Understanding and Tackling Over-Dilution in Graph Neural Networks

Junhyun Lee
Korea University
Seoul, South Korea
ljhyun33@korea.ac.kr

Veronika Thost
MIT-IBM Watson AI Lab
Massachusetts, United States
veronika.thost@ibm.com

Bumsoo Kim
Chung-Ang University
Seoul, South Korea
bumsoo@cau.ac.kr

Jaewoo Kang*
Korea University
Seoul, South Korea
kangj@korea.ac.kr

Tengfei Ma*
Stony Brook University
New York, United States
Tengfei.Ma@stonybrook.edu

Abstract

Message Passing Neural Networks (MPNNs) hold a key position in machine learning on graphs, but they struggle with unintended behaviors, such as over-smoothing and over-squashing, due to irregular data structures. The observation and formulation of these limitations have become foundational in constructing more informative graph representations. In this paper, we delve into the limitations of MPNNs, focusing on aspects that have previously been overlooked. Our observations reveal that even within a single layer, the information specific to an individual node can become significantly diluted. To delve into this phenomenon in depth, we present the concept of *Over-dilution* and formulate it with two dilution factors: *intra-node dilution* for attribute-level and *inter-node dilution* for node-level representations. We also introduce a transformer-based solution that alleviates over-dilution and complements existing node embedding methods like MPNNs. Our findings provide new insights and contribute to the development of informative representations.

CCS Concepts

• **Computing methodologies** → **Neural networks**; **Learning latent representations**; • **Theory of computation** → **Graph algorithms analysis**.

Keywords

Graph Neural Networks, Over-dilution, Node Attribute

ACM Reference Format:
Junhyun Lee, Veronika Thost, Bumsoo Kim, Jaewoo Kang, and Tengfei Ma. 2025. Understanding and Tackling Over-Dilution in Graph Neural Networks. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (ACM KDD 2025).* ACM, New York, NY, USA, 22 pages. https://doi.org/XXXXXXX.XXXXXXX

*Corresponding authors.

ACM KDD 2025, Toronto, ON, Canada

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Graph Neural Networks (GNNs) have emerged as powerful tools for learning on graph-structured data, largely owing to their ability to exploit structural information. In particular, Message Passing Neural Networks (MPNNs) have garnered significant attention due to their simple yet effective mechanism [9]. Numerous extensions have been proposed to boost their expressivity and address challenges such as over-smoothing [5, 23, 26, 27, 29, 36, 39, 41], over-squashing [1, 16, 30, 33], and over-correlation [17]. These efforts primarily focus on the propagation and aggregation of node-level representations.

Despite these advances, a critical aspect that has received comparatively less attention is the preservation of *attribute-level* information. In many applications—ranging from social networks to recommender systems—node attributes (e.g., demographic information, academic interests, or other contextual features) provide essential clues for prediction tasks [10, 12, 14, 21, 22]. Importantly, the relevance of each attribute is often dynamic and context-dependent; an attribute that is critical for one node may be less informative for another. For instance, while one sub-community of nodes in a social network might benefit from location attributes, another sub-community might derive greater value from academic interests instead of locations. Such local variability underscores the need for approaches that can adjust attribute weights dynamically rather than relying on a global, uniform treatment.

In this paper, we introduce the concept of *over-dilution* to describe the degradation of attribute-level details in the final node representations produced by MPNNs. As illustrated in Figure 1, we further decompose this phenomenon into two cascading subphenomena:

Intra-node dilution: This occurs when a node possesses a rich set of attributes, yet the naive aggregation process dilutes the contribution of individually important attributes. Crucially, it is not just the volume of attributes that matters, but also the fact that the importance of each attribute is inherently local and dynamic. Without adaptive weighting mechanisms, essential attribute information may be overwhelmed by less relevant details.

Inter-node dilution: This refers to the loss of a node's distinct attribute-level signals when it aggregates an overwhelming amount of information from its neighbors. The cumulative effect can further obscure the nuanced, context-specific attributes that are key to accurate representation.

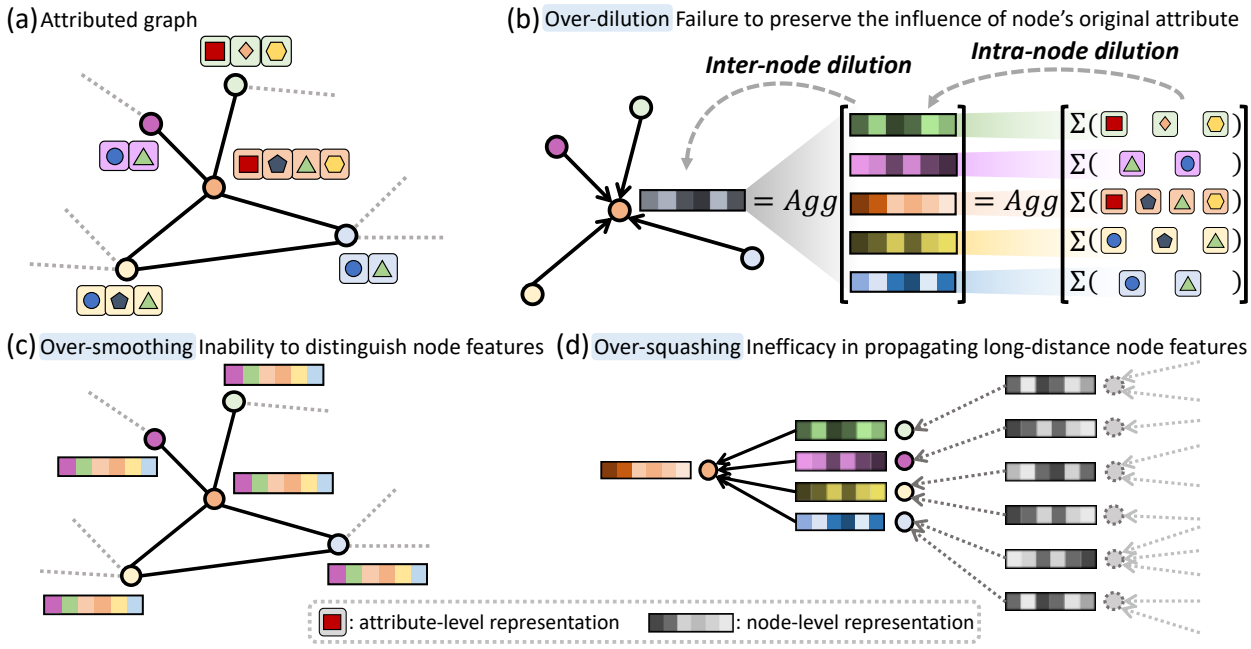

Figure 1: (a) illustrates an attributed graph, where each node is initially characterized by a set of attributes with corresponding embedding vectors. (b) demonstrates the two-step dilution process in the first layer of MPNNs. The first step, intra-node dilution, involves forming the node-level representation by summing its attribute representations, leading to reduced influence of each attribute, especially with an increasing number of attributes. The second step, inter-node dilution, occurs during message propagation, where a node's representation is integrated with those of its neighboring nodes. (c) and (d) illustrate the phenomena of over-smoothing and over-squashing, respectively.

To mitigate the over-dilution problem, we propose a transformer-based architecture [34] that treats attribute representations as tokens. The key advantage of our approach is its capacity to dynamically adjust the weights of attribute representations based on the local context of each node. Rather than imposing a uniform importance across all nodes, our method selectively emphasizes the attributes most relevant to a given node's characteristics and neighborhood. This flexible weighting mechanism ensures that critical attribute-level information is preserved, complementing existing node embedding methods rather than replacing them.

Our main contributions are summarized as follows:

- We introduce and formalize the *over-dilution* phenomenon, distinguishing it through two sub-phenomena: *intra-node dilution* (capturing the loss of dynamic, locally important attribute-level details) and *inter-node dilution* (capturing the dilution of node-level distinctiveness).
- We shift the focus from the traditional propagation of node-level representations to the preservation of attribute-level information, highlighting the necessity of adapting attribute weights to local contexts.
- We develop a transformer-based architecture that integrates seamlessly with any existing node embedding technique, and we validate its effectiveness both theoretically and empirically in mitigating the over-dilution issue.

2 Preliminaries

Attributed graphs are of the form $\mathcal{G} = (\mathcal{T}, \mathcal{V}, \mathcal{E})$ that consists of sets of attributes $\mathcal{T}$, nodes $\mathcal{V}$, and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Let $\mathcal{T}_v$ be a subset for attributes $t \in \mathcal{T}$ that node $v \in \mathcal{V}$ is associated with. $N_{\mathcal{V}} = |\mathcal{V}|$ and $N_{\mathcal{T}} = |\mathcal{T}|$ indicate the total numbers of nodes and attributes, respectively. The node feature matrix and the adjacency matrix are represented as $X \in \mathbb{R}^{N_{\mathcal{V}} \times N_{\mathcal{T}}}$ and $A \in \mathbb{R}^{N_{\mathcal{V}} \times N_{\mathcal{V}}}$, respectively. We assume that each node has a discrete binary vector, $X_v \in \mathbb{R}^{N_{\mathcal{T}}}$, indicating existence of attributes where $X_{v,t}$=1 if the node v has attribute t, otherwise $X_{v,t}$=0. The embedding of attribute t is represented as $z_t \in \mathbb{R}^d$ with dimension d, which is a randomly initialized representation.

2.1 Message Passing Neural Networks

In MPNNs, the representations of nodes are calculated through a series of layers, where each layer consists of two main operations: the *Update* function and the *Aggregate* function. The update function is used to transform the node representation and the aggregate function is used to combine information from neighboring nodes. This process is repeated for multiple layers, thereby refining the node representations and extracting higher-level features from the graph. We formulate MPNNs as:

$$\begin{aligned} h_v^{(l)} &= \sigma(\text{Aggregate}(\{\text{Update}(h_u^{(l-1)}) | u \in \tilde{\mathcal{N}}(v)\})) \\ &= \sigma\Big(\sum_{u \in \tilde{\mathcal{N}}(v)} \alpha_{vu} h_u^{(l-1)} W^{(l)}\Big) \end{aligned} \tag{1}$$

where $\tilde{\mathcal{N}}(v)$ is a set of neighbor nodes of v including itself, $W^{(l)} \in \mathbb{R}^{d\times d}$ is the learnable parameter at l-th layer, and Aggregate$(\cdot)$ denotes the aggregate function for neighbor nodes. $h^{(0)} = XW^{(0)} \in \mathbb{R}^{N_\mathcal{V}\times d}$ is the initial node feature matrix with learnable parameter $W^{(0)} \in \mathbb{R}^{N_\mathcal{T}\times d}$ and dimension d. In this context, the t-th row of $W^{(0)}$ is equivalent to z_t, the representation of the corresponding attribute. The parameter α_{vu} denotes the aggregation coefficient assigned to the edge connecting neighbor node u to the center node v in the aggregation function. This coefficient is calculated as $\frac{1}{\sqrt{\deg(v)\deg(u)}}$ in the case of GCN, or as an attention coefficient between nodes v and u in GAT. The receptive field of node v is defined as: $\mathcal{B}_l(v) := \{u \in \mathcal{V} \mid s_\mathcal{G}(v,u) \le l\}$, where $s_\mathcal{G}$ is the standard shortest-path distance on the graph $\mathcal{G}$ and $l \in \mathbb{N}$ is the radius.

2.2 Over-Smoothing and Over-Squashing

Over-smoothing refers to the phenomenon where the model excessively propagates information between nodes, leading to a loss of distinguishability of their representations [26, 27, 39]. In the process of exchanging information through message propagation, all nodes have similar representations and noise is conveyed alongside important information [5, 23].

Over-squashing is a problem that arises when exponentially increasing amounts of information are compressed into a fixed-size vector [1]. This leads to a bottleneck, particularly in the extended paths within a graph, which hinders GNNs from fitting long-range signals and causes them to fail to propagate messages originating from distant nodes. As a result, the performance is typically compromised, where the task necessitates long-range interaction [33].

As illustrated in Figure 1, over-dilution is distinguished from over-smoothing and over-squashing by its unique focus on the aspect of information retention within *individual* nodes. Whereas over-smoothing involves the excessive blending of features among *neighboring* nodes and over-squashing deals with the compression of information from *distant* nodes, over-dilution distinctively highlights the importance of preserving a node's attribute-level information. This distinction marks a significant shift in focus compared to the other phenomena. A detailed conceptual comparison of MPNN limitations, including over-correlation [17], is provided in Appendix A and illustrated in Figure 4.

3 Over-dilution

In this section, we introduce a new concept named over-dilution that refers to the diminishment of a node's information at the attribute level. To assess the severity of over-dilution, we define the dilution factor, as a metric that measures the retention of node's original attribute information in the updated representation. In the context of graphs, this factor can be decomposed into two cascaded components as described in Eq (2). We define the intra-node dilution factor δ_v^{intra} mainly for attribute representations and the inter-node dilution factor δ^{inter} for node representations. As depicted in Figure 1 (b), the attribute representation is diluted during the first step of message passing (i.e. *Update*) and then subsequently diluted in the second step (i.e. *Aggregate*) in form of the node-level representation. Therefore, the dilution factor of attribute t at node v can be defined as corresponding to two cascaded steps:

$$\delta_{v,t} = \delta_v^{\text{intra}}(t) * \delta^{\text{inter}}(v) \tag{2}$$

where $\delta_v^{\text{intra}}(t)$ represents the intra-node dilution factor of attribute t at the node v and $\delta^{\text{inter}}(v)$ represents the inter-node dilution factor of node v in the graph. We utilize the Jacobian matrix of node representations to quantify dilution factors, employing a method of assessing influence distribution akin to that used by Xu et al. and Topping et al..

Taking the Computers dataset as a primary example, consider a node with 204 attributes (the median value for the number of attributes) and 19 neighboring nodes (the median degree). In this scenario, each attribute of the node would be diluted to $1/204 * 1/20$, or roughly 0.025%, in a single layer when using either the *mean* or *sum* as the aggregation operator.

3.1 Intra-Node Dilution Factor: Measuring Attribute Influence within Each Node

The intra-node dilution factor is a metric that quantifies the degree to which an attribute is diluted at a specific node. We measure the influence of z_t on $h_v^{(0)}$ indicating how much the representation of attribute t affects the initial representation of node v.

