Towards Deep Attention in Graph Neural Networks: Problems and Remedies

Soo Yong Lee [1]   Fanchen Bu [2]   Jaemin Yoo [3]   Kijung Shin [1 2]

Abstract

Graph neural networks (GNNs) learn the representation of graph-structured data, and their expressiveness can be further enhanced by inferring node relations for propagation. Attention-based GNNs infer neighbor importance to manipulate the weight of its propagation. Despite their popularity, the discussion on deep graph attention and its unique challenges has been limited. In this work, we investigate some problematic phenomena related to deep graph attention, including vulnerability to over-smoothed features and smooth cumulative attention. Through theoretical and empirical analyses, we show that various attention-based GNNs suffer from these problems. Motivated by our findings, we propose AERO-GNN, a novel GNN architecture designed for deep graph attention. AERO-GNN provably mitigates the proposed problems of deep graph attention, which is further empirically demonstrated with (**a**) its adaptive and less smooth attention functions and (**b**) higher performance at deep layers (up to 64). On 9 out of 12 node classification benchmarks, AERO-GNN outperforms the baseline GNNs, highlighting the advantages of deep graph attention. Our code is available at https://github.com/syleeheal/AERO-GNN.

1. Introduction

Graph neural networks (GNNs) are a class of neural networks for representation learning on graph-structured data. Recently, GNNs have been successfully applied to a wide range of graph-related tasks, including social influence prediction (Qiu et al., 2018), traffic forecast (Derrow-Pinion et al., 2021), physical system modeling (Sanchez-Gonzalez et al., 2020), product recommendation (Wu et al., 2022), and drug discovery (Stokes et al., 2020).

GNNs widely adopt message-passing frameworks, composed of two main pillars: feature transformation and propagation (a.k.a. neighborhood aggregation) (Wu et al., 2019; Gilmer et al., 2017). Feature transformation updates node features from previous layers' features. Propagation involves each node passing its own features to its neighbors, and for each node, the passed-down neighbor features are aggregated to update its own node features. In this framework, a propagation layer determines the propagation weight for each adjacent node pair, and each additional layer allows nodes to propagate to one more hop of neighbors.

Attention-based GNNs aim to learn to propagate by inferring the relational importance between node pairs. Graph attention infers relational importance for node pairs. Among many, GAT and its variants (Veličković et al., 2018; Wang et al., 2019; Brody et al., 2021; Shi et al., 2021; Kim & Oh, 2021; Bo et al., 2021; Wang et al., 2021a; Yang et al., 2021; He et al., 2021) learn edge attention. The *edge-attention* models infer the importance, or weight, of each neighbor w.r.t. each source node by applying an attention mechanism between adjacent node pairs. During propagation, each source node aggregates features from its neighbors based on the inferred importance.

Another class of GNNs, which we call *hop-attention* (or *layer-attention*) models, learns the relative importance of each hop (Liu et al., 2020; Chien et al., 2021; Zhang et al., 2022; Chanpuriya & Musco, 2022). Hop-attention models apply attention coefficients at every propagation layer to express the relative importance of a given hop in determining the final node features. Thereby, hop-attention models learn which hops, among multi-hop neighbors, each source node should attend to during propagation. Intuitively, edge-attention models learn importance *within* each hop, and hop-attention models learn importance *of* each hop.

Can the existing attention-based GNNs learn expressive attention over deep layers? Prior research does not provide a clear answer. While many studies report the over-smoothing problem of *node features* at deep layers (Li et al., 2018;

[1]Kim Jaechul Graduate School of Artificial Intelligence, KAIST, Daejeon, Republic of Korea [2]School of Electrical Engineering, KAIST, Daejeon, Republic of Korea [3]Heinz College of Information Systems and Public Policy, Carnegie Mellon University, Pittsburgh, PA, USA. Correspondence to: Kijung Shin <kijungs@kaist.ac.kr>.

Chen et al., 2020a;b; Liu et al., 2020), they do not address how it may relate to *graph attention*. Some works focus on designing expressive graph attention layers (Brody et al., 2021; Shi et al., 2021; Kim & Oh, 2021; Bo et al., 2021; Yang et al., 2021; He et al., 2021), but their scope is limited to shallow model depth. Even for attention-based GNNs that generalize their performance over deep layers (Liu et al., 2020; Wang et al., 2021a; Chien et al., 2021; Zhang et al., 2022), they do not explicitly discuss properties related to deep attention. Very recently, Zhao et al. (2023) explored the relationship between model depth and attention function for graph transformers. Their analysis, however, was confined to the relationship between model depth and transformer-style attention to substructures. In short, the theoretical underpinnings to understand deep graph attention are under-explored.

In this work, we investigate some problematic phenomena related to deep graph attention. Through both theoretical and empirical analyses, we show that the *vulnerability to over-smoothed features* and *smooth cumulative attention* limit the graph attention from becoming expressive over deep layers. Specifically, several representative attention-based GNNs, including GATv2 (Brody et al., 2021), FAGCN (Bo et al., 2021), GPRGNN (Chien et al., 2021), and DAGNN (Liu et al., 2020), suffer from these problems.

Motivated by our analyses, we propose a novel graph attention architecture, <u>A</u>ttentive d<u>E</u>ep p<u>RO</u>pagation-GNN (AERO-GNN). We theoretically demonstrate that AERO-GNN can mitigate the stated problems, which is further elaborated empirically by AERO-GNN's (**a**) adaptive and less smooth attention functions and (**b**) higher performance at deep layers (up to 64). On 9 out of 12 node classification benchmarks, including both homophilic and heterophilic graphs, AERO-GNN outperforms all the baselines, highlighting the advantages of deep graph attention.

In summary, our central contributions are two-fold:

1. **Theoretical Findings.** We formulate two theoretical limitations of deep graph attention. AERO-GNN provably mitigates the problems, whereas the representative attention-based GNNs inevitably suffer from them.

2. **Empirical Findings.** AERO-GNN shows superior performance in node classification benchmarks. Also, compared to the representative attention-based GNNs, AERO-GNN learns more adaptive and less smooth attention functions at deep layers.

2. Preliminaries

Graphs. Let $G = (V, E)$ be a graph with node set V and edge set $E \subseteq \binom{V}{2}$.[1] Let $n = |V|$ and $m = |E|$ denote the

---

[1]We assume undirected, unweighted graphs, but our theoretical

number of nodes and edges, resp. WLOG, we assume $V = [n] = \{1, 2, \ldots, n\}$. Let $A = A(G) \in \{0,1\}^{n \times n}$ denote the adjacency matrix of G, where $A_{ij} = 1$ iff $(i, j) \in E$, and we use $D = \mathrm{diag}(d_1, d_2, \ldots, d_n)$ to denote the degree matrix of G, where d_i is the degree of node i. Let $X \in \mathbb{R}^{n \times d_x}$ denote the initial node feature matrix, where each node $i \in V$ has node feature X_i of dimension d_x.

Message-Passing GNNs. Feature transformation and propagation are the two main building blocks of message-passing GNNs (Wu et al., 2019; Gilmer et al., 2017). Feature transformation updates the features of each node based on the node's features of previous layers. A feature transformation layer can be expressed as: $H^{(k)} = \sigma(H^{(k-1)}W^{(k)}), \forall 1 \leq k \leq k_{max}$,[2] where k denotes the index of each layer with k_{max} being the total number of layers, $H^{(k)} \in \mathbb{R}^{n \times d_H}$ is the *hidden node feature matrix* at layer k, $W^{(k)}$ is the weight matrix at layer k, and σ is an activation function.

On the other hand, a propagation layer passes node features to its neighbors, and each node's features are updated as an aggregation of the features. Specifically, a propagation layer can be expressed as follows: $H^{(k)} = \tilde{A}H^{(k-1)}, \forall 1 \leq k \leq k_{max}$, where $\tilde{A} = (D + I)^{-1/2}(A + I)(D + I)^{-1/2}$ is the symmetrically normalized adjacency matrix with self-loops.

Many GNNs fuse the two operations in one layer, such that $H^{(k)} = \sigma(\tilde{A}H^{(k-1)}W^{(k)})$ (Welling & Kipf, 2016; Veličković et al., 2018; Chen et al., 2020b). Some others use only propagation layers to update $H^{(k)}$, after obtaining $H^{(0)}$ with feature transformation (Gasteiger et al., 2018; Liu et al., 2020; Chien et al., 2021). Note that $\tilde{A}$ determines the magnitude in which each node's features propagate to its neighbors, and the number of propagation layers determines the number of hops to propagate to.

Edge Attention. Edge-attention GNNs (e.g., GAT and its variants) learn an edge-attention matrix $\mathcal{A}^{(k)} = (\alpha_{ij}^{(k)}) \in \mathbb{R}^{n \times n}$ at each propagation layer (i.e., hop) k,[3] where each edge attention coefficient $\alpha_{ij}^{(k)}$ can be seen as the importance of node j w.r.t node i at layer k. At each propagation layer, the edge attention coefficients are used to weigh propagation between each adjacent node pair.

Hop Attention. Hop-attention GNNs learn a hop-attention matrix $\Gamma^{(k)} = \mathrm{diag}(\gamma_1^{(k)}, \gamma_2^{(k)}, \ldots, \gamma_n^{(k)}) \in \mathbb{R}^{n \times n}$ at each propagation layer k. With hop attention, different importance $\gamma_i^{(k)}$ can be assigned at different layers k for every node i. Typically,

$$Z^{(k)} = \sum_{\ell=0}^{k} \Gamma^{(\ell)} H^{(\ell)}, \forall 1 \leq k \leq k_{max}, \quad (1)$$

$$= \sum_{\ell=0}^{k} \Gamma^{(\ell)} \tilde{A}^{\ell} H^{(0)}, \forall 1 \leq k \leq k_{max}, \quad (2)$$

---

results can be easily extended to directed and/or weighted graphs.

[2]Usually, $H^{(0)}$ is a function on the initial features, i.e., X.

[3]Single-head attention is assumed for simplicity.

Table 1: Attention Functions of the Attention-Based GNNs

Model	Pre-normalized Edge Attention	Normalized Edge Attention	Hop Attention
GATv2	$\check{\alpha}_{ij}^{(k)} = \exp((W_{edge}^{(k)})^\top \sigma(H_i^{(k-1)}\|H_j^{(k-1)}))$	$\alpha_{ij}^{(k)} = \check{\alpha}_{ij}^{(k)}/\sum_{j' \in N(i)} \check{\alpha}_{ij'}^{(k)}$	$\gamma_i^{(k)} = 1$
FAGCN	$\check{\alpha}_{ij}^{(k)} = \tanh((W_{edge}^{(k)})^\top (Z_i^{(k-1)}\|Z_j^{(k-1)}))$	$\alpha_{ij}^{(k)} = \check{\alpha}_{ij}^{(k)}/\sqrt{d_i d_j}$	$\gamma_i^{(k)} = c_\gamma, \forall k, i$ (*)
GPRGNN	$\check{\alpha}_{ij}^{(k)} = 1$	$\alpha_{ij}^{(k)} = 1/\sqrt{d_i d_j}$	$\gamma_i^{(k)} = \gamma_j^{(k)} = c_{k;\gamma}, \forall k$ (**)
DAGNN	$\check{\alpha}_{ij}^{(k)} = 1$	$\alpha_{ij}^{(k)} = 1/\sqrt{d_i d_j}$	$\gamma_i^{(k)} = \mathrm{sigmoid}(W_{hop}^\top H_i^{(k)})$ (***)

(*) for FAGCN, c_γ is a hyperparameter fixed before training and is identical for all k and i
(**) for GPRGNN, $c_{k;\gamma}$ is a learnable variable that can be different for different k, but is identical for all i given any fixed k
(***) for DAGNN, the same W_{hop} is used for each layer

Table 2: Propagation of the Attention-Based GNNs

Model	k-Layer Propagation	Rephrased Propagation w.r.t $T^{(k)}$'s
GATv2	$H^{(k)} = (\prod_{\ell=k}^1 \mathcal{A}^{(\ell)}) X (\prod_{\ell'=1}^k W^{(\ell')})$	$H^{(k)} = T^{(k)} X \prod_{\ell=1}^k W^{(\ell)}$
FAGCN	$Z^{(k)} = (\Gamma + \Gamma\mathcal{A}^{(k)} + \Gamma\mathcal{A}^{(k)}\mathcal{A}^{(k-1)} + \cdots + \Gamma\prod_{\ell=k}^1 \mathcal{A}^{(\ell)}) H^{(0)}$ (*)	$Z^{(k)} = \sum_{\ell=0}^k T^{(\ell)} H^{(0)}$
GPRGNN/DAGNN	$Z^{(k)} = (\Gamma^{(0)} + \Gamma^{(1)}\mathcal{A} + \Gamma^{(2)}\mathcal{A}^2 + \cdots + \Gamma^{(k)}\mathcal{A}^k) H^{(0)}$ (**)	$Z^{(k)} = \sum_{\ell=0}^k T^{(\ell)} H^{(0)}$

(*) recall that for FAGCN, $\Gamma^{(k)}$ is identical for all k, and we use a simplified notation Γ to denote all $\Gamma^{(k)}$'s
(**) recall that for GPRGNN and DAGNN, $\mathcal{A}^{(k)}$ is identical for all k, and we use $\mathcal{A}$ to denote all $\mathcal{A}^{(k)}$'s (here, $\mathcal{A}^k$ is the k-th power of $\mathcal{A}$)

where the sum of hidden feature matrix $H^{(\ell)}$'s up to layer k, each weighted by the hop attention matrix $\Gamma^{(\ell)}$, expresses the *layer-aggregated* node feature matrix $Z^{(k)}$. Equation (2) highlights that the hop attention matrix $\Gamma^{(\ell)}$ can be seen as learning weights of ℓ hop neighbors expressed in $\tilde{A}^\ell$ (A similar analysis can be found in Dong et al. (2021a)).

3. Theoretical Analysis on Deep Attention

Can attention functions of the representative attention-based GNNs remain expressive over deeper layers? Prior research has suggested possible reasons for the performance degradation of deep GNNs, including the over-smoothing of node features, over-squashing (Alon & Yahav, 2021), and over-correlation (Jin et al., 2022). However, discussion dedicated to theoretical limitations of *deep graph attention* has been little. In this section, we formulate two theoretical limitations of the representative attention-based GNNs concerning their ability to remain expressive over deeper layers.

3.1. A Systematic Understanding of Graph Attention

A systematic understanding of graph attention, integrating edge and hop attention, would allow us to discuss various attention-based GNNs and their theoretical limitations within the same framework.

Cumulative Attention. We have observed that edge and hop attention achieve the same goal (i.e., inferring the relational importance between node pairs), but from two different perspectives. Consequently, they both learn weights in which each node's features propagate to its neighbors. This observation motivates us to integrate edge and hop attention under the same umbrella. To this end, we propose a concept of *cumulative attention matrix*, denoted by $T^{(k)}$, which intuitively represents attention between all node pairs within

k hops (or equivalently, at layer k) that considers both edge and hop attentions.

Formally, given any k_{max}, for each $0 \le k \le k_{max}$,

$$T^{(k)} = \Gamma^{(k)} \prod_{\ell=k}^1 \mathcal{A}^{(\ell)}, \qquad (3)$$

where $T^{(0)} = \Gamma^{(0)}$.[4]

Rephrased Propagation w.r.t $T^{(k)}$'s. Here, we discuss various representative attention-based GNNs (spec., GATv2, FAGCN, GPRGNN, and DAGNN). We detail their attention functions in Table 1. Those models are used throughout our theoretical analyses and empirical evaluation.

GATv2 and FAGCN learn edge attention $\alpha_{ij}^{(k)}$. GATv2 uses hidden features $(H_i\|H_j)$, whereas FAGCN layer-aggregated features $(Z_i\|Z_j)$, in computing each edge attention coefficient $\alpha_{ij}^{(k)}$. Notably, their $\alpha_{ij}^{(k)}$ have different bounds, such that GATv2's $\alpha_{ij}^{(k)} \in (0,1)$ and FAGCN's $\alpha_{ij}^{(k)} \in (-1,1)$. They do not learn hop attention coefficients, which can be expressed as a constant (1 for GATv2 and c_γ for FAGCN).

