Oversmoothing Alleviation in Graph Neural Networks: A Survey and Unified View

Yufei Jin[1*] and Xingquan Zhu[2†]

[1] Dept. of Electrical Engineering & Computer Science, Florida Atlantic University, Boca Raton, 33431, FL, USA.
[2] Dept. of Electrical Engineering & Computer Science, Florida Atlantic University, Boca Raton, 33431, FL, USA.

*Corresponding author(s). E-mail(s): yjin2021@fau.edu;
Contributing authors: xzhu3@fau.edu;
†These authors contributed equally to this work.

Abstract

Oversmoothing is a common challenge in learning graph neural networks (GNN), where, as layers increase, embedding features learned from GNNs quickly become similar or indistinguishable, making them incapable of differentiating network proximity. A GNN with shallow layer architectures can only learn short-term relation or localized structure information, limiting its power of learning long-term connection, evidenced by their inferior learning performance on heterophilous graphs. Tackling oversmoothing is crucial for harnessing deep-layer architectures for GNNs. To date, many methods have been proposed to alleviate oversmoothing. The vast difference behind their design principles, combined with graph complications, make it difficult to understand and even compare the difference between different approaches in tackling the oversmoothing. In this paper, we propose `ATNPA`, a unified view with five key steps: Augmentation, Transformation, Normalization, Propagation, and Aggregation, to summarize GNN oversmoothing alleviation approaches. We first propose a taxonomy for GNN oversmoothing alleviation which includes three themes to tackle oversmoothing. After that, we separate all methods into six categories, followed by detailed reviews of representative methods, including their relation to `ATNPA`, and discussion of their niche, strength, and weakness. The review not only draws an in-depth understanding of existing methods in the field but also shows a clear road map for future study.

Keywords: Oversmoothing, graph neural networks, graph embedding, graph learning, review

1 Introduction

Graph Neural Networks (GNN) [1–3] have become prevalent in support learning from networked data, especially after the success of the Graph Convolution Network (GCN) [4]. The main goal of GNN is to learn feature representation [5] for network entities, such as nodes or edges, in order to support downstream tasks, like node classification or link prediction. While GNN has achieved competitive performance in many benchmark graph datasets, it is known to only perform well with shallow layer architectures but cannot learn long-term node-node relation well. One consequence of such inability leads to its inferior performance on heterophilous graph [6].

It has been shown that simply stacking GNN layers to build a deep architecture cannot learn well due to the observed oversmoothing phenomenon [7, 8]. Oversmoothing can be described as a phenomenon that all node emebeddings, after deep GNN layers, become similar to each other. Several measures such as Dirichlet energy [9] and Mean Average Distances (MAD) have been proposed to quantify the extent of oversmoothing of a model [10]. In this paper, unless otherwise specified, Dirichlet energy will be used as the major measure of oversmoothing for analysis. Figure 1 demonstrates the GCN embedding learning results from the Karate network [11], where increasing GCN layers from 1 to 3 results in better class separability whereas increasing the layer further results in embedding features with deteriorated class separability. Figure 2(b) demonstrates the oversmoothing phenomenon on the Cora network [12] where all node embedding becomes similar to each other.

GNN models equipped with oversmoothing alleviation can in general accommodate more GNN layers and therefore allow nodes to have a larger receptive fields [13]. As a result, models aiming to alleviate oversmoothing also tend to gain advantage over heterophilous datasets. Such dual relation has been observed in several studies [14, 15].

1.1 Research Gap and Motivations

Indeed, many works have been proposed to tackle oversmoothing in GNNs, by using different types of design principles. For example, energy-control approaches aim to increase initial energy or keep energy from exponential decay during propagation in GNNs. Other methods focus on decoupling topology propagation and feature transformation. The vast difference in their design principles, combined with complicated graph topology and message passing mechanisms, making it difficult to understand and even compare their difference in tackling the oversmoothing. Very few survey papers exist to review methods tackling GNN oversmoothing challenges. A recent survey [16] has compared several methods, and pointed out that some of the existing methods (such as GCNII [17], GraphCon [9]) cannot increase their model performance with deep layers despite the oversmoothing measure (*i.e.*, Dirichlet energy) is preserved to be constant among layers, mainly because of lacking expressive power. Therefore, existing survey [16] is mainly focused on reviewing method drawback from the expressive power perspective.

To date, there is no literature focusing on summarizing and comparing different alleviation approaches. Collectively, there is a missing knowledge of main themes and categorization of existing methods in the field, which may help researchers understand design principles to tackle oversmoothing. Individually, there is a lack of comparison of main stream approaches (*e.g.* strength and weakness) to guide future research.

Classical GNN message passing includes two critical operations: message aggregation and message update, where message aggregation focus on how to aggregate information passed from connected neighbors and message update focuses on how to update the node information to the next iteration or next layer. In many works, augmentation prior to training, such as feature or edge dropout, is also a part of standard pipeline, which can be simply described as a stochastic masking over original data sources, including graph topology and node features for graph learning.

To alleviate oversmoothing for GNN learning, most existing methods are motivated by modifying or changing the three operations, including augmentation, message aggregation, and message update, in the GNN training process with the objective of obtaining final embeddings without oversmoothing. Nevertheless, there is no systematic study of existing methods on how the three main operations affect the oversmoothing problem. To close the gap and provide guidance for future studies of oversmoothing, we propose a taxonomy with a unified view of five key steps: Augmentation, Transformation, Normalization, Propagation, and Aggregation, to summarize GNN oversmoothing alleviation approaches. The unified view therefore provides a systematic study of the three operations (augmentation, message aggregation, and message update) with more detailed analysis and categorization of existing methods.

We notice that the three major themes only correspond to three main operations in classical GNN but also reflect the underlying different principles that are leveraged to alleviate oversmoothing. The methods we selected, despite various motivations, inspirations, and structures, all fall into the three major themes and their underlying principles.

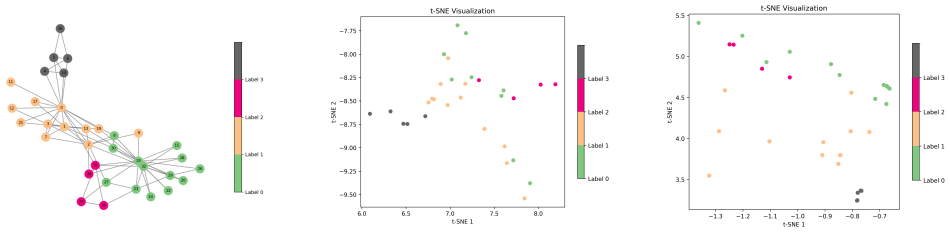
1.2 Contributions

In this paper, we propose to unify existing methods in the same form and study their connections in tackling GNN oversmoothing. Our study not only provides a taxonomy and a unified view, `ATNPA` to summarize all methods using common math formulations, but also separates them into three themes and six subgroups, by taking their unique designs into consideration. The survey outlines the differences between methods in each group, explains their rationality, and addresses their limitations. Our review has a number of math formulas, because reviewed papers are heavy in math formulations. To precisely summarize and highlight their difference, we keep representative methods' backbone formulas in the review for a better understanding.

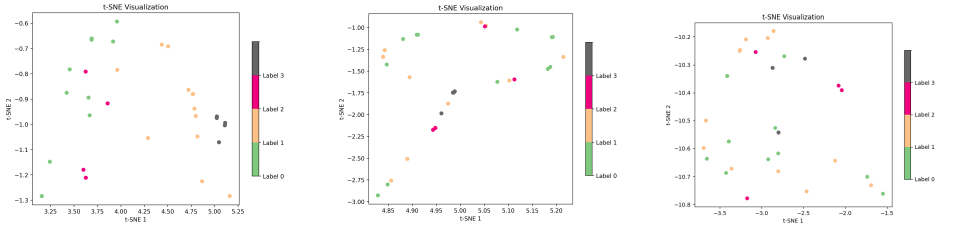
The remainder of the paper is structured as follows. Section 2 provides the definition of oversmoothing problem and its common measures in existing works. Section 3 describes our taxonomy for GNN oversmoothing problems from different angles. Section 4 introduces our main unified view and the six categories of existing methods, followed by a detailed discussion over each category. Section 5 concludes our works along with future guidelines for oversmoothing problems. For ease of reference, Table 1 summarizes key symbols and notations used in the paper.

2 Problem Notation

A graph with n nodes is denoted by $G(V, E, X)$, where $V = \{v_1, \ldots, v_n\}$ is the vertex set with $|V| = n$, E is the edge set, and $X \in \mathbb{R}^{n \times m}$ is the node content matrix recording m dimensional attributes for each node. For ease of representation, we use $A \in \mathbb{R}^{n \times n}$ to



(a) Karate Club Topology (b) 1-layer GCN Embedding (c) 3-layers GCN Embedding



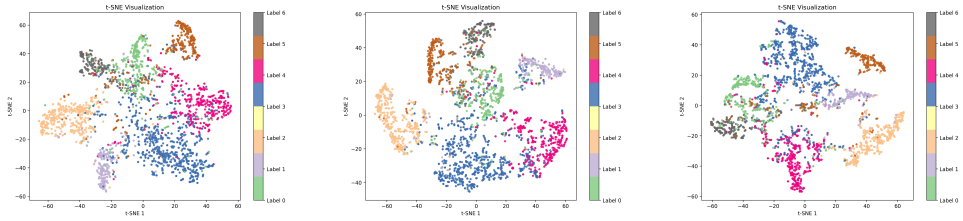
(d) 5-layers GCN Embedding (e) 10-layers GCN Embedding (f) 20-layers GCN Embedding

Fig. 1: Visualization of GCN node embedding results with respect to increasing number of layers (Karate club network [11]). Nodes are color-coded with same colored nodes belonging to the same class. From top to bottom, left to right, (a) denotes the Karate club network topoloy, (b) to (f) denotes node embedding from 1-layer GCN to 20-layer GCN, respectively. As GCN layer increases from 1-layer to 3-layer, the embedding achieve better class separability (*i.e.,* better results). As layer continuously, from 5-layers to 20-layers, GCN embedding loss node separability.