Definition 3.1. (Intra-node dilution factor). *For a graph $\mathcal{G} = (\mathcal{T}, \mathcal{V}, \mathcal{E})$, let z_t be the representation of attribute $t \in \mathcal{T}$ and $h_v^{(0)}$ denote the initial feature representation of node $v \in \mathcal{V}$, which is calculated from the representations of attribute subset $\mathcal{T}_v$ that node v possesses. The influence score $I_v(t)$ attribute t on node v is the sum of the absolute values of the elements in the Jacobian matrix $\left[\frac{\partial h_v^{(0)}}{\partial z_t}\right]$. We define the intra-node dilution factor as the influence distribution by normalizing the influence scores: $\delta_v^{intra}(t) = I_v(t)/\Sigma_{s\in\mathcal{T}_v} I_v(s)$. In detail, with the all-ones vector e:*

$$\delta_v^{\text{intra}}(t) = e^T \left[\frac{\partial h_v^{(0)}}{\partial z_t}\right] e \Big/ \sum_{s\in\mathcal{T}_v} e^T \left[\frac{\partial h_v^{(0)}}{\partial z_s}\right] e \tag{3}$$

Case 1. (Occurrence of intra-node dilution) *Intra-node dilution occurs when a node-level representation is computed by equally weighting and fusing attribute-level representations, irrespective of their individual importance. The over-dilution effect at the intra-node level becomes more pronounced as the number of attributes increases.*

For example, given node v where the important attributes are sparse compared to the total number of attributes $|\mathcal{T}_v|$, the influence of the key attributes get proportionally limited to $1/|\mathcal{T}_v|$. In MPNNs, the representation of node v is calculated by summing or averaging the representations of attributes $t \in \mathcal{T}_v$ as $h_v^{(0)} = XW^{(0)} = \sum_{t\in\mathcal{T}_v} z_t$ or $h_v^{(0)} = \sum_{t\in\mathcal{T}_v} \frac{z_t}{|\mathcal{T}_v|}$. Therefore, the intra-node dilution factor $\delta_v^{\text{intra}}(t)$ takes the constant value $\frac{1}{|\mathcal{T}_v|}$ for all attributes at each node v. This implies that the influence of each attribute on the representation of a node is treated as equal and, as the number of attributes increases, the impact of important attributes on the representation of the node is diluted. Given that attributes possess different levels of practical importance, their influences may be diluted in cases where only a small subset of attributes is crucial for the node representation.

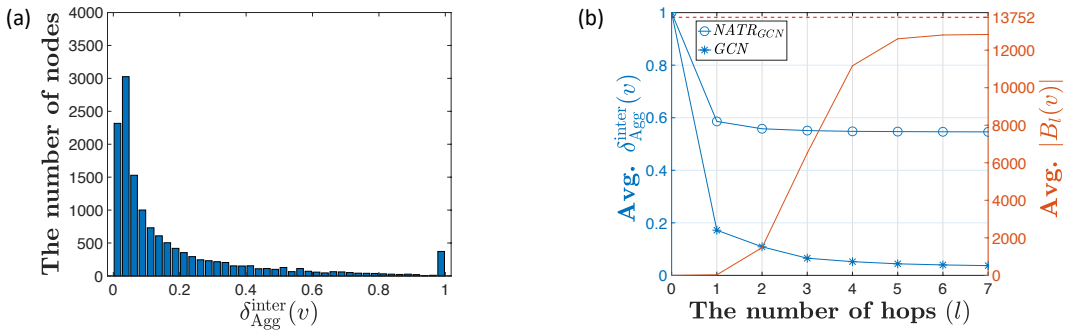

Figure 2: (a) The histogram of the inter-dilution factor (aggregation-only) values after single layer of *GCN* in Computers dataset. (b) The average of the inter-dilution factor (aggregation-only) with the left y-axis and the average size of the receptive field with the right y-axis in the Computers dataset. The analyses on each dataset and each hop count are detailed in Figures 7 and 8 in the Appendix.

Table 1: Summary of key limitations in GNNs, along with the formal expressions for the corresponding quantities of interest and their conceptual explanations. Here, MAD and $\mathcal{E}$ denote Mean Average Distance and Dirichlet energy, respectively. The parameters α and β represent the upper bounds of the derivatives of the update and message functions. Additionally, $\hat{A}$ denotes the normalized adjacency matrix, and $\rho(\cdot,\cdot)$ denotes the Pearson correlation coefficient.

Identified Issues	Formal Expression for the Quantity of Interest	Description
Over-smoothing	$MAD(h^{(l)}) = \frac{1}{v}\sum_{v\in\mathcal{V}}\sum_{u\in\mathcal{N}_v} 1 - \frac{h_v^{(l)\top} h_u^{(l)}}{\|h_v^{(l)}\|\|h_u^{(l)}\|}$ $\mathcal{E}(h^{(l)}) = \frac{1}{v}\sum_{v\in\mathcal{V}}\sum_{u\in\mathcal{N}_v} \|h_v^{(l)} - h_u^{(l)}\|_2^2$	The degree of homogeneity in the node-level feature space.
Over-squashing	$\left\|\frac{\partial h_v^{(l)}}{\partial h_u^{(0)}}\right\| \le (\alpha\beta)^l (\hat{A}^l)_{vu}$	The influence of node u's initial features on node v's final representation.
Over-correlation	$Corr(h^{(l)}) = \frac{1}{d(d-1)}\sum_{i\neq j} \lvert\rho(h_{:,i}^{(l)}, h_{:,j}^{(l)})\rvert$	The redundancy among the node-level feature dimensions.
Over-dilution	$\delta_{v,t} = \delta_v^{\text{intra}}(t) * \delta^{\text{inter}}(v)$	The compounded influence of attribute t on node v's final representation.
	$\delta_v^{\text{intra}}(t) = e^T \left[\frac{\partial h_v^{(0)}}{\partial z_t}\right] e \Big/ \sum_{s\in\mathcal{T}_v} e^T \left[\frac{\partial h_v^{(0)}}{\partial z_s}\right] e$	The influence of attribute t on node v's initial node-level representation.
	$\delta^{\text{inter}}(v) = e^T \left[\frac{\partial h_v^{(l)}}{\partial h_v^{(0)}}\right] e \Big/ \sum_{u\in\mathcal{V}} e^T \left[\frac{\partial h_v^{(l)}}{\partial h_u^{(0)}}\right] e$	The influence of node v's initial representation on its final representation.

3.2 Inter-node Dilution Factor: Measuring Node Influence on Final Representation

The inter-node dilution factor of each node is calculated by considering the influence of the initial node representation on the output representation in the last layer and the influences of all other nodes. We adapt the Jacobian matrix of node representation, as introduced by Xu et al. for quantifying the influence of one node on another, to measure the influence of each node on itself.

Definition 3.2. (Inter-node dilution factor). *Let $h_v^{(0)}$ be the initial feature and $h_v^{(l)}$ be the learned representation of node $v \in \mathcal{V}$ at the l-th layer. We define the inter-node dilution factor as the normalized influence distribution of node-level representations: $\delta^{\text{inter}}(v) = I_v(v)/\Sigma_{u\in\mathcal{V}} I_v(u)$, or*

$$\delta^{\text{inter}}(v) = e^T \left[\frac{\partial h_v^{(l)}}{\partial h_v^{(0)}}\right] e \Big/ \sum_{u\in\mathcal{V}} e^T \left[\frac{\partial h_v^{(l)}}{\partial h_u^{(0)}}\right] e \tag{4}$$

In MPNNs, the representation $h_v^{(l)}$ is calculated from the non-linear transformation (i.e. Update$(\cdot)$) and the aggregation of the representations $h_u^{(l-1)}$ for $u \in \tilde{N}(v)$. To observe the effect of the aggregation exclusively, we eliminate the effect of the non-linear transformation by setting all weight and initial node feature matrices to be the identity matrix. We define $\delta^{\text{inter}}_{\text{Agg}}(v)$, which is the exclusive version of the inter-node dilution factor, with $W^{(l)} = W^{(l-1)} = \ldots = W^{(1)} = h^{(0)} = I_{N_\mathcal{V}} \in \mathbb{R}^{N_\mathcal{V}\times N_\mathcal{V}}$. The output representation of the aggregation-only version of *GCN* is calculated as $h'^{(l)} = (\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}})^l I_{N_\mathcal{V}}$, where $\tilde{A}$ indicates the adjacency matrix with self-loop and $\tilde{D}$ is the corresponding degree

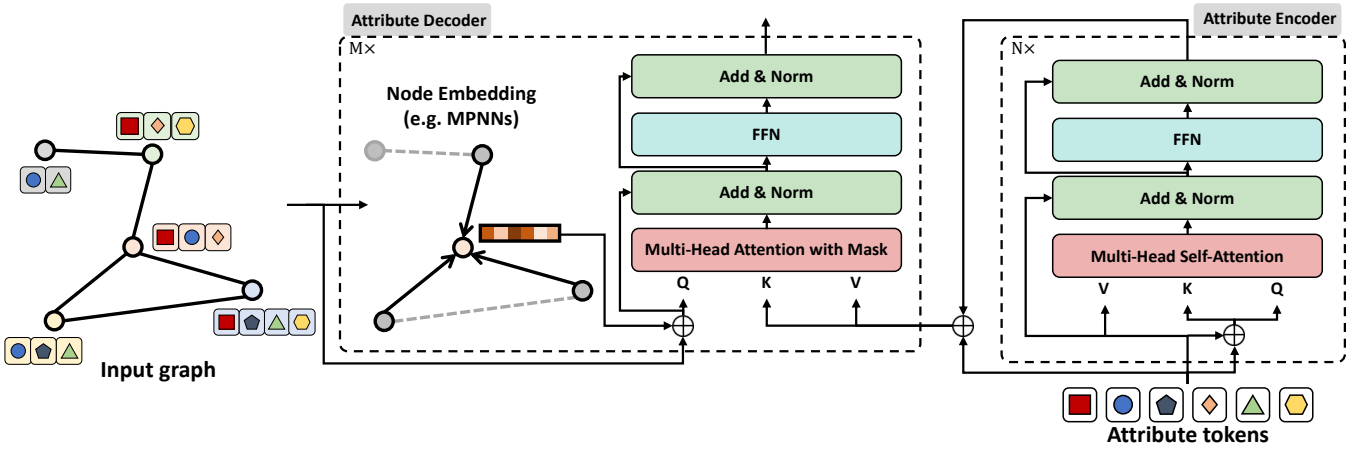

Figure 3: The overall architecture of the Node Attribute Transformer (*NATR*) comprises of two main components: the attribute encoder for attribute-level representations and the attribute decoder for node-level representations. It can be combined with any node embedding modules such as MPNNs and graph transformers.

matrix. The numerator of $\delta_{\text{Agg}}^{\text{inter}}(v)$ is calculated from:

$$\frac{\partial h_v'^{(l)}}{\partial h_v^{(0)}} = \underbrace{\prod_{i=1}^{l} \alpha_{vv}^{(i)} \cdot \frac{\partial h_v^{(0)}}{\partial h_v^{(0)}}}_{\text{for } l \geq 1} + \underbrace{\sum_{u \in \tilde{N}(v) \setminus \{v\}} \sum_{k=1}^{l-1} \left(\prod_{\substack{j=k+2 \\ k \leq l-2}}^{l} \alpha_{vv}^{(j)} \right) \alpha_{vu}^{(k+1)} \frac{\partial h_u'^{(k)}}{\partial h_v^{(0)}}}_{\text{for } l \geq 2} \tag{5}$$

where $\alpha_{vu}^{(i)}$ indicates the aggregation coefficient from node u to the node v at i-th layer. The former term, which is defined for $l \geq 1$, indicates the preserved amount of the representation of node v and the latter term, which is defined for $l \geq 2$, indicates the returned amount of representation of node v from neighbors after more than two hops aggregation. The denominator of $\delta_{\text{Agg}}^{\text{inter}}(v)$ is calculated from:

$$\sum_{u \in \mathcal{V}} \frac{\partial h_v'^{(l)}}{\partial h_u^{(0)}} = \sum_{x \in \tilde{N}(v)} \sum_{u \in \mathcal{V}} \sum_{k=0}^{l-1} \left(\prod_{\substack{j=k+2 \\ k \leq l-2}}^{l} \alpha_{vv}^{(j)} \right) \alpha_{vx}^{(k+1)} \frac{\partial h_x'^{(k)}}{\partial h_u^{(0)}} \tag{6}$$

Case 2. (Occurrence of inter-node dilution 1) *For a node v and its adjacent nodes, which are denoted as $\tilde{N}(v)$, inter-node dilution occurs when the aggregation coefficient of the self-loop, α_{vv}, is significantly smaller than the sum of the coefficients of the other edges connecting node v and its adjacent nodes: $\alpha_{vv} \ll \sum_{u \in \tilde{N}(v) \setminus \{v\}} \alpha_{vu}$.*

The inter-node dilution factor for the aggregation-only at a single layer is calculated as:

$$\delta_{\text{Agg}}^{\text{inter}}(v) = e^T \left[\alpha_{vv} \frac{\partial h_v^{(0)}}{\partial h_v^{(0)}} \right] e \Big/ e^T \left[\sum_{u \in \tilde{N}(v)} \alpha_{vu} \frac{\partial h_u^{(0)}}{\partial h_u^{(0)}} \right] e = \frac{\alpha_{vv}}{\sum_{u \in \tilde{N}(v)} \alpha_{vu}} \tag{7}$$

In most MPNNs, the inter-node dilution occurs when the degree (i.e. $|\mathcal{N}(v)|$) is high. For *GCN*, it can even occur with the low degree if the neighbor nodes have smaller degrees compared to node v, because the aggregation coefficient for self-loop is defined as $\alpha_{vv} = \frac{1}{\deg(v)}$ while the coefficients for edges with neighbor nodes are defined as $\alpha_{vu} = \frac{1}{\sqrt{\deg(v)\deg(u)}}$. As shown in the Figure 2(a), a significant number of nodes exhibits low $\delta_{\text{Agg}}^{\text{inter}}(v)$ values even after one-hop aggregation.

