Self-Supervised GNN that Jointly Learns to Augment

Zekarias T. Kefato
KTH Royal Institute of Technology
Stockholm, Sweden
zekarias@kth.se

Sarunas Girdzijauskas
KTH Royal Institute of Technology
Stockholm, Sweden
sarunasg@kth.se

Hannes Stärk
Technical University of Munich
Munich, Germany
hannes.staerk@tum.de

Abstract

Self-supervised Learning (SSL) aims at learning representations of objects without relying on manual labeling. Recently, a number of SSL methods for graph representation learning have achieved performance comparable to SOTA semi-supervised Graph Neural Networks (GNNs). One of the key challenges is data-augmentation, for which existing methods rely on heuristically crafted techniques. In this study, we propose a novel method for jointly learning both the augmentation and representation by leveraging the inherent signal encoded in the graph. Besides, to allow efficient use of resources we propose a new architecture that augments in the latent space as opposed to the input space. We carried out experiments using 14 publicly available datasets on three node classification tasks. The results show that our method achieves comparable performance with semi-supervised GNNs, and it is on par with SSL methods for GNNs. Source code is available at https://github.com/zekarias-tilahun/graph-surgeon

1 Introduction

In SSL, we seek to learn representations of objects (e.g. images and graphs) without relying on manual labeling, of particular interest in this study is SSL for GNN (SSL-GNN). Our emphasis is on methods based on the Siamese architecture [4, 5, 15, 12, 7, 9, 6, 24, 13, 26, 23, 11, 22]. Self-supervised techniques using Siamese networks learn representations that benefit from data-augmentation (perturbation) [24], and devising suitable augmentation techniques is one of the main challenges. Since there are no standard augmentation techniques for graphs, existing studies rely on different heuristics and/or trial and error [22].

A Siamese architecture uses two (left and right) networks, and two differently augmented views of the same object are fed to these networks. Learning is carried out by maximizing the agreement between the outputs of these networks. Obviously, this leads to a trivial solution, and hence one has to devise a prevention strategy. To this end, one can use contrastive terms; however, since drawing truly negative samples is difficult recent studies propose asymmetry as a solution. Nonetheless, asymmetric methods rely on careful engineering tricks.

In this study, we propose a novel SSL-GNN architecture called SURGEON (**s**elf-s**u**per**r**vised **G**NN that jointly l**e**arns t**o** augme**n**t). Unlike existing methods that rely on heuristics and/or trial and error, we design SURGEON in such a way that data augmentation is jointly learned with the graph representation. Furthermore, SURGEON neither uses explicit negative samples nor employs engineering tricks. Instead, we use a principled constrained optimization objective that draws

NeurIPS 2021Workshop on Self-Supervised Learning: Theory and Practice.

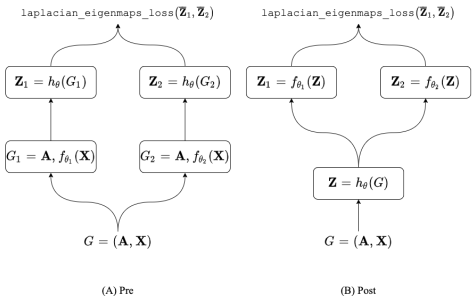

Figure 1: The architecture of SURGEON, (A) first generates two augmented views of an input signal from two learnable augmenter functions f_{θ_1} and f_{θ_2} and then passes them to a shared GNN encoder h_θ. Meanwhile, (B) first encodes or maps the input signal to a latent space and passes the latent representation through the two learnable augmenters, f_{θ_1} and f_{θ_2}.

inspiration from Laplacian Eigenmaps [2] to prevent a trivial solution. In addition, to improve the use of resources, we propose a new architecture that does augmentation in the latent space as opposed to the input feature space.

We have carried out extensive experiments using 14 publicly available datasets on the node classification task. We compare SURGEON against strong SOTA semi-supervised and self-supervised GNNs. The results show that SURGEON is comparable to semi-supervised methods, on average just 2 percentage points away from the best performing ones. However, it is on par with the SSL-GNN baselines.