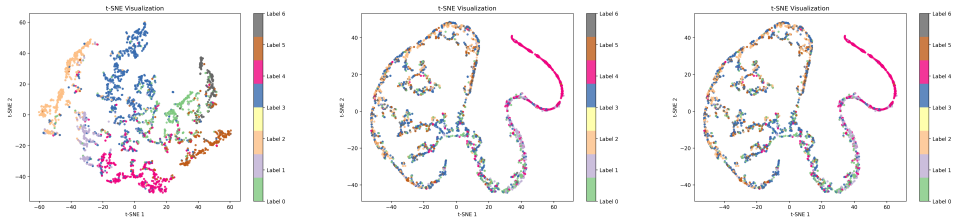
denote adjacency matrix of G, with $A[i,j] = 1$ if an edge connects v_i and v_j, or 0 otherwise. Learning node embedding (or feature representation) is essential for graph neural networks. Meanwhile, because embedding learning is often carried out in a layer-by-layer fashion, we use $H^l \in \mathbb{R}^{n \times f}$ to denotes feature embedding learned at the l^{th} layer (where each node is denoted by an f dimensional latent features). $\sigma(\cdot)$ denotes a non-linearity activation function. In the following, we define operators commonly used in GNN learning and will be using these operators in the later analysis.

Definition 1 (**Feature encoders: $\mathtt{f}_\theta(\cdot)$ and $\mathtt{F}_\theta(\cdot)$**). *We use $\mathtt{f}_\theta(\cdot)$ to denote a content based feature encoder, parameterized by learnable parameters θ, converting node attributes X into latent feature space. This can be achieved by using simple multi-layer perceptron (MLP) or more sophisticated learners, such as CNN or LSTM (for network having image or text as node content). Likewise, $\mathtt{F}_\theta(\cdot)$ denotes graph convolution operators which leverage both node content and network topology to derive latent features. Because feature encoders typically work in a layer-wise manner, we use following notations to denote their propagation between layers.*

$$H^{l-1} \leftarrow \mathtt{f}_\theta(H^{l-1}); \tag{1}$$



(a) 1-layer GCN Embedding (b) 3-layers GCN Embedding (c) 5-layers GCN Embedding



(d) 10-layers GCN Embedding (e) 15-layers GCN Embedding (f) 20-layers GCN Embedding

Fig. 2: Visualization of oversmoothing from GCN embedding learning (Cora network [12]). Nodes are color-coded with same colored nodes belonging to the same class. From top to bottom, left to right, (a) to (f) denotes node embedding from 1-layer GCN to 20-layer GCN, respectively. As GCN layer increases from 1-layer to 5-layer, the embedding achieve better class separability (*i.e.,* better results). As layer continuously, from 10-layers to 20-layers, GCN embedding loss node separability.

$$H^l \leftarrow F_\theta(H^{l-1}, A) : H^1 = X \tag{2}$$

Where Eq. (1) represents encoding node feature only before propagation and Eq. (2) shows the propagation process after feature encoding.

Batch normalization has been proven to be an effective component in deep neural architectures in many fields such as computer vision and natural language process. Inspired by the success of batch normalization [18] and subspace theorem [19], normalization techniques have been proposed to alleviate the oversmoothing in graph neural networks.

Definition 2 (**Normalization operator: $\mathtt{NT}(\cdot)$**). *We use $\mathtt{NT}(\cdot)$ to denote normalization techniques in GNNs, where $\cdot$ input could be learnt embedding H only or combined with topology A for normalization to accommodate graph structure. An example of $\mathtt{NT}(\cdot)$ is the PairNorm method [20] as follows where where H and $\bar{H}$ denote node embedding and its mean, s is a hyperparameter, n is the number of nodes, and $\|(\cdot)\|_2$ denote L2 norm.*

$$\mathtt{NT}(H) = \frac{s\sqrt{n}(H - \bar{H})}{\|H\|_2} \tag{3}$$

Note that $\cdot$ input for $\mathtt{NT}(\cdot)$ could be both X and A. While normalization techniques are different, the principle behind is the same: to preserve Dirichlet energy (an important measure for oversmoothing) [9] or to reduce the variance of the learned embeddings [21].

Table 1: Summary of key symbols and notations.

Notations	Descriptions
$G = (V, E, X)$	An attributed graph with node set (V), edge set (E), node feature content (X)
$A \in \mathbb{R}^{n \times n}$	An adjacency matrix
$\tilde{A}$	An augmented or synthesized adjacency matrix
$\sigma(\cdot)$	A non-linearity activation function
H, H', H''	Graph node embedding, the changing rate of H (first derivative), and the changing rate of H' (second derivative)
$H^l \in \mathbb{R}^{n \times f}$	Graph node feature embedding learned at the l^{th} layer
H_c^l	Graph node feature embedding learned through convolution (or in general any local aggregation)
U	U denotes the velocity of embedding changing (assume graph node feature embedding evolves continuously with time t)
$\mathtt{f}_\theta(\cdot)$	A content-based feature encoder parameterized by θ
$\mathtt{F}_\theta(\cdot)$	A graph convolution operator parameterized by θ
$\mathtt{NT}(\cdot)$	Normalization techniques in GNNs
$\mathtt{LA}(\cdot)$	Layer-wise aggregations
$\mathtt{Aug}(\cdot)$	Topology augmentation function

Definition 3 (**Layer aggregator: $\mathtt{LA}(\cdot)$**). *We use $\mathtt{LA}(\cdot)$ to denote layer-wise aggregations that aggregate embeddings learnt from current and preceding layers. Examples of aggregation include concatenate, max pooling, and LSTM-attention operations [22].*

$$H^l \leftarrow \mathtt{LA}(\cup_{i=1}^{l} H^i) \tag{4}$$

Notice that Eq. (4) often occurs in dense-based approach and can be treated as an ensemble trick over different layers.

Definition 4 (**Topology augmentation operator: $\mathtt{Aug}(\cdot)$**). *We use $\mathtt{Aug}(\cdot)$ to denote topology augmentation function using given input to generate an adjacency matrix $\tilde{A}$. For example $\mathtt{Aug}(H^l, X)$ uses node attributes X and latent features at layer l to generate an adjacency matrix $\tilde{A}$.*

A common choice of $\mathtt{Aug}(\cdot)$ could be symmetric Laplacian, Laplacian, First-order Chebyshev approximation (akin to GCN) following traditional spectral graph theory. Other choices include different random masking schemes such as random edge dropping [23] which is proven to be effective both empirically [23] and theoretically [19], and learnable attention matrix that has the same structure as A (examples include transformer architecture [24] and diffusivity in GRAND [25]).

2.1 Oversmoothing Definition

According to [19], oversmoothing is defined as features exponentially converging to a subspace that is invariant to the propagation matrix A. Assume $M \in \mathbb{R}^{n \times k}$, $k \ll n$ is a subspace invariant to A (or A's augmentation $\mathtt{Aug}(A)$) *i.e.*, for $\forall \Omega \in M$, $A\Omega \in M$ as well. $D_M(H)$ is

defined as the distance between H and its closest element in M, *i.e.*,

$$D_M(H) = \inf_{\Omega \in M} \|H - \Omega\|_F^2 \tag{5}$$

where $\|\cdot\|_F^2$ is the matrix norm. Oversmoothing indicates that $D_M(H^l) \to 0$ exponentially converges, *w.r.t* the increase of layer value l.

Likewise, a node similarity measure μ is defined with two axioms [16]. $\exists c \forall i \in V$ such that $V_i = c$ and $\mu(c) = 0$; $\mu(x + y) \leq \mu(x) + \mu(y)$, *i.e.*, $\mu(\cdot)$ satisfies triangle inequality. Then, oversmoothing is defined below with $\mu(H^l) \to 0$ when $l \to \infty$, where C_1 and C_2 are constants and l is the layer number.

$$\mu(H^l) \leq C_1 e^{-C_2 l} \tag{6}$$

The definition above is similar to the definition in [19] with $\mu(\cdot)$ defined as $D_M(\cdot)$ with the measure decay rate limit to exponential decay. The second definition is often used in diffusion-based system analysis such as [9] with μ as the Dirichlet energy of the system while the first definition is often used in GNN-backbone methods such as EGNN [26] and DropEdge [23].

2.2 Oversmoothing Measures

A commonly used oversmoothing measure is Dirichlet energy which can be defined as:

$$\varepsilon_{\text{DE}}(H^l) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j \in N(i)} \|H_i^l - H_j^l\|_2^2 \tag{7}$$

where H_i^l, H_j^l are feature vectors reflecting nodes L_2 distance with respect to its neighbors. In practice, the specific distance metric can be treated as a learnable parameter or hyperparameter.

Two examples of using this measure include (1) the coefficient selection of EGNN based on the lower bound of $\varepsilon_{\text{DE}}(H^l)$, and (2) G2-gating directly leveraging $\varepsilon_{\text{DE}}(H^l)$ to compute the coefficient for each node and feature channels. We comment here that $\varepsilon_{\text{DE}}(H^l)$ can reflect the current convergence state of the model but cannot accurately guide the model to learn correct local oversmoothing.

Another commonly used measure is Mean Average Distance (MAD) which is defined as:

$$\varepsilon_{\text{MAD}}(H^l) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j \in N(i)} \|1 - \frac{(H_i^l)^T (H_j^l)}{\|H_i^l\| \|H_j^l\|}\|_2^2 \tag{8}$$

where we can observe that MAD simply replaced L_2 distance metric to cosine similarity compared with Eq. (7). Note that MAD is closely related to cosine similarity and therefore only considers the direction of the two embeddings and ignores their magnitude difference. Compared with Dirichlet energy measure, caution on feature magnitude is needed when using the MAD measure.