Case 3. (Occurrence of inter-node dilution 2) *Inter-node dilution occurs at node v as the size of its receptive field $|\mathcal{B}_l(v)|$ increases.*

As explained in Xu et al. and Topping et al., the size of the receptive field grows exponentially as the number of layers increases. Consequently, the information from a larger number of nodes is integrated, resulting in a dilution of the information specific to each individual node. Figure 2(b) illustrates the average of this relationship between the inter-node dilution factor (aggregation-only) and the average size of the receptive field in the Computers dataset, as the number of hops increases.

4 Node Attribute Transformer

In this section, we describe the architecture of Node Attribute Transformer (*NATR*) in details. As illustrated in Figure 3, *NATR* consists of the attribute encoder and the attribute decoder. While the encoder is designed to consider the correlation between attributes, the decoder plays a crucial role in mitigating over-dilution. It integrates

attribute representations across all layers, addressing inter-node dilution, and assigns greater weight to important attributes, tackling intra-node dilution, as discussed in Section 6.1.

4.1 Attribute Encoder

Given a set of randomly initialized representations of attribute tokens $z_t^{(0)} \in \mathbb{R}^{d_\mathcal{T}}$ and its matrix form $Z^{(0)} \in \mathbb{R}^{N_\mathcal{T} \times d_\mathcal{T}}$ with $d_\mathcal{T}$ dimension, the attribute representation $Z^{(n)}$, which is the output of n-layer of the attribute encoder, is obtained as: $Z^{(n)} = \text{SelfAttn}^{(n)}(Z^{(n-1)})$, where $\text{SelfAttn}^{(n)}$ is the n-th layer of the attribute encoder containing Multi-Head Self-Attention (MHSA), Add&Norm [2], and Feed-Forward Network (FFN) layers as illustrated in the Figure 3. We add $z_t^{(0)}$ to $z^{(n)}$ for the key and the query at all encoder layers like the positional encoding and it is omitted in the formulation for simplicity. After N layers of attribute encoder in total, the attribute representation $z_t = z_t^{(N)} + z_t^{(0)}$, which is $Z \in \mathbb{R}^{N_\mathcal{T} \times d_\mathcal{T}}$ in a matrix form, is fed to the attribute decoder. For simplicity, we use the same dimension ($d_\mathcal{T} = d$) for attribute-level and node-level representations.

4.2 Attribute Decoder

The attribute decoder is comprised of the node embedding module, Multi-Head Attention (MHA), Add&Norm, and FFN. The output of the attribute encoder, Z is used to calculate the key $K^{(m)} = ZW_{DEC,K}^{(m)}$ and the value $V^{(m)} = ZW_{DEC,V}^{(m)}$ in the MHA of m-th decoder layer. The query $Q^{(m)} = H^{(m)}W_{DEC,Q}^{(m)}$ is calculated from the output of the node embedding module $H^{(m)} \in \mathbb{R}^{N_\mathcal{V} \times d}$, such as MPNNs, at the m-th decoder layer:

$$H^{(m)} = \mathit{NodeModule}(\tilde{H}^{(m-1)}, A) \tag{8}$$

where $\tilde{H}^{(m-1)}$ is the output of the previous decoder layer ($\tilde{H}^{(0)} = H^{(0)} = XW^{(0)}$) and A represents the adjacency matrix. We add $H^{(0)}$ before calculating the query at all decoder layers and it is also omitted in the formulation for simplicity. We denote the node embedding module in the subscript as $\mathit{NATR}_{\mathit{NodeModule}}$. If $H^{(m)}$ is updated by *GCN* layer with the formulation $H^{(m)} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\tilde{H}^{(m-1)}W^{(m)}$, the model is denoted as $\mathit{NATR}_{\mathit{GCN}}$. Then, the attention coefficient for each attribute at MHA is calculated according to the node-level representation $Q^{(m)}$.

$$\begin{aligned} O^{(m)} &= \text{MHA}(Q^{(m)}, K^{(m)}, V^{(m)}) \\ &= \text{Concat}(\text{head}_1, ..., \text{head}_h)W_{DEC,O}^{(m)} \end{aligned} \tag{9}$$

where $\text{head}_i = \text{softmax}(\frac{Q_i K_i^\top}{\sqrt{d}})V_i$. We use masks in the MHA to merge the representations of the attributes possessed by each node exclusively. The aggregated representation of neighbor nodes $H^{(m)}$ is added to the output $O^{(m)} \in \mathbb{R}^{N_\mathcal{V} \times d}$ and then fed to Normalization layer followed by FFN layer. After additional normalization layer with skip connection, the final representation at m-th layer of attribute decoder, $\tilde{H}^{(m)} \in \mathbb{R}^{N_\mathcal{V} \times d}$ is used as the input feature of the node embedding module at the next decoder layer.

$$\begin{aligned} G^{(m)} &= \text{Norm}(H^{(m)} + O^{(m)}) \\ \tilde{H}^{(m)} &= \text{Norm}(\text{FFN}(G^{(m)}) + G^{(m)}) \end{aligned} \tag{10}$$

Note that $H^{(m)}$ is the representation which is aggregated from neighbors and $O^{(m)}$ is the representation of each node. Therefore, we can control the inter-node dilution factor by changing as $G^{(m)} = \text{Norm}((1-\lambda)H^{(m)} + \lambda O^{(m)})$, where $0 \leq \lambda \leq 1$ can be a hyperparameter, a learnable parameter, or an attention coefficient. In this work, we use the original formulation in Eq. (10).

Plug-in Version of NATR. We also provide a variant of *NATR* which is a plug-in version. In the scenario where an established node embedding model (e.g. MPNNs) is already trained and running in the industry, *NATR* can be easily incorporated into the existing model as a form of the separate architecture which is illustrated in Appendix I. The plug-in version is also available for single layer models such as SGC [38]. The difference from standard *NATR* is that the node embedding module is not nested inside the decoder but operated separately.

5 Experiments

We evaluate *NATR* on four benchmark datasets using the OGB pipeline [13] [1]. To validate the informativeness of the node representations, we perform both link prediction and node classification tasks. All experiments are repeated 20 times, and the average performance is reported. As baselines, we select *GCN* [19], *SGC* [38], *GAT* [35], *GCNII* [6], and *GRAPHORMER* [40]. For $\mathit{NATR}_{\mathit{SGC}}$, we adapt the plug-in version of the architecture.

In *GCN* and *SGC*, aggregation coefficients are computed based on node degree, so the inter-node dilution factor is influenced by the topology. In contrast, *GAT* computes its aggregation coefficients using attention mechanisms, which means its inter-node dilution factor is determined by the node representations. Meanwhile, *GCNII* updates node representations while preserving the initial node-level feature $h^{(0)}$. Although *GRAPHORMER* learns aggregation coefficients for node-level representations, both it and *GCNII* still suffer from intra-node dilution when constructing $h^{(0)}$. Detailed experimental settings, extended results for various node embedding modules (e.g., SAGE, GATV2, ARMA, and PNA [3, 7, 11]), analyses of transformer-based architectures' limitations (such as complexity and model size), and ablation studies are provided in the Appendix.

5.1 Datasets

The Computers and Photo datasets are segments of the Amazon co-purchase graph [24, 32]. In these datasets, nodes represent products, and edges indicate that two products are frequently purchased together. A bag-of-words representation derived from product reviews is used as a set of attributes. The Cora ML dataset also employs a bag-of-words as attributes; however, in this case, nodes represent documents, and edges indicate citation links between them [4, 25]. In the OGB-DDI dataset, provided by Wishart et al. and Hu et al., each node represents a drug, and edges represent interactions between drugs. We extract node attributes from the molecular structures in DrugBank DB [37] by generating Morgan Fingerprints (radius 3, 1024 bits) using RDKit. Nodes that are not supported by RDKit or DrugBank are removed, and the corresponding graph is subsequently reconstructed as $\text{OGB-DDI}_{\text{SUBSET}}$. The $\text{OGB-DDI}_{\text{FULL}}$ dataset includes all nodes and edges; unsupported nodes are assigned a dummy attribute.

[1] The code is available at: https://anonymous.4open.science/r/Over-dilution-F164/

Table 2: Dataset statistics. $|\mathcal{T}_v|$ and degree are related to intra- and inter-node dilutions, respectively.

	$\lvert\mathcal{V}\rvert$	Avg. Degree	Median Degree	$\lvert\mathcal{T}\rvert$	Avg. $\lvert\mathcal{T}_v\rvert$	Median $\lvert\mathcal{T}_v\rvert$	Max. $\lvert\mathcal{T}_v\rvert$
Computers	13752	30.393	19	767	267.2	204	767
Photo	7650	26.462	18	745	258.8	193	745
Cora ML	2995	4.632	3	2879	50.5	49	176
OGB-DDI$_{\text{SUBSET}}$	3531	499.582	500	1024	58.2	56	270
OGB-DDI$_{\text{FULL}}$	4267	500.544	446	1024+1	49.1	51	271

Table 3: Experimental results of the link prediction with Hits@20 performance (top) and the node classification with MAD score (bottom).

	Computers	Photo	Cora ML	OGB-DDI$_{\text{SUBSET}}$	OGB-DDI$_{\text{FULL}}$
GCN	31.01 ±3.37	51.05 ±5.45	75.93 ±4.36	76.11 ±5.92	68.18 ±9.24
NATR$_{GCN}$	**42.38** ±3.21	**58.12** ±4.18	**77.04** ±2.61	**78.51** ±4.03	**73.07** ±8.16
GAT	24.73 ±4.96	48.23 ±7.43	72.42 ±3.45	61.46 ±11.51	29.02 ±12.52
NATR$_{GAT}$	**40.63** ±3.97	**56.06** ±3.54	**74.10** ±3.22	**80.68** ±2.32	**77.80** ±6.79
SGC	30.37 ±2.73	51.31 ±4.80	74.49 ±3.03	41.04 ±7.12	39.19 ±7.87
NATR$_{SGC}$	**36.99** ±3.34	**57.42** ±4.38	**77.20** ±2.85	**86.79** ±3.66	**76.99** ±10.91
GCNII	30.00 ±3.69	52.05 ±3.23	74.66 ±3.12	67.72 ±6.29	66.52 ±9.11
NATR$_{GCNII}$	**37.47** ±4.56	**56.98** ±3.45	**74.98** ±2.57	**81.45** ±3.55	**78.00** ±8.71
Graphormer	25.94 ±4.32	48.58 ±5.76	66.36 ±2.82	74.48 ±4.61	74.22 ±8.36
NATR$_{Graphormer}$	**30.71** ±3.68	**51.56** ±4.21	**67.81** ±2.42	**87.52** ±3.91	**80.82** ±7.44

	Computers		Photo		Cora ML	
	Accuracy	MAD	Accuracy	MAD	Accuracy	MAD
GCN	80.12 ±1.71	0.46 ±0.03	88.50 ±2.11	0.83 ±0.06	78.71 ±2.00	0.55 ±0.03
NATR$_{GCN}$	**81.70** ±2.75	**0.82** ±0.04	**90.84** ±1.26	**0.91** ±0.02	**80.39** ±2.28	**0.68** ±0.03
GAT	80.86 ±1.95	0.63 ±0.05	88.87 ±2.04	0.57 ±0.04	77.35 ±2.02	**0.84** ±0.04
NATR$_{GAT}$	**81.39** ±2.12	**0.67** ±0.03	**89.23** ±1.93	**0.89** ±0.02	**79.36** ±1.66	0.74 ±0.02
SGC	80.31 ±1.53	0.26 ±0.03	89.18 ±1.67	0.45 ±0.07	79.30 ±1.89	0.34 ±0.02
NATR$_{SGC}$	**80.63** ±2.30	**0.68** ±0.03	**89.60** ±1.74	**0.78** ±0.05	**80.22** ±1.03	**0.92** ±0.02
GCNII	81.26 ±0.93	0.65 ±0.03	90.41 ±1.88	0.86 ±0.04	78.65 ±1.83	0.72 ±0.03
NATR$_{GCNII}$	**83.44** ±1.18	**0.81** ±0.01	**91.02** ±0.98	**0.90** ±0.03	**83.02** ±1.44	**0.77** ±0.02
Graphormer	82.93 ±1.59	0.55 ±0.04	89.74 ±2.23	0.66 ±0.05	79.20 ±2.40	0.81 ±0.02
NATR$_{Graphormer}$	**86.92** ±2.05	**0.70** ±0.03	**92.39** ±1.61	**0.87** ±0.04	**83.89** ±1.99	**0.91** ±0.03

5.2 Tasks

Link Prediction. We evaluate link prediction performance, a task for which attribute-level representation is particularly important [12, 22]. For the OGB-DDI$_{\text{FULL}}$ dataset, the attributes indicate substructures of chemical compounds, thereby providing valuable information about potential interactions between drugs in a biological system. Overall performance is reported in Table 3 (top), and performance as a function of the number of layers is presented in Table 4.

Node Classification. Although smoothing node representations to be more similar to their neighboring nodes can benefit node classification tasks on homogeneous graphs, such smoothing may come at the expense of preserving individual node features. Our experimental results, however, demonstrate that *NATR* maintains individual node representations without impeding performance. We also assess the smoothness of node representations using the Mean Average Distance (MAD) metric [5]. As shown in Table 3 (bottom), the results indicate that the *NATR* architecture effectively addresses

Table 4: Hits@20 performance on Computers dataset by the number of layers.

	2 Layers	3 Layers	4 Layers	5 Layers
GCN	**31.01**	30.84	28.97	26.99
$GCN_{\mathcal{JK}}$	**29.47**	27.85	28.00	27.49
$NATR_{GCN}$	39.81	41.54	40.96	**42.38**
GAT	**24.73**	21.07	11.52	4.15
$GAT_{\mathcal{JK}}$	**27.22**	24.54	23.90	23.98
$NATR_{GAT}$	39.51	39.58	**40.63**	40.21
SGC	**30.37**	25.78	24.30	23.87
$NATR_{SGC}$	**36.99**	36.47	35.31	34.01

the over-smoothing issue by preserving the unique representation of each node.

6 Analysis

6.1 Improvements in the Dilution Factors of NATR

The intra-node dilution factor. Unlike the $1/|\mathcal{T}_v|$ approach used in MPNNs, *NATR* can enhance the representation of important attributes while suppressing others. The intra-node dilution factor for attribute t is calculated as $\exp(Q_v K_t^\top)/\sum_{s\in\mathcal{T}_v}\exp(Q_v K_s^\top)$, which is the attention coefficient at node v. In comparison to *GCN*, $NATR_{GCN}$ increases $\delta_v^{\text{intra}}(t)$ in 38.07% of all cases with a median increase of +30.31% and a maximum increase of +4005.60% on the Computers dataset. The detailed statistics are reported in the Appendix.

