Conference on Neural Information Processing Systems

An Empirical Study of Graph Contrastive Learning

Presented by **Yanqiao ZHU**

yanqiao.zhu@cripac.ia.ac.cn

@ https://SXKDZ.github.io

Center for Research on Intelligent Perception and Computing
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences

Joint work with Yichen XU, Qiang LIU, and Shu WU

Outline

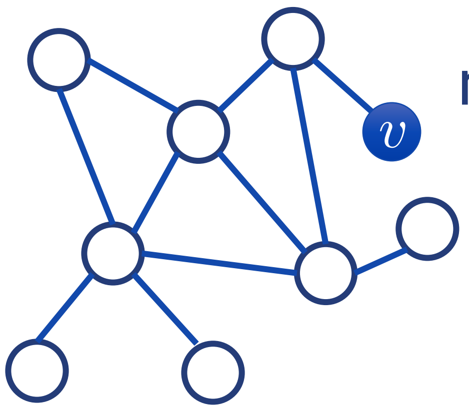
1. Background
2. A General GCL Paradigm
3. Experiments and Analysis
4. Conclusion

Outline

1. **Background**
2. A General GCL Paradigm
3. Experiments and Analysis
4. Conclusion

Representation Learning on Graphs

- Goal: efficient feature learning for machine learning on graphs
- Low-dimensional node embeddings encode both structural and attributive information.



node2vec

$$f : \mathbb{R}^F \times \{0,1\}^N \rightarrow \mathbb{R}^{F'}$$

$$\mathbb{R}^{F'} \quad F' \ll F$$

Node embeddings

Challenges for Deep Learning (on Graphs)

- Scarcity of labeled data
  - It is often expensive to obtain high-quality labels at scale in real world.
  - → GNNs overfit to small training data and fail to learn reusable, task-invariant knowledge.

- Out-of-distribution prediction
  - Test examples tend to be very different from training examples.
  - → GNNs extrapolate poorly.

[Sagawa et al., 2020] S. Sagawa et al., An Investigation of Why Overparameterization Exacerbates Spurious Correlations, in *ICML*, 2020.

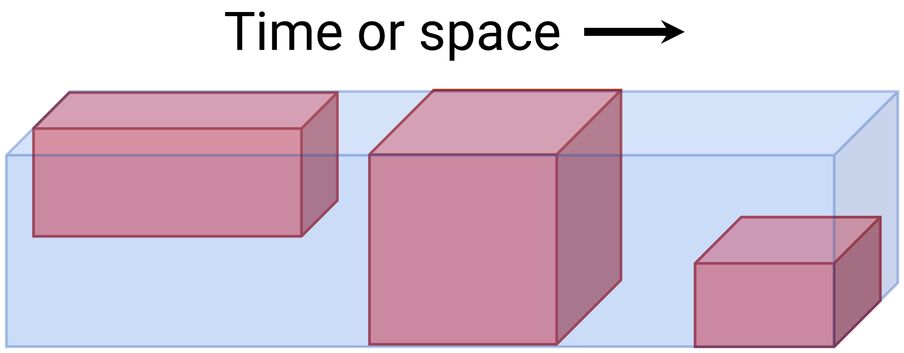
Self-supervised learning comes to rescue!

- Self-Supervised Learning (SSL) techniques have been hugely successful in computer vision and natural language processing.
  - Improve label efficiency.
  - Improve out-of-distribution performance.

> *“Labels are the opium of the machine learning researcher.”*
>
> --- Jitendra Malik
>
> *“The future is self-supervised!”*
>
> --- Yann LeCun

Self-supervised learning comes to rescue!

- Self-supervised methods obtain supervisory signals from the data itself, often leveraging the underlying structure in the data.
  - **Proxy tasks**: to capture dependencies among different dimensions of the data by predicting any part of it from any other observed part --- all without relying on labels.
- Examples:
  - Predict the future from the past.
  - Predict the masked from the visible.
  - Predict any occluded parts from all available parts.