2.3 Oversmoothing Benchmarks

Table 2 lists commonly used benchmark datasets by existing methods for oversmoothing validation. The datasets are listed and ordered, from left to right, based on the number of times each dataset is used in descending order. Cora, Citeseer, and Pubmed datasets are three common homophilic benchmarks used. Texas, Cornell, Wisconsin are three common heterophilic benchmarks used. Frequently used datasets are all small or medium scale ranging from 10^3 to 10^4 number of nodes. Few methods test shared large-scale heterophilic or homophilic benchmarks over 10^5 number of nodes.

Oversmoothing problem occurs when GNN layers are stacked, aiming to learning global and longer hop-size node relations. For node classification tasks, homophilic graphs assume that nodes intend to share same labels with their nearby neighbors and the heterophilic graphs have more nodes sharing same labels with distant hops away. Heterophilic graphs could show more benefits and improvements for the models with deep GNN layers and reflect the model's ability to learn the global node relationship[15]. Therefore, we recommend that future studies focus more on datasets that are heterophilic for testing the effectiveness of oversmoothing alleviation. Moreover, it can be observed that few large-scale heterophilic datasets are commonly tested by existing methods, resulting a lack of recognition of scalability comparison among existing methods.

3 A Taxonomy for GNN Oversmoothing Alleviation

In this section, we first outline message propagation process commonly used in GNN learning (Sec 3.1), then summarize main themes to tackle oversmoothing (Sec 3.2). After that, we propose a taxonomy for GNN oversmoothing alleviation, as shown in Fig. 3.

3.1 GNN Message Propagation

Deep neural architectures typically require ability to preserve long-term information passing. To achieve the goal, an inter-layer information delivery mechanism is used to regulate information passing process between layers. We briefly separate such processes into the following two subgroups.

3.1.1 Traditional Approaches: Residual vs. Dense Connections

Residual connection and dense connection are two common approaches to achieve inter-layer information passing. Research has shown that such simple architectures can achieve long-term information preservation, and therefore be beneficial to alleviate oversmoothing in general.

To preserve long-term information during deep layer propagation, two types of connections are commonly used in existing works, namely residual connection and dense connection:

Residual Connection in message passing scheme can be defined in Eq. (9) where α, β can be hyperparamter constant but can also be learned.

$$H^l \leftarrow F_\theta(H^{l-1}, A) + \alpha H^{l-1} + \beta H^1 \tag{9}$$

We note that Eq. (9) provides a choice for node embeddings update to preserve part of its original information H^{l-1} and H^1 rather than entirely change to its aggregated message $F_\theta(H^{l-1}, A)$, which typically leads to smooth update.

Dense Connection is defined in Eq. (10). Being dense, it implies that embeddings at the current layer H^l aggregate information from all preceding layers, including $l-1, l-2, \ldots$ and so on.

$$H^l \leftarrow \texttt{LA}(\cup_{i=1}^{l} H^i): \quad H^l = \texttt{F}_\theta(H^{l-1}, A) \tag{10}$$

From model expressive power perspective, because dense connection can be considered as a linear version of residual connection, residual connection is more expressive in general.

3.1.2 Complex Approaches: Dynamics and Recurrence Relation

Recently, physics-informed approaches are proposed to first model the entire graph learning process as a continuous time process (second order partial differential equation PDE or ordinary differential equation ODE) and then use different methods to discretize continuous system, leading to a nuanced recurrence relation different from traditional GNN schemes.

A general physics-informed system (assuming a static graph) can be defined in Eq. (11), where H is the learned node embedding and l is the layer number from the model's perspective or iteration number from the solver's perspective. H' and H'' stand for the changing rate of H (first derivative) and changing rate of H' (second derivative), respectively. A time t variable acting as a continuous feature propagation corresponds to GNN feature propagation with layer l increases.

$$H'' \leftarrow \texttt{F}_\theta(H, H', H'', A, t) \tag{11}$$

Discretizaing Eq. (11) with different discretization schemes induces a recurrence relation similar to residual connection or dense connection but with a more complex structure. An example of discretization could be

$$\begin{aligned}(H^l)' \leftarrow (H^{l-1})' + \beta(\sigma(F_\theta(A, H^{l-1})) \\ -\gamma H^{l-1} - \alpha(H^{l-1})') \end{aligned} \tag{12}$$

$$H^l \leftarrow H^{l-1} + \beta(H^l)' \tag{13}$$

where Eq. (12) and Eq. (13) shows a common form with iterative form for both its first-derivative (embedding update rate) update equation and embedding update equation.

The principle behind the physics-informed system is that energy preserved in the physics system while the system evolving can fit into Dirichlet energy measure and a discretization method therefore keeps Dirichlet energy from exponential decay and alleviates oversmoothing accordingly.

Definition 5 (**Message propagation operator: `Update`$(\cdot)$**). *We use* `Update`$(\cdot)$ *to denote an abstraction of the message propagation process in GNN learning, such as residual connection, dense connection, different recurrence relation, and implicit Euler discretization [25], etc.*

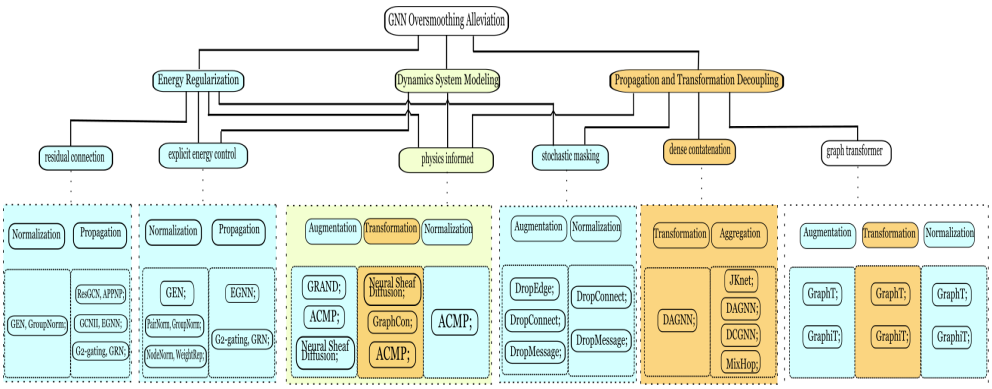


Fig. 3: The proposed taxonomy of oversmooth alleviation approaches for deep graph neural networks. The taxonomy includes three themes and six categories of methods for oversmooth alleviation. Some approaches, such as physics-inform methods, are used by all three themes. Others, such as residual connection, dense concatenation, graph transformer, are only used by a unique theme.

3.2 Themes to Tackle Oversmoothing

To tackle oversmoothing, different design principles have been proposed. The themes behind these approaches are largely driven by modeling iterative GNN learning as energy regularization or as continuous system process. Here, the concept of energy can correspond to any oversmoothing measure or distance metric and we will refer to it as Dirichlet energy for simplicity and consistency unless otherwise specified. We summarize main themes behind existing GNN oversmoothing approaches into following three types. Table 3 lists representative methods and corresponding type of approaches employed to tackle oversmoothing.

3.2.1 Energy Regularization

Initial Energy Regularization:

As defined in Sec 2.1, oversmoothing implies that the whole system energy is exponentially decayed to zero. Stochastic masking based methods provide a simple solution to alleviate the oversmoothing issue with GNN-backbone. The analysis [19] has shown that with a relatively less dense graph, GCN is less likely to suffer from information loss (*i.e.*, oversmoothing). Therefore, DropEdge [23] randomly reducing the density of the graph in the beginning naturally alleviates oversmoothing. Similarly, EGNN uses orthogonal weight initialization [26] to ensure each layer's initial energy is upper bounded at the starting point of training. Both methods are consistent with the analysis [19] that energy is related to both propagation matrix $\texttt{aug}(A)$ and learnable weight W.

Energy Decay Regularization:

Armed with the measure of oversmoothing (Dirichlet energy), existing methods optimize the structure (*e.g.*, GraphCon [9]), coefficient(*e.g.*, G2-gating [15]), and learned features (*e.g.*,

PairNorm [20]) to control the energy of the generated embeddings from decaying exponentially with the layer increases. Such designs provide a theoretical assurance for embeddings to not become oversmooth. However, simply maintaining embedding energy from exponential decay does not necessarily result in a model with good performance. A recent study [16] shows that although G2-gating, GCNII, and GraphCon have similar ability in maintaining embedding energy, as the layer increases, G2-gating enhances its model expressive power (through learned coefficients), resulting in better performance than GCNII and GraphCon. Empirical studies and theoretical analysis are needed to deepen the understanding of a model's capability in maintaining energy *vs.* expressive power.

Methods that prevent energy from decaying exponentially can be divided into three main types: structure preserved, feature normalization preserved, and coefficient preserved. The structure preserved method mostly comes from different combinations of residual connection, dense connection, and discretization of the diffusion equation, leading to different recurrence relations of the Updating function. Feature normalization preserved method includes those normalization techniques that directly apply to features such as PairNorm, NodeNorm, etc. The coefficient preserved type is about how to select the coefficient of each residual component to preserve Dirichlet energy, including EGNN, G2-gating, etc.

3.2.2 Dynamics System Modeling

Instead of regulating the energy decay using normalization or other approaches, an alternative solution is to model the process as a discretized dynamic system with explicit control on system evolving and avoidance of the fixed point convergence. To this end, physics-inspired continuous systems have been leveraged as a starting point for constructing the new family of graph learning structures. The continuous systems equipped with Dirichlet energy are augmented with non-linearity and discretized with different discretization schemes, resulting in complex recurrence relations different from traditional residual and dense-based methods [9]. Different dynamic systems provide rich properties inheriting from their continuous form analysis that traditional GNN do not have.