The inter-node dilution factor. The final representation of node v at the last layer $\tilde{H}_v^{(M)}$ is calculated based on two node-level representations: $H_v^{(M)}$ and $O_v^{(M)}$. The term $H_v^{(M)}$ contains information from its neighboring nodes, while $O_v^{(M)}$ pertains exclusively to node v. The inter-node dilution factor of *NATR* is defined as:

$$\delta^{\text{inter}}(v) = \frac{e^T\left[\frac{\partial\tilde{H}_v^{(M)}}{\partial H_v^{(0)}} + \sum_{m=1}^{M}\frac{\partial\tilde{H}_v^{(M)}}{\partial O_v^{(m)}}\right]e}{e^T\left[\sum_{u\in\mathcal{V}}\frac{\partial\tilde{H}_v^{(M)}}{\partial H_u^{(0)}} + \sum_{m=1}^{M}\frac{\partial\tilde{H}_v^{(M)}}{\partial O_v^{(m)}}\right]e} \tag{11}$$

Even when $\frac{\partial\tilde{H}_v^{(M)}}{\partial H_v^{(0)}}$ value of the numerator is smaller than $\sum_{u\in\mathcal{V}}\frac{\partial\tilde{H}_v^{(M)}}{\partial H_u^{(0)}}$ value of the denominator as in MPNNs, the factor value can still be high in *NATR*. This is primarily due to the contribution from $\sum_{m=1}^{M}\frac{\partial\tilde{H}_v^{(M)}}{\partial O_v^{(m)}}$ helping each node preserve its own feature as shown in Figure 2(b). As demonstrated in Table 4, the performance of MPNNs deteriorates as the depth of the layers increases, whereas *NATR* models exhibit performance gains. In the case of $NATR_{SGC}$, because we adapt the plug-in version that uses over-diluted representations as queries, the performance is slightly decreased. MPNNs with jumping knowledge (JK) [39], which concatenate the outputs of all layers, alleviate the performance drop compared to original models. However, JK models fail to improve performance implying that they are inadequate in utilizing the information as the number of layers increases.

Table 5: Hits@5 performance on subsets of Computers dataset.

	$\mathcal{E}_{Q1}$	$\mathcal{E}_{Q4}$
GCN	19.96	42.69
$NATR_{GCN}$	23.96 (+20.04%)	45.18 (+5.84%)
GAT	13.57	34.86
$NATR_{GAT}$	24.46 (+80.38%)	43.49 (+24.74%)
SGC	19.39	38.61
$NATR_{SGC}$	24.10 (+24.29%)	39.72 (+2.89%)

Table 6: Comparison with various models: (A)-*MLP*, (B)-*GCN*, (C)-GCN_{JK}, (D)-$NATR_{MLP}$ with a *MLP* encoder, (E)-$NATR_{MLP}$ with a SelfAttn encoder, (F)-$NATR_{GCN}$. The model utilizing the correlation between attributes is indicated by CORR, MP denotes the use of message passing, and Attn indicates that the value is determined by the attention mechanism.

	$\delta_v^{intra}(t)$	$\delta_{\text{Agg}}^{inter}(v)$	Corr	MP	Hits@20
(A)	$1/\lvert\mathcal{T}_v\rvert$	high	✗	✗	20.37
(B)	$1/\lvert\mathcal{T}_v\rvert$	low	✗	✓	31.01
(C)	$1/\lvert\mathcal{T}_v\rvert$	high	✗	✓	29.47
(D)	Attn	high	✗	✗	33.26
(E)	Attn	high	✓	✗	34.89
(F)	Attn	high	✓	✓	42.38

6.2 Effectiveness of NATR

To explore the effectiveness of *NATR*, we measure performance on subsets standing for over-diluted nodes $\mathcal{V}_{Q1}$ and less-diluted nodes $\mathcal{V}_{Q4}$ after two hops aggregation, which are defined as:

$$\mathcal{V}_{Q1} = \{v\in\mathcal{V} \mid \delta_{\text{Agg}}^{\text{inter}}(v) \le Q1\}$$
$$\mathcal{V}_{Q4} = \{v\in\mathcal{V} \mid \delta_{\text{Agg}}^{\text{inter}}(v) \ge Q3 \ \wedge\ \delta_{\text{Agg}}^{\text{inter}}(v) \ne 1\} \tag{12}$$

where $Q1$ and Q_3 represent the first and the third quartiles, which divide the set into the bottom 25% and the top 25% of $\delta_{\text{Agg}}^{\text{inter}}(v)$ values, respectively. We define two subsets of corresponding edges as $\mathcal{E}_{Q1} = \{(i,j)\in\mathcal{E} \mid i\in\mathcal{V}_{Q1} \vee j\in\mathcal{V}_{Q1}\}$ and $\mathcal{E}_{Q4} = \{(i,j)\in\mathcal{E} \mid i\in\mathcal{V}_{Q4} \vee j\in\mathcal{V}_{Q4}\}$. The isolated nodes, defined as those with $\delta_{\text{Agg}}^{\text{inter}}(v) = 1$, are excluded. As shown in Table 5, *NATR* models demonstrate improved performance on both subsets, with particularly noteworthy improvement on over-diluted nodes compared to MPNNs.

Furthermore, we compare models with various conditions as described in Table 6. The distinction between (A) and (D) lies in the weight assigned to attribute-level representations. Both are *MLP*-based models but (D) alleviates the intra-node dilution by mixing attributes according to the attention coefficients. The comparison with (D) and (E) shows the effectiveness of the correlation between attributes through SelfAttn in the attribute encoder. The *GCN* models, (B) and (C), improve performance compared to (A) as a result of incorporating contextual information from neighboring nodes. The complete model (F) exploits *GCN* as a node embedding

module, allowing attribute representation to be fused while taking into account the context of the graph.

7 Conclusions

In this work, we introduced the notion of *over-dilution* to elucidate the limitation inherent to MPNNs. To assess the over-dilution effect, we proposed a formal framework to analyze this phenomenon, delineating it into two aspects: *intra-node dilution* and *inter-node dilution*, which deal with the preservation of information at attribute and node levels, respectively. Our findings demonstrate the efficacy of employing a transformer-based architecture to mitigate the effects of over-dilution. This work not only sheds light on the limitations of MPNNs but also lays the groundwork for advancing data mining on graphs.

Acknowledgments

Jaewoo Kang and Junhyun Lee were supported by the Ministry of Science and ICT (NRF-2023R1A2C3004176), the Institute of Information & Communications Technology Planning & Evaluation (IITP-2025-RS-2020-II201819), and the Ministry of Health and Welfare of South Korea (HR20C002103).

References

[1] Uri Alon and Eran Yahav. 2021. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *Proceedings of International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=i80OPhOCVH2

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[3] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. 2021. Graph neural networks with convolutional arma filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[4] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *Proceedings of International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=r1ZdKJ-0W

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

[6] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and Deep Graph Convolutional Networks. In *Proceedings of International conference on machine learning (ICML)*. 1725–1735.

[7] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. 2020. Principal Neighbourhood Aggregation for Graph Nets. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS) 33*. Curran Associates, Inc., 13260–13271.

[8] Vijay Prakash Dwivedi and Xavier Bresson. 2021. A Generalization of Transformer Networks to Graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications* (2021).

[9] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of International conference on machine learning (ICML)*. PMLR, 1263–1272.

[10] Neil Zhenqiang Gong, Ameet Talwalkar, Lester Mackey, Ling Huang, Eui Chul Richard Shin, Emil Stefanov, Elaine Shi, and Dawn Song. 2014. Joint link prediction and attribute inference using a social-attribute network. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 2 (2014), 1–20.

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NIPS)*. 1024–1034.

[12] Yu Hao, Xin Cao, Yixiang Fang, Xike Xie, and Sibo Wang. 2021. Inductive link prediction for nodes having only attribute information. In *Proceedings of International Joint Conferences on Artificial Intelligence (IJCAI)*. 1209–1215.

[13] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. *arXiv preprint arXiv:2005.00687* (2020).

[14] Xiao Huang, Jundong Li, and Xia Hu. 2017. Accelerated attributed network embedding. In *Proceedings of the 2017 SIAM international conference on data mining (SDM)*. SIAM, 633–641.

[15] Md Shamim Hussain, Mohammed J Zaki, and Dharmashankar Subramanian. 2022. Global self-attention as a replacement for graph convolution. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 655–665.

[16] Adarsh Jamadandi, Celia Rubio-Madrigal, and Rebekka Burkholz. 2024. Spectral Graph Pruning Against Over-Squashing and Over-Smoothing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=EMkrwJY2de

[17] Wei Jin, Xiaorui Liu, Yao Ma, Charu Aggarwal, and Jiliang Tang. 2022. Feature Overcorrelation in Deep Graph Neural Networks: A New Perspective. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 709–719.

[18] Jinwoo Kim, Dat Tien Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. 2022. Pure Transformers are Powerful Graph Learners. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=um2BxfgkT2_

[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of International Conference on Learning Representations (ICLR)*.

[20] Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. 2021. Rethinking Graph Transformers with Spectral Attention. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (Eds.). https://openreview.net/forum?id=huAdB-Tj4yG

[21] Jundong Li, Harsh Dani, Xia Hu, and Huan Liu. 2017. Radar: Residual Analysis for Anomaly Detection in Attributed Networks.. In *Proceedings of International Joint Conferences on Artificial Intelligence (IJCAI)*, Vol. 17. 2152–2158.

[22] Jundong Li, Liang Wu, Kewei Cheng, and Huan Liu. 2018. Streaming link prediction on dynamic atributed networks. In *Proceedings of ACM International Conference on Web Search and Data Mining (WSDM)*. Association for Computing Machinery, Inc, 369–377.

[23] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[24] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval*. 43–52.

[25] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* 3, 2 (2000), 127–163.

[26] Hoang Nt and Takanori Maehara. 2019. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550* (2019).

[27] Kenta Oono and Taiji Suzuki. 2020. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *Proceedings of International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=S1ldO2EFPr

[28] Wonpyo Park, Woong-Gi Chang, Donggeon Lee, Juntae Kim, et al. 2022. GRPE: Relative Positional Encoding for Graph Transformer. In *ICLR2022 Workshop Machine Learning for Drug Discovery*.

[29] Furong Peng, Kang Liu, Xuan Lu, Yuhua Qian, Hongren Yan, and Chao Ma. 2024. TSC: A Simple Two-Sided Constraint against Over-Smoothing. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 2376–2387.

[30] Chendi Qian, Andrei Manolache, Christopher Morris, and Mathias Niepert. 2024. Probabilistic Graph Rewiring via Virtual Nodes. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=LpvSHL9lcK

[31] Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. 2022. Recipe for a General, Powerful, Scalable Graph Transformer. *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)* 35 (2022).

[32] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of Graph Neural Network Evaluation. *CoRR* abs/1811.05868 (2018). http://arxiv.org/abs/1811.05868

[33] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. 2022. Understanding over-squashing and bottlenecks on graphs via curvature. In *Proceedings of International Conference on Learning Representations (ICLR)*.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*. 5998–6008.

[35] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proceedings of International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=rJXMpikCZ

[36] Liangtian Wan, Huijin Han, Lu Sun, Zixun Zhang, Zhaolong Ning, Xiaoran Yan, and Feng Xia. 2024. Flexible Graph Neural Diffusion with Latent Class Representation Learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 2936–2947.

[37] David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. 2018. DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic acids research* 46, D1 (2018), D1074–D1082.

[38] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *Proceedings of International conference on machine learning (ICML)*. PMLR, 6861–6871.

[39] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *Proceedings of International conference on machine learning (ICML)*. PMLR, 5453–5462.

[40] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do Transformers Really Perform Badly for Graph Representation?. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*. https://openreview.net/forum?id=OeWooOxFwDa

[41] Lingxiao Zhao and Leman Akoglu. 2020. PairNorm: Tackling Oversmoothing in GNNs. In *Proceedings of International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=rkecl1rtwB

A Over-smoothing and Over-squashing

Over-smoothing and over-squashing are both related to over-dilution in that they both pertain to the distortion of information resulting from the irregular structure of graphs. The over-dilution phenomenon is a comprehensive concept that encompasses not only the node-level representation, such as over-smoothing and over-squashing, but also the attribute-level representation, which is essential for achieving informative representations on graphs. Therefore, regardless of the circumstances that trigger over-smoothing and over-squashing, over-dilution can occur at the intra-node level when the number of attributes contained by the nodes becomes excessively large. The comparison of concepts, including over-correlation, is illustrated in Figure 4.

Comparison with over-smoothing

Over-smoothing refers to the occurrence of similar representations among the nodes after a few layers of message passing, akin to the effect of a low-pass filter [5, 23, 26, 27, 39]. This is caused by the exchange of information among nodes, which is more likely to occur when nodes have large receptive fields in multi-hop layers. In contrast, over-dilution can occur even with a single aggregation layer as described in the Case 2 of the main text. As shown in the histogram of $\delta_{\text{Agg}}^{\text{inter}}(v)$ for one hop aggregation in Figure 2(a) of the main text, some nodes already have low $\delta_{\text{Agg}}^{\text{inter}}(v)$ values. Apart from this comparison of two concepts, *NATR* alleviates the over-smoothing issue as shown in the Table 4 of the main text.

Comparison with over-squashing

Both phenomena are related in that they may result from the concentration of information from a large number of nodes. However, Over-squashing refers to the propagation of long-range information from node u to node v, while Over-dilution refers to the attenuation of information for individual nodes. Over-dilution is distinct from Over-squashing in that it can occur even in the first layer (i.e. with a small receptive field) according to the aggregation coefficients. Specifically, Over-dilution at node v can occur when $\alpha_{vv} \ll \sum_{u \in \tilde{\mathcal{N}}(v) \setminus \{v\}} \alpha_{vu}$, where α_{vu} is defined according to the topology for GCN ($\frac{1}{\sqrt{\deg(v)\deg(u)}}$) and to the attention coefficient for GAT ($\text{softmax}(\frac{\exp(a_{vu})}{\sum_{w \in \tilde{\mathcal{N}}(v)} \exp(a_{vw})}$). It is true that Over-dilution occurs more often as the range increases, but as can be seen in the histogram of $\delta_{\text{Agg}}^{\text{inter}}(v)$, it also occurs frequently in short-range.

