Three Revisits to Node-Level Graph Anomaly Detection: Outliers, Message Passing and Hyperbolic Neural Networks

Jing Gu
Duke Kunshan University
jg481@duke.edu

Dongmian Zou
Duke Kunshan University
dongmian.zou@duke.edu

Abstract

Graph anomaly detection plays a vital role for identifying abnormal instances in complex networks. Despite advancements of methodology based on deep learning in recent years, existing benchmarking approaches exhibit limitations that hinder a comprehensive comparison. In this paper, we revisit datasets and approaches for unsupervised node-level graph anomaly detection tasks from three aspects. Firstly, we introduce outlier injection methods that create more diverse and graph-based anomalies in graph datasets. Secondly, we compare methods employing message passing against those without, uncovering the unexpected decline in performance associated with message passing. Thirdly, we explore the use of hyperbolic neural networks, specifying crucial architecture and loss design that contribute to enhanced performance. Through rigorous experiments and evaluations, our study sheds light on general strategies for improving node-level graph anomaly detection methods.

1 Introduction

Node-level anomaly detection on graphs refers to the process of identifying nodes exhibiting behaviors or characteristics deviating from normal. It finds applications in various domains, such as uncovering spammer groups in online reviews [1], detecting network intrusions or failures [2], and identifying camouflaged fraudsters [3]. Although traditional anomaly detection algorithms may extend to the graph domain, they may not perform well, given the complexity of graph data. To address this challenge, in recent years, many graph anomaly detection (GAD) algorithms based on deep learning techniques have been developed [4].

Many applications require node-level GAD to be performed in an unsupervised manner, where there are no known normal instances available for training. Another challenge lies in the scarcity of benchmarking datasets with labeled anomalies. As a result, previous studies [5–8] have evaluated their algorithms on datasets containing artificially injected outlier nodes. More recently, Liu et al. [9] have provided systematic benchmarking for these algorithms using the same idea for outlier injection. Although the datasets can distinguish the performance of the baseline methods, the outlier injection methods are lopsided to certain regimes. This has limited the extent to which various outlier detection algorithms can be thoroughly tested and compared. In this work, we address unsupervised node-level GAD by proposing different outlier injection methods. Interestingly, we observe a decline of accuracy in the performance of the baseline methods. We thus revisit this problem by proposing novel architectures. Our proposal stems from using the hyperbolic space due to their large capacity which may be used to better separate normal nodes and outliers. Our contributions are summarized as follows.

- We introduce new outlier injection methods for GAD datasets. The generated anomalies are more comprehensive and rely more on graph information.
- We analyze and evaluate the necessity of message passing used in neural networks for GAD. Our comparison of methods reveals an unexpected disadvantage when using message passing.
- We introduce novel neural network architectures that use the hyperbolic space and other essential modules that result in improved performance in various datasets exposed under various outliers.

Gu and Zou, Three Revisits to Node-Level Graph Anomaly Detection: Outliers, Message Passing and Hyperbolic Neural Networks. *Proceedings of the Second Learning on Graphs Conference (LoG 2023)*, PMLR 231, Virtual Event, November 27–30, 2023.

2 Related Works

Graph anomaly detection Among algorithms for GAD, a common strategy is reconstruction-based, which involves training an autoencoder to reconstruct either the graph attributes or the graph adjacency matrix, and using the reconstruction error as an anomaly score. The underlying principle is that the bottleneck in autoencoders limits their expressivity, forcing the model to only learn representations from normal patterns but not outliers. One classical example is the multilayer perceptron autoencoder (MLPAE) [10], which reconstructs node attributes according to the mean squared error (MSE). The MSE also serves as the anomaly score. Some reconstruction-based models incorporate structural information, taking both node attributes and graph adjacency matrices as input. Such models include [6, 11–13]. For instance, GCNAE [11] differs from MLPAE in its use of graph convolutional network (GCN) [14], or more specifically, the message passing scheme, enabling nodes to aggregate information from their neighbors. Moreover, certain models focus on reconstructing both attributes and adjacency matrices [5, 8, 15, 16]. While these approaches leverage additional information for reconstruction, they might not always yield improved performance. This is because the presence of outliers can corrupt the information used for reconstruction, potentially leading to suboptimal results.

Aside from reconstruction, there are also models that take on a generative approach, such as generative adversarial attributed nework (GAAN) [7]. GAAN trains a generator to produce fake graph nodes, an encoder to map nodes into a latent space, and a discriminator to differentiate real and fake nodes. The underlying rationale is that outlier nodes might be grouped among the fake nodes. Traditional anomaly detection algorithms [17–19] can also be used in graph datasets. Liu et al. [9] performed a comprehensive comparison of these methods. However, their outlier injection methods following previous works [5–8] are not sufficiently comprehensive. Despite the lack of other previous works on outlier injection, Waniek et al. [20] designed an adversarial attack method. Their method does not rely on the targeted neural networks and can thus be regarded as an outlier rejection approach.

Hyperbolic neural networks In recent years, hyperbolic neural networks have emerged in the field of machine learning. Thanks to their inherent capacity, hyperbolic spaces can represent data relationships with minimal distortion. Methods that establish hierarchical representations through embeddings in hyperbolic space [21, 22] demonstrate the advantages of hyperbolic geometry in representing complex data structures compared to Euclidean geometry. Following this, Ganea et al. [23] introduced the hyperbolic neural network (HNN), utilizing the Poincaré Ball model of hyperbolic space. Shimizu et al. [24] later extended the model to HNN++. Additionally, Nickel and Kiela [25] developed hyperbolic networks working with the Lorentz model. While earlier methods often engage operations in the tangent space, a more recent alternative [26] performs operation directly in the hyperbolic space. Furthermore, building on the success of GCN in graph data representation, many recent works [27–31] designed hyperbolic graph neural networks that accommodate the message passing operation in the hyperbolic space. A key element in hyperbolic networks is the utilization of hyperbolic distances, which play a crucial role as decision criteria in various models [22, 25, 32, 33].

Neural network models often require the computation of mean or centroid of a set of points. In the hyperbolic space, a widely used method is the Frechét mean [34]. However, existing methods for Frechét mean mainly rely on iterative approaches [35, 36]. Using the squared Lorentzian distance defined in [37], Law et al. [38] introduced the closed-form expression for finding the centroid in Lorentz model, which could be viewed as Frechét mean in pseudo-hyperbolic space. Another choice for mean is the Einstein midpoint [39], but it may cause numerical instability because it involves mapping to and from the Klein model. Additionally, Sala et al. [40] proposed a different mean, pseudo-Euclidean mean, in the hyperboloid model, which could be mapped to the Poincaré ball model.

3 Outlier Injection Methods

3.1 Previous Outlier Injection Methods and Their Problems

We first review commonly used contextual and structural outlier injection methods that are adopted in [5–8] and benchmarked in [9].

Notation We denote an attributed and labeled graph as a quadruple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{y})$. Here, $\mathcal{V} = \{1, 2, \ldots, n_{\mathcal{V}}\}$ is the vertex set with size $|\mathcal{V}| = n_{\mathcal{V}}$. $\mathcal{E} \subseteq \{(i,j) : i,j \in \mathcal{V}\}$ is the edge set with $|\mathcal{E}| = n_{\mathcal{E}}$. $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n_{\mathcal{V}}}]^\top \in \mathbb{R}^{n_{\mathcal{V}} \times n_{\mathbf{x}}}$ is the node attribute matrix, where $\mathbf{x}_i \in \mathbb{R}^{n_{\mathbf{x}}}$ represents the attributes of the i-th node. $\mathbf{y} = [y_1, y_2, \ldots, y_{n_{\mathcal{V}}}]^\top$ is the label vector, where y_i represents the class of node i.

Table 1: Statistics of datasets without outlier injection.

	Squirrel	Chameleon	Actor	Cora	Citeseer	Amazon	PubMed	Flickr	ogbn-arxiv
$n_{\mathcal{V}}$	5,201	2,277	7,600	2,708	3,312	13,752	19,717	89,250	169,343
$n_{\mathcal{E}}$	198,493	31,421	26,752	5,278	4,660	245,861	44,327	449,878	1,157,799
$n_{\mathbf{x}}$	2,089	2,325	932	1,433	3,703	767	500	500	128
$n_{\mathcal{E}}/((n_{\mathcal{V}}^2+n_{\mathcal{V}})/2)$	0.0147	0.0121	0.0009	0.0014	0.0008	0.0026	0.0002	0.0001	0.0001
Degree	76.3	27.6	7.0	3.9	2.8	35.8	4.5	10.1	13.7
Hyperbolicity δ [41, 42]	1.5	2.0	1.5	3.0	4.5	1.5	2.5	1.5	2

Also, $\mathcal{E}$ derives the adjacency matrix $\mathbf{A}=[\mathbf{a}_1,\mathbf{a}_2,...,\mathbf{a}_{n_{\mathcal{V}}}]^\top \in \mathbb{R}^{n_{\mathcal{V}}\times n_{\mathcal{V}}}$, where $A_{ij}=1$ if $(i,j)\in\mathcal{E}$ and $A_{ij}=0$ if otherwise. We denote the node attribute matrix and the adjacency matrix after injecting outliers as $\widetilde{\mathbf{X}}=[\widetilde{\mathbf{x}}_1,\widetilde{\mathbf{x}}_2,...,\widetilde{\mathbf{x}}_{n_{\mathcal{V}}}]^\top$ and $\widetilde{\mathbf{A}}=[\widetilde{\mathbf{a}}_1,\widetilde{\mathbf{a}}_2,...,\widetilde{\mathbf{a}}_{n_{\mathcal{V}}}]^\top$, respectively. We also consider a node to be linked to itself, setting $\widetilde{A}_{ii}=1$ for all node i.

Contextual outlier injection First, each row of $\mathbf{X}$ is normalized according to the l_1 norm: $\mathbf{x}_i'=\mathbf{x}_i/\|\mathbf{x}_i\|_1$. We denote the normalized matrix as $\mathbf{X}'$. Then, o nodes from $\mathcal{V}$ are sampled as candidates for contextual outliers. We denote this candidate set of nodes as $\mathcal{V}_c$. For each node i in $\mathcal{V}_c$, q nodes are sampled from the reference set $\mathcal{V}_r=\mathcal{V}\setminus\mathcal{V}_c$ without replacement, from which we pick the one which is the farthest in l_2 norm, i.e., $j=\mathrm{argmax}_k\|\mathbf{x}_i'-\mathbf{x}_k'\|$. Finally, we replace $\mathbf{x}_i$ with $\mathbf{x}_j$.

Structural outlier injection The structural outlier injection involves the creation of t groups of anomalous nodes, each with size s. Specifically, we sample $o=t\times s$ nodes from $\mathcal{V}$ without replacement to be candidates for the structural outliers. These o nodes are then randomly partitioned into t non-overlapping groups, each with size s. For each group, we first add edges to make the s nodes fully connected. Subsequently, we drop each edge with probability p.

Although the above methods have long served as important benchmarks, we have observed that the outliers generated by these approaches could be distinguished simply by examining the norms of $\{\widetilde{\mathbf{x}}_i\}_{i=1}^{n_{\mathcal{V}}}$ and $\{\widetilde{\mathbf{a}}_i\}_{i=1}^{n_{\mathcal{V}}}$. In fact, the farthest vector as an outlier tends to have a large l_2 norm. Likewise, when node i becomes a structural outlier, it will have a large number of neighbors, leading to a large $\|\widetilde{\mathbf{a}}_i\|_1$. Table 1 shows statistics of commonly used datasets. We observe that $n_{\mathcal{E}}/((n_{\mathcal{V}}^2+n_{\mathcal{V}})/2)$ has small numerical values for all datasets, which implies the sparsity of the underlying graphs and overall small values of $\|\widetilde{\mathbf{a}}_i\|_1$. Therefore, large $\|\widetilde{\mathbf{a}}_i\|_1$ easily stands out.

To test whether the norm information alone distinguishes normal nodes and outliers, we consider the following score function: $\mathrm{score}_{\mathrm{norm}}(i)=\alpha\|\widetilde{\mathbf{x}}_i\|+(1-\alpha)\|\widetilde{\mathbf{a}}_i\|_1$, where α is a hyperparameter balancing contextual and structural information. We consider the commonly used graph datasets listed in Table 1 (reviewed in Appendix C.1), and the following settings considered in [9]: (1) cntxt.: injecting o contextual outliers; (2) strct.: injecting o structural outliers; (3) cntxt.+strct.: injecting $o/2$ contextual outliers and $o/2$ structural outliers. For each dataset, o is taken to be approximately 5% of $n_{\mathcal{V}}$. The specific values of o and other outlier injection parameters are reported in Appendix C.2.

We report the Area Under the Receiver Operating Characteristic Curve (ROC-AUC) results in detecting contextual outlier nodes with $\alpha=1$ and structural outlier nodes with $\alpha=0$ using $\mathrm{score}_{\mathrm{norm}}$ in Table 2. For completeness, we report the ROC-AUC and corresponding Average Precision (AP) results for all three settings with $\alpha=1$ and $\alpha=0$ respectively in Appendix C.3. The high scores observed in these results indicate that contextual and structural outliers could be detected by using only the corresponding l_2 norms with high accuracy.

Table 2: Outlier node detection results using the score function $\mathrm{score}_{\mathrm{norm}}(i)=\alpha\|\widetilde{\mathbf{x}}_i\|+(1-\alpha)\|\widetilde{\mathbf{a}}_i\|_1$. For contextual outlier detection, $\alpha=1$; and for structural outlier detection, $\alpha=0$. The complete list of results are presented in the Appendix C.3. Mean and standard deviation of ROC-AUC (%) taken over 3 trials are reported.

$\mathrm{score}_{\mathrm{norm}}(i)$	Outlier	Squirrel	Chameleon	Actor	Cora	Citeseer	Amazon	PubMed	Flickr	ogbn-arxiv
$\|\widetilde{\mathbf{x}}_i\|$	cntxt.	97.3±0.0	94.5±0.0	91.7±0.1	90.0±0.9	89.7±0.7	98.5±0.0	90.6±0.2	94.4±0.2	95.2±0.0
$\|\widetilde{\mathbf{a}}_i\|_1$	strct.	82.7±0.6	86.9±0.5	96.3±0.1	95.8±0.2	96.4±0.0	91.0±0.1	87.9±0.2	94.8±0.0	96.7±0.0

To examine whether the benchmarking results reported in previous works benefit from the above phenomenon, we conduct an experiment comparing two settings. In addition to the original setting, we introduce a new setting where l_2 normalization is applied, ensuring that each attribute has a unit norm. Specifically, we normalized each row of $\mathbf{X}$ by its l_2 norm: $\mathbf{x}_i'=\mathbf{x}_i/\|\mathbf{x}_i\|$. In this new

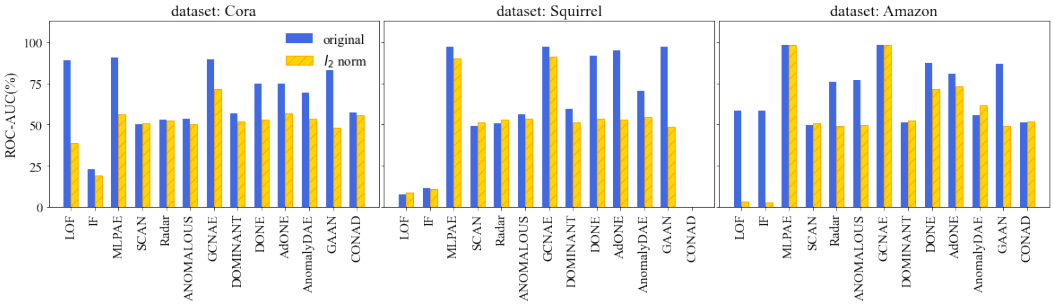

Figure 1: Comparison of model's mean ROC-AUC (%) in detecting contextual outliers injected in Cora, Squirrel, and Amazon datasets with and without l_2 normalization.

setting, it becomes impossible to detect outliers based on the $\|\mathbf{x}_i\|$ criterion. Figure 1 compares the ROC-AUC scores of various GAD models under the original setting (unpatterned bars) and the l_2 normalization setting (patterned bars). We observe that, for all models that achieve a considerable performance (ROC-AUC score much larger than 50%) under the original setting, l_2 normalization leads to a decline in the ROC-AUC score. In all methods but MLPAE and GCNAE, the decline is very significant, indicating that the effectiveness of these methods is significantly affected by the difference in norms of node attributes.

3.2 Proposed Outlier Injection Methods

To eliminate the effect discussed in Section 3.1, we propose the following outlier injection methods where outliers are created based on graph information rather than attributes themselves.

"Path" outlier injection As an alternative for contextual outliers, we propose the "path" outlier. For each node i in candidate set $\mathcal{V}_c$, we replace the attributes of node i with that of node j farthest (according to the shortest-path distance $d_{\mathcal{G}}$) from node i amongst a reference set consisting of q nodes. That is, $j = \mathrm{argmax}_k d_{\mathcal{G}}(i,k)$. To ensure a path could be found between arbitrary nodes, we first convert the graph to undirected, and remove all isolated nodes from the candidate and reference sets.

DICE-n outlier injection To ensure that outlier nodes have similar node degrees with normal nodes, we adapt the "disconnect internally, connect externally" (DICE) approach from [20] to generate structurally perturbed outlier nodes. Originally, DICE is designed as an adversarial and is targeted at the entire graph, disconnecting edges between nodes of the same class and connecting edges between nodes of different classes. To accommodate this method to node-level anomaly detection, our approach differs in restricting the perturbations to the candidate nodes and we call it DICE-n to emphasize the difference. Specifically, we sample o nodes from $\mathcal{V}$ to form the candidate set $\mathcal{V}_c$. For each node $i \in \mathcal{V}_c$, let $\mathcal{V}_{r_1} = \{k;(i,k) \in \mathcal{E}, y_i = y_k\}$ be the reference set containing the nodes that are neighbors of i and are within the same class y_i as i. Let r be the percentage of edges to permute for each candidate node. We sample $\lceil |\mathcal{V}_{r_1}| \times r \rceil$ nodes from $\mathcal{V}_{r_1}$ to form a subset, and for each node j in this subset, we remove the edge (i,j) from $\mathcal{E}$. If the graph $\mathcal{G}$ is undirected, we also disconnect (j,i). To avoid changing the number of neighbors of i, denoting $\mathcal{V}_{r_2} = \{k;(i,k) \notin \mathcal{E}, y_i \neq y_k\}$, we sample a subset of size $\lceil |\mathcal{V}_{r_1}| \times r \rceil$ from $\mathcal{V}_{r_2}$. For each node j in this subset, we add (i,j) (and also (j,i) if $\mathcal{G}$ is undirected) to $\mathcal{E}$. In implementation, due to sparsity of adjacency matrix, we chose to disconnect edges randomly disregarding their classes.

To verify that our outlier injection methods do not suffer from the problems introduced in Section 3.1, we consider the same experiments of outlier detection using only $\mathrm{score}_{\mathrm{norm}}$. Similar to the contextual and structural outliers, We consider the following mixture: (1) "path": injecting o "path" outliers; (2) DICE: injecting o DICE-n outliers; (3) "path"+DICE: injecting $o/2$ "path" outliers and $o/2$ DICE-n outliers. The results are reported in Appendix C.3, where we observe that all the ROC-AUC scores are around 50%, indicating that it is impossible to perform outlier detection solely based on norm information for our proposed outliers.