3.2.3 Propagation and Transformation Decoupling

Oversmoothing is essentially tied to the feature propagation through network topology. Another way of avoid oversmoothing is to decouple the feature learning from feature propagation. Such decoupling can be achieved through two paths: (1) positional-encoding and (2) simple stacking. Positional-encoding-based methods are mostly graph transformers where graph structure information is encoded first and then concatenated with features to feed into the transformer structure. We comment here that this type of method treats the structure as plain feature information and therefore does not involve propagation operation that causes oversmoothing. Therefore, we will not discuss this type of method in detail in this survey. The simple stacking-based method, like SGC [49] and DAGNN[40]), first applies feature transformation without the adjacency matrix being involved and then applies the power of the adjacency matrix to encoded features. The final learned embedding can be summarized into a kernel or diffusion-based adjacency matrix that convolutes with encoded features.

3.3 GNN Oversmoothing Alleviation Taxonomy

Based on the three themes to tackle the oversmoothing in the GNNs, we propose a taxonomy in Fig. 3, which categories all methods into three themes and six subgroups. Under the newly proposed taxonomy, some approaches, such as physics-inform methods, are used by all three themes. Whereas others, such as residual connection, dense concatenation, graph transformer, are only used by a unique theme. The taxonomy helps lay the foundation for ATNPA, the unified view and categorization (Sec 4). In the next session, we will present unified view and categorization, and review representative methods in each category, including their key steps and relation to the ATNPA, as well as their rationality in tackling the oversmoothing challenges.

4 ATNPA: Unified View and Categorization

The three themes to tackle oversmoothing differ significantly in their principles, and such difference is even more profound in respective methods' implementation. To delve into the analysis of these seemly different approaches, a unified view ATNPA with five major steps (Augmentation, Transformation, Normalization, Propagation, and Aggregation) is proposed to help review and understand how different approaches address the oversmoothing.

$$\text{Augmentation:} \quad \tilde{A} \leftarrow \texttt{Aug}(X, A) \tag{14}$$

$$\text{Transformation:} \quad H_c^l \leftarrow \texttt{F}_\theta(H^{l-1}, \tilde{A}) \tag{15}$$

$$\text{Normalization:} \quad H_c^l \leftarrow \texttt{NT}(H_c^l) \tag{16}$$

$$\text{Propagation:} \quad H^l \leftarrow \texttt{NT}(\texttt{Update}(H_c^l, H^{l-1}, H^1)) \tag{17}$$

$$\text{Aggregation:} \quad H^l \leftarrow \texttt{LA}(\cup_{i=1}^{l} H^i) \tag{18}$$

The ATNPA unified view, defined from Eq. (14) to Eq. (18), outlines an abstract-level framework majority GNN methods follow, with all operators being defined in previous sections. Notice that after augmentation as in Eq. (14), it is rare to use the original matrix A unless it is in a self-supervised learning settings with multi-views of graphs utilized [50]. In the following, we categorize all methods into six categorizes, and review each category in details in the succeeding subsections.

4.1 Categorization

Following the three major themes in Sec 3.2, we categorize existing alleviation methods based on critical changes they made compared with the vanilla GNN scheme, and link them to the ATNPA unified view framework. First, we summarize its general properties and show their implicit connections. Then, we discuss specific methods within each category in detail.

Energy Regularization is one major theme we categorize for alleviating oversmoothing. Both Residual connection and Dense concatenation belong to this major theme with its unique property of alleviating energy decay thorough layer updates and they separately focus on adapting the propagation step Eq. (17) and aggregation step Eq. (18). Notice that some of the methods in this category also belongs to dynamic system modeling as the entire layer

stacking process can be modeled as a dynamic system and alleviating oversmoothing can be considered as controlling the energy change in the system.

Residual connection:

Residual-based models explicitly add skip- or residual-connection to the ATNPA's Propagation step at Eq. (17). Examples include APPNP [36], ResGCN [35], GCNII, GEN [37], EGNN, G2-gating, GRN, etc). Initial works, such as ResGCN, APPNP are inspired by residual connection in the computer vision field [51] and only contains either skip-connection H^1 or residual-connection H^{l-1}. GCNII covers both skip-connection H^1 and residual connection H^{l-1} to its update() function and shows better performance. However, its residual coefficient remains hyperparameter. Later, EGNN and G2-gating focus on selection of each residual coefficient under the principle of preserving Dirichlet energy among layers and generalize the residual connection trick to the gating mechanism. GRN further generalizes the constraint of Dirichlet energy to a learnable metric for better adaptation.

Dense Concatenation:

Dense concatenation based methods explicitly aggregate all layer embeddings into the final embeddings, which is reflected in ATNPA's Aggregation step at Eq. (18). Examples include JKnet [22], DenseGCN [41], MixHop [42], Scattering GCN [52] and DAGNN [40].

In addition to the residual-based energy change regularization, many methods focus on explicit energy control directly over the node embeddings learned and this corresponds to our Normalization step at Eq. (16) and Propagation step at Eq. (17).

Energy Control:

Energy-controlled models introduce normalization techniques that control Dirichlet energy or feature variance of the learned embeddings to explicitly optimizing the measure of oversmoothing and alleviate oversmoothing. This is reflected at ATNPA's Normalization step at Eq. (16) and Propagation step at Eq. (17). Examples include EGNN, PairNorm, NodeNorm [21], GroupNorm [38], G2-gating.

While the two main branches above focus on regularizing the energy change thorough layer stacking, another branch of alleviating oversmoothing for energy controlling energy is to modify the initial energy (corresponding to regularize the initial condition in terms of dynamic system). This type of methods often focus on modifying our ATNPA's Augmentation step at Eq. (14).

Stochastic Masking:

Stochastic-mask (random-mask) based models randomly mask or drop edges/nodes of the original graph, corresponding to changes in ATNPA's Augmentation step at Eq. (14), and then use resulted stochastic graph for propagation. Examples include DropEdge, Drop-connect [43], DropMessage [44].

Combined from the two principles stated from with either boundary condition control and system update control (corresponding to a complete solver for PDE and ODE system), inspired from dynamic system modeling in physics, a branch of method focus on explicit construct continuous ODE or PDE related to graph diffusion and discretize it. As solving ODE or PDE requires both boundary condition (initial condition) and update equation, this

type of methods falls into our ATNPA's Augmentation step at Eq. (14) and Propagation step at Eq. (17).

Physics-inform Process:

Physics-informed methods first model a continuous ODE or PDE related to graph diffusion equation and then discretize the continuous equation with different discretization methods. This leads to nuanced recurrence relation and possibly a combined propagation matrix learnt from both features and topology, which corresponds to ATNPA's Augmentation step at Eq. (14) and Propagation step at Eq. (17). Examples include GraphCon, GRAND [25], ACMP [46], Neural Sheaf Diffusion [47].

Finally, inspired by recently emerging transformer techniques in sequence generation problems, which can learn long-context sequence relationship, an ensemble of capturing both local relation with shallow layer GNN learning and global relation learning with transformers are studied. It is also worthnoting that transformer structure can be treated as sequence propagating on a dynamic complete graph.

Graph Transformer:

Transformer-based methods integrate transformer structure into GNN backbones and leverage different combination or integration to allow models to learn both long-term relation (from transformer capacity) and local relation (from GNN capacity). A recent study [53] categorizes transformer-type models into three types: (1) Graph auxiliary Type (GA) such as GraphTrans [54] and GraphBert [55], (2) positional encoder type (PE) such as Graphormer [56], and (3) improved attention matrix from graph (AT) such as GraphiT [48] and graphT [24]. Among the three types, PE can be considered as a decoupling of feature and topology learning, and AT types mostly resemble to GNN backbones to alleviate oversmoothing. As a result, these approaches are reflected in ATNPA's Augmentation and Transformation steps.

4.2 Residual Connection Methods

Early example of residual-based deep GNN method is APPNP [36] and GCNII [17], which are inspired from image field residual architecture [51]. APPNP can be summarized as (assuming f_θ as a one-layer MLP):

$$\tilde{A} \leftarrow \texttt{Aug}(A): \quad H^1 \leftarrow \sigma(XW) \tag{19}$$

$$H^l \leftarrow (1-\alpha)\tilde{A}H^{l-1} + \alpha H^1 \tag{20}$$

Notice that Eq. (20) focus solely on preserve original feature information (skip-connection) without residual connection and no non-linearity function is used for feature update.

GCNII can be summarized as:

$$\tilde{A} \leftarrow \texttt{Aug}(A): \quad H^1 \leftarrow \sigma(AXW^1) \tag{21}$$

$$H^l \leftarrow \sigma(\tilde{A}(\alpha H^{l-1} + (1-\alpha)H^1)(\beta I + (1-\beta)W^l)) \tag{22}$$

where its update equation Eq. (22) combines both residual connection and its skip connection. The residual coefficient remains to be hyperparameter.

GEN is an extension of GCNII method and can be summarized as:

$$H^l \leftarrow \mathtt{F}_\theta(H^{l-1} + H_c^l): \quad H_c^l \leftarrow s \cdot \|H^{l-1}\|_2 \frac{H_c^l}{\|H_c^l\|_2} \tag{23}$$

where the regularization is inspired from batch normalization [18] and it empirically works very well.

EGNN [26] uses a slightly more complex structure that includes both skip connection and residual connection:

$$H^l \leftarrow \sigma(((1 - c_{min})AH^{l-1} + \alpha H^{l-1} + \beta H^1)W^l) \tag{24}$$

where $\alpha + \beta = c_{min}$ and c_{min} is a positive hyperparameter chosen to satisfy the lower bound of initial Dirichlet energy. Its main motivation is to choose an appropriate c_{min} which is an lower bound of initial Dirichlet energy to keep the Dirichlet energy in a controllable range during propagation. Therefore, EGNN is an explicit energy preserving technique compared with implicit energy control by physics-informed methods.

