NORMALIZED SPACE ALIGNMENT: A VERSATILE METRIC FOR REPRESENTATION SPACE DISCREPANCY MINIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce a manifold analysis technique for quantifying the discrepancy between two representation spaces. Normalized Space Alignment (NSA) aims to compare pairwise distances between two point clouds. Our technique provides a robust means of comparing representations across different layers and models, with a particular focus on Graph Neural Networks (GNNs) to explore their unique capabilities. We show that our technique acts as a pseudometric, satisfies the properties of a similarity metric, and is continuous and differentiable. We also demonstrate that NSA can serve as an effective loss function by utilizing it in autoencoders to preserve representation structure for dimensionality reduction. Furthermore, our empirical analysis showcases that NSA consistently outperforms or matches the results of previous techniques while offering computational efficiency. Its versatility extends to robustness analysis and various neural network training and representation learning applications, highlighting its wide applicability and potential to enhance the performance of neural networks.

1 INTRODUCTION

Graph data arises in a number of application domains ranging from social sciences, infrastructure, neuroscience, and biology. Compared to unstructured data, graph analysis is more challenging due to interdependence among entities and poses challenges to the application of conventional machine learning methods that often rely on vector-based data representations. Graph Neural Networks (GNNs) are powerful tools for modeling and predicting networked data behavior (Kipf & Welling, 2017; Veličković et al., 2018). They incorporate propagation modules for aggregating neighbor-based feature and topology information and pooling modules for abstract representation creation.

GNNs have been applied for node-level tasks (node classification, node regression, node clustering, etc.), edge-level tasks (edge classification, link prediction, flow prediction, etc.) and graph-level tasks (graph classification, graph regression, graph explanation, etc.) (Zhou et al., 2020; Wu et al., 2019; Zhang et al., 2022; Ranjan et al., 2022; da Silva et al., 2021; Huang et al., 2023). Such methods represent graph elements in a latent space and solve the downstream task using such representations.

Graph representations need to be meaningful for the task at hand; representations and distances between them can provide insights into representation specifics for a task, whether representations can be transferred from one task to another, and whether representations are robust (Finn et al., 2017). But how to compare the similarity of representations is an unsolved problem. Early proposed measures were based on variants of "Canonical Correlation Analysis (CCA)" (Morcos et al., 2018; Raghu et al., 2017), and "Centered Kernel Alignment (CKA)" (Kornblith et al., 2019). Recently Barannikov et al. (2022) proposed methods for comparing two data representations using "Representation Topology Divergence (RTD)" that measures the dissimilarity in multi-scale topology between two point clouds of equal size with a one-to-one correspondence between points.

There exists a compelling need for a novel approach that bridges the gap between two distinct yet essential attributes: computational complexity and differentiability. While CKA offer an efficient means of assessing similarity, it has been proven to be lacking on several fronts (Davari et al., 2023; Williams et al., 2021; Ding et al., 2021). On the other hand, metrics like RTD provide differentiability but at the expense of computational complexity, making them impractical for large-scale datasets.

Addressing this need for a similarity metric that marries the computational efficiency of CKA with the differentiability of RTD is crucical. Such a metric not only empowers practitioners to navigate high-dimensional spaces with ease but also facilitates the seamless integration of similarity-based techniques into gradient-based optimization pipelines, thereby unlocking new frontiers in machine learning and data analysis.

We present a distance preserving similarity metric called "Normalized Space Alignment (NSA)." NSA measures the work done to align all the points in one point cloud P to another point cloud Q when both point clouds are normalized to lie in an unit space. Both point clouds can lie in different ambient spaces as long as there is a one to one mapping between the data points. The work done is measured as the difference between the normalized distance of a point to all other points in the two representations. When computed for all the points in the point cloud, the mean of the absolute work done defines the NSA between P and Q.

Though our ideas are broadly applicable to neural networks and all kinds of data, we target most of our presentation and experiments to graphs because they are unstructured and as a result, the pose the biggest challenge in creating meaningful representations. In other applications, we show how NSA serves as a structure-preserving loss function when employed in general neural networks. It can be used for dimensionality reduction with autoencoders (Trofimov et al., 2023; Moor et al., 2020; Rudolph et al., 2019), robustness analysis (Jin et al., 2020b; Zhang & Zitnik, 2020), and facilitating backward compatibility neural networks with evolving data.

In summary, we make the following contributions:

- NSA is proposed and established as pseudometric and a similarity index. NSA's quadratic computational complexity (in the number of points) is much better than some of the existing measures (e.g., cubic in the number of simplices for RTD). NSA has a strong performance on the similarity index sanity test (Kornblith et al., 2019), has a good correlation to test accuracy during training, and can accurately predict the convergence of representations across layers (Section 3).
- NSA can act as a loss function for neural networks due to its continuity and differentiability properties. NSA's ability to act as a structure preserving loss for dimensionality reduction with autoencoders is presented (Section 4). The autoencoder supported by NSA loss outperforms previous works like TopoAE (Moor et al., 2020) and RTD-AE (Trofimov et al., 2023) on several real world datasets.
- NSA can capture structural discrepancies that result from adversarial attacks (Section 5). NSA's performance on global poisoning and evasion attacks on several GNN architectures is evaluated. NSA shows a high correlation with misclassification rate across different degrees of perturbation. Furthermore, it can also rank different architectures. Simple tests with NSA can provide insights on par with other works that review and investigate the robustness of GNNs (Mujkanovic et al., 2022; Jin et al., 2020a).
- It is observed that training different GNN architectures for a shared downstream task results in comparable representation spaces. Conversely, when these architectures are trained on the same dataset but for different downstream tasks, the resulting representation spaces are dissimilar (Section 6). This suggests that focusing on aligning the data's representation space with a task-specific template, rather than optimizing on downstream task accuracy, can yield significant benefits in performance and explanations.

2 Related Work

Centered Kernel Alignment (CKA): CKA, a widely-used similarity metric, is based on Representational Similarity Matrix (RSM) and aims to measure the similarity between two representations, while maintaining invariance to scaling and rotation. The linear variant, employs mean-centered representations and a linear kernel to construct the RSM, followed by the use of the Hilbert-Schmidt Independence Criterion (HSIC) for RSM comparison. The resulting HSIC value is then normalized. However, recent studies have highlighted limitations of CKA. For instance, it has been shown that CKA lacks sensitivity to the removal of low variance principal components from representations (Ding et al., 2021). Additionally, it does not satisfy the triangle inequality, making it problematic for use as a discrepancy minimization objective function in downstream tasks (Williams et al., 2021).

Representation Topology Divergence (RTD): RTD, a Topology-based Representational Similarity Measure, approximates representation manifolds using simplices and computes RSMs for two representations, R and Q, via Euclidean distance. It employs a Vietoris Rips Filtration to identify active topological features in $\min G(R, Q)$ but not in $G(R)$. The set of all persistence intervals is then used to quantify the value of RTD. RTD's strength lies in its stronger correlation with model prediction disagreements compared to CKA. However, it has a cubic runtime complexity relative to the number of simplices due to the Vietoris Rips filtration, limiting its practicality to $n < 500$. We also observed considerable runtime variations across datasets.

Dimensionality Reduction: Recent research in dimensionality reduction addresses the challenges of high-dimensional data. Classical methods like PCA and MDS, as well as local structure-preserving approaches like t-SNE (van der Maaten & Hinton, 2008), UMAP (McInnes et al., 2018), and PaCMAP (Wang et al., 2021), remain popular. However, these methods are less practical for large real-world datasets due to high computational demands. Autoencoders (Hinton & Salakhutdinov, 2006) and variational autoencoders (Kingma & Welling, 2022) offer interpretable low-dimensional representations but may not preserve the initial data's topology. Topological autoencoders (Moor et al., 2020) introduced an additional loss term to maintain topological characteristics in latent representations. Similarly, RTD-AE (Trofimov et al., 2023) enhanced autoencoders with their topology-preserving metric RTD and outperformed TopoAE and classical techniques, establishing itself as the state-of-the-art in dimensionality reduction.

3 Normalized Space Alignment

NSA comes under a category of similarity metrics called Representational Similarity Matrix-Based (RSM) Measures (Klabunde et al., 2023). Given a representation $\mathbf{R} : N \times D$, all RSM based measures generate a matrix of instance wise similarities $D \in \mathbf{R}^{N \times N}$ where $D_{i,j} := d(R_i, R_j)$. Now for two representations $\mathbf{R}$ and $\mathbf{R}'$, we obtain two RSMs which we can operate upon to compute a similarity measure. Kornblith et al. (2019); Székely et al. (2007); Kriegeskorte et al. (2008); Chen (2022) are popular examples of RSM based similarity measures.

3.1 Definition

The following formula is used to compute the NSA between two point clouds $X = \{x_1, \ldots, x_N\}$ and $Y = \{y_1, \ldots, y_N\}$. $d(\cdot, \cdot)$ denotes the euclidean distance between two vectors.

$$\mathsf{NSA}(X, Y) = \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \left| \frac{d(x_i, x_j)}{\max_{x \in X} d(x, 0)} - \frac{d(y_i, y_j)}{\max_{y \in Y} d(y, 0)} \right|.$$

Essentially, we compute the differences of normalized distances (L1-norm) of a point to all other points in the two representations, and then take the average of all these differences.

3.2 NSA as a Pseudometric

We show that NSA is a psuedometric by proving the necessary properties in the Appendix: • $\mathsf{NSA}(X, X) = 0$ (lemma 1), • Symmetry (lemma 2), • Non-negativity (lemma 3), • Triangle Inequality (lemma 4).

3.3 NSA as a Similarity Metric

We adopt the necessary conditions of Invariance to Isotropic Scaling, Invariance to Orthogonal Transformation, and (Not) Invariance to Invertible Linear Transformation (ILT) as proposed by Kornblith et al. (2019) for a similarity metric. We establish these condition in Appendix B: lemma 7), lemma 9, and B.3. These conditions ensure that the similarity metric would be unaffected by rotation or rescaling of the representation space. Kornblith et al. (2019) proved that Invariance to ILT for a similarity metric gives the same result for any representation having width greater than or equal to the dataset size. They also discuss other scenarios wherein Invariance to ILT would be detrimental to similarity indices.