4 Strategies for Outlier Detection

For many GAD methods and many datasets, the outlier injection methods proposed in Section 3.2 lead to a noticeable decline in overall performance when compared to the previous contextual and structural outliers. Before presenting their numerical results in Section 5, we first delve into strategies that can effectively enhance the performance of outlier detection in this section.

4.1 No Message Passing

In tasks such as node classification, the success of GCN can be attributed to message passing, where information propagates through the graph to boost the performance [14]. Indeed, graph neural networks with message passing have proved more effective than MLP in those tasks. However, in outlier detection, the diffusion across the graph does not solely involve information, but can also include potential outliers. Moreover, implementing message passing may severely constrain the expressiveness of the neural network. Specifically, it is well known that GCN may lose discriminative power where node representations converge to a common average [43–45], called the oversmoothing phenomenon. Although limiting model expressivity can be helpful for anomaly detection, such as in the case of the bottleneck in autoencoders, the extremely low representation power caused by oversmoothing is still harmful.

To see this clearly, we analyze the performance of contextual outlier detection. Although it is in general difficult to perform analysis for GCN-based GAD models, we can consider the following two artificial models: a linear MLPAE whose latent dimension is one, and a GCNAE that produces a single average feature due to oversmoothing. The MLPAE has more distinguishable reconstruction errors than the GCNAE in the scenario described in the following lemma.

Lemma 4.1. *Suppose a graph $\mathcal{G}$ contains $n_{\mathcal{V}}$ nodes, among which n_{normal} nodes are normal ($n_{\text{normal}} > n_{\mathcal{V}}/2$), each with the same unit-norm feature $\mathbf{x}_{\text{normal}}$; and the remaining $n_{\mathcal{V}} - n_{normal}$ nodes are outliers, each with the same unit-norm feature $\mathbf{x}_{\text{outlier}}$ such that $\langle \mathbf{x}_{\text{normal}}, \mathbf{x}_{\text{outlier}} \rangle = 0$. Then,*

1. *the optimal MLPAE has zero reconstruction error for normal nodes and unit reconstruction error for anomalous nodes;*
2. *the optimal GCNAE has reconstruction error $\sqrt{2}\,(1 - n_{\text{normal}}/n_{\mathcal{V}})$ for normal nodes and $\sqrt{2} n_{\text{normal}}/n_{\mathcal{V}}$ reconstruction error for anomalous nodes.*

We present the proof of Lemma 4.1 in Appendix B.1. As to experimental results for methods with and without message passing, Figure 1 has already demonstrated the striking similarity in performance between MLPAE and GCNAE. Moreover, the negative impact of message passing on performance is further evidenced across other scenarios, as will be illustrated in Section 5. It is important to note that when using a GAD model without message passing, we are not omitting the edge information because we still use it in reconstructing the graph adjacency matrices.

4.2 Hyperbolic Neural Networks

Disregarding edge information in feature extraction limits the spread of outliers, yet it might also neglect the relation among features. To address this, we advocate for employing the hyperbolic space as the underlying domain, aiming to preserve geometry of the attributes. Additionally, hyperbolic spaces are known to have large capacity and thus may split normal nodes and outliers with large margins.

Our model comprises two encoding layers, one structural decoding layer, and two contextual decoding layers. Each layer involves feature transformation and centralization. Here, no normalization is taken beyond centralization since dividing by standard deviation easily causes numerical instability.

There are two hyperbolic coordinate systems ("models") we consider: the Lorentz model $\mathbb{L}^n$ and Poincaré ball model $\mathbb{B}^n$, where n is the dimension. Preliminaries on hyperbolic geometry are reviewed in Appendices A.2. For completeness, in Appendix A.3, we also include the construction of our model in the Euclidean space, whose architecture and loss differ from baseline Euclidean models.

4.2.1 Feature Transformation and Centralization

Notation We denote an input graph feature in the Euclidean space as $\mathbf{x}^{E,0}$ (omitting the subscript i for the node). In the hyperbolic space $\mathcal{H}$, the features are exponentially mapped to $\mathbf{x}^{\mathcal{H},0} = \exp_{\mathbf{o}}^{\mathcal{H}}(\mathbf{x}^{E,0})$, where $\mathbf{o}$ denotes the origin in $\mathcal{H}$.

Lorentz We employ fully hyperbolic linear layers [26] for the Lorentz model, mapping the input $\mathbf{x}^{\mathcal{L}} \in \mathbb{L}^n$ to $f_{\mathcal{L}}(\mathbf{x}^{\mathcal{L}}) \in \mathbb{L}^m$. The function $f_{\mathcal{L}}(\cdot)$ is given by

$$f_{\mathcal{L}}(\mathbf{x}^{\mathcal{L}}) = \begin{bmatrix} \sqrt{\|h(\mathbf{x}^{\mathcal{L}})\|^2 - 1/\kappa} \\ h(\mathbf{x}^{\mathcal{L}}) \end{bmatrix}, \quad h(\mathbf{x}^{\mathcal{L}}) = \frac{\lambda\sigma(\mathbf{v}^\top \mathbf{x}^{\mathcal{L}} + b')}{\|\mathbf{W}\tau(\mathbf{x}^{\mathcal{L}}) + \mathbf{b}\|}(\mathbf{W}\tau(\mathbf{x}^{\mathcal{L}}) + \mathbf{b}), \tag{1}$$

where $\mathbf{v} \in \mathbb{R}^{n+1}$ and $\mathbf{W} \in \mathbb{R}^{m \times (n+1)}$ are trainable weights, $b' \in \mathbb{R}$ and $\mathbf{b} \in \mathbb{R}^m$ are trainable biases, $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid function, τ is the activation function which we took to be the identity function, and $\lambda > 0$ is the trainable parameter that scales range. Next, we apply centralization under the Lorentz model by utilizing parallel transportation. More specifically,

$$g_{\mathcal{L}}(\mathbf{x}^{\mathcal{L}}) = \exp_{\mathbf{o}}^{\mathcal{L}}(\mathrm{PT}_{\boldsymbol{\mu} \to \mathbf{o}}^{\mathcal{L}}(\log_{\boldsymbol{\mu}}^{\mathcal{L}}(\mathbf{x}^{\mathcal{L}}))), \quad \boldsymbol{\mu} = \sum_{i=1}^{n_{\text{batch}}} v_i \mathbf{x}_i / (\sqrt{-\kappa} \left| \| \sum_{i=1}^{n_{\text{batch}}} v_i \mathbf{x}_i \|_{\mathcal{L}} \right|). \tag{2}$$

where $\boldsymbol{\mu}$ is the hyperbolic centroid [38] with v_i taken to be 1 for all i, and n_{batch} represents the number of samples in the batch.

Poincaré For the feature transformation under the Poincaré Ball model, we adopt the Möbius matrix-vector multiplication (reviewed in Appendix A.2.2) from HNN [23] and bias addition with a Euclidean bias defined in [27]. Specifically, the input $\mathbf{x}^{\mathcal{B}} \in \mathbb{B}^n$ is mapped to $f_{\mathcal{B}}(\cdot) \in \mathbb{B}^m$ given by

$$\mathbf{y} = \mathrm{proj}(\mathbf{W}^{\otimes}(\mathbf{x}^{\mathcal{B}})),\ \mathbf{z} = \mathrm{proj}(\exp_{\mathbf{y}}^{\mathcal{B}}(\mathrm{PT}_{\mathbf{o} \to \mathbf{y}}^{\mathcal{B}}(\mathbf{b}))),\ f_{\mathcal{B}}(\mathbf{z}) = \mathrm{proj}(\exp_{\mathbf{o}}^{\mathcal{B}}(\mathrm{ReLU}(\log_{\mathbf{o}}^{\mathcal{B}}(\mathbf{z})))). \tag{3}$$

Here, $\mathbf{W} \in \mathbb{R}^{m \times (n+1)}$ is the trainable weight, $\mathbf{b} \in \mathbb{R}^m$ is the trainable bias, and activation $\mathrm{ReLU}(x) = \max(0, x)$ is implemented on the tangent space at the origin. The function $\mathrm{proj}(\cdot)$ is used to constrain the embeddings within the Poincaré Ball [22], where $\mathrm{proj}(\mathbf{x}) = (\mathbf{x}/\|\mathbf{x}\| - \epsilon)$ with ϵ being a small constant if $\|\mathbf{x}\| > 1$, and $\mathrm{proj}(\mathbf{x}) = \mathbf{x}$ if otherwise. Unlike (2), the centroid in the Poincaré model has no closed-form expression. Therefore, we implement centralization through translation on the tangent space. The function $g_{\mathcal{B}}(\cdot)$ is defined as

$$g_{\mathcal{B}}(\mathbf{x}^{\mathcal{B}}) = \exp_{\mathbf{o}}^{\mathcal{B}}(\log_{\mathbf{o}}^{\mathcal{B}}(\mathbf{x}^{\mathcal{B}}) - \boldsymbol{\mu}), \quad \boldsymbol{\mu} = \sum_{i=1}^{n_{\text{batch}}} \log_{\mathbf{o}}^{\mathcal{B}}(\mathbf{x}^{\mathcal{B}}) / n_{\text{batch}}. \tag{4}$$

Although the Lorentz and Poincaré models are isometric to each other, their implementations are not equivalent. Besides training schemes and different implementation of centralization, the exponential map used in HNN has a subtle assumption of Euclidean metric that is twice as large as the regular Euclidean metric. Consequently, $d_{\mathcal{L}}$ is smaller than $d_{\mathcal{B}}$, as formulated in the following lemma.

Lemma 4.2. *Let $\mathbf{x}^E$ and $\mathbf{y}^E$ be distinct unit-norm features in the same Euclidean space so that $\mathbf{x}^E \neq \mathbf{y}^E$ and $\|\mathbf{x}^E\| = \|\mathbf{y}^E\| = 1$. Let $\mathbf{x}^{\mathcal{B}} = \exp_{\mathbf{o}}^{\mathcal{B}}(\mathbf{x}^E)$ and $\mathbf{y}^{\mathcal{B}} = \exp_{\mathbf{o}}^{\mathcal{B}}(\mathbf{y}^E)$ be their exponential maps into the Poincaré model; and $\mathbf{x}^{\mathcal{L}} = \exp_{\mathbf{o}}^{\mathcal{L}}(\mathbf{x}^E)$ and $\mathbf{y}^{\mathcal{L}} = \exp_{\mathbf{o}}^{\mathcal{L}}(\mathbf{y}^E)$ their exponential maps into the Lorentz model, respectively. Then*

$$\|\mathbf{x}^E - \mathbf{y}^E\| \leq d_{\mathcal{L}}(\mathbf{x}^{\mathcal{L}}, \mathbf{y}^{\mathcal{L}}) < d_{\mathcal{B}}(\mathbf{x}^{\mathcal{B}}, \mathbf{y}^{\mathcal{B}}), \tag{5}$$

where the first equality holds if and only if $\mathbf{x}^E = -\mathbf{y}^E$.

We present the proof of Lemma 4.2 in Appendix B.2. From the lemma, we see that the hyperbolic distance in the Poincaré model tends to be larger. Therefore, we hypothesize that normal nodes and outliers can have more distinguishable reconstruction distances. We thus recommend using the Poincaré model and will provide experimental analysis in Section 5.

4.2.2 Losses and Training

We present losses used in our models that are different than previous GAD works. Denote our reconstructed node features as $\widehat{\mathbf{X}} = [\widehat{\mathbf{x}}_1, \widehat{\mathbf{x}}_2, \ldots, \widehat{\mathbf{x}}_{n_{\mathcal{V}}}]^\top$. For structural reconstruction of the adjacency matrix, we utilize the Fermi-Dirac decoder [46], which has previously been applied in link prediction tasks using hyperbolic models [22, 27]. Let $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{n_{\mathcal{V}}}]^\top$ represent the output of the structural decoding layer. The Fermi-Dirac decoder computes the probability of an edge being connected, given the distance $d(\mathbf{h}_i, \mathbf{h}_j)$ between two node embeddings $\mathbf{h}_i$, $\mathbf{h}_j$, via

$$p((i,j) \in \mathcal{E} | \mathbf{h}_i, \mathbf{h}_j) = \mathcal{Z}^{-1} e^{-(d(\mathbf{h}_i, \mathbf{h}_j)^2 - r)/t}, \quad p((i,j) \notin \mathcal{E} | \mathbf{h}_i, \mathbf{h}_j) = \mathcal{Z}^{-1}. \tag{6}$$

where the normalization factor $\mathcal{Z} = 1 + e^{-(d(\mathbf{h}_i, \mathbf{h}_j)^2 - r)/t}$. We note $d(\cdot, \cdot)$ is taken to be the distance corresponding to the model. In all our experiments, we simply took $r = 0$ and $t = 1$. The derivation of these equations is reviewed in Appendix A.5. We use these probabilities to obtain the reconstructed adjacency matrix $\widehat{\mathbf{A}}$, where each entry $\widehat{A}_{ij} = p((i,j) \in \mathcal{E} | \mathbf{h}_i, \mathbf{h}_j)$.

The network is trained by minimizing the loss function $\ell = \alpha \ell_c + (1 - \alpha)\ell_s$, where ℓ_c represents the contextual reconstruction error, ℓ_s denotes the structural reconstruction error, and $\alpha \in [0, 1]$ is

a hyperparameter that balances the influence of these two losses. In both hyperbolic models, the contextual error is defined as $\ell_c^{\mathcal{H}} = \sum_{i=1}^{n_{\mathcal{V}}} \ell_c^{\mathcal{H}}(i)/n_{\mathcal{V}} = \sum_{i=1}^{n_{\mathcal{V}}} d(\mathbf{x}_i^{\mathcal{H},0}, \widehat{\mathbf{x}}_i^{\mathcal{H}})/n_{\mathcal{V}}$, where d stands for the respective hyperbolic distance.

For the structural error, we employ the cross entropy loss. However, since $\widetilde{\mathbf{A}}$ is sparse, the dataset is unbalanced in the two classes. To ensure a balanced treatment of both 0 and 1 entries of $\widetilde{\mathbf{A}}$, we introduce an additional normalization. Specifically, $\ell_s = \sum_{i=1}^{n_{\mathcal{V}}} \ell_s(i)/n_{\mathcal{V}}$, with

$$\begin{aligned}\ell_s(i) &= -\frac{\sum_{j=1}^{n_{\mathcal{V}}} \mathbb{1}_{(i,j)\in\mathcal{E}} \log p((i,j)\in\mathcal{E}|\mathbf{h}_i,\mathbf{h}_j)}{\sum_{j=1}^{n_{\mathcal{V}}} \mathbb{1}_{(i,j)\in\mathcal{E}}} - \frac{\sum_{j=1}^{n_{\mathcal{V}}} \mathbb{1}_{(i,j)\notin\mathcal{E}} \log p((i,j)\notin\mathcal{E}|\mathbf{h}_i,\mathbf{h}_j)}{\sum_{j=1}^{n_{\mathcal{V}}} \mathbb{1}_{(i,j)\notin\mathcal{E}}} \\ &= \frac{\sum_{j=1}^{n_{\mathcal{V}}} \widetilde{A}_{ij}((d(\mathbf{h}_i,\mathbf{h}_j)^2 - r)/t)}{\sum_{j=1}^{n_{\mathcal{V}}} \widetilde{A}_{ij}} + \frac{\sum_j \widetilde{A}_{ij} \log \mathcal{Z}}{\sum_{j=1}^{n_{\mathcal{V}}} \widetilde{A}_{ij}} + \frac{\sum_{j=1}^{n_{\mathcal{V}}} (1-\widetilde{A}_{ij}) \log \mathcal{Z}}{\sum_{j=1}^{n_{\mathcal{V}}} (1-\widetilde{A}_{ij})},\end{aligned} \tag{7}$$

where $\mathbb{1}_{\text{event}}$ is the indicator function taking value 1 if event is true or 0 if event is false. An efficient matrix-wise implementation of the loss is derived in Appendix A.4.

Node outlier detection is accomplished by evaluating the outlier score

$$\text{score}(i) = \alpha \ell_c(i) + (1-\alpha)\ell_s(i). \tag{8}$$

In general, this α is a hyperparameter. However, interestingly, superior results were achieved when setting $\alpha = 0$ (Section 5.2). In this configuration, contextual reconstruction error is disregarded and therefore, it utilizes contextual information during encoding but not decoding; and it utilizes structural information for decoding but not encoding. Given our assumption of anomalies existing in both contextual and structural information available to the models, we prevent them from propagating by "looking only once" at them.

5 Experiments

5.1 Experimental setup

We conduct experiments on datasets containing various types of outlier nodes as defined in Section 3, and specify the training parameters in Appendix C.1. The benchmark models and our Euclidean model were optimized using Adam, while our Lorentz and Poincaré models were optimized using Riemannian Adam [47]. The experiments were conducted using NVIDIA GeForce RTX 3090 GPUs, and we provide the running time and complexity analysis in Appendix C.4. Experiments are conducted on datasets containing various types of outlier nodes defined in Section 3. We share the code implementation on GitHub at `https://github.com/Jing-DS/HNN_GAD`.

5.2 Results and Discussion

Benchmark We present the ROC-AUC results for both our models and the baseline models on the Cora, Squirrel, and Amazon datasets in Tables 3-5. Additionally, the AP results for these datasets, along with the results for the remaining datasets, are provided in Appendix C.5.

Our models significantly outperform the baseline models in detecting "path", DICE-n, and "path"+DICE-n outliers. This suggests that our models are adept at distinguishing more intricate and graph-based outliers. Moreover, our models display competence in identifying traditional contextual and structural outliers. Our models achieve the best results in most datasets when both types of outliers are present. Among the three alternatives, our Poincaré model is the best in most scenarios, validating our claim following Lemma 4.2.

Comparing models with/without message passing and contextual loss In Tables 6 and 7, we report the ROC-AUC results of our models, with or without message passing and contextual loss, on Cora and Squirrel datasets. We report results on other datasets and AP scores in Appendix C.6.

[1]NaN denotes error occured during training due to numerical instability.