Similar to EGNN, G2-gating uses simple residual GCN as a backbone with the form

$$H^l \leftarrow (1 - \sigma(\varepsilon_{DE}(F_\theta(H^{l-1}, A)))H^{l-1} + \sigma(\varepsilon_{DE}(F_\theta(H^{l-1}, A)))H^{l-1}F_\theta(H^{l-1}, A) \tag{25}$$

G2-gating's main contribution to oversmoothing lies on its controllable message dropping mechanism similar to DropMessage method. G2-gating drops message after message aggregation while DropMessage drop messages before message aggregation. Therefore G2-gating has a more controllable way to preserve Dirichlet energy. We will discuss G2-gating's message dropping mechanism, GEN and EGNN's normalization technique and coefficient selection in the energy-based model in details.

Discussion:

Note that all above methods fit into ATNPA's unified view by making changes to the Propagation (Eq. (17)) steps. In general, residual based methods have been primarily focused on learning coefficients for each component (*i.e.*, coefficients for H^l, H^{l-1}, or $F_\theta(H^{l-1}, A)$ *etc.*). Nevertheless, there is insufficient study and theoretical analysis about the order of each residual components in terms of their position *w.r.t* activation function $\sigma(\cdot)$. To date, GEN is the only work that empirically validated that order they proposed works better than GCNII.

4.3 Dense Concatenation Methods

Existing dense-based methods include JKnet, DAGNN, and DCGCN [41]. While Mixhop and Scattering GCN do not explicitly show oversmoothing benefits, they have a similar structure as DCGCN except that the aggregation is performed on fixed multi-hop embeddings instead of previous embeddings. JKnet provides different options over the final aggregation for layer embeddings. Here we consider the concatenate version aligned with DAGNN. Final

embeddings learnt from JKnet$_{cat}$ can be summarized as:

$$H^L \leftarrow \sum_{i=1}^{L} c_i H^i : \quad H^l \leftarrow \sigma(AH^{l-1}W^l) \ \& \ H^1 \leftarrow X \tag{26}$$

where the aggregation stage is only applied to the final layer and with concatenation aggregation followed by projection. This can be described as the summation of each layer embeddings with a weight c_i learned from the projection layer, as defined in Eq. (26). For DAGNN, final embeddings can be summarized as (assuming one layer MLP in the beginning):

$$H^L \leftarrow \sum_{i=1}^{L} c_i H^i : \quad H^l \leftarrow AH^{l-1} \ \& \ H^1 \leftarrow \sigma(XW) \tag{27}$$

It can be observed that the two dense-based methods share similar final aggregation scheme (*i.e.*, final embedding can be considered as a linear combination of layer embeddings). Yet, the embedding learnt in the intermediate layers is different. JKnet$_{cat}$ still includes learnable parameters in the middle and keep non-linearity while DAGNN removes both parts. Without non-linearity and learnable paramters, DAGNN essentially becomes a diffusion kernel based method similar to [57]. We can see that both methods fit into ATNPA's unified view in Transformation Eq. (15) and Aggregation Eq. (18), which is the key component for dense connection based method.

Instead of applying dense connection only to the final layer aggregation, DCGCN introduces layer aggregation at every layer in a recurrence style:

$$H^l \leftarrow F_\theta(\cup_{i=1}^{l-1} H^l, A) \tag{28}$$

which is still a variant of ATNPA's Aggregation (Eq. (15)) with a slight difference in the order of aggregation before convolution instead of after convolution.

Discussion:

We note that dense-connection can be considered as an extension of residual connection with all previous embedding being used rather than only the initial embedding or previous layer embedding. Both dense-based methods and residual-based methods can be treated as attempts of positioning residual components at different locations. However, there is a lack of theoretical analysis and empirical study comparing the two types of methods in general. It is easy to observe that ignoring non-linearity, both methods can be explained in a Markov Random Walk framework [49]. Nevertheless, we shall point out that non-linearity is an important component for increasing model capacity and expressive power in terms of deep layers and therefore should not be discarded in analysis.

4.4 Stochastic-masking Methods

Randomly dropping edges or nodes is commonly considered as an augmentation technique to avoid overfitting. It has been shown that random edge dropping is also beneficial for over-smoothing alleviation [23], where the key step is to randomly generate the adjacency matrix

with a subset of edges from original edges and obtain the masked adjacency $\tilde{A}$ by

$$\tilde{A} \leftarrow \texttt{Aug}(A): \quad \texttt{Aug}(A) = \texttt{Bern}(p) \odot A \tag{29}$$

where $\texttt{Bern}(p)$ is a matrix filled with Bernoulli distribution elements and p controls the drop rate. The motivation behind edge dropping is the subspace theorem [19] that indicates less connected graph leading to slow convergence of oversmoothing state. The random-mask modification fits into ATNPA's Augmentation step Eq. (14).

DropConnect generalizes DropEdge to edges of each feature channels instead of edges of all features. Specifically, DropConnect create different random masked adjacency matrix $\tilde{A}$ for each features instead of one shared random masked adjacency matrix for all features.

Similar to DropConnect, DropMessage [44] has recently been proposed to unify different masking methods including edge dropping, node dropping, and Dropout. Its key modification is:

$$H^l \leftarrow A\tilde{H}^{l-1}W: \quad \tilde{H}^{l-1} = H^{l-1} \odot Bern(p) \tag{30}$$

where $Bern(p)$ is a feature matrix filled with Bernoulli distribution elements and p controls the drop rate. Note that $\tilde{H}^{l-1}$ becomes a random variable matrix and each time $\tilde{H}^{l-1}_{ij}$ is accessed during matrix production, it will be randomized.

Discussion:

DropEdge prefers a shared masked adjacency matrix throughout layers instead of layer wise masking as empirically a layer-wise variant has the risk of overfitting and have additional computation cost. Additionally, Dropout method is complementary to DropEdge and applying both of them is beneficial to the model performance [23]. DropMessage unified them together and show theoretical that message dropping techniques increase Shannon Entropy of propagated message compared with dropping edges, nodes or features alone, which alleviates oversmoothing. Compared with DropEdge, DropConnect which change augmentation step in ATNPA's unified view. DropMessage can be considered as combining augmentation and normalization steps in ATNPA's unified view.

4.5 Energy Control Methods

Energy-based methods share common motivation of controlling generated embeddings in each layer with constraints on either preserving Dirichlet energy or reducing feature variance. Examples of preserving Dirichlet energy include EGNN, G2-gating, PairNorm, GroupNorm, while NodeNorm reduce feature variance and WeightRep directly reparameterize the learnable parameters to allow independence of input features .

Compared with GCNII randomly searching coefficient α, β for each residual component, EGNN [26] explicitly limits the coefficient searching to satisfy the lower bound of the initial Dirichlet energy and control the initialized Dirichlet energy by orthogonal weight initialization. However, the coefficient is still a scalar shared for each node and feature channels and is determined by fine tuning hyperparameters. G2-gating [15] provides a way of computing coefficients according to the graph gradient, which is essentially the Dirichlet energy and uses the gating mechanism to control features that tend to converge to stop updating and therefore

avoid treating coefficient as hyperparameter. In addition, G2-gating generalizes scalar coefficient to a matrix coefficient in the shape of embedding matrix, providing fine-grained energy control. Assuming that ideal embedding is that all the nodes sharing the same labels converge to the same embedding (*i.e.*, locally oversmooth) while across labels, node embeddings should be different (*i.e.*, large Dirichlet energy). The gating mechanism prevents node embeddings from converging globally but also limit the local oversmoothing. Therefore, G2-gating method produces only sub-optimal solutions.

Unlike G2-gating and EGNN that have a residual-GNN backbone, PairNorm [20] normalizes the feature matrix X directly according to Eq. (3) without requiring a residual component. The theoretical analysis is based on SGC which ignores non-linearity. Similar to EGNN, PairNorm's main idea is to keep the underlying distance (such as total pairwise distance) the same, before *vs.* after the layer propagation. Empirically, PairNorm alleviates oversmoothing issue but its peroformance does not improve with layer increasing. The author suggests that PairNorm may not be beneficial to standard dataset such as Cora and need a more nuanced setting (*i.e.*, missing features), whereas other methods have shown performance gain in the standard dataset. A potential reason behind PairNorm's performance degradation, *w.r.t* layer increasing, is that the normalization used in PairNorm results in less expressive power for models and therefore cannot perform well, as suggested by [16].

GroupNorm [38] uses a simple residual-GNN backbone similar to G2-gating. Unlike G2-gating focusing on determining proper coefficients, GroupNorm normalizes the features by first assigning nodes to groups (*i.e.*, clustering) and then normalizes nodes within groups to push nodes within clusters to locally oversmooth. Empirically, GroupNorm reports the results of miss features settings to validate the performance gain, which shows similar problem as in PairNorm, suggesting that normalization techniques seem to weaken the expressive power of the models with layers increasing in general.

WeightRep [45] suggests that learning a proper weight can avoid oversmoothing at any layer and dynamically construct an input-dependent weight and show both empirically and theoretically that such weight can avoid oversmoothing. However, it is observed that performance degradation persists after deep layers, suggesting other factors that might cause the performance degradation.

Discussion:

We note that both normalization and coefficient computation approaches fit into `ATNPA`'s unified view in Normalization Eq. (16) and Propagation Eq. (17). Direct normalization on features such as mean substraction and variance shifting empirically reduce model capacity and expressive power while coefficients learning seem to be a more promising direction to not only keep Dirichlet energy but also preserve model expressive power.

