Geometric Instability of Graph Neural Networks on Large Graphs

Emily Morris[*]
University of Cambridge
efm40@cam.ac.uk

Haotian Shen[*]
University of Cambridge
hs831@cam.ac.uk

Weiling Du[*]
University of Cambridge
wd301@cam.ac.uk

Muhammad Hamza Sajjad[*]
University of Cambridge
mhs57@cam.ac.uk

Borun Shi[*][†]
Neo4j
brian.shi@neo4j.com

Abstract

We analyse the geometric instability of embeddings produced by graph neural networks (GNNs). Existing methods are only applicable for small graphs and lack context in the graph domain. We propose a simple, efficient and graph-native Graph Gram Index (GGI) to measure such instability which is invariant to permutation, orthogonal transformation, translation and order of evaluation. This allows us to study the varying instability behaviour of GNN embeddings on large graphs for both node classification and link prediction.

1 Introduction

Graph representation learning [1] has seem much recent success in solving problems over relational data. The importance of stochastic effects is evident in graph learning. Earlier methods such as DeepWalk [2] and node2vec [3] leverage random walk. Graph Neural Networks (GNN)[4] contain stochastic components such as sampling and batching which allow training models on large real-world graphs. From a theoretical point of view, adding randomness is proven to improve expressiveness and performance [5][6]. The impact of randomness common in standard machine learning frameworks also carries over to the graph domain, such as in gradient descent and dataset splits[7]. It is therefore important to understand how all the stochastic components together affect graph learning models at a granular level.

Stability of embeddings and models have been extensively studied outside of the domain of graphs[8][9][10][11]. Yet the practical impact of randomness on GNNs has been very rarely studied. [12] recently empirically showed that final predictions given by GCN [13] and GAT[14] vary significantly on individual nodes despite relatively stable overall accuracies. At a finer level, [15] showed the instability in embeddings produced by several shallow methods and GraphSAGE [16] and [17] specifically for node2vec. To the best of our knowledge, all previous studies are carried out on small graphs for node classification and stability indices from other domains are directly borrowed.

Additional motivations of understanding embedding geometric stability include improving reproducibility and reliability [11], reducing retraining effort when embeddings are used by multiple downstream systems by measuring drift [18], and understanding how a model works by studying its embedding space.

The main contributions of this paper are:

Section 2 We formalise the notion of geometric stability index for embeddings and examine existing methods.

[*] All authors have equal contribution.

[†] Corresponding author.

E. Morris, H. Shen, W. Du, M. Sajjad, B. Shi, Geometric Instability of Graph Neural Networks on Large Graphs (Extended Abstract). Presented at the Second Learning on Graphs Conference (LoG 2023), Virtual Event, November 27–30, 2023.

Section 3 We propose a simple, time and space efficient and graph-native stability index, the Graph Gram index (GGI). We motivate and show that GGI is invariant to permutation, orthogonal transformation, translation and the order of evaluation.

Section 4 We show the geometric stability behaviour of several popular GNNs on graphs of varying sizes and levels of homophily for node classification and link prediction.

2 Geometric stability

Notation. Let $G = (V, E)$ denote a graph, $|V|$ the number of nodes, $|E|$ the number of relationships. Each node has an input feature in $\mathbb{R}^n$, and a final embedding in $\mathbb{R}^d$. For a given node at position i in some ordered list, we refer to that node as v_i. We refer to the final embedding of that node as $z_i \in \mathbb{R}^{1\times d}$. Whenever we need to refer to the embedding of some node v_i by name, we will abuse the notation slightly as z_{v_i}. Let $Z \in \mathbb{R}^{|V|\times d}$ denote the embeddings of all nodes in the graph. Since we are concerned with multiple embeddings produced from different configurations, let superscript Z^k denote the embeddings produced from configuration k, unless otherwise stated. Let N denote the overall number of configurations.