Table 3: Mean and standard deviation of ROC-AUC (%) scores on Cora dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
LOF	42.5±1.2	38.6±2.1	53.8±0.6	48.9±0.9	48.9±2.7	51.4±2.7
IF	32.9±1.4	18.8±1.3	49.1±1.8	51.9±2.3	52.6±3.4	49.8±0.7
MLPAE	52.2±2.4	56.3±2.0	50.3±3.5	51.6±1.7	54.1±0.5	51.6±2.8
SCAN	72.6±1.1	50.8±1.6	**96.1**±0.6	49.8±0.3	49.4±0.2	47.8±0.2
Radar	59.1±2.5	52.5±1.5	59.0±2.7	47.2±2.8	47.5±2.0	49.1±1.3
ANOMALOUS	52.1±2.6	50.4±1.8	61.0±0.3	46.7±1.6	46.3±1.4	48.0±1.0
GCNAE	60.1±0.6	71.7±0.9	50.6±1.1	52.0±0.9	53.9±2.4	50.2±1.3
DOMINANT	75.9±1.2	52.1±0.3	94.6±0.2	51.9±1.7	49.1±2.3	48.9±3.4
DONE	72.4±3.2	53.1±0.7	95.3±0.3	50.0±2.6	54.0±2.8	48.4±0.8
AdONE	74.8±1.0	56.5±1.1	94.4±0.2	50.3±2.8	55.3±1.2	48.0±1.2
AnomalyDAE	73.3±1.0	53.7±0.8	90.1±0.1	52.7±0.7	53.8±1.6	49.5±2.2
GAAN	50.3±3.0	48.1±2.0	50.1±1.8	49.4±2.4	47.9±1.7	48.9±1.4
CONAD	74.0±0.4	55.5±1.8	94.6±0.2	51.7±2.8	51.2±3.1	45.1±2.0
GINAE	58.8±0.4	63.5±1.3	59.5±3.7	52.3±2.2	52.6±3.1	52.1±3.0
Ours (Euclidean)	79.4±0.1	66.8±0.7	87.3±2.9	74.3±2.3	71.7±1.3	80.5±0.4
Ours (Lorentz)	70.2±2.7	64.0±0.7	79.5±1.2	66.8±1.6	63.7±1.1	69.6±2.1
Ours (Poincaré)	**81.3**±2.3	**72.5**±1.8	90.3±0.9	**79.2**±0.9	**76.1**±1.7	**86.6**±0.5

Table 4: Mean and standard deviation of ROC-AUC (%) scores on Squirrel dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
LOF	29.8±1.1	8.4±0.1	50.4±1.5	30.6±1.4	10.7±0.1	48.8±0.7
IF	29.5±1.7	10.6±0.3	50.3±0.7	54.5±1.4	61.2±0.7	50.5±1.4
MLPAE	70.0±0.7	90.0±0.8	48.7±1.4	47.3±0.3	44.1±0.6	48.8±0.6
SCAN	66.2±0.9	51.5±1.5	83.3±0.2	44.6±0.9	49.5±1.1	39.7±1.3
Radar	56.0±1.1	52.8±1.3	61.6±0.5	44.4±0.6	48.3±1.4	42.6±0.6
ANOMALOUS	53.5±0.7	53.6±1.2	58.9±2.3	45.7±0.2	51.3±0.2	42.5±1.3
GCNAE	70.0±0.3	**91.3**±1.2	52.3±2.3	44.9±1.5	44.9±0.6	50.4±2.6
DOMINANT	68.0±1.6	51.5±0.5	86.2±0.3	52.5±2.5	51.4±0.9	50.7±1.6
DONE	74.7±0.6	53.4±1.3	92.3±0.3	52.7±1.7	51.2±0.9	50.5±0.4
AdONE	75.7±0.6	52.8±0.9	**98.6**±0.1	52.2±1.1	52.7±1.0	57.6±2.3
AnomalyDAE	70.1±3.6	54.8±3.0	83.4±0.5	50.1±1.4	49.6±1.7	54.1±3.5
GAAN	54.9±2.2	48.4±2.7	60.7±1.0	53.9±0.7	51.9±0.6	56.2±0.5
CONAD	NaN[1]	NaN	NaN	NaN	NaN	NaN
GINAE	62.0±4.8	76.0±3.2	54.7±5.5	51.5±1.6	52.3±1.7	54.4±2.5
Ours (Euclidean)	71.1±13.3	49.8±1.0	53.0±0.9	60.6±15.5	57.9±10.9	49.0±0.8
Ours (Lorentz)	79.9±1.9	70.9±0.6	91.8±0.5	82.2±1.4	75.5±1.4	88.0±0.3
Ours (Poincaré)	**81.6**±2.0	80.7±1.2	84.4±2.2	**85.0**±1.1	**77.6**±2.1	**90.3**±1.4

Table 5: Mean and standard deviation of ROC-AUC (%) scores on Amazon dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
LOF	26.3±1.0	3.3±0.2	48.4±0.7	45.5±1.2	38.9±0.3	51.2±1.9
IF	26.5±0.6	2.8±0.1	50.1±0.8	49.1±0.4	46.0±0.8	49.9±0.7
MLPAE	74.5±0.1	98.2±0.1	51.1±1.0	53.2±0.5	56.7±0.4	49.7±0.4
SCAN	69.3±0.5	50.8±0.6	90.1±0.1	48.0±0.9	50.1±0.3	45.8±0.4
Radar	59.4±1.7	49.2±1.1	72.6±1.6	47.2±0.5	51.6±0.7	47.8±2.1
ANOMALOUS	58.9±0.4	49.7±1.0	70.9±0.3	48.6±1.1	49.7±0.2	47.5±0.6
GCNAE	74.1±0.9	**98.4**±0.0	50.6±0.2	53.7±0.5	57.4±1.2	50.3±0.2
DOMINANT	73.8±0.9	52.3±0.5	92.1±0.3	51.3±1.0	52.8±0.4	50.3±0.7
DONE	82.7±1.1	71.8±2.1	89.7±1.1	52.5±0.6	52.5±0.2	50.3±0.2
AdONE	83.1±0.3	73.2±1.9	91.3±0.3	51.9±1.1	55.5±1.8	50.8±1.0
AnomalyDAE	76.8±0.4	61.6±0.3	91.9±0.1	51.4±1.1	51.2±1.8	49.8±0.4
GAAN	56.1±0.7	49.1±2.6	62.9±1.5	52.3±0.7	53.1±1.3	52.4±0.5
CONAD	72.6±0.6	52.0±0.2	92.2±0.1	51.3±1.2	52.4±0.8	50.5±1.5
GINAE	78.4±1.0	84.6±1.4	74.1±2.0	56.0±0.0	53.1±1.3	61.1±5.0
Ours (Euclidean)	89.7±0.5	83.9±0.6	98.2±0.2	86.6±0.6	79.6±1.5	95.6±0.5
Ours (Lorentz)	87.7±2.5	81.0±1.7	97.6±0.6	89.2±3.2	84.9±3.3	93.9±0.8
Ours (Poincaré)	**90.2**±1.0	86.5±1.0	**98.7**±0.1	**93.6**±0.6	**90.0**±0.9	**96.7**±0.5

The performances with no message passing are substantially better than those with message passing, which is consistent with our analysis in Section 4.1.

As previously mentioned, models that exclude contextual loss in both the loss and score functions ($\alpha = 0$) have overall better results than those with equal weights assigned to structural and contextual reconstruction errors ($\alpha = 0.5$). This is the case even when only contextual outliers are considered. This indicates that, in our current formulation, reconstructing the node attributes may add to additional error caused by outliers and we thus recommend the setting with $\alpha = 0$.

Why Poincaré models work We visualize the distribution of the pairwise node embeddings for the Cora dataset in Figure 2. We also report the result on Squirrel dataset in Appendix C.7. The figures reveal that, thanks to their capacity, Poincaré models tend to map originally disconnected nodes farther apart. Additionally, the curve of disconnected nodes of the other models displays a peak near

Table 6: Comparison of models with/without message passing and contextual loss on Cora dataset. Mean and standard deviation of ROC-AUC (%) taken over 3 trials are reported.

	message	α	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
Ours (Euclidean)	No	0	79.4±0.1	66.8±0.7	87.3±2.9	74.3±2.3	71.7±1.3	80.5±0.4
		0.5	46.0±2.7	47.3±1.8	44.1±1.2	49.8±2.5	52.7±1.1	51.6±1.4
	Yes	0	68.6±1.7	49.3±3.7	82.7±0.5	65.0±1.7	51.8±1.8	79.4±0.3
		0.5	68.4±2.1	54.2±1.9	82.0±1.7	68.5±2.1	55.9±0.7	77.8±2.5
Ours (Lorentz)	No	0	70.2±2.7	64.0±0.7	79.5±1.2	66.8±1.6	63.7±1.1	69.6±2.1
		0.5	70.2±3.6	56.8±0.4	78.5±4.3	68.9±1.8	61.7±2.9	71.2±1.5
	Yes	0	70.6±1.3	50.1±1.2	89.5±1.6	69.3±1.1	53.9±0.9	87.1±1.8
		0.5	72.2±0.7	52.7±0.5	90.1±0.3	71.6±0.6	61.3±0.9	86.5±0.9
Ours (Poincaré)	No	0	**81.3**±2.3	**72.5**±1.8	90.3±0.9	**79.2**±0.9	**76.1**±1.7	86.6±0.5
		0.5	77.7±2.8	59.4±1.0	89.0±0.2	74.4±1.5	72.7±0.9	82.7±0.4
	Yes	0	72.2±1.6	50.7±1.7	**91.6**±0.8	71.2±2.9	54.4±1.9	**87.9**±0.3
		0.5	71.9±2.6	53.2±0.8	91.5±1.6	75.1±0.5	62.6±2.0	85.7±0.9

Table 7: Comparison of models with/without message passing and contextual loss on Squirrel dataset. Mean and standard deviation of ROC-AUC (%) taken over 3 trials are reported.

	message	α	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
Ours (Euclidean)	No	0	71.1±13.3	49.8±1.0	53.0±0.9	60.6±15.5	57.9±10.9	49.0±0.8
		0.5	62.3±0.4	70.0±0.5	52.1±1.3	49.6±0.3	50.2±1.2	48.2±1.4
	Yes	0	47.9±3.3	49.3±0.7	35.9±14.4	57.9±0.7	50.3±1.4	62.9±6.7
		0.5	41.1±3.3	52.7±2.1	25.8±3.5	51.9±0.8	51.5±1.0	46.7±5.0
Ours (Lorentz)	No	0	79.9±1.9	70.9±0.6	**91.8**±0.5	82.2±1.4	75.5±1.4	88.0±0.3
		0.5	78.0±1.7	67.8±2.6	85.9±0.4	79.9±1.5	74.0±2.0	86.0±1.0
	Yes	0	62.0±1.2	63.8±0.8	65.4±4.1	74.0±1.3	62.8±1.1	85.0±0.5
		0.5	62.1±1.5	62.6±2.9	66.4±1.9	71.3±1.5	65.0±1.2	82.5±1.3
Ours (Poincaré)	No	0	**81.6**±2.0	**80.7**±1.2	84.4±2.2	**85.0**±1.1	**77.6**±2.1	**90.3**±1.4
		0.5	76.8±2.3	74.8±1.0	82.1±0.6	84.4±2.0	77.1±1.9	89.3±0.8
	Yes	0	38.2±1.5	50.9±1.7	27.7±1.5	61.5±0.1	50.9±1.6	70.3±1.1
		0.5	40.6±1.1	52.4±1.1	33.6±2.8	60.4±1.0	50.8±0.9	66.6±0.8

the origin, overlapping with that of connected nodes, which could potentially result in more severe misclassification. This observation underscores the advantage of the Poincaré model in capturing and representing node attribute information.

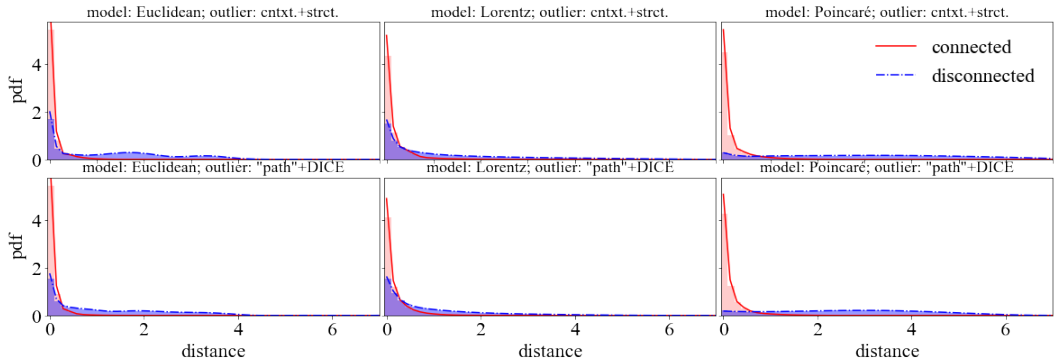

Figure 2: Probability density function for pairwise distances between embeddings of nodes that are originally connected ($\{d(\mathbf{h}_i,\mathbf{h}_j)\}_{(i,j)\in\mathcal{E}}$) and disconnected ($\{d(\mathbf{h}_i,\mathbf{h}_j)\}_{(i,j)\notin\mathcal{E}}$) of the Euclidean, Lorentz, and Poincaré models in Cora dataset injected with cntxt.+strct. and "path"+DICE outliers.

6 Conclusion

In this work, we proposed new outlier injection methods that pose challenge for traditional GAD methods. Additionally, we investigated designs of neural network architecture and loss functions for reconstruction-based outlier detection, identifying hyperbolic neural networks without message passing as effective approaches.

In our approach, we attributed the excelling performance of hyperbolic methods to the large capacity and thus high distinguishability. In the future, we plan to delve deeper into this regime to gain a more comprehensive understanding. Furthermore, we intend to explore the tradeoff between contextual and structural reconstructions to gain additional insights and potential improvements.

References

[1] Junting Ye and Leman Akoglu. Discovering opinion spammer groups by network footprints. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part I 15*, pages 267–282. Springer, 2015. 1

[2] Qi Ding, Natallia Katenka, Paul Barford, Eric Kolaczyk, and Mark Crovella. Intrusion as (anti) social communication: characterization and detection. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 886–894, 2012. 1

[3] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 315–324, 2020. 1

[4] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z Sheng, Hui Xiong, and Leman Akoglu. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021. 1

[5] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 594–602. SIAM, 2019. 1, 2, 17

[6] Haoyi Fan, Fengbin Zhang, and Zuoyong Li. AnomalyDAE: Dual autoencoder for anomaly detection on attributed networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5685–5689. IEEE, 2020. 2, 17

[7] Zhenxing Chen, Bo Liu, Meiqing Wang, Peng Dai, Jun Lv, and Liefeng Bo. Generative adversarial attributed network anomaly detection. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1989–1992, 2020. 2, 17

[8] Xu Yuan, Na Zhou, Shuo Yu, Huafei Huang, Zhikui Chen, and Feng Xia. Higher-order structure based anomaly detection on attributed networks. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 2691–2700. IEEE, 2021. 1, 2, 17

[9] Kay Liu, Yingtong Dou, Yue Zhao, Xueying Ding, Xiyang Hu, Ruitong Zhang, Kaize Ding, Canyu Chen, Hao Peng, Kai Shu, et al. BOND: Benchmarking unsupervised outlier node detection on static attributed graphs. *Advances in Neural Information Processing Systems*, 35: 27021–27035, 2022. 1, 2, 3, 13, 17, 18

[10] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*, pages 4–11, 2014. 2, 17

[11] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016. 2, 17

[12] Jundong Li, Harsh Dani, Xia Hu, and Huan Liu. Radar: Residual analysis for anomaly detection in attributed networks. In *IJCAI*, volume 17, pages 2152–2158, 2017. 17

[13] Zhen Peng, Minnan Luo, Jundong Li, Huan Liu, Qinghua Zheng, et al. ANOMALOUS: A joint modeling approach for anomaly detection on attributed networks. In *IJCAI*, pages 3513–3519, 2018. 2, 17

[14] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. 2, 5

[15] Sambaran Bandyopadhyay, Lokesh N, Saley Vishal Vivek, and M Narasimha Murty. Outlier resistant unsupervised deep architectures for attributed network embedding. In *Proceedings of the 13th international conference on web search and data mining*, pages 25–33, 2020. 2, 17

[16] Zhiming Xu, Xiao Huang, Yue Zhao, Yushun Dong, and Jundong Li. Contrastive attributed network anomaly detection with data augmentation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 444–457. Springer, 2022. 2, 17

[17] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000. 2, 17

[18] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1):1–39, 2012. 17

[19] Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas AJ Schweiger. SCAN: a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 824–833, 2007. 2, 17

[20] Marcin Waniek, Tomasz P Michalak, Michael J Wooldridge, and Talal Rahwan. Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2(2):139–147, 2018. 2, 4, 13

[21] Kevin Verbeek and Subhash Suri. Metric embedding, hyperbolic space, and social networks. In *Proceedings of the thirtieth annual symposium on Computational geometry*, pages 501–510, 2014. 2

[22] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30, 2017. 2, 6, 13

[23] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *Advances in neural information processing systems*, 31:5345–5355, 2018. 2, 6

[24] Ryohei Shimizu, YUSUKE Mukuta, and Tatsuya Harada. Hyperbolic neural networks++. In *International Conference on Learning Representations*, 2021. 2

[25] Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning*, pages 3779–3788. PMLR, 2018. 2

[26] Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Fully hyperbolic neural networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5672–5686, 2022. 2, 6

[27] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems*, 32:4868–4879, 2019. 2, 6

[28] Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. *Advances in Neural Information Processing Systems*, 32:8230–8241, 2019.

[29] Gregor Bachmann, Gary Bécigneul, and Octavian Ganea. Constant curvature graph convolutional networks. In *International Conference on Machine Learning*, pages 486–496. PMLR, 2020.

[30] Jindou Dai, Yuwei Wu, Zhi Gao, and Yunde Jia. A hyperbolic-to-hyperbolic graph convolutional network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 154–163, 2021.

[31] Yiding Zhang, Xiao Wang, Chuan Shi, Nian Liu, and Guojie Song. Lorentzian graph convolutional networks. In *Proceedings of the Web Conference 2021*, pages 1249–1261, 2021. 2

[32] Menglin Yang, Zhihao Li, Min Zhou, Jiahong Liu, and Irwin King. HICF: Hyperbolic informative collaborative filtering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2212–2221, 2022. 2

[33] Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. Poincaré glove: Hyperbolic word embeddings. In *International Conference on Learning Representations*, 2019. 2

[34] H. Karcher. Riemannian center of mass and mollifier smoothing. *Communications on pure and applied mathematics*, 30(5):509–541, 1977. 2

[35] Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. Learning mixed-curvature representations in product spaces. In *International conference on learning representations*, 2018. 2

[36] Aaron Lou, Isay Katsman, Qingxuan Jiang, Serge Belongie, Ser-Nam Lim, and Christopher De Sa. Differentiating through the fréchet mean. In *International Conference on Machine Learning*, pages 6393–6403. PMLR, 2020. 2

[37] John G Ratcliffe, S Axler, and KA Ribet. *Foundations of hyperbolic manifolds*, volume 149. Springer, 1994. 2

[38] Marc Law, Renjie Liao, Jake Snell, and Richard Zemel. Lorentzian distance learning for hyperbolic representations. In *International Conference on Machine Learning*, pages 3672–3681. PMLR, 2019. 2, 6

[39] Abraham A Ungar. *Analytic hyperbolic geometry: Mathematical foundations and applications*. World Scientific, 2005. 2

[40] Frederic Sala, Chris De Sa, Albert Gu, and Christopher Ré. Representation tradeoffs for hyperbolic embeddings. In *International conference on machine learning*, pages 4460–4469. PMLR, 2018. 2

[41] Edmond Jonckheere, Poonsuk Lohsoonthorn, and Francis Bonahon. Scaled gromov hyperbolic graphs. *Journal of Graph Theory*, 57(2):157–180, 2008. 3

[42] Aaron B Adcock, Blair D Sullivan, and Michael W Mahoney. Tree-like structure in large social and information networks. In *2013 IEEE 13th international conference on data mining*, pages 1–10. IEEE, 2013. 3

[43] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in GNNs. In *International Conference on Learning Representations*, 2020. 5

[44] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. In *ICML2020 Workshop on Graph Representation Learning and Beyond*, 2020.