3.4 Complexity Analysis

NSA's computational complexity is given by $O(N^2D)$ where N is the number of datapoints and D is $\max(D(R), D(R'))$ where R and R' are the two representation spaces and $D(\cdot)$ is the dimensionality of the space. In practice, NSA's computation is rapid as we use torch.cdist to compute pairwise distances on the GPU. CKA has similar complexity to NSA. While RTD also starts off with $O(N^2D)$ operations to generate pairwise distances, the barcode computation is cubic in the number of simplices involved. Running times of NSA-AE are given in Table 2. NSA-AE is several times faster than RTD-AE and it can run on much larger batch sizes.

3.5 Improving Robustness of NSA

Formally, NSA is normalized by the point that is furthest away from the origin for the representation space. In practice, NSA tends to yield more robust results and exhibits reduced susceptibility to outliers when distances are normalized by scaling relative to a quantile of the distances measured from the origin (e.g., 0.98 quantile). This quantile-based approach ensures that the distances are adjusted proportionally, taking into account the spread of values among the points with respect to their distances from the origin. It also ensures that any outliers do not affect the rescaling of the point cloud to the unit space.

3.6 Comparing Different Initializations

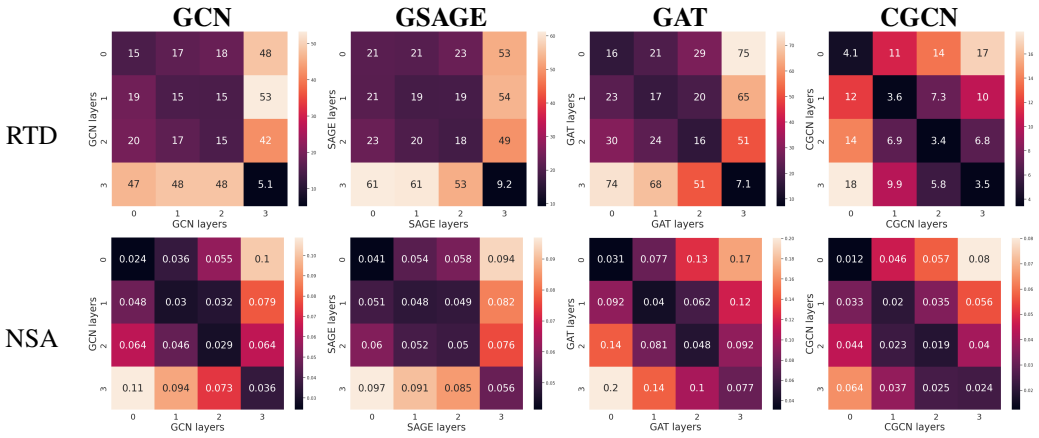
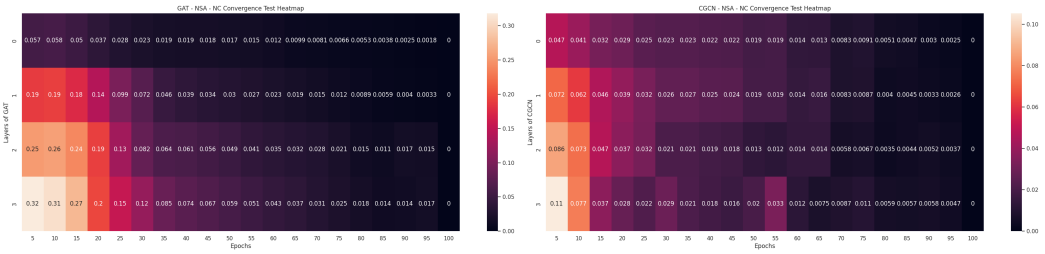


Figure 1: Sanity Tests for Node Classification (Amazon Computers Dataset). The heatmaps show layer-wise dissimilarity values for two different initialization of the same dataset on four different GNN architectures. The first row shows the RTD values and the second row shows the NSA values. NSA shows a stronger layer-wise correlation. Results for CKA' are in Appendix C

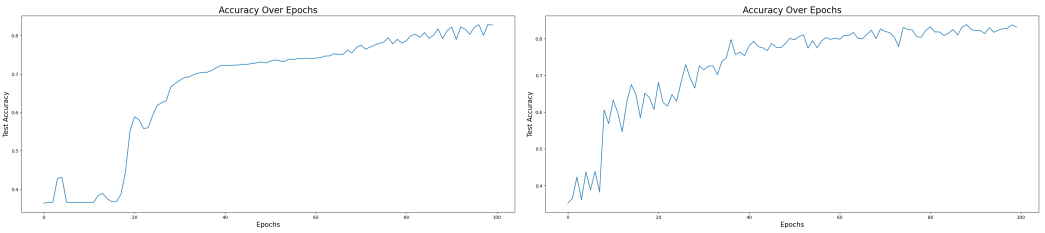
We begin with a basic sanity test for evaluating similarity metrics as suggested by Kornblith et al. (2019). When we have two networks that are structurally identical but trained from different initial conditions, we expect that, for each layer in one, the most similar layer in the other should be the corresponding layer that matches in structure. We perform tests on four different GNN architectures; Graph Convolution Networks (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), Graph Attention Networks (Veličković et al., 2018) and ClusterGCN (Chiang et al., 2019). We perform the tests when the models on trained on the task on Node Classification on the Amazon Computers dataset (Shchur et al., 2019) in Figure 1. We also have sanity tests for the models when they are trained on Link Prediction in Appendix D. We also showcase the results of CKA' and RTD to empirically demonstrate that NSA is more nuanced than RTD and CKA', and is capable of identifying patterns that are expected based on the design of the architectures. Both NSA and RTD compare dissimilarity while CKA compares similarity. To ensure ease of comparison between the metrics, throughout this paper we use $1 - \mathsf{CKA}$ to show the performance of CKA and label it CKA'. For further details on the architectures, model setup and hyperparameters, please refer to Appendix L.

3.7 CONVERGENCE TESTS



(a) GAT Epochwise Convergence Heatmap

(b) CGCN Epochwise Convergence Heatmap



(c) Test Accuracy

(d) Test Accuracy

Figure 2: Convergence Tests for Node Classification on the Amazon Computers Dataset. The representation for each layer is compared epochwise against that layer's final epoch representation. NSA's convergence corresponds strongly with the test accuracy convergence for both models

We examine epoch-wise convergence by comparing representations between the current and final epochs. Figure 2 illustrates our results for GAT and CGCN when trained on the Amazon Dataset on node classification. We observe that NSA's convergence corresponds with the test accuracy convergence of the model. We show results for other architectures on node classification in Figure 17 and showcase the results for link prediction in Figure 16

4 NSA AS A LOSS FUNCTION

Here, we establish the viability of NSA as a loss function and use this formulation to define an autoencoder NSA-AE. To establish the use of NSA as a loss function, we need to demonstrate its non-negativity (lemma 3), nullity (lemma 1) and continuity, and develop a differentiation scheme. Non-negativity and nullity are shown in Appendix A. The other two properties are proven below.

4.1 DIFFERENTIABILITY OF NSA

To derive the sub-gradient $\frac{\partial \mathsf{NSA}(X,Y)}{\partial x_i}$, we first define the following notation. Let $\mathbb{I} : \{T, F\} \to \{0, 1\}$ be a function that takes as input some condition and outputs 0 or 1 such that $\mathbb{I}(T) = 1$ and $\mathbb{I}(F) = 0$. Let $D_X = \max_{x \in X}(d(x, 0))$. Then it is easy to see that

$$\frac{\partial D_X}{\partial x_i} = \frac{x_i}{d(x_i, 0)} \mathbb{I}\{\arg\max_{x \in X}(d(x, 0)) = i\}.$$

Also, notice that $\frac{\partial d(x_i, x_j)}{x_i} = \frac{x_i - x_j}{d(x_i, x_j)}$, and that for $j \neq i$ and $k \neq i$, $\frac{\partial d(x_j, x_k)}{x_i} = 0$.

Let $m_X^{i,j} = \frac{d(x_i, x_j)}{D_X}$ and $m_Y^{i,j} = \frac{d(y_i, y_j)}{D_Y}$. Hence, combining, we get

$$\frac{\partial m_X^{j,k}}{\partial x_i} = \frac{\frac{\partial d(x_j, x_k)}{\partial x_i}}{D_X} - \frac{d(x_j, x_k) \frac{\partial D_X}{\partial x_i}}{(D_X)^2}, \text{ where these partial differentials are derived above.}$$

Lastly, notice that $\frac{\partial \left| m_X^{j,k} - m_Y^{j,k} \right|}{\partial m_X^{j,k}} = (-1)^{\mathbb{I}(m_Y^{j,k} > m_X^{j,k})}$.

Combining all the above, we get

$$\frac{\partial \mathsf{NSA}(X,Y)}{\partial x_i} = \frac{1}{N^2} \sum_{1 \le j,k \le N} \frac{\partial \left| m_X^{j,k} - m_Y^{j,k} \right|}{\partial m_X^{j,k}} \frac{\partial m_X^{j,k}}{\partial x_i} = \frac{1}{N^2} \sum_{1 \le j,k \le N} (-1)^{\mathbb{I}(m_Y^{j,k} > m_X^{j,k})} \frac{\partial m_X^{j,k}}{\partial x_i}.$$

Sub-gradients $\frac{\partial \mathsf{NSA}(X,Y)}{\partial y_i}$ can be similarly derived.

4.2 Continuity

We prove the continuity of NSA by showing that NSA is a composition of continuous functions (Carothers, 2000, Theorem 5.10). Let

$$f_{i,j}(X,Y) = \left| \frac{d(x_i, x_j)}{\max_{x \in X}(d(x,0))} - \frac{d(y_i, y_j)}{\max_{y \in Y}(d(y,0))} \right|.$$

$$\text{It's easy to see that} \qquad \mathsf{NSA}(X,Y) = \frac{1}{N^2} \sum_{1 \le i,j \le N} f_{i,j}(X,Y).$$

Since a sum of continuous functions is continuous, all we need to show is that $f_{i,j}(X,Y)$ is continuous for all $1 \le i,j \le N$. This is easy to see because $f_{i,j}(X,Y)$ can be seen as a composition of $d(\cdot,\cdot)$ and $\max(\cdot)$, along with the algebraic operations of subtraction, division and taking the absolute value. All the above operations are continuous everywhere[1], we get that their composition is also continuous everywhere. Hence, $\mathsf{NSA}(\cdot,\cdot)$ is continuous.

