ON RECOVERABILITY OF GRAPH NEURAL NETWORK REPRESENTATIONS

Maxim Fishman[*1,2], **Chaim Baskin**[*2], **Evgenii Zheltonozhskii**[*2], **Ron Banner**[1], **Avi Mendelson**[2]
[1] Habana Labs – An Intel company, Caesarea, Israel
[2] Technion – Israel Institute of Technology

ABSTRACT

Despite their growing popularity, graph neural networks (GNNs) still have multiple unsolved problems, including finding more expressive aggregation methods, propagation of information to distant nodes, and training on large-scale graphs. Understanding and solving such problems require developing analytic tools and techniques. In this work, we propose the notion of *recoverability*, which is tightly related to information aggregation in GNNs, and based on this concept, develop the method for GNN embedding analysis. Through extensive experimental results on various datasets and different GNN architectures, we demonstrate that estimated recoverability correlates with aggregation method expressivity and graph sparsification quality. The code to reproduce our experiments is available at `https://github.com/Anonymous1252022/Recoverability`.

1 INTRODUCTION

Over the last decade, deep learning allowed researchers to tackle multiple hard tasks, previously considered intractable. For example, convolutional neural networks (CNNs) have been successfully applied to computer vision problems such as image classification (He et al., 2016), object detection (Ren et al., 2015), and semantic segmentation (Ronneberger et al., 2015). Nevertheless, while CNNs are successful in processing pixels, multiple other modalities such as 3D meshes (Wu et al., 2019), social networks (Ribeiro et al., 2017), brain connections (Shapson-Coe et al., 2021) and many others, require the processing of irregular data. As a result, in recent years, researchers have been looking into ways to exploit deep learning methods, such as CNNs, in order to work with graph structured data. Deep learning on graphs and, in particular, graph neural networks (GNNs, Gori et al., 2005; Scarselli et al., 2008; Kipf & Welling, 2017), based on message passing (Gilmer et al., 2017), has become a very popular tool for machine learning with graphs.

Despite their exceptional ability to learn graph-based data representations, GNNs still suffer from some important problems. One issue is related to aggregation method expressiveness. Kipf & Welling (2017) proposed graph convolution networks (GCNs), which aggregate information from neighboring nodes but cannot distinguish between them, Hu et al. (2020) integrated a self-attention mechanism (Vaswani et al., 2017) into aggregation. Another problem is lack of ability to propagate information between distant nodes in the graph. Li et al. (2018) conjectured that this phenomenon is a result of *over-smoothing*, Alon & Yahav (2021) offered another explanation – *over-squashing*. Finally, training on large-scale graphs is a significant obstacle in integrating GNNs in real-life problems. The diameter (maximal distance between two nodes) is often small (3–5) even for large graphs. Thus, GCNs with only 3–5 layers must aggregate information from the whole graph to calculate embedding of a single node, which requires a large amount of memory and compute. Graph sparsification by dropping a subset of edges (Srinivasa et al., 2020; Rathee et al., 2021) can diminish the problem.

In this work, we introduce the notion of *recoverability* and demonstrate its tight relationship to information aggregation in GNNs. Recoverability provides an alternative insight into the aforementioned problems and can serve as a good tool for understanding their roots and finding better solutions for them. First, we provide a qualitative definition of recoverability and its empirical counterpart, recoverability loss. We then use reproducing kernel Hilbert space (RKHS) embedding for estimating

[*] Equal contribution.

recoverability loss in an efficient and differentiable way. Finally, we show how to use recoverability for measuring the quality of embedding method.

2 Method

We characterize neural graph embedding as having three parameters: an aggregation algorithm, a certain number of layers, and a graph sparsification algorithm. Our goal is to compare different embeddings based on those parameters and see which one performs better on the given data. To achieve this goal, we define a method for analyzing neural graph embeddings. First, the problem of measuring the quality of a given embedding is reduced to the problem of determining how well one can recover one random variable from another. Second, we define a new concepts of recoverability and recoverability loss. Third, we demonstrate how to compute recoverability loss in a differentiable and efficient way. Finally, we present an algorithm for measuring graph embedding quality using recoverability.

From an embedding to recoverability. In the node property prediction task, given graph $G = (V, E)$ equipped with node features $\mathcal{X} = \{x_v\}_{v\in V}$ and labels $\mathcal{Y} = \{y_v\}_{v\in V}$ distributed according to some probability distribution $(x_v, y_v) \sim (X, Y)$, one should predict the label of each node in the graph. To solve this task we embed the graph with node features $(G, \mathcal{X})$ into a latent space, acquiring some representation of each node, which we denote as $h_v \sim H$, and then apply a classifier to the embedding h_v, trying to predict label y_v. The success of the above procedure is highly dependent on the ability to learn a map between random variables H and Y. Since different embedding methods produce different random variables, we can reduce the problem of measuring the embedding method's quality to measuring the ability to learn random variable Y from random variable H. This leads to the natural question of how much information do we have in random variable H to recover Y. We term the ability to recover Y from a given H *recoverability*.