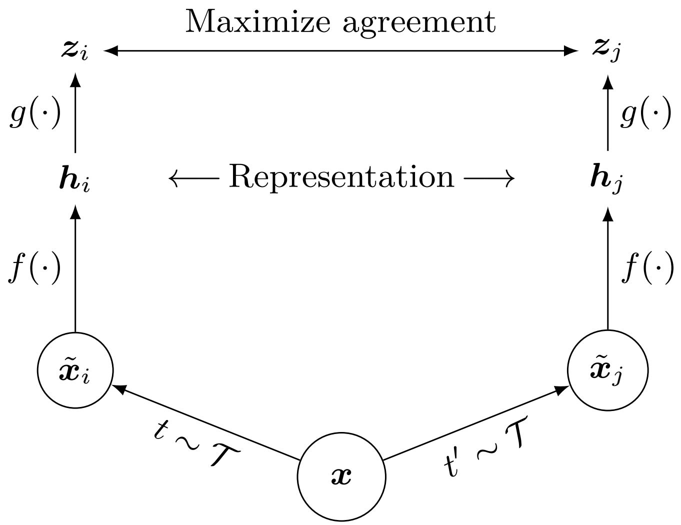
[Jing et al., 2021] L. Jing and Y. Tian, Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey, *TPAMI*, 2021.

Outline

1. Background
2. **A General GCL Paradigm**
3. Experiments and Analysis
4. Conclusion

The Contrastive Learning Paradigm

- Contrastive Learning (CL) aims to maximize the agreement of latent representations under stochastic data augmentation.

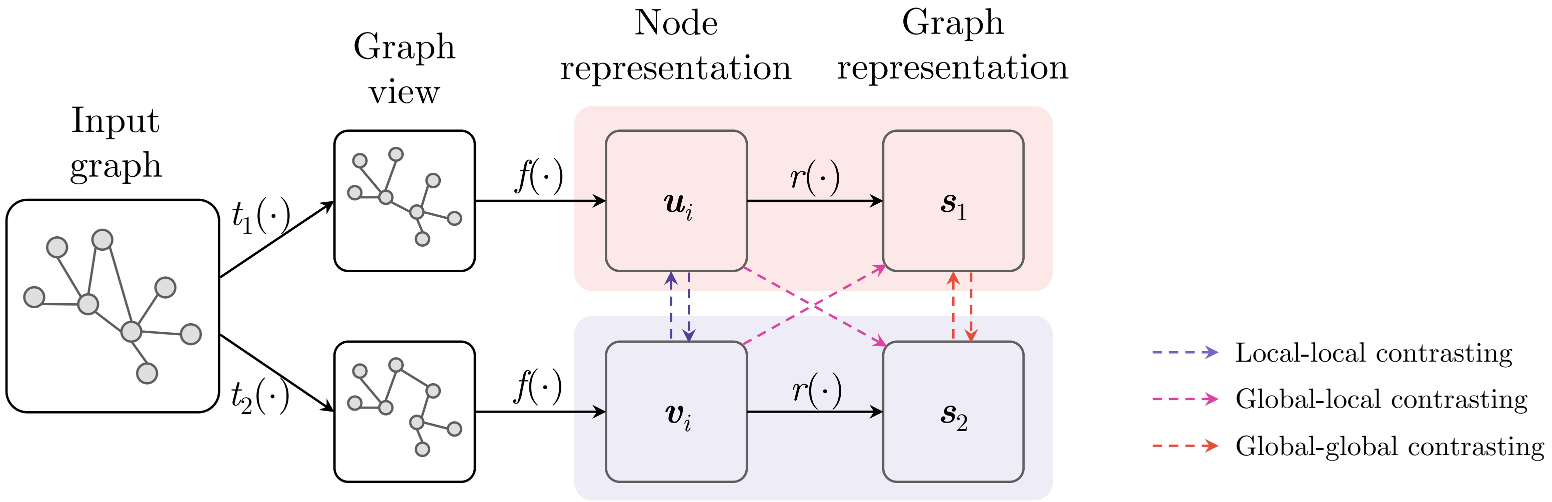


Distinguish a pair of representations from two augmentations of the same sample (positives) apart from $(n-1)$ pairs of representations from different samples (negatives).

[Chen et al., 2020] T. Chen et al., A Simple Framework for Contrastive Learning of Visual Representations, in *ICML*, 2020.