On the other hand, GPRGNN and DAGNN learn hop attention $\gamma_i^{(k)}$. GPRGNN's hop attention is not explicitly bounded and, thus, can learn negative hop attention $\gamma_i^{(k)}$. While $\gamma_i^{(k)} \in (0,1)$ for DAGNN, it learns *node-adaptive* hop attention. Neither GPRGNN nor DAGNN learns nontrivial edge attention, and their edge attention coefficients are equivalently degree-normalized constants, i.e., $\alpha_{ij}^{(k)} = 1/\sqrt{d_i d_j}, \forall i, j, k$.

The discussed GNNs consist of a representative and diverse set of attention functions, yet we can succinctly rephrase them w.r.t $T^{(k)}$ (see Table 2).[5]

[4]For FAGCN, $T^{(k)} = \Gamma^{(k)} \prod_{\ell=k_{max}}^{k_{max}-k+1} \mathcal{A}^{(\ell)}$. However, in our theoretical analyses, it is equivalent to Eq. (3) when k_{max} is fixed.

[5]FAGCN and GATv2's propagation formulae are rewritten from their original form. Refer to Appendix B for the details.

3.2. Theoretical Problems of Deep Graph Attention

Two assumptions are used throughout our theoretical analyses, and proper normalization and preprocessing may always satisfy them in practice.

Assumption 1. The graph G is connected and non-bipartite, and the initial node features in X are pairwise distinct (i.e., $X_i \neq X_j, \forall i \neq j \in V$) and finite (i.e., $\|X_i\|_F < \infty, \forall i \in V$).

Assumption 2. There exists a global constant $C_{param} > 0$ such that, for each considered GNN model, each entry of each parameter matrix or vector (e.g., $W^{(k)}$'s) and each entry of each intermediate variable (e.g., $H^{(k)}$'s and $Z^{(k)}$'s) is bounded in $[-C_{param}, C_{param}]$.

Problem 1: Vulnerability to Over-Smoothing. In the first problem, we establish a connection between attention functions[6] and the node feature over-smoothing. Specifically, we examine the *vulnerability* and *resistance* of attention functions to over-smoothed node features. Informally, if the pre-normalized edge attention coefficients $\check{\alpha}_{ij}^{(k)}$'s (resp., hop attention coefficients $\gamma_i^{(k)}$'s) are always identical for the node pairs (resp., nodes) with identical (over-smoothed) hidden features, the attention function is vulnerable to over-smoothing. [7] For simplicity, we use $f_{att} = f_{att}(\theta)$ to denote $(\check{\mathcal{A}}, \Gamma)$, where θ denotes GNN parameters.

Definition 1. Given $G = (V, E)$ and initial node features X satisfying Assumption 1, suppose that $H_i^{(k')} = H_{i'}^{(k')}$ and $H_j^{(k')} = H_{j'}^{(k')}$ holds (due to the over-smoothing of node features) for some $(i, j), (i', j') \in E$ and $k' \geq 1$. We say that f_{att} is **vulnerable to over-smoothing** (V2OS), if $\forall \theta$, $\check{\alpha}_{ij}^{(k'+1)} = \check{\alpha}_{i'j'}^{(k'+1)}$ and $\gamma_i^{(k')} = \gamma_{i'}^{(k')}$; that f_{att} is **weakly resistant to over-smoothing** (WR2OS), if $\exists \theta$, $\check{\alpha}_{ij}^{(k'+1)} \neq \check{\alpha}_{i'j'}^{(k'+1)}$ or $\gamma_i^{(k')} \neq \gamma_{i'}^{(k')}$; and that f_{att} is **strongly resistant to over-smoothing** (SR2OS), if $\exists \theta$, $\check{\alpha}_{ij}^{(k'+1)} \neq \check{\alpha}_{i'j'}^{(k'+1)}$ and $\gamma_i^{(k')} \neq \gamma_{i'}^{(k')}$.

Remark 1. Definition 1 is practically significant. A GNN with WR2OS/SR2OS f_{att} can possibly remain expressive, even when some node features in intermediate layers begin to over-smooth. An expressive f_{att} can mitigate widely known problems of deep GNNs, including over-smoothing and over-squashing.

Theorem 1. *For GATv2, GPRGNN, and DAGNN, f_{att} is V2OS; for FAGCN, f_{att} is WR2OS (but not SR2OS).*

Proof. All the proofs are in Appendix A. □

Problem 2: Smooth Cumulative Attention. If Problem 1 does not exist, such that the attention coefficients are non-

---

[6]We see attention matrices as functions here.

[7]We use such a definition using the *equality* of node features for simplicity. See Appendix A.4 for a relaxed version of Definition 1.

---

trivial for any model depth, can the attention functions remain expressive over deeper layers? We argue not. For the representative attention-based GNNs, we show that the cumulative attention matrices $T^{(k)}$'s become over-smoothed over increasing layers, such that different nodes have attention close to each other, up to a positive scaling factor. *This is critically contrary to the goal of attention.* Formally, we define a smoothness score $S : \mathbb{R}^{n \times n} \to \mathbb{R}_{\geq 0}$ by

$$S(T) = \sum_{(i,j) \in \binom{[n]}{2}} \left\| \frac{T_i^{(k)}}{\|T_i^{(k)}\|_1} - \frac{T_j^{(k)}}{\|T_j^{(k)}\|_1} \right\|_1 / \binom{n}{2}.^8 \quad (4)$$

$S(\cdot)$ is bounded, and a smaller $S(T^{(k)})$ indicates that $T^{(k)}$ is smoother. Specifically, when $S(T^{(k)}) = 0$, all the rows of $T^{(k)}$ become equivalent up to a positive scaling factor. See Appendix A for more theoretical properties of $S(\cdot)$.

Theorem 2. *Given $G = (V, E)$ and X with Assumptions 1 and 2 satisfied, for GATv2, GPRGNN, and DAGNN, $\lim_{k \to \infty} S(T^{(k)}) = 0$; for FAGCN, $\lim_{k \to \infty} T_{ij}^{(k)} = 0, \forall i, j$.*

By Theorem 2, for the aforementioned attention-based GNNs, $S(T^{(k)})$ converges to 0 and loses expressiveness as k goes to infinity. Importantly, in our proofs, we show that problem 2 occurs *without assuming problem 1*.

Below, we provide some insights into the reason why the GNNs without *node-adaptive hop attention* or *negative attention* may suffer from the problem stated above.

Definition 2. Given $G = (V, E)$, and $i, j \in V$, the set of **attention path** between i and j at layer k, denoted as $\mathcal{P}^{(k)}(i, j)$, is defined as $\{(i = v_0, v_1, v_2, \ldots, v_{k-1}, v_k = j) : (v_{l-1}, v_l) \in E, \forall 1 \leq l \leq k\}$, i.e., the set of all the paths of length k from i to j in G.[9] The **degree of intersection** between two paths $\mathbf{p}_v = (v_0, v_1, v_2, \ldots, v_{k-1}, v_k)$ and $\mathbf{p}_u = (u_0, u_1, u_2, \ldots, u_{k-1}, u_k)$, denoted by $\mathrm{doi}(\mathbf{v}, \mathbf{u})$, is defined as $|\{t : 1 \leq t \leq k, v_{t-1} = u_{t-1}, v_t = u_t\}|$.

Remark 2. The intuition of Definition 2 is that we can decompose each entry $T_{ij}^{(k)}$ with respect to attention paths from j to i. Specifically, $T_{ij}^{(k)} = \gamma_i^{(k)} \sum_{(j=v_0,v_1,\ldots,v_k=i) \in \mathcal{P}^{(k)}(j,i)} \prod_{\ell=1}^{k} \alpha_{v_{\ell-1},v_\ell}^{(\ell)}, \forall i, j, k$.

Lemma 1. *Given $G = (V, E)$ and initial node features satisfying Assumption 1, $i, j, x \in V$, and any $N_1, N_2 > 0$, there exists K such that $|\{(\mathbf{p}_i, \mathbf{p}_j) : \mathbf{p}_i \in \mathcal{P}^{(k)}(i, x), \mathbf{p}_j \in \mathcal{P}^{(k)}(j, x), \mathrm{doi}(\mathbf{p}_i, \mathbf{p}_j) \geq N_1\}| \geq N_2, \forall k \geq K$.*

Proof. All the proofs are in Appendix A. □

By Lemma 1 and Remark 2, the constituent terms of T_{ix} and T_{jx} increasingly intersect at deeper layers for any i, j, x.

---

[8]If T_i is a zero vector, we let $T_i / \|T_i\|_1$ be a zero vector too. The definition is similar to the smoothness metric defined in Liu et al. (2020).

[9]Unlike *simple* paths, repeated nodes are allowed.

In other words, bounds of f_{att} and growing degree of intersection in $T^{(k)}$ together can cause problem 2, irrespective of f_{att}'s expressive power at each layer. This partially explains why some GNNs without node-adaptive hop attention or negative attention end up with a smooth cumulative attention $T^{(k)}$. More details can be found in the proof of Theorem 2 in Appendix A.

While we focused on GATv2, which is a representative variant of GAT, in our theoretical analysis, our analysis can be easily extended to other GAT variants, such as Super-GAT (Kim & Oh, 2021) with a self-supervised loss term, and CATs (He et al., 2021) further using structural features. See Appendix A for more discussions on those variants.

4. Proposed Method: AERO-GNN

In this section, we introduce the proposed model, **A**ttentive d**E**ep p**RO**pagation-GNN (AERO-GNN). We present the model overview and discuss its attention functions. Finally, we show how AERO-GNN provably addresses the theoretical limitations, with a discussion on model complexity (spec., number of parameters).

4.1. Model Overview

The feature transformation and propagation of AERO-GNN consist of:

$$H^{(k)} = \begin{cases} \text{MLP}(X), & \text{if } k = 0, \\ \mathcal{A}^{(k)} H^{(k-1)}, & \text{if } 1 \le k \le k_{max}, \end{cases} \quad (5)$$

$$Z^{(k)} = \sum_{\ell=0}^{k} \Gamma^{(\ell)} H^{(\ell)}, \forall 1 \le k \le k_{max}, \quad (6)$$

$$Z^* = \sigma(Z^{(k_{max})}) W^*, \quad (7)$$

where MLP is a multi-layer perceptron for the feature transformation, k_{max} is the total number of layers, W^* is a learnable weight matrix, Z^* is the final output node features, and $\sigma = \text{ELU}$ (Clevert et al., 2016) is the activation function.

AERO-GNN computes the edge attention matrix $\mathcal{A}^{(k)}$ and the hop attention matrix $\Gamma^{(k)}$ with learnable parameters. The propagation of AERO-GNN can also be written in terms of the cumulative attention matrices $T^{(k)}$'s in Eq. (3), like the other attention-based GNNs (see Table 2). Specifically,

$$Z^{(k)} = \sum_{\ell=0}^{k} T^{(\ell)} H^{(0)}, \forall 1 \le k \le k_{max}. \quad (8)$$

4.2. Using Layer-Aggregated Features

We design AERO-GNN to use the layer-aggregated features $Z^{(k)}$ in computing both the edge attention $\mathcal{A}^{(k)}$ and the hop attention $\Gamma^{(k)}$ at each layer k to make it resistant to over-smoothing. In Theorem 1, FAGCN is the only model that is WR2OS (*weakly resistant to over-smoothing*), since it uses $Z^{(k)}$'s for computing its edge attention (see Table 1).

Even if the node features become over-smoothed at deep layers, a GNN using layer-aggregated features $Z^{(k)}$ can adjust the edge (and hop) attention coefficients based on the cumulative information over multiple layers to allow attention functions to be non-trivial.

However, the magnitude of $Z^{(k)}$ may increase as k increases, as shown in Eq. (6), which may cause instability. We, thus, utilize weight-decay (Chen et al., 2020b) to re-scale Z. Formally, the re-scaling is done by

$$\begin{cases} \lambda_k & = \log(\frac{\lambda}{k} + 1 + \epsilon) \\ \tilde{Z}^{(k)} & = \lambda_k Z^{(k)} \end{cases}$$

where $\lambda > 0$ is a hyperparameter, and $\epsilon > 0$ is a small number (we use $\epsilon = 10^{-6}$) ensuring that λ_k does not converge to 0. We use $\tilde{Z}^{(k)}$ in the computation of both attention functions.

4.3. Attention Functions

Edge Attention. At every layer $1 \le k \le k_{max}$, we compute the pre-normalized edge attention $\check{\mathcal{A}}^{(k)} = (\check{\alpha}_{ij}^{(k)})$ and the (normalized) edge attention $\mathcal{A}^{(k)} = (\alpha_{ij}^{(k)})$ as follows:

$$\begin{cases} \check{\alpha}_{ij}^{(k)} & = \text{softplus}((W_{edge}^{(k)})^\top \sigma(\tilde{Z}_i^{(k-1)} \| \tilde{Z}_j^{(k-1)})) \\ \alpha_{ij}^{(k)} & = \check{\alpha}_{ij}^{(k)} / \sqrt{\sum_{j' \in N(i)} \check{\alpha}_{ij'}^{(k)} \sum_{i' \in N(j)} \check{\alpha}_{ji'}^{(k)}} \end{cases}$$

where $W_{edge}^{(k)}$ is a learnable weight vector, $\tilde{Z}_i^{(k-1)}$ is the i-th row of $\tilde{Z}^{(k-1)}$, and $\mathcal{A}^{(k)}$ is symmetrically normalized.

In AERO-GNN, we use the symmetric normalization, instead of the row-wise normalization used in GATv2, due to its theoretical and empirical superiority (Wang et al., 2018; 2021b; He et al., 2020), especially w.r.t. training stability. Softplus (Zheng et al., 2015) is used to positively map edge attention, with two primary advantages over two other mapping functions, exp and tanh. Compared to exp used in GATv2, softplus has the higher computational stability (Kleshchevnikov, 2020; Nbro, 2020). Note that the bound of tanh, in addition to degree normalization, essentially makes $\lim_{k \to \infty} T_{ij}^{(k)} = 0, \forall i, j$, for FAGCN (see Theorem 2).

Hop Attention. At each layer $0 \le k \le k_{max}$, we use $H^{(k)}$ and $\tilde{Z}^{(k-1)}$ (for $k \ge 1$) to compute the hop attention $\Gamma^{(k)}$:

$$\begin{cases} \gamma_i^{(0)} & = (W_{hop}^{(0)})^\top \sigma(H_i^{(0)}) + b_{hop}^{(0)} \\ \gamma_i^{(k)} & = (W_{hop}^{(k)})^\top \sigma(H_i^{(k)} \| \tilde{Z}_i^{(k-1)}) + b_{hop}^{(k)}, \forall 1 \le k \le k_{max} \end{cases}$$

where $W_{hop}^{(k)}$ is a learnable weight vector and b_{hop} is a learnable bias scalar.

Motivated by Theorem 2, we use node-adaptive hop attention to alleviate the problem of "intersecting attention paths" stated in Lemma 1, and we allow both positive and negative hop attention coefficients to prevent $S(T^{(k)})$ from converging to zero as k goes to infinity (note that $\gamma_i^{(k)}$'s of the same sign within the same layer k cannot change $S(T^{(k)})$).

