentations of the input graph. Subsequently, representations are learned by comparing congruent representations

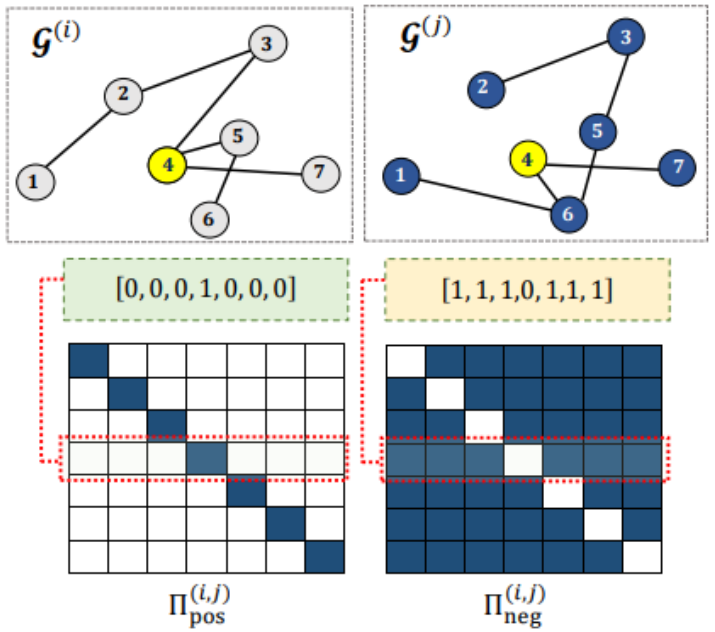
Maysam Behmanesh

Maks Ovsjanikov

https://github.com/maysambehmanesh/SGCL

"Graphs Smoothed Graph Contrastive Learning via Seamless Proximity Integration," M. Behmanesh, and M. Ovsjanikov, LoG, 2024

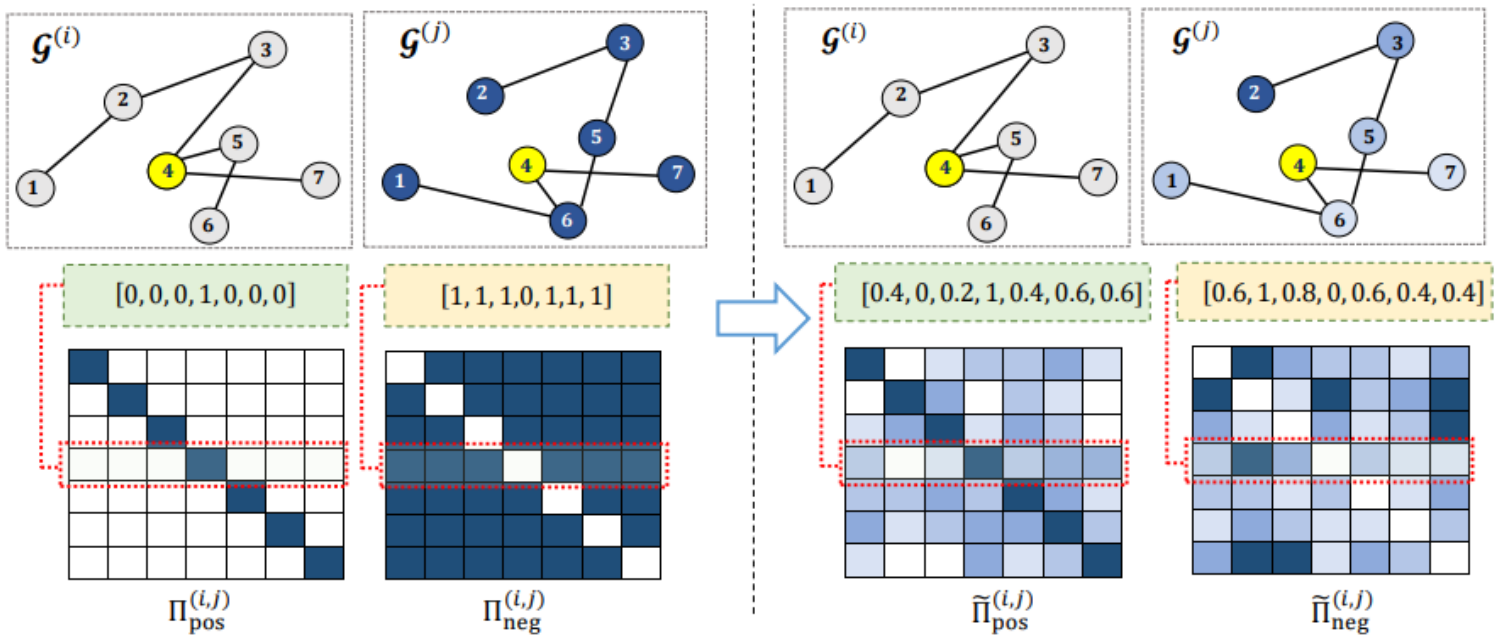
Our intuition:
Going beyond simple binary categorization of positive and negative points