B Intra-dilution factor

Unlike MPNNs, which assign equal weight (i.e. $\frac{1}{|\mathcal{T}_v|}$) to all attributes during the construction of the node representation, *NATR* merges attribute representations based on the attention coefficients between node and attribute representations (i.e. $\exp(Q_v K_t^\top)/\sum_{s \in \mathcal{T}_v} \exp(Q_v K_s^\top)$). To examine the ability of *NATR* to prevent over-dilution at the intra-node level, we construct synthetic dataset. While keeping the original graph topology (i.e. adjacency matrix) of the CoraML dataset, we generate a new attribute set and assign 10 attributes to each class as key attributes. The key attributes assigned to each class are highly likely to be held by the node (e.g. with $p = 0.8$), while the remaining non-key attributes are less likely to be held (e.g. with $p = 0.2$). The train, validation, test sets include 1000, 210, and 1785 nodes, respectively. Overall, for all nodes in the synthetic data's test set, $NATR_{GCN}$ amplifies the influence of key attributes in the node representation by approximately 1508.62% in comparison to non-key attributes. This approach effectively curbs over-dilution of key attributes at the intra-node level. We randomly sample 10 nodes for each class in the test set and visualize the intra-node dilution factors of $NATR_{GCN}$ in the Figure 5.

We define a new quantitative metric $\tilde{\delta}_v^{\text{intra}}(t)$ as below to examine how much *NATR* adjusts the dilution factors compared to the original ones $\hat{\delta}_v^{\text{intra}}(t) = \frac{1}{|\mathcal{T}_v|}$.

$$\tilde{\delta}_v^{\text{intra}}(t) = \frac{\delta_v^{\text{intra}}(t) - \hat{\delta}_v^{\text{intra}}(t)}{\hat{\delta}_v^{\text{intra}}(t)} * 100(\%)$$

where $\delta_v^{\text{intra}}(t)$ is the intra-node dilution factor of *NATR*. we consider the Computers dataset (3,675,081 instances of the intra-node dilution factor). $NATR_{GCN}$ enhances the importance of 38.07% cases ($\tilde{\delta}_v^{\text{intra}}(t) > 0$) compared to *GCN*, with a median gain +30.31% and a maximum gain +4005.60%. $NATR_{GCN}$ suppresses the importance of 61.92% cases ($\tilde{\delta}_v^{\text{intra}}(t) < 0$) compared to *GCN*, with median -29.46% and minimum -95.82%. In contrast to MPNNs, where attributes are equally diluted regardless of their importance, NATR is designed to prioritize important attributes ($\tilde{\delta}_v^{\text{intra}}(t) > 0$) and give less weight to unimportant ones ($\tilde{\delta}_v^{\text{intra}}(t) < 0$). The detailed statistics are reported in Table 7.

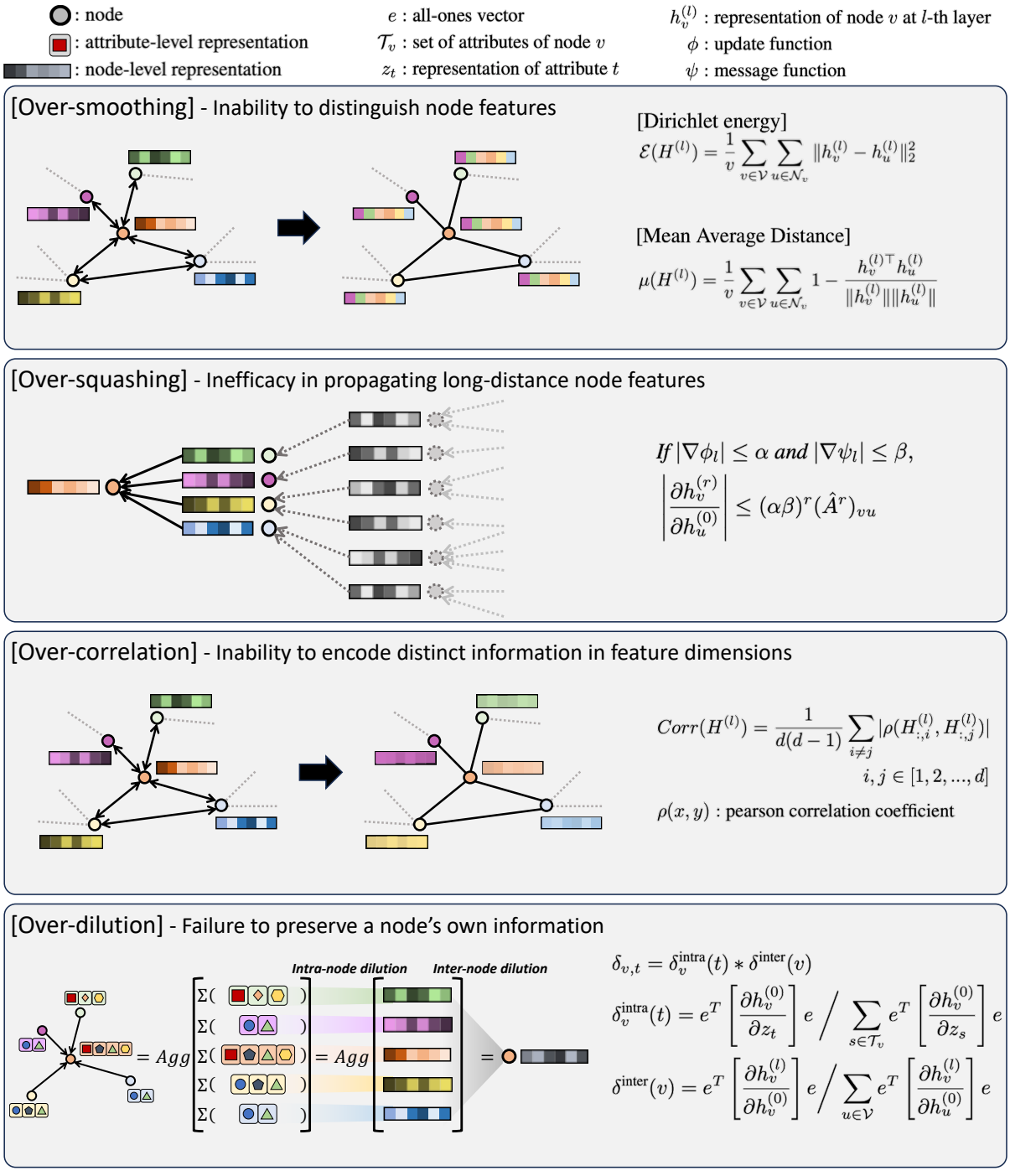

Figure 4: Comparison of the concepts: Over-smoothing, Over-squashing, Over-correlation, and Over-dilution.

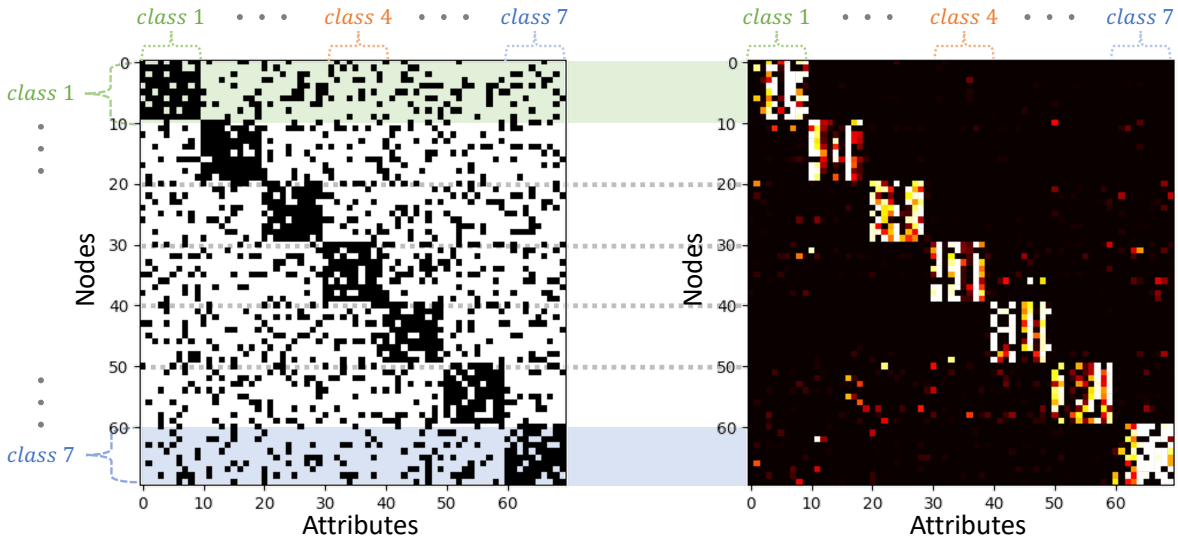

Figure 5: This figure presents a visualization of the synthetic node feature matrix (left) and the intra-node dilution factors of $NATR_{GCN}$ (right). The y-axis represents nodes sorted according to the class and the x-axis represents attributes. We randomly sample 10 nodes for each class in the test set for this visualization. We assign the 10 attributes to each class sequentially, identifying them as the block-diagonal elements in the matrix and principal attributes for class prediction. The remaining attributes are considered noise. As shown in the figure on the right, $NATR_{GCN}$ amplifies the attribute representations pertinent to each node's class, while diminishing the influence of other attributes. On average, across all nodes in the synthetic data test set, $NATR_{GCN}$ boosts the strength of the key attributes in the node representation by 1508.62% compared to non-key attributes, thus preventing their over-dilution at the intra-node level.

Table 7: (left) $NATR_{GCN}$ enhances the importance of attributes $\bar{\delta}_v^{\text{intra}}(t) > 0$. Remarkably, there are instances where the improvement ranges from 2 to 40 times compared to GCN, which constitutes 12.53% of the enhanced cases. (right) $NATR_{GCN}$ suppresses the importance of attributes $\bar{\delta}_v^{\text{intra}}(t) \leq 0$.

$\bar{\delta}_v^{\text{INTRA}}(t) > 0$	Count	Proportion	$\bar{\delta}_v^{\text{INTRA}}(t) \leq 0$	Count	Proportion
0% ∼ +10%	285,150	20.38%	0% ∼ -10%	344,872	15.15%
+10% ∼ +20%	229,552	16.41%	-10% ∼ -20%	395,988	17.40%
+20% ∼ +30%	179,963	12.86%	-20% ∼ -30%	419,467	18.43%
+30% ∼ +40%	139,937	10.00%	-30% ∼ -40%	398,607	17.51%
+40% ∼ +50%	108,893	7.78%	-40% ∼ -50%	327,231	14.38%
+50% ∼ +60%	852,85	6.10%	-50% ∼ -60%	222,725	9.79%
+60% ∼ +70%	67,096	4.80%	-60% ∼ -70%	118,015	5.19%
+70% ∼ +80%	53,009	3.79%	-70% ∼ -80%	41,757	1.83%
+80% ∼ +90%	41,827	2.99%	-80% ∼ -90%	6,988	0.31%
+90% ∼ +100%	33,254	2.38%	-90% ∼ -100%	185	0.01%
+100% ∼ +4006%	175,280	12.53%			
		100.0%			100.0%

C Inter-dilution factor

As described in Eq. (4) of the main text, the inter-node dilution factor is defined as:

$$\delta^{\text{inter}}(v) = \frac{e^T \left[\frac{\partial h_v^{(l)}}{\partial h_v^{(0)}} \right] e}{\sum_{u \in \mathcal{V}} e^T \left[\frac{\partial h_v^{(l)}}{\partial h_u^{(0)}} \right] e}$$

To observe the effect of the aggregation exclusively, we eliminate the effect of the non-linear transformation by setting all weight and initial node feature matrices to be the identity matrix. The inter-dilution factor (aggregation-only) $\delta^{\text{inter}}_{\text{Agg}}(v)$ is calculated as:

$$\delta^{\text{inter}}_{\text{Agg}}(v) = \frac{e^T \left[\frac{\partial h'^{(l)}_v}{\partial h^{(0)}_v} \right] e}{\sum_{u \in \mathcal{V}} e^T \left[\frac{\partial h'^{(l)}_v}{\partial h^{(0)}_u} \right] e}, \quad \text{where} \quad h'^{(l)} = (\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}})^l I_{N_{\mathcal{V}}} \text{ for } GCN$$

C.1 The numerator of $\delta^{\text{inter}}_{\text{Agg}}(v)$

The numerator of $\delta^{\text{inter}}_{\text{Agg}}(v)$ is calculated from:

$$\frac{\partial h'^{(l)}_v}{\partial h^{(0)}_v} = \underbrace{\prod_{i=1}^{l} \alpha^{(i)}_{vv} \cdot \frac{\partial h^{(0)}_v}{\partial h^{(0)}_v}}_{\text{for } l \geq 1} + \underbrace{\sum_{u \in \tilde{N}(v) \setminus \{v\}} \sum_{k=1}^{l-1} \left(\prod_{\substack{j=k+2 \\ k \leq l-2}}^{l} \alpha^{(j)}_{vv} \right) \alpha^{(k+1)}_{vu} \frac{\partial h'^{(k)}_u}{\partial h^{(0)}_v}}_{\text{for } l \geq 2}$$

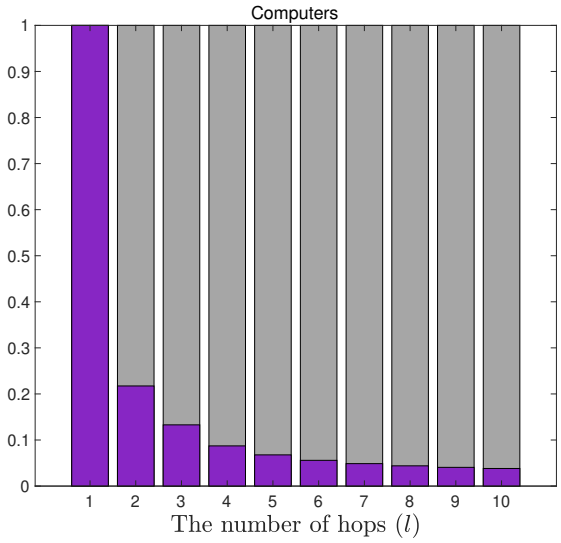

Figure 6: The purple bar represents the average ratio of the former term $e^T \left[\left(\frac{1}{\mathbf{deg}(v)} \right)^l \frac{\partial h^{(0)}_v}{\partial h^{(0)}_v} \right] e$, which is the amount (influence) of the initial node-level feature $h^{(0)}_v$ that is not propagated to other nodes. The gray bar represents the average ratio of the latter term $e^T \left[\sum_{k=1}^{l-1} \sum_{u \in \tilde{N}(v) \setminus \{v\}} \frac{1}{\sqrt{\mathbf{deg}(v)}\sqrt{\mathbf{deg}(u)}} \left(\frac{1}{\mathbf{deg}(v)} \right)^{l-k-1} \frac{\partial h'^{(k)}_u}{\partial h^{(0)}_v} \right] e$, which is the amount (influence) of the representation preserved by its neighbors. It implies that in the *GCN*, the representation of each node is primarily maintained by its neighboring nodes, with the representation being 'received back' from them after two hops aggregation.