4.6 Physics-inform Process

Physics-informed methods consider GNN learning as a continuous system and derive solutions by formulating the system's evolving as a model propagation process. In this context, the time component t in continuous system corresponds to GNN based model's layer concept. Different discretization methods provide a complex family of methods indicated by a continuous system and most GNN based backbone can be considered as an explicit Euler

discretization (only considering the recurrence relation or the Update function in GNN framework) [25]. Examples of continuous systems include GRAND, GraphCon, ACMP, Neural Sheaf Diffusion, and G2-gating (It was first reviewed as GNN backbones, but is also related to the continuous system).

GRAND [25] leverages graph diffusion PDE equations as the continuous system and performs both explicit and implicit Euler discretization. The diffusivity is modeled with an attention structure Eq. (31) related to node features and edges Eq. (32):

$$\texttt{Aug}(X) \leftarrow \sigma(\frac{(KX)^T(QX)}{d_k}) \tag{31}$$

$$\texttt{Aug}(X, A) \leftarrow (\texttt{Aug}(X) > \epsilon) \odot A \tag{32}$$

where $\sigma(\cdot)$ is a non-linearity activation softmax function. K and Q are learnable parameters, and d_k is the hidden dimension for K which is used as normalization. ϵ is a threshold value to sparsify attention matrix $\texttt{Aug}(X)$ and $\odot$ denotes element-wise multiplication. Eq. (31) is the diffusion variant and Eq. (32) is rewired variants for GRAND. With both discretization, the key component fits into ATNPA's unified view in Augmentation Eq. (14).

Similar to GRAND, ACMP [46] modifies the graph diffusion equation to a particle interaction system. It generalizes GRAND's Aug() in Eq. (32) by adding a negative constant to the attention weight so that the attention could be negative. This allows the nodes to not only attract but also repulse each other through learning. Additionally, to control the upper bound Dirichlet energy, a well-shaped function (called double-well potential) is added as a regularization (equivalent to feature normalization) to avoid infinite Dirichlet energy growth. It fits into ATNPA's Augmentation and Normalization steps, despite a very different origin (particle system interpretation).

GraphCon [9] leverages a graph dynamic system of non-linear ODEs:

$$U' \leftarrow F_\theta(A, H, t) - \gamma H - \alpha U \tag{33}$$

$$H' \leftarrow U \tag{34}$$

where H' is the first order derivative with respect to time t (a default setting at physics) and U' is equivalent to H''. After discretization, t is essentially equivalent to layer l in GNN backbones. Following IMEX (implicit-explicit) time discretization [58], GraphCon obtains a new recurrence:

$$\begin{aligned} U^n \leftarrow U^{n-1} &+ \Delta(t)(\sigma(F_\theta(A, H^{n-1}, t^{n-1})) \\ &- \gamma H^{n-1} - \alpha U^{n-1}) \end{aligned} \tag{35}$$

$$H^n \leftarrow H^{n-1} + \Delta(t) U^n \tag{36}$$

where $\Delta(t)$ is the discretization step.

Discussion:

Diffusion systems above share a common point of establishing a connection between the feature changing rate H' and the graph gradient $\sum_{j \in Neighbor(i)} |h_i - h_j|$ which is Dirichlet energy for one node. A discretization of the system then provides a unique complex recurrence relation that preserves established connections.

The niche of diffusion-based methods stem from the design that the system preserves Dirichlet energy (mitigates oversmoothing) through the complex residual recurrence structure, avoiding fixed point convergence at exponential rate and small perturbation deviates the fixed point away in the GraphCon case. This makes diffusion-based method unique, compared with other works that preserve energy through explicit feature value control or coefficient control.

Neural Sheaf diffusion is an approach using cellular sheaf theory to model evolving of the features at each layer and the geometry of the graph [47]. The augmentation to Sheaf Diffusion, similar to GCN augmentation, constructs a continuous differential equation as:

$$(H^t)' \leftarrow -\sigma(\texttt{Aug}_\theta(A, H^t)W_1^t H^t W_2^t) \tag{37}$$

where H^t unlike common feature matrix with dimension $n \times d$ with d as hidden dimension, each node feature is vertically stacked and H^t is of dimension $nd \times 1$. $\texttt{Aug}_\theta(A, H^t)$ also produces an $nd \times nd$ matrix with $n \times n$ subblocks of dimension $d \times d$. The discrete version of Eq. (37) becomes

$$(H^t) \leftarrow H^{t-1} - \sigma(\texttt{Aug}_\theta(A, H^{t-1})W_1^{t-1} H^{t-1} W_2^{t-1}) \tag{38}$$

Discussion:

We comment here that the extra nd dimensions provide each feature channel with a possibly different propagation channel compared with the original settings where the propagation channel binds to the node level. The idea behind is similar to G2-gating where they also have a multi-rate coefficient matrix to control the update fine-grained to each feature of each node instead of each node. Another point about Sheaf Diffusion is that they use shared weight among each block, *i.e.* W_1^{t-1} can be decomposed as the Kronecker product of the Identity matrix and a learnable $W_1^{'}$ with dimension $d \times d$ and therefore reduce the exponential number of parameter increase, which in term indicates an assumption that one feature correlation is shared among graph topology.

4.7 Transformer-based Methods

Transformer has shown superior performance in long-term relation learning [59]. GNN has been proven to be effective on local relation learning and performance deteriorates when both global and local relation exists (*i.e.*, graphs with middle homophily scores [60]. Combining transformer and GNN architecture has been used to capture both long and short-term relations and improve model performance on heterophilous graphs. With the connection between heterophily and oversmoothing [14], we consider transformer-based methods candidates for alleviating oversmoothing.

There are mainly three types of approaches according to the position of the two components, PE (positional encoding), GA (graph auxiliary), and AT (attention matrix from graph).

Admittedly, many graph transformers simultaneously use several techniques. To understand the role each part plays in the learning process, we will discuss each component individually and fit them into the proposed framework. PE-type can be roughly considered as projecting certain graph properties to feature space and then aggregating the projected graph features with node features. The aggregated feature is then fed into the transformer block. A general form of PE for one transformer block is therefore:

$$X \leftarrow \texttt{Transformer}(X, \tilde{A}): \quad \tilde{A} \leftarrow \texttt{Aug}(A) \tag{39}$$

Discussion about PA:

Unlike normal graph convolution, $\texttt{Transformer}()$ can be considered as a complex feature transformation, where X and $\tilde{A}$ are not convoluted but are processed in a transformer style. Because it avoids convolution directly, it can considered as a decoupled feature and topology learning. Theoretical analysis of model expressive power between transformer and graph convolution is lacking in existing research and potential analysis is necessary to justify the learnability of such structure.

Discussion about GA:

As GA type stacks transformer block with graph convolution block, making it hard to analyze and the concept of oversmoothing becomes vague in this case. Briefly speaking, we can treat the transformer block as a complex feature Transformation or Normalization steps of ATNPA. Then GA-type transformer can be treated as a common GNN framework with complex normalization applied in the middle. Since the transformer does not ensure the energy of learned node embeddings, the GA-based component will not necessarily help alleviate oversmoothing.

AT-type graph transformers, such as GraphT and GraphiT, have unique $\texttt{Aug}(\cdot)$ components, defined in Eq. (40) for GraphT and Eq. (41) for GraphiT, which share a striking similarity to attention-based diffusivity structures, such as GRAND and ACMP.

$$\texttt{Aug}(X, A) \leftarrow \sigma(\frac{(XQ)(XK)^T}{d_k} \odot A) \tag{40}$$

$$\texttt{Aug}(X, A) \leftarrow \sigma(\frac{(XQ)(XK)^T}{d_k} \odot \kappa(A)) \tag{41}$$

where σ is the softmax nonlinear activation function, d_k is the hidden dimension of X, and $\kappa(A) \in \mathbb{R}^{n \times n}$ denotes a transformation of A, such as graph Laplacian.

Discussion about AT:

The main difference between GraphT Eq. (40) and GRAND Eq. (32) is the location of the non-linearity activation function $\sigma(\cdot)$. The similarity between GRAND and transformer-based methods comes from the diffusivity modeled as an attention structure in GRAND and the diffusivity can fit into the $\texttt{Aug}(\cdot)$ Augmentation step in ATNPA. AT-type methods can be treated as a graph rewiring approach. Since the rewired graph will be more sparse compared with the original graph and the rewired process can be done in each layer, we can consider

them as an extension of a controllable masking mechanism aiming to increase the energy at each layer.

5 Experimental Results Reported From Existing Methods Under Same Settings

To verify the effectiveness of existing methods, we report each method's best report results under similar/same settings on public datasets including three most frequently used homophilic and three heterophilic datasets. The results are shown in Table 4. For each dataset, the best performance is bold-faced, and the second best performance is in italic format. In the table, we also highlight four methods with bold fonts that achieve significant performance boost over either homophilic datasets and heterophilic datasets, including GCNII, EGNN, G2-gating, and GRN. It is worth noting that all methods are residual based with three methods focus on limiting residual coefficients based on dirichlet energy constraint or learnable metric constraint. This result suggests a promising direction following the principle of energy change regularization over propagation and residual-based GNN structure. Additionally, we observe that GRAND also shows good performance gain with its physics-informed structure, suggesting the future of dynamic system based approaches.