[45] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3438–3445, 2020. 5

[46] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010. 6, 15

[47] Gary Becigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. In *International Conference on Learning Representations*, 2019. 7

[48] James W Anderson. *Hyperbolic geometry*. Springer Science & Business Media, 2006. 13

[49] W. Peng, T. Varanka, A. Mostafa, H. Shi, and G. Zhao. Hyperbolic deep neural networks: A survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, December 2021. doi: 10.1109/TPAMI.2021.3136921. 13

[50] Rishi Sonthalia and Anna Gilbert. Tree! i am no tree! i am a low dimensional hyperbolic embedding. *Advances in Neural Information Processing Systems*, 33:845–856, 2020. 13

[51] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, volume 8, page 1, 2012. 17

[52] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008. 17

[53] 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. *Advances in neural information processing systems*, 33:22118–22133, 2020. 17

[54] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018. 17

[55] 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. 17

[56] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021. 17

[57] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 807–816, 2009. 17

[58] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. 17

[59] Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455*, 2022. 28

Appendix

We review preliminaries in Appendix A. We offer proofs for the technical results in the main text in Appendix B. We present more details and additional results for the experiments in Appendix C.

A Preliminaries

A.1 Relevance to Real Scenarios

We briefly motivate how our outlier injection methods in Section 3 align with real-world applications.

The original DICE method, initially introduced in [20], was specifically motivated by addressing real-world scenarios, particularly within the context of terrorist networks. Even when using a simplified version of DICE, it is possible to easily hide a terrorist's identity, posing a significant security threat. The ability to detect such anomalies is important, particularly in the context of security applications. Our DICE-n outlier injection method is inherently aligned with this motivation.

Regarding the "path" method, it shares the same real-world relevance as the contextual outlier in [9]. However, focusing on GAD problem, we have designed the outlier in a way where it is less likely to easily detect outliers based on attributes alone. We have considered a more graph-based approach that considers the graph distance instead of Euclidean distance of features between nodes.

A.2 Hyperbolic Geometry

Hyperbolic geometry is a non-Euclidean geometry that has recently gained significant attention in machine learning due to its unique properties. Unlike Euclidean geometry, hyperbolic geometry introduces a space which is negatively curved. We refer to [48] for a mathematical introduction and [49] for a comprehensive survey on deep learning in the hyperbolic spaces.

The use of hyperbolic geometry in machine learning stems from its capacity to represent data. Balls with the same radii will have a larger volume in the hyperbolic space than in the Euclidean space. In particular, hyperbolic spaces can model hierarchical and tree-like structures with arbitrarily small distortion [22, 50] and are therefore suitable for representing graph data.

Next, we review some fundamental geometric concepts.

Geodesic Geodesics are locally distance-minimizing curves on manifolds. Unlike straight paths in the Euclidean space, geodesics in the hyperbolic space follow curved paths. The lengths of these paths are used to measure distances between points.

Tangent space The tangent space at a point on a manifold is a vector space that captures the local behavior of the manifold at that specific point. Given a hyperbolic manifold $\mathcal{M}$, we denote the tangent space at $\mathbf{x} \in \mathcal{M}$ as $\mathcal{T}_{\mathbf{x}}\mathcal{M}$. The dimension of the tangent space is equal to the dimension of the manifold.

Exponential and logarithmic maps Exponential and logarithmic maps describe the connection between the manifold and the tangent space. Given $\mathbf{x},\mathbf{y} \in \mathcal{M}$ and $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$, the exponential map $\exp_{\mathbf{x}} : \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathcal{M}$ projects $\mathbf{v}$ to $\gamma(1) \in \mathcal{M}$, where γ is the geodesic satisfying $\gamma(0) = \mathbf{x}$ and $\gamma'(0) = \mathbf{v}$. The logarithmic map $\log_{\mathbf{x}} : \mathcal{M} \to \mathcal{T}_{\mathbf{x}}\mathcal{M}$ defines an inverse process, satisfying $\log_{\mathbf{x}}(\exp_{\mathbf{x}}(\mathbf{v})) = \mathbf{v}$.

Parallel transport Given $\mathbf{x},\mathbf{y} \in \mathcal{M}$, parallel transport $\mathrm{PT}^{\mathcal{M}}_{\mathbf{x}\to\mathbf{y}}$ translates the vector $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$ to $\mathrm{PT}^{\mathcal{M}}_{\mathbf{x}\to\mathbf{y}}(\mathbf{v}) \in \mathcal{T}_{\mathbf{y}}\mathcal{M}$ along the geodesic from $\mathbf{x}$ to $\mathbf{y}$.

There are several coordinate systems, or models, for describing points in the hyperbolic space. In our paper, we have used the Lorentz model and the Poincaré model. We briefly review them as follows. Note that in general the curvature can be taken to be other negative constants, but for simplicity we take it as -1.

A.2.1 Hyperbolic Geometry for Lorentz Model

The n-dimensional Lorentz model $\mathbb{L}^n = (\mathcal{L}, \mathfrak{g}^{\mathcal{L}})$ is the manifold $\mathcal{L}$ isometrically embedded in the $(n+1)$-dimensional Minkowski space. It comes with the metric tensor $\mathfrak{g}^{\mathcal{L}} = \mathrm{diag}(-1,1,...,1)$, and constant negative curvature $\kappa = -1$. Each vector in $\mathbb{L}^n$ can be represented using $(n+1)$ coordinates. Each $\mathbf{x} \in \mathbb{L}^n$ satisfies $\langle \mathbf{x},\mathbf{x} \rangle_{\mathcal{L}} = 1/\kappa$, where $\langle \cdot,\cdot \rangle_{\mathcal{L}}$ is the Minkowski inner product defined as follows.

Inner product For $\mathbf{x},\mathbf{y}\in\mathbb{L}^n$, the inner product is defined as

$$\langle\mathbf{x},\mathbf{y}\rangle_{\mathcal{L}}=-x_0y_0+\sum_{i=1}^{n}x_iy_i,\mathbf{x},\mathbf{y}\in\mathbb{L}^n. \tag{9}$$

Distance The distance between two vectors $\mathbf{x},\mathbf{y}\in\mathbb{L}^n$ is expressed as

$$d_{\mathcal{L}}(\mathbf{x},\mathbf{y})=\frac{1}{\sqrt{-\kappa}}\cosh^{-1}(\kappa\langle\mathbf{x},\mathbf{y}\rangle_{\mathcal{L}}). \tag{10}$$

Exponential and logarithmic maps Given $\mathbf{x},\mathbf{y}\in\mathbb{L}^n,\mathbf{x}\neq\mathbf{y}$ and $\mathbf{v}\in\mathcal{T}_{\mathbf{x}}\mathbb{L}^n\setminus\{\mathbf{0}\}$, the exponential and logarithmic maps in $\mathbb{L}^n$ are expressed by

$$\begin{aligned}&\exp_{\mathbf{x}}^{\mathcal{L}}(\mathbf{v})=\cosh(\phi)\mathbf{x}+\phi^{-1}\sinh(\phi)\mathbf{v},\phi=\sqrt{-\kappa}\|\mathbf{v}\|_{\mathcal{L}},\\&\log_{\mathbf{x}}^{\mathcal{L}}(\mathbf{y})=\frac{\cosh^{-1}(\psi)}{\sqrt{-\kappa}}\frac{\mathbf{y}-\psi\mathbf{x}}{\|\mathbf{y}-\psi\mathbf{x}\|_{\mathcal{L}}},\psi=\kappa\langle\mathbf{x},\mathbf{y}\rangle_{\mathcal{L}},\end{aligned} \tag{11}$$

where $\|\cdot\|_{\mathcal{L}}=\sqrt{\langle\cdot,\cdot\rangle_{\mathcal{L}}}$.

Parallel transport Given $\mathbf{x},\mathbf{y}\in\mathbb{L}^n$ and $\mathbf{v}\in\mathcal{T}_{\mathbf{x}}\mathcal{L}$, in $\mathbb{L}^n$, parallel transport is expressed as

$$\mathrm{PT}_{\mathbf{x}\to\mathbf{y}}^{\mathcal{L}}(\mathbf{v})=\frac{\langle\mathbf{y},\mathbf{v}\rangle_{\mathcal{L}}}{-1/\kappa-\langle\mathbf{x},\mathbf{y}\rangle_{\mathcal{L}}}(\mathbf{x}+\mathbf{y}). \tag{12}$$

A.2.2 Hyperbolic Geometry for Poincaré Model

The Poincaré ball model with constant negative curvature $\kappa=-1$ and unit radius is defined as $\mathbb{B}^n=(\mathcal{B},\mathfrak{g}^{\mathcal{B}})$. Here, the manifold $\mathcal{B}=\{\mathbf{x}\in\mathbb{R}^n:\|\mathbf{x}\|<1\}$ is the n-dimensional unit ball, and the metric tensor $\mathfrak{g}^{\mathcal{B}}=\lambda_{\mathbf{x}}^2\mathbf{I}$, where $\lambda_{\mathbf{x}}=2/(1-||\mathbf{x}||^2)$ is the conformal factor and $\mathbf{I}\in\mathbb{R}^{n\times n}$ represents the identity matrix.

Distance Distance between two points $\mathbf{x},\mathbf{y}\in\mathbb{B}^n$ is defined as

$$d_{\mathcal{B}}(\mathbf{x},\mathbf{y})=\cosh^{-1}\left(1+2\frac{\|\mathbf{x}-\mathbf{y}\|^2}{(1-\|\mathbf{x}\|^2)(1-\|\mathbf{y}\|^2)}\right). \tag{13}$$

Möbius addition Given $\mathbf{x},\mathbf{y}\in\mathbb{B}^n$, Möbius addition is defined as

$$\mathbf{x}\oplus\mathbf{y}:=\frac{(1+2\langle\mathbf{x},\mathbf{y}\rangle+\|\mathbf{y}\|^2)\mathbf{x}+(1-\|\mathbf{x}\|^2)\mathbf{y}}{1+2\langle\mathbf{x},\mathbf{y}\rangle+\|\mathbf{x}\|^2\|\mathbf{y}\|^2}. \tag{14}$$

Möbius scalar multiplication For $r\in\mathbb{R}$ and $\mathbf{x}\in\mathbb{B}^n\setminus\{\mathbf{0}\}$, Möbius scalar multiplication is defined as

$$r\otimes\mathbf{x}:=\tanh(r\tanh^{-1}(\|\mathbf{x}\|))\frac{\mathbf{x}}{\|\mathbf{x}\|}. \tag{15}$$

Möbius matrix-vector multiplication Given $\mathbf{M}\in\mathbb{R}^{m\times n}$ and $\mathbf{x}\in\mathbb{B}^n$, for $\mathbf{M}\mathbf{x}\neq\mathbf{0}$, Möbius matrix-vector multiplication is given by

$$\mathbf{M}^{\otimes}(\mathbf{x}^{\mathcal{B}})=\tanh\left(\frac{\|\mathbf{M}\mathbf{x}\|}{\|\mathbf{x}\|}\tanh^{-1}(\|\mathbf{x}\|)\right)\frac{\mathbf{M}\mathbf{x}}{\|\mathbf{M}\mathbf{x}\|}. \tag{16}$$

Exponential and logarithmic maps For $\mathbf{x},\mathbf{y}\in\mathbb{B}^n,\mathbf{x}\neq\mathbf{y}$ and $\mathbf{v}\in\mathcal{T}_{\mathbf{x}}\mathbb{B}^n\setminus\{\mathbf{0}\}$, the exponential and logarithmic maps in $\mathbb{B}^n$ are given by

$$\begin{aligned}&\exp_{\mathbf{x}}^{\mathcal{B}}(\mathbf{v})=\mathbf{x}\oplus\left(\tanh\left(\frac{\lambda_{\mathbf{x}}\|\mathbf{v}\|}{2}\right)\frac{\mathbf{v}}{\|\mathbf{v}\|}\right),\\&\log_{\mathbf{x}}^{\mathcal{B}}(\mathbf{y})=\frac{2}{\lambda_{\mathbf{x}}}\tanh^{-1}(\|-\mathbf{x}\oplus\mathbf{y}\|)\frac{-\mathbf{x}\oplus\mathbf{y}}{\|-\mathbf{x}\oplus\mathbf{y}\|}.\end{aligned} \tag{17}$$

Parallel transport Given $\mathbf{x},\mathbf{y}\in\mathbb{B}^n$ and $\mathbf{v}\in\mathcal{T}_{\mathbf{x}}\mathcal{B}$, parallel transport in $\mathbb{B}^n$ is defined as

$$\mathrm{PT}_{\mathbf{x}\to\mathbf{y}}^{\mathcal{B}}(\mathbf{v})=\log_{\mathbf{y}}^{\mathcal{B}}(\mathbf{y}\oplus\exp_{\mathbf{x}}^{\mathcal{B}}(\mathbf{v}))=\frac{\lambda_{\mathbf{x}}}{\lambda_{\mathbf{y}}}\mathbf{v}. \tag{18}$$

A.3 Euclidean Model

For completeness, we also outline the construction of our model in the Euclidean space. In the Euclidean model, each layer operates with the input $\mathbf{x}^E \in \mathbb{R}^n$ being transformed to $f_E(\mathbf{x}^E) = \mathrm{ReLU}(\mathbf{W}\mathbf{x}^E + \mathbf{b}) \in \mathbb{R}^m$. Here, $\mathbf{W}$ and $\mathbf{b}$ are trainable parameters, and the activation function $\mathrm{ReLU}(x) = \max(0,x)$ is applied.

We implemented batch centralization on the embeddings following the feature transformation. In the case of the Euclidean model, we performed centralization using the function $g_E(\mathbf{x}^E) = \mathbf{x}^E - \boldsymbol{\mu}^E$, where $\boldsymbol{\mu}^E = \sum_{i=1}^{n_{\text{batch}}} \mathbf{x}_i^E / n_{\text{batch}} \in \mathbb{R}^m$.

A.4 Pairwise Distance Computing Using Matrices

Given matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]^\top \in \mathbb{R}^{n \times m}$, to obtain the distance matrix $\mathbf{D}$ where $D_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$ without iteratively computing each entry or row, we use the equations below in our algorithm.

Euclidean Given $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^m, \forall i = 1, 2, \dots, n\}$, and the equation $d_E(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\| = \sqrt{\|\mathbf{x}_i\|^2 + \|\mathbf{x}_j\|^2 - 2(\mathbf{x}_i)^\top \mathbf{x}_j}$, we can compute $\mathbf{D}$ by

$$\mathbf{D} = d_E(\mathbf{X}) = \left(\left(\sum_j X_{ij}^2 \right) \boxplus \left(\sum_j X_{ij}^2 \right)^\top - 2\mathbf{X}\mathbf{X}^\top \right)^{\frac{1}{2}}. \tag{19}$$

Here, the "$\boxplus$" refers to addition with broadcasting. For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$, we can define "$\boxplus$" as

$$\mathbf{x} \boxplus \mathbf{y}^\top = \begin{bmatrix} | & | & & | \\ \mathbf{x} & \mathbf{x} & \cdots & \mathbf{x} \\ | & | & & | \end{bmatrix}_{m \times m} + \begin{bmatrix} - \mathbf{y}^\top - \\ - \mathbf{y}^\top - \\ \vdots \\ - \mathbf{y}^\top - \end{bmatrix}_{m \times m}. \tag{20}$$

Lorentz The distance in $\mathbb{L}$, for $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{L}^m$, is defined as $d_{\mathcal{L}}(\mathbf{x}_i, \mathbf{x}_j) = \cosh^{-1}(\langle \mathbf{x}_i, \mathbf{x}_j \rangle_{\mathcal{L}})$. For $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{L}^m, \forall i = 1, 2, ..., n\}$, we have

$$\mathbf{D} = d_{\mathcal{L}}(\mathbf{X}) = \cosh^{-1}\left(\left(-\mathbf{x}_{:,0} \mathbf{x}_{:,0}^\top + \mathbf{X}_{:,1:} \mathbf{X}_{:,1:}^\top \right) \right). \tag{21}$$

Poincaré Given $\mathbf{x}_i$, $\mathbf{x}_j \in \mathbb{B}^m$, the distance in $\mathbb{B}$ is given by $d_{\mathcal{B}}(\mathbf{x}_i, \mathbf{x}_j) = \cosh^{-1}\left(1 + 2 \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{(1 - \|\mathbf{x}_i\|^2)(1 - \|\mathbf{x}_j\|^2)}\right)$. For $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{B}^m, \forall i = 1, 2, ..., n\}$, we can define

$$\mathbf{D} = d_{\mathcal{B}}(\mathbf{X}) = \cosh^{-1}\left(1 + 2 \frac{\left(\sum_j X_{ij}^2 \right) \boxplus \left(\sum_j X_{ij}^2 \right)^\top - 2\mathbf{X}\mathbf{X}^\top}{\left(1 - \left(\sum_j X_{ij}^2 \right) \right) \left(1 - \left(\sum_j X_{ij}^2 \right) \right)^\top} \right). \tag{22}$$

A.5 Fermi-Dirac Decoder

Fermi-Dirac Decoder [46] originates from the physics equation that describes the probability of a state being occupied by particles. Specifically,

$$p(n) = \frac{1}{\mathcal{Z}} e^{-n(E-\mu)/kT}, \tag{23}$$

where $\mathcal{Z} = \sum_n e^{-n(E-\mu)/kT}$ is the grand partition function, E is the energy of the system when occupied by a single particle, μ is the chemical potential, and T is the temperature. The term "Fermi-Dirac" comes from the Fermi-Dirac distribution $\bar{n}_{FD}$, which describes the average occupancy of a state when the particles are fermions. For fermions, the occupancy n could take values of 0 and 1, so $\mathcal{Z}$ simplifies to $\mathcal{Z} = 1 + e^{-(E-\mu)/kT}$ and

$$\bar{n}_{FD} = \sum_n n p(n) = 0 \cdot \frac{1}{\mathcal{Z}} + 1 \cdot \frac{1}{\mathcal{Z}} e^{-(E-\mu)/kT} = \frac{1}{1 + e^{(E-\mu)/kT}}. \tag{24}$$

Viewing a connected edge as occupation $n = 1$ and a disconnected edge as $n = 0$, we take the squared distance $d(\mathbf{h}_i, \mathbf{h}_j)^2$ to be energy E, where $\mathbf{h}_i$ and $\mathbf{h}_j$ are the embeddings of node i and j,

and hyperparameters r and t to be μ and T, respectively. Given the distance $d(\mathbf{h}_i,\mathbf{h}_j)$, the probability of an edge being connected can be computed as

$$p((i,j)\in\mathcal{E}|\mathbf{h}_i,\mathbf{h}_j)=p(n=1)=\frac{1}{\mathcal{Z}}e^{-(d(\mathbf{h}_i,\mathbf{h}_j)^2-r)/t}=\frac{1}{1+e^{(d(\mathbf{h}_i,\mathbf{h}_j)^2-r)/t}}. \tag{25}$$

This probability expression mirrors the structure of the Fermi-Dirac distribution $\bar{n}_{FD}$. Similarly, we can calculate the probability of edge being disconnected by

$$p((i,j)\notin\mathcal{E}|\mathbf{h}_i,\mathbf{h}_j)=p(n=0)=\frac{1}{\mathcal{Z}}. \tag{26}$$

B Proofs

B.1 Proof of Lemma 4.1

Proof. 1. Since the MLPAE is linear with latent dimension one, it finds the best one-dimensional subspace that fits the data. Clearly, when optimized, this subspace should lie in the two-dimensional plane spanned by $\mathbf{x}_{\text{normal}}$ and $\mathbf{x}_{\text{outlier}}$, for otherwise projecting the subspace into the plane will reduce the distance between $\mathbf{x}_{\text{normal}}$ and the subspace as well as the distance between $\mathbf{x}_{\text{outlier}}$ and the subspace.