Stability indices that have been used to evaluate the geometric stability of embeddings of graph models are either borrowed from the NLP domain [15], or from analytic topology [17]. We give the definitions of one similarity index from each area and refer to the remaining ones in Appendix A.

Second-order cosine. Given two embedding matrices Z^l, Z^m. For each node v_i, defined an ordered list $\{u_1, \ldots, u_K\} = \mathcal{N}_k^l(v_i) \cup \mathcal{N}_k^m(v_i)$ where $\mathcal{N}_k^l(v_i)$ is the k nearest neighbour of node i under configuration l. Let $s^l(v_i)$ denote a vector about v_i where the j^{th} entry is $s_j^l(v_i) = cossim(z_i^l, z_{u_j}^l)$. Similarly we can define $s^m(v_i)$. Let $s_k^{cos[l,m]}(z_i^l, z_i^m) = cossim(s_{v_i}^l, s_{v_i}^m)$. Cosine distance are sometimes equivalently used. The second order cosine similarity of N configurations is then averaging over all nodes and all configuration pairs, i.e $\frac{1}{N^2\times|V|}\sum_{l,m\in N\times N}\sum_{i\in V} s_k^{cos[l,m]}(z_i^l, z_i^m)$. This stability index was originally used to detect words semantic shifts over time [19].

Wasserstein distance. Given two embedding matrices Z^l, Z^m, the Wasserstein distance between them is $W(Z^l, Z^m) = (\inf_{\eta:V\to V}\sum_{i\in V}\|z_i - z_{\eta(i)}\|_2^2)^{1/2}$. Where $\inf_{\eta:V\to V}$ is over all bijections between nodes. The average across all possible configuration pairs are taken as the stability index.

While these indices have shown preliminary instability results, there are several improvements to be made. First all of, most of the existing similarity indices are inefficient to compute, evident with the common *averaging over all nodes and all configuration pairs*. Finding k nearest neighbours (kNN) itself is a costly operation[20]. Opting for approximate kNN[21] introduces confounding randomness. Several of the neighbour-based indices contain hyperparameters of their own (k), which for the same reason should ideally be avoided. We give the full time and space complexity in Appendix B.

For any index that directly compares embeddings across configurations (i.e to compare z_i^l, z_i^m), an alignment step such as the solving the orthogonal Procrustes problem[22] (definition in Appendix C) needs to be done first. While embedded node neighbours can still carry important information, the interpretation of semantically similar words no longer applies. Indices from analytic topology (such as Wasserstein distance) are originally used to measure the overlap of probability distributions over metric space, which again does not map to any natural interpretation for graph embedding. Same drawback applies for Hausdorff distance (we elaborate more in Appendix D).

We propose that a stability index for embeddings produced by graph machine learning models should ideally be time and space efficient to compute, free of hyperparameters of its own, and have an intuitive and simple interpretation in the context of graphs. Having such an index enables easier analysis of embeddings produced by graph models on large real-world graphs.

3 Graph Gram Index

We propose a simple index called the Graph Gram Index (GGI).

$Z^l {Z^l}^T$ is indeed the Gram matrix, which contains all covariance information between z_i^l, z_j^l. Using the Gram matrix avoids solving the Procrustes alignment step which would be required if we calculate

```
Graph Gram Index
procedure GGI:($Z^1 \times (Z^2 \times (\ldots Z^{N-1} \times Z^N)$)
    Input: $Z^1, \ldots, Z^N$
    Output: Stability index $s \in [0, 1]$
    for $l \in [1, \ldots, N]$ do
        Center and normalize $Z^l$
        $S^l = A \circ Z^l {Z^l}^T$
        $s^l = \frac{1}{2|E|} \sum_{i,j \in |V| \times |V|} S^l[i, j]$
    $s = \mathrm{std}(s^l : l \in [1, \ldots, N])$
    where A is the adjacency matrix and $\circ$ is the hadamard product.
```

$$\begin{tikzcd}
Z^1 \ldots Z^N \arrow[r, "\forall l, m \in N \times N"] \arrow[d, "\text{Summary structure}"] & (Z^l, Z^m) \arrow[r, "\text{Align+Compare}"] & s^{l,m} \arrow[dd, "\text{Average}"] \\
S^1 \ldots S^N \arrow[d, "\text{Pool}"] & & \\
s^1 \ldots s^N \arrow[rr, "\text{std}"] & & s
\end{tikzcd}$$