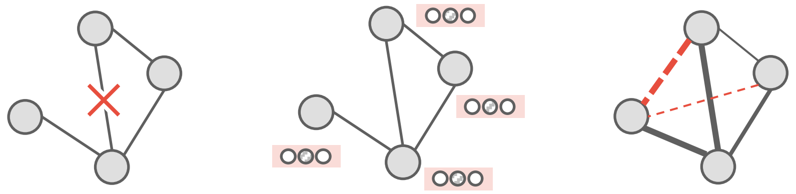
Contrastive Learning on Graphs

- Characterize Graph Contrastive Learning (GCL) models:
  - Data augmentation
  - Contrasting mode
  - Contrastive objectives
  - Negative mining strategies

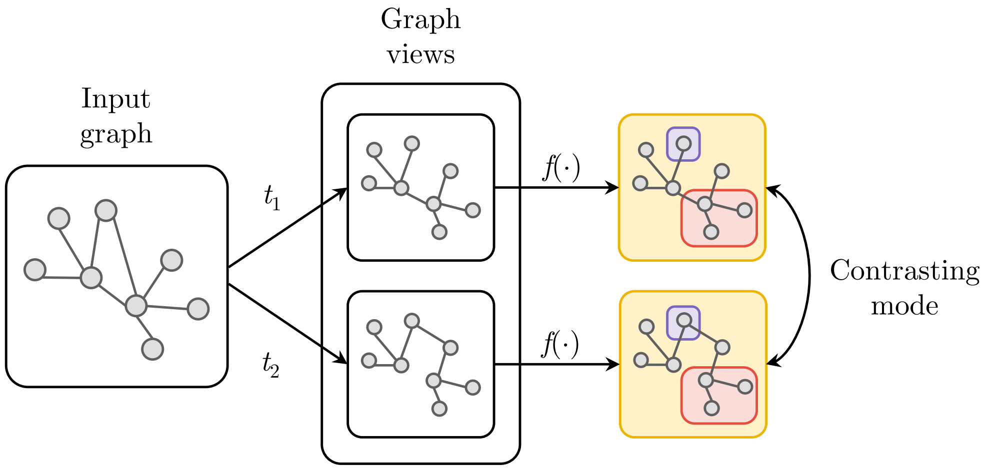


Design Dimensions

- Data augmentations: generate graph views
  - Topology augmentation
  - Feature augmentation

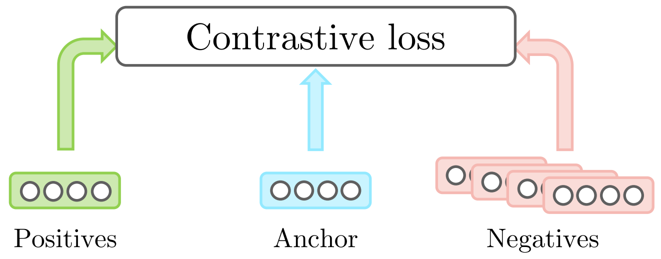


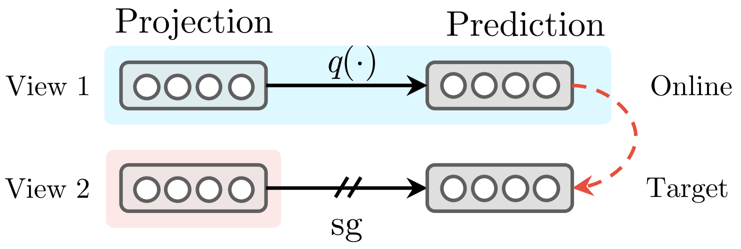
- Contrasting modes: specify positive and negative samples
  - Same-scale contrasting
    - Global-global contrast
    - Local-local contrast
  - Cross-scale contrasting
    - Global-local contrast
    - Local/global-context contrast



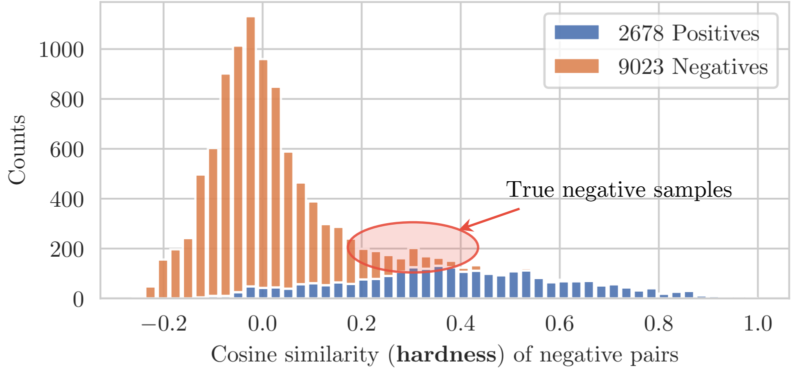
Design Dimensions (cont.)