Recoverability and recoverability loss. Given two random variables H and Y, with the values in $U \subset \mathbb{R}^d$ and $\mathbb{R}$, respectively, we say that Y is fully recoverable from H if there exists a continuous function $f : U \to \mathbb{R}$ such that $f(H) = Y$. In the general case, however, we do not have such a relation between H and Y, which means that we cannot fully recover Y from H. Nevertheless, H may still contain some information from which we can partially recover Y. To measure the amount of such information in H, we define the set $\mathcal{C}_H = \{f(H) \mid f : U \to \mathbb{R} \text{ is continuous function}\}$ i.e., the collection of all random variables that could be fully recovered from H, and then evaluate the distance between random variable Y and the set $\mathcal{C}_H$: $\rho(Y|H) = \inf_{Z\in\mathcal{C}_H} d(Y, Z)$, where d is some distance function. We denote the value $\rho(Y|H)$ as the recoverability loss. When we are given two different random variables H_1 and H_2 (i.e., two different embeddings of $(G, \mathcal{X})$) such that $\rho(Y|H_1) < \rho(Y|H_2)$, Y is more recoverable from H_1 than from H_2 (which means that embedding method 1 is better than method 2). This case is demonstrated in Fig. C.3.

Recoverability loss estimation. We denote the empirical estimation of $\rho(Y|H)$ on a finite collection of samples $\{(h_n, y_n)\}_{n\in[N]}$ by $\rho^*(Y|H)$ and use the distance induced form the L_p-norm, where $p \in [1, \infty)$.

Theorem 1 *The empirical estimation of recoverability loss is given by* $\rho^*(Y|H) = \frac{1}{N^{1/p}}\|(\mathbb{I} - \Pi)y\|_p$, *where* Π *is an orthogonal projection to the* $\operatorname{Im}(K)$ *and* $K_{ij} = \exp\left(-\frac{\|h_1 - h_2\|^2}{2\sigma^2}\right)$ *is a Gram matrix.*

We provide the proof of the Theorem 1 in Appendix A.

An algorithm for embedding quality estimation. In Algorithm 2, we first, apply an embedding method EM to $(G, \{x_n\}_{n\in[N]})$ to produce $h_n \sim H$, which is equivalent in time to a single forward pass of GNN on the full graph. We then use $\{h_n\}_{n\in N}$ to estimate $\rho(Y|H)$ using Algorithm 1. The time complexity of Algorithm 1 is $\mathcal{O}\big(Nm^2\big)$, where N is the number of nodes in the graph and m is the batch size of Algorithm 1. This allows fast approximation of the recoverability loss, as compared to training a GNN model on a given dataset.

Algorithm 1 $\rho(Y|H)$ estimation.

```
Input: $\{(h_n, y_n)\}_{n\in[N]}$
Output: $\rho^*(Y|H)$
for $j^{\text{th}}$ batch $\left\{(\tilde{h}_i, \tilde{y}_i)\right\}_{i\in[m]} \subseteq \{(h_n, y_n)\}_{n\in[N]}$ do
    $K_{kl} \leftarrow \exp\left(-\frac{\|\tilde{h}_k - \tilde{h}_l\|^2}{2\sigma^2}\right)$    Compute distance matrix.
    $U, \Lambda \leftarrow \text{eig}(K)$    Compute eigendecomposition $K = U^{-1}\Lambda U$ of $K$.
    $\Pi \leftarrow \sum_{\lambda_i>0} u_i u_i^T$    Compute projection to $im(K)$.
    $\rho_j^*(Y|H) \leftarrow \frac{1}{m^{1/p}} \|(\mathbb{I} - \Pi)\tilde{y}\|_p$
end for
$\rho^*(Y|H) = \frac{m}{N} \sum_j \rho_j^*(Y|H)$
```

Algorithm 2 Embedding method quality.

```
Input: $(G, \{x_n\}_{n\in[N]})$, $\{y_n\}_{n\in[N]}$ and embedding method EM
Output: embedding method EM quality
$\{h_n\}_{n\in[N]} \leftarrow \text{EM}((G, \{x_n\}_{n\in[N]}))$    Generate outcomes of $H$.
$\rho^*(Y|H) \leftarrow Algorithm\_1(\{(h_n, y_n)\}_{n\in[N]})$    Use Algorithm 1 to estimate $\rho$.
```

3 EXPERIMENTS

In this section we provide extensive experimental results on real datasets and different GNN architectures, where we show various applications of the proposed method for graph embedding analysis. The experimental settings are given in Appendix C.

The usefulness of edges. The first experiment shows an interesting property of the datasets appearing in Table C.2, which could help elucidate why GNNs with a SAGEConv layer can be used efficiently for training on these. We took three consecutive SAGEConv layers without learnable parameters as embedding method EM, i.e., only the aggregation parts (as shown in Appendix C), and applied it to the node features. Let X denote the random variable of node features, H the node embedding (after EM application) and Y the node classes. From Table 1 we can see that $\rho^*(Y|H) < \rho^*(Y|X)$. In other words, Y is more recoverable from H (node embedding) than from X (node features). Additionally, we observe an interesting correlation between recoverability loss drop and homophily. In the Reddit2 and ogbn-products datasets, which have a relatively high homophily, the recoverability loss drops relatively sharply, whereas in the Flickr dataset, which has a relatively low homphily, the recoverability loss changes relatively slightly.

Correlation between recoverability and aggregation method quality. This experiment shows the correlation between recoverability loss and aggregation method quality on a given dataset, i.e., the test accuracy after the training on a GNN model defined in Appendix C. The results are shown in Figs. 1 and C.4 to C.7, where the y axis is the test accuracy of the trained GNN model on a given dataset, and the x axis is the inverse of the estimated recoverability loss. For GraphConv, GCNConv, SAGEConv and GINConv, the estimation of recoverability loss was done only on the aggregation part, according to Appendix C. For GATv2Conv, the recoverability loss was computed for the node embedding of the trained model. From all plots it can be seen that as the recoverability loss drops, the test accuracy becomes higher.