Figure 1: Commutative diagram showing the approach of GGI. All previous methods follow the bottom-left path whereas we use the top-right, which scales better to large graphs and offers a flexible framework to create other indices.

$z_i^l z_j^m$. Intuitively, this is achieved because we *compare within one set of embeddings to generate a structure summary, and compare the summaries across configurations,* illustrated in Figure 1

An analogous approach underpins Centered Kernel Alignment[23](Appendix E), a *similarity* measure to compare internal representations across general neural networks (of any domain).

Applying a hadamard product of the adjacency matrix over the Gram matrix means that we only capture the node pairs that are actual edges in the graph. Intuitively, a stable model produces embeddings whose node pairs that correspond to an edge should remain similar, in their respective embedding spaces, *up to any equivariant transformation satisfied by the specific model.* Note that a node pair corresponding to an edge could potentially lie far apart in the embedding space, which previous stability indices would not consider. The framework of GGI is flexible enough to allow various extension, we list some future work in Appendix I.

3.1 Invariance properties

To the best of our knowledge, there is no prior study about the geometric invariances[24][25] a stability index should satisfy. It is important to define such invariances, because the meaningful geometric instability any index captures should be the ones that consumers of the embeddings (final GNN layer or downstream model[26][27]) are not equivariant to. At the same time, it should ideally not be only invariant to an overly specific set of transformations that few models satisfy.

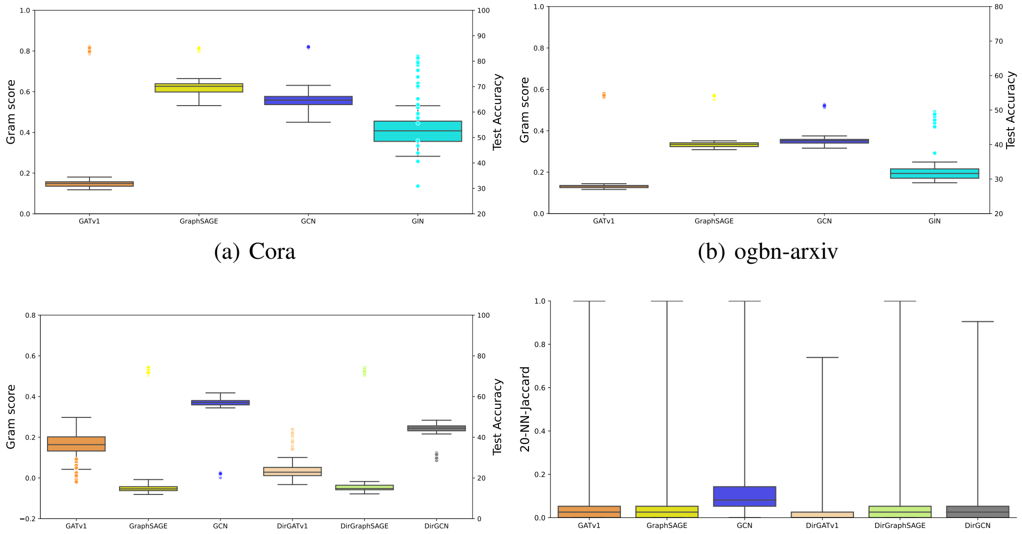
Hence, we propose that it is desirable to be invariant to node permutation, orthogonal transformation and translation. In addition it should be invariant to the order of evaluating a given set of configurations. kNN-Jaccard and second-order cossim for example are not invariant to permutation due to it's own nondeterminism from sampling. We list some future work in Appendix I.