- Contrastive objectives: score likelihood of sample pairs
  - Negative-sample-based or negative-sample-free





- Negative mining strategies
  - Debias selection of false negatives
  - Upweight hard negative samples



Outline

1. Background
2. A General GCL Paradigm
3. **Experiments and Analysis**
4. Conclusion

Experimental Configurations

- Datasets and tasks

	Dataset	Domain	#Graphs	Avg. #nodes	Avg. #edges	#Features	#Classes
Unsupervised node classification	Wiki	Knowledge base	1	11,701	216,123	300	10
	Computer	Social networks	1	13,752	245,861	767	10
	CS	Citation networks	1	18,333	81,894	6,805	15
	Physics	Citation networks	1	34,493	247,962	8,415	5
Unsupervised graph classification	NCI1	Biochemical molecules	4110	29.87	32.30	—	2
	PROTEINS	Bioinformatics	1,133	39.06	72.82	29	2
	IMDB-M	Social networks	1,500	13.00	65.94	—	3
	COLLAB	Social networks	5,000	74.49	2457.78	—	3

- Evaluation protocols
  - Unsupervised training followed by linear evaluation (logistic regression) on fixed embeddings

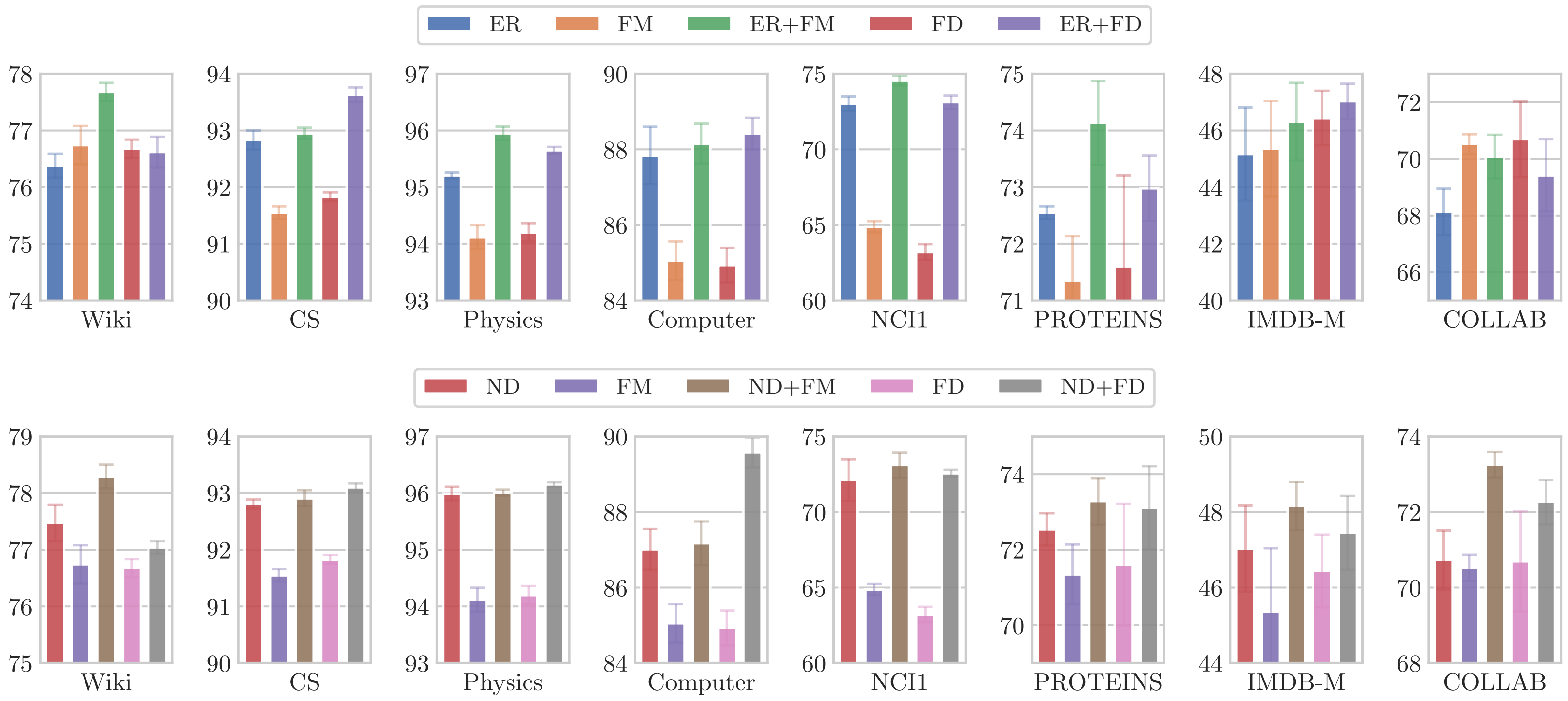
Recipes for Effective GCL (1/6)