4.3 Defining NSA-AE: Autoencoder using NSA

To evaluate the efficacy of NSA as a structural discrepancy minimization metric, we take inspiration from TopoAE and RTD-AE, and use NSA as loss function in autoencoders for dimensionality reduction, thus defining NSA-AE. MDS and ISOMAP also aim to preserve pairwise distances but they do not scale well. Since NSA can be used in autoencoders with mini-batch training, NSA-AE runs almost as fast as a regular autoencoder. We compare the performance of NSA-AE against PCA, UMAP, a regular autoencoder, TopoAE and RTD-AE on four real world datasets.

The performance of NSA-AE is evaluated on the structural and topological similarity between the input data X and the latent data Z. In order to evaluate the performance, we use: (1) linear correlation of pairwise distances, (2) triplet distance ranking accuracy (Wang et al., 2021), (3) RTD, (4) triplet distance ranking accuracy between cluster centers, and (5) NSA. As seen in Table 1, NSA-AE has better correlation between the original data and the latent data compared to the other models. NSA-AE outperforms the other approaches on all metrics but RTD, where it ranks just below RTD-AE. NSA-AE achieves MSE and running times similar to a normal autoencoder while previous works are several times slower, as shown in Table 2 in the Appendix.

5 Analyzing Adversarial Attacks with NSA

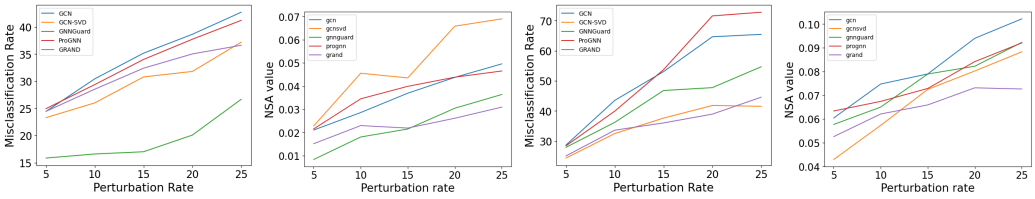
Addressing the issue of adversarial attacks in machine learning models, particularly in the context of Graph Neural Networks (GNNs), is of paramount importance. Adversarial attacks pose a significant threat in various domains, including social networks, recommendation systems, and cybersecurity, where GNNs are extensively employed. To fortify GNNs against adversarial challenges, the incorporation of a structure-preserving minimization term in the training process is a promising approach. Such a term enforces the preservation of key structural characteristics within the data, rendering the model less susceptible to adversarial perturbations. However, the efficacy of this approach greatly hinges on the availability of a suitable similarity metric capable of discerning even the subtlest perturbations in the representation space.

[1]Note that division is not continuous when the denominator goes to zero. Since, the denominators are $\max_{x \in X}(d(x,0))$ and $\max_{y \in Y}(d(y,0))$, as long as neither of the representations is all zero, the denominator does not go to zero.

Dataset	Method	Quality measure				
		L. C.	T. A.	RTD	T. A. C.C	NSA
MNIST	PCA	0.910	0.871 ± 0.008	6.69 ± 0.21	$\mathbf{0.986 \pm 0.117}$	0.0817 ± 0.0025
	UMAP	0.424	0.620 ± 0.013	18.06 ± 0.48	0.824 ± 0.381	0.2305 ± 0.0031
	AE	0.801	0.778 ± 0.007	7.47 ± 0.20	0.828 ± 0.377	0.0571 ± 0.0011
	TopoAE	0.765	0.771 ± 0.010	6.16 ± 0.23	0.886 ± 0.318	0.0477 ± 0.0011
	RTD-AE	0.837	0.811 ± 0.004	$\mathbf{4.26 \pm 0.14}$	0.842 ± 0.365	0.1694 ± 0.0024
	NSA-AE	**0.942**	$\mathbf{0.885 \pm 0.006}$	5.44 ± 0.12	0.944 ± 0.230	$\mathbf{0.0198 \pm 0.0001}$
F-MNIST	PCA	0.978	0.951 ± 0.006	5.91 ± 0.12	$\mathbf{1.000 \pm 0.0}$	0.1722 ± 0.0038
	UMAP	0.592	0.734 ± 0.012	12.16 ± 0.39	0.916 ± 0.277	0.1420 ± 0.0011
	AE	0.872	0.850 ± 0.008	5.60 ± 0.21	0.926 ± 0.262	0.0527 ± 0.0028
	TopoAE	0.875	0.854 ± 0.009	4.27 ± 0.15	0.946 ± 0.226	0.111 ± 0.0027
	RTD-AE	0.949	0.902 ± 0.004	$\mathbf{3.05 \pm 0.12}$	0.972 ± 0.165	0.0349 ± 0.0015
	NSA-AE	**0.987**	$\mathbf{0.952 \pm 0.002}$	4.11 ± 0.21	0.992 ± 0.089	$\mathbf{0.0091 \pm 0.0001}$
CIFAR-10	PCA	0.972	0.926 ± 0.009	4.99 ± 0.16	$\mathbf{0.994 \pm 0.077}$	0.1809 ± 0.0046
	UMAP	0.756	0.786 ± 0.010	12.21 ± 0.22	0.956 ± 0.205	0.1316 ± 0.0026
	AE	0.834	0.836 ± 0.006	4.07 ± 0.28	0.920 ± 0.271	0.0616 ± 0.0019
	TopoAE	0.889	0.854 ± 0.007	3.89 ± 0.11	0.942 ± 0.234	0.0625 ± 0.0014
	RTD-AE	0.971	0.922 ± 0.002	$\mathbf{2.95 \pm 0.08}$	0.976 ± 0.153	0.0113 ± 0.0003
	NSA-AE	**0.985**	$\mathbf{0.936 \pm 0.004}$	3.07 ± 0.11	0.984 ± 0.125	$\mathbf{0.0077 \pm 0.0001}$
COIL-20	PCA	**0.966**	0.932 ± 0.005	6.49 ± 0.23	$\mathbf{0.992 \pm 0.090}$	0.2204 ± 0.0
	UMAP	0.274	0.567 ± 0.016	15.50 ± 0.67	0.669 ± 0.471	0.1104 ± 0.0
	AE	0.850	0.836 ± 0.008	9.57 ± 0.27	0.889 ± 0.314	0.0758 ± 0.0
	TopoAE	0.804	0.805 ± 0.011	7.33 ± 0.21	0.885 ± 0.319	0.0676 ± 0.0
	RTD-AE	0.908	0.871 ± 0.005	$\mathbf{5.89 \pm 0.10}$	0.891 ± 0.311	0.0523 ± 0.0
	NSA-AE	0.955	$\mathbf{0.919 \pm 0.004}$	7.46 ± 0.23	0.939 ± 0.240	$\mathbf{0.0157 \pm 0.0}$

Table 1: Autoencoder results. NSA-AE outperforms or almost matches all other approaches on all the evaluation metrics. RTD-AE, which explicitly minimizes on RTD has a slightly lower RTD value while PCA has marginally higher Triplet Ranking Accuracy on Cluster Centers.

In this experimental study, we subject five distinct Graph Neural Network (GNN) architectures to both poisoning and evasion adversarial attacks using projected gradient descent (Xu et al., 2019). We use the regular GCN along with four robust GNN variants; SVD-GCN (Entezari et al., 2020), GNNGuard (Zhang & Zitnik, 2020), GRAND (Zhang & Zitnik, 2020) and ProGNN (Jin et al., 2020b). To assess the vulnerability of these architectures, we manipulated the initial adjacency matrices by introducing perturbations ranging from 5% to 25%. The objective was to gauge the impact of these perturbations on the misclassification rates of the GNN models and to see if NSA shows a strong correlation to the misclassification rates over different perturbation rates.



(a) (b) (c) (d)

Figure 3: Robustness tests with NSA. (a) Variation of Misclassification Rate against Data Perturbation Rate for GNN architectures under global evasion attack. (b) NSA against perturbation rate for GNN architectures under global evasion attack. (c) Variation of Misclassification Rate against Data Perturbation Rate for GNN architectures under global poisoning attack. (b) NSA against perturbation rate for GNN architectures under global poisoning attack

Remarkably, our findings revealed a compelling pattern: the variation of NSA over perturbation rate for all the architectures showed a distinct similarity to the trend of the misclassification rates

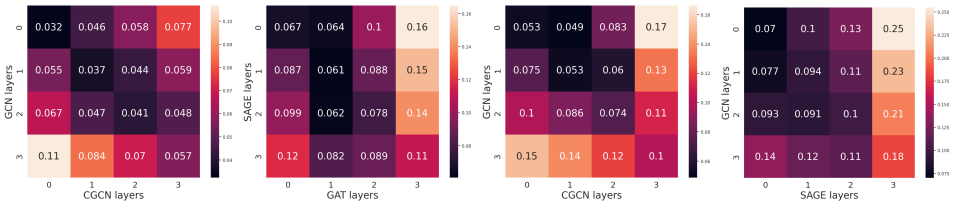
exhibited by the various GNN architectures. NSA also shows in Figure 3(a) that SVD-GCN's performance on global evasion attacks is much worse than the misclassification rate over various perturbation budgets. This is consistent with the failure of SVD-GCN on adaptive attacks as observed in (Mujkanovic et al., 2022). GNNGuard, emphasizing GNN architecture, and GRAND, focusing on training principles, consistently exhibit a strong performance in both attack scenarios. ProGNN performs on par with a normal GCN while SVD-GCN has the most erratic performance. Our ranking of GNNs is loosely consistent with the previous works that rank GNNs on robustness to adversarial attacks (Jin et al., 2020a; Mujkanovic et al., 2022). These results shed light on the intricate relationship between graph structure and model performance, offering insights into the effectiveness of our metric as a valuable tool for evaluating GNN resilience.

6 Are GNNs Task Specific Learners?

We use NSA to investigate whether representations produced by GNNs are specific to the downstream task. For this, we explored the similarity of representations of different GNN architectures when trained on the same task versus their similarity when trained on different downstream tasks.

6.1 Cross Architecture Tests on the Same Downstream Task

NSA



(a) GCN vs CGCN (b) GraphSAGE vs GAT (c) GCN vs CGCN (d) GCN vs GSAGE

Figure 4: Cross Architecture Tests using NSA on the Amazon Computers Dataset. (a) Layerwise NSA values between GCN and ClusterGCN on Node Classification (b) Layerwise NSA values between GraphSAGE and GAT on Node Classification (c) Layerwise NSA values between GCN and ClusterGCN on Link Prediction (d) Layerwise NSA values between GCN and GSAGE on Link Prediction. Similar architectures showcase a layerwise pattern when trained on the same task.