Lemma. *GGI is invariant to node permutation, orthogonal transformation and translation of embeddings, and order of evaluation.*

Proof follows from simple applications of properties of matrix multiplication and isometries which we include in Appendix F.

4 Experiments

Node classification. We evaluate four popular baseline GNN models, GCN, GraphSAGE, GAT, GIN[30] on cora and ogbn-arxiv[31]. Additionally, we evaluate the directional version of the models[32] on roman-empire (a heterophilic dataset)[33]. We measure the stability of each model across 30 different random seeds. Full training setup is detailed in Appendix H.



(a) Cora (b) ogbn-arxiv

(c) roman-empire (d) roman-empire 20-Jaccard

Figure 2: Stability and test accuracy plots. The box plots represent the distributions of summary Gram scores. The scatter plots represent test accuracies. The spread of the box plots and scatter plots represent GGI and variance of accuracies.

Models establish consistent stability behaviour for datasets of varying sizes 2(a) 2(b), but GATv1 shows contrasting stability behaviour for homophilic and heterophilic datasets 2(c). The impact of other dataset characteristics such as degree distribution remains as an open problem.

There is some evidence of a positive correlation between geometric and downstream stability. There also seems to be a weaker correlation between geometric stability and accuracy itself. Intuitively, this suggests a ‘good and consistent’ model for a dataset learns a fixed (up to equivariance) shape of point clouds. A more detailed ablation study of the impact of architectural (e.g number of layers) and hyperparameter choices (e.g embedding dimension) is required to draw a stronger conclusion.

The difficulties of applying previous indices also become apparent. For ogbn-arxiv, it takes over a day compute 20NN-Jaccard, and prohibitively long for all other indices. It however only takes **less than 20 minutes** to calculate GGI with a single GPU. The lack of invariance properties makes it hard to obtain conclusive results. 20NN-Jaccard is completely uninformative for roman-empire dataset 2(d). A confounding cause could be that measuring neighbouring embeddings is less effective for a heterophilic dataset. Moreoever, 20NN-Jaccard and second order cossim give contradictory results for cora (Appendix G).

	Cora	ogbn-arxiv	roman-empire
GATv1	1.7	0.7	6.7 (3.0)
GraphSAGE	3.0	1.2	1.7 (1.6)
GCN	4.0	1.6	1.8 (1.6)
GIN	6.7	3.1	

Table 1: GGI for given models and datasets.

Link prediction To the best of our knowledge there’s no previous literature studying the geometric stability of embeddings produced by link prediction models. We evaluate batched versions of GraphSAGE and GCN (with GraphSAINT sampling[34]) on ogbl-citation2[31]. Standard training setup is detailed in Appendix H.

We observe GGI for batched GraphSAGE and GCN to be *2.5%, 2.6%* respectively with box plot in Appendix H Figure 4. Interestingly, both GGI index and box-plot statistics align with the observations we see during full-batch training of respective models, on different datasets and even tasks.

References

[1] William L. Hamilton. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159. 1

[2] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450329569. doi: 10.1145/2623330.2623732. URL https://doi.org/10.1145/2623330.2623732. 1

[3] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 855–864. ACM, 2016. doi: 10.1145/2939672.2939754. URL https://doi.org/10.1145/2939672.2939754. 1

[4] 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, 2009. doi: 10.1109/TNN.2008.2005605. 1

[5] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 333–341. SIAM, 2021. 1

[6] Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2112–2118. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/291. URL https://doi.org/10.24963/ijcai.2021/291. Main Track. 1

[7] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation, 2019. 1

[8] Angana Borah, Manash Pratim Barman, and Amit Awekar. Are word embedding methods stable and should we care about it? *Proceedings of the 32nd ACM Conference on Hypertext and Social Media*, 2021. URL https://api.semanticscholar.org/CorpusID:233296559. 1