- Topology augmentation greatly affects model performance. Augmentation functions that produce sparser graphs generally lead to better performance.

	Aug.	Node				Graph			
		Wiki	CS	Physics	Computer	NCI1	PROTEINS	IMDB-M	COLLAB
	None	68.52±0.39	90.76±0.05	93.69±0.73	80.62±0.62	58.49±2.21	70.94±1.13	45.07±1.70	66.21±0.92
Topo.	EA	72.65±0.43	92.73±0.10	94.77±0.05	83.40±0.64	70.80±0.55	71.17±0.63	44.80±1.43	68.12±0.63
	ER	76.38±0.21	92.83±0.17	95.21±0.05	**87.84±0.76**	73.03±0.48	72.55±0.11	45.17±1.64	68.13±0.82
	EF	74.10±0.67	92.99±0.15	94.88±0.06	86.68±0.73	73.95±0.49	70.64±1.67	44.15±1.21	67.92±0.93
	ND	**77.47±0.32**	92.81±0.08	**95.99±0.12**	87.01±0.54	72.12±1.38	**72.54±0.43**	**47.03±1.14**	70.73±0.78
	PPR	69.28±0.22	92.25±0.07	OOM	85.06±0.53	58.70±0.51	71.69±1.12	45.27±0.85	68.51±0.67
	MKD	69.87±0.12	92.62±0.14	OOM	82.46±0.58	57.21±0.31	71.31±0.11	45.07±1.16	68.09±0.88
	RWS	76.74±0.20	**93.48±0.08**	95.04±0.11	87.60±0.63	**75.11±1.14**	71.79±0.82	44.95±0.82	**70.85±0.89**
Feat.	FM	**76.74±0.34**	91.55±0.11	94.12±0.21	**85.05±0.51**	**64.87±0.36**	71.35±0.79	45.36±1.68	70.52±0.35
	FD	76.68±0.16	**91.83±0.08**	**94.20±0.16**	84.93±0.46	63.21±0.51	**71.60±1.61**	**46.44±0.96**	**70.69±1.33**