We tested layerwise representational similarity on the task of node classification (on the Amazon Computers Dataset) across different GNN architectures: GCN, ClusterGCN (CGCN), GraphSAGE, and GAT. Just like in the sanity tests, models that are similar in architecture or training paradigms have a higher layerwise similarity: this implies that the highest similarity is between GCN and ClusterGCN that differ only in their training paradigms. We also observe (a less pronounced) linear relationship between the other pairs of architectures. These results are shown in Figure 4.

6.2 Architecture Tests on Different Downstream Tasks

We conducted experiments with different downstream tasks to assess layerwise similarity between two models, both of which shared structural identity except for disparities in their final layers. Specifically, we employed node classification and link prediction as the two downstream tasks for our investigations. We present our findings in Figure 5. They reveal that there is little relationship across layers for any of the four architectures.

The results presented in this section indicate that that different GNN architectures have similar representation spaces when trained on the same downstream task and conversely, similar architectures have different representation spaces when trained on different downstream tasks. Extending this idea further, it should be possible to train Graph Neural Networks to conform to a task specific representation template. This structural template will be agnostic of GNN architectures and provide a high degree of functional similarity. If we train GNNs to minimize discrepancy loss with such a task specific template, we could train more directly and without adding downstream layers of tasks.

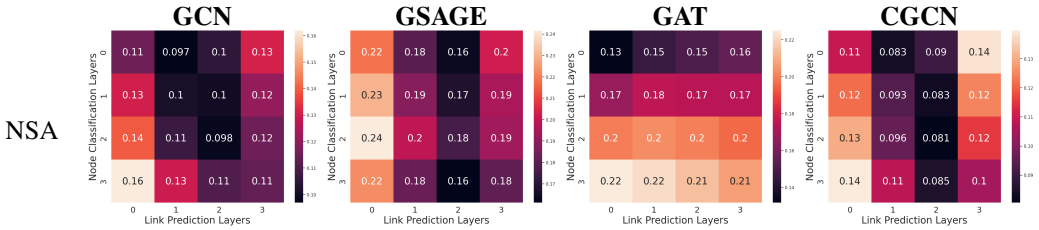


Figure 5: Effect of Downstream Task on Representations Achieved by the Same Architecture (Amazon Computers Dataset). The layerwise dissimilarity of four GNN architectures is compared on two different downstream tasks: Node Classification and Link Prediction. There is no observable correlation across layers suggesting that the GNNs generate different representation spaces for the same dataset on different downstream tasks.

7 Discussion and Conclusion

In conclusion, we have demonstrated that the proposed measure of NSA is simple, efficient, and useful across many aspects of analysis/synthesis of representation spaces: robustness across initializations, convergence across epochs, autoencoders, adversarial attacks, and effects of downstream tasks across architectures. Its simplicity should allow it to be useful in many applications and generalizations where scalability is desired. We point out some such avenues next.

One possibility for future research would be to combine NSA with measures of intrinsic dimensionality (Camastra & Staiano, 2016; Campadelli et al., 2015). This is motivated by the observation that real-world data presented in a high-dimensional space usually lies along a manifold of much lower dimension (Goodfellow et al., 2016). A challenge here would be ensuring that the resulting measure does not add much of a computational overhead to NSA.

Another potential modification to NSA involves squaring the pairwise differences instead of employing the L1 norm, a variation that, in our initial experiments, yielded performance similar to the L1 variant but necessitated a distinct set of hyperparameters for optimization. Additionally, the choice of the distance metric and the normalization term in NSA warrant further consideration.

Empirical findings by Kornblith et al. (2019) have indicated minimal improvement with the RBF variant of the Centered Kernel Alignment (CKA) over the linear kernel CKA. However, it is crucial to recognize that in high-dimensional spaces, the Euclidean distance metric may not be particularly effective, as highlighted by theoretical works such as Bellman's "The Curse of Dimensionality" (Bellman, 1961). Despite this, Euclidean distance remains prevalent in various domains, including contrastive (Chen et al., 2020) or triplet losses (Schroff et al., 2015), style transfer (Johnson et al., 2016) and similarity indices, often yielding successful empirical results. Our future investigations intend to explore alternative distance measures, potentially considering non-linear options such as geodesic distance and assessing their viability.

It is conceivable to extend the concept of task-specific templates, as discussed in Section 6, to scenarios where domain-specific information guides the embedding of objects. Take, for instance, the context of signed networks, where it is established that pairs with positive connections should exhibit proximity, while those with negative connections should be positioned distantly. Disconnected ('dont care') pairs would ideally find themselves in an intermediate position. The challenge lies in specifying such embeddings in a manner that allows NSA to optimize them directly, thereby bypassing the need for intricate embedding methods as proposed by Huang et al. (2022). This opens up a fascinating avenue for exploration, as we seek innovative approaches to articulate and optimize these embedding specifications within the context of NSA.

Finally, on the question of structural similarity and functional similarity, Davari et al. (2023) highlight CKA's susceptibility to subset translations and situations where CKA changes while functional behavior remains consistent. While NSA is more tuned to functional similarity, we plan to carry out a similar analysis in the future.

8 REPRODUCIBILITY STATEMENT

Our code is anonymously available at https://anonymous.4open.science/r/NSA. The code includes notebooks with instructions to reproduce all the experiments presented in this paper. To reproduce Table 1 and Table 2 you will access the NSA_AE folder. The hyperparameter setup to reproduce the autoencoder results are in Table 3. To reproduce all the heatmaps in the paper, you will need to access the GNN_analysis folder. The hyperparameter setup to reproduce these results is given in Table 4. To reproduce the results in Section 5, you will access the Adversarial Analysis folder. No hyperparameter setup is necessary to run the notebooks in this section.

REFERENCES

Serguei Barannikov, Ilya Trofimov, Nikita Balabin, and Evgeny Burnaev. Representation topology divergence: A method for comparing neural network representations. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 1607–1626. PMLR, 17–23 Jul 2022.

Richard E. Bellman. Curse of dimensionality. *Mathematical Programming*, 1(1):19–32, 1961.

Francesco Camastra and Antonino Staiano. Intrinsic dimension estimation: Advances and open problems. *Information Sciences*, 328:26–41, 2016.

Paola Campadelli, Elena Casiraghi, Claudio Ceruti, and Alessandro Rozza. Intrinsic dimension estimation: Relevant techniques and a benchmark framework. *Mathematical Problems in Engineering*, 2015:1–21, 2015.

N. L. Carothers. *Real analysis /.* Cambridge University Press, Cambridge :, 2000. Includes bibliographical references and index.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 2020. URL http://proceedings.mlr.press/v119/chen20j.html.

Zuohui Chen. Revisit similarity of neural network representations from graph perspective. 2022. URL https://api.semanticscholar.org/CorpusID:249147422.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pp. 257–266, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330925. URL https://doi.org/10.1145/3292500.3330925.

Arlei Lopes da Silva, Furkan Kocayusufoglu, Saber Jafarpour, Francesco Bullo, Ananthram Swami, and Ambuj Singh. Combining physics and machine learning for network flow estimation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=l0V53bErniB.

MohammadReza Davari, Stefan Horoi, Amine Natik, Guillaume Lajoie, Guy Wolf, and Eugene Belilovsky. Reliability of CKA as a similarity measure in deep learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=8HRvyxc606.

Frances Ding, Jean-Stanislas Denain, and Jacob Steinhardt. Grounding representation similarity through statistical testing. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=_kwj6V53ZqB.

Negin Entezari, Saba A. Al-Sayouri, Amirali Darvishzadeh, and Evangelos E. Papalexakis. All you need is low (rank): Defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, pp. 169–177, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368223. doi: 10.1145/3336191.3371789. URL https://doi.org/10.1145/3336191.3371789.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. 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. 1126–1135. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. Von 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_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. doi: 10.1126/science.1127647. URL https://www.science.org/doi/abs/10.1126/science.1127647.

Zexi Huang, Arlei Silva, and Ambuj Singh. Pole: Polarized embedding for signed networks. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM '22, pp. 390–400, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391320. doi: 10.1145/3488560.3498454. URL https://doi.org/10.1145/3488560.3498454.

Zexi Huang, Mert Kosan, Sourav Medya, Sayan Ranu, and Ambuj Singh. Global counterfactual explainer for graph neural networks. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pp. 141–149, 2023.

Wei Jin, Yaxin Li, Han Xu, Yiqi Wang, and Jiliang Tang. Adversarial attacks and defenses on graphs: A review and empirical study. *CoRR*, abs/2003.00653, 2020a. URL https://arxiv.org/abs/2003.00653.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, pp. 66–74, New York, NY, USA, 2020b. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3403049. URL https://doi.org/10.1145/3394486.3403049.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (eds.), *Computer Vision – ECCV 2016*, pp. 694–711, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46475-6.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

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.

Max Klabunde, Tobias Schumacher, Markus Strohmaier, and Florian Lemmerich. Similarity of neural network models: A survey of functional and representational measures. *arXiv preprint arXiv:2305.06329*, 2023.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov

(eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 2019. URL http://proceedings.mlr.press/v97/kornblith19a.html.

Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. Representational similarity analysis - connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2, 2008. ISSN 1662-5137. doi: 10.3389/neuro.06.004.2008. URL https://www.frontiersin.org/articles/10.3389/neuro.06.004.2008.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2010.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018. doi: 10.21105/joss.00861. URL https://doi.org/10.21105/joss.00861.

Michael Moor, Max Horn, Bastian Rieck, and Karsten Borgwardt. Topological autoencoders. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7045–7054. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/moor20a.html.

Ari S. Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 5732–5741, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/a7a3d70c6d17a73140918996d03c014f-Abstract.html.

Felix Mujkanovic, Simon Geisler, Stephan Günnemann, and Aleksandar Bojchevski. Are defenses for graph neural networks robust? In *Neural Information Processing Systems, NeurIPS*, 2022.

Sameer A Nene, Shree K Nayar, and Hiroshi Murase. Columbia object image library (coil-20). In *Technical report CUCS-005-96*, 1996.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6078–6087, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Rishabh Ranjan, Siddharth Grover, Sourav Medya, Venkatesan Chakaravarthy, Yogish Sabharwal, and Sayan Ranu. Greed: A neural framework for learning graph distance functions. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, November 29-Decemer 1, 2022*, 2022.