[9] Furkan Gürsoy, Mounir Haddad, and Cécile Bothorel. Alignment and stability of embeddings: measurement and inference improvement. *Neurocomputing*, 553:126517, 2023. 1

[10] Maria Antoniak and David Mimno. Evaluating the stability of embedding-based word similarities. *Transactions of the Association for Computational Linguistics*, 6:107–119, 2018. doi: 10.1162/tacl_a_00008. URL https://aclanthology.org/Q18-1008. 1

[11] David Picard. Torch.manual_seed(3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision. *CoRR*, abs/2109.08203, 2021. URL https://arxiv.org/abs/2109.08203. 1

[12] Max Klabunde and Florian Lemmerich. On the prediction instability of graph neural networks, 2022. 1

[13] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017. 1

[14] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *ICLR*, 2018. 1

[15] Tobias Schumacher, Hinrikus Wolf, Martin Ritzert, Florian Lemmerich, Jan Bachmann, Florian Frantzen, Max Klabunde, Martin Grohe, and Markus Strohmaier. The effects of randomness on the stability of node embeddings. *CoRR*, abs/2005.10039, 2020. URL https://arxiv.org/abs/2005.10039. 1, 2

[16] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *NIPS*, pages 1024–1034, 2017. 1

[17] Celia Hacker and Bastian Rieck. On the surprising behaviour of node2vec, 2022. 1, 2

[18] Ahmed El-Kishky, Thomas Markovich, Serim Park, Chetan Verma, Baekjin Kim, Ramy Eskander, Yury Malkov, Frank Portman, Sofía Samaniego, Ying Xiao, and Aria Haghighi. Twhin: Embedding the twitter heterogeneous information network for personalized recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pages 2842–2850, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539080. URL https://doi.org/10.1145/3534678.3539080. 1

[19] William L Hamilton, Jure Leskovec, and Dan Jurafsky. Cultural shift or linguistic drift? comparing two computational measures of semantic change. In *Proceedings of the conference on empirical methods in natural language processing. Conference on empirical methods in natural language processing*, volume 2016, page 2116. NIH Public Access, 2016. 2

[20] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019. 2

[21] Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018. 2

[22] Garmt B Dijksterhuis John C Gower. *Procrustes Problems*. Oxford Academic, 2007. 2

[23] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019. 3, 8

[24] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks, 2019. 3

[25] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021. 3

[26] Yann Le Cun, Ido Kanter, and Sara A. Sona. Second order properties of error surfaces: Learning time and generalization. In *Proceedings of the 3rd International Conference on Neural Information Processing Systems*, NIPS'90, pages 918–924, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc. ISBN 1558601848. 3

[27] An Mei Chen, Haw-minn Lu, and Robert Hecht-Nielsen. On the geometry of feedforward neural network error surfaces. *Neural Comput.*, 5(6):910–927, nov 1993. ISSN 0899-7667. doi: 10.1162/neco.1993.5.6.910. URL https://doi.org/10.1162/neco.1993.5.6.910. 3

[28] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks, 2022. 10

[29] MohammadReza Davari, Stefan Horoi, Amine Natik, Guillaume Lajoie, Guy Wolf, and Eugene Belilovsky. Reliability of cka as a similarity measure in deep learning. *arXiv preprint arXiv:2210.16156*, 2022. 10

[30] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019. 4

[31] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020. 4

[32] Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael Bronstein. Edge directionality improves learning on heterophilic graphs, 2023. 4

[33] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: are we really making progress?, 2023. 4

[34] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. GraphSAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJe8pkHFwS. 4

[35] Oswin Aichholzer, Helmut Alt, and Günter Rote. Matching shapes with a reference point. *Int. J. Comput. Geometry Appl.*, 7:349–363, 01 1997. 8

[36] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6076–6085. Curran Associates, Inc., 2017. 8

[37] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. 9

[38] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings, 2016. 9

Acknowledgements

We thank all reviewers for their helpful review, including the suggestions of adding a heterophilic dataset and coplotting downstream accuracies.