2 SURGEON

We consider an undirected graph $G = (\boldsymbol{A}, \boldsymbol{X})$, where $\boldsymbol{A} \in [0, 1]^{N \times N}$ is the adjacency matrix and $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ is a feature matrix, where each row i in $\boldsymbol{X}$ encodes an attribute signal $\boldsymbol{x}_i \in \mathbb{R}^F$ for node i. Since SURGEON is agnostic to the type of GNN architecture, we simply assume that we have access to a GNN encoder $\boldsymbol{Z} = h_\theta(\boldsymbol{A}, \boldsymbol{X})$ parameterized by θ. The architecture of our model is shown in Fig. 1

2.1 Learning Data Augmentation

Given a graph G, in existing methods two augmented views $G_1 = t_1(G)$ and $G_2 = t_2(G)$ are generated by applying augmentation techniques $t_1 \sim \mathcal{T}$ and $t_2 \sim \mathcal{T}$, sampled from a set of predefined techniques, $\mathcal{T}$. However, these techniques perform well on some datasets and poorly on some others, and it is not well understood why this is the case yet. One remedy is to automatically learn the relevant augmentation from the graph signal, and this has not been explored yet.

In this study, we propose a simple yet novel data augmentation technique that is learned based on the signal encoded in the graph. Therefore, we replace t_1 and t_2, with trainable functions f_{θ_1} and f_{θ_2} parametrized by θ_1 and θ_2. We model f as an MLP, *i.e.*, two augmented views are generated as $\boldsymbol{X}_1 = f_{\theta_1}(\boldsymbol{X})$, $\boldsymbol{X}_2 = f_{\theta_2}(\boldsymbol{X})$, where $\boldsymbol{X}_1 \in \mathbb{R}^{N \times D}$ and $\boldsymbol{X}_2 \in \mathbb{R}^{N \times D}$ and $\theta_1 = \{\boldsymbol{W}_1^{(l)} : l = 1, \dots L_1\}$ and $\theta_2 = \{\boldsymbol{W}_2^{(l)} : l = 1, \dots L_2\}$, where L_1 and L_2 are the number of layers of f_{θ_1} and f_{θ_2}. In order for the graph signal to govern the learned augmentations $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, the key design choice in SURGEON is to jointly learn θ_1 and θ_2 with θ.

Augmentations are commonly performed in the input space, as in Fig. 1 (A), which we refer to as the *Pre* architecture. In order to improve resource usage, we propose a new architecture as shown in Fig. 1 (B) that does augmentation in the latent space, and we refer to it as the *Post* architecture.

2.2 Joint Training of the Parameters

The following discussion assumes the *pre* architecture. In the forward pass, SURGEON produces two representations $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ as $\boldsymbol{Z}_1 = h_\theta(\boldsymbol{A}, f_{\theta_1}(\boldsymbol{X}))$ and $\boldsymbol{Z}_2 = h_\theta(A, f_{\theta_2}(\boldsymbol{X}))$. Our goal in a SSL-GNN framework is to maximize the agreement between these two representations. To this end, we closely follow Laplacian Eigenmaps [2] and minimize the mean squared error between the normalized representations (unit vectors) of two data points. Though in Laplacian Eigenmaps the two data points correspond to different objects (e.g., two different nodes, images), in our case, these are just the unit vectors $\bar{\boldsymbol{Z}}_1$ and $\bar{\boldsymbol{Z}}_2$ obtained from $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ respectively. Hence our objective is:

$$\mathcal{L}_\theta = ||\bar{\boldsymbol{Z}}_1 - \bar{\boldsymbol{Z}}_2||_F^2 \tag{1}$$