Table 3: Node Classification Performance on Real-World Graphs

Dataset	Chameleon	Squirrel	Actor	Texas	Cornell	Wisconsin	Computer	Photo	Wiki-CS	Pubmed	Citeseer	Cora	A.R.
Homophily	0.04	0.03	0.01	0.00	0.02	0.05	0.70	0.77	0.57	0.66	0.63	0.77	
GCN	67.97 ± 2.5	53.33 ± 1.3	30.57 ± 0.7	65.65 ± 4.8	58.41 ± 3.3	62.02 ± 5.9	81.27 ± 1.4	90.24 ± 1.3	79.08 ± 0.5	79.54 ± 0.4	72.50 ± 0.5	83.15 ± 0.5	9.1
APPNP	53.04 ± 2.2	40.37 ± 1.5	35.49 ± 1.0	79.89 ± 4.2	80.16 ± 5.9	84.24 ± 4.6	81.27 ± 1.4	91.12 ± 1.2	79.05 ± 0.5	79.90 ± 0.3	73.06 ± 0.3	83.60 ± 1.3	7.8
GCN-II	60.38 ± 1.9	48.76 ± 2.4	35.77 ± 1.0	78.59 ± 6.6	78.84 ± 6.6	83.20 ± 4.7	84.24 ± 1.2	91.81 ± 0.9	79.28 ± 0.6	80.14 ± 0.6	73.20 ± 0.8	85.33 ± 0.5	5.5
A-DGN	69.63 ± 2.0	57.77 ± 1.9	36.41 ± 1.0	82.22 ± 4.8	83.14 ± 6.7	85.84 ± 4.0	83.70 ± 1.5	90.53 ± 1.3	79.11 ± 0.6	78.68 ± 0.6	70.16 ± 0.9	79.84 ± 0.9	6.4
GAT	68.01 ± 2.5	54.49 ± 1.7	30.36 ± 0.9	60.46 ± 6.2	58.22 ± 3.7	63.59 ± 6.1	84.46 ± 1.3	89.88 ± 1.1	79.44 ± 0.5	78.94 ± 0.4	71.89 ± 0.6	83.78 ± 0.5	8.5
GATv2	69.06 ± 2.2	57.67 ± 2.4	30.27 ± 0.8	60.32 ± 7.0	58.35 ± 3.8	61.94 ± 4.7	84.19 ± 1.2	89.87 ± 1.2	79.64 ± 0.5	79.12 ± 0.3	71.15 ± 1.2	83.88 ± 0.6	8.9
GATv2^R	70.88 ± 1.9	61.23 ± 1.5	33.73 ± 0.9	60.68 ± 6.6	57.32 ± 4.5	60.61 ± 5.1	81.73 ± 2.2	88.71 ± 1.7	79.75 ± 0.6	78.28 ± 0.4	71.00 ± 0.8	82.42 ± 0.6	9.3
GT	69.34 ± 1.2	55.04 ± 1.9	36.29 ± 1.0	84.08 ± 5.6	80.00 ± 4.9	84.80 ± 4.3	84.38 ± 1.3	91.28 ± 1.1	79.93 ± 0.5	79.04 ± 0.5	70.16 ± 0.8	82.09 ± 0.7	5.6
FAGCN	60.98 ± 2.3	42.20 ± 1.8	35.67 ± 0.9	77.00 ± 7.7	78.32 ± 6.3	82.41 ± 3.8	82.79 ± 2.7	91.99 ± 1.0	79.27 ± 0.6	79.19 ± 0.4	71.55 ± 0.8	83.88 ± 0.5	7.5
DMP	63.79 ± 4.1	34.19 ± 7.6	28.30 ± 2.7	66.08 ± 7.0	56.41 ± 5.5	62.73 ± 4.5	70.58 ± 11.3	82.63 ± 4.1	56.41 ± 7.8	70.07 ± 4.1	59.12 ± 4.4	75.05 ± 3.8	12.8
MixHop	60.30 ± 2.1	41.05 ± 2.0	36.48 ± 1.2	77.73 ± 7.3	75.95 ± 5.7	82.12 ± 4.5	79.52 ± 2.1	89.45 ± 1.5	78.59 ± 0.7	80.10 ± 0.4	71.42 ± 0.9	81.61 ± 0.8	9.3
GPRGNN	66.92 ± 1.7	46.32 ± 1.5	35.58 ± 0.9	81.51 ± 6.6	76.86 ± 7.1	84.06 ± 5.2	85.82 ± 0.9	92.41 ± 0.7	79.67 ± 0.5	80.28 ± 0.4	71.59 ± 0.8	84.20 ± 0.5	5.2
DAGNN	54.99 ± 2.0	40.03 ± 1.4	33.69 ± 1.0	61.35 ± 6.1	63.89 ± 7.0	62.27 ± 4.2	85.83 ± 0.8	92.30 ± 0.7	79.31 ± 0.6	80.44 ± 0.5	73.16 ± 0.6	84.43 ± 0.5	7.2
AERO-GNN	71.58 ± 2.4	61.76 ± 2.4	36.57 ± 1.1	84.35 ± 5.2	81.24 ± 6.8	84.80 ± 3.3	86.69 ± 1.4	92.50 ± 0.7	79.95 ± 0.5	80.59 ± 0.5	73.20 ± 0.6	83.90 ± 0.5	1.4

• In each column, ■ indicates ranking the first, and ■ indicates ranking the second. A.R. denotes average ranking.

Importantly, $b_{hop}^{(k)}$ should be initialized as 1s. This contributes significantly to model training stability by biasing the hop attention $\gamma_i^{(k)}$'s to be initialized positive.

4.4. Theoretical Merits

We summarize below the theoretical merits of AERO-GNN. First, AERO-GNN provably mitigates the problems of deep graph attention stated in Section 3 (Theorems 1 and 2).

Theorem 3. *For AERO-GNN, f_{att} is SR2OS (see Def. 1).*

Theorem 4. *Given $G = (V, E)$ and X with Assumptions 1 and 2 satisfied, for AERO-GNN, $\forall T^{(k)}$ (spec., even if $S(T^{(k)}) = 0$), $\exists \theta$ such that $S(T^{(k+1)}) > 0$.*[10]

The number of parameters used by AERO-GNN is comparable to, or even smaller than, those of the edge-attention GNNs. We ignore the parameters used in computing the first hidden features ($H^{(0)}$) and those in the output layer, since they are used in all GNN models. Thus, we only consider the number of *additional parameters*. Analysis for more models can be found in Appendix E.

Theorem 5. *Given the dimension d_H of hidden node features and the number of layers k_{max}, for AERO-GNN and FAGCN, the number of additional parameters is $\Theta(k_{max}d_H)$; for GATv2, the number is $\Theta(k_{max}d_H^2)$.*

Proof. All the proofs are in Appendix A. □

5. Experiments

In this section, we conduct experiments to demonstrate the empirical strengths of AERO-GNN and elaborate on the theoretical analyses.

[10]Recall that θ represents all the parameters in the GNN model.

5.1. Experimental Settings

Datasets. We use 12 node classification benchmark datasets, among which 6 are homophilic and 6 are heterophilic (McPherson et al., 2001; Pei et al., 2020; Lim et al., 2021a). In all the experiments, we use the publicly available train-validation-test splits, unless otherwise specified. We use *sparse-labeled* training for homophilic graphs and *dense-labeled* training for heterophilic graphs (small and large proportions of train labels, respectively; refer to Appendix C for details).

Baseline Methods. The baseline methods consist of various representative attention-based GNNs, including both *edge-attention GNNs* (GAT, GATv2, GATv2^R, GT (Shi et al., 2021), FAGCN, DMP (Yang et al., 2021))[11] and *hop-attention GNNs* (GPRGNN, DAGNN, MixHop (Abu-El-Haija et al., 2019)). In addition to some *simple GNNs* (GCN (Welling & Kipf, 2016), APPNP (Gasteiger et al., 2018)), *deep GNNs* without attention also serve as baselines (GCN-II (Chen et al., 2020b), A-DGN (Gravina et al., 2023)). See Appendix E and F for their details.

Experiment Details. The Adam optimizer (Kingma & Ba, 2015) is used to train the models, and the best parameters are selected based on early stopping. In measuring model performance (Section 5.2), we use 100 predetermined random seeds and report the mean ± standard deviation (SD) of classification accuracy over 100 trials. When analyzing attention coefficient distribution (Section 5.3), attention coefficients are averaged over 10 trials.

5.2. Node-Classification Performance

We evaluate the performance of each model on 12 real-world node classification benchmarks.

[11]GATv2^R is a GATv2 model with initial residual connection.
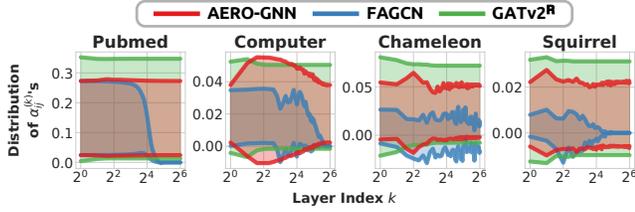
(a) The distribution of $\alpha_{ij}^{(k)}$'s for each k. The shaded area represents mean ± 1 SD.
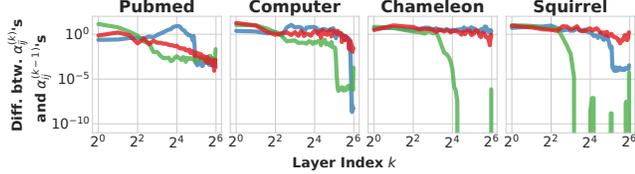
(b) The differences between $\alpha_{ij}^{(k)}$'s and $\alpha_{ij}^{(k-1)}$'s for each k. Formally, $\sqrt{\sum_{i,j}(\alpha_{ij}^{(k)} - \alpha_{ij}^{(k-1)})^2}$ is reported for each k.

Figure 1: **Statistics of $\alpha_{ij}^{(k)}$'s for each k, with $k_{max} = 64$.** *Only AERO-GNN learns edge-, hop-, and graph-adaptive edge attention over deep layers.*

On Heterophilic Graphs. As described in Section 5.1, dense-labeled training is used on heterophilic graphs, following prior research. On heterophilic datasets, AERO-GNN obtains significant and consistent performance gains compared to the baseline models. Even in relatively small graphs (e.g., *texas*, *cornell*, *wisconsin*), where three propagation layers are enough to reach most nodes, AERO-GNN outperforms all the baseline attention-based GNNs. In other words, even when the relative advantage of a deeper model is small, AERO-GNN still maintains competitive performance. Despite being specifically designed to address the heterophily problem, FAGCN, DMP, and GPRGNN are outperformed by AERO-GNN.

On Homophilic Graphs. In line with the prior research, sparse-labeled training is used for homophilic graphs. This poses a distinct set of challenges from dense-labeled training in heterophilic graphs, especially for attention-based GNNs. That is because, in homophilic graphs, the relative importance of each neighbor may not vary as significantly as it does in heterophilic graphs. Additionally, with a small number of train labels, deep and complex models may prone to overfitting. As a result, only a few models achieve strong performance in both settings (GCN-II, GT, and GPRGNN perform the best among the baselines). Despite such difficulties, AERO-GNN demonstrates strong performance, ranking first in 5 out of the 6 homophilic datasets.

For further evaluation, we discuss AERO-GNN's performance with multi-head attention and with datasets proposed by Platonov et al. (2023) in Appendix G, where AERO-GNN still has the best average ranking among the competitors.
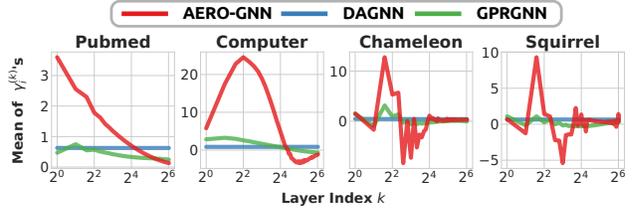
(a) The mean value of $\gamma_i^{(k)}$'s for each k.
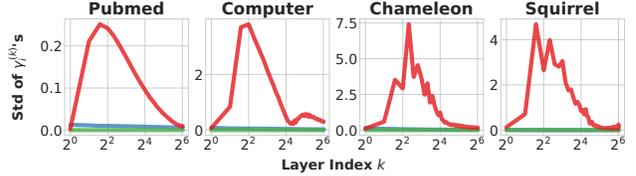
(b) The SD of $\gamma_i^{(k)}$'s for each k.

Figure 2: **Statistics of $\gamma_i^{(k)}$'s for each k, with $k_{max} = 64$.** *Only AERO-GNN learns node-, hop-, and graph-adaptive hop attention over deep layers.*

5.3. Empirical Elaboration of the Theoretical Analysis

In this section, we empirically elaborate on the theoretical limitations of the representative attention-based GNNs (Theorems 1, 2) and show that AERO-GNN effectively mitigates the problems (Theorems 3, 4). Specifically, compared to the baselines, we show that AERO-GNN has

- *edge/node-, hop-, and graph*-adaptive attention function,
- *less smooth* and *un-smoothing* cumulative attention $T^{(k)}$,
- and *higher* performance at deep layers.

Here, we bring our focus back to GATv2, FAGCN, GPRGNN, DAGNN, and AERO-GNN. [12]

Statistics of the Attention Coefficients. According to Theorems 1 and 3, only AERO-GNN would have both attention functions resistant to the over-smoothed node features. While FAGCN would only have a resistant edge-attention function, the attention coefficient distributions of the other models are expected to shrink or remain stationary with the increasing number of layers (when the over-smoothing of node features is more likely to occur). To test the hypothesis, we train 64 layers of each model and conduct post-hoc analysis of their learned attention distributions.

First, we study the edge-attention coefficients $\alpha_{ij}^{(k)}$'s. For each k, Figure 1 presents (**a**) the distribution of $\alpha_{ij}^{(k)}$'s and (**b**) the Frobenius norm of the difference between the attention coefficient at layer k and $(k-1)$, i.e., $\left\| (\alpha_{ij}^{(k)} - \alpha_{ij}^{(k-1)})_{ij} \right\|_F = \sqrt{\sum_{i,j}(\alpha_{ij}^{(k)} - \alpha_{ij}^{(k-1)})^2}$. [13] If the

---

[12] Training of vanilla GATv2 becomes unstable over deeper layers. Thus, for a fair comparison, we use GATv2^R instead in the following sections.

[13] The layer index k (the x-axis) is reversed for GATv2^R and FAGCN. Refer to Appendix G.4. for rationales.

distributions shrink or remain stationary over k for all graphs, we have strong evidence that attention coefficients are not working properly at deep layers.

For FAGCN, all the coefficients approach zero, and SDs of GATv2's attention coefficients remain stationary for all graphs (see Figure 1(a)). Both FAGCN and GATv2's attention coefficients remain stationary at deep layers, with near 0 difference between $\alpha_{ij}^{(k)}$ and $\alpha_{ij}^{(k-1)}$ at large k (see Figure 1(b)).

Only AERO-GNN's edge attention distributions do not shrink or remain stationary over k. To elaborate, AERO-GNN learns edge attention coefficients that are *edge-adaptive* (high variances in Figure 1(a)), *hop-adaptive* (high differences in Figure 1(b)), and *graph-adaptive* (diverse patterns for different graphs in Figure 1(a)). The results for all 12 datasets are in Appendix G.

We now investigate hop-attention coefficients $\gamma_i^{(k)}$'s. Figure 2(a) shows the mean value and (b) the SD of $\gamma_i^{(k)}$'s, for each layer k. Each figure respectively suggests how hop-adaptive and node-adaptive $\gamma_i^{(k)}$'s are.

Again, as expected, hop attention distributions of DAGNN remain stationary over deep layers, and hence, they are less adaptive to node, hop, or graph (see stationary values in Figure 2(a) and the very small SD values in Figure 2(b)). Since GPRGNN's hop attention $\gamma_i^{(k)}$'s are free parameters, it learns hop-adaptive and graph-adaptive $\gamma_i^{(k)}$'s regardless of over-smoothing. However, they are not node-adaptive.

In stark contrast, hop attention coefficients of AERO-GNN are *node-adaptive* (high SD in Figure 2(b)), *hop-adaptive* (mean value changes over different layers in Figure 2(a)), and *graph-adaptive* (diverse patterns for different graphs in Figure 2(a) and (b)). The results for all 12 datasets are in Appendix G.

Through this series of empirical analyses, we present strong evidence that attention functions of the representative attention-based GNNs, except for those of AERO-GNN, are vulnerable to node feature over-smoothing and fail to remain expressive over deep layers.

Cumulative Attention and Model Performance. According to Theorems 2 and 4, only AERO-GNN can avoid completely-smoothed cumulative attention (i.e. $S(T^{(k)}) = 0$) over deep layers, highlighting its capacity to learn meaningful attention at any model depth. Here, we empirically elaborate on the theoretical analysis.