Marco Rudolph, Bastian Wandt, and Bodo Rosenhahn. Structuring autoencoders. *CoRR*, abs/1908.02626, 2019. URL http://arxiv.org/abs/1908.02626.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015. doi: 10.1109/cvpr.2015.7298682. URL https://doi.org/10.1109%2Fcvpr.2015.7298682.

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

Gábor J. Székely, Maria L. Rizzo, and Nail K. Bakirov. Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, 35(6):2769 – 2794, 2007. doi: 10.1214/009053607000000505. URL https://doi.org/10.1214/009053607000000505.

Ilya Trofimov, Daniil Cherniavskii, Eduard Tulchinskii, Nikita Balabin, Evgeny Burnaev, and Serguei Barannikov. Learning topology-preserving data representations. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lIu-ixf-Tzf.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik. Understanding how dimension reduction tools work: An empirical approach to deciphering t-sne, umap, trimap, and pacmap for data visualization. *Journal of Machine Learning Research*, 22(201):1–73, 2021. URL http://jmlr.org/papers/v22/20-1061.html.

Alex H Williams, Erin Kunz, Simon Kornblith, and Scott Linderman. Generalized shape metrics on neural representations. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=L9JM-pxQOl.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596, 2019. URL http://arxiv.org/abs/1901.00596.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 3961–3967. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/550. URL https://doi.org/10.24963/ijcai.2019/550.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method, 2020.

Xiang Zhang and Marinka Zitnik. Gnnguard: Defending graph neural networks against adversarial attacks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9263–9275. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/690d83983a63aa1818423fd6edd3bfdb-Paper.pdf.

Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Trans. on Knowl. and Data Eng.*, 34(1):249–270, jan 2022. ISSN 1041-4347. doi: 10.1109/TKDE.2020.2981333. URL https://doi.org/10.1109/TKDE.2020.2981333.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020. ISSN 2666-6510. doi: https://doi.org/10.1016/j.aiopen.2021.01.001. URL https://www.sciencedirect.com/science/article/pii/S2666651021000012.

A PROOFS FOR NSA AS A PSEUDOMETRIC

A.1 $\mathsf{NSA}(X, X) = 0$

Lemma 1 (Identity)**.** *Let X be a point cloud over some space, then* $\mathsf{NSA}(X, X) = 0$.

Proof. Let $X = \{x_1, \ldots, x_N\}$. Let $\max_{x \in X}(d(x, 0)) = D$. Then

$$NSA(X, X) = \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \left| \frac{d(x_i, x_j)}{D} - \frac{d(x_i, x_j)}{D} \right|.$$

Simplifying, we get

$$NSA(X, X) = \frac{1}{N^2 D} \sum_{1 \leq i,j \leq N} |d(x_i, x_j) - d(x_i, x_j)| = 0.$$

□

Lemma 2 (Symmetry)**.** *Let X, Y be two point clouds of the same size, then* $\mathsf{NSA}(X, Y) = \mathsf{NSA}(Y, X)$.

Proof. Let $X = \{x_1, \ldots, x_N\}$ and $Y = \{y_1, \ldots, y_N\}$. Let $\max_{x \in X}(d(x, 0)) = D_X$ and $\max_{y \in Y}(d(y, 0)) = D_Y$. Then

$$\mathsf{NSA}(X, Y) = \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \left| \frac{d(x_i, x_j)}{D_X} - \frac{d(y_i, y_j)}{D_Y} \right|.$$

Since $|a - b| = |b - a|$,

$$\mathsf{NSA}(X, Y) = \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \left| \frac{d(y_i, y_j)}{D_Y} - \frac{d(x_i, x_j)}{D_X} \right|.$$

Hence, $\mathsf{NSA}(X, Y) = \mathsf{NSA}(Y, X)$. □

Lemma 3 (Non-negativity)**.** *Let X, Y be two point clouds of the same size, then* $\mathsf{NSA}(X, Y) \geq 0$.

Proof. Let $X = \{x_1, \ldots, x_N\}$ and $Y = \{y_1, \ldots, y_N\}$. Let $\max_{x \in X}(d(x, 0)) = D_X$ and $\max_{y \in Y}(d(y, 0)) = D_Y$. Then

$$\mathsf{NSA}(X, Y) = \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \left| \frac{d(x_i, x_j)}{D_X} - \frac{d(y_i, y_j)}{D_Y} \right|.$$

Since, for all i, j,

$$\left| \frac{d(x_i, x_j)}{D_X} - \frac{d(y_i, y_j)}{D_Y} \right| \geq 0,$$

we get $\mathsf{NSA}(X, Y) \geq 0$. □

Lemma 4 (Triangle inequality)**.** *Let X, Y and Z be three point clouds of the same size, then* $\mathsf{NSA}(X, Z) \leq \mathsf{NSA}(X, Y) + \mathsf{NSA}(Y, Z)$.

Proof. Let $X = \{x_1, \ldots, x_N\}$, $Y = \{y_1, \ldots, y_N\}$ and $Z = \{z_1, \ldots, z_N\}$. Let $\max_{x \in X}(d(x, 0)) = D_X$, $\max_{y \in Y}(d(y, 0)) = D_Y$ and $\max_{z \in Z}(d(z, 0)) = D_Z$. Then

$$\mathsf{NSA}(X, Z) = \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \left| \frac{d(x_i, x_j)}{D_X} - \frac{d(z_i, z_j)}{D_Z} \right|.$$

For each i, j, add and subtract $\frac{d(y_i, y_j)}{D_Y}$, then

$$\mathsf{NSA}(X, Z) = \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \left| \left(\frac{d(x_i, x_j)}{D_X} - \frac{d(y_i, y_j)}{D_Y} \right) + \left(\frac{d(y_i, y_j)}{D_Y} - \frac{d(z_i, z_j)}{D_Z} \right) \right|.$$

Since $|a+b| \leq |a| + |b|$,

$$\mathsf{NSA}(X,Z) \leq \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \left(\left| \frac{d(x_i,x_j)}{D_X} - \frac{d(y_i,y_j)}{D_Y} \right| + \left| \frac{d(y_i,y_j)}{D_Y} - \frac{d(z_i,z_j)}{D_Z} \right| \right).$$

Hence,

$$\mathsf{NSA}(X,Z) \leq \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \left| \frac{d(x_i,x_j)}{D_X} - \frac{d(y_i,y_j)}{D_Y} \right| + \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \left| \frac{d(y_i,y_j)}{D_Y} - \frac{d(z_i,z_j)}{D_Z} \right|,$$

or $\mathsf{NSA}(X,Z) \leq \mathsf{NSA}(X,Y) + \mathsf{NSA}(Y,Z)$. □

From Lemma 1, Lemma 2, Lemma 3 and Lemma 4, we see that NSA is a psuedometric over the space of point clouds.

B Proofs for NSA as a similarity metric

B.1 Invariance to Isotropic Scaling

Lemma 5 (Invariance to Isotropic scaling in the first coordinate)**.** *Let X and Y be two point clouds of the same size. Let $c \in \mathbf{R}$ and $c \neq 0$, X_c be the point cloud with each point in X scaled by a factor of c, then $\mathsf{NSA}(X,Y) = \mathsf{NSA}(X_c,Y)$.*

Proof. Let $X = \{x_1, \ldots, x_N\}$ and $Y = \{y_1, \ldots, y_N\}$, then $X_c = \{cx_1, \ldots, cx_N\}$. Let $\max_{x \in X}(d(x,0)) = D_X$, $\max_{y \in Y}(d(y,0)) = D_Y$ and $\max_{x \in X_c}(d(x,0)) = D_{X_c}$. Since, each point in X_c is the c times each point in X. Then, we can write $D_{X_c} = \max_{x \in X}(d(cx,0))$. Since, for any two points, x_1 and x_2, $d(cx_1,cx_2) = |c|d(x_1,x_2)$, then $D_{X_c} = |c| \max_{x \in X}(d(x,0))$. Hence, $D_{x_c} = |c|D_X$. From the definition of NSA,

$$\mathsf{NSA}(X_c,Y) = \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \left| \left(\frac{d(cx_i,cx_j)}{D_{X_c}} - \frac{d(y_i,y_j)}{D_Y} \right) \right|.$$

Again, using $d(cx_i,cx_j) = |c|d(x_i,x_j)$ and $D_{X_c} = |c|D_X$,

$$\mathsf{NSA}(X_c,Y) = \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \left| \left(\frac{|c|d(x_i,x_j)}{|c|D_X} - \frac{d(y_i,y_j)}{D_Y} \right) \right|.$$

Simplifying,

$$\mathsf{NSA}(X_c,Y) = \frac{1}{N^2} \sum_{1 \leq i,j \leq N} \left| \left(\frac{d(x_i,x_j)}{D_X} - \frac{d(y_i,y_j)}{D_Y} \right) \right|.$$

Hence, $\mathsf{NSA}(X_c,Y) = \mathsf{NSA}(X,Y)$. □

By Lemma 2, we can see that this gives us invariance to isotropic scaling in the second coordinate as well.

Lemma 6 (Invariance to Isotropic scaling in the second coordinate)**.** *Let X and Y be two point clouds of the same size. Let $c \in \mathbf{R}$ and $c \neq 0$, Y_c be the point cloud with each point in Y scaled by a factor of c, then $\mathsf{NSA}(X,Y) = \mathsf{NSA}(X,Y_c)$.*

Combining Lemma 5 and $Lemma$ 6, we get the required lemma.

Lemma 7 (Invariance to Isotropic scaling)**.** *Let X and Y be two point clouds of the same size. Let $c_1, c_2 \in \mathbf{R}$, $c_1 \neq 0$ and $c_2 \neq 0$, X_{c_1} be the point cloud with each point in X scaled by a factor of c_1 and Y_{c_2} be the point cloud with each point in Y scaled by a factor of c_2, then $\mathsf{NSA}(X,Y) = \mathsf{NSA}(X_{c_1},Y_{c_2})$.*

B.2 Invariance to Orthogonal transformation

Lemma 8 (Invariance to Orthogonal transformation in the first coordinate)**.** *Let X and Y be two point clouds of the same size. Let U be an orthogonal transformation on the point space of X, X_U be the point cloud with each point in X transformed using U, then* $\mathsf{NSA}(X, Y) = \mathsf{NSA}(X_U, Y)$.