However, Eq. 1 admits a trivial solution, that is, collapse into a single point or a subspace [2]. For this reason, we modify E.q. 1 and incorporate an orthonormality constraint inspired by the Laplacian Eigenmaps. Moreover, we want to jointly optimize the parameters of the augmenters. To achieve these goals, first we add the constraints $\bar{\boldsymbol{Z}}_1 \bar{\boldsymbol{Z}}_1{}^T = \boldsymbol{I}_N$ and $\bar{\boldsymbol{Z}}_2 \bar{\boldsymbol{Z}}_2{}^T = \boldsymbol{I}_N$ and second we use the gradients to update all the parameters, θ, θ_1 and θ_2. We can relax the constraints using the Lagrangian and obtain a regularized objective, which we call Laplacian Eigenmaps loss as

$$\mathcal{L}_{\theta,\theta_1,\theta_2} = ||\bar{\boldsymbol{Z}}_1 - \bar{\boldsymbol{Z}}_2||_F^2 + \gamma\big(||\bar{\boldsymbol{Z}}_1 \bar{\boldsymbol{Z}}_1{}^T - \boldsymbol{I}_N||_F + ||\bar{\boldsymbol{Z}}_2 \bar{\boldsymbol{Z}}_2{}^T - \boldsymbol{I}_N||_F\big) \tag{2}$$

Eq. 2 encourages positive pairs across $\bar{\boldsymbol{Z}}_1$ and $\bar{\boldsymbol{Z}}_2$ to be similar to each other, and the orthonormality constraint ensures that each row in $\bar{\boldsymbol{Z}}_1$ or $\bar{\boldsymbol{Z}}_2$ is similar to itself and orthonormal to other rows. Consequently, a trivial solution is avoided [2].

Eq. 2 can be improved [1] by replacing the $||\bar{\boldsymbol{Z}}_1 \bar{\boldsymbol{Z}}_1{}^T - \boldsymbol{I}_N||_F$ and $||\bar{\boldsymbol{Z}}_2 \bar{\boldsymbol{Z}}_2{}^T - \boldsymbol{I}_N||_F$ terms, which require an $N \times N$ identity matrix, with $||\bar{\boldsymbol{Z}}_1{}^T \bar{\boldsymbol{Z}}_1 - \boldsymbol{I}_{F_L}||_F$ and $||\bar{\boldsymbol{Z}}_2{}^T \bar{\boldsymbol{Z}}_2 - \boldsymbol{I}_{F_L}||_F$, where the final representation dimension $F_L \ll N$.

3 Experiments

We validate the practical use of SURGEON using 14 publicly available datasets, ranging from small to large-scale graphs. A detailed description is available in Appendix A.1. We compare SURGEON against 11 state-of-the-art baselines grouped into two

- *Semi-Supervised*: Six of the baselines are methods that use a fraction of the node labels during training, three of which (GCN [14], GAT [20], GRAPHSAGE [10]) are used for small and medium-size graphs, and the rest (CLUSTERGCN [8], GRAPHSAINT [25], and PPRGO [3]) for large-scale graphs.
- *Self-supervised*: There are five methods under this group, three of which (DGI [21], MVGRL [11], and GCA [26]) use a contrastive architecture to prevent a trivial solution and the other two SELFGNN [13] and BGRL [19] use asymmetry. Because these two techniques extend the same method, BYOL [9], for visual representation to graph representation and use the same code base, we present them as one.

3.1 Experimental Protocol

For all the datasets we have three splits, training, validation and test. For some of them, we use the splits provided by PyTorch Geometric, and for the rest, we randomly split them into 5% training, 15% validation and 80% test sets. We tune the hyperparameters of all the algorithms using Bayesian optimization, however for a fair comparison, we fix the representation dimension to 128. In addition, we run all the models for 500 epochs and take the epoch with the best validation score.

We train the semi-supervised methods using the training split and tune their hyperparameters using the validation set. Finally, we use the test set to infer the labels and report their performance. For the self-supervised methods, we train them without any label and tune them using the validation set. Following standard practice, they are evaluated under the linear protocol. This means that we freeze the models and add a logistic regression (linear) classifier on top. The classifier is trained using only the training split for 100 and 500 epochs, for small and large datasets, respectively. Similar to the semi-supervised setting, we use the test set to simply predict the labels and report prediction quality.

We have three types of node classification tasks, which are binary, multi-class, and multi-label classifications. Similar to existing studies, for the binary and multi-class tasks, we use accuracy, and for the multi-label, the Area Under the Receiver Operating Characteristic Curve (ROC-AUC). Unless a different setting is stated, we assume the aforementioned protocol.