Figure 3(a) shows the smoothness scores of cumulative attention matrices $S(T^{(k)})$. [14] AERO-GNN generally has less smooth $T^{(k)}$ over deep layers. Notably, there often oc-

---
¹⁴ Recall that a lower $S(T^{(k)})$ indicates that the cumulative attention $T^{(k)}$ is more smoothed
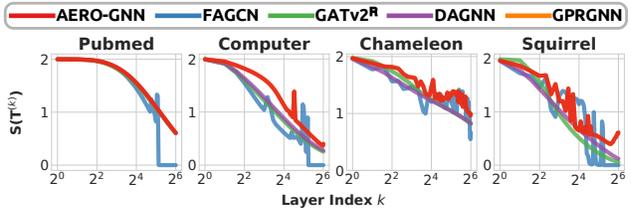
(a) The smoothness score $S(T^{(k)})$ of the cumulative attention matrix $T^{(k)}$ for each k.
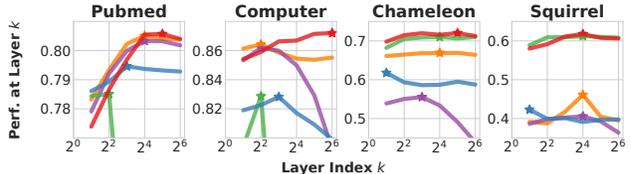
(b) The model performance at each layer $k \in \{2, 4, 8, 16, 32, 64\}$. A star ($\star$) indicates best performance of each model.

Figure 3: **The smoothness of $T^{(k)}$ and the model performance.** *AERO-GNN has (**a**) less smoothed $T^{(k)}$ over deep layers and (**b**) higher best performance achieved at a deeper layer.*

Table 4: Best Performing Layers Used in Table 3

Dataset	CM	SQ	AT	TX	CN	WC	CP	PT	WK	PM	CS	CR
GCN-II	8	4	4	4	4	4	4	**32**	4	16	**32**	**64**
A-DGN	16	**32**	**32**	2	**32**	2	8	8	16	8	16	4
GATv2^R	16	16	16	**8**	2	2	4	4	2	4	2	2
FAGCN	2	2	6	7	2	7	7	8	3	8	4	4
GPRGNN	16	16	4	4	4	**8**	4	4	**32**	16	8	8
DAGNN	5	20	5	5	5	5	5	10	5	20	10	10
AERO-GNN	**32**	16	4	**8**	4	4	**32**	**32**	16	**32**	**32**	32

curs un-smoothing of $T^{(k)}$, such that $S(T^{(k)}) > S(T^{(k-1)})$, only for AERO-GNN and FAGCN. [15] This phenomenon is attributable to the use of negative attention in both models. However, $S(T^{(k)})$ of FAGCN quickly converges to 0. These findings resonate with Theorem 4, showing that AERO-GNN's attention function does remain expressive at deep layers. The results for all 12 datasets are in Appendix G.

Figure 3(b) illustrates the model performance at layer $k \in \{2, 4, 8, 16, 32, 64\}$. AERO-GNN generally achieves better performance over deeper layers. Meanwhile, the performance of the representative attention-based GNNs often drops, even significantly, at deep layers. Table 4 further shows that AERO-GNN generally achieves its best performances at deeper layers than the attention-based GNNs.

In this section, we empirically evaluated the adaptiveness of attention coefficients, smoothness of $T^{(k)}$, and performance with depth for each model. This set of empirical observa-

---
¹⁵ If $S(T^{(k)}) \leq S(T^{(k-1)})$ always holds, then $S(T^{(k)})$ is convergent. See Corollary 1 in Appendix A.

tions, in concert with the theoretical findings, indicate that AERO-GNN indeed learns the most expressive deep graph attention.

6. Discussion

Graph attention has had a significant impact on the research and applications of GNNs. Many attention-based GNNs have been designed, investigating how to better infer relations between node pairs. Some models introduced more features or loss terms (Kim & Oh, 2021; Wang et al., 2019; He et al., 2021), and some others relaxed their attentions to include negative values (Bo et al., 2021; Chien et al., 2021; Yang et al., 2021).

On the other hand, making GNNs deeper to enhance their expressivity has been an important setback to GNN research. A number of problems have been proposed to limit its expressiveness to increase over depth. Not to mention oversmoothing, but also over-squashing and over-correlation have been pointed out as possible problems in building deep GNNs. As such, techniques that modulate propagation or aggregation functions (Chamberlain et al., 2021; Gravina et al., 2023; Li et al., 2020; Bodnar et al., 2022), residual connection functions (Li et al., 2019; Chen et al., 2020b; Li et al., 2021), hidden features (Zhou et al., 2020; Zhao & Akoglu, 2020; Guo et al., 2023), or graph topology (Rong et al., 2020; Chen et al., 2020a; Zeng et al., 2021; Topping et al., 2022; Bodnar et al., 2022) have been applied to build deeper GNNs.

In this work, we bridge the two research directions, addressing two underexplored questions: (**a**) what are the unique challenges in deep graph attention, and (**b**) how can we design provably more expressive deep graph attention? We argue that the representative attention-based GNNs suffer from the proposed set of problems, possibly on top of the general problems in deep GNNs. Thus, we design AERO-GNN to theoretically and empirically mitigate the problems.

Under a larger context, these findings extend prior literature on limitations to *deep attention in general*. Specifically, similar problems of deep attention smoothing have been reported for transformers (Vaswani et al., 2017; Dong et al., 2021b), in both natural language processing (Shi et al., 2022) and computer vision (Gong et al., 2021; Zhou et al., 2021) domains. We demonstrate that attention-based GNNs share related, yet distinct, problems and propose a novel solution. Hence, we expect this work will inspire future research on deep attention and graph learning in various directions.

The generalizability of the present work is limited in that linear propagation is assumed to define $T^{(k)}$ of GATs. Also, we deliberately suppress non-linearity in the *propagation layers* of AERO-GNN to focus on the ability to learn dynamic receptive fields, expressed in $T^{(k)}$. Still, it can be

a promising future work to add non-linear propagation to AERO-GNN to address other challenges and applications to more complex tasks.

Acknowledgements

References

Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., Ver Steeg, G., and Galstyan, A. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International Conference on Machine Learning*. PMLR, 2019.

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Baranwal, A., Fountoulakis, K., and Jagannath, A. Graph convolution for semi-supervised classification: Improved linear separability and out-of-distribution generalization. In *International Conference on Machine Learning*. PMLR, 2021.

Bo, D., Wang, X., Shi, C., and Shen, H. Beyond low-frequency information in graph convolutional networks. In *AAAI Conference on Artificial Intelligence*, 2021.

Bodnar, C., Di Giovanni, F., Chamberlain, B., Liò, P., and Bronstein, M. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. In *Advances in Neural Information Processing Systems*, 2022.

Brody, S., Alon, U., and Yahav, E. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2021.

Chamberlain, B., Rowbottom, J., Gorinova, M. I., Bronstein, M., Webb, S., and Rossi, E. Grand: Graph neural diffusion. In *International Conference on Machine Learning*. PMLR, 2021.

Chanpuriya, S. and Musco, C. N. Simplified graph convolution with heterophily. In *Advances in Neural Information Processing Systems*, 2022.

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., and Sun, X. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI Conference on Artificial Intelligence*, 2020a.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*. PMLR, 2020b.

Chen, Y., Xiong, W., and Li, F. Convergence of infinite products of stochastic matrices: A graphical decomposition criterion. *IEEE Transactions on Automatic Control*, 61(11):3599–3605, 2016.

Chien, E., Peng, J., Li, P., and Milenkovic, O. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations*, 2016.

Derrow-Pinion, A., She, J., Wong, D., Lange, O., Hester, T., Perez, L., Nunkesser, M., Lee, S., Guo, X., Wiltshire, B., et al. Eta prediction with graph neural networks in google maps. In *ACM CIKM International Conference on Information & Knowledge Management*, 2021.

Dong, H., Chen, J., Feng, F., He, X., Bi, S., Ding, Z., and Cui, P. On the equivalence of decoupled graph convolution network and label propagation. In *The Web Conference*, 2021a.

Dong, Y., Cordonnier, J.-B., and Loukas, A. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*. PMLR, 2021b.

Fey, M. and Lenssen, J. E. Fast graph representation learning with pytorch geometric. In *International Conference on Learning Representations*, 2019.

Gasteiger, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2018.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*. PMLR, 2017.

Gong, C., Wang, D., Li, M., Chandra, V., and Liu, Q. Vision transformers with patch diversification. *arXiv preprint arXiv:2104.12753*, 2021.

Gravina, A., Bacciu, D., and Gallicchio, C. Anti-symmetric dgn: A stable architecture for deep graph networks. In *International Conference on Learning Representations*, 2023.

Guo, X., Wang, Y., Du, T., and Wang, Y. Contranorm: A contrastive learning perspective on oversmoothing and beyond. In *International Conference on Learning Representations*, 2023.

He, T., Ong, Y. S., and Bai, L. Learning conjoint attentions for graph neural nets. In *Advances in Neural Information Processing Systems*, 2021.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., and Wang, M. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

Jin, W., Liu, X., Ma, Y., Aggarwal, C., and Tang, J. Feature overcorrelation in deep graph neural networks: A new perspective. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2022.

Kim, D. and Oh, A. H. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations*, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kleshchevnikov, V. Softplus transform as a more numerically stable way to enforce positive constraint, 2020. URL https://github.com/pyro-ppl/numpyro/issues/855.

Li, G., Müller, M., Thabet, A., and Ghanem, B. Deepgcns: Can gcns go as deep as cnns? In *IEEE/CVF International Conference on Computer Vision*, 2019.

Li, G., Xiong, C., Thabet, A., and Ghanem, B. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020.

Li, G., Müller, M., Ghanem, B., and Koltun, V. Training graph neural networks with 1000 layers. In *International Conference on Machine Learning*. PMLR, 2021.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI Conference on Artificial Intelligence*, 2018.

Lim, D., Hohne, F., Li, X., Huang, S. L., Gupta, V., Bhalerao, O., and Lim, S. N. Large scale learning on non-homophilous graphs: New benchmarks and strong

simple methods. In *Advances in Neural Information Processing Systems*, 2021a.

Lim, D., Li, X., Hohne, F., and Lim, S.-N. New benchmarks for learning on non-homophilous graphs. *arXiv preprint arXiv:2104.01404*, 2021b.

Liu, M., Gao, H., and Ji, S. Towards deeper graph neural networks. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

Ma, Y., Liu, X., Shah, N., and Tang, J. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*, 2021.

McPherson, M., Smith-Lovin, L., and Cook, J. M. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

Mernyei, P. and Cangea, C. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

Nbro. Is there any particular reason why you use softplus to make sure the scale is non-negative?, Jan 2020. URL https://github.com/tensorflow/probability/issues/751.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

Platonov, O., Kuznedelev, D., Diskin, M., Babenko, A., and Prokhorenkova, L. A critical look at the evaluation of gnns under heterophily: Are we really making progress? In *International Conference on Learning Representations*, 2023.

Qiu, J., Tang, J., Ma, H., Dong, Y., Wang, K., and Tang, J. Deepinf: Social influence prediction with deep learning. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

Rozemberczki, B., Allen, C., and Sarkar, R. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):1–22, 2021.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*. PMLR, 2020.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Shi, H., GAO, J., Xu, H., Liang, X., Li, Z., Kong, L., Lee, S. M., and Kwok, J. Revisiting over-smoothing in bert from the perspective of graph. In *International Conference on Learning Representations*, 2022.

Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., and Sun, Y. Masked label prediction: Unified message passing model for semi-supervised classification. In *International Joint Conference on Artificial Intelligence*, 2021.

Stokes, J. M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N. M., MacNair, C. R., French, S., Carfrae, L. A., Bloom-Ackermann, Z., et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4): 688–702, 2020.

Tang, J., Sun, J., Wang, C., and Yang, Z. Social influence analysis in large-scale networks. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2009.

Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong, X., and Bronstein, M. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Wang, G., Ying, R., Huang, J., and Leskovec, J. Improving graph attention networks with large margin-based constraints. *arXiv preprint arXiv:1910.11945*, 2019.

Wang, G., Ying, R., Huang, J., and Leskovec, J. Multi-hop attention graph neural network. In *International Joint Conference on Artificial Intelligence*, 2021a.

Wang, X. and Zhang, M. How powerful are spectral graph neural networks. In *International Conference on Machine Learning*. PMLR, 2022.

Wang, X., Cheng, M., Eaton, J., Hsieh, C.-J., and Wu, F. Attack graph convolutional networks by adding fake nodes. *arXiv preprint arXiv:1810.10751*, 2018.

Wang, Y., Jin, J., Zhang, W., Yu, Y., Zhang, Z., and Wipf, D. Bag of tricks for node classification with graph neural networks. *arXiv preprint arXiv:2103.13355*, 2021b.

Wei, R., Yin, H., Jia, J., Benson, A. R., and Li, P. Understanding non-linearity in graph neural networks from the bayesian-inference perspective. In *Advances in Neural Information Processing Systems*, 2022.

Welling, M. and Kipf, T. N. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *International Conference on Machine Learning*. PMLR, 2019.

Wu, S., Sun, F., Zhang, W., Xie, X., and Cui, B. Graph neural networks in recommender systems: A survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

Wu, X., Chen, Z., Wang, W., and Jadbabaie, A. A non-asymptotic analysis of oversmoothing in graph neural networks. In *International Conference on Learning Representations*, 2023.

Yang, L., Li, M., Liu, L., Wang, C., Cao, X., Guo, Y., et al. Diverse message passing for attribute with heterophily. In *Advances in Neural Information Processing Systems*, 2021.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*. PMLR, 2016.

Yoo, J., Lee, M.-C., Shekhar, S., and Faloutsos, C. Slendergnn: Accurate, robust, and interpretable gnn, and the reasons for its success. *arXiv preprint arXiv:2210.04081*, 2022.

Zeng, H., Zhang, M., Xia, Y., Srivastava, A., Malevich, A., Kannan, R., Prasanna, V., Jin, L., and Chen, R. Decoupling the depth and scope of graph neural networks. In *Advances in Neural Information Processing Systems*, 2021.

Zhang, W., Sheng, Z., Yin, Z., Jiang, Y., Xia, Y., Gao, J., Yang, Z., and Cui, B. Model degradation hinders deep graph neural networks. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2022.

Zhao, H., Ma, S., Zhang, D., Deng, Z.-H., and Wei, F. Are more layers beneficial to graph transformers? In *International Conference on Learning Representations*, 2023.

Zhao, L. and Akoglu, L. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations*, 2020.

Zheng, H., Yang, Z., Liu, W., Liang, J., and Li, Y. Improving deep neural networks using softplus units. In *IEEE International Joint Conference on Neural Networks*, 2015.

Zhou, D., Kang, B., Jin, X., Yang, L., Lian, X., Jiang, Z., Hou, Q., and Feng, J. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021.

Zhou, K., Huang, X., Li, Y., Zha, D., Chen, R., and Hu, X. Towards deeper graph neural networks with differentiable group normalization. In *Advances in Neural Information Processing Systems*, 2020.

A. Proofs and Additional Theoretical Results

In this section, we provide proofs of the theoretical claims in the main text and some additional theoretical results.

A.1. Regarding Problem 1

Proof of Theorem 1. We analyze $\tilde{\mathcal{A}}^{(k)}$'s and $\Gamma^{(k)}$'s for each considered GNN model.

GATv2. For GATv2, for all i, j, k,

$$\tilde{\alpha}_{ij}^{(k)} = \exp((W_{edge}^{(k)})^\top \sigma(H_i^{(k-1)} \| H_j^{(k-1)})),$$

and $\gamma_i^{(k)} \equiv 1$. It is easy to see that $\tilde{\alpha}_{ij}^{(k)}$ is totally determined by $H_i^{(k)}$ and $H_j^{(k)}$, and $\gamma_i^{(k+1)}$ is constant, for all i, j. Thus, if the conditions in Definition 1 hold, then for any fixed θ, $\tilde{\alpha}_{ij}^{(k)} = \tilde{\alpha}_{i'j'}^{(k)}$ and $\gamma_i^{(k+1)} = \gamma_{i'}^{(k+1)}$ Hence, for GATv2, f_{att} is V2OS.