Proof. Let $X = \{x_1, \ldots, x_N\}$ and $Y = \{y_1, \ldots, y_N\}$, then $X_U = \{Ux_1, \ldots, Ux_N\}$. Let $\max_{x \in X}(d(x, 0)) = D_X$, $\max_{y \in Y}(d(y, 0)) = D_Y$ and $\max_{x \in X_U}(d(x, 0)) = D_{X_U}$. Since, each point in X_U is the U times each point in X. Then, we can write $D_{X_U} = \max_{x \in X}(d(Ux, 0))$. Since, for any two points, x_1 and x_2, $d(Ux_1, x_2) = d(x_1, U^T x_2)$, and $U^T 0 = 0$, then $D_{X_c} = \max_{x \in X}(d(x, 0))$. Hence, $D_{x_c} = D_X$. From the definition of NSA,

$$\mathsf{NSA}(X_U, Y) = \frac{1}{N^2} \sum_{1 \le i,j \le N} \left| \left(\frac{d(Ux_i, Ux_j)}{D_{X_c}} - \frac{d(y_i, y_j)}{D_Y} \right) \right|.$$

Again, using $d(Ux_i, Ux_j) = d(x_i, U^T U x_j)$, and $U^T U = I$,

$$\mathsf{NSA}(X_U, Y) = \frac{1}{N^2} \sum_{1 \le i,j \le N} \left| \left(\frac{d(x_i, x_j)}{D_X} - \frac{d(y_i, y_j)}{D_Y} \right) \right|.$$

Hence, $\mathsf{NSA}(X_U, Y) = \mathsf{NSA}(X, Y)$. □

By Lemma 2, we get invariance to orthogonal transformation in the second coordinate too and combining, we get the required lemma.

Lemma 9 (Invariance to Orthogonal transformation)**.** *Let X and Y be two point clouds of the same size. Let U_1 be an orthogonal transformation on the point space of X, X_{U_1} be the point cloud with each point in X transformed using U_1. Similarly, let U_2 be an orthogonal transformation on the point space of Y, Y_{U_2} be the point cloud with each point in Y transformed using U_2, then* $\mathsf{NSA}(X, Y) = \mathsf{NSA}(X_{U_1}, Y_{U_2})$.

B.3 Not Invariant under Invertible Linear Transformation (ILT)

We can easily see this with a counter example. Let $X = \{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$. Let $Y = \{(1, 1, 0), (1, 0, 1), (0, 1, 1)\}$. Define A to be an invertible linear map such that $A(1, 0, 0) = (2, 0, 0)$, $A(0, 1, 0) = (0, 1, 0)$ and $A(0, 0, 1) = (0, 0, 1)$. Let X_A be the point cloud with each point in X transformed using A. Then from some calculation, we can see that $\mathsf{NSA}(X, Y) = 0$ but $\mathsf{NSA}(X_A, Y) = \frac{1}{9}$. Hence, we can see that NSA is not invariant under Invertible Linear Transformations.

C Sanity Tests for Node Classification

We show the results for node classification across GNN architectures for CKA'. We also show the heatmaps for node classification on the Flickr Dataset for NSA. The heatmaps for the sanity tests on the task of Node Classification are given in Figure 6

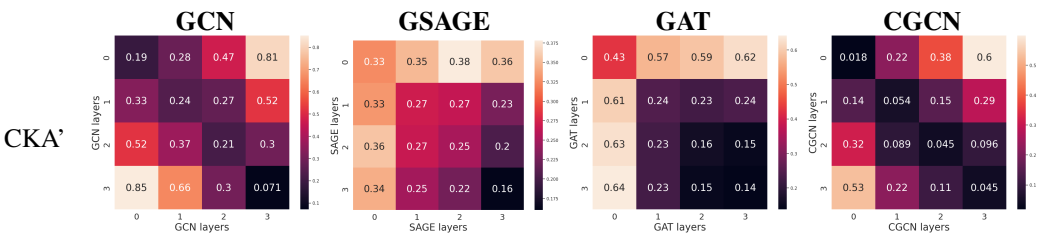


Figure 6: Sanity Tests for Node Classification for CKA' on Amazon Computers Dataset

D SANITY TESTS FOR LINK PREDICTION

We show the results for link prediction across GNN architectures for all three similarity metrics. We observe that although all 3 metrics pass the sanity test, NSA shows the best gradient in similarity across adjacent layers of some models like GAT and CGCN. The heatmaps for the sanity tests for Link Prediction are given in Figure 7

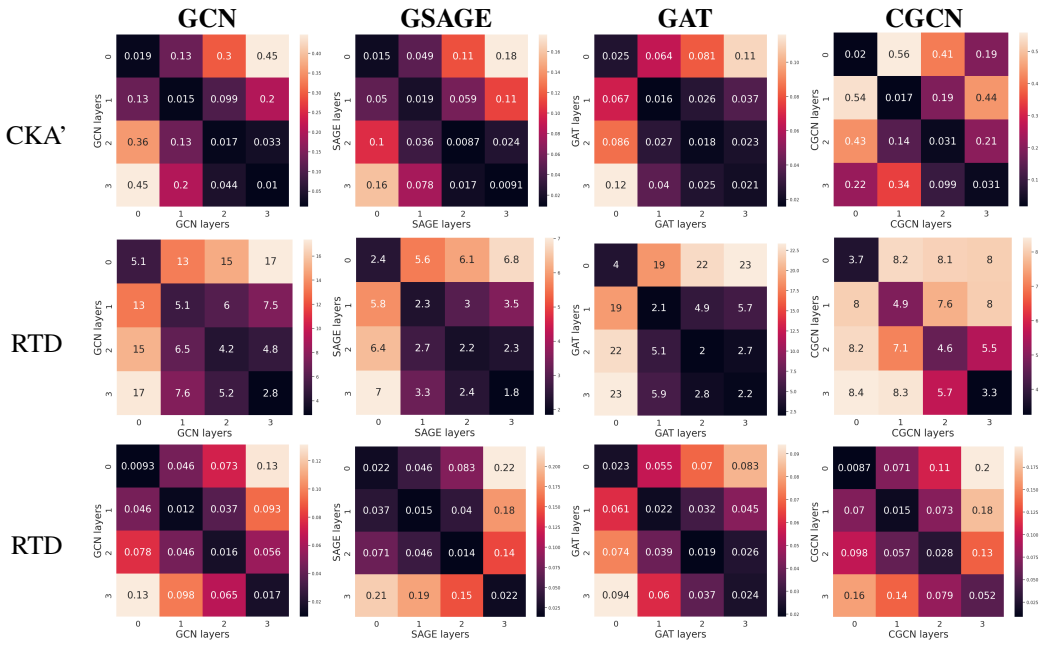


Figure 7: Sanity Tests for Link Prediction

E SANITY TESTS ON THE FLICKR DATASET

A node classification sanity test is performed on the Flickr (Zeng et al., 2020) dataset to prove that NSA works across any dataset. We show the results for NSA only in Figure 8

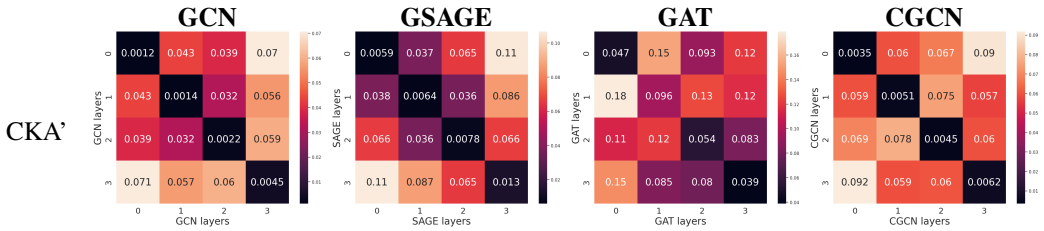


Figure 8: Sanity Test with the Flickr Dataset using NSA

F CROSS DOWNSTREAM TASK FOR CKA′ AND RTD

We show the performance of CKA' and RTD when we test how GNN architectures compare across different downstream tasks. Similar to NSA, CKA' and RTD fail to show any noticeable pattern across layers. The heatmaps for cross downstream tasks with CKA' and RTD are given in Figure 9

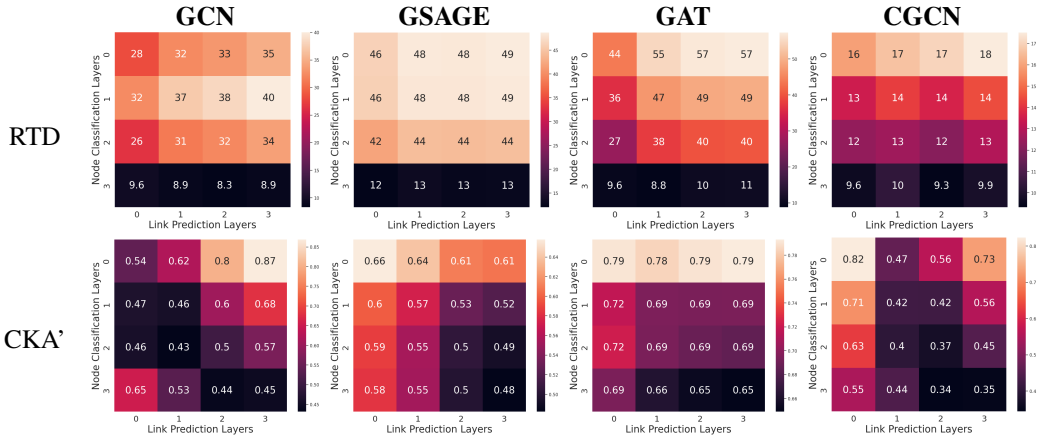
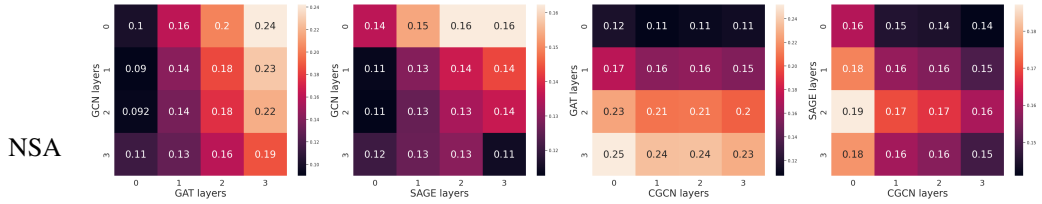


Figure 9: Cross Downstream Task Tests CKA′ and RTD

G Cross Architecture Tests

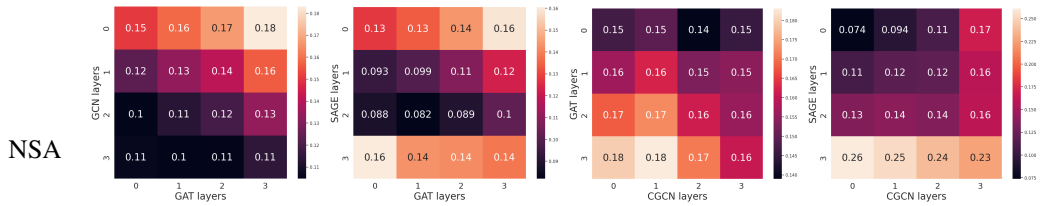
G.1 Cross Architecture Tests with NSA

The cross architecture test results for NSA on link prediction and node classification for the architectures not shown in Figure 4 are given in Figure 10 and Figure 11. These are architectures with low degree of similarity hence the we do not observe a strong linear relationship across layers with NSA.