3.2 Results

The node classification results are reported in Tables 1 and 2. Overall, SURGEON is comparable to the semi-supervised baselines and on par (sometimes marginally better and at times marginally lower than) the self-supervised baselines.

Datasets	Algorithms							
	Semi-Supervised			Contrastive			Asymmetric	SURGEON
	GCN	GAT	GRAPHSAGE	DGI	MVGRL	GCA	SELFGNN/ BGRL	
Cora	**60.1±.001**	58.27±.003	57.45±.003	50.66±.001	39.42±.193	37.64±.014	54.61±.135	**56.33±.07**
DBLP	82.7±.002	**82.88±.002**	81.39±.005	78.87±.002	69.2±.052	81.16±.007	81.32±.071	**81.48±.09**
PubMed	**85.62±.001**	84.98±.002	84.73±.001	84.28±.001	77.99±.315	82.76±.005	84.6±.076	**84.94±.091**
Physics	**95.4±.001**	95.02±.002	†	94.92±.001	91.18±.024	†	**95.11±.07**	**95.11±0.025**
CS	**91.87±.001**	91.07±.002	91.44±.001	91.72±.001	87.18±.095	88.01±.005	**92.23±.01**	92.03±.0
Computers	**88.54±.003**	88.3±.006	87.93±.004	80.28±.004	78.57±.14	74.04±.005	**86.23±.139**	85.16±.133
Photo	93.02±.003	93.18±002	**93.64±.002**	92.36±.06	86.04±.12	84.93±.009	**92.87±.08**	92.27±.05
Actor	28.38±.008	28.62±.01	**33.88±.007**	29.93±.007	**63.3±.03**	27.39±.01	29.41±1.46	30.19±.34
WikiCS	76.87±.006	77.38±.005	**77.41±.006**	70.01±.007	61.7±.52	75.25±.006	75.34±.528	**75.59±.11**
Facebook	**89.5±.002**	89.3±.01	89.25±.002	82.42±.001	78.88±0.045	86.29±0.004	**86.38±.084**	84.92±0.015
Flickr	51.66±.001	42.35±.001	**52.11±.001**	45.94±.001	†	†	**51.26±.528**	50.91±.054
Github	**86.14±.001**	86.16±.001	85.77±.001	83.84±.001	83.93±0.032	†	85.58±.053	**85.7±.028**

Table 1: The classification accuracy results along with the standard deviation. The bold highlight indicates the best performing algorithms from both the semi-supervised and self-supervised methods. † indicates that the algorithm has crashed because of an out-of-memory error.

Algorithms	Datasets	
	Yelp	Reddit
CLUSTERGCN (semi)	78.21	95.33
GRAPHSAINT (semi)	75.62	95.73
PPRGO (semi)	77.7	91.8
SURGEON	77.44	91.22

Table 2: The prediction quality for the large-scale datasets. ROC-AUC for Yelp and Accuracy for Reddit. For this experiment, we use semi-supervised and scalable GNN architectures as the full-batch ones do not fit in GPU memory. In addition, all the SSL-GNN baselines throw an out-of-memory error.

As expected, the semi-supervised models are consistently better than the self-supervised ones, except for Actor. MVGRL gives the best result, with more than 90% improvement over the best performing method. This comes as a result of using higher-order augmentation that happens to be beneficial for the Actor dataset. A similar performance is not observed for MVGRL on the other datasets. This provides a motivation for automatically learning high-order topology signals that benefit some datasets. As stated earlier, this will be covered in future work.

However, our finding showcases that just using the learned attribute augmentations and without requiring explicit negative samples, one can achieve a performance consistently close to semi-supervised models across a number of datasets and classification tasks. On average, our model is at most 2 percentage points away from the best performing semi-supervised method. Moreover, it scales to large networks with hundreds of millions of edges, where the other SSL-GNN methods failed to handle. Ablation studies and implementation details are available in the appendix.

4 Conclusion

In this paper, we propose a self-supervised graph representation learning method called SURGEON based on the Siamese network. Unlike prior methods that rely on heuristics for data augmentation, our method jointly learns the data augmentation with the representation (embedding) guided by the signal encoded in the graph. By capitalizing on the flexibility of the learnable augmentations, we propose an alternative new strategy for augmentation, called post-augmentation, which happens after an encoding. This is in contrast to the standard pre-augmentation strategy that happens before the encoding. We also show that the alternative strategy significantly improves the scalability and efficiency of our model.