Correlation between recoverability and the graph sparsification method quality. In this experiment we show that the quality of the given sparsification method correlates with recoverability. We use two simple sparsification methods. In **Random** sparsification we randomly drop 90% of the edges from the graph. In **Max d** sparsification we find the maximal value of d that satisfies $\sum_{v\in V} \min\{d, d_{\text{in}}(v)\} \leqslant 0.1|E|$, where $d_{\text{in}}(v)$ is an input degree of v, and for each node v in the graph we randomly leave $\min\{d, d_{\text{in}}(v)\}$ of input edges. To evaluate quality of sparsification, we first apply it to the given dataset and then train a three-layer SAGEConv GNN mode (defined in

Dataset	$\rho^*(\boldsymbol{Y}\vert\boldsymbol{X})$	$\rho^*(\boldsymbol{Y}\vert\boldsymbol{H})$
Reddit2	0.088	0.066
ogbn-arxiv	0.085	0.079
Flickr	0.188	0.183
PPI	0.374	0.287
ogbn-products	0.041	0.017

Table 1: The recoverability loss ρ before and after application of embedding method EM. One can see how recoverability loss ρ decreases when embedding method EM is applied to the node features.

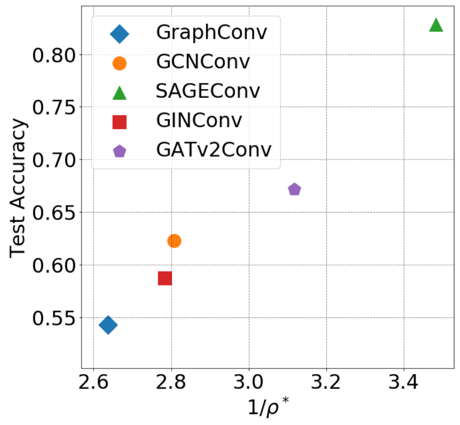

Figure 1: The correlation between the test accuracy and the recoverability loss for PPI dataset.

Dataset	**Sparsification**	
	Random	**Max d**
Reddit2	0.083 / 0.906	0.069 / 0.938
ogbn-arxiv	0.109 / 0.568	0.084 / 0.586
Flickr	0.214 / 0.466	0.191 / 0.470
PPI	0.313 / 0.598	0.266 / 0.617
ogbn-products	0.020 / 0.705	0.017 / 0.767

Table 2: Comparison of two sparsification methods, where 90% of the edges were dropped from the graphs. Each element in the table is "ρ / test accuracy". One can see the correlation between recoverability loss and the test accuracy.

Appendix C) on sparsified data. We also compute the recoverability loss of the aggregation part, as described in Appendix C. The results are shown in Table 2. We see that the **Max d** sparsification is better than **Random**, and correspondingly, the recoverability loss of **Max d** is lower than that of **Random**.

4 Conclusions

In this work, we defined recoverability, provided an efficient and differentiable method for estimation of recoverability loss and showed how it can be used for measuring the quality of GNN embedding. We demonstrated empirically that recoverability is tightly related to information aggregation in GNNs across various datasets and multiple architectures. Computing the recoverability loss is efficient and does not require training, and thus can be done quickly on large datasets. To conclude, the notion of recoverability could provide an essential tool for understanding the roots of and providing better solutions for the problems with GNNs.

REFERENCES

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=i80OPhOCVH2. (cited on p. 1)

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=F72ximsx7C1. (cited on p. 11)

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with PyTorch geometric, May 2019. URL https://github.com/pyg-team/pytorch_geometric. (cited on p. 10)

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/gilmer17a.html. (cited on p. 1)

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005. (cited on p. 1)

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html. (cited on pp. 10 and 11)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. URL https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html. (cited on p. 1)

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJlWWJSFDH. (cited on p. 1)

George S. Kimeldorf and Grace Wahba. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41(2):495 – 502, 1970. doi: 10.1214/aoms/1177697089. URL https://doi.org/10.1214/aoms/1177697089. (cited on p. 7)

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl. (cited on pp. 1 and 10)

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. URL https://ojs.aaai.org/index.php/AAAI/article/view/11604. (cited on p. 1)

Charles A. Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research*, 7(95):2651–2667, 2006. URL http://jmlr.org/papers/v7/micchelli06a.html. (cited on p. 7)

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4602–4609, Jul. 2019. doi: 10.1609/aaai.v33i01.33014602. URL https://ojs.aaai.org/index.php/AAAI/article/view/4384. (cited on p. 10)

Mandeep Rathee, Zijian Zhang, Thorben Funke, Megha Khosla, and Avishek Anand. Learnt sparsification for interpretable graph neural networks. *arXiv preprint arXiv:2106.12920*, 2021. URL https://arxiv.org/abs/2106.12920. (cited on p. 1)

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html. (cited on p. 1)

Manoel Horta Ribeiro, Pedro H. Calais, Yuri A. Santos, Virgílio A. F. Almeida, and Wagner Meira Jr. "Like sheep among wolves": Characterizing hateful users on twitter. *arXiv preprint arXiv:1801.00317*, 2017. (cited on p. 1)

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015. (cited on p. 1)