(a) GCN vs GAT (b) GCN vs GSAGE (c) GAT vs ClusterGCN (d) GSAGE vs CGCN

Figure 10: Cross Architecture Tests using Normalized Space Alignment for Node Classification for the remaining architectures



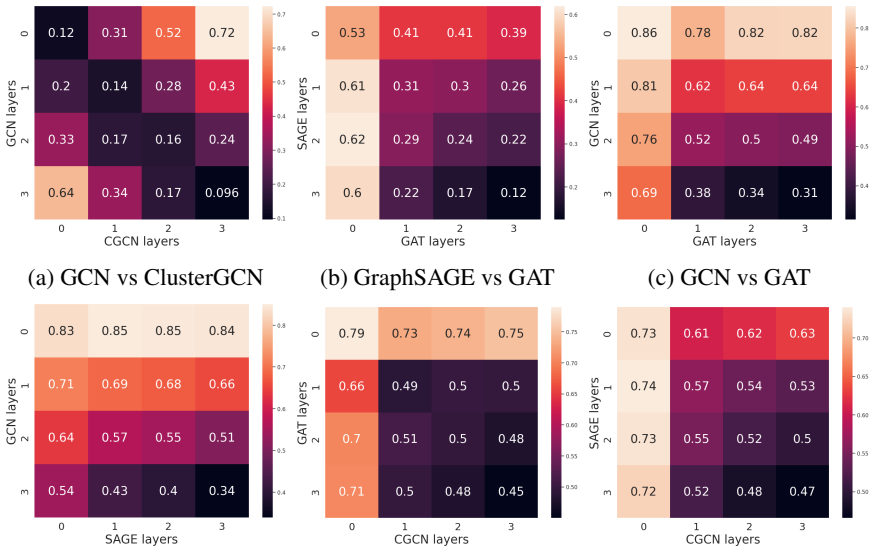
(a) GCN vs GAT (b) GraphSAGE vs GAT (c) GAT vs ClusterGCN (d) GSAGE vs CGCN

Figure 11: Cross Architecture Tests using Normalized Space Alignment for Link Prediction for the remaining four architectures

G.2 Cross Architecture Tests with RTD and CKA′ on Node Classification

The cross architecture test results for CKA' and RTD on Node Classification are shown in Figure 12 and Figure 13

Centered Kernel Alignment'

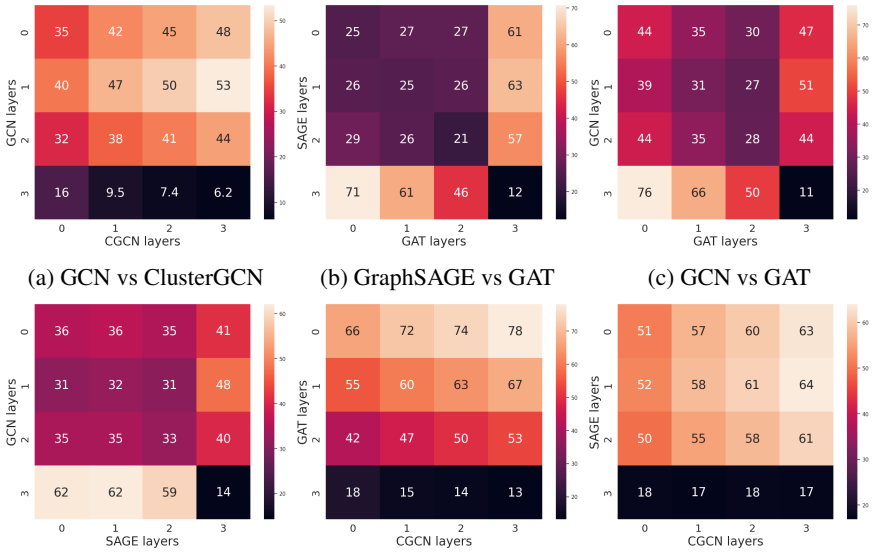


(a) GCN vs ClusterGCN (b) GraphSAGE vs GAT (c) GCN vs GAT

(d) GCN vs GraphSAGE (e) GAT vs ClusterGCN (f) GSAGE vs CGCN

Figure 12: Cross Architecture Tests using Centered Kernel Alignment' for Node Classification

Representation Topology Divergence



(a) GCN vs ClusterGCN (b) GraphSAGE vs GAT (c) GCN vs GAT

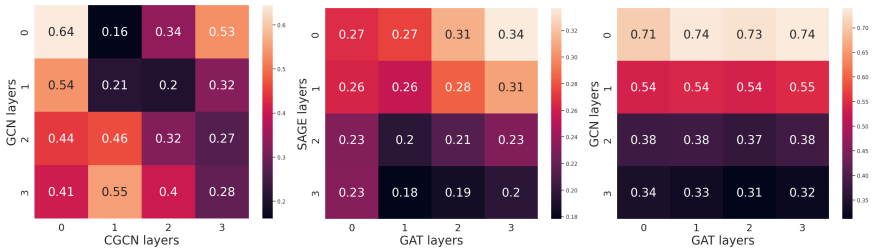
(d) GCN vs GraphSAGE (e) GAT vs ClusterGCN (f) GSAGE vs CGCN

Figure 13: Cross Architecture Tests using Representation Topology Divergence for Node Classification

G.3 Cross Architecture Tests with RTD and CKA' on Link Prediction

The cross architecture test results for CKA' and RTD on Link Prediction are shown in Figure 14 and Figure 15. We observe that CKA' and RTD do not show a layerwise pattern as well as NSA does. We can observe that RTD manages to capture the low dissimilarity between corresponding layers of GCN and ClusterGCN and for some other models but CKA' fails to capture the layerwise similarity for any architecture combination.

Centered Kernel Alignment'

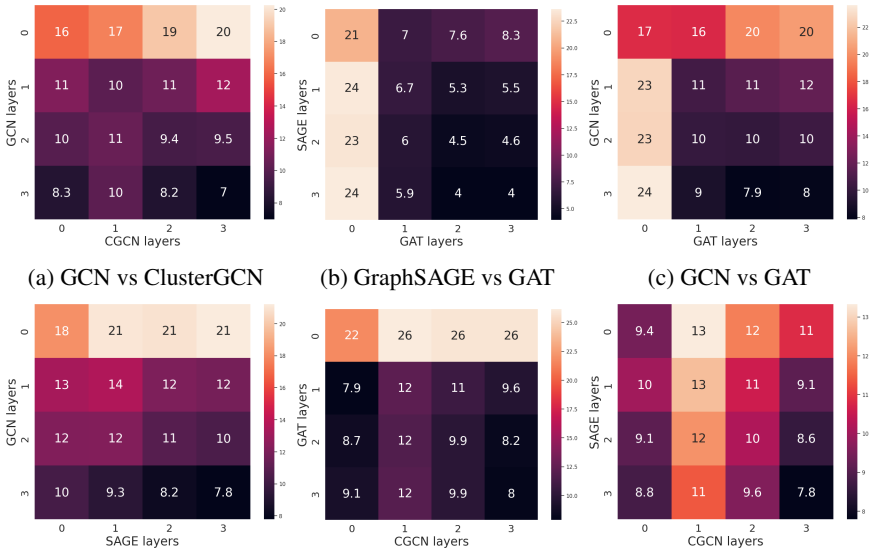


(a) GCN vs ClusterGCN (b) GraphSAGE vs GAT (c) GCN vs GAT

(d) GCN vs GraphSAGE (e) GAT vs ClusterGCN (f) GSAGE vs CGCN

Figure 14: Cross Architecture Tests using Centered Kernel Alignment' for Link Prediction

Representation Topology Divergence



(a) GCN vs ClusterGCN (b) GraphSAGE vs GAT (c) GCN vs GAT

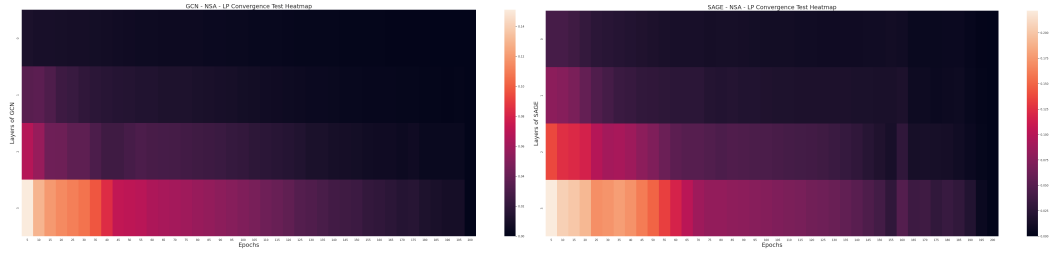
(d) GCN vs GraphSAGE (e) GAT vs ClusterGCN (f) GSAGE vs CGCN

Figure 15: Cross Architecture Tests using Representation Topology Divergence for Link Prediction

H CONVERGENCE TESTS

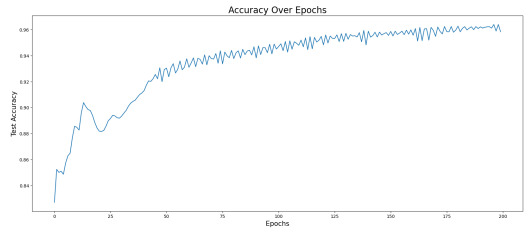
I LINK PREDICTION

The convergence tests for Link Prediction across all four architectures are given in Figure 16. Link Prediction models were trained for 200 epochs and show similar patterns to the ones observed with node classification. NSA also captures the variations in the final layer of ClusterGCN causing oscillation in the test accuracy in Figure 16 (f). All convergence tests are performed with NSA only.