$$\Pi_{pos} \in \{0,1\}^{N \times N}$$

$$\Pi_{neg} \in \{0,1\}^{N \times N}$$

Our intuition:
Going beyond simple binary categorization of positive and negative points

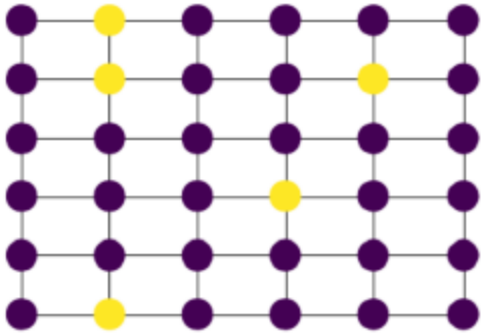


✗ $\Pi_{pos} \in \{0,1\}^{N\times N}$
$\Pi_{neg} \in \{0,1\}^{N\times N}$

✓ $\widetilde{\Pi}_{pos} \in [0,1]^{N\times N}$
$\widetilde{\Pi}_{neg} \in [0,1]^{N\times N}$

SGCL

Question:
How can proximity information be effectively incorporated into contrastive loss?



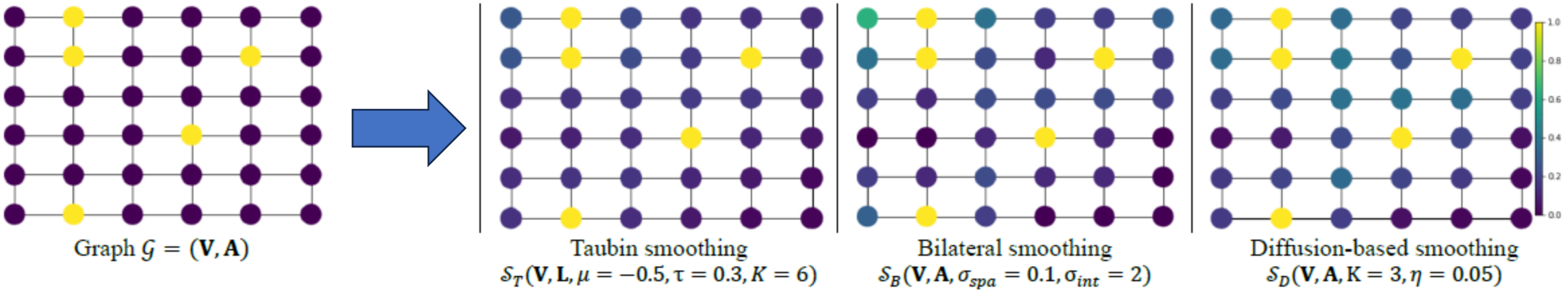
Graph $\mathcal{G} = (\mathbf{V}, \mathbf{A})$

Input:
binary matrix $\Pi \in \{0,1\}^{N \times N}$

SGCL

Our Intuition:
Applying a smoothing approach for graph

Smoothing involves iteratively updating node values based on the values of their neighboring nodes



Input:
binary matrix $\Pi \in \{0,1\}^{N \times N}$

Output:
smooth matrix $\tilde{\Pi} \in [0,1]^{N \times N}$

Smoothing approaches:

Smoothing involves iteratively updating node values based on the values of their neighboring nodes

1. Taubin smoothing

$$\mathbf{V}^{(k+1)} = (\mathbf{I} + \tau\mathbf{L})\left((\mathbf{I} + \mu\mathbf{L})\mathbf{V}^{(k)}\right), \quad \mu < 0, \tau > 0, \text{and } \mu < -\tau$$