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008. (cited on p. 1)

Alexander Shapson-Coe, Michał Januszewski, Daniel R. Berger, Art Pope, Yuelong Wu, Tim Blakely, Richard L. Schalek, Peter Li, Shuohong Wang, Jeremy Maitin-Shepard, Neha Karlupia, Sven Dorkenwald, Evelina Sjostedt, Laramie Leavitt, Dongil Lee, Luke Bailey, Angerica Fitzmaurice, Rohin Kar, Benjamin Field, Hank Wu, Julian Wagner-Carena, David Aley, Joanna Lau, Zudi Lin, Donglai Wei, Hanspeter Pfister, Adi Peleg, Viren Jain, and Jeff W. Lichtman. A connectomic study of a petascale fragment of human cerebral cortex. *bioRxiv*, 2021. doi: 10.1101/2021.05.29.446289. URL https://www.biorxiv.org/content/early/2021/05/30/2021.05.29.446289. (cited on p. 1)

Rakshith S Srinivasa, Cao Xiao, Lucas Glass, Justin Romberg, and Jimeng Sun. Fast graph attention networks using effective resistance based graph sparsification, 2020. (cited on p. 1)

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html. (cited on p. 1)

Zizhao Wu, Ming Zeng, Feiwei Qin, Yigang Wang, and Jiří Kosinka. Active 3-d shape cosegmentation with graph convolutional networks. *IEEE computer graphics and applications*, 39(2):77–88, 2019. (cited on p. 1)

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km. (cited on p. 11)

A THE PROOF OF THEOREM 1

Let $U \subset \mathbb{R}^d$ be compact[1],

$$C(U) = \{f : U \to \mathbb{R} \mid f \text{ is continuous function}\}, \tag{A.1}$$

and

$$\forall h_1, h_2 \in U \qquad k(h_1, h_2) = \exp\left(-\frac{\|h_1 - h_2\|^2}{2\sigma^2}\right). \tag{A.2}$$

For each $h \in U$, define continuous function $k(h, \cdot) = \phi_h(\cdot)$, and construct the following functional space:

$$\mathcal{H}_0 = span(\{\phi_h(\cdot) \mid \forall h \in U\}) \tag{A.3}$$

Define an inner product on $\mathcal{H}_0$ by:

$$\left\langle \sum_{i=1}^{n} a_i \phi_{h_i}(\cdot), \sum_{j=1}^{m} b_j \phi_{h_j}(\cdot) \right\rangle = \sum_{i=1}^{n} \sum_{j=1}^{m} a_i b_j k(h_i, h_j) \tag{A.4}$$

Let $\mathcal{H}$ be the completion of $\mathcal{H}_0$ with respect to this inner product. Now $\mathcal{H}$ is an RKHS built from kernel $k(\cdot, \cdot)$.

Since $k(\cdot, \cdot)$ is universal kernel (Micchelli et al., 2006), the set $\mathcal{H}$ is dense in $C(U)$ with respect to the supremum norm, i.e., for any $f \in C(U)$:

$$\forall \epsilon > 0 \ \exists g \in \mathcal{H} \ \text{ s.t. } \sup_{h \in U} |f(h) - g(h)| < \epsilon \tag{A.5}$$

In addition, $\mathcal{H}$ has reproducing property:

$$\forall f \in \mathcal{H} \ \forall h \in U \quad f(h) = \langle f, \phi_h(\cdot) \rangle \tag{A.6}$$

and thus we have:

$$\rho(Y|H) = \inf_{Z \in \mathcal{C}_H} d(Y, Z) = \inf_{f \in C(U)} d(Y, f(H)) = \inf_{f \in \mathcal{H}} d(Y, f(H)) = \inf_{f \in \mathcal{H}} d(Y, \langle f, \phi_H(\cdot) \rangle) \tag{A.7}$$

We denote the estimation of ρ on a finite collection of samples $(h, y) = \{(h_n, y_n)\}_{n \in [N]}$ by ρ^* and use the distance induced from L_p-norm, where $p \in [1, \infty)$. Thus we have:

$$\rho^*(Y|H) = \inf_{f \in \mathcal{H}} \left(\frac{1}{N} \sum_{i \in [N]} |y_i - \langle f, \phi_{h_i}(\cdot) \rangle|^p \right)^{1/p} \tag{A.8}$$

From the *representer theorem* (Kimeldorf & Wahba, 1970) there exists $f^* \in \mathcal{H}$ of the following form:

$$f^* = \sum_{i \in [N]} \alpha_i \phi_{h_i}(\cdot) \tag{A.9}$$

which minimizes $\rho^*(Y|H)$. Thus:

$$\rho^*(Y|H) = \left(\frac{1}{N} \sum_{i \in [N]} |y_i - \langle f^*, \phi_{h_i}(\cdot) \rangle|^p \right)^{1/p} = \tag{A.10}$$

$$= \min_{\alpha_j \in \mathbb{R}} \left(\frac{1}{N} \sum_{i \in [N]} |y_i - \sum_{j \in [N]} \alpha_j k(h_i, h_j)|^p \right)^{1/p} = \min_{\alpha \in \mathbb{R}^N} \frac{1}{N^{1/p}} \|y - K\alpha\|_p \tag{A.11}$$

[1]It is not a restrictive assumption that U is compact since all tensor values in neural networks are bounded.

where $K_{ij} = k(h_i, h_j)$ is a Gram matrix.