GAT Variants. Several variants of GAT are worth mentioning. CATs (He et al., 2021) use structural features to compute the attention coefficients at each layer. This enables CATs to partially overcome the limitation of the original GAT, and it is easy to see that f_{att} is WR2OS but not SR2OS (since CATs only essentially use identical hop attention). SuperGAT (Kim & Oh, 2021), another GAT variant, further uses self-supervision for its attention function. However, the attention function of SuperGAT is still completely dependent on the node features in the current layer, and thus for SuperGAT, f_{att} is V2OS as GATv2.

FAGCN. For FAGCN, for all i, j, k,

$$\tilde{\alpha}_{ij}^{(k)} = \tanh((W_{edge}^{(k)})^\top (Z_i^{(k-1)} \| Z_j^{(k-1)})),$$

and $\gamma_i^{(k)} \equiv \gamma_0$, for some constant γ_0. If the conditions in Definition 1 hold, then for any fixed θ, Since $X_i \neq X_j, \forall i, j$, there exist $W^{(1)}$ (and $W^{(2)}$) such that $H_i^{(0)}, H_j^{(0)}, H_{i'}^{(0)}, H_{j'}^{(0)}$ are all distinct. $H^{(0)}$ is used to compute $Z^{(k)}$ for each k, and $Z^{(k)}$ is used to compute $\mathcal{A}^{(k+1)}$ for each k. Thus, even if the conditions in Definition 1 hold, there still exists θ such that $\tilde{\alpha}_{ij}^{(k)} \neq \tilde{\alpha}_{i'j'}^{(k)}$, but $\gamma_i^{(k+1)} = \gamma_{i'}^{(k+1)}$. Hence, for FAGCN, f_{att} is WR2OS.

GPRGNN. For GPRGNN, for all i, j, k, $\tilde{\alpha}_{ij}^{(k)} \equiv 1$, and $\gamma_i^{(k)} \equiv \gamma_0$, for some learnable scalar γ_0. Hence, it is easy to see that for GPRGNN, f_{att} is V2OS.

DAGNN. For DAGNN, for all i, j, k, $\tilde{\alpha}_{ij}^{(k)} \equiv 1$, and

$$\gamma_i^{(k)} = \text{sigmoid}((W_{hop}^{(k)})^\top H_i^{(k)}).$$

With similar reasoning for GATv2, it is easy to see that for DAGNN, f_{att} is also V2OS. $\square$

Proof of Theorem 3. We now prove how AERO-GNN mitigates problem 1.

AERO-GNN. For AERO-GNN, for all i, j and $k > 0$,

$$\tilde{\alpha}_{ij}^{(k)} = \text{softplus}((W_{edge}^{(k)})^\top \sigma(\tilde{Z}_i^{(k-1)} \| \tilde{Z}_j^{(k-1)})),$$

and

$$\gamma_i^{(k)} = (W_{hop}^{(k)})^\top \sigma(H_i^{(k)} \| \tilde{Z}_i^{(k-1)}) + b_{hop}^{(k)}.$$

Since $X_i \neq X_j, \forall i, j$, there exist $W^{(1)}$ (and $W^{(2)}$ when $W^{(2)}$ is also used) such that $H_i^{(0)}, H_j^{(0)}, H_{i'}^{(0)}, H_{j'}^{(0)}$ are all distinct. $H_i^{(0)}$ is used to compute $\gamma_i^{(0)}$ for each i. $\Gamma^{(0)}$ is used to compute $Z^{(k)}$ for each k, and $Z^{(k)}$ is used to compute $\Gamma^{(k+1)}$ for each k. Thus, even if the conditions in Definition 1 hold, there still exists θ such that $\tilde{\alpha}_{ij}^{(k)} \neq \tilde{\alpha}_{i'j'}^{(k)}$ and $\gamma_i^{(k+1)} \neq \gamma_{i'}^{(k+1)}$. $\square$

A.2. Regarding Problem 2

Lemma 2. $S(T)$ *is bounded,* $\forall T \in \mathbb{R}^{n \times n}$.

Proof. A lower bound 0 is straightforward. Regarding an upper bound, for any T,

$$
\begin{aligned}
S(T) &= \Big(\sum_{i,j \in \binom{[n]}{2}} \left\| \frac{T_i}{\|T_i\|_1} - \frac{T_j}{\|T_j\|_1} \right\|_1 \Big) / \binom{n}{2} \\
&\leq \Big(\sum_{i,j \in \binom{[n]}{2}} \left\| \frac{T_i}{\|T_i\|_1} \right\|_1 + \left\| \frac{T_j}{\|T_j\|_1} \right\|_1 \Big) / \binom{n}{2} \\
&\leq 2
\end{aligned}
$$

$\square$

Corollary 1. *For an infinite series* $\{T_k\}_{k=0}^\infty$, *if* $S(T_{k+1}) < S(T_k), \forall k$, *then* $S(T_k)$ *is convergent, i.e.,* $\lim_{k \to \infty} S(T_k)$ *exists and is finite.*

Proof. It is straightforward by the monotone convergence theorem. $\square$

Lemma 3. *For a matrix* T *with at least one non-zero entry,* $S(T) = 0$ *if and only if* $\text{rank}(T) \leq 1$.

Proof. It is easy to see that $T_i / \|T_i\|_1$ are all identical for all i, if $\text{rank}(T) \leq 1$; and $\exists i \neq j, T_i / \|T_i\|_1 \neq T_j / \|T_j\|_1$, if $\text{rank}(T) \geq 2$. $\square$

Remark 3. Lemmas 2 and 3 also hold when we use other norms in $S(\cdot)$.

Proof of Theorem 2. We analyze $S(T^{(k)})$ as $k \to \infty$ for each model.

GATv2. For GATv2, for each k, $\Gamma^{(k)} = I$ and thus $T^{(k)} = \prod_{\ell=1}^k \mathcal{A}^{(\ell)}$, where $\alpha_{ij}^{(\ell)} = \frac{\tilde{a}_{ij}^{(\ell)}}{\sum_{j' \in N(i)} \tilde{a}_{ij'}^{(\ell)}} \in (0, 1), \forall i, j$ (see Table 1). Specifically, $\sum_j \alpha_{ij}^{(\ell)} = 1, \forall i, \ell$, i.e., $\mathcal{A}^{(\ell)}$ is row-wise stochastic, for each ℓ. Since the product of two row-wise

stochastic matrices is again still row-wise stochastic, $T^{(\ell)}$ is also row-wise stochastic, for each ℓ. Fix any k, we have $T^{(k+1)} = T^{(k)}\mathcal{A}^{(k+1)}$, and

$$
\begin{aligned}
S(T^{(k+1)}) &= \sum_{i,j} \left\| \frac{T_i^{(k+1)}}{\left\| T_i^{(k+1)} \right\|_1} - \frac{T_j^{(k+1)}}{\left\| T_j^{(k+1)} \right\|_1} \right\|_1 \\
&= \sum_{i,j} \left\| T_i^{(k+1)} - T_j^{(k+1)} \right\|_1 \\
&= \sum_{i,j} \sum_x \left| T_{ix}^{(k+1)} - T_{jx}^{(k+1)} \right| \\
&= \sum_{i,j} \sum_x \left| \sum_{y \in N(x)} T_{iy}^{(k)} \mathcal{A}_{yx}^{(k+1)} - \sum_{y \in N(x)} T_{jy}^{(k)} \mathcal{A}_{yx}^{(k+1)} \right| \\
&= \sum_{i,j} \sum_x \left| \sum_{y \in N(x)} (T_{iy}^{(k)} - T_{jy}^{(k)}) \mathcal{A}_{yx}^{(k+1)} \right| \\
&\leq \sum_{i,j} \sum_x \sum_{y \in N(x)} \left| T_{iy}^{(k)} - T_{jy}^{(k)} \right| \left| \mathcal{A}_{yx}^{(k+1)} \right| \\
&= \sum_{i,j} \sum_x \sum_{y \in N(x)} \left| T_{iy}^{(k)} - T_{jy}^{(k)} \right| \mathcal{A}_{yx}^{(k+1)} \\
&= \sum_{i,j} \sum_y \left| T_{iy}^{(k)} - T_{jy}^{(k)} \right| \sum_{x \in N(y)} \mathcal{A}_{yx}^{(k+1)} \\
&= \sum_{i,j} \sum_y \left| T_{iy}^{(k)} - T_{jy}^{(k)} \right| = S(T^{(k)}),
\end{aligned}
$$

where the inequality holds as equality if and only if $T_i^{(k)} = T_j^{(k)}, \forall i, j$, which means $S(T^{(k)}) = 0$ and thus $S(T^{(k')})$ remains 0 for all $k' \geq k$. [16] Otherwise, the inequality holds as a strict inequality, i.e., $S(T^{(k+1)}) < S(T^{(k)}), \forall k$, which, by Corollary 1, implies that $S(T^{(k)})$ converges to a constant as k goes to infinity. In conclusion, either $S(T^{(k)}), \forall k \geq k'$ remains zero after some k', or it strictly decreases and converges and as k goes to infinity.

Moreover, by Theorem 3 in Chen et al. (2016), and Assumption 1 (specifically, G is connected and non-bipartite), if there exists $c'' > 0$ s.t. for all k, each nonzero entry of $\mathcal{A}^{(k)}$ is at least c'', then $T^{(k)} = \prod_{\ell=1}^{k} \mathcal{A}^{(\ell)}$ converges to a matrix with equal rows, which implies that $S(T^{(k)})$ converges to 0. Indeed, by Assumption 2, for all $i, j, k, \tilde{\alpha}_{ij}^{(k)} = \exp((W_{edge}^{(k)})^\top \sigma(H_i^{(k-1)} \| H_j^{(k-1)}))$, where $(W_{edge}^{(k)})^\top \sigma(H_i^{(k-1)} \| H_j^{(k-1)}) \in [-nc_{param}^2, nc_{param}^2]$, and thus each nonzero $\alpha_{ij}^{(k)}$ satisfies that

$$
\alpha_{ij}^{(k)} \geq \frac{\exp(-nc_{param}^2)}{\exp(-nc_{param}^2) + (n-1)\exp(nc_{param}^2)}.
$$

We complete the proof for GATv2 by letting

$$
c'' = \frac{\exp(-nc_{param}^2)}{\exp(-nc_{param}^2) + (n-1)\exp(nc_{param}^2)}.
$$

GAT Variants. The proof for GATv2 is mainly based on (**a**)

[16]Here, we ignore $(/\binom{n}{2})$ in computing $S(T^{(k)})$ for simplicity.

trivial hop attention function and (**b**) the row-wise stochasticity introduced by the $\mathrm{softmax}$-based normalization. Trivial hop attention and $\mathrm{softmax}$ normalization scheme are also used in other variants, including SuperGAT (Kim & Oh, 2021) and CATs (He et al., 2021). Therefore, for SuperGAT and CATs, the conclusion is the same as for GATv2, i.e., $\lim_{k \to \infty} S(T^{(k)}) = 0$.

FAGCN. For FAGCN, we have $\forall i, j, k, \alpha_{ij}^{(k)} = \tilde{\alpha}_{ij}^{(k)}/\sqrt{d_i d_j}$, where $\tilde{\alpha}_{ij}^{(k)} \in (-1, 1)$ due to the usage of $\tanh(\cdot)$ (see Table 1), and $\Gamma^{(k)}$ (γ_i^k) is a constant, and we let Γ (γ) denote it. Let $\|\cdot\|_2$ and $\|\cdot\|_F$ denote the spectral norm and the Frobenius norm, respectively. For any fixed k_0, $\left\| Z^{(k_0)} \right\|_2 \leq \left\| Z^{(k_0)} \right\|_F \leq n\sqrt{c_{param}}$, and thus

$$
\begin{aligned}
\tilde{\alpha}_{ij}^{(k_0+1)} &\leq \tanh((W_{edge}^{(k_0+1)})^\top (Z_i^{(k_0)} \| Z_j^{(k_0)})) \\
&\leq \tanh(2\sqrt{n} c_{param} \left\| Z^{(k_0)} \right\|_2) \\
&\leq \tanh(2\sqrt{n} c_{param} n c_{param}) \\
&= \tanh(2n^{1.5} c_{param}^2).
\end{aligned}
$$

Therefore, the spectral norm of $\mathcal{A}^{(k_0)}$ is smaller than that of $c_{sn}\tilde{A}$ (see the proof for GPRGNN and DAGNN), where $c_{sn} = \tanh(2n^{1.5} c_{param}^2)$. Specifically,

$$
\left\| \mathcal{A}^{(k)} \right\|_2 < \left\| c_{sn}\tilde{A} \right\|_2 \leq c_{sn} \left\| \tilde{A} \right\|_2 \leq c_{sn},
$$

which, by the submultiplicity of the spectral norm, gives $\left\| \prod_{\ell=1}^{k} \mathcal{A}^{(\ell)} \right\|_2 \leq c_{sn}^k$, and thus $\lim_{k \to \infty} \left\| \prod_{\ell=1}^{k} \mathcal{A}^{(\ell)} \right\|_2 = 0$. Since $\Gamma^{(k)}$ is bounded, $\lim_{k \to \infty} \left\| T^{(k)} \right\|_2 = 0$, by which we conclude that $\lim_{k \to \infty} T_{ij}^{(k)} = 0, \forall i, j$, completing the proof of FAGCN.

GPRGNN and DAGNN. For GPRGNN and DAGNN, $\mathcal{A}^{(k)} = \tilde{A}, \forall k$, where $\tilde{A} = (D+I)^{-1/2}(A+I)(D+I)^{-1/2}$, and thus

$$
\begin{aligned}
T^{(k)} &= \Gamma^{(k)} \prod_{\ell=k}^{1} \mathcal{A} \\
&= \Gamma^{(k)} \prod_{\ell=k}^{1} \tilde{A} \\
&= \Gamma^{(k)} \tilde{A}^k \\
&= \Gamma^{(k)} (D+I)^{-1/2}((A+I)(D+I)^{-1})^{-k+1}(A+I)(D+I)^{-1/2}.
\end{aligned}
$$

Since G is connected and non-bipartite, by the Ergodic theorem on Markov chains, $(D+I)^{-1/2}((A+I)(D+I)^{-1})^{-k+1}$ converges to a rank-one matrix (let P_T denote this rank-one matrix) as $k \to \infty$. Furthermore, since $(A+I)(D+I)^{-1/2}$ is bounded for any given G, formally, $\forall \epsilon < 0, \exists K > 0, k \geq K \Rightarrow \left\| \tilde{A}^k - P_T(A+I)(D+I)^{-1/2} \right\| \leq \epsilon$, where $\mathrm{rank}(P_T(A+I)(D+I)^{-1/2}) \leq \mathrm{rank}(P_T) = 1$. Since S is continuous, and $S(P_T(A+I)(D+I)^{-1/2}) = 0$ (see Lemma 3), $S(\tilde{A}^k)$ converges to 0 as $k \to \infty$. Since $\Gamma^{(k)}$ is diagonal and the signs of all entries are identical for both

DAGNN and GPRGNN, $S(\Gamma^{(k)}\tilde{A}^k) = S(\tilde{A}^k)$, completing the proof for GPRGNN and DAGNN.

$\square$

Proof of Theorem 4. We now prove how AERO-GNN mitigates problem 2.