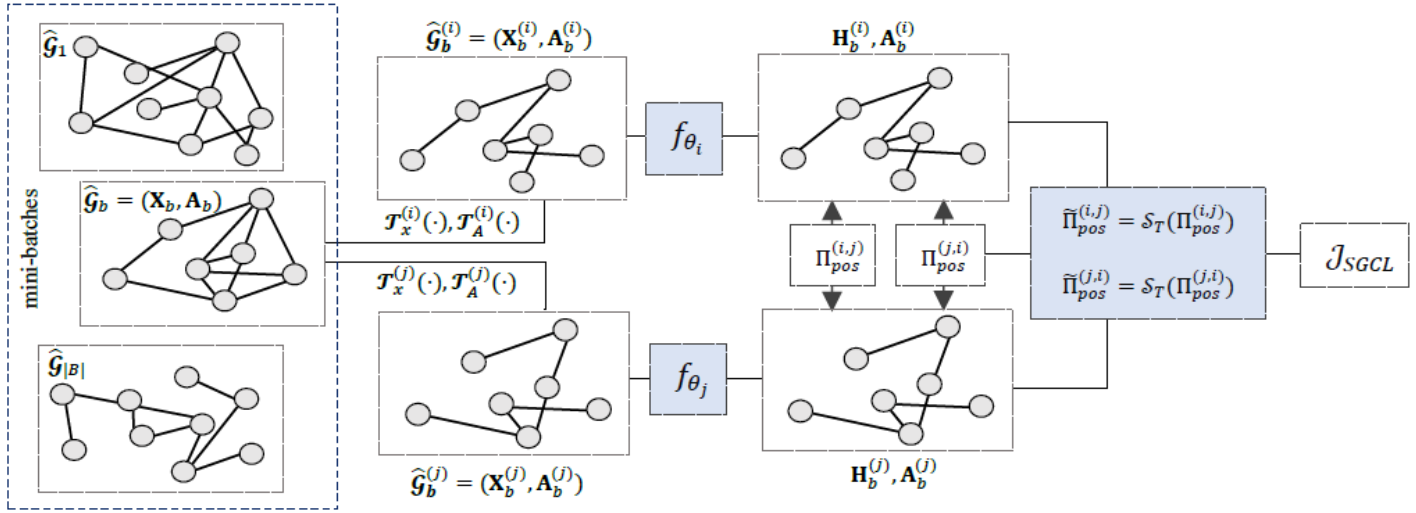
2. Bilateral smoothing

$$\tilde{\mathbf{v}}_{\boldsymbol{i}} = \frac{\sum_{j\in\mathcal{N}_k(i)} w(i,j)\mathbf{v}_j}{\sum_{j\in\mathcal{N}_k(i)} w(i,j)}, \quad \text{where} \quad w(i,j) = \exp\left(-\frac{d_{spa}(i,j)}{2\sigma_{spa}^2} - \frac{d_{int}(i,j)}{2\sigma_{int}^2}\right)$$

3. Diffusion-based smoothing

$$\mathbf{v}_i^{(k+1)} = \mathbf{v}_i^{(k)} + \eta\,\bar{\mathbf{v}}_i^{(k)}, \text{ where } \bar{\mathbf{v}}_i^{(k)} = \sum_{j\in\mathcal{N}(i)} \mathrm{v}_j$$

SGCL – **Architecture**



Contrastive Loss:

$$\mathcal{L}_{SGCL}^{(i,j)} = \| \tilde{\Pi}_{pos}^{(i,j)} \odot (\mathbf{1} - \mathbf{C}^{(i,j)}) \|_F^2 + \lambda \| \tilde{\Pi}_{neg}^{(i,j)} \odot \mathbf{C}^{(i,j)} \|_F^2$$