Consider the plane spanned by $\mathbf{x}_{\text{normal}}$ and $\mathbf{x}_{\text{outlier}}$. A natural coordinate system sets $\mathbf{x}_{\text{normal}}=(1,0)^\top$ and $\mathbf{x}_{\text{outlier}}=(0,1)^\top$. Accordingly, the subspace representing the MLPAE output can be represented as $\mathbb{L}=\{(t\cos\theta,t\sin\theta)\}$ which is parameterized by θ.

The MSE reconstruction error is thus given by

$$\ell(\theta)=\frac{1}{n_{\mathcal{V}}}\big(n_{\text{normal}}d^2(\mathbf{x}_{\text{normal}},\mathbb{L})+(n_{\mathcal{V}}-n_{\text{normal}})d^2(\mathbf{x}_{\text{outlier}},\mathbb{L})\big) \tag{27}$$

$$=\frac{1}{n_{\mathcal{V}}}\Bigg(n_{\text{normal}}\big((1-\cos^2\theta)^2+\cos^2\theta\sin^2\theta\big)+ (n_{\mathcal{V}}-n_{\text{normal}})\big(\sin^2\theta\cos^2\theta+(\sin^2\theta-1)^2\big)\Bigg) \tag{28}$$

$$=\left(1-\frac{n_{\text{normal}}}{n_{\mathcal{V}}}\right)+\left(1-2\frac{n_{\text{normal}}}{n_{\mathcal{V}}}\right)\cos^2\theta. \tag{29}$$

Since $1-2\frac{n_{\text{normal}}}{n_{\mathcal{V}}}<0$, the minimum of (29) is $1-\frac{n_{\text{normal}}}{n_{\mathcal{V}}}$, which is achieved if and only if $\cos\theta=0$, yielding

$$d(\mathbf{x}_{\text{normal}},\mathbb{L})=0,\qquad d(\mathbf{x}_{\text{outlier}},\mathbb{L})=1. \tag{30}$$

2. Firstly, the output of GCNAE, denoted as $\mathbf{x}$, should lie in the line passing through $\mathbf{x}_{\text{normal}}$ and $\mathbf{x}_{\text{outlier}}$, for otherwise projecting $\mathbf{x}$ into this line would reduce both $\|\mathbf{x}-\mathbf{x}_{\text{normal}}\|$ and $\|\mathbf{x}-\mathbf{x}_{\text{outlier}}\|$ and thus the reconstruction error. Let

$$\mathbf{x}=(1-t)\mathbf{x}_{\text{normal}}+t\mathbf{x}_{\text{outlier}},\ t\in\mathbb{R}. \tag{31}$$

The MSE reconstruction error is given by

$$\ell(\mathbf{x})=n_{\text{normal}}\|\mathbf{x}-\mathbf{x}_{\text{normal}}\|^2+(n_{\mathcal{V}}-n_{\text{normal}})\|\mathbf{x}-\mathbf{x}_{\text{outlier}}\|^2 \tag{32}$$

$$=n_{\text{normal}}t^2\|\mathbf{x}_{\text{normal}}-\mathbf{x}_{\text{outlier}}\|^2+(n_{\mathcal{V}}-n_{\text{normal}})(1-t)^2\|\mathbf{x}_{\text{normal}}-\mathbf{x}_{\text{outlier}}\|^2 \tag{33}$$

$$=2\big(n_{\text{normal}}t^2+(n_{\mathcal{V}}-n_{\text{normal}})(1-t)^2\big). \tag{34}$$

Clearly, the minimizer of the quadratic function (34) is $t=1-\frac{n_{\text{normal}}}{n_{\mathcal{V}}}$, leading to

$$\|\mathbf{x}-\mathbf{x}_{\text{normal}}\|=\sqrt{2}\left(1-\frac{n_{\text{normal}}}{n_{\mathcal{V}}}\right),\qquad \|\mathbf{x}-\mathbf{x}_{\text{outlier}}\|=\sqrt{2}\frac{n_{\text{normal}}}{n_{\mathcal{V}}}. \tag{35}$$

□

B.2 Proof of Lemma 4.2

Proof. For convenience, we denote

$$d:=\big\|\mathbf{x}^E-\mathbf{y}^E\big\|. \tag{36}$$

First, the exponential maps of $\mathbf{x}^E$ and $\mathbf{y}^E$ in the hyperbolic space under the Lorentz model are given by

$$\mathbf{x}^{\mathcal{L}}=\begin{bmatrix}\cosh(1)\\ \sinh(1)\mathbf{x}^E\end{bmatrix},\qquad \mathbf{y}^{\mathcal{L}}=\begin{bmatrix}\cosh(1)\\ \sinh(1)\mathbf{y}^E\end{bmatrix}. \tag{37}$$

Their hyperbolic distance is given by

$$d_{\mathcal{L}}(\mathbf{x}^{\mathcal{L}},\mathbf{y}^{\mathcal{L}}) = \cosh^{-1}\left(\cosh^2(1) - \sinh^2(1)\langle \mathbf{x}^E, \mathbf{y}^E\rangle\right) \tag{38}$$

$$= \cosh^{-1}\left(\cosh^2(1) - \sinh^2(1)\left(1 - \frac{d^2}{2}\right)\right) \tag{39}$$

$$= \cosh^{-1}\left(1 + \frac{\sinh^2(1)}{2} d^2\right). \tag{40}$$

A routine calculus shows that, when $0 \le d \le 2$,

$$1 + \frac{\sinh^2(1)}{2} d^2 \ge \cosh(d). \tag{41}$$

Since both $\mathbf{x}^E$ and $\mathbf{y}^E$ have unit norms, $0 \le d \le 2$ holds and thus

$$d_{\mathcal{L}}(\mathbf{x}^{\mathcal{L}},\mathbf{y}^{\mathcal{L}}) \ge d. \tag{42}$$

Next, the exponential maps of $\mathbf{x}^E$ and $\mathbf{y}^E$ in the hyperbolic space under the Poincaré model are given by

$$\mathbf{x}^{\mathcal{B}} = \tanh(1)\mathbf{x}^E, \qquad \mathbf{y}^{\mathcal{B}} = \tanh(1)\mathbf{y}^E. \tag{43}$$

Their hyperbolic distance is given by

$$d_{\mathcal{B}}(\mathbf{x}^{\mathcal{B}},\mathbf{y}^{\mathcal{B}}) = \cosh^{-1}\left(1 + 2\frac{\|\mathbf{x}^{\mathcal{B}} - \mathbf{y}^{\mathcal{B}}\|^2}{(1 - \|\mathbf{x}^{\mathcal{B}}\|^2)(1 - \|\mathbf{y}^{\mathcal{B}}\|^2)}\right) \tag{44}$$

$$= \cosh^{-1}\left(1 + \frac{2\tanh^2(1)}{(1 - \tanh^2(1))^2} d^2\right). \tag{45}$$

Since

$$\frac{\sinh^2(1)}{2} < \frac{2\tanh^2(1)}{(1 - \tanh^2(1))^2}, \tag{46}$$

it holds that $d_{\mathcal{L}}(\mathbf{x}^{\mathcal{L}},\mathbf{y}^{\mathcal{L}}) < d_{\mathcal{B}}(\mathbf{x}^{\mathcal{B}},\mathbf{y}^{\mathcal{B}})$ unless $d = 0$, where $d_{\mathcal{L}}(\mathbf{x}^{\mathcal{L}},\mathbf{y}^{\mathcal{L}}) = d_{\mathcal{B}}(\mathbf{x}^{\mathcal{B}},\mathbf{y}^{\mathcal{B}}) = 0$. □

C More Details and Results

C.1 Datasets and Baselines

Dataset We assess our model's performance across the datasets in Table 1, which includes citation networks comprising Cora [51], Citeseer [51], PubMed [52], and obgn-arxiv [53]. We also incorporate Amazon Computers (Amazon) [54] and the image-sharing social network Flickr [55]. Additionally, we use the Wikipedia Network datasets [56], namely Squirrel and Chameleon, along with the Actor Co-occurrence Network dataset [57] (Actor).

Baselines We compare our models (denoted as Euclidean, Lorentz, Poincaré) with benchmark models for outlier node detection, as organized in previous benchmarking research [9]. The benchmark models include LOF [17], IF [18], MLPAE [10], SCAN [19], Radar [12], ANOMALOUS [13], GCNAE [11], DOMINANT [5], DONE [15], AdONE [15], AnomalyDAE [6], GAAN [7], and CONAD [16]. The omission of the GUIDE model [8] from our comparative analysis is attributed to its substantial memory consumption. Additionally, we have implemented a GIN-based [58] model by replacing the GCN used in GCNAE with GIN. We call this new model GINAE.

Parameters During training, we used a learning rate of 0.005, dropout 0.1, regularization weight 0.001, and hidden dimensions 32. The number of trained epochs for Squirrel, Chameleon, Actor, Cora, and Citeseer datasets is set to 300. For the PubMed, Flickr, and ogbn-arxiv datasets, we used a batch size of 1028 in training our models, and trained 100, 20, 10 epochs respectively. For the training of benchmarking models on the three datasets, we took the batch size to be 128 and epoch number to be 10, 2, 1 respectively, since the sampling of neighbors of certain models restricts the batch size from being too large. For the special hyperparameters of certain models, we referenced the choices in [9]. We took $\epsilon = 0.5$ and $\mu = 5$ for SCAN, $\theta = 40$ and $\eta = 5$ for AnomalyDAE, and the noise dimension to be 16 for GAAN. For models that incorporated both contextual and structural loss, we set $\alpha = 0.5$ as the hyperparameter to balance the weights of the two losses.

We implemented the injection of contextual and structural outliers, as well as the benchmarking models, using code sourced from `https://github.com/pygod-team/pygod/`. For construction of hyperbolic models, we utilized code available at `https://github.com/HazyResearch/hgcn`.

C.2 Outlier Injection Details

In Table 8, we provide the specific values of the number of injected outliers o, which are taken to be approximately 5% of $n_{\mathcal{V}}$. We take $p=0.2$ for structural outliers, same as in [9], and $r=0.5$ for DICE-n outliers in all datasets. For structural outlier injection, the values of the parameter s are given in Table 8, and $t=o/m$. s is set to be around twice the degree of graph, except in Squirrel and Chameleon datasets, where we adopt a smaller s to avoid t being too smaller. For contextual and "path" outlier injections, we take the parameter $q=s$.

Table 8: Parameters for outlier injection.

	Squirrel	Chameleon	Actor	Cora	Citeseer	Amazon	PubMed	Flickr	ogbn-arxiv
$n_{\mathcal{V}}$	5,201	2,277	7,600	2,708	3,312	13,752	19,717	89,250	169,343
Degree	76.3	27.6	7.0	3.9	2.8	35.8	4.5	10.1	13.7
o	280	120	390	140	180	700	1,000	4,480	8,400
s	70	30	15	10	10	70	10	20	30

C.3 Outlier Detection Results Using Norm Information

In Tables 9–12, we report the ROC-AUC and AP results in outlier detection using the score function $\mathrm{score}_{\mathrm{norm}}$ defined in Section 3.1.

Table 9: Outlier node detection results using the score function $\mathrm{score}_{\mathrm{norm}}(i)=\|\widetilde{\mathbf{x}}_i\|$. Mean and standard deviation of ROC-AUC (%) taken over 3 trials are reported.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
Squirrel	72.7±0.5	97.3±0.0	50.8±1.5	56.7±0.8	60.5±2.3	50.3±1.3
Chameleon	72.3±1.5	94.5±0.0	49.0±2.5	52.8±1.1	54.0±0.6	47.3±2.6
Actor	71.5±0.7	91.7±0.1	48.7±0.5	50.9±1.5	50.4±0.9	49.9±1.1
Cora	68.8±0.4	90.0±0.9	49.5±3.0	49.6±2.3	48.9±2.1	50.0±1.1
Citeseer	70.9±1.1	89.7±0.7	49.5±1.2	50.5±0.8	50.7±0.8	49.3±1.1
Amazon	74.6±0.8	98.5±0.0	50.4±0.8	53.5±1.0	56.6±0.8	49.9±0.4
PubMed	69.9±0.1	90.6±0.2	50.5±0.3	52.7±0.6	56.1±0.3	50.8±0.7
Flickr	72.1±0.1	94.4±0.2	49.9±0.1	49.4±0.2	49.4±0.0	49.6±0.3
ogbn-arxiv	72.5±0.2	95.2±0.0	50.3±0.1	48.0±0.1	45.8±0.2	50.1±0.2

Table 10: Outlier node detection results using the score function $\mathrm{score}_{\mathrm{norm}}(i)=\|\widetilde{\mathbf{x}}_i\|$. Mean and standard deviation of AP (%) taken over 3 trials are reported.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
Squirrel	23.5±0.2	52.0±0.5	5.6±0.2	6.4±0.1	6.9±0.5	5.5±0.2
Chameleon	16.2±0.6	33.7±0.3	5.4±0.4	5.7±0.0	5.7±0.1	5.0±0.4
Actor	13.5±0.2	28.4±0.3	5.1±0.0	5.2±0.2	5.2±0.1	5.2±0.1
Cora	13.1±1.1	26.8±1.2	5.7±0.9	5.0±0.3	4.6±0.2	5.3±0.4
Citeseer	14.0±1.6	27.7±1.1	5.7±0.4	5.5±0.1	5.7±0.2	5.3±0.2
Amazon	33.4±1.1	69.3±0.7	5.3±0.2	5.7±0.1	6.2±0.2	5.0±0.1
PubMed	13.4±0.1	27.1±0.1	5.1±0.1	5.6±0.1	6.3±0.1	5.3±0.0
Flickr	18.5±0.1	39.9±0.5	5.1±0.0	4.9±0.1	4.9±0.0	5.0±0.0
ogbn-arxiv	22.6±0.2	49.9±0.2	5.0±0.0	4.7±0.0	4.4±0.0	5.0±0.0

Table 11: Outlier node detection results using the score function $\mathrm{score}_{\mathrm{norm}}(i)=\|\widetilde{\mathbf{a}}_i\|_1$. Mean and standard deviation of ROC-AUC (%) taken over 3 trials are reported.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
Squirrel	67.1±1.2	50.2±1.0	82.7±0.6	50.0±0.5	50.8±0.8	51.1±2.9
Chameleon	67.7±1.6	51.2±2.5	86.9±0.5	49.0±3.3	50.7±2.1	49.3±2.1
Actor	73.9±0.6	48.9±2.0	96.3±0.1	47.7±2.2	49.4±2.3	50.0±0.5
Cora	72.9±1.3	50.4±2.7	95.8±0.2	50.6±2.0	54.1±1.8	50.9±1.9
Citeseer	74.5±1.5	49.8±1.0	96.4±0.0	55.5±0.9	60.2±0.7	50.1±0.2
Amazon	69.9±0.5	50.7±0.2	91.0±0.1	51.0±1.0	51.9±0.4	50.7±0.2
PubMed	68.4±0.3	50.5±0.6	87.9±0.2	49.4±0.3	49.9±1.0	51.0±0.5
Flickr	72.3±0.3	50.0±0.3	94.8±0.0	49.5±0.2	49.8±0.4	49.7±0.5
ogbn-arxiv	73.4±0.1	50.2±0.1	96.7±0.0	50.0±0.2	49.9±0.3	50.0±0.3

Table 12: Outlier node detection results using the score function $\mathrm{score}_{\mathrm{norm}}(i) = \|\widetilde{\mathbf{a}}_i\|_1$. Mean and standard deviation of AP (%) taken over 3 trials are reported.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
Squirrel	9.9±0.9	5.5±0.1	16.3±1.2	5.5±0.1	5.6±0.2	5.7±0.5
Chameleon	11.8±1.0	5.4±0.3	20.5±2.3	5.6±0.3	5.5±0.6	5.3±0.2
Actor	18.6±0.5	5.1±0.2	41.4±1.2	5.0±0.2	5.1±0.3	5.3±0.1
Cora	17.4±1.2	5.3±0.3	37.6±1.2	5.7±0.5	5.8±0.2	5.4±0.2
Citeseer	20.2±0.9	5.7±0.3	42.2±0.2	6.9±0.1	6.9±0.2	5.4±0.1
Amazon	11.3±0.1	5.2±0.0	21.2±0.2	5.3±0.1	5.3±0.1	5.3±0.1
PubMed	9.5±0.1	5.1±0.0	17.3±0.2	5.0±0.0	5.2±0.2	5.2±0.1
Flickr	14.5±0.1	5.0±0.0	31.2±0.0	5.0±0.0	5.0±0.0	5.0±0.1
ogbn-arxiv	19.6±0.2	5.0±0.0	43.3±0.1	5.0±0.0	4.9±0.0	5.0±0.0

C.4 Running Time and Complexity Analysis

Running time In Table 13, we report the running time for the outlier detection models to train 1 epoch on the Cora dataset. Time consumption for IF, LOF, SCAN algorithms are not included because they are non-iterative.

Table 13: Time consumption (ms) of outlier detection models to train 1 epoch on the Cora dataset. Mean and standard deviation taken over 10 trials are reported.

Model	Time(ms)
MLPAE	3.8±0.2
Radar	3.2±0.1
ANOMALOUS	4.0±0.2
GCNAE	9.6±0.3
DOMINANT	17.3±0.8
DONE	17.9±1.1
AdONE	20.6±0.3
AnomalyDAE	23.5±3.2
GAAN	10.5±0.7
CONAD	25.8±0.7
Ours (Euclidean)	22.2±8.3
Ours (Lorentz)	51.1±8.5
Ours (Poincaré)	58.7±7.1

Complexity analysis We conducted complexity analysis on our models in the Euclidean and hyperbolic spaces. Let n_V denote the number of nodes. For simplicity we assume there are L layers, each with the same input and output dimensionality, denoted as n.

1. Euclidean model
   - Linear layer: each linear layer of the Euclidean model consists of feature transformation, bias addition, and activation as defined in Appendix A.3. The time complexity is $O(n_V \cdot n^2)$, and space complexity is $O(n_V \cdot n + n^2)$.
   - Centralization layer: each centralization layer involves of calculation of mean values and translation according to the mean. The time complexity is $O(n_V \cdot n)$ and space complexity is $O(n_V \cdot n)$.
   - Loss and score function: since we took the score function to be the same as the loss function, they have the same complexity. In this step, the calculation of distance matrix in Equation (19) has time complexity $O(n_V^2 \cdot n)$ and space complexity $O(n_V^2 + n_V \cdot n)$. The computing of structural loss in Equation (7) has time complexity $O(n_V^2)$ and space complexity $O(n_V^2)$.

   Therefore, for the Euclidean model, the time complexity is $O(n_V \cdot n^2 \cdot L + n_V^2 \cdot n)$ and the space complexity is $O(n_V^2 + n_V \cdot n \cdot L + n^2 \cdot L)$.
2. Lorentz model
   - Exponential map: the data is first exponentially mapped to the Lorentz model $\mathbb{L}$ at the origin according to Equation (11). This step has time complexity $O(n_V \cdot n)$ and space complexity $O(n_V \cdot n)$.
   - Linear layer: for the fully hyperbolic linear layer for the Lorentz model given by Equation (1), the time complexity is $O(n_V \cdot n^2)$, and space complexity is $O(n_V \cdot n + n^2)$.
   - Centralization layer: the centralization layer defined in Equation (2) consists of the calculation of centroid and parallel transport. The calculation of centroid in Equation (2) has time complexity $O(n_V \cdot n)$ and space complexity $O(n_V \cdot n)$, and the parallel transport defined in Equation (12) has time complexity $O(n_V \cdot n)$ and space complexity $O(n_V \cdot n)$.