AERO-GNN. For AERO-GNN, fix any k, consider first $S(\bar{T}^{(k)})$, where $\bar{T}^{(k)} = \prod_{\ell=1}^{k} \mathcal{A}^{(\ell)}$. If $S(\bar{T}^{(k)}) > 0$, then simply letting $W_{hop}^{(k)}$ be a zero vector and letting $b_{hop}^{(k)} = 1$ suffices, since in that case $\Gamma^{(k)} = I$, and $S(T^{(k)}) = S(\prod_{\ell=1}^{k} \mathcal{A}^{(\ell)}) > 0$. Otherwise, if $S(\prod_{\ell=1}^{k} \mathcal{A}^{(\ell)}) = 0$, i.e., $T_i/\|T_i\|_1 = T_j/\|T_j\|_1$, $\forall i, j$, by the formulae of Γ, we claim that $\exists \theta$ s.t. $\gamma_i^{(k)}\gamma_{i'}^{(k)} < 0$, for at least one pair i, j. This is true since as shown in the proof of Theorem 4, we can find a pair i, j such that $\sigma(H_i^{(k)}\|\tilde{Z}_i^{(k-1)}) \neq \sigma(H_j^{(k)}\|\tilde{Z}_j^{(k-1)})$, and thus we can find $W_{hop}^{(k)}$ and $b_{hop}^{(k)}$ such that $\gamma_i^{(k)}\gamma_{i'}^{(k)} < 0$. Specifically, WLOG, suppose that the t-th entry of $\sigma(H_i^{(k)}\|\tilde{Z}_i^{(k-1)})$ and $\sigma(H_j^{(k)}\|\tilde{Z}_j^{(k-1)})$ are different, then we can let $W_{hop}^{(k)}$ be the unit vector with $(W_{hop}^{(k)})_t = 1$ and all the other entries being zero, and let $b_{hop}^{(k)} = -((\sigma(H_i^{(k)}\|\tilde{Z}_i^{(k-1)}))_t + (\sigma(H_j^{(k)}\|\tilde{Z}_j^{(k-1)}))_t)/2$. Thus,

$$S(T^{(k)}) \geq \left\| \frac{\gamma_i^{(k)}T_i}{\|\gamma_i^{(k)}T_i\|_1} - \frac{\gamma_{i'}^{(k)}T_j}{\|\gamma_{i'}^{(k)}T_j\|_1} \right\| = 2\left\| \frac{T_i}{\|T_i\|_1} \right\| > 0,$$

completing the proof.[17]

$\square$

Proof of Lemma 1. Since G is non-bipartite, it contains a cycle of odd length. WLOG, suppose such a cycle **c** includes a node $y \in V$ (possibly, $y \in \{i, j, x\}$) and is of length $2s + 1$ for some $s \in \mathbb{N}$. Since G is connected, there exists a path from i to y, one from j to y, and one from y to x. Let d_{iy}, d_{jy}, and d_{yx} be the length of the shortest path from i to y, from j to y, and from y to x, respectively. WLOG, suppose $d_{iy} \leq d_{jy}$. Now, set $K = N_1 + 2sN_2 + d_{jy} + d_{yx}$, and fix any $k \geq K$, since a length-s cycle includes y, there exists $q \leq d_{jy} + s$ such that there exist length-q walks $\mathbf{w}_{iy;q}$ and $\mathbf{w}_{jy;q}$ from i to y and from j to y, respectively, and $q + d_{yx} \equiv k \pmod 2$ (an additional cycle from y back to y via **c** can be added to ensure this). Now, fix any neighbor z of x, for each $0 \leq r \leq N_2 - 1$, we can construct a pair of walks: it goes first from i (resp., j) to y via $\mathbf{w}_{iy;q}$ (resp., $\mathbf{w}_{jy;q}$), and then goes via **c** for $2r$ rounds ($2rs$ steps) back to y, and then goes to x in d_{yx} via the shortest path, and finally goes back and forth between x and z until the length of the walk reaches k. The degree of intersection between such a pair of walks is at least $k - q \geq K - d_{jy} - s \geq (2s-1)N_2 + d_{yx} + N_1 \geq N_1$, and

---

[17]Moreover, since we use edge attention (compared to GPRGNN and DAGNN), and we have control over the magnitude of $\Gamma^{(k)}$ (compared to all the others), we can make sure that $T^{(k)}$ does not converge to an all-zero matrix.

the number of such pairs is at least the number of r values, which is N_2, completing the proof. $\square$

A.3. Regarding the Number of Parameters

Proof of Theorem 5. For AERO-GNN, the additional parameters include $W_{edge}^{(k)}$'s, $W_{hop}^{(k)}$'s, and $b_{hop}^{(k)}$'s, where for each k, $W_{edge}^{(k)} \in \mathbb{R}^{2d_H}$, $W_{hop}^{(k)} \in \mathbb{R}^{2d_H}$, and $b_{hop}^{(k)} \in \mathbb{R}$. Therefore, the number of additional parameters is $k_{max}(4d_H + 1) = \Theta(k_{max}d_H)$.

For FAGCN, the additional parameters include $W_{edge}^{(k)}$'s, where for each k, $W_{edge}^{(k)} \in \mathbb{R}^{2d_H}$. Therefore, the number of additional parameters is $k_{max}(2d_H) = \Theta(k_{max}d_H)$.

For GATv2, the additional parameters include $W^{(k)}$'s and $W_{edge}^{(k)}$'s, where for each k, $W^{(k)} \in \mathbb{R}^{d_H \times d_H}$ and $W_{edge}^{(k)} \in \mathbb{R}^{2d_H}$. Therefore, the number of additional parameters is $k_{max}(d_H^2 + 2d_H) = \Theta(k_{max}d_H^2)$. $\square$

Discussions on the complexity of other baselines can be found in Appendix E.

A.4. A Relaxed yet Equivalent Variant of Definition 1

In Definition 1 in the main text, we have used the condition on the identity of the features of several nodes. Such a condition has been used for the ease and simplicity of presentation. However, we acknowledge that exactly equal node features are not common in practice, and thus, we provide here a relaxed yet equivalent variant of Definition 1.

Definition 3 (A relaxed yet equivalent variant of Definition 1). Given $G = (V, E)$ and initial node features X satisfying Assumption 1, we say that f_{att} is *vulnerable to over-smoothing* (V2OS), if given any $\epsilon > 0$, there exists $\xi > 0$, such that $\forall \theta$, $\left\| \check{\alpha}_{ij}^{(k'+1)} - \check{\alpha}_{i'j'}^{(k'+1)} \right\| \leq \epsilon$ and $\left\| \gamma_i^{(k')} - \gamma_{i'}^{(k')} \right\| \leq \epsilon$, for any $(i, j), (i', j') \in E$ and $k' \geq 1$ with $\left\| H_x^{(k')} - H_y^{(k')} \right\| \leq \alpha$, for each $\{x, y\} \in \binom{\{i,j,i',j'\}}{2}$; we say that f_{att} is *weakly resistant to over-smoothing* (WR2OS), if given any $\epsilon, \xi > 0$ and any $(i, j), (i', j') \in E$ and $k' \geq 1$ with $\left\| H_x^{(k')} - H_y^{(k')} \right\| \leq \xi$, for each $\{x, y\} \in \binom{\{i,j,i',j'\}}{2}$, there exists θ, such that $\left\| \check{\alpha}_{ij}^{(k'+1)} - \check{\alpha}_{i'j'}^{(k'+1)} \right\| > \epsilon$ or $\left\| \gamma_i^{(k')} - \gamma_{i'}^{(k')} \right\| > \epsilon$; and we say that f_{att} is *strongly resistant to over-smoothing* (SR2OS), if given any $\epsilon, \xi > 0$ and any $(i, j), (i', j') \in E$ and $k' \geq 1$ with $\left\| H_x^{(k')} - H_y^{(k')} \right\| \leq \xi$, for each $\{x, y\} \in \binom{\{i,j,i',j'\}}{2}$, there exists θ, such that $\left\| \check{\alpha}_{ij}^{(k'+1)} - \check{\alpha}_{i'j'}^{(k'+1)} \right\| > \epsilon$ and $\left\| \gamma_i^{(k')} - \gamma_{i'}^{(k')} \right\| > \epsilon$.

Remark 4. With the above variant of Definition 1, Theorems 1 and 3 still hold. We use Definition 1 in the main text for ease and simplicity of presentation.

Table 5: Node Classification Performance of GATs with and without Non-Linear Propagation

Dataset	Chameleon	Squirrel	Actor	Texas	Cornell	Wisconsin	Computer	Photo	Wiki-CS	Pubmed	Citeseer	Cora
Homophily	0.04	0.03	0.01	0.00	0.02	0.05	0.70	0.77	0.57	0.66	0.63	0.77
GATv2	**69.06** ± 2.2	**57.67** ± 2.4	30.27 ± 0.8	60.32 ± 7.0	58.35 ± 3.8	**61.94** ± 4.7	84.19 ± 1.2	89.87 ± 1.2	**79.64** ± 0.5	79.12 ± 0.3	71.15 ± 1.2	**83.88** ± 0.6
GATv2 *(Lin)*	68.73 ± 2.0	57.54 ± 1.8	**30.45** ± 1.0	**60.81** ± 6.1	**58.65** ± 4.8	60.20 ± 5.8	**85.01** ± 0.8	**90.02** ± 0.8	79.53 ± 0.5	**79.19** ± 0.4	**71.40** ± 1.0	83.87 ± 0.5
GATv2^R	70.88 ± 1.9	**61.23** ± 1.5	**33.73** ± 0.9	60.68 ± 6.6	57.32 ± 4.5	**60.61** ± 5.1	81.73 ± 2.2	88.71 ± 1.7	**79.75** ± 0.6	78.28 ± 0.4	71.00 ± 0.8	**82.42** ± 0.6
GATv2^R *(Lin)*	**70.92** ± 1.7	60.40 ± 1.6	33.67 ± 0.7	**61.35** ± 7.2	**58.92** ± 3.6	56.08 ± 5.8	**83.13** ± 1.5	**89.42** ± 1.1	79.68 ± 0.8	**78.44** ± 0.4	**71.36** ± 0.7	82.42 ± 0.5
GT	**69.34** ± 1.2	**55.04** ± 1.9	**36.29** ± 1.0	84.08 ± 5.6	**80.00** ± 4.9	**84.80** ± 4.3	84.38 ± 1.3	**91.28** ± 1.1	**79.93** ± 0.5	**79.04** ± 0.5	70.16 ± 0.8	**82.09** ± 0.7
GT *(Lin)*	66.75 ± 1.4	49.46 ± 2.0	36.24 ± 0.9	**84.59** ± 5.4	79.73 ± 4.9	83.33 ± 5.3	**85.29** ± 1.2	90.58 ± 1.7	78.88 ± 0.5	78.93 ± 0.5	**70.35** ± 0.6	81.97 ± 0.6

- *(Lin)* denotes the linear propagation version of the model.
- In each column, **bold** indicates higher performance between the two versions of the GATs.

B. Rephrased Propagation

In this section, we explain how we rephrase the propagation formulae for GATv2 and FAGCN and obtain the formulae in Table 2.

GATv2. Many prior studies on GNNs have dropped non-linearity for their theoretical analyses (Wang et al., 2019; Li et al., 2018; Ma et al., 2021; Baranwal et al., 2021; Liu et al., 2020; Chien et al., 2021). In defining cumulative attention $T^{(k)}$, we also remove GATv2's non-linearity *in the propagation layers*, while the non-linearity in computing edge attention coefficients $a_{ij}^{(k)}$'s is kept. The detailed rephrasing is

$$H^{(k)} = \sigma(\mathcal{A}^{(k)} H^{(k-1)} W^{(k)}) \rightarrow \mathcal{A}^{(k)} H^{(k-1)} W^{(k)}$$

We define GATv2's $T^{(k)}$ from this rephrase, such that $T^{(k)} = \prod_{\ell=k}^{1} \mathcal{A}^{(\ell)}$. [18]

We claim that removing non-linearity in the propagation layers may not significantly affect the performance and expressive power of GATv2, especially in the context of node classification. In fact, two theoretical works have argued that non-linear propagation is not always beneficial in the context of node classification:

- Wei et al. (2022) use the Contextual Stochastic Block Model (CSBM) to show that non-linear propagation and linear propagation return similar performance, except when the node features are extremely informative.

- Wu et al. (2023) also use CSBM, and they show that adding non-linearity after propagation does not improve model performance.

One theoretical work has argued for the universality of linear GNNs.

[18]$\Gamma^{(k)}$ is removed because it is the identity matrix for GATv2. Recall that $T^{(k)}$ intuitively expresses attention between all node pairs at layer k. Thus, the definition of cumulative attention $T^{(k)}$ after rephrase assumes linear propagation, such that the attention GATv2 expresses for k hop neighbors is equal to $\prod_{l=k}^{1} \mathcal{A}^{(l)}$

- Wang & Zhang (2022) show that GNNs without non-linearity are universal w.r.t. one-dimensional prediction, if there are no multiple eigenvalues of graph Laplacian or missing frequency components in node features.

Some empirical evidence also exists in line with the theoretical findings:

- Liu et al. (2020) report that the hidden node features become smooth at a quicker rate for GCN (a non-linear propagation model) than for DAGNN (a linear propagation model).

- Zhang et al. (2022) argues with empirical evidence that stacked non-linearity may relate to performance degradation in deep GNNs.

- Moreover, many GNNs achieve their state-of-the-art performance in node classification without non-linearity in their propagation layers, such as APPNP, GPRGNN, DAGNN, FAGCN, ASGC (Chanpuriya & Musco, 2022), JacobiConv (Wang & Zhang, 2022), and SlenderGNN (Yoo et al., 2022).

This does not mean that using non-linear propagation is futile. However, we argue that removing non-linearity from GATv2's propagation layer may not significantly compromise its performance and expressive power, especially in the context of node classification. Finally, in Table 5, we empirically evidence our claim by comparing the performance of original GATs to the rephrased, linear propagating GATs. We observe that GATs with linear propagation have competitive performance.

FAGCN. The original propagation equations of FAGCN are as follows:

$$H^{(0)} = \sigma(W^{(1)} X)$$
$$Z^{(k)} = \Gamma Z^{(0)} + \mathcal{A}^{(k)} Z^{(k-1)}$$
$$Z^{*} = W^{(2)} Z^{(k)}$$

Table 6: Statistics of the Benchmark Datasets

Dataset	Chameleon	Squirrel	Actor	Texas	Cornell	Wisconsin	Computer	Photo	Wiki-CS	Pubmed	Citeseer	Cora
# Nodes	2,277	5,201	7,600	183	183	251	13,752	7,650	11,701	19,717	3,327	2,708
# Edges	31,371	198,353	26,659	279	277	450	245,861	119,081	215,603	44,324	4,552	5,278
# Features	2,325	2,089	932	1,703	1,703	1,703	767	745	300	500	3,703	1,433
# Labels	5	5	5	5	5	5	10	8	10	3	6	7
Homophily	0.04	0.03	0.01	0.00	0.02	0.05	0.70	0.77	0.57	0.66	0.63	0.77
Split Type	Fixed	Fixed	Fixed	Fixed	Fixed	Fixed	Random	Random	Fixed	Fixed	Fixed	Fixed
Data Split (%)	48/32/20	48/32/20	48/32/20	48/32/20	48/32/20	48/32/20	2.5/2.5/95	2.5/2.5/95	5.0/15/50	0.3/2.5/5.0	5.2/18/37	3.6/15/30

We rephrase the above equations by

$$Z^{(k)}$$
$$= \Gamma H^{(0)} + \mathcal{A}^{(k)} Z^{(k-1)}$$
$$= \Gamma H^{(0)} + \mathcal{A}^{(k)}(\Gamma H^{(0)} + \mathcal{A}^{(k-1)} Z^{(k-2)})$$
$$= \Gamma H^{(0)} + \Gamma \mathcal{A}^{(k)} H^{(0)} + \mathcal{A}^{(k)} \mathcal{A}^{(k-1)} Z^{(k-2)})$$
$$= \Gamma H^{(0)} + \Gamma \mathcal{A}^{(k)} H^{(0)} + \mathcal{A}^{(k)} \mathcal{A}^{(k-1)}(\Gamma H^{(0)} + \mathcal{A}^{(k-2)} Z^{(k-3)}))$$
$$= \cdots$$
$$= (\Gamma + \Gamma \mathcal{A}^{(k)} + \Gamma \mathcal{A}^{(k)} \mathcal{A}^{(k-\ell)} + \cdots + \Gamma \prod_{\ell=k}^{1} \mathcal{A}^{(\ell)}) H^{(0)}.$$

C. Benchmark Datasets

C.1. Dataset Description

We use 12 node classification benchmark datasets in our experiments:

- The *chameleon* and *squirrel* datasets are webpage networks of Wikipedia (Pei et al., 2020; Rozemberczki et al., 2021), where each node represents a webpage on Wikipedia, and two nodes are adjacent if mutual links exist between the two corresponding web pages. For each node, the node features are informative nouns on the corresponding webpage, and the node label represents the category of the average monthly traffic of the corresponding webpage.