A Previous geometric stability indices

Aligned cosine. Given two set of embeddings Z^l, Z^m, solve the orthogonal Procrustes problem to obtain $Q^{l,m} \in \mathbb{R}^{d \times d}$. For a given node i, $s_i^{l,m}$=$cossim(s_i^l Q^{l,m}, s_i^m)$. The aligned cosine stability index is the average over all nodes and configuration pairs.

kNN-Jaccard. For a given node i and configuration l, find it's k-nearest neighbours in l and m. The kNN-Jaccard overlap $s(z_i^l, z_i^m) = \frac{N_k^l(v_i) \cap N_k^m(v_m)}{N_k^l(v_i) \cup N_k^m(v_m)}$. The kNN-Jaccard stability index is the average over all nodes and configuration pairs.

Hausdorff distance. Given two embeddings Z^l, Z^m, the Hausdorff distance is $d_H(Z^l, Z^m) = max\{\sup_{z_i \in Z^l} d(z_i, Z^m), \sup_{z_i \in Z^m} d(z_i, Z^l)\}$. The Hausdorff distance stability index is the average over all configuration pairs.

B Time and space complexities

Aligned cosine time complexity. To solve orthogonal Procrustes of Z^l and Z^m where Z has dim $|V| * d$, it's equivalent to solve SVD of $Z^l * Z^{mT}$ which takes $O(|V|^3)$. Then for all pairs of runs, it takes $O(N^2|V|^3)$. Then finding $s^{acs}(z_i^{;} z_m^l)$ = $cos(z_i^l Q^{l,m}, z_i^m)$ takes $O(d^2)$ for multiplication (and $O(d)$ for cos). For all seed pairs for all nodes, it takes $O(|V|N^2d^2)$, adding up to $O(N^2|V|^3) + O(|V|N^2d^2)$.

Aligned cosine space complexity. Remember all $Q^{l,m}$ takes $O(N^2d^2)$.

kNN-Jaccard time complexity. For kNN for a vector v_i, $O(|V|d)$. All kNN neighbours for all nodes, across all runs is $O(|V|^2Nd)$. For a node v_i, to find similarity between it's two embeddings for two runs l, m, $s(z_i^l, z_i^m) = \frac{N_k^l(v_i) \cap N_k^m(v_m)}{N_k^l(v_i) \cup N_k^m(v_m)}$, takes times $O(k)$. Over all nodes and configuration pairs it takes $O(|V|^2Nd + |V|N^2k)$.

kNN-Jaccard space complexity. All neighbour lists for all nodes across runs, $O(|V|Nk)$ number of nodes. All embeddings take $O(Nd|V|)$ which adds up to $O(|V|Nk) + O(Nd|V|)$.

Second-order cossim time complexity. kNN for all nodes across all runs, $O(|V|^2Nd)$. For each node v_i, find $N_k^l(v_i) \cup N_k^m(v_i)$ in $O(k)$. Then $s_j^l(v_i) = cossim(z_i^l, \phi^l(u_j))$ for each u_j in unioned neighrbous takes $O(d)$ for one u_j and $O(dk)$ altogether. $s^m(v_i)$ in $O(dk)$ time as well. Then to compute $s_k^{cos}(z_i^l, z_i^m) = cossim(s^l, s^m)$ takes $O(k)$, the length of two vectors. So altogether $O(dk + k) = O(dk)$. Repeat this for all pairs across configurations and all nodes takes $O(dkN^2|V|)$ time. Therefore $O(|V|^2Nd + dkN^2|V|)$ overall.