Furthermore, the method does not require explicit contrastive terms or negative sampling. However, contrary to related studies with no contrastive terms, we employ a scalable principled constrained optimization inspired by Laplacian Eigenmaps, as opposed to engineering tricks to prevent collapse.

We perform an extensive empirical evaluation of the proposed method using 14 publicly available datasets on three types of node classification tasks. Besides, we compare the method with strong SOTA baselines, six semi-supervised GNNs, and five self-supervised GNNs. Our finding shows that SURGEON is comparable to the semi-supervised GNNs and on-par with the self-supervised ones.

References

[1] A. Bardes, J. Ponce, and Y. LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning, 2021.

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[3] A. Bojchevski, J. Klicpera, B. Perozzi, A. Kapoor, M. Blais, B. Rózemberczki, M. Lukasik, and S. Günnemann. Scaling graph neural networks with approximate pagerank. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Jul 2020.

[4] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, page 737–744, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[5] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *CoRR*, abs/2006.09882, 2020.

[6] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294, 2021.

[7] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020.

[8] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh. Cluster-gcn. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Jul 2019.

[9] J. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. Á. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent: A new approach to self-supervised learning. *CoRR*, abs/2006.07733, 2020.

[10] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs, 2018.

[11] K. Hassani and A. H. K. Ahmadi. Contrastive multi-view representation learning on graphs. *CoRR*, abs/2006.05582, 2020.

[12] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick. Momentum contrast for unsupervised visual representation learning. *CoRR*, abs/1911.05722, 2019.

[13] Z. T. Kefato and S. Girdzijauskas. Self-supervised graph neural networks without explicit negative sampling. *CoRR*, abs/2103.14958, 2021.

[14] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks, 2017.

[15] I. Misra and L. van der Maaten. Self-supervised learning of pretext-invariant representations. *CoRR*, abs/1912.01991, 2019.

[16] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang. Geom-gcn: Geometric graph convolutional networks, 2020.

[17] B. Rozemberczki, C. Allen, and R. Sarkar. Multi-scale attributed node embedding, 2021.

[18] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann. Pitfalls of graph neural network evaluation, 2019.

[19] S. Thakoor, C. Tallec, M. G. Azar, R. Munos, P. Veličković, and M. Valko. Bootstrapped representation learning on graphs, 2021.

[20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks, 2018.

[21] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm. Deep graph infomax, 2018.

[22] Y. You, T. Chen, Y. Shen, and Z. Wang. Graph contrastive learning automated. *CoRR*, abs/2106.07594, 2021.

[23] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen. Graph contrastive learning with augmentations. *CoRR*, abs/2010.13902, 2020.

[24] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow twins: Self-supervised learning via redundancy reduction. *CoRR*, abs/2103.03230, 2021.

[25] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna. Graphsaint: Graph sampling based inductive learning method, 2020.

[26] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, WWW '21, page 2069–2080, New York, NY, USA, 2021. Association for Computing Machinery.

A Appendix

A.1 Dataset Description

All of the datasets are collected from PyTorch Geometric (PyG) [1], and grouped as

- Citation Networks (*Cora Full*, *DBLP*, and *PubMed*): Paper to paper citation networks, and we classify papers into different subjects [10].
- Co-Author Networks (Computer Science (*CS*) an *Physics*): Author collaboration network from Microsoft Academic Graph, and the task is to predict the active field of authors [18].
- Co-Purchased Products Network (*Computers* and *Photo*): Co-purchased products from the respective categories on Amazon, and the task is to predict the refined categories [18].
- Wikipedia (*Actor* and *WikiCS*): WikiCS contains Wikipedia hyperlinks between Computer Science articles, and we classify articles into branches of CS [18], and Actor contains actors co-occurrence on the same Wikipedia article and we classify actors into groups based word of actors' Wikipedia [16].
- Social (*Facebook*, *Flickr*, *GitHub*, *Reddit*, and *Yelp*): Facebook contains a page to page graph of verified Facebook sites, and we want to classify pages into their categories [17]. Flickr contains a network of images based on common properties (e.g., geo-location) along with their description, and the task is to predict a unique tag of an image [25]. GitHub contains the social network of developers, and we want to classify developers as web or machine learning developers [17]. Yelp is also the social network of Yelp users, and we predict business categories each user has reviewed. For Reddit, we predict the subreddits (communities) of user posts [10, 25].