- The *actor* dataset is the actor-only induced subgraph of a film-director-actor-writer network obtained from Wikipedia webpages (Pei et al., 2020; Tang et al., 2009), where each node represents an actor, and two nodes are adjacent if the two corresponding actors appear on the same Wikipedia webpage. For each node, the node features are derived from the keywords on the Wikipedia webpage of the corresponding actor, and the node label is determined by the words on the webpage.

- The *texas*, *cornell*, and *wisconsin* datasets are extracted from the WebKB dataset (Pei et al., 2020), where each node represents a webpage, and two nodes are adjacent if a hyperlink between the two corresponding webpages exists. For each node, the node features are the bag-of-words features of the corresponding webpage, and the node label is the category of the webpage.

- The *computer* and *photo* datasets are Amazon co-purchase networks (Shchur et al., 2018), where each node represents a product, and two nodes are adjacent if the two corresponding products are frequently purchased together. For each node, the node features are the bag-of-words features of the customer reviews of the corresponding product, and the node label is the category of the product.

- The *wiki-cs* dataset is a webpage network of Wikipedia (Mernyei & Cangea, 2020), where each node represents a Wikipedia webpage related to computer science, and two nodes are adjacent if a hyperlink exists between the two corresponding webpages. For each node, the node features are the GloVe word embeddings of the corresponding webpage, and the node label represents the article category of the webpage.

- The *pubmed*, *citeseer*, and *cora* datasets are citation networks (Yang et al., 2016), where each node represents a research article, and two nodes are adjacent if a citation exists between the two corresponding articles. For each node, the node features are the bag-of-words features of the corresponding article, and the node label is the category of the research domain of the article.

Following Chien et al. (2021), we preprocess the directed graphs among the datasets to be undirected, which is commonly done for node classification. For measuring the degree of homophily of each dataset, we use the homophily metric suggested by Lim et al. (2021b). In Table 6, we present the basic statistics of the benchmark datasets after preprocessing.

C.2. Train/Val/Test Split

Experiments are generally conducted under the most commonly used settings. Two different training schemes are used: sparse- and dense-labeled training. Sparse- and dense-labeled training, respectively, refer to having small and large proportions of train labels. In line with the prior works, we conducted experiments with sparse-labeled training for the homophilic graphs (Welling & Kipf, 2016; Veličković et al., 2018; Gasteiger et al., 2018; Liu et al., 2020) and dense-labeled training for the heterophilic graphs (Pei et al., 2020; Chien et al., 2021; Bo et al., 2021).

Overall, we used publicly available splits. The public splits of *cora*, *citeseer*, and *pubmed* were given by (Yang et al., 2016), and *wiki-cs* by (Mernyei & Cangea, 2020). Since *computer* and *photo* datasets do not have publicly available splits, we follow the methodology of prior works (Chien et al., 2021) and use a random (2.5%, 2.5%, 95%) split for each trial. For heterophilic graphs, the public splits were provided by Pei et al. (2020).

D. Hyperparameters

In this section, we detail the hyperparameter search space of the 14 GNN models used in the experiments. Overall, for each baseline model, we try to maintain the experimental settings and hyperparameters search spaces used in the original paper.

Learning Rate and Weight Decay. For all the methods, we set the learning rate as 0.01 for sparse-labeled training and 0.005 for dense-labeled training. For models with separate decay weights for parameters in propagation and feature transformation layers, we use WD_{prop} and WD_{ft} to denote each.

Hidden Dimension d_H. We generally maintain the same hidden dimension sizes for all models. For GAT, GATv2, GAT-Res, and GT, we use 8 attention heads and set the hidden dimension as 8, so that the total hidden dimension = 64. For FAGCN, we follow the original paper, setting the hidden dimension = 16 for homophilic graphs and setting the hidden dimension = 32 for heterophilic graphs. For others, the hidden dimension is set to 64.

Number of Layers k_{max}. For APPNP and GPRGNN, the number of propagation layers is set as 10, as proposed in the original papers. For GCN, we use two layers. Mixhop uses two dense linear layers, and we search for the best number of propagation layers per linear layer. For AERO-GNN, we use a two-layer MLP to obtain $H^{(0)}$ for dense-labeled training, while a one-layer MLP is used for the sparse-labeled training. For others, we search their best-performing layer.

Dropout. For AERO-GNN, an additional dropout layer is optionally added before the last linear layer to obtain Z^* from $Z^{(k_{max})}$, when doing so is helpful for the performance (specifically, in *actor*, *texas*, *photo*, *wiki-cs*, *citeseer*, and *cora* datasets).

Search Space. For each model, we choose the hyperparameters with which the model returns the best mean performance across 5 fixed random seeds. For the search spaces with a combinatorial size of below 300, we use grid search. For the search spaces consisting of at least 300 different combinations, (including GATv2^R, FAGCN, GPRGNN, GCN-II, A-DGN, and AERO-GNN), we run 300 search trials using a Bayesian optimization algorithm.

Below, we list the hyperparameter search space for each model:

1. **GCN**:
   - WD $\in \{1e-2, 5e-3, 1e-3, 5e-4, 1e-4\}$
   - dropout $\in \{0.5, 0.6, 0.7, 0.8\}$
   - L2 $\in \{0.0, 5e-4\}$

2. **GAT, GATv2, GT, DMP**:
   - WD $\in \{1e-2, 5e-3, 1e-3, 5e-4, 1e-4\}$
   - dropout $\in \{0.5, 0.6, 0.7, 0.8\}$
   - $k_{max} \in \{2, 3\}$
   - L2 $\in \{0.0, 5e-4\}$

3. **GATv2^R**:
   - WD $\in \{$1e-2, 5e-3, 1e-3, 5e-4, 1e-4$\}$
   - dropout $\in \{0.5, 0.6, 0.7, 0.8\}$
   - $k_{max} \in \{2, 4, 8, 16, 32\}$
   - residual connection weight $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$
   - L2 $\in \{0.0, 5e-4\}$

4. **FAGCN**:
   - WD $\in \{1e-2, 5e-3, 1e-3, 5e-4, 1e-4\}$
   - dropout $\in \{0.4, 0.5, 0.6, 0.7, 0.8\}$
   - $k_{max} \in \{1, 2, 3, 4, 5, 6, 7, 8\}$
   - residual connection weight $\epsilon \in \{0.1, 0.2, ..., 0.9, 1.0\}$
   - L2 $\in \{0.0, 5e-4\}$

5. **APPNP**:
   - WD $\in \{1e-2, 5e-3, 1e-3, 5e-4, 1e-4\}$
   - dropout $\in \{0.5, 0.6, 0.7, 0.8\}$
   - L2 $\in \{0.0, 5e-4\}$
   - return probability $\alpha \in \{0.1, 0.3, 0.5, 0.9\}$

6. **GPRGNN**:
   - WD $\in \{1e-2, 5e-3, 1e-3, 5e-4, 1e-4\}$
   - dropout $\in \{0.5, 0.6, 0.7, 0.8\}$
   - $k_{max} \in \{4, 8, 16, 32\}$
   - L2 $\in \{0.0, 5e-4\}$
   - return probability $\alpha \in \{0.1, 0.3, 0.5, 0.9\}$

7. **DAGNN**:
   - WD $\in \{0, 2e-2, 1e-2, 5e-3, 1e-3, 5e-4, 1e-4, 5e-5, 1e-5\}$
   - dropout $\in \{0.5, 0.6, 0.7, 0.8\}$
   - $k_{max} \in \{5, 10, 20\}$
   - L2 $\in \{0.0, 5e-4\}$

8. **MixHop**:

   - WD $\in \{1e-2, 5e-3, 1e-3, 5e-4, 1e-4\}$
   - dropout $\in \{0.5, 0.6, 0.7, 0.8\}$
   - $k_{max} \in \{6, 10\}$[19]
   - L2 $\in \{0.0, 5e-4\}$

9. **GCN-II**:

   - $\text{WD}_{ft} \in \{1e-3, 5e-4, 1e-4, 5e-5, 1e-5, 5e-6, 1e-6\}$
   - $\text{WD}_{prop} \in \{1e-2, 5e-3, 1e-3, 5e-4, 1e-4\}$
   - dropout $\in \{0.5, 0.6, 0.7, 0.8\}$
   - $k_{max} \in \{4, 8, 16, 32, 64\}$
   - residual connection weight $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$
   - weight decay $\lambda \in \{0.5, 1.0, 1.5\}$
   - L2 $\in \{0.0, 5e-4\}$

10. **A-DGN**:

    - $\text{WD}_{ft} \in \{1e-3, 5e-4, 1e-4, 5e-5, 1e-5, 5e-6, 1e-6\}$
    - $\text{WD}_{prop} \in \{1e-2, 5e-3, 1e-3, 5e-4, 1e-4\}$
    - dropout $\in \{0.0, 0.3, 0.5, 0.7\}$
    - $k_{max} \in \{2, 4, 8, 16, 32\}$
    - discretization step $\epsilon \in \{1, 0.1, 0.01, 0.001, 0.0001\}$
    - diffusion strength $\lambda \in \{1, 0.1, 0.01, 0.001, 0.0001\}$
    - L2 $\in \{0.0, 5e-4\}$

11. **AERO-GNN**:

    - $\text{WD}_{ft} \in \{4e-2, 2e-2, 1e-2, 5e-3, 1e-3, 5e-4, 1e-4\}$
    - $\text{WD}_{prop} \in \{2e-2, 1e-2, 5e-3, 1e-3, 5e-4, 1e-4\}$
    - dropout $\in \{0.5, 0.6, 0.7, 0.8\}$
    - $k_{max} \in \{4, 8, 16, 32\}$[20]
    - weight decay $\lambda \in \{0.25, 0.5, 1.0\}$

E. Model Complexity Analysis

In Theorem 5, we have discussed the number of parameters for AERO-GNN, FAGCN, and GATv2. Below, we analyze the model complexity for all the baselines. Recall that we ignore the parameters used in computing the first hidden features ($H^{(0)}$) and those in the output layer, since they are used in all GNN models. That is, we only consider the number of *additional parameters* other than those parameters.

We observe the number of additional parameters used in AERO-GNN is comparable to, or even smaller than, those in the other representative GNNs.

---

[19] k_{max} of Mixhop is the number of fully-connected layers (2) $\times$ the number of propagation layers (3 or 5)

[20] AERO-GNN's performance sometimes marginally increase when $k_{max} = 64$, but it often saturates between k_{max} of 16 and 32. For efficient search, we did not include $k_{max} = 64$.

GCN and GCN-II. The additional parameters include $W^{(k)}$'s, where for each k, $W^{(k)} \in \mathbb{R}^{d_H \times d_H}$. Therefore, the number of additional parameters is $k_{max} d_H^2 = \Theta(k_{max} d_H^2)$.

A-DGN. The additional parameters include $2W$'s, where each $W \in \mathbb{R}^{d_H \times d_H}$. Therefore, the number of additional parameters is $2d_H^2 = \Theta(d_H^2)$.

GAT, GATv2, and GATv2^R. The additional parameters include $W^{(k)}$'s and $W_{edge}^{(k)}$'s, where for each k, $W^{(k)} \in \mathbb{R}^{d_H \times d_H}$ and $W_{edge}^{(k)} \in \mathbb{R}^{2d_H}$. Therefore, the number of additional parameters is $k_{max} d_H^2 + k_{max}(2d_H) = \Theta(k_{max} d_H^2)$.

GT. The additional parameters include $W_1^{(k)}$'s, $W_2^{(k)}$'s, $W_3^{(k)}$'s, and $W_4^{(k)}$'s, where for each k, $W^{(k)} \in \mathbb{R}^{d_H \times d_H}$. Therefore, the number of additional parameters is $k_{max}(4d_H^2) = \Theta(k_{max} d_H^2)$.

DMP. The additional parameters include $W^{(k)}$'s and $W_{edge}^{(k)}$'s, where for each k, $W^{(k)} \in \mathbb{R}^{d_H \times d_H}$ and $W_{edge}^{(k)} \in \mathbb{R}^{d_H \times d_H}$. Therefore, the number of additional parameters is $k_{max}(2d_H^2) = \Theta(k_{max} d_H^2)$.

FAGCN. The additional parameters include $W_{edge}^{(k)}$'s, where for each k, $W_{edge}^{(k)} \in \mathbb{R}^{2d_H}$. Therefore, the number of additional parameters is $k_{max}(2d_H) = \Theta(k_{max} d_H)$.

APPNP. APPNP does not require additional parameters after obtaining $H^{(0)}$. Therefore, the number of additional parameters is 0.

DAGNN. The additional parameters include a $W_{hop} \in \mathbb{R}^{d_C}$, where d_C = the number of labels. Therefore, the number of additional parameters is $d_C = \Theta(d_C)$.

GPRGNN The number of additional parameters is $\Theta(k_{max})$ since at each layer, a constant number of parameters are used to compute the attention coefficients.

AERO-GNN. The additional parameters include $W_{edge}^{(k)}$'s, $W_{hop}^{(k)}$'s, and $b_{hop}^{(k)}$'s, where for each k, $W_{edge}^{(k)} \in \mathbb{R}^{2d_H}$, $W_{hop}^{(k)} \in \mathbb{R}^{2d_H}$, and $b_{hop}^{(k)} \in \mathbb{R}$. Therefore, the number of additional parameters is $k_{max}(4d_H + 1) = \Theta(k_{max} d_H)$.

We conclude again that the number of additional parameters used in AERO-GNN is comparable to, or even smaller than, those in the other attention-based GNNs that use nontrivial edge attention. Most baseline models use $\Theta(k_{max} d_H^2)$ additional parameters, while AERO-GNN and FAGCN use $\Theta(k_{max} d_H)$ additional parameters.

F. Implementation Details

Here, we provide details about how the GNN models are implemented.

Table 7: AERO-GNN Performance with Multi-Head Attention

Dataset	Chameleon	Squirrel	Actor	Texas	Cornell	Wisconsin	Computer	Photo	Wiki-CS	Pubmed	Citeseer	Cora
Homophily	0.04	0.03	0.01	0.00	0.02	0.05	0.70	0.77	0.57	0.66	0.63	0.77
AERO-GNN (1-head)	71.58 ± 2.4	**61.76 ± 2.4**	**36.57 ± 1.1**	**84.35 ± 5.2**	81.24 ± 6.8	**84.80 ± 3.3**	**86.69 ± 1.4**	92.50 ± 0.7	**79.96 ± 0.5**	**80.65 ± 0.5**	**73.20 ± 0.6**	**84.02 ± 0.5**
AERO-GNN (4-head)	**72.15 ± 1.7**	61.50 ± 2.0	36.00 ± 1.3	82.78 ± 5.7	**81.30 ± 7.2**	82.98 ± 4.9	85.67 ± 1.4	**92.50 ± 0.8**	79.40 ± 0.5	80.53 ± 0.5	72.01 ± 1.2	83.93 ± 0.6

GAT, GATv2, FAGCN. We use official implementations from PyTorch Geometric (Fey & Lenssen, 2019). The dropout layer is applied to edge attention coefficients. The same weight matrix $W^{(k)}$ is applied to obtain both the source node and destination node features $H_i^{(k)}, H_j^{(k)}$.

GATv2^R. Following GCN-II and APPNP, we added an initial residual connection to GATv2. After obtaining $H^{(0)}$ from the initial feature matrix X, $H^{(0)}$ is added to hidden features of the next layers by residual weight ϵ. Formally, GATv2^R layer is defined as $Z^{(k)} = \sigma(((1 - \epsilon)\mathcal{A}^{(k)}H^{(k)} + \epsilon H^{(0)})W^{(k)})$.