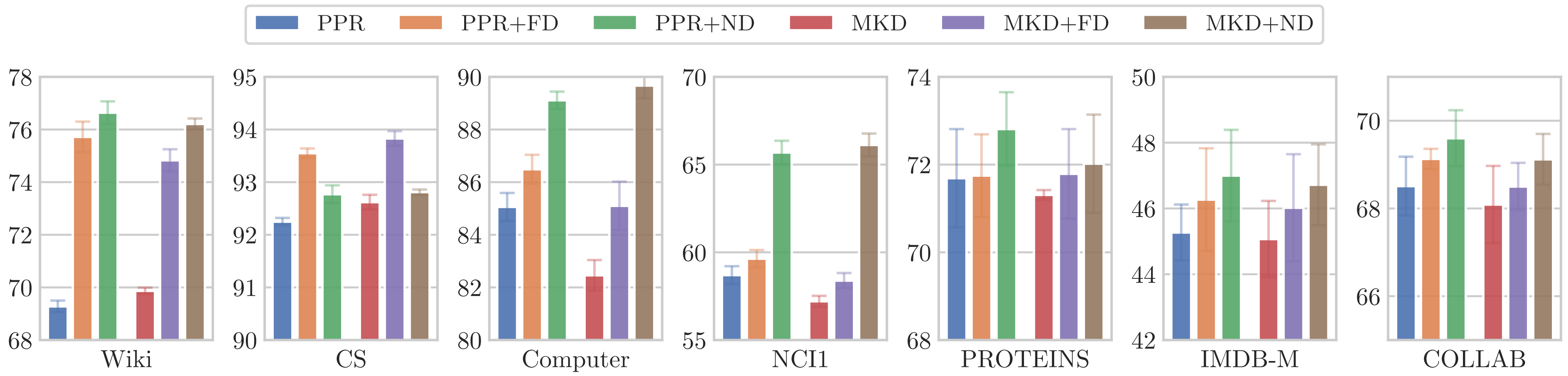
Recipes for Effective GCL (2/6)

- Feature augmentations bring additional benefits to GCL. Compositional augmentations at both structure and attribute level benefit GCL most.



Recipes for Effective GCL (3/6)

- Deterministic augmentation schemes should be accompanied by stochastic augmentations.



Recipes for Effective GCL (4/6)

- Same-scale contrasting generally performs better. Downstream tasks of different granularities favor different contrasting modes.

(a) Node classification

Obj.	Wiki		CS		Physics		Computer	
	L–L	G–L	L–L	G–L	L–L	G–L	L–L	G–L
InfoNCE	**79.09±0.15**	77.73±0.94	**92.45±0.83**	90.60±0.06	**95.95±0.92**	93.23±0.96	**88.15±0.59**	76.24±0.93
JSD	**78.83±0.95**	78.71±0.19	**92.18±1.00**	91.31±0.62	**94.32±0.28**	94.12±0.04	**82.02±0.76**	78.27±0.05
TM	**78.42±0.88**	76.53±0.85	**91.91±0.31**	90.11±0.61	**94.11±0.60**	92.78±0.12	69.67±0.88	**76.38±0.75**
BL	76.83±0.80	75.34±0.43	93.10±0.94	88.55±0.43	94.81±0.98	94.09±0.83	**87.79±0.94**	85.43±0.23
BT	80.41±0.15	—	**94.16±0.02**	—	**96.55±0.12**	—	86.86±0.97	—
VICReg	**80.79±0.12**	—	93.46±0.08	—	95.59±0.23	—	86.39±0.32	—

(b) Graph classification

Obj.	NCI1			PROTEINS			IMDB-M			COLLAB		
	L–L	G–L	G–G	L–L	G–L	G–G	L–L	G–L	G–G	L–L	G–L	G–G
InfoNCE	73.10±0.83	72.35±0.21	**73.95±0.89**	73.28±0.62	71.57±0.92	**75.73±0.09**	48.16±0.64	47.36±0.48	**49.69±0.44**	73.25±0.34	70.92±0.22	**73.72±0.12**
JSD	**73.56±0.32**	73.29±0.31	70.93±0.17	**73.88±0.31**	73.15±0.42	73.67±0.45	48.31±1.17	**48.61±1.21**	48.31±1.35	70.40±0.31	**72.62±0.35**	71.60±0.32
TM	**72.43±0.21**	71.21±0.19	72.31±0.22	72.17±0.51	72.13±1.48	**73.78±0.47**	48.38±0.20	47.75±1.24	**48.58±0.62**	68.85±0.45	69.47±0.20	**72.97±0.47**
BL	**77.22±0.13**	75.97±0.23	**76.70±0.31**	**77.75±0.43**	77.32±0.21	**78.17±0.59**	**54.64±0.43**	54.21±1.01	**55.32±0.21**	**73.95±0.25**	73.35±0.24	**74.92±0.33**
BT	72.49±0.22	—	70.53±1.11	74.87±0.68	—	74.38±0.56	48.50±0.65	—	49.53±0.42	71.70±0.53	—	73.00±0.42
VICReg	72.31±0.34	—	71.60±0.36	74.61±1.15	—	74.38±0.57	46.75±1.47	—	50.28±0.55	68.88±0.34	—	72.50±0.31