Decompose y into two parts:

$$y = y_{\|} + y_{\perp} \tag{A.12}$$

where $y_{\|} \in im(K)$ and $\forall v \in im(K)\ \ y_{\perp}^T v = 0$. Then we have:

$$\rho^*(Y|H) = \frac{1}{N^{1/p}} \|y_{\perp}\|_p \tag{A.13}$$

Since K is positive semi-definite, we have the following eigendecomposition:

$$K = U \Lambda U^T \tag{A.14}$$

where columns of unitary matrix U are eigenvectors of K and Λ is a diagonal matrix of eigenvalues. Let:

$$\lambda_1, \lambda_2, ..., \lambda_k, 0, 0, ..., 0 \tag{A.15}$$

be the descending order of eigenvalues, where $\lambda_k > 0$. Then:

$$\Pi = \sum_{i \in [k]} u_i u_i^T \tag{A.16}$$

is an orthogonal projection into $im(K)$ subspace. Consequently:

$$\rho^*(Y|H) = \frac{1}{N^{1/p}} \|(\mathbb{I} - \Pi) y\|_p \tag{A.17}$$

where $\mathbb{I}$ is an identity matrix.

B SYNTHETIC DATA EXPERIMENTS

If we extend the collection of continuous functions in $\mathcal{C}_H$ (Section 2) to the measurable functions, and take as distance d the one induced from the L_2 norm on random variables, then the $Z \in \mathcal{C}_H$ that minimizes $\rho(Y|H)$ is almost everywhere equivalent to conditional expectation $\mathbb{E}[Y|H]$. Consequently, conditional expectation $\mathbb{E}[Y|H]$ can be used for testing $\rho(Y|H)$ values on synthetic data.

1-D: To demonstrate the ability of recoverability to capture the existence of a continuous map from one random variable to another, we use a simple 1D experiment. Let X be a normally distributed random variable, and let Z and W be defined as follows:

$$Z = f_1(X) = \text{sign}(X) \cdot X^2 \tag{B.18}$$

$$W = f_2(X) = X^2. \tag{B.19}$$

Since f_1 is invertible and f_2 is not, we can fully recover X from Z but not from W. Of course, we can fully recover Z and W from X, since we explicitly defined continuous maps f_1 and f_2.

For this test we generated 1000 samples of X and estimated ρ^* with Algorithm 1. The results are shown in Table B.1.

Theoretical recoverability loss for Table B.1:

$$\mathbb{E}[Z|X] = Z \quad \Rightarrow \quad \rho(Z|X) = \sqrt{\mathbb{E}[(Z-Z)^2]} = 0 \tag{B.20}$$

$$\mathbb{E}[X|Z] = X \quad \Rightarrow \quad \rho(X|Z) = \sqrt{\mathbb{E}[(X-X)^2]} = 0 \tag{B.21}$$

$$\mathbb{E}[W|X] = W \quad \Rightarrow \quad \rho(W|X) = \sqrt{\mathbb{E}[(W-W)^2]} = 0 \tag{B.22}$$

$$\mathbb{E}[X|W=w] = \tag{B.23}$$

$$= \mathbb{E}[\mathbb{1}_{X\geq 0}X|W=w] + \mathbb{E}[\mathbb{1}_{X<0}X|W=w] = \tag{B.24}$$

$$= \mathbb{E}[\mathbb{1}_{\sqrt{w}\geq 0}\sqrt{w}] + \mathbb{E}[\mathbb{1}_{-\sqrt{w}<0}(-\sqrt{w})] = 0 \tag{B.25}$$

$$\rho(X|W) = \sqrt{\mathbb{E}[(X-0)^2]} = 1 \tag{B.26}$$

100-D: Now, let X and N be 100-dimensional random vectors, with independent normally distributed entries, and Y be defined as

$$Y = \sum_{i\in[100]} (X_i + \alpha \cdot N_i), \tag{B.27}$$

where α is some parameter. When α tends to zero, Y can be fully recovered from X, and when $|\alpha|$ is large, the noise N dominates the value of X, and consequently Y cannot be recovered from X. This behavior is visualized in Fig. B.1, where $\rho^*(Y|X)$, estimated on 1000 samples, is compared to its theoretical value.

Theoretical recoverability loss for Fig. B.1:

$$\mathbb{E}[Y|X=x] = \sum_{i\in[100]} (x_i + \alpha \cdot N_i) = \sum_{i\in[100]} x_i \tag{B.28}$$

$$\mathbb{E}[Y|X] = \sum_{i\in[100]} X_i \tag{B.29}$$

$$\rho(Y|X) = \sqrt{\mathbb{E}[(Y - \sum_{i\in[100]} X_i)^2]} = |\alpha|\sqrt{100} \tag{B.30}$$

	ρ^* (mean ± std)	ρ
$\rho^*(X\|Z)$	0.119 ± 0.004	0
$\rho^*(X\|W)$	0.974 ± 0.026	1
$\rho^*(Z\|X)$	0.099 ± 0.013	0
$\rho^*(W\|X)$	0.110 ± 0.019	0

Table B.1: The estimated distance from X to $\mathcal{C}_W$ is larger than the distance from X to $\mathcal{C}_Z$, since we cannot fully recover X from W.