A brief summary of the datasets is provided in Table 3. We also group them into two, as large (Yelp and Reddit) and small (the rest).

A.2 Ablation Study

In the following, we investigate the impact of different aspects of SURGEON.

A.2.1 Loss Function

We have seen the regularized optimization objective in Section 2.2 and also shown a way to improve it. In the following, we analyze the effect of using the original vs. the improved loss function with respect to prediction accuracy and resource usage. For the qualitative experiment, we train both flavors for 100 epochs and just 1 epoch otherwise. The results of this experiment are reported in Fig 2. As expected, both flavors achieve similar qualitative performance. Nonetheless, the improved version is significantly better than the original one in terms of memory usage and run time.

[1] https://pytorch-geometric.readthedocs.io/en/latest/index.html

Dataset	N	M	F	C	Task
Cora Full	19,793	126,842	8,710	70	MCC
DBLP	17,716	105,734	1,639	4	MCC
PubMed	19,717	88,648	500	3	MCC
Physics	34,493	495,924	8,415	5	MCC
CS	18,333	163,788	6,805	15	MCC
Computers	13,752	491,722	467	10	MCC
Photo	7,650	238,162	745	8	MCC
Facebook	22,470	342,004	128	4	MCC
Flickr	89,250	899,756	500	7	MCC
GitHub	37,700	578,006	128	2	BC
WikiCS	11,701	297,110	300	10	MCC
Actor	7,600	30,019	932	5	MCC
Yelp	716,847	13,954,819	300	100	MLC
Reddit	232,965	114,615,892	602	41	MCC

Table 3: Summary of the datasets, and $N = |V|$, $M = |E|$, F is the number of features, and C is the number of classes. BC, MCC and MLC represent binary, multi-class and multi-label classification, respectively

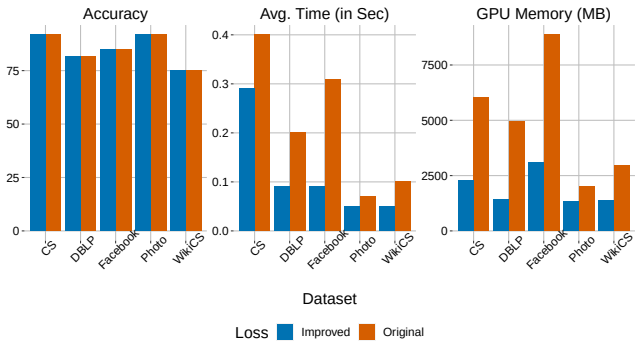

Figure 2: Comparison of the original and improved loss function in terms of accuracy, memory usage and run time (time to finish an epoch).

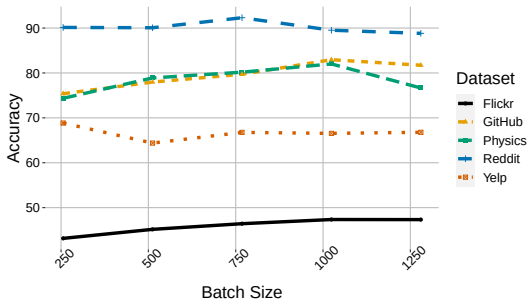

Figure 3: Effect of batch size on SURGEON's performance

A.2.2 Batch Size

As contrastive signals are indirectly injected because of the orthogonality constraint of Eq. 2, it is important to analyze the impact of batch size to see if a large batch size is needed to effectively avoid trivial solutions. For this reason, we train the model using sampled neighborhood subgraphs [10] instead of full-batch, and both the model and the linear head are trained for 100 epochs. The results are reported in Fig. 3, and in general, performance is directly proportional to batch size until a certain point. For small datasets, there is improvement up to 1024. The largest improvements are for the GitHub and Physics datasets, which are 7.57 (from 75.38% to 82.95%) and 7.66 percentage points of accuracy (from 74.35% to 82.01%), respectively. However, that is not the case for the larger ones (Reddit and Yelp), where both smaller and larger batch sizes give a comparable performance. Overall, we have not observed qualitative differences for batch sizes bigger than 1024. In our experiments, batch size greater than 1024 is only related to faster training, not improved quality.