Recipes for Effective GCL (5/6)

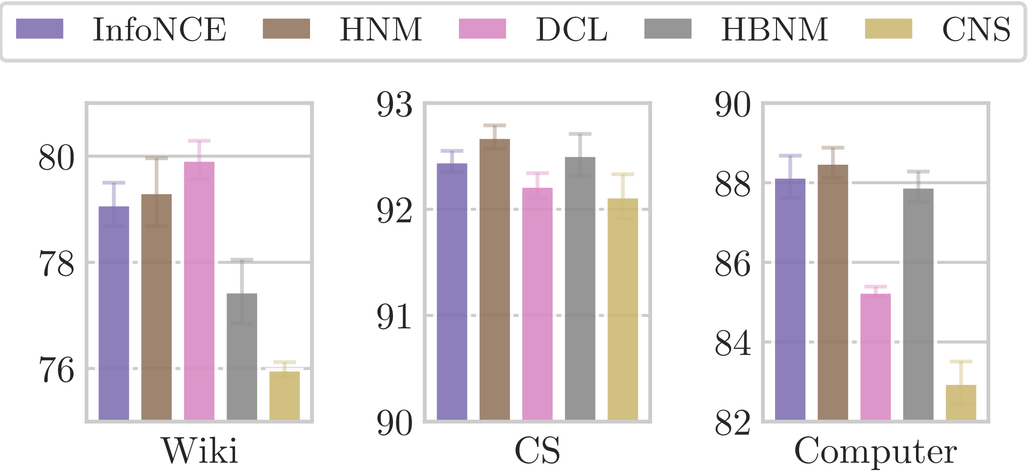
- Among negative-sample-based objectives, the use of InfoNCE objective leads to consistent improvements across all settings.
- Bootstrapping Latent and Barlow Twins losses obtain promising performance on par with their negative-sample-based counterparts yet reduce the computational burden without explicit negative samples.

Obj.	L–L	G–L	G–G
InfoNCE	6,311	2,977	2,271
JSD	6,309	2,845	2,269
TM	6,271	2,977	2,269
BL	2,235	2,247	2,187
BT	2,419	—	2,201
VICReg	2,465	—	2,232

Memory usage (MB) on the PROTEINS dataset

Recipes for Effective GCL (6/6)

- Existing negative mining techniques based on calculating embedding similarities bring limited benefit to GCL.
- Dilemma: the harder a negative sample is, the more likely it is a positive sample.



Outline

1. Background
2. A General GCL Paradigm
3. Experiments and Analysis
4. **Conclusion**

Concluding Remarks

- In this work, we analyze design choices for each GCL component.
- We conduct extensive empirical studies over a comprehensive set of benchmarking tasks and datasets.
- Our rigorous empirical study reveal several interesting findings of GCL that may be helpful for developing future algorithms.

Future Directions

- Towards automated data augmentation
- Principled understanding of the performance gap between pretext and downstream tasks
- Structure-aware negative sampling
- Objectives for large-scale graph datasets
- GCL for complex graphs

Open-Source Library: 🌝🌚 PyGCL

- PyGCL features modularized GCL components from published papers, standardized evaluation, and experiment management.
- Implement GRACE with few lines of code

```python
aug1 = A.Compose([A.EdgeRemoving(pe=0.3), A.FeatureMasking(pf=0.3)])
aug2 = A.Compose([A.EdgeRemoving(pe=0.3), A.FeatureMasking(pf=0.3)])

gconv = GConv(input_dim=dataset.num_features, hidden_dim=32, activation=torch.nn.ReLU, num_layers=2)
encoder_model = Encoder(encoder=gconv, augmentor=(aug1, aug2), hidden_dim=32, proj_dim=32)
contrast_model = DualBranchContrast(loss=L.InfoNCE(tau=0.2), mode='L2L', intraview_negs=True)

z, z1, z2 = encoder_model(data.x, data.edge_index, data.edge_attr)
h1, h2 = [encoder_model.project(x) for x in [z1, z2]]
loss = contrast_model(h1, h2)
loss.backward()
optimizer.step()
```

THANKS

Code Library

Paper

Slides