6 Conclusion

In this paper, we reviewed and analyzed GNN oversmoothing alleviation methods. We argued that despite of dramatic differences in their design principles and math formulations, existing approaches share three common themes in their motivations to tackle oversmoothing, and the commonality allows us to summarize them into six categories. To allow in-depth understanding and analysis of all methods, we proposed `ATNPA`, which uses five steps to distill properties and architectures of existing methods and shows that existing oversmoothing alleviation methods are variants by introducing changes to one or multiple steps of the `ATNPA`. The unified view allows a clear understanding on how oversmoothing is alleviated for individual methods, strength and weakness of each type of methods, and possible future study directions. We drew discussion and remarks on representative methods, and observed that despite many methods focusing on constraining energy of the learned embeddings, diffusion-based methods use a physics-inspired structure to keep energy, while residual-based methods use a simple structure but focus on tuning coefficient or directly applying normalization to features to preserve energy. In addition, the modeling of network propagation has evolved from a static topology to dynamically learn topology, or to seek respective adjacency matrix for each feature instead of one shared topology for all features.

The paper summarizes three main themes according to the underlying principles. With the theme dynamic system modeling combine two principles behind energy regularization and propagation and transformation decoupling. It is possible that our paper may overlook some works that are not explicitly focused on oversmoothing. Nevertheless, we are confident that reviewed methods cover major oversmoothing alleviation branches and the underling principles leveraged are coherent to the problem and are discussed to the best of our knowledge.

From a high abstract level, we highlight that oversmoothing alleviation is equivalent to dynamic system control with constraints. It is therefore suggested in the future studies to focus on two main principles for system controlling: energy initialization or boundary condition control and energy change control thorough propagation. Moreover, our `ATNPA` view provides an excellent component-wise separation for future researchers to modify according to the highlighted two principles. Practically, we notice that existing methods lack a unified comparisons for large-scale benchmark on both heterophilic and homophilic datasets. Scalability of existing methods could therefore become an important future study for real-world applications.

Acknowledgements. This work has been supported in part by the US National Science Foundation (NSF) under Grant Nos. IIS-2236579, IIS-2302786, and IOS-2430224.

Declarations

- Funding: This work has been supported in part by the US National Science Foundation (NSF) under Grant Nos. IIS-2236579, IIS-2302786, and IOS-2430224.
- Conflict of interest/Competing interests: The authors have no conflicts of interest to declare that are relevant to the content of this article.
- Ethics approval and consent to participate: N/A
- Consent for publication:
- Data availability:
- Materials availability:
- Code availability:
- Author contribution: Y. Jin and X. Zhu made equal contribution to the article.

References

[1] Gori, M., Monfardini, G., Scarselli, F.: A new model for learning in graph domains. In: IEEE International Joint Conference on Neural Networks (IJCNN), vol. 2, pp. 729–7342 (2005)

[2] Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. In: International Conference on Learning Representations (ICLR) (2014)

[3] Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: 30th International Conference on Neural Information Processing Systems (NIPS) (2016)

[4] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR, Toulon, France, April 24-26, 2017, Conference Track Proceedings (2017). https://openreview.net/forum?id=SJU4ayYgl

[5] Zhang, D., Yin, J., Zhu, X., Zhang, C.: Network representation learning: A survey. IEEE Transactions on Big Data **6**, 3–28 (2017)

[6] Zhu, J., Rossi, R.A., Rao, A., Mai, T., Lipka, N., Ahmed, N.K., Koutra, D.: Graph neural networks with heterophily. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI) (2021)

[7] Li, Q., Han, Z., Wu, X.-M.: Deeper insights into graph convolutional networks for semi-supervised learning. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI) (2018)

[8] NT, H., Maehara, T.: Revisiting graph neural networks: All we have is low-pass filters. ArXiv **abs/1905.09550** (2019)

[9] Rusch, T.K., Chamberlain, B.P., Rowbottom, J.R., Mishra, S., Bronstein, M.M.: Graph-coupled oscillator networks. In: International Conference on Machine Learning (2022). https://api.semanticscholar.org/CorpusID:246608169

[10] Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., Sun, X.: Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In: Proceddings of the 34th AAAI Conference on Artificial Intelligence (AAAI-20) (2019)

[11] Rozemberczki, B., Kiss, O., Sarkar, R.: Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In: Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), pp. 3125–3132 (2020). ACM

[12] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine **29**(3), 93 (2008) https://doi.org/10.1609/aimag.v29i3.2157

[13] Alon, U., Yahav, E.: On the bottleneck of graph neural networks and its practical implications. In: International Conference on Learning Representations (ICLR) (2021). https://openreview.net/forum?id=i80OPhOCVH2

[14] Yan, Y., Hashemi, M., Swersky, K., Yang, Y., Koutra, D.: Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. 2022 IEEE International Conference on Data Mining (ICDM), 1287–1292 (2021)

[15] Rusch, T.K., Chamberlain, B.P., Mahoney, M.W., Bronstein, M.M., Mishra, S.: Gradient gating for deep multi-rate learning on graphs. In: International Conference on Learning Representations (2023)

[16] Rusch, T.K., Bronstein, M.M., Mishra, S.: A survey on oversmoothing in graph neural networks. arXiv:2303.10993 (2023) [cs.LG]

[17] Chen, M., Wei, Z., Huang, Z., Ding, B., Li, Y.: Simple and deep graph convolutional

networks. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 1725–1735 (2020). https://proceedings.mlr.press/v119/chen20v.html

[18] Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. ICML'15, pp. 448–456 (2015)

[19] Oono, K., Suzuki, T.: Graph neural networks exponentially lose expressive power for node classification. In: International Conference on Learning Representations (ICLR) (2020). https://openreview.net/forum?id=S1ldO2EFPr

[20] Zhao, L., Akoglu, L.: Pairnorm: Tackling oversmoothing in gnns. In: International Conference on Learning Representations (ICLR) (2020)

[21] Zhou, K., Dong, Y., Wang, K., Lee, W.S., Hooi, B., Xu, H., Feng, J.: Understanding and resolving performance degradation in deep graph convolutional networks. Proceedings of the 30th ACM International Conference on Information & Knowledge Management (2020)

[22] Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 5453–5462 (2018). https://proceedings.mlr.press/v80/xu18c.html

[23] Rong, Y., Huang, W., Xu, T., Huang, J.: Dropedge: Towards deep graph convolutional networks on node classification. In: International Conference on Learning Representations (2020). https://openreview.net/forum?id=Hkx1qkrKPr

[24] Dwivedi, V.P., Bresson, X.: A generalization of transformer networks to graphs. AAAI Workshop on Deep Learning on Graphs: Methods and Applications (2021)

[25] Chamberlain, B.P., Rowbottom, J., Gorinova, M.I., Webb, S.D., Rossi, E., Bronstein, M.M.: GRAND: Graph neural diffusion. In: The Symbiosis of Deep Learning and Differential Equations (2021). https://openreview.net/forum?id=_1fu_cjsaRE

[26] Zhou, K., Huang, X., Zha, D., Chen, R., Li, L., Choi, S.-H., Hu, X.: Dirichlet energy constrained learning for deep graph neural networks. Advances in neural information processing systems (2021)

[27] Yang, Z., Cohen, W.W., Salakhutdinov, R.: Revisiting semi-supervised learning with graph embeddings. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48. ICML'16, pp. 40–48. JMLR.org, ??? (2016)

[28] Pei, H., Wei, B., Chang, K.C.-C., Lei, Y., Yang, B.: Geom-gcn: Geometric graph convolutional networks. In: International Conference on Learning Representations (2020). https://openreview.net/forum?id=S1e2agrFvS

[29] Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of Graph Neural Network Evaluation (2019). https://arxiv.org/abs/1811.05868

[30] Rozemberczki, B., Allen, C., Sarkar, R.: Multi-scale attributed node embedding. Journal of Complex Networks **9**(2), 014 (2021) https://doi.org/10.1093/comnet/cnab014 https://academic.oup.com/comnet/article-pdf/9/2/cnab014/40435146/cnab014.pdf

[31] Wang, K., Shen, Z., Huang, C., Wu, C.-H., Dong, Y., Kanakia, A.: Microsoft academic graph: When experts are not enough. Quantitative Science Studies **1**(1), 396–413 (2020) https://doi.org/10.1162/qss_a_00021 https://direct.mit.edu/qss/article-pdf/1/1/396/1760880/qss_a_00021.pdf

[32] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Burges, C.J., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 26. Curran Associates, Inc., ??? (2013). https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

[33] McCallum, A., Nigam, K., Rennie, J.D.M., Seymore, K.: Automating the construction of internet portals with machine learning. Information Retrieval **3**, 127–163 (2000)

[34] Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17, pp. 1025–1035. Curran Associates Inc., Red Hook, NY, USA (2017)

[35] Li, G., Müller, M., Thabet, A., Ghanem, B.: Deepgcns: Can gcns go as deep as cnns? In: The IEEE International Conference on Computer Vision (ICCV) (2019)

[36] Klicpera, J., Bojchevski, A., Günnemann, S.: Predict then propagate: Graph neural networks meet personalized pagerank. In: International Conference on Learning Representations (2018). https://api.semanticscholar.org/CorpusID:67855539

[37] Li, G., Xiong, C., Thabet, A.K., Ghanem, B.: Deepergcn: All you need to train deeper gcns. ArXiv **abs/2006.07739** (2020)

[38] Zhou, K., Huang, X., Li, Y., Zha, D., Chen, R., Hu, X.: Towards deeper graph neural networks with differentiable group normalization. In: Advances in Neural Information Processing Systems (2020)

[39] Jin, Y., Zhu, X.: Graph Rhythm Network: Beyond Energy Modeling for Deep Graph Neural Networks . In: 2024 IEEE International Conference on Data Mining (ICDM), pp. 723–728. IEEE Computer Society, Los Alamitos, CA, USA (2024).

https://doi.org/10.1109/ICDM59182.2024.00083 . https://doi.ieeecomputersociety.org/10.1109/ICDM59182.2024.00083

[40] Liu, M., Gao, H., Ji, S.: Towards deeper graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2020). ACM

[41] Guo1, Z., Zhang, Y., Teng, Z., Lu, W.: Densely connected graph convolutional networks for graph-to-sequence learning. Transactions of the Association for Computational Linguistics **7**, 297–312 (2019)

