# Self-Supervised GNN that Jointly Learns to Augment

Zekarias T. Kefato†, Sarunas Girdzijauskas† and Hannes Stärk††
†KTH Royal Institute of Technology ††Technical University of Munich
zekarias@kth.se,sarunasg@kth.se,hannes.staerk@tum.de

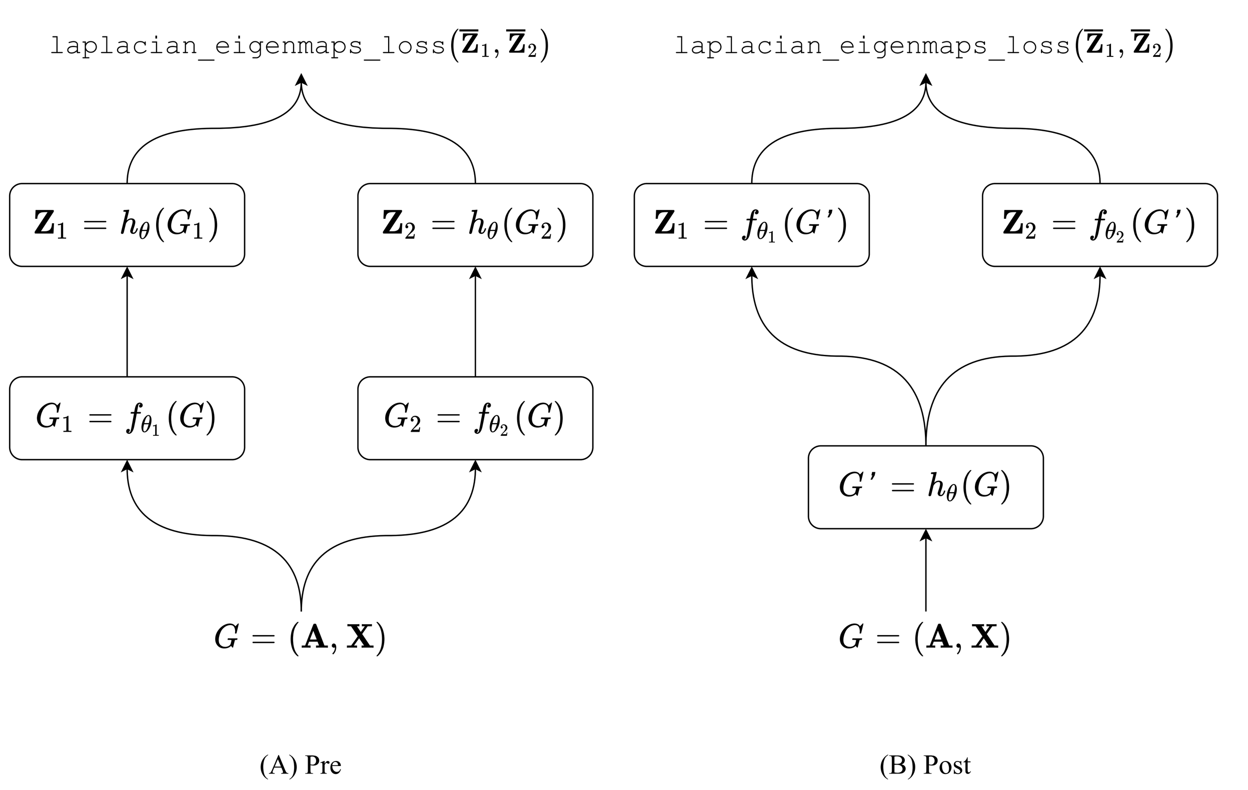
## Overview

- Self-supervised Learning (SSL) aims at learning representations of objects without relying on manual labeling.
- Graph data augmentation is a key challenge in SSL for graphs, existing studies use heuristics for graph data augmentation
- We propose a self-supervised GNN framework called Surgeon that jointly learns both the data augmentation and representation.
- Surgeon doesn't require explicit negative sampling, i.e., needs only positive pairs.
- Instead of engineering tricks as in previous works, Surgeon employs a principled objective based on Laplacian Eigenmaps that avoids collapse.

## Inputs

- A graph $G = (\mathbf{A}, \mathbf{X})$, where $\mathbf{A} \in [0,1]^{N\times N}$ is the adjacency matrix and $\mathbf{X}^{N\times F}$ is a feature matrix. $N$ is the number of nodes and $F$ is the number of features.
- A GNN encoder $h_\theta$, $\mathbf{Z} = h_\theta(A, X) = h_\theta(G)$

## Surgeon Architecture




## Learning

- Surgeon learns node representations in a self-supervised fashion using a *pre* or *post* architecture shown in the **Architecture** box.
  - **pre**: augmentation in the input space.
  - **post**: augmentation in the latent space and introduced to improve resource usage
- $f_{\theta_1}$ and $f_{\theta_2}$ are functions to generate two augmented views of each node.
  - Learned, i.e. governed by the signal encoded in the graph
- The unit vectors, $\bar{\mathbf{Z}}_1$ and $\bar{\mathbf{Z}}_2$, are the learned representations for the two views