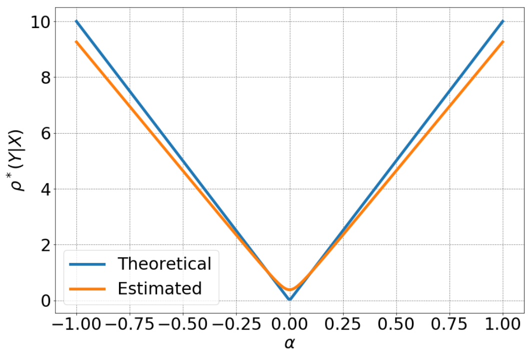

Figure B.1: Comparison between estimated $\rho^*(Y|X)$ and its theoretical value for different α.

Name	Nodes	Edges	Feature dim.	Classes	Multilabel	Train	Val	Test	Directed	Homophily
Reddit2	232,965	11,606,919	602	41	–	153,932	23,699	55,334	–	0.782
ogbn-arxiv	169,343	1,166,243	128	40	–	90,941	29,799	48,603	✓	0.654
Flickr	89,250	449,878	500	7	–	44,625	22,312	22,313	–	0.319
PPI	56,944	793,632	50	121	✓	44,906	6,514	5,524	–	0.620†
ogbn-products	2,449,029	61,859,140	100	47	–	196,615	39,323	2,213,091	–	0.808

Table C.2: Dataset statistics. † The homophily of PPI is the average over all classes.

C Real data experiments

Experimental setting: Dataset statistics, used in the following experiments, are summarized in Table C.2.

Throughout the experiments we use five different embedding layers[2]:

- GraphConv (Morris et al., 2019):
$$x_i' = \Theta_1 x_i + \Theta_2 \sum_{j \in N(i)} e_{ji} x_j \tag{C.31}$$
where Θ_1 and Θ_2 are trainable parameters.
- GCNConv (Kipf & Welling, 2017):
$$x_i' = \Theta \sum_{j \in N(i) \cup \{i\}} \frac{e_{ji}}{\sqrt{\hat{d}_j \hat{d}_i}} x_j \tag{C.32}$$
where $\hat{d}_i = 1 + \sum_{j \in N(i)} e_{ji}$, and Θ are trainable parameters.
- SAGEConv (Hamilton et al., 2017):
$$x_i' = W_1 x_i + W_2 \operatorname*{mean}_{j \in N(i)} \{x_j\} \tag{C.33}$$
where W_1 and W_2 are trainable parameters.

[2]The notation is taken from PyTorch Geometric (Fey & Lenssen, 2019).

- GINConv (Xu et al., 2019):

$$x'_i = h_\Theta \left((1 + \epsilon)x_i + \sum_{j \in N(i)} x_j \right) \tag{C.34}$$

where h_Θ is an MLP and ϵ could be trainable.

- GATv2Conv (Brody et al., 2022):

$$x'_i = \alpha_{ii}\Theta x_i + \sum_{j \in N(i)} \alpha_{ij}\Theta x_j \tag{C.35}$$

where:

$$\alpha_{ij} = \frac{\exp\big(a^T LeakyReLU(\Theta[x_i||x_j])\big)}{\sum_{k \in N(i) \cup \{i\}} \exp(a^T LeakyReLU(\Theta[x_i||x_k]))} \tag{C.36}$$

and Θ are trainable parameters.

For experimental purposes, we separate the embedding part from the classifier part in a GNN model, as described in Section 2. Therefore, for training, we use a GNN model consisting of two consecutive blocks, where the first block is an embedding built from $k \in \{1 \dots 9\}$ layers of one of the considered types, and the second block is a classifier consisting of three fully connected layers. Each layer except for the last one is followed by ReLU activation and dropout.

Aggregation method quality. To evaluate the quality of the aggregation part of some embedding method, we fix learnable parameters as identities and apply the resulting layer to the graph multiple times, defining an embedding method EM, which can be evaluated using Algorithm 2. For example, for SAGEConv (Hamilton et al., 2017), defined as follows:

$$x'_i = W_1 \cdot x_i + W_2 \cdot \underset{j \in N(i)}{\text{mean}} \{x_j\}, \tag{C.37}$$

we fix matrices W_1 and W_2 to identity.

Sparsification method quality. Similarly, to evaluate sparsification, we apply sparsification SM to the graph G, producing a sparser graph $\tilde{G} = (V, \tilde{E})$. Then we use Algorithm 2 with $(\tilde{G}, \{x_n\}_{n \in [N]})$, $\{y_n\}_{n \in [N]}$ and embedding method EM as input, as shown in Fig. C.2.

Propagation of information to distant nodes. This experiment demonstrates the depth problem in GNNs and its correlation with the recoverability. We took two GNN models with three and nine embedding layers of type SAGEConv each (the model is defined in Appendix C), and trained them on datasets from Table C.2. The results are given in Table C.3. For the Reddit2, ogbn-arxiv and ogbn-products datasets, we did not see a significant change in test accuracy after the training, whereas for the Flickr and PPI datasets, there was a degradation in accuracy for nine-layer architecture. We collected the recoverability loss of each embedding layer in the trained nine-layer architecture. The results are shown in Fig. C.8. For Reddit2, ogbn-arxiv and ogbn-products, we have a similar pattern of ρ behavior: it smoothly decreases, meaning that each consecutive layer aggregates more and more information to recover Y. For Flickr and PPI, we see different behavior. In Flickr, ρ increases, indicating that it does not aggregate any useful information from neighboring nodes to recover Y. In PPI, ρ first decreases and starts to increase after three layers, which indicates that the *over-smoothing* or *over-squashing* problem is occurring.