- Loss and score function: for the computing of distance matrix in Equation (21), the time complexity is $O(n_V^2 \cdot n)$ and the space complexity is $O(n_V^2 + n_V \cdot n)$. The computing of structural loss is the same as the Euclidean model, so it has time complexity $O(n_V^2)$ and space complexity $O(n_V^2)$.

Therefore, for the Lorentz model, the time complexity is $O(n_V \cdot n^2 \cdot L + n_V^2 \cdot n)$ and the space complexity is $O(n_V^2 + n_V \cdot n \cdot L + n^2 \cdot L)$.

3. Poincaré model

- Exponential map: the data is exponentially mapped to the Poincaré model at the origin according to Equation (17), which has time complexity $O(n_V \cdot n)$ and space complexity $O(n_V \cdot n)$.
- Linear layer: the linear layer defined in Equation (3) consists of the Möbius matrix-vector multiplication, bias addition, and activation. The Möbius matrix-vector multiplication given by Equation (16) has time complexity $O(n_V \cdot n^2)$ and space complexity $O(n_V \cdot n + n^2)$. The bias addition involves parallel transport in Equation (18) and exponential map, which have time complexity $O(n_V \cdot n)$ and space complexity $O(n_V \cdot n)$. The activation is composed of logarithmic mapping in Equation (17), applying activation on the tangent space, and exponential mapping, which has time complexity $O(n_V \cdot n)$ and space complexity $O(n_V \cdot n)$.
- Centralization layer: the centralization layer defined in Equation (4) consists of logarithmic mapping, applying the same centralization layer as the Euclidean model on the tangent space, and exponential mapping. This step has time complexity $O(n_V \cdot n)$ and space complexity $O(n_V \cdot n)$.
- Loss and score function: The computing of distance matrix in Equation (22) has time complexity $O(n_V^2 \cdot n)$ and space complexity $O(n_V^2 + n_V \cdot n)$. We calculated the structural loss in the same way as the Euclidean model, which has time complexity $O(n_V^2)$ and space complexity $O(n_V^2)$.

Therefore, Poincaré model has time complexity is $O(n_V \cdot n^2 \cdot L + n_V^2 \cdot n)$ and space complexity $O(n_V^2 + n_V \cdot n \cdot L + n^2 \cdot L)$.

Overall, for all the three models that we consider, the time complexity is $O(n_V \cdot n^2 \cdot L + n_V^2 \cdot n)$ and the space complexity is $O(n_V^2 + n_V \cdot n \cdot L + n^2 \cdot L)$. We note that complexities are of the same order for Euclidean and hyperbolic models, but there is a scalar difference that results in the longer running time of hyperbolic models.

C.5 Additional Benchmarking Results

In Tables 14–16, we report AP results of models in outlier detection in datasets Cora, Squirrel, and Amazon. In Tables 17–28, we report ROC-AUC and AP results of outlier detection in Chameleon, Actor, Citeseer, PubMed, Flickr, and ogbn-arxiv.

Table 14: Mean and standard deviation of AP (%) scores on Cora dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
LOF	4.5±0.1	4.6±0.8	6.0±0.2	5.3±0.2	5.2±0.3	5.3±0.2
IF	3.7±0.1	3.0±0.0	5.4±0.3	5.5±0.5	5.8±0.6	5.4±0.2
MLPAE	7.9±0.6	13.5±0.9	5.6±1.0	5.5±0.2	6.1±0.2	5.9±0.8
SCAN	16.8±0.7	5.3±0.2	**42.7**±1.3	5.2±0.0	5.1±0.0	5.1±0.0
Radar	6.2±0.6	6.3±0.9	5.8±0.4	4.6±0.3	4.6±0.2	4.8±0.2
ANOMALOUS	5.0±0.3	5.3±0.2	6.0±0.1	4.5±0.1	4.5±0.1	4.6±0.1
GCNAE	9.5±1.2	**15.6**±1.4	5.5±0.5	5.9±0.2	5.8±0.3	5.4±0.4
DOMINANT	17.0±0.7	5.4±0.1	31.8±0.9	5.5±0.3	5.0±0.2	5.0±0.6
DONE	**20.6**±7.8	5.9±0.3	38.7±3.8	5.5±0.7	7.5±0.4	5.0±0.2
AdONE	15.3±0.8	6.5±0.3	31.3±1.5	5.5±0.5	6.3±0.7	4.9±0.2
AnomalyDAE	11.4±0.1	5.8±0.1	20.5±0.4	5.6±0.3	5.7±0.2	5.2±0.4
GAAN	5.5±0.3	5.0±0.2	5.6±0.5	5.6±0.7	5.1±0.3	5.1±0.1
CONAD	15.6±0.4	6.0±0.3	32.0±1.2	5.6±0.7	5.7±0.7	4.5±0.1
GINAE	7.5±0.1	11.0±2.3	7.2±0.8	5.6±0.1	6.4±1.4	5.4±0.5
Ours (Euclidean)	19.8±2.4	13.0±0.6	23.6±6.2	27.6±3.4	16.0±2.0	42.2±2.9
Ours (Lorentz)	10.5±1.9	10.0±0.7	12.9±0.4	14.3±2.0	9.4±0.4	16.9±1.3
Ours (Poincaré)	19.9±2.5	15.3±0.8	26.7±1.0	**34.2**±0.9	**17.7**±2.6	**57.6**±1.7

Table 15: Mean and standard deviation of AP (%) scores on Squirrel dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
LOF	3.7±0.1	3.2±0.0	5.4±0.2	3.9±0.1	3.4±0.0	5.3±0.1
IF	3.8±0.1	3.5±0.1	5.6±0.4	6.2±0.2	7.4±0.4	5.5±0.4
MLPAE	17.3±0.6	33.0±1.6	5.4±0.0	5.2±0.3	4.6±0.0	5.5±0.2
SCAN	8.5±0.2	5.6±0.2	14.6±0.2	5.0±0.0	5.3±0.1	5.0±0.0
Radar	7.0±0.2	8.0±0.8	6.2±0.1	4.6±0.1	5.5±0.3	4.3±0.0
ANOMALOUS	5.9±0.4	8.0±0.9	5.9±0.3	4.7±0.1	5.8±0.1	4.3±0.1
GCNAE	17.0±1.2	**36.5**±1.6	5.8±0.4	4.9±0.3	4.7±0.0	5.8±0.5
DOMINANT	10.9±0.6	5.8±0.1	19.3±0.7	6.4±0.8	5.9±0.1	6.1±0.3
DONE	14.8±0.6	5.8±0.3	24.8±0.7	6.6±0.4	5.6±0.2	6.9±1.2
AdONE	**30.5**±3.5	5.9±0.2	**64.7**±1.0	11.1±0.9	5.7±0.2	19.9±2.8
AnomalyDAE	17.0±7.8	6.0±0.5	18.7±1.8	8.0±3.9	6.2±1.0	12.0±8.7
GAAN	6.4±0.6	5.3±0.4	7.2±0.2	6.1±0.2	5.6±0.1	6.7±0.1
CONAD	NaN	NaN	NaN	NaN	NaN	NaN
GINAE	8.4±1.6	14.1±3.4	6.4±0.6	5.6±0.2	6.2±0.8	5.9±0.4
Ours (Euclidean)	18.1±9.0	5.4±0.0	5.9±0.2	17.8±17.5	15.5±14.3	5.3±0.1
Ours (Lorentz)	25.7±5.2	22.6±2.5	28.2±2.4	35.7±2.0	**34.7**±5.6	35.2±5.4
Ours (Poincaré)	24.9±2.2	33.4±3.9	18.6±1.5	**49.0**±3.2	**34.7**±1.6	**62.7**±1.3

Table 16: Mean and standard deviation of AP (%) scores on Amazon dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
LOF	3.3±0.1	2.6±0.0	5.0±0.1	4.5±0.1	3.9±0.0	5.4±0.3
IF	3.3±0.1	2.6±0.0	5.2±0.1	5.1±0.1	4.7±0.1	5.0±0.1
MLPAE	31.9±1.0	66.3±1.1	5.2±0.1	5.8±0.2	6.6±0.1	5.1±0.1
SCAN	10.1±0.2	5.2±0.1	22.0±0.1	4.9±0.0	5.1±0.0	4.9±0.0
Radar	6.0±0.3	4.8±0.1	8.1±0.4	4.5±0.1	5.3±0.0	4.5±0.2
ANOMALOUS	5.8±0.1	5.3±0.4	7.6±0.1	4.6±0.2	4.9±0.0	4.4±0.0
GCNAE	32.7±1.1	**68.4**±0.6	5.2±0.1	5.7±0.1	6.4±0.4	5.2±0.0
DOMINANT	13.1±0.4	5.5±0.1	23.4±0.7	5.3±0.1	5.7±0.2	5.3±0.1
DONE	15.9±1.8	11.2±0.6	19.9±2.1	5.6±0.2	5.6±0.1	5.1±0.1
AdONE	17.1±0.4	10.2±1.0	23.9±0.5	5.6±0.3	6.6±0.7	5.2±0.1
AnomalyDAE	13.0±0.1	6.5±0.1	23.3±0.6	5.4±0.4	5.4±0.2	5.0±0.1
GAAN	5.7±0.1	4.8±0.1	7.3±0.3	5.4±0.1	5.5±0.1	5.6±0.1
CONAD	12.6±0.5	5.4±0.0	23.8±0.3	5.4±0.2	5.6±0.3	5.3±0.2
GINAE	14.7±1.9	33.0±13.6	9.0±0.7	6.0±0.2	5.6±0.5	6.5±1.2
Ours (Euclidean)	44.3±3.9	35.5±2.1	62.6±2.3	49.8±0.6	40.9±2.4	64.5±3.2
Ours (Lorentz)	**50.9**±8.5	43.7±4.7	63.1±6.2	58.9±7.0	52.6±8.0	65.2±2.7
Ours (Poincaré)	49.9±2.4	35.4±2.3	**75.9**±1.7	**73.3**±1.4	**55.5**±4.2	**84.1**±1.3

Table 17: Mean and standard deviation of ROC-AUC (%) scores on Chameleon dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
LOF	34.0±0.8	19.1±0.8	50.8±3.2	35.7±0.3	25.2±0.7	49.8±2.2
IF	36.7±1.1	20.9±1.1	50.6±0.7	51.0±2.2	53.0±2.1	52.1±3.4
MLPAE	63.4±3.5	74.8±3.9	51.8±1.1	48.9±3.2	50.1±1.8	49.5±4.7
SCAN	64.1±1.9	49.0±0.0	81.2±0.2	45.3±0.9	49.0±1.6	41.8±0.5
Radar	59.8±1.4	51.5±3.7	70.3±2.7	48.1±2.5	51.8±1.1	49.3±0.3
ANOMALOUS	62.5±2.0	53.3±1.1	67.2±3.0	46.3±1.5	50.4±1.7	49.0±1.1
GCNAE	61.3±2.2	74.3±1.9	50.1±1.6	50.7±2.1	48.7±2.7	48.2±0.2
DOMINANT	73.7±0.6	51.9±3.4	91.6±0.2	52.6±2.8	50.8±2.6	53.3±4.5
DONE	72.0±1.6	50.9±3.2	89.6±0.8	56.0±1.3	56.4±3.8	52.8±2.3
AdONE	74.7±1.4	51.4±2.2	**97.0**±0.3	56.5±2.5	52.3±2.7	63.6±1.6
AnomalyDAE	58.2±4.5	67.6±1.2	55.4±6.8	67.5±0.3	63.1±5.7	72.2±3.3
GAAN	56.2±2.4	54.9±3.5	54.2±3.2	52.8±2.2	53.3±1.7	53.5±2.4
CONAD	NaN	NaN	NaN	NaN	NaN	NaN
Ours (Euclidean)	86.5±0.8	79.8±0.6	95.1±0.8	80.3±2.0	76.6±2.9	91.4±0.8
Ours (Lorentz)	86.0±1.6	77.7±2.9	96.7±0.6	82.9±2.9	74.5±1.8	91.4±1.6
Ours (Poincaré)	**88.9**±1.8	**84.2**±1.5	93.9±1.1	**87.4**±2.2	**83.0**±2.5	**92.3**±0.3

Table 18: Mean and standard deviation of AP (%) scores on Chameleon dataset.

	cntxt.+strct.	**cntxt.**	**strct.**	**"path"+DICE**	**"path"**	**DICE**
LOF	3.8±0.1	3.4±0.0	5.5±0.5	4.0±0.0	4.1±0.2	5.2±0.2
IF	4.2±0.1	3.8±0.1	5.5±0.1	5.8±0.6	5.5±0.4	5.8±0.4
MLPAE	8.9±1.1	13.7±1.6	6.1±0.4	5.1±0.3	5.1±0.2	5.3±0.7
SCAN	7.7±0.4	5.2±0.0	13.0±0.1	4.9±0.0	5.2±0.1	4.8±0.0
Radar	6.7±0.3	7.7±2.0	8.2±0.6	5.0±0.3	5.7±0.3	5.2±0.4
ANOMALOUS	7.0±0.7	7.5±0.8	7.6±0.5	4.8±0.4	5.3±0.1	4.9±0.1
GCNAE	8.5±0.7	14.5±1.5	5.4±0.4	5.8±0.8	4.9±0.2	5.4±0.5
DOMINANT	16.9±2.9	5.8±0.6	28.7±1.4	6.4±0.6	6.0±0.8	7.7±1.7
DONE	15.6±3.3	6.3±1.0	20.5±1.7	6.8±0.4	10.1±3.9	5.9±0.2
AdONE	27.2±2.9	5.6±0.1	**54.7**±3.3	10.8±1.0	6.3±0.7	17.8±5.2
AnomalyDAE	14.8±2.2	16.9±1.6	18.1±2.9	16.2±1.6	12.2±4.7	22.0±2.5
GAAN	6.3±0.5	6.0±0.7	6.5±0.8	6.2±0.7	6.0±0.5	5.9±0.6
CONAD	NaN	NaN	NaN	NaN	NaN	NaN
Ours (Euclidean)	**34.2**±3.0	27.7±0.8	44.9±1.0	40.8±3.4	33.3±7.5	56.2±4.9
Ours (Lorentz)	32.2±6.4	28.0±7.0	44.7±4.0	34.4±4.0	24.2±4.8	47.2±3.7
Ours (Poincaré)	32.6±4.0	**32.5**±3.4	39.8±3.5	**48.5**±0.9	**35.9**±1.8	**59.8**±2.4

Table 19: Mean and standard deviation of ROC-AUC (%) scores on Actor dataset.

	cntxt.+strct.	**cntxt.**	**strct.**	**"path"+DICE**	**"path"**	**DICE**
LOF	40.4±2.0	31.5±2.0	51.1±0.3	48.5±1.4	48.1±0.9	48.5±3.3
IF	37.1±2.0	24.6±4.8	50.6±1.1	47.8±1.2	51.1±1.4	51.3±0.6
MLPAE	55.7±3.6	49.5±1.6	51.6±1.3	49.1±0.3	51.6±1.2	49.0±1.1
SCAN	74.6±0.3	50.0±0.2	**98.9**±0.3	50.1±0.1	49.9±0.1	49.9±0.1
Radar	71.7±1.1	66.3±1.5	73.6±1.1	49.0±0.9	51.0±0.5	44.9±0.8
ANOMALOUS	69.3±0.9	66.8±1.3	72.8±0.9	48.5±1.1	47.4±0.8	45.4±0.9
GCNAE	61.9±0.5	**72.9**±1.5	50.0±0.3	49.2±1.2	50.7±0.8	50.4±0.7
DOMINANT	73.5±0.7	51.9±2.7	92.0±0.0	50.5±1.8	51.2±2.4	46.3±1.0
DONE	**77.1**±0.4	56.8±3.9	95.5±0.1	48.2±0.4	49.8±1.2	48.2±2.5
AdONE	73.8±1.2	52.7±1.0	90.8±0.7	47.5±1.4	51.1±2.5	48.0±0.5
AnomalyDAE	67.5±0.6	48.3±0.8	85.9±0.4	48.6±1.4	48.4±0.9	44.5±0.2
GAAN	50.5±0.6	52.4±1.6	53.0±0.7	51.2±0.6	48.6±0.9	50.3±1.1
CONAD	71.9±0.7	52.7±0.6	92.2±0.1	48.0±1.5	50.4±0.8	46.6±0.4
Ours (Euclidean)	55.2±1.6	57.1±1.0	60.1±0.6	51.9±1.0	52.7±0.9	52.2±0.7
Ours (Lorentz)	62.7±3.9	47.9±3.5	77.1±0.8	53.9±0.2	50.7±0.8	**55.8**±2.3
Ours (Poincaré)	60.1±2.6	56.2±0.6	66.6±1.5	**55.2**±2.3	**56.8**±1.0	55.6±1.8

Table 20: Mean and standard deviation of AP (%) scores on Actor dataset.

	cntxt.+strct.	**cntxt.**	**strct.**	**"path"+DICE**	**"path"**	**DICE**
LOF	4.3±0.2	4.1±0.1	5.3±0.1	4.9±0.2	4.8±0.1	5.1±0.4
IF	4.0±0.2	3.5±0.2	5.2±0.3	4.8±0.2	5.4±0.2	5.6±0.2
MLPAE	8.4±0.9	11.3±0.3	5.5±0.2	5.1±0.0	5.4±0.2	5.0±0.1
SCAN	**44.5**±0.6	5.1±0.0	**92.1**±1.4	5.1±0.0	5.1±0.0	5.1±0.0
Radar	9.6±0.6	12.3±1.2	8.5±0.3	5.2±0.3	5.4±0.3	4.8±0.2
ANOMALOUS	9.2±0.4	**12.6**±0.6	8.2±0.2	5.1±0.1	4.9±0.2	4.7±0.2
GCNAE	9.7±0.4	16.6±1.1	5.2±0.1	5.1±0.1	5.3±0.1	5.3±0.2
DOMINANT	12.7±0.6	5.3±0.4	23.3±0.0	5.1±0.2	5.4±0.4	4.5±0.1
DONE	25.5±2.0	7.5±3.0	41.4±1.8	4.9±0.1	5.4±0.3	4.7±0.3
AdONE	13.2±0.2	5.4±0.2	21.7±1.1	4.8±0.1	5.3±0.4	4.8±0.0
AnomalyDAE	9.5±0.1	5.1±0.1	14.9±0.3	4.9±0.2	4.9±0.2	4.3±0.0
GAAN	5.1±0.2	5.4±0.2	5.6±0.2	5.4±0.1	5.1±0.0	5.3±0.1
CONAD	12.3±0.2	5.4±0.1	23.6±0.3	4.9±0.2	5.3±0.1	4.5±0.1
Ours (Euclidean)	5.6±0.2	6.5±0.3	6.0±0.0	6.7±0.8	6.6±0.2	7.0±0.2
Ours (Lorentz)	7.1±1.0	5.0±0.6	9.9±0.3	6.1±0.3	5.5±0.3	6.9±1.1
Ours (Poincaré)	6.2±0.5	6.3±0.3	6.9±0.2	**8.4**±0.9	**7.7**±0.4	**8.4**±0.7