Second-order cossim space complexity. All neighbour lists for all nodes across runs, $O(|V|Nk)$ number of nodes. All embeddings of all runs need to be stored, $O(Nd|V|)$. Each node v_i also need to store it's second order similarity vector $s_l(v_i)$ for each run l, so $O(kN|V|)$ float numbers. Added up it's $O(|V|Nk) + O(Nd|V|) + O(kN|V|)$.

Hausdorff distance time complexity. Finding (max) of supremum over all configuration pairs takes $O(|V|^2N^2d)$.

Wasserstein distance time complexity. All bijections between sets of size $|V|$ can be computed in $|V|^2$. Then summing the norm of differences takes $O(|V|d)$ time. Over all $|V|^2$ bijections it takes $O(|V|^3d)$ time.

C Orthogonal Procrustes Problem

Given two matrix X, Y, the orthogonal Procrustes problem finds an orthogonal matrix Ω that best transforms X into Y. It solves $e = \arg\min_\Omega \|XT - Y\|_F$. The problem reduces to computing SVD of YX^T which takes time $\mathcal{O}(n^3)$.

D Hausdorff Distance

Given to subsets X, Y of a metric space (M, d), the Hausdorff distance $d_H(X, Y) = max\{\sup\limits_{X \in X} d(x, Y), \sup\limits_{y \in Y} d(y, X)\}$. When X, Y are compact, d lifts to a proper metric d_H over the subsets. Intuitively, this is because 'Hausdorffness' provides the level of separation and compactness guarantees the density of open sets together allows the notion of distances.

The most natural application of Hausdorff distance in computer science is for matching (two) subimages[35]. This however does not apply for graph embeddings. Each point in the embedding spaces corresponds to a specific node and the distance between $z^l_{v_i}$ and an arbitrary farthest $z^m_{v_j}$ does not immediately measure any geometric embedding stability. Moreover the choice of Hausdorffness (in the spectrum of separation axioms, T_1) in our context is somewhat arbitrary. We do not require all the nice properties a Hausdorff topological space has about the space's limits and metrizability. Some definition of T_0-distance or $T_{3\frac{1}{2}}$ (Tychonoff)-distance could have been defined as well.

E Centered Kernel Alignemnt

CKA[23] addresses the problem of measuring similarity of the internal representations of two different neural networks (across datasets) that extends previous methods such as canonical correlation analysis[36].

We draw attention the below extremely simple equation which they use, for matrix $X \in \mathbb{R}^{N \times d_1}, Y \in \mathbb{R}^{N \times d_2}$

$$\langle \text{vec}(XX^T), \text{vec}(YY^T) \rangle = \text{tr}(XX^TYY^T) = \|Y^TX\|_F^2$$

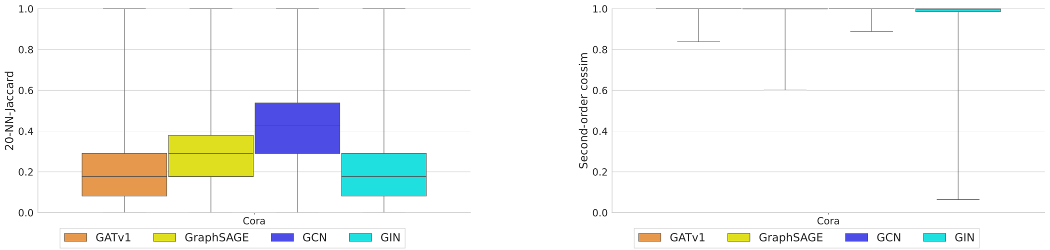
While algebraically simple, this crucially removes computation between Y and X, at the cost of computing within X and Y independently. This methodology is also one of our motivations for using the Gram matrix and separating GGI from previous indices by following the different path in the commutative diagram.