For $l = 1$,

$$\frac{\partial h'^{(1)}_v}{\partial h^{(0)}_v} = \alpha^{(1)}_{vv} \frac{\partial h^{(0)}_v}{\partial h^{(0)}_v}$$

For $l = 2$,

$$\frac{\partial h'^{(2)}_v}{\partial h^{(0)}_v} = \alpha^{(1)}_{vv} \alpha^{(2)}_{vv} \frac{\partial h^{(0)}_v}{\partial h^{(0)}_v} + \sum_{u \in \tilde{N}(v) \setminus \{v\}} \alpha^{(2)}_{vu} \frac{\partial h'^{(1)}_u}{\partial h^{(0)}_v}$$

For $l = 3$,

$$\frac{\partial h_v'^{(3)}}{\partial h_v^{(0)}} = \alpha_{vv}^{(3)}\alpha_{vv}^{(2)}\alpha_{vv}^{(1)}\frac{\partial h_v^{(0)}}{\partial h_v^{(0)}} + \sum_{u\in\tilde{N}(v)\setminus\{v\}}\left(\alpha_{vv}^{(3)}\alpha_{vu}^{(2)}\frac{\partial h_u'^{(1)}}{\partial h_v^{(0)}} + \alpha_{vu}^{(3)}\frac{\partial h_u'^{(2)}}{\partial h_v^{(0)}}\right)$$

For $l = 4$,

$$\frac{\partial h_v'^{(4)}}{\partial h_v^{(0)}} = \alpha_{vv}^{(4)}\alpha_{vv}^{(3)}\alpha_{vv}^{(2)}\alpha_{vv}^{(1)}\frac{\partial h_v^{(0)}}{\partial h_v^{(0)}} + \sum_{u\in\tilde{N}(v)\setminus\{v\}}\left(\alpha_{vv}^{(4)}\alpha_{vv}^{(3)}\alpha_{vu}^{(2)}\frac{\partial h_u'^{(1)}}{\partial h_v^{(0)}} + \alpha_{vv}^{(4)}\alpha_{vu}^{(3)}\frac{\partial h_u'^{(2)}}{\partial h_v^{(0)}} + \alpha_{vu}^{(4)}\frac{\partial h_u'^{(3)}}{\partial h_v^{(0)}}\right)$$

For *GCN*,

$$\begin{aligned}
\frac{\partial h_v'^{(l)}}{\partial h_v^{(0)}} &= \frac{1}{\sqrt{\deg(v)}}\cdot\operatorname{diag}\left(1_{f_v^{(l)}>0}\right)\cdot W^{(l)}\cdot\sum_{u\in\tilde{N}(v)}\frac{1}{\sqrt{\deg(u)}}\frac{\partial h_u^{(l-1)}}{\partial h_v^{(0)}}\\
&= \frac{1}{\sqrt{\deg(v)}}\cdot\sum_{u\in\tilde{N}(v)}\frac{1}{\sqrt{\deg(u)}}\frac{\partial h_u'^{(l-1)}}{\partial h_v^{(0)}}\\
&= \frac{1}{\deg(v)}\frac{\partial h_v'^{(l-1)}}{\partial h_v^{(0)}} + \sum_{u\in\tilde{N}(v)\setminus\{v\}}\frac{1}{\sqrt{\deg(v)}\sqrt{\deg(u)}}\frac{\partial h_u'^{(l-1)}}{\partial h_v^{(0)}}\\
&= \frac{1}{\deg(v)}\left(\frac{1}{\deg(v)}\frac{\partial h_v'^{(l-2)}}{\partial h_v^{(0)}} + \sum_{u\in\tilde{N}(v)\setminus\{v\}}\frac{1}{\sqrt{\deg(v)}\sqrt{\deg(u)}}\frac{\partial h_u'^{(l-2)}}{\partial h_v^{(0)}}\right)\\
&\quad + \sum_{u\in\tilde{N}(v)\setminus\{v\}}\frac{1}{\sqrt{\deg(v)}\sqrt{\deg(u)}}\frac{\partial h_u'^{(l-1)}}{\partial h_v^{(0)}}\\
&= \left(\frac{1}{\deg(v)}\right)^{l}\frac{\partial h_v^{(0)}}{\partial h_v^{(0)}} + \sum_{k=1}^{l-1}\sum_{u\in\tilde{N}(v)\setminus\{v\}}\frac{1}{\sqrt{\deg(v)}\sqrt{\deg(u)}}\left(\frac{1}{\deg(v)}\right)^{l-k-1}\frac{\partial h_u'^{(k)}}{\partial h_v^{(0)}}
\end{aligned}$$

where $f_v^{(l)}$ represents the pre-activated feature of $h_v^{(l)}$ and $k, j \in \mathbb{N}$.

We can interpret the former term $\prod_{i=1}^{l}\alpha_{vv}^{(i)}\cdot\frac{\partial h_v^{(0)}}{\partial h_v^{(0)}}$ as the amount (influence) of the initial node-level feature $h_v^{(0)}$ that is not propagated to other nodes.

For the latter term $\sum_{u\in\tilde{N}(v)\setminus\{v\}}\sum_{k=1}^{l-1}\left(\prod_{\substack{j=k+2\\k\le l-2}}^{l}\alpha_{vv}^{(j)}\right)\alpha_{vu}^{(k+1)}\frac{\partial h_u^{(k)}}{\partial h_v^{(0)}}$, $\frac{\partial h_u^{(k)}}{\partial h_v^{(0)}}$ is the amount (influence) of the representation of node v that node u has at k-th layer. It is passed to node v at the next layer, which is $(k+1)$-th layer, with aggregation coefficient $\alpha_{vu}^{(k+1)}$. Then it is regarded as a portion of own representation $\frac{\partial h_v^{(k+1)}}{\partial h_v^{(0)}}$ and diluted with a factor $\prod_{\substack{j=k+2\\k\le l-2}}^{l}\alpha_{vv}^{(j)}$, which is the product of aggregation coefficients for self-loop from $(k+2)$-th layer to (l)-th layer.

Figure 6 shows the ratio between the former term and the latter term.

C.2 The denominator of $\delta_{\mathbf{Agg}}^{\mathbf{inter}}(v)$

The denominator of $\delta_{\mathrm{Agg}}^{\mathrm{inter}}(v)$ is calculated from:

$$\sum_{u\in\mathcal{V}}\frac{\partial h_v'^{(l)}}{\partial h_u^{(0)}} = \sum_{x\in\tilde{N}(v)}\sum_{u\in\mathcal{V}}\sum_{k=0}^{l-1}\left(\prod_{\substack{j=k+2\\k\le l-2}}^{l}\alpha_{vv}^{(j)}\right)\alpha_{vx}^{(k+1)}\frac{\partial h_x'^{(k)}}{\partial h_u^{(0)}}$$

For $l = 1$,

$$\sum_{u\in\mathcal{V}}\frac{\partial h_v'^{(1)}}{\partial h_u^{(0)}} = \sum_{u\in\tilde{N}(v)}\alpha_{vu}^{(1)}\frac{\partial h_u^{(0)}}{\partial h_u^{(0)}}$$

For $l = 2$,

$$\sum_{u\in\mathcal{V}}\frac{\partial h_v'^{(2)}}{\partial h_u^{(0)}} = \sum_{x\in\tilde{N}(v)}\left(\alpha_{vv}^{(2)}\alpha_{vx}^{(1)}\frac{\partial h_x^{(0)}}{\partial h_x^{(0)}} + \sum_{u\in\mathcal{V}}\alpha_{vx}^{(2)}\frac{\partial h_x'^{(1)}}{\partial h_u^{(0)}}\right)$$

For $l = 3$,

$$\sum_{u\in\mathcal{V}} \frac{\partial h_v'^{(3)}}{\partial h_u^{(0)}} = \sum_{x\in\tilde{N}(v)} \left(\alpha_{vv}^{(3)} \alpha_{vv}^{(2)} \alpha_{vx}^{(1)} \frac{\partial h_x^{(0)}}{\partial h_x^{(0)}} + \sum_{u\in\mathcal{V}} \alpha_{vv}^{(3)} \alpha_{vx}^{(2)} \frac{\partial h_x'^{(1)}}{\partial h_u^{(0)}} + \sum_{u\in\mathcal{V}} \alpha_{vx}^{(3)} \frac{\partial h_x'^{(2)}}{\partial h_u^{(0)}} \right)$$
$$= \sum_{x\in\tilde{N}(v)} \sum_{u\in\mathcal{V}} \left(\alpha_{vv}^{(3)} \alpha_{vv}^{(2)} \alpha_{vx}^{(1)} \frac{\partial h_x^{(0)}}{\partial h_u^{(0)}} + \alpha_{vv}^{(3)} \alpha_{vx}^{(2)} \frac{\partial h_x'^{(1)}}{\partial h_u^{(0)}} + \alpha_{vx}^{(3)} \frac{\partial h_x'^{(2)}}{\partial h_u^{(0)}} \right)$$

For $l = 4$,

$$\sum_{u\in\mathcal{V}} \frac{\partial h_v'^{(4)}}{\partial h_u^{(0)}} = \sum_{x\in\tilde{N}(v)} \sum_{u\in\mathcal{V}} \left(\alpha_{vv}^{(4)} \alpha_{vv}^{(3)} \alpha_{vv}^{(2)} \alpha_{vx}^{(1)} \frac{\partial h_x^{(0)}}{\partial h_u^{(0)}} + \alpha_{vv}^{(4)} \alpha_{vv}^{(3)} \alpha_{vx}^{(2)} \frac{\partial h_x'^{(1)}}{\partial h_u^{(0)}} + \alpha_{vv}^{(4)} \alpha_{vx}^{(3)} \frac{\partial h_x'^{(2)}}{\partial h_u^{(0)}} + \alpha_{vx}^{(4)} \frac{\partial h_x'^{(3)}}{\partial h_u^{(0)}} \right)$$

For *GCN*,

$$\sum_{u\in\mathcal{V}} \frac{\partial h_v'^{(l)}}{\partial h_u^{(0)}} = \sum_{x\in\tilde{N}(v)} \sum_{u\in\mathcal{V}} \sum_{k=0}^{l-1} \left(\frac{1}{\deg(v)} \right)^{l-k-1} \frac{1}{\sqrt{\deg(v)}\sqrt{\deg(u)}} \frac{\partial h_x'^{(k)}}{\partial h_u^{(0)}}$$

C.3 The final form of $\delta_{\mathbf{Agg}}^{\mathbf{inter}}(v)$

The final form of $\delta_{\text{Agg}}^{\text{inter}}(v)$ is defined as:

$$\delta_{\text{Agg}}^{\text{inter}}(v) = \frac{e^T \left[\prod_{i=1}^{l} \alpha_{vv}^{(i)} \frac{\partial h_v^{(0)}}{\partial h_v^{(0)}} + \sum_{u\in\tilde{N}(v)\setminus\{v\}} \sum_{k=1}^{l-1} \left(\prod_{\substack{j=k+2 \\ k\le l-2}}^{l} \alpha_{vv}^{(j)} \right) \alpha_{vu}^{(k+1)} \frac{\partial h_u'^{(k)}}{\partial h_v^{(0)}} \right] e}{\sum_{u\in\mathcal{V}} e^T \left[\sum_{x\in\tilde{N}(v)} \sum_{u\in\mathcal{V}} \sum_{k=0}^{l-1} \left(\prod_{\substack{j=k+2 \\ k\le l-2}}^{l} \alpha_{vv}^{(j)} \right) \alpha_{vx}^{(k+1)} \frac{\partial h_x'^{(k)}}{\partial h_u^{(0)}} \right] e}$$

For $l = 1$ (used in Case 2 of the main text),

$$\delta_{\text{Agg}}^{\text{inter}}(v) = \frac{e^T \left[\alpha_{vv}^{(1)} \frac{\partial h_v^{(0)}}{\partial h_v^{(0)}} \right] e}{e^T \left[\sum_{u\in\tilde{N}(v)} \alpha_{vu}^{(1)} \frac{\partial h_u^{(0)}}{\partial h_u^{(0)}} \right] e} = \frac{\alpha_{vv}^{(1)}}{\sum_{u\in\tilde{N}(v)} \alpha_{vu}^{(1)}}$$

For $l = 2$,

$$\delta_{\text{Agg}}^{\text{inter}}(v) = \frac{e^T \left[\alpha_{vv}^{(1)} \alpha_{vv}^{(2)} \cdot \frac{\partial h_v^{(0)}}{\partial h_v^{(0)}} + \sum_{u\in\tilde{N}(v)\setminus\{v\}} \alpha_{vu}^{(2)} \frac{\partial h_u'^{(1)}}{\partial h_v^{(0)}} \right] e}{e^T \left[\sum_{x\in\tilde{N}(v)} \sum_{u\in\mathcal{V}} \left(\alpha_{vv}^{(2)} \alpha_{vx}^{(1)} \frac{\partial h_x^{(0)}}{\partial h_u^{(0)}} + \sum_{u\in\mathcal{V}} \alpha_{vx}^{(2)} \frac{\partial h_x'^{(1)}}{\partial h_u^{(0)}} \right) \right] e}$$

Figure 7 shows that the inter-dilution factor (aggregation-only) is dramatically decreased as the number of layers and the size of the receptive field increase for all datasets. The histograms of the inter-dilution factor (aggregation-only) and the average size of the receptive field are shown in Figure 8.