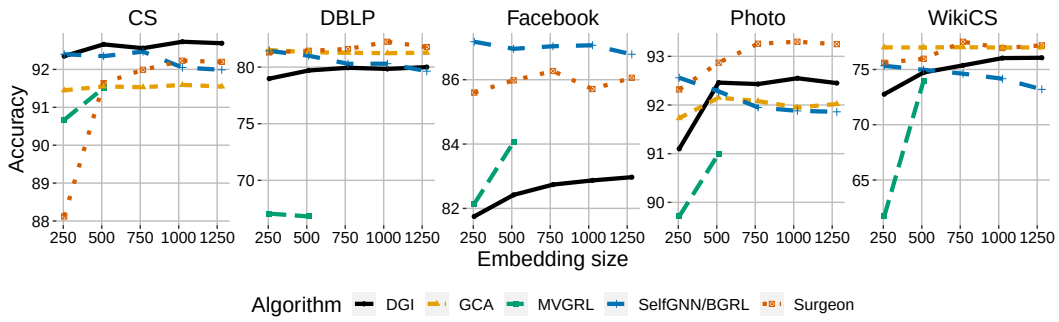

Figure 4: Effect of embedding size on SURGEON and the SSL-GNN baselines.

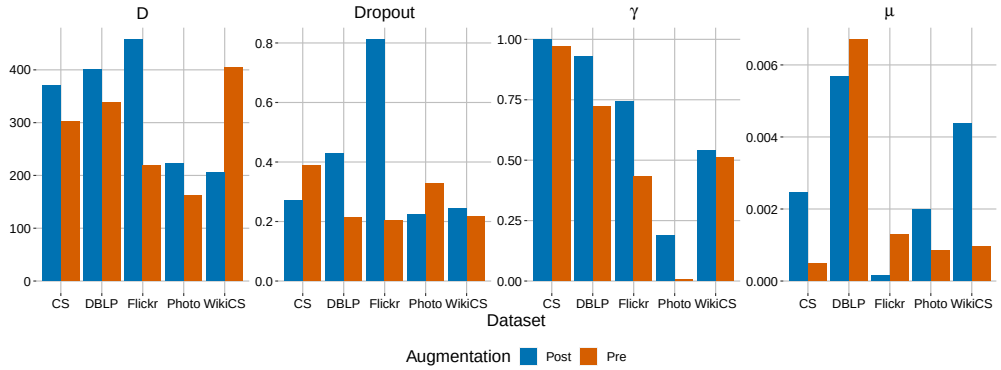

Figure 5: Tuned values of the hyperparameters of SURGEON for the pre and post augmentation techniques. D is the number of features after augmentation and the GNN encoder for pre and post augmentations, respectively. γ the weight of the constraint in Eq. 2, and μ is the learning rate.

A.2.3 Embedding Size

To provide a perspective, for this analysis we include the baselines. We use the same setting as the first experiment (Tables 1 and 2), and we examine embedding sizes in $[256, 512, 768, 1024, 1280]$. SURGEON and DGI have the tendency to improve as we increase the embedding size. On the other hand, GCA and SELFGNN/BGRL, stay the same or decrease. For MVGRL, it seems that it has the tendency to improve proportionally to the embedding size. However, we were able to observe only for 256 and 512, as it throws an out-of-memory error for larger values.

A.2.4 Pre vs. Post Augmentation

In terms of performance, both augmentation techniques give equivalent results. The only difference is that the hyperparameters need to be tuned separately for each one. In Fig. 5 we show the final configuration of the hyperparameters that maximize prediction accuracy on the validation set, which are obtained using Bayesian optimization. As can be seen from the figure, they converge to different values.