Table 21: Mean and standard deviation of ROC-AUC (%) scores on Citeseer dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
LOF	56.1±1.2	58.6±1.2	50.9±1.7	48.4±1.0	44.9±1.7	50.1±2.1
IF	43.1±1.2	36.4±1.3	50.4±0.9	47.5±1.4	51.8±0.9	52.3±1.8
MLPAE	60.1±1.1	69.7±1.5	51.3±1.8	48.8±2.9	47.4±0.8	48.2±1.0
SCAN	74.1±0.0	50.3±1.2	**97.9**±0.2	50.4±0.7	51.2±0.3	49.7±0.5
Radar	47.0±0.1	54.6±3.0	45.8±0.4	39.4±3.0	32.1±0.6	41.3±0.2
ANOMALOUS	43.6±0.8	49.2±2.4	38.5±0.1	34.7±0.7	30.3±0.5	37.2±0.8
GCNAE	59.8±2.2	**71.8**±2.0	50.4±2.1	50.5±1.5	48.2±1.9	49.7±0.5
DOMINANT	75.4±1.3	53.8±1.3	95.9±0.1	52.9±3.3	58.1±1.7	48.1±2.4
DONE	74.1±1.8	56.7±0.9	94.3±0.1	53.2±0.9	61.8±1.0	47.1±2.8
AdONE	74.1±1.3	53.1±2.0	95.4±0.1	55.1±1.0	59.9±1.9	46.6±0.8
AnomalyDAE	74.8±2.0	55.7±0.9	92.5±0.2	51.2±1.6	59.8±0.9	48.8±1.6
GAAN	51.6±1.9	50.4±1.5	53.1±1.7	48.5±2.9	52.2±1.8	49.8±2.6
CONAD	75.4±0.7	54.9±2.3	96.0±0.1	53.3±2.0	55.7±1.2	47.7±1.5
Ours (Euclidean)	72.0±1.0	61.8±1.2	86.8±3.3	71.6±2.0	75.2±2.0	73.5±0.7
Ours (Lorentz)	72.4±1.1	60.7±3.3	85.1±1.2	71.3±1.4	75.8±0.7	73.7±2.0
Ours (Poincaré)	**77.8**±3.4	69.9±0.8	90.2±0.7	**78.8**±1.4	**82.4**±0.7	**78.3**±3.0

Table 22: Mean and standard deviation of AP (%) scores on Citeseer dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
LOF	6.5±0.4	7.8±0.4	5.7±0.3	5.2±0.1	4.8±0.2	5.6±0.3
IF	4.7±0.3	4.0±0.1	5.5±0.3	5.3±0.3	5.8±0.3	5.8±0.1
MLPAE	8.5±0.2	13.6±1.3	5.5±0.3	5.6±0.3	5.3±0.1	5.5±0.3
SCAN	**26.3**±0.4	5.5±0.2	**60.3**±0.8	5.5±0.1	5.6±0.1	5.4±0.0
Radar	5.7±0.1	9.6±0.6	4.7±0.0	4.1±0.2	3.7±0.0	4.2±0.0
ANOMALOUS	4.4±0.1	5.2±0.5	4.2±0.0	3.8±0.0	3.6±0.0	3.9±0.0
GCNAE	8.9±1.2	14.2±0.7	6.0±0.4	5.6±0.3	5.4±0.3	6.0±0.2
DOMINANT	18.9±0.7	6.0±0.3	39.2±1.0	6.0±0.7	7.2±0.1	5.1±0.3
DONE	17.1±2.0	7.5±0.5	31.3±0.4	6.3±0.3	9.7±1.9	5.6±0.9
AdONE	17.6±0.5	6.2±0.6	36.0±0.8	6.8±0.1	7.8±0.7	5.0±0.1
AnomalyDAE	14.7±1.5	6.5±0.3	26.3±0.5	5.7±0.1	7.6±0.4	5.1±0.2
GAAN	6.1±0.5	5.9±0.2	6.6±0.4	5.5±0.3	6.1±0.6	5.6±0.4
CONAD	18.9±0.8	6.1±0.4	39.9±0.5	6.0±0.4	6.6±0.3	5.0±0.2
Ours (Euclidean)	13.9±0.4	10.7±0.4	22.2±3.0	20.6±2.9	14.7±1.5	33.9±3.2
Ours (Lorentz)	13.0±0.8	9.1±0.8	19.1±2.0	14.1±0.9	14.9±1.1	22.6±4.1
Ours (Poincaré)	21.1±5.1	**14.7**±1.6	29.9±1.8	**29.5**±1.3	**20.1**±0.1	**40.6**±4.1

Table 23: Mean and standard deviation of ROC-AUC (%) scores on PubMed dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
LOF	45.1±0.3	39.0±1.3	49.3±1.3	49.8±0.5	48.8±0.9	48.9±0.8
IF	35.3±0.5	21.5±0.2	51.2±0.3	48.3±0.9	46.3±0.2	49.5±0.4
MLPAE	68.4±0.8	85.5±0.5	49.6±1.2	52.8±0.7	54.4±0.2	50.1±0.9
SCAN	74.2±0.2	50.0±0.1	**98.6**±0.2	50.0±0.1	49.9±0.1	50.0±0.0
Radar	58.1±0.7	54.0±0.4	62.9±0.7	50.7±1.1	50.2±0.7	51.2±0.6
ANOMALOUS	58.2±1.2	53.4±1.2	64.0±0.6	50.7±0.4	49.7±0.2	52.3±0.4
GCNAE	68.0±0.8	85.9±0.8	50.1±0.1	52.5±0.8	55.6±0.8	50.7±0.7
DOMINANT	75.8±0.5	60.0±0.5	92.3±0.1	40.5±0.7	52.5±1.2	29.1±0.2
DONE	69.0±0.1	57.0±1.4	81.9±0.1	49.5±0.7	48.4±2.5	49.5±1.4
AdONE	70.6±1.2	52.0±0.1	87.9±0.1	50.7±0.9	50.6±0.4	50.1±0.5
AnomalyDAE	69.0±0.5	59.3±1.3	73.6±2.3	41.5±4.1	56.0±2.1	22.4±0.6
GAAN	50.0±1.4	49.8±0.5	50.4±0.2	49.8±0.4	50.2±0.8	50.2±0.7
CONAD	71.1±0.9	63.5±0.2	79.2±0.4	45.5±0.2	52.5±1.1	41.2±0.7
Ours (Euclidean)	89.1±0.7	85.6±0.9	94.9±0.4	88.0±1.3	88.8±0.5	89.1±0.2
Ours (Lorentz)	**91.4**±0.0	**87.3**±0.4	94.8±0.1	90.6±0.2	89.6±0.2	90.7±0.5
Ours (Poincaré)	91.2±0.6	86.9±1.1	94.7±0.1	**91.3**±0.2	**90.6**±0.2	**92.0**±0.2

Table 24: Mean and standard deviation of AP (%) scores on PubMed dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
LOF	4.7±0.1	4.2±0.2	5.1±0.3	5.1±0.0	4.9±0.1	5.0±0.2
IF	3.7±0.0	3.0±0.0	5.3±0.1	4.8±0.1	4.6±0.1	5.0±0.0
MLPAE	12.1±0.6	21.9±0.3	5.0±0.2	5.6±0.1	5.9±0.2	5.2±0.1
SCAN	**46.3**±0.7	5.1±0.0	**92.5**±1.2	5.1±0.0	5.1±0.0	5.1±0.0
Radar	5.6±0.1	5.7±0.2	6.1±0.1	4.8±0.1	5.1±0.0	4.7±0.1
ANOMALOUS	5.6±0.2	5.7±0.2	6.2±0.1	4.8±0.0	5.0±0.1	4.8±0.0
GCNAE	11.4±0.4	21.9±1.7	5.2±0.1	5.7±0.1	6.2±0.1	5.2±0.1
DOMINANT	13.0±0.3	6.1±0.1	24.2±0.1	4.2±0.1	5.3±0.2	3.4±0.0
DONE	8.7±0.0	6.3±0.4	12.1±0.2	4.9±0.1	4.8±0.2	5.2±0.1
AdONE	10.0±0.4	5.3±0.0	16.6±0.2	5.2±0.2	5.1±0.1	5.1±0.1
AnomalyDAE	11.4±1.7	7.1±0.4	11.9±3.1	4.4±0.6	6.3±0.6	3.0±0.0
GAAN	5.3±0.2	4.9±0.1	5.4±0.2	5.1±0.1	5.0±0.1	5.2±0.1
CONAD	10.3±0.5	6.9±0.0	15.4±0.5	4.4±0.0	5.5±0.2	4.0±0.1
Ours (Euclidean)	30.1±1.3	28.0±1.1	37.6±1.4	28.5±2.1	35.8±0.6	26.9±0.7
Ours (Lorentz)	34.8±0.6	29.1±0.8	37.7±1.0	32.2±0.7	35.7±0.8	28.3±1.2
Ours (Poincaré)	34.2±0.6	**29.6**±1.3	37.7±1.0	**33.9**±0.4	**38.4**±0.5	**31.6**±0.1

Table 25: Mean and standard deviation of ROC-AUC (%) scores on Flickr dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
LOF	58.9±0.2	68.6±0.6	49.9±0.4	48.1±0.7	47.0±0.5	50.1±0.5
IF	29.1±0.3	8.5±0.1	50.3±0.2	50.7±0.2	51.5±0.2	50.2±0.2
MLPAE	71.8±0.1	**93.2**±0.3	50.1±0.1	50.0±0.5	49.7±0.3	49.7±0.5
SCAN	74.4±0.1	50.0±0.0	**98.8**±0.1	50.0±0.0	50.0±0.0	50.0±0.0
Radar	OOM[2]	OOM	OOM	OOM	OOM	OOM
ANOMALOUS	OOM	OOM	OOM	OOM	OOM	OOM
GCNAE	71.7±0.1	92.9±0.3	50.0±0.7	49.8±0.1	49.4±0.1	50.3±0.2
DOMINANT	79.5±0.1	61.1±0.6	97.5±0.0	31.6±0.5	50.1±0.1	13.8±0.1
DONE	**83.0**±2.3	65.7±13.0	83.4±0.3	49.1±0.5	49.5±0.8	49.9±0.4
AdONE	64.7±0.3	36.2±1.8	90.8±0.7	50.1±0.2	50.5±0.4	50.2±0.4
AnomalyDAE	63.1±0.9	72.3±1.9	48.5±0.4	31.0±1.0	49.7±0.2	8.9±1.6
GAAN	50.1±0.5	50.2±0.1	50.1±0.2	48.3±0.5	50.2±0.2	46.9±0.2
CONAD	65.9±0.5	65.3±0.1	66.5±0.4	44.3±0.2	49.6±0.4	39.2±0.1
Ours (Euclidean)	62.2±6.2	45.3±2.5	73.3±14.3	50.8±0.6	50.6±0.5	49.9±1.4
Ours (Lorentz)	60.8±4.0	51.2±7.6	68.6±0.3	**54.5**±0.4	**55.5**±0.7	**53.2**±0.9
Ours (Poincaré)	57.8±4.9	54.4±6.9	68.0±1.1	53.3±0.2	52.6±1.0	**53.2**±0.6

Table 26: Mean and standard deviation of AP (%) scores on Flickr dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
LOF	8.6±0.1	14.2±0.2	5.0±0.1	4.8±0.1	4.6±0.1	5.0±0.1
IF	3.3±0.0	2.6±0.0	5.1±0.0	5.1±0.0	5.2±0.0	5.1±0.0
MLPAE	17.6±0.4	**36.6**±0.5	5.1±0.0	5.0±0.1	4.9±0.1	5.0±0.0
SCAN	**49.0**±0.2	5.0±0.0	**95.2**±0.1	5.0±0.0	5.0±0.0	5.0±0.0
Radar	OOM	OOM	OOM	OOM	OOM	OOM
ANOMALOUS	OOM	OOM	OOM	OOM	OOM	OOM
GCNAE	17.5±0.2	35.7±0.3	5.0±0.1	5.0±0.0	5.0±0.0	5.1±0.1
DOMINANT	29.5±0.3	6.7±0.1	59.7±0.2	3.5±0.0	5.1±0.0	2.8±0.0
DONE	17.7±3.3	9.6±4.7	14.8±0.6	4.9±0.1	5.0±0.2	5.0±0.1
AdONE	13.8±1.6	4.2±0.2	29.4±2.2	5.0±0.0	5.1±0.0	5.1±0.0
AnomalyDAE	9.5±0.3	11.5±0.7	6.1±0.1	3.5±0.0	5.0±0.0	2.6±0.0
GAAN	5.0±0.0	5.1±0.0	5.1±0.0	4.8±0.1	5.0±0.0	4.6±0.1
CONAD	8.5±0.2	7.6±0.0	9.8±0.1	4.3±0.0	5.0±0.1	3.8±0.0
Ours (Euclidean)	8.7±3.1	4.6±0.5	14.3±10.9	5.1±0.0	5.1±0.1	5.0±0.2
Ours (Lorentz)	6.8±0.7	5.5±1.2	8.2±0.2	**5.7**±0.0	**6.0**±0.2	5.4±0.1
Ours (Poincaré)	6.5±1.0	6.2±1.4	8.4±0.6	5.4±0.0	5.5±0.2	**5.5**±0.1

[2]OOM denotes out of memory error.

Table 27: Mean and standard deviation of ROC-AUC (%) scores on ogbn-arxiv dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
LOF	51.6±0.2	51.7±0.2	49.9±0.1	51.6±0.5	52.2±0.2	49.6±0.5
IF	62.7±0.4	76.0±1.0	50.1±0.3	53.0±0.6	55.0±0.1	50.0±0.4
MLPAE	62.9±0.3	75.0±0.6	50.3±0.3	52.9±0.1	56.1±0.3	49.9±0.2
SCAN	73.1±0.1	50.0±0.1	96.5±0.0	49.2±0.1	50.1±0.2	48.5±0.1
Radar	OOM	OOM	OOM	OOM	OOM	OOM
ANOMALOUS	OOM	OOM	OOM	OOM	OOM	OOM
GCNAE	63.1±0.6	**76.3**±0.5	50.2±0.1	53.2±0.2	55.7±0.2	50.2±0.4
DOMINANT	**87.0**±0.1	75.2±0.4	**99.1**±0.1	52.3±0.4	55.5±0.2	49.4±0.2
DONE	81.0±0.7	65.4±2.3	97.2±0.3	51.0±0.2	52.1±0.3	50.5±0.2
AdONE	79.8±0.6	60.4±2.2	92.4±3.1	51.4±0.1	52.5±0.3	50.4±0.2
AnomalyDAE	71.5±0.6	60.3±1.0	82.2±0.4	50.6±0.2	51.3±0.3	49.7±0.1
GAAN	46.8±0.2	46.5±1.1	50.5±0.1	50.0±0.5	49.7±0.3	50.0±0.1
CONAD	81.6±0.3	75.7±0.2	87.4±0.2	52.6±0.2	55.8±0.2	50.0±0.4
Ours (Euclidean)	71.8±3.9	58.7±11.6	76.6±2.8	63.9±7.0	58.1±6.1	60.7±7.5
Ours (Lorentz)	84.1±2.6	71.3±3.9	94.9±0.6	73.2±1.7	77.6±1.3	**73.5**±1.7
Ours (Poincaré)	82.3±0.4	72.7±4.1	93.7±1.0	**75.8**±0.9	**78.3**±0.9	72.5±0.4

Table 28: Mean and standard deviation of AP (%) scores on ogbn-arxiv dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
LOF	5.4±0.0	5.4±0.1	5.0±0.0	5.2±0.1	5.3±0.0	4.9±0.1
IF	11.4±0.0	21.9±0.4	5.0±0.0	5.3±0.1	5.6±0.0	5.0±0.0
MLPAE	11.1±1.3	16.6±0.1	5.0±0.1	5.3±0.0	5.7±0.1	5.0±0.0
SCAN	18.8±0.1	5.0±0.0	47.4±0.2	4.9±0.0	5.0±0.0	4.9±0.0
Radar	OOM	OOM	OOM	OOM	OOM	OOM
ANOMALOUS	OOM	OOM	OOM	OOM	OOM	OOM
GCNAE	12.2±0.4	**23.8**±0.4	5.0±0.0	5.3±0.0	5.6±0.0	5.0±0.0
DOMINANT	**65.0**±0.1	18.3±0.0	**98.4**±0.1	5.1±0.1	5.5±0.1	4.8±0.0
DONE	32.2±0.9	7.9±0.9	64.9±3.3	5.5±0.0	5.2±0.1	5.9±0.1
AdONE	21.3±0.5	6.5±0.4	38.8±3.0	5.3±0.1	5.3±0.0	5.5±0.1
AnomalyDAE	22.4±0.4	12.8±1.3	29.5±1.2	5.0±0.0	5.1±0.0	4.9±0.0
GAAN	4.7±0.0	4.6±0.1	5.0±0.0	5.0±0.1	4.9±0.1	5.0±0.0
CONAD	37.7±0.6	20.6±0.4	52.6±0.4	5.2±0.0	5.6±0.0	4.9±0.0
Ours (Euclidean)	12.2±1.8	12.0±6.3	10.0±0.9	11.3±4.2	8.1±3.2	7.3±1.8
Ours (Lorentz)	32.7±3.6	16.9±4.3	39.5±4.0	18.4±4.2	**36.3**±3.9	**12.9**±2.6
Ours (Poincaré)	29.1±0.9	21.6±7.3	44.8±3.2	**24.0**±2.2	35.3±3.3	11.4±0.6

C.6 Additional Results Comparing the Use of Message Passing and Contextual Loss

In Tables 29 and 30, we report AP results of our models with or without message passing and contextual loss in outlier detection on datasets Cora and Squirrel. In Tables 31-36, we report ROC-AUC and AP results of outlier detection in Chameleon, Actor, and Citeseer.