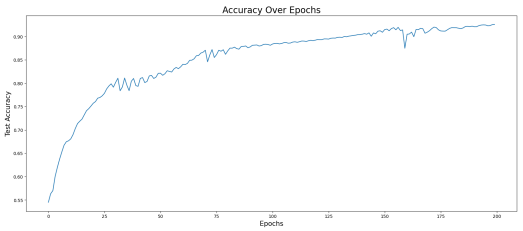


(a) GCN Epochwise Convergence Heatmap

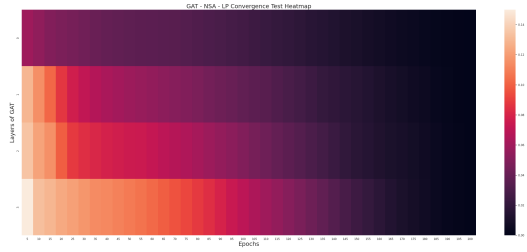
(b) SAGE Epochwise Convergence Heatmap



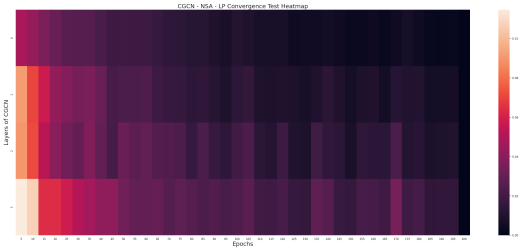
(c) Test Accuracy



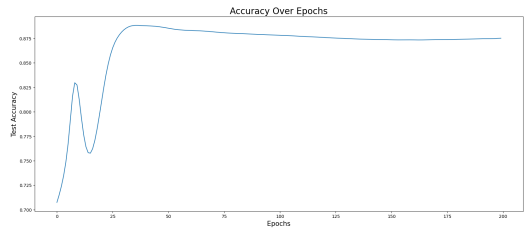
(d) Test Accuracy



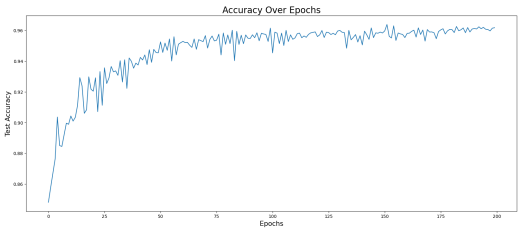
(e) GAT Epochwise Convergence Heatmap



(f) CGCN Epochwise Convergence Heatmap



(g) Test Accuracy

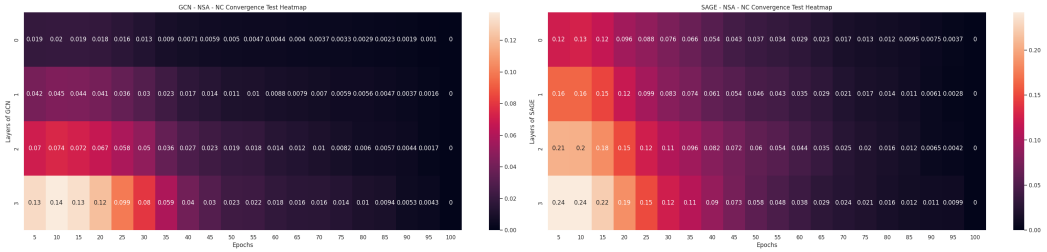


(h) Test Accuracy

Figure 16: Convergence Tests on Link Prediction

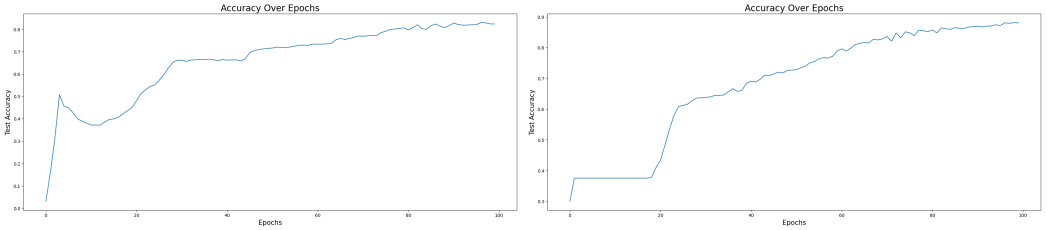
I.1 NODE CLASSIFICATION

The convergence tests on Node Classification for Graph Attention Network and ClusterGCN are given in Figure 17. Just like GCN and GraphSAGE, we observe a strong correlation between layer-wise NSA convergence and test accuracy for Graph Attention Network and ClusterGCN



(a) GCN Epochwise Convergence Heatmap

(b) SAGE Epochwise Convergence Heatmap



(c) Test Accuracy

(d) Test Accuracy

Figure 17: Convergence Tests on Node Classification

J Running Time and Reconstruction Loss for Autoencoder

We show the results in Table 2

Dataset	Method	Metric		
		Time per epoch	Train MSE	Test MSE
MNIST	AE	2.344	7.64e-3	8.62e-3
	TopoAE	39.168	6.89e-3	8.16e-3
	RTD-AE	51.608	9.36e-3	1.07e-2
	NSA-AE	5.816	8.75e-3	1.00e-2
F-MNIST	AE	6.436	8.99e-03	9.79e-03
	TopoAE	37.26	8.94-03	9.83e-03
	RTD-AE	59.94	1.12e-02	1.24e-02
	NSA-AE	5.764	9.49e-03	1.04e-02
CIFAR-10	AE	5.16	1.56e-02	1.65e-02
	TopoAE	58.664	1.54e-02	1.68e-02
	RTD-AE	56.172	1.60e-02	1.91e-02
	NSA-AE	6.996	1.58e-02	1.71e-02
COIL-20	AE	2.012	1.67e-02	-
	TopoAE	4.876	1.09e-02	-
	RTD-AE	16.404	1.90e-02	-
	NSA-AE	8.716	1.80e-02	-

Table 2: Reconstruction Loss and Time Per Epoch for different Autoencoder architectures. All architectures were trained for 250 epochs with the auxiliary loss (TopoLoss, RTD, NSA) kicking in after 60 epochs.

K AUTOENCODER HYPERPARAMETERS

The hyperparameter setup for all the autoencoder architectures is detailed in Table 3. All autoencoder architectures are trained using pytorch-lightning and the input data is normalized using a MinMaxScaler.

Dataset Name	Batch Size	LR	Hidden Dim	Layers	Epochs	Metric Start Epoch
MNIST	256	10^-4	512	3	250	60
F-MNIST	256	10^-4	512	3	250	60
CIFAR-10	256	10^-4	512	3	250	60
COIL-20	256	10^-4	512	3	250	60

Table 3: Autoencoder Hyperparameters. All four architectures used the same hyperparameters. To ensure similarity of testing conditions we replicate the hyperparameter setup from Trofimov et al. (2023)

L GNN HYPERPARAMETERS

The hyperparameter setup for all the GNN architectures is detailed in Table 4

Architecture	Layers	Hidden Dim	LR	Epochs	Additional Info
GCN	4	128	0.001	200	-
GraphSAGE	4	128	0.001	200	Mean Aggregation
GAT	4	128	0.001	200	8 heads with Hid Dim 8 each
CGCN	4	128	0.001	200	8 subgraphs

Table 4: GNN Architecture Information

M GNN METRICS

The test accuracy for node classification and the ROC AUC value for link prediction is detailed in Table 5

Architecture	Accuracy (NC)	ROC AUC (LP)
GCN	0.8257	0.8638
GraphSAGE	0.8800	0.8068
GAT	0.8273	0.7997
CGCN	0.8200	0.8716

Table 5: GNN Metric Data on Amazon Computer Dataset. Test Accuracy is for Node Classification and ROC AUC Score is for Link Prediction

N DATASETS

We report the statistics of the datasets used for the empirical analysis of GNNs in Table 6. We report the exact statistics of the autoencoder datasets in Table 7.

N.1 GRAPH DATASETS

The Amazon computers dataset is a subset of the Amazon co-purchase graph (Shchur et al., 2019). The nodes represent products on amazon and the edges indicate that two products are frequently bought together. The node features are the product reviews encoded in bag-of-words format. The class labels represent the product category.

The Flickr dataset is an undirected graph where nodes represent images uploaded to Flickr (Zeng et al., 2020). An edge exists between the two nodes if the images share common properties such as same user, geographic location or gallery. The node features are again bag-of-word representation of the images. The labels are the tags of the images manually classified into 7 different categories.

Dataset	**Amazon Computer**	**Flickr**
Number of Nodes	13752	89250
Number of Edges	491722	899756
Average Degree	35.76	10.08
Node Features	767	500
Labels	10	7

Table 6: Dataset Statistics for Graph

N.2 AUTOENCODER DATASETS

Four diverse real-world datasets were utilized for our experiments: MNIST (LeCun et al., 2010), Fashion-MNIST (F-MNIST) (Xiao et al., 2017), COIL-20 (Nene et al., 1996), and CIFAR-10 (Krizhevsky & Hinton, 2009), to comprehensively evaluate the performance of our autoencoder model. MNIST, comprising 28x28 grayscale images of handwritten digits, serves as a foundational benchmark for image classification and feature extraction tasks. Fashion-MNIST extends this by offering a similar format but with 10 classes of clothing items, making it an ideal choice for fashion-related image analysis. COIL-20 presents a unique challenge, with 20 object categories, where each category consists of 72 128x128 color images captured from varying viewpoints, offering a more complex 3D object recognition scenario. Lastly, CIFAR-10 introduces color and additional complexity, featuring 60,000 32x32 color images across 10 object classes, catering to real-world image analysis and deep learning challenges. Our experimentation across these datasets provides valuable insights into the versatility and effectiveness of our autoencoder approach for diverse image analysis tasks.

Dataset	**Classes**	**Train Size**	**Test Size**	**Image Size**	**Data Type**
MNIST	10	60,000	10,000	28x28 (784)	Grayscale
Fashion-MNIST (F-MNIST)	10	60,000	10,000	28x28 (784)	Grayscale
COIL-20	20	1,440	-	128x128 (16384)	Color
CIFAR-10	10	60,000	10,000	32x32*3 (3072)	Color

Table 7: Dataset Statistics for AE