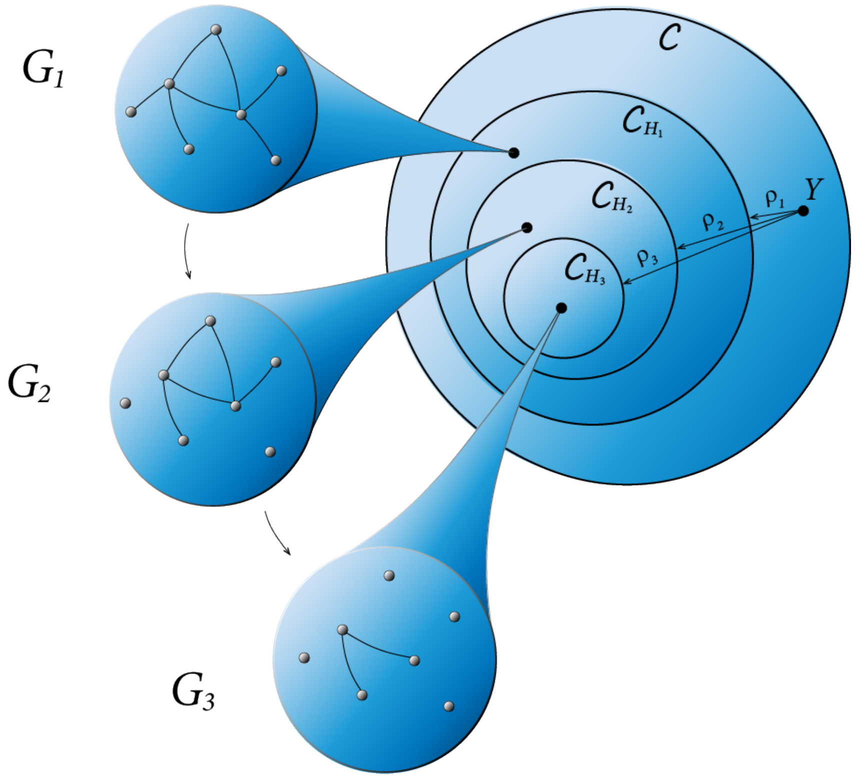

Figure C.2: The relation between graph sparsification and recoverability loss. G_2 and G_3 are sparsified versions of graph G_1, and Y is the node class random variable. The fewer edges the graph G_i has, the smaller the correspondent space $\mathcal{C}_{H_i}$ is, and, consequently, recoverability loss ρ is higher.

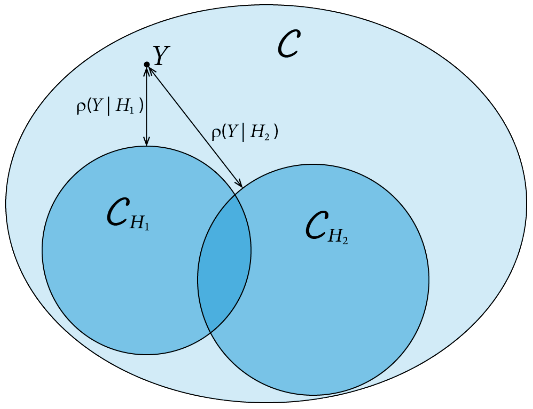

Figure C.3: For random variables H_i,$\mathcal{C}_{H_i}$ is the collection of all random variables that are fully recoverable from H_i. The distance from random variable Y $\rho(Y|H_1)$ is smaller than $\rho(Y|H_2)$, and thus Y is better recoverable from H_1 than from H_2.

D TECHNICAL NOTES

- There is a numerical instability in gradient computation of eigendecomposition $K = U\Lambda U^T$. The problem comes from equal eigenvalues in:

$$\frac{\partial L}{\partial K} = U\left\{\left(\tilde{K}^T \circ \left(U^T \frac{\partial L}{\partial U}\right)\right) + \left(\frac{\partial L}{\partial \Lambda}\right)_{diag}\right\} U^T \tag{D.38}$$

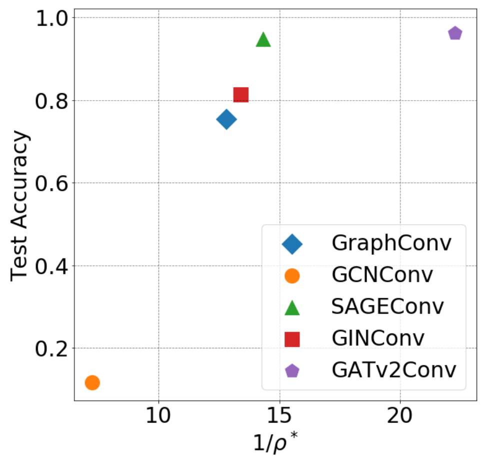

Figure C.4: The correlation between the test accuracy and the recoverability loss for Reddit2 dataset.