$\mathbf{C}^{(i,j)}$: normalized cosine similarity between the embeddings

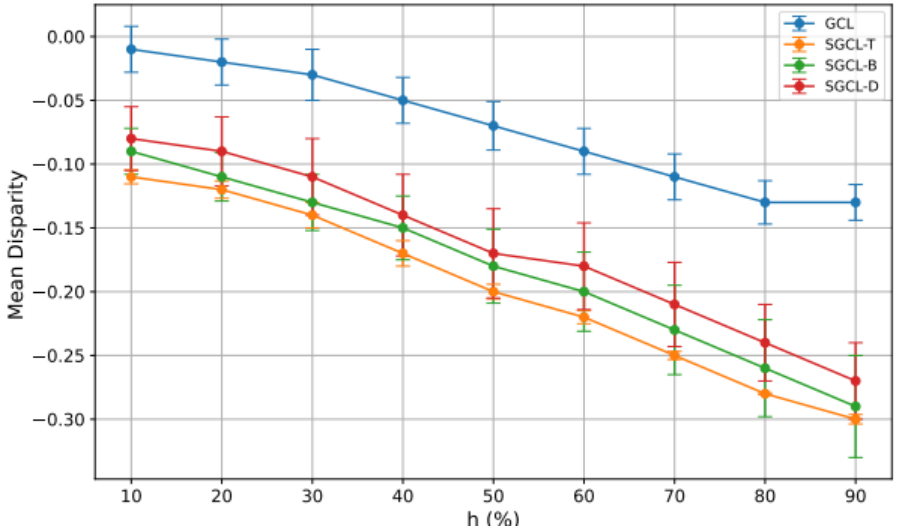
Feature space analysis

Dirichlet energy $E(\mathbf{X}) = \mathbf{X}^T\mathbf{L}\mathbf{X} = \frac{1}{2}\sum_{i,j} a_{ij}\|\mathbf{x}_i - \mathbf{x}_j\|^2$ measures the *smoothness* of the signal $\mathbf{X}$ over the graph

Disparity measure:

$$D_{disparity}(f_\theta) = \frac{1}{|E_{intra}|}\sum_{(i,j)\in E_{intra}} \Delta_{ij} - \frac{1}{|E_{inter}|}\sum_{(i,j)\in E_{inter}} \Delta_{ij}$$

where $\Delta_{ij} = \frac{1}{2}a_{ij}\|x_i - x_j\|^2$



✓ A *lower* disparity value is better

Model	Cora	Citeseer	Pubmed	CoauthorCS	Computers	Photo	ogbn-arxiv
GCL (GRACE)	0.66±0.05	0.63±0.02	0.51±0.03	0.58±0.02	0.64±0.03	0.65±0.03	0.56±0.03
SGCL-T	0.49±0.05	0.58±0.05	0.50±0.01	0.53±0.03	0.56±0.01	0.60±0.05	0.51±0.02
SGCL-B	0.63±0.03	**0.51±0.05**	0.49±0.02	0.52±0.01	**0.52±0.05**	0.59±0.02	0.48±0.05
SGCL-D	**0.47±0.02**	0.54±0.03	**0.49±0.01**	**0.46±0.05**	0.55±0.05	**0.43±0.03**	**0.45±0.04**

SGCL

Results: Node Classification

Model	Cora	Citeseer	Pubmed	CoauthorCS	Computers	Photo
DGI [10]	82.3±0.6	71.8±0.7	76.8±0.6	92.15±0.63	83.95±0.47	91.61±0.22
GRACE [11]	83.3±0.4	72.1±0.5	73.63±0.20	91.12±0.20	89.53±0.35	92.78±0.45
MVGRL [12]	83.11±0.12	73.3±0.5	84.27±0.04	92.11±0.12	87.52±0.11	91.74±0.07
BGRL [21]	83.77± 0.57	73.07±0.06	84.62±0.35	93.31±0.13	90.34±0.19	93.17±0.3
G-BT [22]	83.63±0.44	72.95±0.17	84.52±0.12	92.95±0.17	88.14±0.33	92.63±0.44
CGRA [23]	83.8±0.4	69.23±1.19	82.8±0.4	92.8±0.5	**90.5±0.4**	92.4±0.2
GRLC [24]	83.5±0.5	72.6±0.6	82.1±0.4	90.36±0.27	88.54±0.23	92.3±0.5
ProGCL-weight [25]	81.91±0.12	69.24±0.21	84.89±0.04	93.51±0.06	89.28±0.15	93.30±0.09
ProGCL-mix [25]	83.71±0.04	68.38±0.3	84.64±0.03	93.67±0.12	89.55±0.16	93.64±0.13
GraphMAE2 [26]	84.5±.0.6	73.4±0.3	81.4±0.5	–	–	–
AUGCL [27]	–	–	–	–	88.94±0.44	93.43±0.32
GREET [28]	83.81±0.87	73.08±0.84	80.29±1.00	**94.65±0.18**	87.94±0.35	92.85±0.31
SGCL-T	84.33±0.45	74.94±0.79	84.25±0.35	92.25±0.15	87.21±0.42	93.12±0.7
SGCL-B	**84.78±0.3**	74.30±1.4	84.1±0.25	92.33±0.4	89.75±0.8	**93.72±0.12**
SGCL-D	84.17±0.43	**75.72±0.59**	**85.12±0.3**	92.14±0.26	86.11±0.3	92.87±0.6