For *NATR*,

$$\delta_{\text{Agg}}^{\text{inter}}(v) = \frac{e^T \left[\prod_{i=1}^{l} \alpha_{vv}^{(i)} \frac{\partial h_v^{(0)}}{\partial h_v^{(0)}} + \sum_{u\in\tilde{N}(v)\setminus\{v\}} \sum_{k=1}^{l-1} \left(\prod_{\substack{j=k+2 \\ k\le l-2}}^{l} \alpha_{vv}^{(j)} \right) \alpha_{vu}^{(k+1)} \frac{\partial h_u'^{(k)}}{\partial h_v^{(0)}} \right] e \; + e^T \left[\sum_{m=1}^{M} \frac{\partial \tilde{H}_v^{(M)}}{\partial O_v^{(m)}} \right] e}{\sum_{u\in\mathcal{V}} e^T \left[\sum_{x\in\tilde{N}(v)} \sum_{u\in\mathcal{V}} \sum_{k=0}^{l-1} \left(\prod_{\substack{j=k+2 \\ k\le l-2}}^{l} \alpha_{vv}^{(j)} \right) \alpha_{vx}^{(k+1)} \frac{\partial h_x'^{(k)}}{\partial h_u^{(0)}} \right] e + e^T \left[\sum_{m=1}^{M} \frac{\partial \tilde{H}_v^{(M)}}{\partial O_v^{(m)}} \right] e}$$

For *NATR* with $l = 1$,

$$\delta_{\text{Agg}}^{\text{inter}}(v) = \frac{e^T \left[\alpha_{vv}^{(1)} \frac{\partial h_v^{(0)}}{\partial h_v^{(0)}} \right] e \; + e^T \left[\frac{\partial \tilde{H}_v^{(1)}}{\partial O_v^{(1)}} \right] e}{e^T \left[\sum_{u\in\tilde{N}(v)} \alpha_{vu}^{(1)} \frac{\partial h_u^{(0)}}{\partial h_u^{(0)}} \right] e + e^T \left[\frac{\partial \tilde{H}_v^{(1)}}{\partial O_v^{(1)}} \right] e}$$

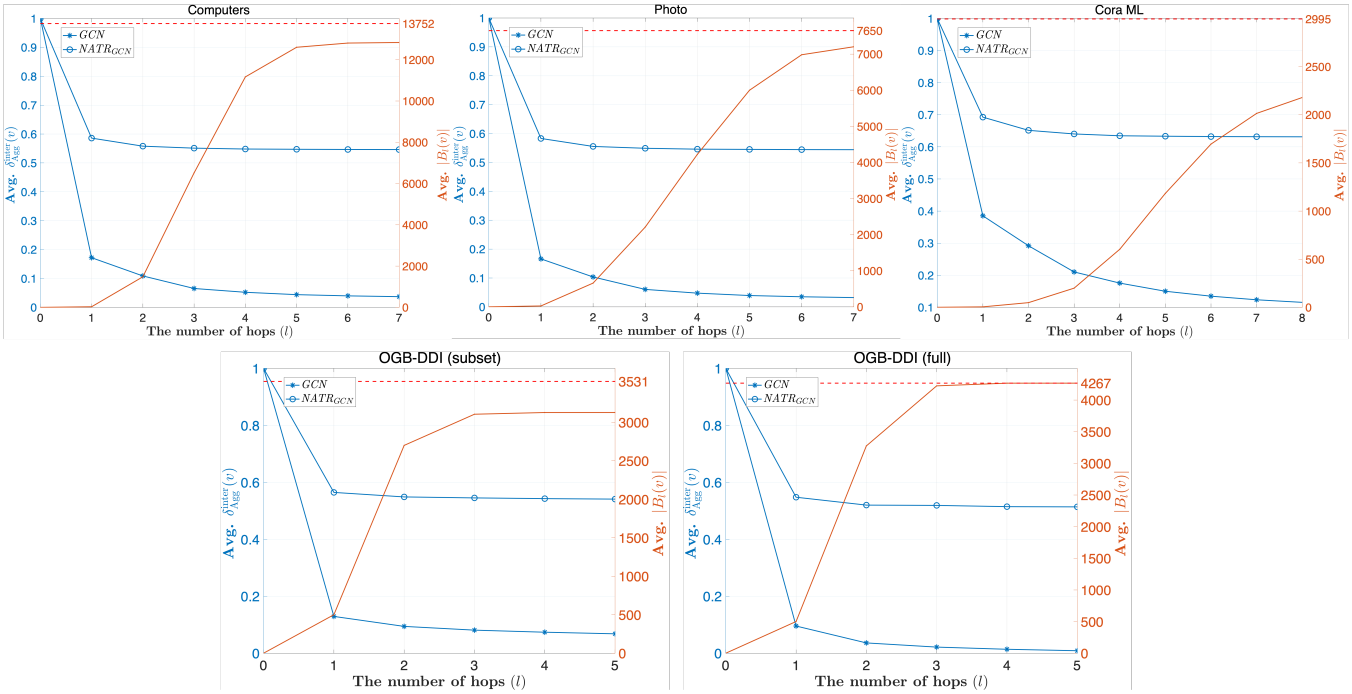

Figure 7: The average of the inter-dilution factor (aggregation-only) with the left y-axis (blue line) and the average size of the receptive field (orange line) with the right y-axis in all datasets. The x-axis represents the number of hops for the aggregation.

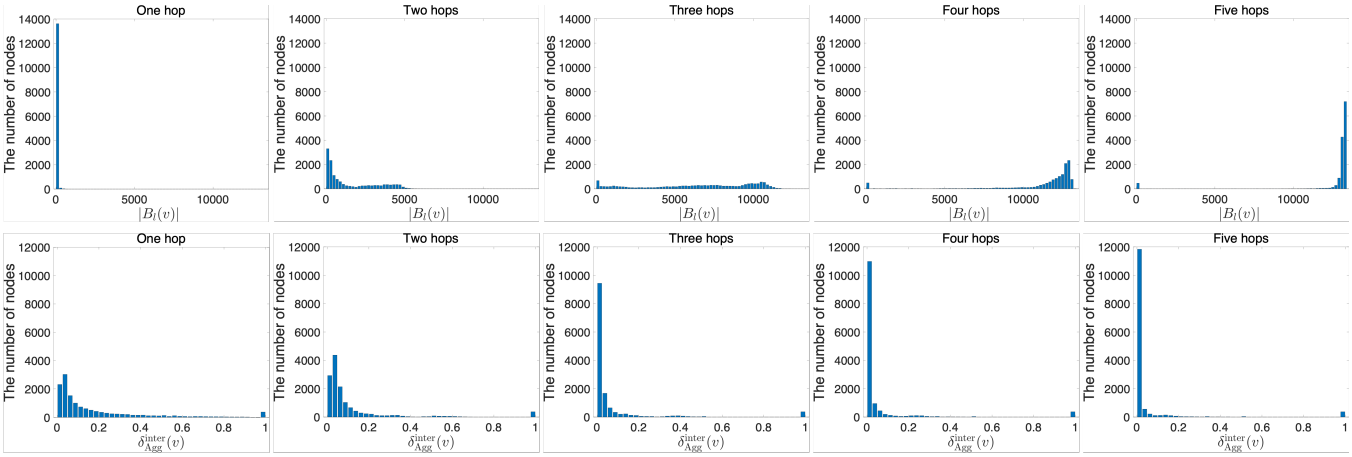

Figure 8: Histograms of the average size of the receptive field (top) and the inter-node dilution factor (aggregation-only) (bottom), from one hop (left) to five hops (right).

For *NATR* with $l = 2$,

$$\delta^{\text{inter}}_{\text{Agg}}(v) = \frac{e^T \left[\alpha^{(1)}_{vv} \alpha^{(2)}_{vv} \cdot \frac{\partial h^{(0)}_v}{\partial h^{(0)}_v} + \sum_{u \in \tilde{N}(v) \setminus \{v\}} \alpha^{(2)}_{vu} \frac{\partial h'^{(1)}_u}{\partial h^{(0)}_v} \right] e + e^T \left[\frac{\partial \tilde{H}^{(2)}_v}{\partial O^{(1)}_v} + \frac{\partial \tilde{H}^{(2)}_v}{\partial O^{(2)}_v} \right] e}{e^T \left[\sum_{x \in \tilde{N}(v)} \sum_{u \in \mathcal{V}} \left(\alpha^{(2)}_{vv} \alpha^{(1)}_{vx} \frac{\partial h^{(0)}_x}{\partial h^{(0)}_u} + \sum_{u \in \mathcal{V}} \alpha^{(2)}_{vx} \frac{\partial h'^{(1)}_x}{\partial h^{(0)}_u} \right) \right] e + e^T \left[\frac{\partial \tilde{H}^{(2)}_v}{\partial O^{(1)}_v} + \frac{\partial \tilde{H}^{(2)}_v}{\partial O^{(2)}_v} \right] e}$$

D Datasets

Table 8: Dataset statistics.

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{E}_{TRAIN}\rvert$	$\lvert\mathcal{E}_{VALID}\rvert$	$\lvert\mathcal{E}_{TEST}\rvert$
Amazon Computers	13752	767	196689	24586	24586
Amazon Photo	7650	745	95265	11908	11908
Cora ML	2995	2879	6936	407	815
OGB-DDI$_{\text{SUBSET}}$	3531	1024	882012	110446	114363
OGB-DDI$_{\text{FULL}}$	4267	1024+1	1067911	133489	133489

As described in Table 8, we split the edge set into train/valid/test without overlap as 80/10/10 for Computers and Photo [32] and 85/5/10 for Cora ML [4]. For OGB-DDI$_{\text{FULL}}$ and OGB-DDI$_{\text{SUBSET}}$ datasets, we use pre-defined data split [13]. We randomly sample the negative edges with the same number of the positive set. We assume that the node has a discrete binary vector $X_v \in \mathbb{R}^{N_{\mathcal{T}}}$, where $X_{v,t} \in \{0, 1\}$ in the paper. However, even in the dataset where each node has a continuous vector, *NATR* works just as well. The values in X_v can be interpreted as the pre-defined magnitude of each attribute in the node. We can binarize these values to fit our assumption because the pre-defined magnitude may not match the importance in representation learning. There are several methods for binarization, such as thresholding and discretization into a set of discrete bins. In another way, we can incorporate the magnitude into the attention coefficient of the decoder as $\exp(Q_v K_t^\top + X_{v,t}) / \sum_{s \in \mathcal{T}_v} \exp(Q_v K_s^\top + X_{v,s})$.

E Details on training

We conduct the grid search for hyperparameter configurations. The search space sets are $\{2, 3, 4, 5\}$ for the number of layers, $\{64, 128, 256\}$ for the hidden dimension, and $\{0.01, 0.005, 0.001, 0.0005\}$ for the learning rate. For *NATR*, d_{FFN} is set to 512 and the number of encoder layers is set to two. We apply the dropout with $p = 0.5$, 10k epochs of the optimization step, and the early stopping with 1k epochs at the hyperparameter search for all models. As done for other transformers, we train *NATR* with the auxiliary loss for the outputs from intermediate layers of the decoder.

F Compatibility with other node embedding models

Table 9: Comparison of Hits@20 on the Computer dataset with various node embedding models.

	SAGE	GATV2	ARMA	PNA$_{t=2}$	PNA$_{t=4}$	GCNII$_{\alpha=0.1}$	GCNII$_{\alpha=0.5}$	Graphormer
Node Model	32.30 ±4.32	28.69 ±2.46	31.42 ±3.07	21.77 ±3.82	29.41 ±3.56	30.00 ±3.69	27.54 ±3.27	25.94 ±4.32
NATR$_{\text{NodeModel}}$	36.11 ±3.75	37.52 ±3.37	37.23 ±3.91	31.62 ±2.66	33.68 ±4.03	37.47 ±4.56	36.78 ±3.03	30.71 ±3.68

The *NATR* architecture can be attached to various node embedding models for improving performance. Any kind of models for learning node-level representations can be exploited as *NodeModule* in the decoder of *NATR*. We conduct extended comparison with various node embedding models and report the results in Table 9. *SAGE* [11] aggregates node representations with the coefficient defined as $\alpha_{vv} = \alpha_{vu} = \frac{1}{\deg(v)}$ and *ARMA* [3] indicates the graph convolution inspired by the auto-regressive moving average filter which is robust to noise and better to capture the global graph structure. *PNA* [7] combines multiple aggregators with scalers for degree, which is a generalized sum aggregator. *GCNII* [6] updates node representations as $h^{(l)} = \left((1-\alpha^{(l)})\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}h^{(l-1)} + \alpha^{(l)}h^{(0)}\right)((1-\beta^{(l)})I + \beta^{(l)}\Theta)$ while preserving the initial feature $h^{(0)}$. The *NATR* architecture shows performance gains through the integration of various node embedding models.

G Investigation of the effectiveness of *NATR*, including ablation studies

Table 10: Ablation studies for *NATR* with the link prediction performance on the Computers dataset.