GT. GT computes edge attention coefficients with a transformer-like function. We used official implementation from PyTorch Geometric. The dropout layer is applied to edge attention coefficients. We do not use the gated function β or Layer Normalization (Ba et al., 2016).

DMP. DMP computes vector edge attention coefficients, instead of a scalar attention coefficient, for each node pair. Then, it uses tanh to bound the edge attention coefficients, without further normalization. Since no official implementation is made public, we use our own implementation. Specifically, we used the DMP-1-Sum variant, where the neighbor information is sum-aggregated.

MixHop. MixHop concatenates the hidden feature $H^{(k)}$'s of different hops, which can be seen as a (simplified) form of hop attention. Simply put, each feature transformation layer comes after the concatenation of node features from multiple propagation layers. We used the implementation of MixHop from Lim et al. (2021a).

GPRGNN. PageRank initialization is used for the hop attention coefficients $\gamma_i^{(k)}$'s. As originally implemented, we do not apply optimizer weight decay for the hop attention $\gamma_i^{(k)}$'s.

GCN-II. We use official implementations from PyTorch Geometric. GCN-II version with the shared weight matrix $W^{(k)}$ for initial and smoothed node feature ($H^{(0)}$ and $H^{(k)}$) is chosen.

A-DGN. We use official implementations from PyTorch Geometric. We used tanh activation and GCN layer as the message passing function.

Table 8: Node Classification Performance on the Filtered Datasets

Dataset	Chameleon(F)	Squirrel(F)
Homophily	0.04	0.04
GCN	43.29 ± 3.8	41.30 ± 1.9
APPNP	40.45 ± 3.1	39.18 ± 1.9
GCN-II	43.01 ± 3.9	42.96 ± 2.6
A-DGN	**46.60 ± 3.8**	**44.70 ± 2.1**
GAT	41.06 ± 3.6	39.34 ± 1.9
GATv2	42.26 ± 2.9	37.93 ± 2.3
GATv2^R	41.38 ± 3.1	39.57 ± 1.7
GT	40.94 ± 3.2	39.28 ± 1.9
FAGCN	**46.14 ± 4.4**	43.25 ± 2.2
DMP	41.45 ± 4.6	41.28 ± 2.2
MixHop	41.25 ± 4.2	39.44 ± 2.2
GPRGNN	40.60 ± 3.3	40.62 ± 2.1
DAGNN	40.30 ± 3.3	37.13 ± 2.0
AERO-GNN	43.30 ± 3.6	**46.23 ± 2.2**

G. Additional Experiments

G.1. Generalization of AERO-GNN to Multi-Heads.

We generalize AERO-GNN for multi-head attention by modifying the dimension of the attention functions. After obtaining $H^{(0)} \in \mathbb{R}^{n \times \eta \times d_H}$ with an MLP, where η denotes the number of attention heads, dimension of attention weights are modified to $W_{edge}^{(k)} \in \mathbb{R}^{\eta \times 2d_H}$ and $W_{hop}^{(k)} \in \mathbb{R}^{\eta \times 2d_H}$. Subsequently, $\alpha_{ij}^{(k)}, \gamma_i^{(k)} \in \mathbb{R}^{\eta}$ and $H^{(k)}, Z^{(k)} \in \mathbb{R}^{n \times \eta \times d_H}$. At k_{max}, the multi-headed node features are concatenated to obtain the final node feature $Z^{(k_{max})} \in \mathbb{R}^{n \times (\eta \times d_H)}$.

We set the hidden dimension to be 16, with 4 attention heads. Consistent with the reports from Brody et al. (2021), we do not find multi-head attention to be always beneficial, while it occasionally improves performance. Refer to Table 7 for details about performance with multi-head attention.

G.2. Datasets Proposed in Platonov et al. (2023).

Recently, Platonov et al. (2023) argued that *chameleon* and *squirrel* datasets may have duplicate nodes and, consequently, train-test leakage problem. Therefore, they proposed to filter their duplicates for node classification. For a comprehensive evaluation, we conduct node classification on the filtered datasets (Table 8). The same experimental settings are maintained.

Consistent with the findings from Platonov et al. (2023), model performance drops significantly in the filtered
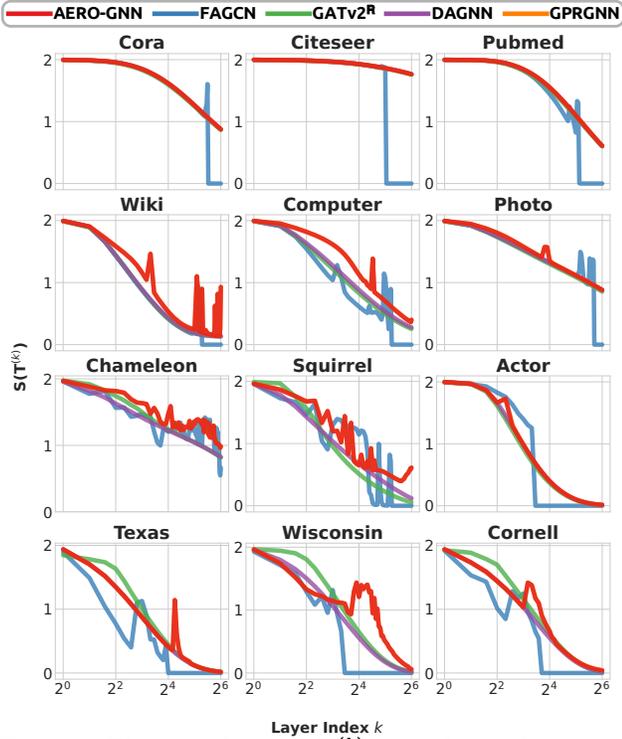
Figure 4: **The smoothness of $T^{(k)}$ for each k with $k_{max} = 64$, for all 12 datasets.**

datasets. Here, A-DGN performs the best, while AERO-GNN ranks second. It is still worth noting that AERO-GNN has the best average ranking among the attention-based GNNs.

G.3. Empirical Evaluation of Cumulative Attention.

As an extension of section 5.3, analyses of cumulative attention $T^{(k)}$ for *all 12 datasets* are provided in Figure 4. Overall, the results are consistent with what is shown in Section 5.3, and hence, we draw the same conclusions. AERO-GNN has consistently less smooth cumulative attention $T^{(k)}$ and often un-smoothes it.

G.4. Empirical Analysis of Attention Coefficients.

As an extension of section 5.3, analyses of attention coefficients for *all 12 datasets* are provided in Figure 5 and 6. Overall, the results are consistent with what is shown in section 5.3, and hence, we draw the same conclusions.

Besides, for Figure 1 and 5, it is noteworthy that we reversed layer index k (the x-axis) for GATv2^R and FAGCN. Specifically, due to their initial residual connection, the two models have $T^{(k)} = \Gamma^{(k)} \prod_{\ell=k_{max}}^{k_{max}-k+1} \mathcal{A}^{(\ell)}$, while other models have $T^{(k)} = \Gamma^{(k)} \prod_{\ell=k}^{1} \mathcal{A}^{(\ell)}$. That is, for the two models, there is an *inverse relationship* between the layer index k and the corresponding hop that the attention coefficients learn to propagate (recall that $T^{(k)}$ expresses attention between all node pairs within k hops).

1. **GATv2**:
   - **Fig. 5 (a)**: Distributions of $\alpha_{ij}^{(k)}$'s remain stationary across layers k's and similar across different graphs (*not* edge-, graph-adaptive).
   - **Fig. 5 (b)**: Differences between $\alpha_{ij}^{(k)}$'s and $\alpha_{ij}^{(k-1)}$'s drop significantly to near 0 at large k for some datasets (*not* hop-adaptive).

2. **FAGCN**:
   - **Fig. 5 (a)**: Distributions of $\alpha_{ij}^{(k)}$'s (Fig. 5 (a)) shrink near 0 at large k for all but *chameleon* dataset (learning failure).
   - **Fig. 5 (b)**: Differences between $\alpha_{ij}^{(k)}$'s and $\alpha_{ij}^{(k-1)}$'s (Fig. 5 (b)) drop significantly to near 0 at large k for some datasets (*not* hop-adaptive).

3. **GRPGNN**:
   - **Fig. 6 (a)**: Means of $\gamma_i^{(k)}$'s are different for different hops and graphs (hop-, graph-adaptive).
   - **Fig. 6 (b)**: Standard deviations of $\gamma_i^{(k)}$'s are always 0 (*not* node-adaptive).

4. **DAGNN**:
   - **Fig. 6 (a)**: Means of $\gamma_i^{(k)}$'s remain stationary across layers k and graphs (*not* hop-, graph-adaptive).
   - **Fig. 6 (b)**: Standard deviations of $\gamma_i^{(k)}$'s are always very small (*not* node-adaptive).

5. **AERO-GNN**:
   - **Fig. 5 (a)**: Distributions of $\alpha_{ij}^{(k)}$'s change across layers k and are different across different graphs (edge-, graph-adaptive).
   - **Fig. 5 (b)**: Differences between $\alpha_{ij}^{(k)}$'s and $\alpha_{ij}^{(k-1)}$'s never drop significantly near 0. (hop-adaptive)
   - **Fig. 6 (a)**: Means of $\gamma_i^{(k)}$'s are different for different hops and graphs (hop-, graph-adaptive).
   - **Fig. 6 (b)**: Standard deviations of $\gamma_i^{(k)}$'s are almost always high and have different distributions across different graphs (node-, graph-adaptive).

Again, we claim that only the attention function of AERO-GNN can remain adaptive to edge/node, hop, and graph at deep layers.
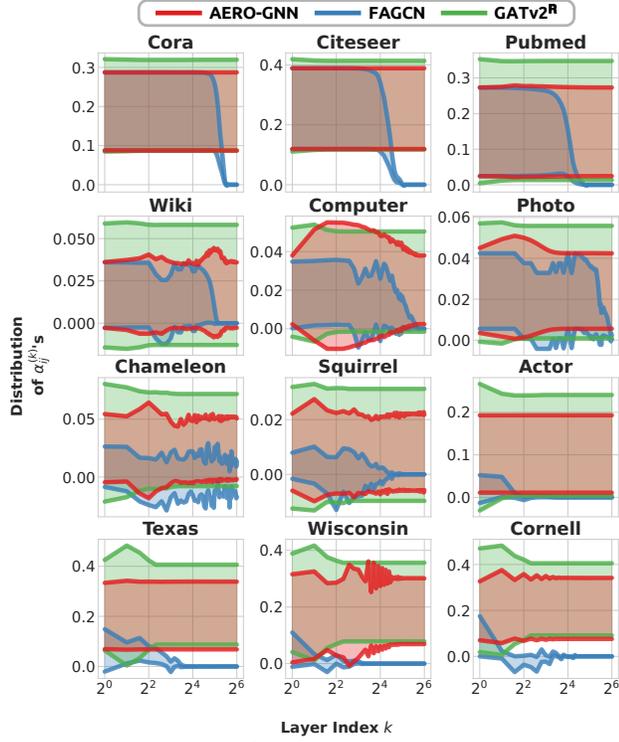
(a) The distribution of $\alpha_{ij}^{(k)}$'s for each k. The shaded area represents mean ± 1 SD.
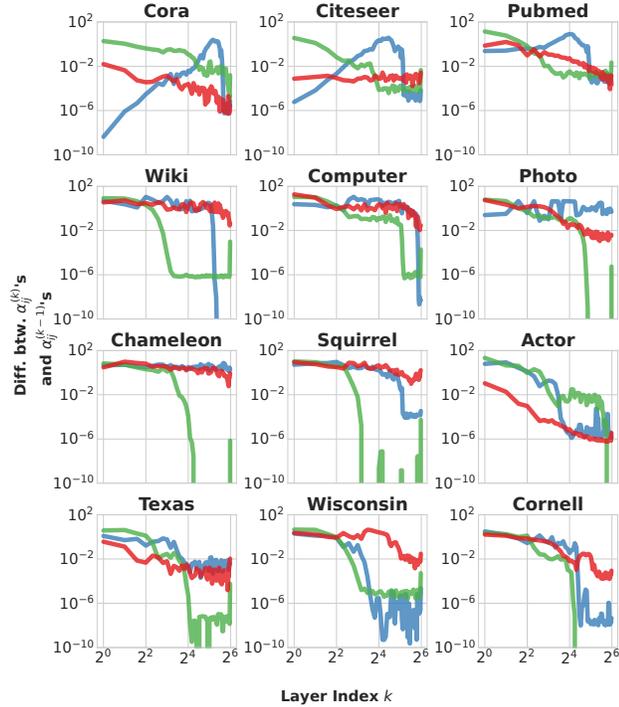
(b) The differences between $\alpha_{ij}^{(k)}$'s and $\alpha_{ij}^{(k-1)}$'s for each k. Formally, $\sqrt{\sum_{i,j}(\alpha_{ij}^{(k)} - \alpha_{ij}^{(k-1)})^2}$ is reported for each k.

Figure 5: **Statistics of $\alpha_{ij}^{(k)}$'s for each k with $k_{max} = 64$, for all 12 datasets.**
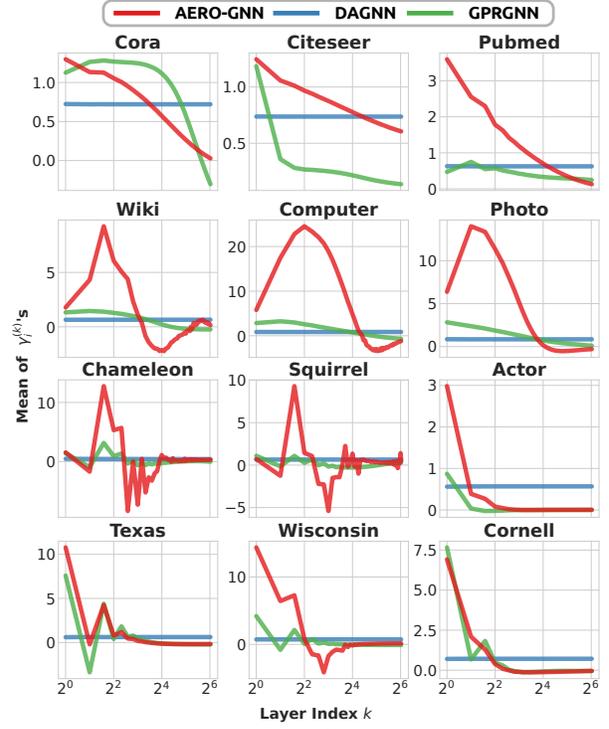
(a) The mean of $\gamma_i^{(k)}$'s for each k.
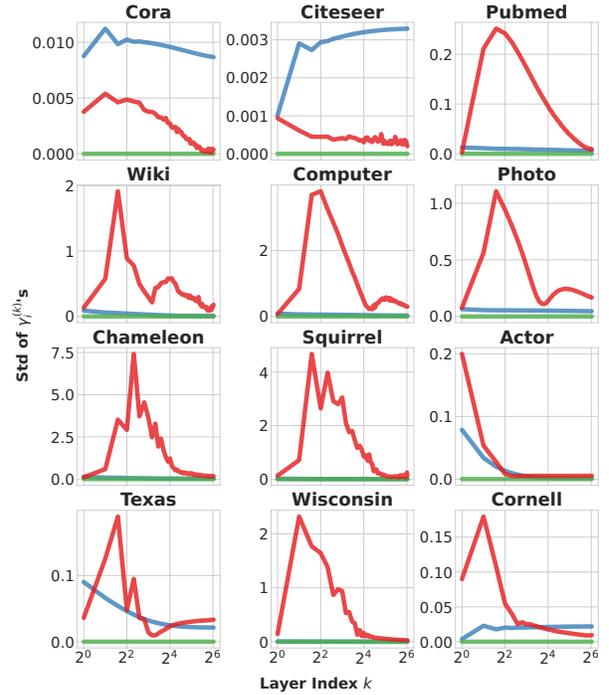
(b) The SD of $\gamma_i^{(k)}$'s for each k.

Figure 6: **Statistics of $\gamma_i^{(k)}$'s for each k with $k_{max} = 64$, for all 12 datasets.**