F Proof of invariance properties of GGI

Proof. *Node permutation: Given a permutation $\sigma : V \to V$ of nodes which induces a bijection on edges $\sigma : E \to E$. $s^l = \frac{1}{2|E|} \sum_{i,j \in |V| \times |V|} S^l[i, j] = \frac{1}{2|E|} \sum_{i,j \in E} (ZZ^T)[i, j] = \frac{1}{2|E|} \sum_{\sigma(i), \sigma(j) \in E} (ZZ^T)[\sigma(i), \sigma(j)]$.*

Orthogonal transformation and translation: $s^l = \frac{1}{2|E|} \sum_{i,j \in E} (ZZ^T)[i, j] = \sum_{i,j \in E} \langle z_i, z_j \rangle = \sum_{i,j \in E} \langle Tz_i, Tz_j \rangle$ for any isometry $T \in \mathbb{R}^{d \times d}$.

Once we obtain s^l scores for each embedding, any final aggregation into a single index can only be a variant to the order of evaluation. Taking standard deviation is not dependent on the ordering.



(a) 20-NN-Jaccard on Cora.

(b) Second-order cossim on Cora.

Figure 3: Box plots of previous stability indicies on Cora. The second-order cossim is uninformative.

G 20NN-Jaccard and second-order cossim node classification results

3(a) shows embeddings of GATv1 and GIN are less stable as compared to GraphSAGE and GCN. 3(b) Second order cossim is however uninformative, as illustrated by 3(b), where all scores are close to1.0, with differences only seen in the min values. This further shows the difficulty of drawing consistent conclusions based on these methods.

H Experimental setup

Code available at: `https://github.com/brs96/geometric-instability-gnn-large-graphs`

Node classification. We use a machine with 60GB of vRAM and 8 vCPU, with.one NVIDIA T4 GPU.

For training: Epochs = 100 and embedding dimension = 64 on Cora. Epochs = 500 and embedding dimension = 128 on ogbn-arxiv. Learning rate = 0.01. Number of layers = 3 (hence embeddings produced by the second layer). Loss = cross entropy. Model training implemented with PyG[37]. PyG loaders are used for dataset splits for cora (*Planetoid*, split from [38]) and ogbn-arxiv (by year). For roman-empire the split is from *HeterophilousGraphDataset* where one of the 10 provided random splits are (randomly) picked for each training setup.

Link prediction. We follow the training setup from batched GraphSAGE and GCN(GraphSAINT). We use a machine with 200GB of vRAM for ogbl-citaiton2. Equal number of positive and negative samples are used for GraphSAGE batches.Binary cross entropy loss of equal positive and negative weights are used. Embedding dimension = 128. Epochs = 200. Learning rate = 0.001. Batch size = 512. We evaluate **10** random seeds due to time and resources required for training on ogbl-citation2. For GraphSAINT sampling, walk length = 3 and number of steps = 100. Dataset split given by PyG ogbl-citation2 loader is used.

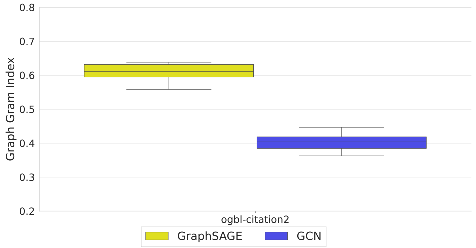


Figure 4: Box plots of GGI on ogbl-citation2.

I Future work

Graph Gram Index. The framework of GGI is flexible enough to allow various extension, for example by using non-Gram matrix based summary structure, using other the $S^l \to s^l$ pooling step such as the Frobenius norm (notably, this step is the same as many graph pooling steps), and replacing std with other summary statistics.

Invariance properties. What kind of invariance properties earlier stability indices satisfy is unstudied. It is also unknown whether the proven set of invariances for GGI (node permutation, orthogonal transformation and translation) is the complete set.

Meaningfully measuring stability under specific equivariance[28] or the sensitivity of any index to a particular transformation (for example, subspace translation of a one class[29]) remains unexplored open questions. It would be an interesting future direction to understand what type of instability is captured by GGI, and by any other index that can be created following our framework.