[42] Abu-El-Haija, S., Perozzi, B., Kapoor, A., Harutyunyan, H., Alipourfard, N., Lerman, K., Steeg, G.V., Galstyan, A.: Mixhop: Higher-order graph convolution architectures via sparsified neighborhood mixing. In: International Conference on Machine Learning (ICML) (2019)

[43] Hasanzadeh, A., Hajiramezanali, E., Boluki, S., Zhou, M., Duffield, N., Narayanan, K., Qian, X.: Bayesian graph neural networks with adaptive connection sampling. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 4094–4104 (2020)

[44] Fang, T., Xiao, Z., Wang, C., Xu, J., Yang, X., Yang, Y.: Dropmessage: Unifying random dropping for graph neural networks. Proceedings of the AAAI Conference on Artificial Intelligence **37**(4), 4267–4275 (2023) https://doi.org/10.1609/aaai.v37i4.25545

[45] Zhuo, Z., Wang, Y., Ma, J., Wang, Y.: Graph neural networks (with proper weights) can escape oversmoothing. In: The 16th Asian Conference on Machine Learning (Conference Track) (2024). https://openreview.net/forum?id=SWHuXfasuW

[46] Wang, Y., Yi, K., Liu, X., Wang, Y.G., Jin, S.: ACMP: Allen-cahn message passing with attractive and repulsive forces for graph neural networks. In: The Eleventh International Conference on Learning Representations (2023). https://openreview.net/forum?id=4fZc_79Lrqs

[47] Bodnar, C., Giovanni, F.D., Chamberlain, B.P., Liò, P., Bronstein, M.M.: Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. In: 36th Conferenceon Neural Information Processing Systems (NeurIPS). (2022)

[48] Mialon, G., Chen, D., Selosse, M., Mairal, J.: Graphit: Encoding graph structure in transformers. arXiv:2106.05667 (2021) [cs.LG]

[49] Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., Weinberger, K.: Simplifying graph convolutional networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 6861–6871 (2019). https://proceedings.mlr.press/v97/wu19e.html

[50] Jin, Y., Zhu, X.: Predictive masking for semi-supervised graph contrastive learning. In: 2022 IEEE International Conference on Big Data (Big Data), pp. 1266–1271 (2022). https://doi.org/10.1109/BigData55660.2022.10020970

[51] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778 (2015)

[52] Min, Y., Wenke, F., Wolf, G.: Scattering gcn: Overcoming oversmoothness in graph convolutional networks. In: Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS2020) (2020)

[53] Min, E., Chen, R., Bian, Y., Xu, T., Zhao, K., Huang, W., Zhao, P., Huang, J., Ananiadou, S., Rong, Y.: Transformer for graphs: An overview from architecture perspective. ArXiv **abs/2202.08455** (2022)

[54] Wu, Z., Jain, P., Wright, M., Mirhoseini, A., Gonzalez, J.E., Stoica, I.: Representing long-range context for graph neural networks with global attention. In: Advances in Neural Information Processing Systems (NeurIPS) (2021)

[55] Zhang, J., Zhang, H., Xia, C., Sun, L.: Graph-bert: Only attention is needed for learning graph representations. arXiv preprint arXiv:2001.05140 (2020)

[56] Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., Liu, T.-Y.: Do transformers really perform badly for graph representation? In: Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems (2021). https://openreview.net/forum?id=OeWooOxFwDa

[57] Gasteiger, J., Weißenberger, S., Günnemann, S.: Diffusion improves graph learning. In: Conference on Neural Information Processing Systems (NeurIPS) (2019)

[58] Hairer, E., Norsett, S., Wanner, G.: Solving Ordinary Differential Equations I: Nonstiff Problems vol. 8, (1993). https://doi.org/10.1007/978-3-540-78862-1

[59] Rong, Y., Bian, Y., Xu, T., Xie, W., Wei, Y., Huang, W., Huang, J.: Self-supervised graph transformer on large-scale molecular data. Advances in Neural Information Processing Systems **33** (2020)

[60] Luan, S., Hua, C., Xu, M., Lu, Q., Zhu, J., Chang, X.-W., Fu, J., Leskovec, J., Precup, D.: When do graph neural networks help with node classification? investigating the homophily principle on node distinguishability. In: Thirty-seventh Conference on Neural Information Processing Systems (2023). https://openreview.net/forum?id=kJmYu3Ti2z

Table 2: Common dataset used for existing methods. Only datasets that are used at least twice are collected. Year reflect the time that the method is proposed. # of Total Usage indicates the number of times the dataset is used for existing method.

Methods/Datasets	Year	Cora [27]	Citeseer [27]	Pubmed [27]	Texas [28]	Cornell [28]	Wisconsin [28]	CouathorCS [29]	Coauthors-Physics [29]	Chameleon [30]	Amazon Photos [29]	Squirrel [30]	ogbn-arxiv [31] [32]	Amazon Computers [29]	Film [28]	Cora-ML [33]	Reddit [34]
APPNP	2018		✓	✓												✓	
GCNII	2020	✓	✓	✓	✓	✓	✓			✓							
GEN	2020												✓				
GroupNorm	2020	✓	✓	✓				✓									
EGNN	2021	✓		✓					✓				✓				
G2-gating	2023				✓	✓	✓			✓		✓			✓		
GRN	2024	✓	✓	✓	✓	✓	✓			✓		✓			✓		
JKnet	2018	✓	✓														✓
MixHop	2019	✓	✓	✓													
DAGNN	2020	✓	✓	✓				✓	✓		✓			✓			
DropEdge	2020	✓	✓	✓													✓
DropConnect	2020	✓	✓													✓	
DropMessage	2023	✓	✓	✓									✓				
PairNorm	2020	✓	✓	✓				✓									
NodeNorm	2020	✓	✓	✓	✓	✓	✓	✓			✓						
WeightRep	2024	✓	✓	✓				✓	✓	✓	✓	✓		✓			
GRAND	2021	✓	✓	✓					✓		✓		✓	✓			
Neural Sheaf Diffusion	2022	✓	✓	✓	✓	✓	✓			✓		✓			✓		
GraphCon	2022	✓	✓	✓	✓	✓	✓										
ACMP	2023	✓	✓	✓	✓	✓	✓		✓		✓			✓			
# of Total Usage	\	17	17	16	7	7	7	5	5	5	5	4	4	4	3	2	2
# of Nodes	\	2708	3327	19717	183	183	251	18333	34493	2277	7650	5201	169343	13752	7600	2810	232965

Table 3: A summary of representative methods *w.r.t* their categorization and properties in tackling oversmoothing.

Methods	Category	Energy(Rewiring)	Energy(Nomalization)	Energy(Coefficient)	Energy(Initialization)	Decoupling	Dynamics	Residual	Dense
ResGCN [35]	residual connection			✓				✓	
APPNP [36]	residual connection			✓		✓		✓	
GCNII [17]	residual connection			✓				✓	
GEN [37]	residual connection/energy control		✓					✓	
EGNN [26]	residual connection/energy control			✓	✓			✓	
GroupNorm [38]	residual connection/energy control		✓					✓	
GRN [39]	residual connection/energy control		✓					✓	
G2-gating [15]	residual connection/energy control/physics-inform			✓			✓	✓	
JKnet [22]	dense concatenation								✓
DAGNN [40]	dense concatenation					✓			✓
DCGNN [41]	dense concatenation								✓
MixHop [42]	dense concatenation								✓
DropEdge [23]	stochastic masking	✓			✓				
DropConnect [43]	stochastic masking	✓	✓		✓				
DropMessage [44]	residual connection/stochastic masking	✓	✓		✓			✓	
PairNorm [20]	energy control		✓						
NodeNorm [21]	energy control		✓						
WeightRep [45]	energy control		✓						
GRAND [25]	physics-inform	✓			✓		✓		
GraphCon [9]	physics-inform						✓		
ACMP [46]	physics-inform	✓	✓	✓			✓		
Neural Sheaf Diffusion [47]	physics-inform	✓					✓		
GraphT[24]/GraphiT [48]	graph transformer	✓			✓			✓	

Table 4: Collected average results from each method following the same setting for six most frequently used datasets with three homophilic and three heterophilic datasets. For the Planetoid dataset (Cora, Citeseer, Pubmed), NA indicates missing reports. Some of the results are reported by other method's paper as its original paper has a different split setting. Bold font indicates best results and italic font indicate second best results.

Method/Dataset	Year	Cora	Citeseer	Pubmed	Texas	Cornell	Wisconsin
APPNP	2018	83.3	71.8	80.1	65.41	73.51	69.02
GCNII	2020	*85.5*	**73.4**	80.2	77.84	76.49	81.57
GroupNorm	2020	82	69.5	79.5	NA	NA	NA
EGNN	2021	**85.7**	NA	80.1	NA	NA	NA
G2-gating	2023	NA	NA	NA	*87.57*	**87.3**	*87.65*
GRN	2024	NA	NA	NA	**89.73**	*86.22*	**88.4**
JKnet	2018	83.3	72.6	79.2	57.3	61.08	50.59
MixHop	2019	81.9	71.4	**80.8**	77.83	73.51	75.88
DAGNN	2020	84.4	73.3	80.5	NA	NA	NA
DropEdge	2020	81.69	71.43	79.06	NA	NA	NA
DropConnect	2020	82.2	71.72	NA	NA	NA	NA
DropMessage	2023	83.33	71.83	79.2	NA	NA	NA
PairNorm	2020	82.1	69.6	78	60.27	58.92	48.43
NodeNorm	2020	83	72.9	*80.7*	78.92	80.54	83.14
WeightRep	2024	82.21	69.04	77.68	NA	NA	NA
GRAND	2021	84.7	73.3	80.4	NA	NA	NA