	Encoder		Decoder		Aux Loss	Hits@20	Remarks
	Num. Layers	Type	Num. Layers	Node Module			
(ref1)	2	SelfAttn	5	*GCN*	✓	42.38 ±3.21	-
(ref2)	2	SelfAttn	2	*GCN*	✓	39.81 ±2.88	-
(*A*)	2	SelfAttn	5	*MLP*	✓	34.89 ±3.42	*MLP for Node Embedding (w/o aggregation)*
(*B*)	2	*MLP*	5	*GCN*	✓	42.03 ±3.06	*MLP for Encoder (w/o correlation of attributes)*
(*C*)	2	*MLP*	5	*MLP*	✓	33.26 ±3.20	*MLP Node Embedding, MLP Encoder*
(*D*)	2	SelfAttn	5	*GCN*	✗	22.50 ±3.32	*w/o intermediate loss*
(*E*)	2	SelfAttn	2	*GCN*	✗	33.04 ±4.09	*Two decoder layers w/o intermediate loss*
(*F*)	1	SelfAttn	5	*GCN*	✓	42.66 ±3.95	*The number of encoder layers*
(*G*)	3	SelfAttn	5	*GCN*	✓	42.62 ±3.55	*The number of encoder layers*
(*H*)	4	SelfAttn	5	*GCN*	✓	43.15 ±2.94	*The number of encoder layers*
(*I*)	5	SelfAttn	5	*GCN*	✓	43.00 ±3.43	*The number of encoder layers*
(*J*)	6	SelfAttn	5	*GCN*	✓	42.34 ±2.87	*The number of encoder layers*
(*K*)	2	SelfAttn	3	*GCN*	✓	41.54 ±3.70	*The number of decoder layers*
(*L*)	2	SelfAttn	4	*GCN*	✓	40.96 ±4.19	*The number of decoder layers*
(*M*)	2	SelfAttn	6	*GCN*	✓	41.06 ±4.36	*The number of decoder layers*
(*N*)	2	SelfAttn	7	*GCN*	✓	44.30 ±3.54	*The number of decoder layers*
(*O*)	2	SelfAttn	8	*GCN*	✓	43.20 ±2.95	*The number of decoder layers*
(*P*)	2	SelfAttn	12	*GCN*	✓	43.38 ±2.94	*The number of decoder layers*
(*Q*)	1	SelfAttn	1	*GCN*	✗	34.98 ±3.77	*Single layer for both encoder and decoder*
(*R*)	1	SelfAttn	1	*GAT*	✗	36.49 ±3.41	*Single layer for both encoder and decoder*

To demonstrate the effectiveness of *NATR*, we report the comparison of models with various conditions in Table 6 of the main text. The illustration to help understanding each model in Table 6 of the main text is shown in Figure 9. The *MLP* model, (A) produces the node-level representation by mixing attribute representations **equally**, without the message propagation. The (A) model suffers from the intra-node dilution but not the inter-node dilution with high value of the inter-node dilution factor ($\delta^{\text{inter}}_{\text{Agg}}(v) = 1$ for all nodes) and the low value of the intra-node dilution factor $\delta^{\text{intra}}_{v}(t) = 1/|\mathcal{T}_v|$. The $NATR_{MLP}$ models, (D) and (E) also do not propagate node representations, but instead mix attribute representations according to attention coefficients (importance of each attribute). This comparison also reveals that *NATR* effectively alleviates the intra-node dilution.

We also provide comprehensive ablation studies with various configurations in Table10. The effectiveness of the node-level aggregation in *NATR* is explained by (a)→(ref1), with a performance gain about 7.49 (21.5%). It implies the importance of both preserving attribute-level representation and aggregating node-level representation. The effectiveness of the encoder (i.e. the correlation between attributes) is demonstrated by (c)→(a) (+1.63, 4.9%) and (b)→(h) (+1.12, 2.7%). The intermediate loss is crucial as the number of decoder layers increases, demonstrated by (d)→(ref1) (+19.88, 88.4%) and (e)→(ref2) (+6.77, 20.5%). We fix the number of encoder layers as two and the number of decoder layers less than six when comparing to MPNNs under controlled conditions in the paper. However, (f)-(j) and (m)-(p) show the potential of *NATR* to boost performance.

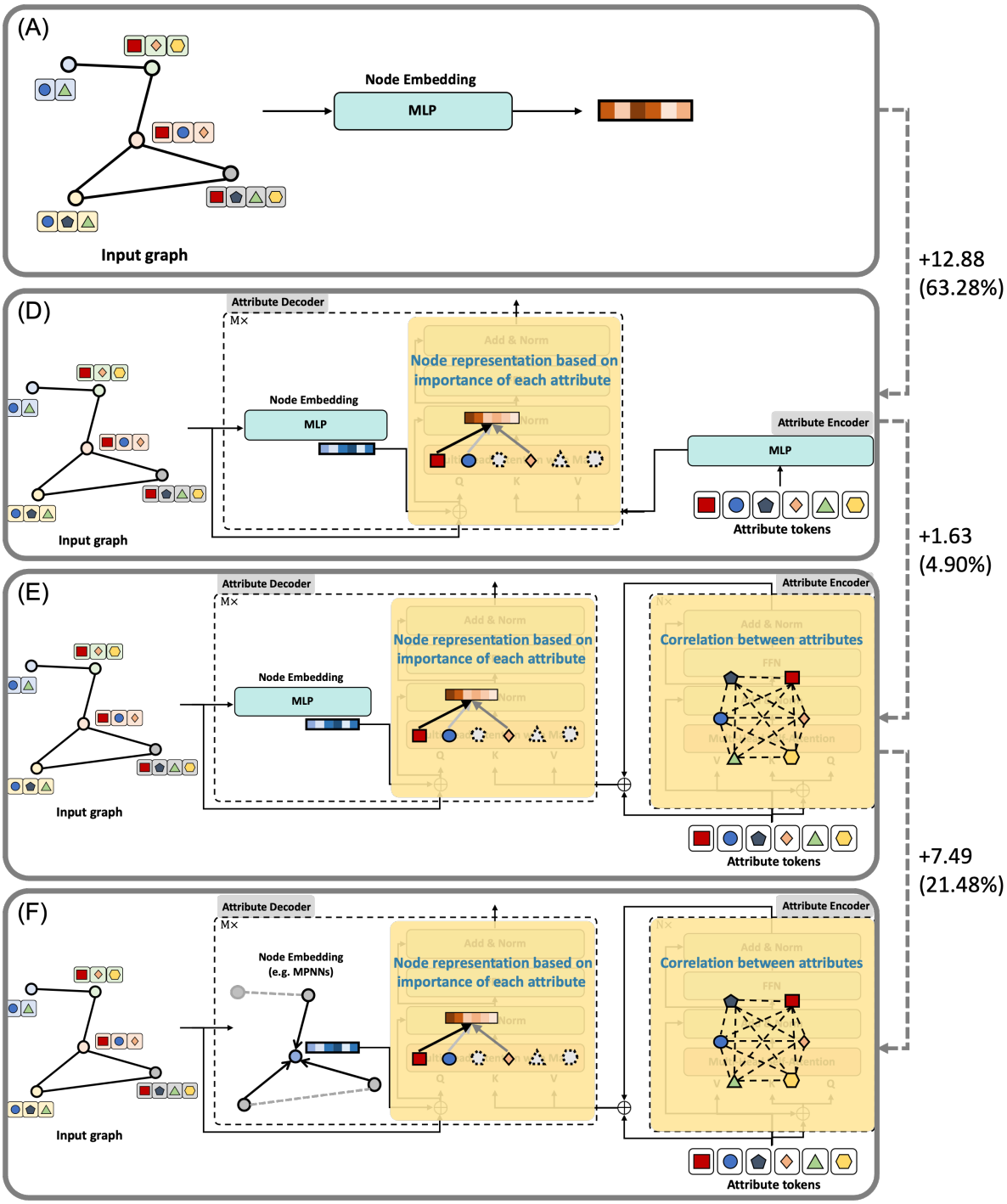

Figure 9: The illustration for models in Table 6 of the paper. Compared to *MLP* model (A), the $NATR_{MLP}$ with *MLP* encoder (D) alleviates the intra-node dilution by mixing attribute representations based on the attention coefficients between the node-level representation and attribute representations, improving performance about 12.88 (63.28%). The model (E) considers the correlation between attributes by using SelfAttn **as the attribute encoder and improves performance about 1.63 (4.90%) over the (D) model. The complete model (F), $NATR_{GCN}$ is equipped with MPNNs (*GCN*) as a node embedding module in the attribute decoder. (F) produces node-level representations on the context of graph structure and uses them as queries for attribute representations. This brings the performance gain about 7.49 (21.48%) over the model (E).**

H Complexity

In this section, we analyze the computational complexity which can be a limitation of *NATR* due to quadratic computation. The computational complexity with omitting d and l is depending on $O(|\mathcal{T}|^2)$ for the attribute encoder and $O(|\mathcal{V}||\mathcal{T}|)$ for the attribute decoder, and $O(|\mathcal{V}|+|\mathcal{E}|)$ for MPNNs. Even though the complexity of the attribute encoder is quadratic to the number of attributes, the number of attributes is relatively smaller than the number of nodes as shown in Table 8 because the attributes are specific information about the nodes, and the number of attributes is rarely increased unlike the number of nodes which can increase with the expansion of the graph. As seen in Table 11, the attribute encoder with SelfAttn has a slightly higher practical computation time of about 1 ms compared to the encoder with *MLP*.

Table 11: P: the number of parameters, S: model size (actual disk size in MB), I: inference time (ms) are reported with $d = 128, d_{FFN} = 512, N_{ENC} = 2$ on Computers dataset. The average of the practical inference time is measured 10k times on GPU - RTX 8000.

	2 Layers			3 Layers			4 Layers			5 Layers		
	P	S	I	P	S	I	P	S	I	P	S	I
$NATR_{GCN}$ w/ *MLP* Enc	830,592	3.18	16.78	1,062,144	4.07	24.98	1,293,696	4.96	33.19	1,525,248	5.85	41.41
$NATR_{GCN}$ w/ SelfAttn Enc	1,194,368	4.58	17.60	1,409,408	5.40	25.80	1,624,448	6.23	33.99	1,839,488	7.05	42.19
$NATR_{GAT}$ w/ *MLP* Enc	831,104	3.19	22.45	1,062,912	4.08	34.04	1,294,720	4.97	45.36	1,526,528	5.86	56.23
$NATR_{GAT}$ w/ SelfAttn Enc	1,194,880	4.58	23.27	1,410,176	5.41	34.48	1,625,472	6.24	46.04	1,840,768	7.06	56.80
$NATR_{SGC}$ w/ *MLP* Enc	813,824	3.12	18.10	1,028,608	3.94	26.10	1,243,392	4.76	34.35	1,458,176	5.59	42.38
$NATR_{SGC}$ w/ SelfAttn Enc	1,177,600	4.51	18.95	1,375,872	5.27	27.10	1,574,144	6.03	35.13	1,772,416	6.79	42.98

I Plug-in version with the pre-trained model

As described the main text, we introduce the plug-in version as a variant of *NATR* for single-layer models such as *SGC*. The plug-in version architecture can be also applied in scenarios where the node embedding model already exists. For example, if a trained model is already being utilized for service execution and it is infeasible to re-train or incorporate it into the decoder of *NATR*, it can be utilized as a query generator of the plug-in version. As shown in Figure 11, we train the *GCN* (green line) with the best configuration and attach it to the plug-in version of *NATR* (orange line). We can find that the $NATR^{pre}_{GCN}$ which is trained from the pre-trained *GCN* converges faster than the $NATR^{scratch}_{GCN}$ which is trained from scratch. It implies that *NATR* architecture is beneficial even for the pre-trained models. The $NATR^{pre}_{GCN}$ model and the $NATR^{scratch}_{GCN}$ show the performance on test set 38.830 ±4.026 and 38.171 ±3.629, respectively.

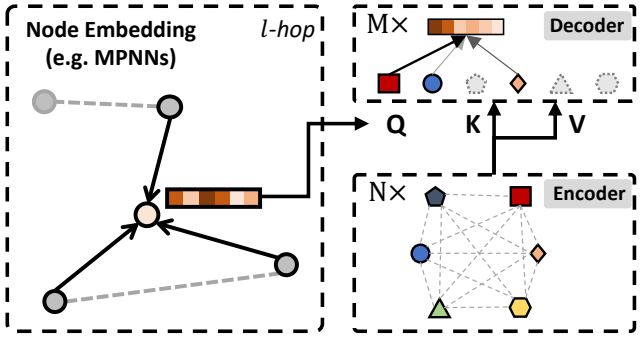

Figure 10: Plug-in version of *NATR*. The node embedding module is operated separately from the decoder to generate queries.

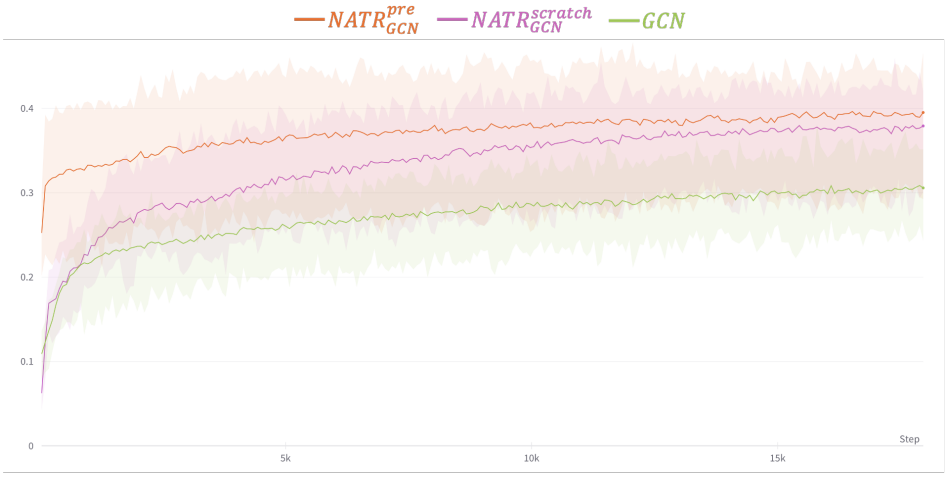

Figure 11: Hits@20 performance on validation set in the Computers dataset. The orange line (top) and the purple line (middle) indicate the performance of *NATR* trained from a pre-trained model and from scratch, respectively. The green line (bottom) indicates the performance of the *GCN* model.

J Related works

J.1 Graph Transformers

NATR is a transformer that can be applied to graph-structured data. Recently, there have been numerous endeavors to utilize the benefits of transformer architecture in the contexts of node embedding and graph embedding such as Graphormer, GRPE, EGT, SAN, TokenGT, and GraphGPS [8, 15, 18, 20, 28, 31, 40]. However, the comparison with graph transformers is out of main scope in this work because we focus on the alleviation of over-dilution phenomenon. The transformer in *NATR* accepts attribute-level representations as inputs (specifically, key and value) to alleviates the over-dilution issue while others take node-level representations for long-range dependencies. It is noteworthy that our model, owing to its general architecture, can be seamlessly integrated with the graph transformers. As reported in Table 9, we exploit Graphormer [40] as a node embedding module in the decoder of $NATR_{Graphormer}$. The graph transformers, which are used to generate graph-level representations, can utilize *NATR* to generate node-level tokens rather than using graph transformers as a node embedding module of *NATR*. In this manner, existing graph transformers and *NATR* can be employed in a complementary fashion.