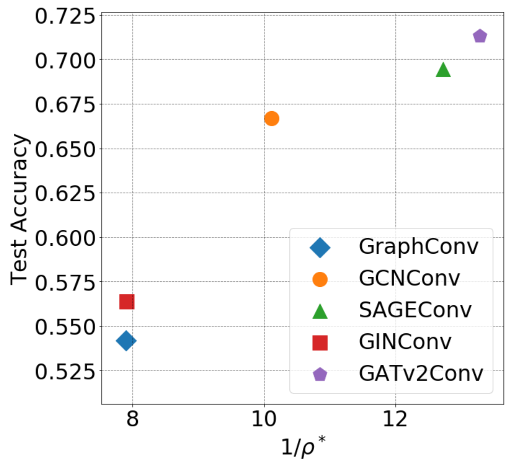

Figure C.5: The correlation between the test accuracy and the recoverability loss for ogbn-arxiv dataset.

Dataset	**Accuracy**	
	3 layers	**9 layers**
Reddit2	0.946	0.943
ogbn-arxiv	0.692	0.691
ogbn-products	0.789	0.792
Flickr	0.532	0.522
PPI	0.831	0.761

Table C.3: Test accuracy of trained models with three and nine SAGEConv embedding layers.

where:

$$\tilde{K}_{ij} = \begin{cases} \frac{1}{\lambda_i - \lambda_j}, & i \neq j \\ 0, & i = j \end{cases} \tag{D.39}$$

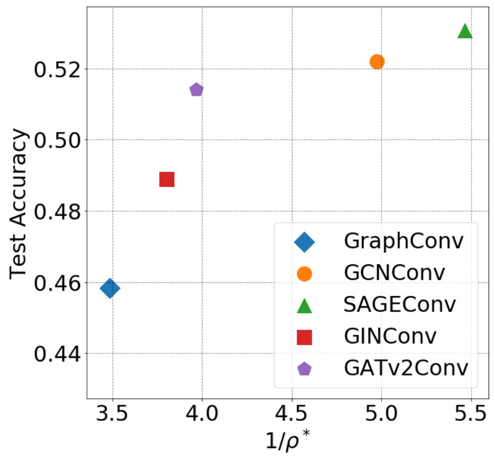

Figure C.6: The correlation between the test accuracy and the recoverability loss for Flickr dataset.

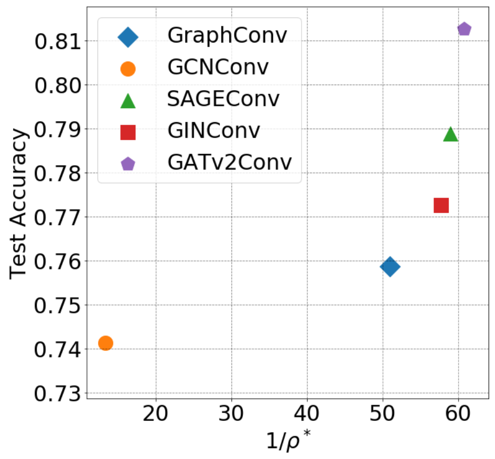

Figure C.7: The correlation between the test accuracy and the recoverability loss for ogbn-products dataset.

and $(\cdot)_{diag}$ means all off-diagonal elements set to 0. To overcome this issue, we changed $\tilde{K}$ to be:

$$\tilde{K}_{ij} = \begin{cases} \frac{1}{\lambda_i - \lambda_j}, & \lambda_i \neq \lambda_j \\ \frac{1}{\epsilon}, & \lambda_i = \lambda_j \\ 0, & i = j \end{cases} \tag{D.40}$$

- When we perform eigendecomposition, we do not have explicit zero eigenvalues. Thus instead of taking the first k non-zero eigenvalues, we simply clamp them between 0 and 1.

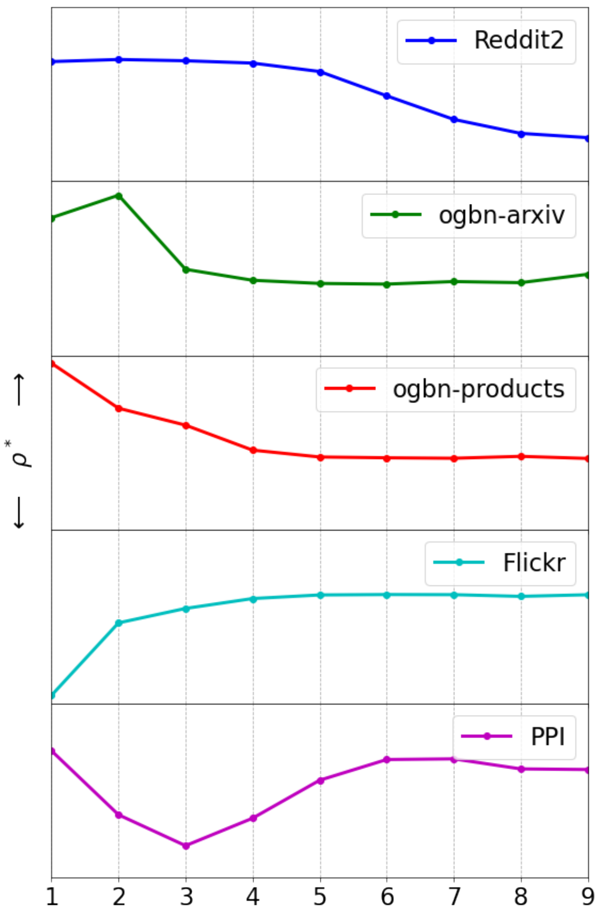

Figure C.8: The recoverability loss of each embedding layer in trained GNN model. The recoverability of the Flickr and PPI datasets behaves differently than the recoverability of the Reddit2, ogbn-arxiv and ogbn-products datasets.