Table 29: Comparison of models with/without message passing and contextual loss on Cora dataset. Mean and standard deviation of AP (%) taken over 3 trials are reported.

	message	α	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
Ours (Euclidean)	No	0	19.8±2.4	13.0±0.6	23.6±6.2	27.6±3.4	16.0±2.0	42.2±2.9
		0.5	4.8±0.2	5.0±0.2	4.6±0.1	5.3±0.4	5.6±0.1	5.3±0.3
	Yes	0	11.1±0.2	5.7±0.6	15.4±0.5	18.2±2.9	5.2±0.1	33.4±1.9
		0.5	9.6±0.7	6.0±0.4	15.8±1.6	21.4±2.1	5.9±0.2	27.7±3.1
Ours (Lorentz)	No	0	10.5±1.9	10.0±0.7	12.9±0.4	14.3±2.0	9.4±0.4	16.9±1.3
		0.5	11.5±2.0	8.9±1.4	12.2±3.0	17.8±4.0	11.7±3.3	19.0±2.1
	Yes	0	13.2±0.4	5.3±0.1	25.2±3.4	27.2±1.2	5.8±0.2	52.1±2.6
		0.5	16.1±0.5	5.8±0.1	25.5±0.5	29.3±1.1	7.0±0.3	52.8±3.7
Ours (Poincaré)	No	0	19.9±2.5	**15.3**±0.8	26.7±1.0	34.2±0.9	**17.7**±2.6	**57.6**±1.7
		0.5	**20.5**±4.2	12.7±0.6	22.7±1.8	31.6±3.2	17.2±2.5	49.1±4.1
	Yes	0	14.9±2.2	5.1±0.2	**31.5**±2.9	28.8±4.3	5.9±0.3	55.4±2.6
		0.5	15.4±2.5	5.9±0.3	30.1±2.8	**34.6**±2.6	7.8±0.6	51.3±1.9

Table 30: Comparison of models with/without message passing and contextual loss on Squirrel dataset. Mean and standard deviation of AP (%) taken over 3 trials are reported.

	message	α	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
Ours (Euclidean)	No	0	18.1±9.0	5.4±0.0	5.9±0.2	17.8±17.5	15.5±14.3	5.3±0.1
		0.5	7.4±0.0	9.7±0.1	5.7±0.2	5.3±0.0	5.4±0.1	5.3±0.2
	Yes	0	4.9±0.4	5.2±0.1	4.2±1.0	6.8±0.2	5.5±0.2	7.5±1.5
		0.5	4.3±0.3	5.8±0.4	3.5±0.2	5.4±0.1	5.6±0.2	4.9±0.6
Ours (Lorentz)	No	0	**25.7**±5.2	22.6±2.5	**28.2**±2.4	35.7±2.0	34.7±5.6	35.2±5.4
		0.5	24.8±2.1	23.7±0.9	23.1±3.1	36.7±1.1	33.2±4.7	35.6±2.5
	Yes	0	12.2±0.4	13.3±1.3	11.0±0.9	23.5±2.4	17.9±2.8	36.7±2.4
		0.5	12.8±0.7	12.9±0.4	14.1±3.1	20.7±2.1	18.3±4.4	31.9±2.6
Ours (Poincaré)	No	0	24.9±2.2	**33.4**±3.9	18.6±1.5	49.0±3.2	34.7±1.6	62.7±1.3
		0.5	24.3±0.1	31.2±1.0	15.9±1.2	**50.6**±1.2	**35.0**±0.8	**65.9**±0.9
	Yes	0	4.9±0.3	5.7±0.1	4.3±0.1	8.3±0.4	5.4±0.2	11.2±0.3
		0.5	5.1±0.2	6.0±0.3	5.5±0.5	7.7±0.2	5.6±0.2	9.9±0.5

Table 31: Comparison of models with/without message passing and contextual loss on Chameleon dataset. Mean and standard deviation of ROC-AUC (%) taken over 3 trials are reported.

	message	α	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
Ours (Euclidean)	No	0	86.5±0.8	79.8±0.6	95.1±0.8	80.3±2.0	76.6±2.9	91.4±0.8
		0.5	53.7±0.4	64.1±2.2	43.4±1.3	50.5±1.3	52.1±2.8	50.2±2.2
	Yes	0	54.8±3.5	50.3±2.2	55.8±3.9	56.4±4.6	48.5±4.0	58.6±2.1
		0.5	53.5±0.4	53.2±1.5	49.4±2.7	55.1±3.3	51.1±0.7	57.2±1.1
Ours (Lorentz)	No	0	86.0±1.6	77.7±2.9	**96.7**±0.6	82.9±2.9	74.5±1.8	91.4±1.6
		0.5	82.5±3.7	74.0±2.4	91.2±3.3	83.4±2.9	75.8±1.6	88.5±3.6
	Yes	0	77.1±4.5	62.7±4.2	89.6±2.8	73.0±2.0	64.4±4.4	89.2±2.7
		0.5	77.4±0.3	70.1±2.6	85.6±2.7	71.3±2.6	63.8±1.9	82.5±1.1
Ours (Poincaré)	No	0	**88.9**±1.8	**84.2**±1.5	93.9±1.1	**87.4**±2.2	**83.0**±2.5	**92.3**±0.3
		0.5	79.1±1.4	73.4±5.5	83.7±2.0	82.0±0.4	77.0±2.1	81.6±2.6
	Yes	0	41.0±0.6	50.9±1.6	41.3±7.0	61.5±2.4	52.8±1.1	71.6±2.0
		0.5	57.0±8.9	53.7±4.6	41.7±6.1	59.6±0.5	52.1±0.8	69.5±1.7

Table 32: Comparison of models with/without message passing and contextual loss on Chameleon dataset. Mean and standard deviation of AP (%) taken over 3 trials are reported.

	message	α	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
Ours (Euclidean)	No	0	34.2±3.0	27.7±0.8	**44.9**±1.0	40.8±3.4	33.3±7.5	56.2±4.9
		0.5	6.0±0.2	8.8±0.8	4.8±0.2	5.4±0.2	5.6±0.3	5.4±0.3
	Yes	0	6.0±1.0	5.2±0.4	5.5±0.5	6.4±1.2	5.7±0.3	6.2±0.2
		0.5	5.6±0.4	5.5±0.3	4.8±0.2	6.1±0.9	5.4±0.2	6.3±0.5
Ours (Lorentz)	No	0	32.2±6.4	28.0±7.0	44.7±4.0	34.4±4.0	24.2±4.8	47.2±3.7
		0.5	32.4±8.7	24.0±2.1	37.9±4.0	36.2±2.8	28.1±7.2	45.6±3.1
	Yes	0	27.3±5.2	18.0±4.5	31.7±3.1	23.2±3.2	18.7±5.1	38.3±7.9
		0.5	22.7±0.7	20.1±3.9	36.3±8.1	21.8±2.7	17.2±1.3	36.2±3.8
Ours (Poincaré)	No	0	32.6±4.0	**32.5**±3.4	39.8±3.5	48.5±0.9	**35.9**±1.8	59.8±2.4
		0.5	**34.9**±3.7	29.1±3.5	38.8±1.8	**54.0**±4.0	32.2±3.7	**65.3**±3.9
	Yes	0	4.7±0.1	5.4±0.4	5.3±0.6	8.9±1.0	6.2±0.2	12.3±1.3
		0.5	7.5±1.9	6.1±0.8	5.5±1.1	7.9±0.2	5.7±0.5	12.4±0.2

Table 33: Comparison of models with/without message passing and contextual loss on Actor dataset. Mean and standard deviation of ROC-AUC (%) taken over 3 trials are reported.

	message	α	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
Ours (Euclidean)	No	0	55.2±1.6	**57.1**±1.0	60.1±0.6	51.9±1.0	52.7±0.9	52.2±0.7
		0.5	51.5±4.2	56.8±8.1	58.6±1.4	50.6±0.5	50.7±1.8	50.5±1.4
	Yes	0	44.6±2.9	48.7±0.7	37.4±1.3	50.1±1.3	50.4±2.0	51.2±1.8
		0.5	42.5±1.0	48.5±0.2	36.7±3.4	49.2±1.5	51.6±1.7	50.0±1.6
Ours (Lorentz)	No	0	**62.7**±3.9	47.9±3.5	**77.1**±0.8	53.9±0.2	50.7±0.8	**55.8**±2.3
		0.5	49.2±2.0	41.3±2.3	57.4±2.0	51.1±0.3	48.8±1.1	51.3±1.0
	Yes	0	45.5±0.6	51.6±0.4	40.6±1.4	51.2±1.0	51.0±1.3	52.7±2.4
		0.5	48.8±0.6	46.0±1.0	49.8±0.8	52.3±0.5	51.2±1.3	52.8±0.6
Ours (Poincaré)	No	0	60.1±2.6	56.2±0.6	66.6±1.5	**55.2**±2.3	**56.8**±1.0	55.6±1.8
		0.5	45.7±3.0	41.2±7.2	57.9±2.5	51.6±0.9	49.1±0.9	51.3±0.9
	Yes	0	42.9±2.1	50.4±0.9	36.5±1.7	53.1±2.3	48.9±1.2	52.5±1.4
		0.5	45.0±0.7	42.5±0.2	50.5±1.0	53.3±1.6	49.6±1.4	55.6±1.6

Table 34: Comparison of models with/without message passing and contextual loss on Actor dataset. Mean and standard deviation of AP (%) taken over 3 trials are reported.

	message	α	**cntxt.+strct.**	**cntxt.**	**strct.**	**"path"+DICE**	**"path"**	**DICE**
Ours (Euclidean)	No	0	5.6±0.2	6.5±0.3	6.0±0.0	6.7±0.8	6.6±0.2	7.0±0.2
		0.5	6.6±0.7	**9.0**±1.0	7.4±0.8	5.4±0.2	5.7±0.5	5.5±0.4
	Yes	0	4.4±0.2	5.0±0.3	3.8±0.1	5.5±0.3	5.3±0.4	5.6±0.5
		0.5	4.3±0.1	4.8±0.1	3.7±0.2	5.2±0.2	5.4±0.2	5.7±0.3
Ours (Lorentz)	No	0	**7.1**±1.0	5.0±0.6	**9.9**±0.3	6.1±0.3	5.5±0.3	6.9±1.1
		0.5	5.9±0.2	6.3±0.8	6.9±0.4	5.7±0.2	5.0±0.2	5.9±0.1
	Yes	0	4.8±0.2	5.5±0.0	4.2±0.2	6.4±0.2	5.7±0.2	7.2±0.8
		0.5	5.2±0.2	4.9±0.2	5.0±0.1	6.8±0.1	5.3±0.2	7.6±0.4
Ours (Poincaré)	No	0	6.2±0.5	6.3±0.3	6.9±0.2	**8.4**±0.9	**7.7**±0.4	**8.4**±0.7
		0.5	5.2±0.4	5.6±0.9	6.6±0.5	5.9±0.6	5.6±0.4	6.2±0.5
	Yes	0	4.4±0.2	5.4±0.4	3.8±0.0	6.7±0.6	5.0±0.2	7.5±0.9
		0.5	4.6±0.2	4.5±0.2	4.8±0.1	6.4±0.3	5.3±0.4	7.9±0.3

Table 35: Comparison of models with/without message passing and contextual loss on Citeseer dataset. Mean and standard deviation of ROC-AUC (%) taken over 3 trials are reported.

	message	α	**cntxt.+strct.**	**cntxt.**	**strct.**	**"path"+DICE**	**"path"**	**DICE**
Ours (Euclidean)	No	0	72.0±1.0	61.8±1.2	86.8±3.3	71.6±2.0	75.2±2.0	73.5±0.7
		0.5	51.8±2.6	54.1±1.0	52.4±1.2	47.6±0.8	50.8±1.0	49.6±2.2
	Yes	0	69.7±1.3	51.2±1.1	87.0±0.8	76.1±0.3	65.1±2.5	**82.5**±4.1
		0.5	67.3±2.0	52.5±1.2	82.5±4.1	72.0±0.6	61.9±1.1	78.6±1.4
Ours (Lorentz)	No	0	72.4±1.1	60.7±3.3	85.1±1.2	71.3±1.4	75.8±0.7	73.7±2.0
		0.5	73.8±2.0	62.6±4.8	86.0±0.1	74.8±2.0	73.0±0.3	75.2±0.9
	Yes	0	67.9±1.1	49.9±1.9	87.9±0.6	71.1±1.9	68.5±1.0	76.7±1.4
		0.5	74.4±0.4	57.2±2.1	88.5±0.8	72.5±0.0	71.4±1.8	76.0±1.7
Ours (Poincaré)	No	0	77.8±3.4	**69.9**±0.8	90.2±0.7	78.8±1.4	**82.4**±0.7	78.3±3.0
		0.5	**78.0**±1.0	63.8±2.9	**90.5**±2.1	**79.5**±0.9	79.1±1.7	79.7±2.2
	Yes	0	69.3±1.7	50.2±1.9	89.2±1.0	70.7±1.2	70.2±0.6	77.6±0.5
		0.5	72.1±0.7	54.8±0.6	88.7±1.0	76.6±0.8	73.3±0.5	80.5±1.8

Table 36: Comparison of models with/without message passing and contextual loss on Citeseer dataset. Mean and standard deviation of AP (%) taken over 3 trials are reported.

	message	α	**cntxt.+strct.**	**cntxt.**	**strct.**	**"path"+DICE**	**"path"**	**DICE**
Ours (Euclidean)	No	0	13.9±0.4	10.7±0.4	22.2±3.0	20.6±2.9	14.7±1.5	33.9±3.2
		0.5	5.9±0.5	6.0±0.2	6.2±0.4	5.2±0.0	5.7±0.3	5.5±0.4
	Yes	0	12.3±1.7	5.7±0.4	21.4±2.1	25.7±3.4	8.1±0.7	37.3±8.8
		0.5	9.9±0.8	5.9±0.2	16.5±3.0	18.1±2.3	7.0±0.1	30.3±1.5
Ours (Lorentz)	No	0	13.0±0.8	9.1±0.8	19.1±2.0	14.1±0.9	14.9±1.1	22.6±4.1
		0.5	15.0±1.7	11.8±2.8	18.1±0.9	20.7±1.2	13.2±0.7	24.9±2.9
	Yes	0	13.6±1.6	5.7±0.5	26.2±1.1	19.0±1.6	8.5±0.1	31.9±1.8
		0.5	18.8±0.9	6.6±0.5	29.2±3.9	19.8±0.6	10.3±1.2	32.2±1.6
Ours (Poincaré)	No	0	**21.1**±5.1	**14.7**±1.6	**29.9**±1.8	29.5±1.3	**20.1**±0.1	40.6±4.1
		0.5	20.0±2.5	12.2±1.3	28.7±3.2	**30.4**±2.2	18.0±1.7	**43.6**±3.8
	Yes	0	15.0±2.3	5.8±0.5	28.9±2.1	20.9±1.6	8.8±0.7	35.2±1.2
		0.5	16.8±2.3	6.1±0.2	26.5±1.2	27.0±2.4	10.9±1.3	33.4±3.8

C.7 Additional Visualization of the Distribution of Pairwise Distances

In Figure 3, we provide visualization of distribution of the pairwise distance of node embeddings of the dataset Squirrel.

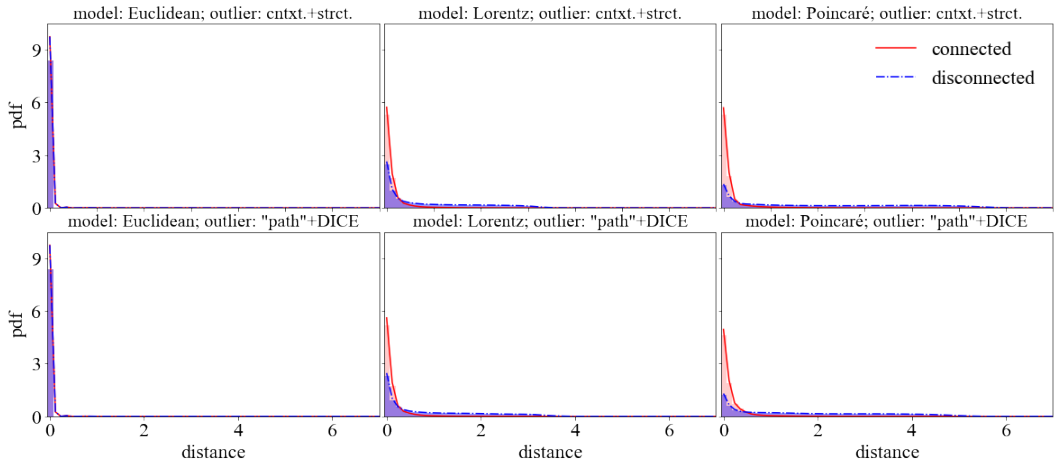

Figure 3: Probability density function (PDF) of the distribution of pairwise distances between embeddings of nodes that are originally connected ($\{d(\mathbf{h}_i, \mathbf{h}_j)\}_{(i,j)\in\mathcal{E}}$) and disconnected ($\{d(\mathbf{h}_i,\mathbf{h}_j)\}_{(i,j)\notin\mathcal{E}}$) of the Euclidean, Lorentz, and Poincaré models in Squirrel dataset injected with cntxt.+strct. and "path"+DICE outliers.

C.8 Results on Transformer Models

To further explore the influence of message passing in anomaly detection, we are also conducting experiments using transformer, which does not rely on message passing. Specifically, we replaced the last encoding layer of our models with a transformer layer [59]. For the hyperbolic models, we first mapped the embedding onto the tangent space at the origin via a logarithmic map, and applied the transformer layer on the tangent space, then mapped its output back to the hyperbolic space via exponential map.

Table 37: Mean and standard deviation of ROC-AUC (%) scores of Transformer model on Cora dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
Transformer (Euclidean)	75.4±0.4	65.9±2.0	86.0±0.5	70.6±3.6	66.1±5.6	79.1±2.2
Transformer (Lorentz)	65.2±1.9	56.0±2.8	79.8±2.8	56.2±2.0	59.5±1.5	54.5±1.2
Transformer (Poincaré)	77.7±1.9	70.8±3.7	86.4±1.7	77.0±1.6	70.7±2.5	77.0±5.4

Table 38: Mean and standard deviation of AP (%) scores of Transformer model on Cora dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
Transformer (Euclidean)	14.6±0.6	12.0±0.9	18.8±1.3	22.8±6.4	13.9±3.1	32.6±4.4
Transformer (Lorentz)	8.2±0.5	6.9±0.6	12.7±2.1	6.7±0.8	7.0±0.5	6.3±0.2
Transformer (Poincaré)	16.9±1.2	15.5±3.8	18.6±2.6	28.2±2.6	16.2±1.9	32.4±11.1

Table 39: Mean and standard deviation of AP (%) scores of Transformer model on Squirrel dataset.

	cont_struc	contextual	structural	path_dice	path	dice
Transformer (Euclidean)	18.5±2.6	27.0±0.9	20.1±1.9	36.4±0.8	33.0±1.5	40.0±0.1
Transformer (Lorentz)	6.5±0.2	6.1±0.5	9.3±0.8	6.5±0.9	6.4±0.5	6.0±0.6
Transformer (Poincaré)	23.5±3.1	32.5±2.1	20.7±2.8	39.2±2.4	28.0±1.7	55.5±5.5

Table 40: Mean and standard deviation of ROC-AUC (%) scores of Transformer model on Squirrel dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
Transformer (Euclidean)	71.0±7.9	69.8±1.8	88.8±1.6	79.5±1.8	74.3±1.7	88.9±1.5
Transformer (Lorentz)	60.4±0.4	53.8±1.3	73.8±3.0	54.8±3.5	53.5±0.6	53.5±2.9
Transformer (Poincaré)	81.9±2.1	80.2±1.7	85.8±3.0	81.7±0.4	73.3±2.1	89.3±1.0

Table 41: Mean and standard deviation of ROC-AUC (%) scores of Transformer model on Amazon dataset.

	cntxt.+strct.	cntxt.	strct.	"path"+DICE	"path"	DICE
Transformer (Euclidean)	85.4±0.6	78.2±0.5	96.1±0.2	84.1±0.7	78.6±2.5	91.7±0.8
Transformer (Lorentz)	71.4±4.1	61.3±9.2	83.9±3.0	64.4±1.4	60.7±2.1	68.0±0.2
Transformer (Poincaré)	78.0±16.6	69.6±15.0	97.9±1.0	88.2±3.1	82.3±3.7	85.2±7.0

Table 42: Mean and standard deviation of AP (%) scores of Transformer model on Amazon dataset.

	cont_struc	contextual	structural	path_dice	path	dice
Transformer (Euclidean)	38.2±2.9	30.8±4.8	41.8±1.9	43.3±2.2	42.5±3.0	38.8±2.7
Transformer (Lorentz)	12.3±3.9	11.2±6.6	14.5±2.1	9.4±1.0	7.7±0.6	9.2±0.3
Transformer (Poincaré)	34.8±20.5	21.8±11.8	66.3±11.4	52.6±10.1	44.1±9.3	30.5±17.2