SGCL

Results: Node Classification (large scale graphs)

Scale	Dataset	#Nodes	#Edges	#Feature	#Class	#CC	h%	Avg. N.D.	Diameter
Large	ogbn-arxiv	169,343	1,166,243	128	40	1	65.4	13.67	23
	ogbn-products	2,449,029	61,859,140	100	47	52,658	80.8	51.54	27
	ogbn-proteins	132,534	39,561,252	8	94	1	91	597	9

Model	ogbn-arxiv	ogbn-products	ogbn-proteins
DGI [10]	67.07±0.5	68.68±0.6	94.11±0.1
GRACE [11]	67.92±0.4	72.10±0.7	94.11±0.2
MVGRL [12]	60.68±0.5	69.90±0.9	93.87±0.3
BGRL [21]	63.88±0.2	66.23±0.5	92.94±0.3
GBT [22]	69.05±0.3	65.74±0.4	94.07±0.3
GraphMAE2 [26]	68.95±0.4	74.32±0.5	–
SGCL-T	**70.89±0.2**	**75.97±0.1**	**94.64±0.2**
SGCL-B	70.34±0.4	74.33±0.4	93.55±0.2
SGCL-D	70.52±0.3	74.15±0.2	93.19±0.1

Results: Graph Classification

Dataset	#Graph	Avg. node	Avg. edge	#Features	#Class
MUTAG	188	17.9	39.6	7	2
PTC-MR	344	14.29	14.69	19	2
IMDB-Binary	1,000	19.8	193.1	1	2
PROTEINS	1,113	39.1	145.6	3	2
ENZYMES	600	32.63	124.3	3	6

Model	IMDB-Binary	PTC-MR	MUTAG	PROTEINS	ENZYMES
InfoGraph [29]	73.0±0.9	61.7±1.4	89.0±1.1	74.4±0.3	50.2±1.4
GraphCL [30]	71.1±0.4	63.6±1.8	86.8±1.3	74.4±0.5	55.1±1.6
MVGRL [12]	74.2±0.7	62.5±1.7	89.7±1.1	71.5±0.3	48.3±1.2
AD-GCL [31]	71.5±1.0	61.2±1.4	86.8±1.3	75.0±0.5	42.6±1.1
BGRL [21]	72.8±0.5	57.4±0.9	86.0±1.8	77.4±2.4	50.7±9.0
LaGraph [32]	73.7±0.9	60.8±1.1	90.2±1.1	75.2±0.4	40.9±1.7
ProGCL-mix [25]	71.6±0.6	–	88.7±1.4	74.5±0.4	–
CGRA [23]	75.6±0.5	**65.7±1.8**	**91.1±2.5**	76.2±0.6	61.1±0.9
AUGCL [27]	72.4±0.8	–	89.2±1.0	75.7±0.4	–
SGCL-T	75.2±2.8	64.0±1.6	89.0±2.3	79.4±1.9	**65.3±3.6**
SGCL-B	73.2±3.7	62.5±1.8	87.0±2.8	**81.6±2.3**	63.7±1.6
SGCL-D	**75.8±1.9**	62.6±1.4	86.0±2.6	81.5±2.3	64.3±2.2

Conclusions

1- Most message-passing-based GNNs are prone to *oversmoothing*

↳ TIDE facilitates information propagation using the diffusion equation
↳ TIDE ensures long-distance communication between nodes

2- Conventional GCL frameworks allocates negative pairs *uniformly*, regardless of their proximity

↳ SGCL incorporates the geometric structure of graph into a smoothed contrastive loss
↳ SGCL intuitively consider proximity information in assigning positive and negative pairs

Thank You For Your Attention

Questions?

Acknowledgements:
Part of this work are supported by the ERC Starting Grant No. 758800 (EXPROTEA) and ERC Consolidator Grant 101087347 (VEGA).