Since the main motivation for introducing the post-augmentation is efficiency, in Fig. 6 we show the memory usage and run time (to finish an epoch) required by these variants. As anticipated, the post-augmentation is significantly faster and has a lot less memory overhead.

A.2.5 Symmetry vs. Asymmetry

SURGEON's architecture can easily be replaced by an asymmetric one. For this reason, we create asymmetry just by adding a prediction head on the left network and inserting batch norm in the GNN encoder [13, 24, 19]. We update the parameters of both the left and right networks using stochastic

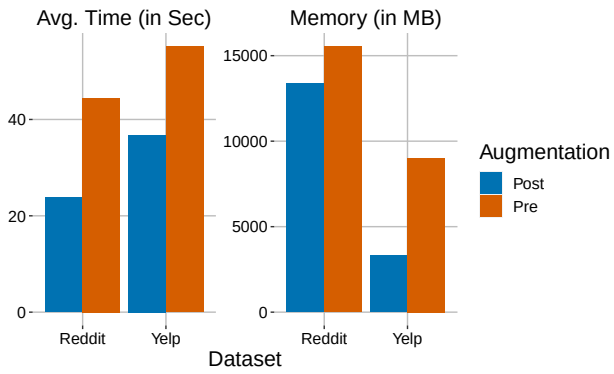

Figure 6: Analysis of Pre and Post Augmentation techniques in terms of memory usage and run time complexity (time to finish an epoch)

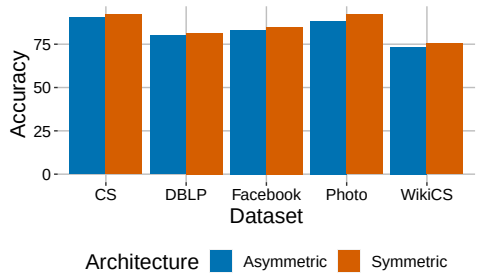

Figure 7: Comparison between SURGEON's architecture (Symmetric) and an Asymmetric architecture

gradient descent. As shown in Fig. 7, the performance of the asymmetric architecture is slightly lower than the symmetric one.

A.2.6 Convergence

Recent studies [26, 24] have shown that SSL-GNN methods usually require a large number of epochs (several thousand) to achieve a performance comparable to semi-supervised methods. In Fig. 8 we show SURGEON's convergence, and usually, 50 epochs are sufficient to achieve comparable performance to semi-supervised methods.

A.3 Implementation Details

SURGEON is implemented using PyTorch and PyTorch Geometric libraries.

For each augmentation head, we use a simple one-layer linear head. To avoid overfitting, we use dropout in both heads.

For the GNN encoder, we use two types of architectures, which are GCN [14] and GRAPHSAGE [10]. A full-batch GCN is used for the small datasets, and a mini-batch GNN based on GRAPHSAGE with neighborhood sampling [10] is used for the larger ones. As stated earlier, one can substitute these

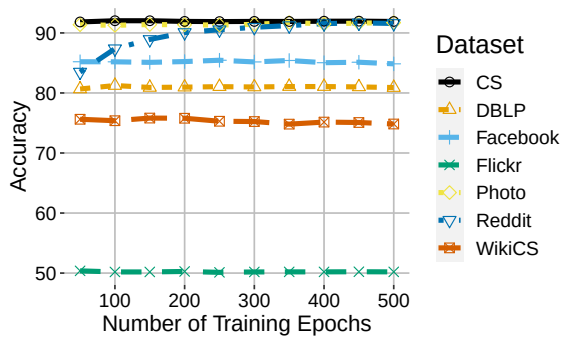

Figure 8: Analysis of the convergence of SURGEON

with any other architecture as necessary. A dropout is also added, and finally, we use a residual connection for the GNN encoder.

We use the output of the GNN encoder as the embedding of nodes for the pre architecture, whereas for the post architecture we use the augmented latent representations.

Although existing SSL-GNN methods [24, 19, 19, 13] require different normalization strategies, such as Batch Norm and Layer Norm, that is not necessary for SURGEON; as a result, no such normalization is used.