## Learning Objective

Based on Laplacian Eigenmaps

$$\mathcal{L}_{\theta,\theta_1,\theta_2} = ||\bar{\mathbf{Z}}_1 - \bar{\mathbf{Z}}_2||_F^2 + \gamma\big(||\bar{\mathbf{Z}}_1\bar{\mathbf{Z}}_1^{\mathbf{T}} - \mathbf{I}_N||_F + ||\bar{\mathbf{Z}}_2\bar{\mathbf{Z}}_2^{\mathbf{T}} - \mathbf{I}_N||_F\big) \quad (1)$$

*Note: $\theta_1$ and $\theta_2$ are jointly learned with $\theta$*

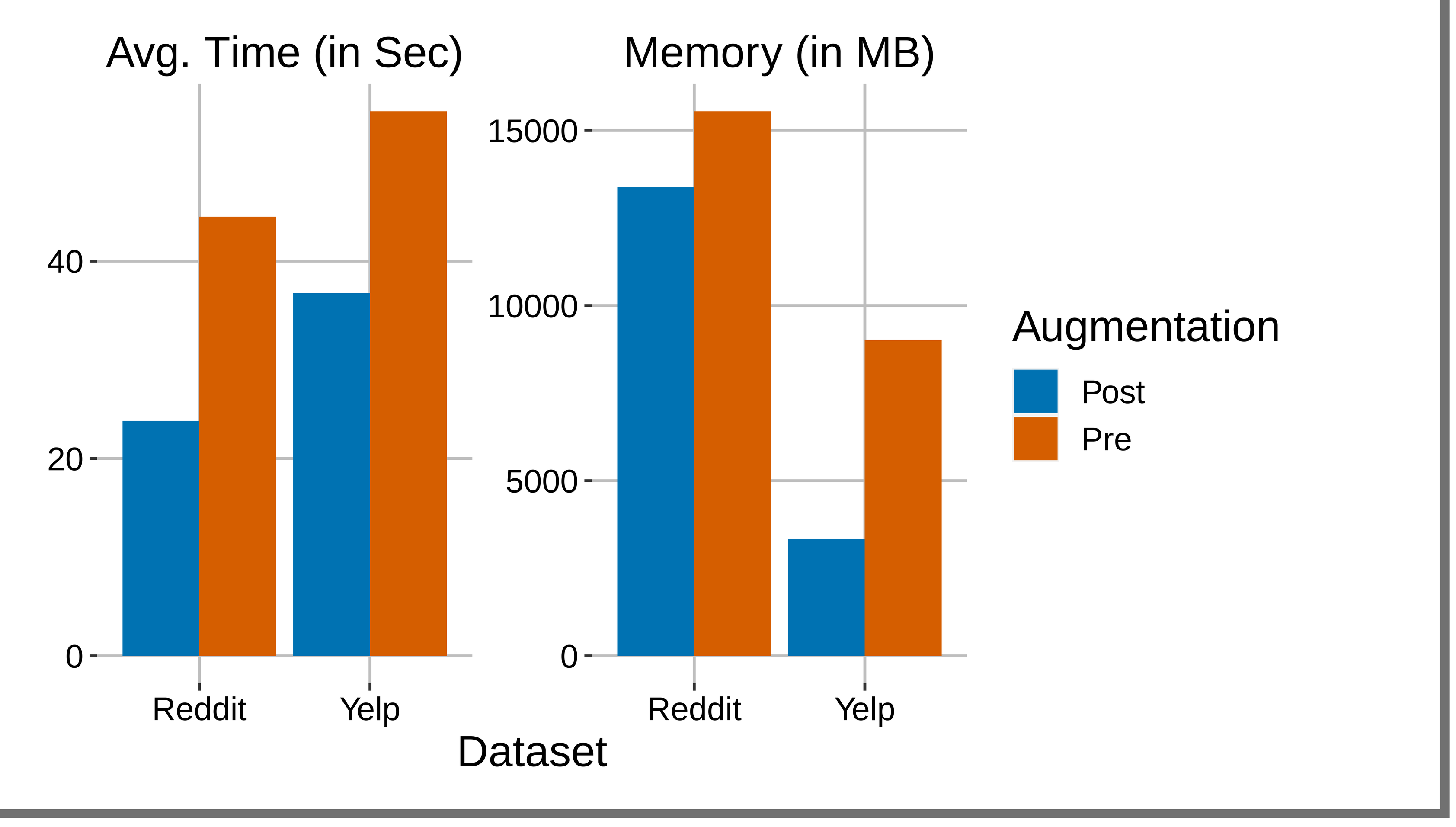
## Node classification (NC) results for small to medium scale datasets

| **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** |

## NC results for large scale datasets

| 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 |

## Resource usage: pre vs post architecture




## Conclusion

- Augmentations can be jointly learned with representation
- Perturbations (e.g. dropping edges and nodes) are ill defined for some domains (e.g. molecular graphs), and hence learned augmentations are more suitable (Future work)
- Post architecture improves resource usage