ATNPA: A Unified View of Oversmoothing Alleviation in Graph Neural Networks

Yufei Jin and Xingquan Zhu
Dept. of Electrical Engineering & Computer Science, Florida Atlantic University, FL-33431, USA
{yjin2021, xzhu3}@fau.edu

Abstract

Oversmoothing is a commonly observed challenge in graph neural network (GNN) learning, where, as layers increase, embedding features learned from GNNs quickly become similar/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 to harness 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 their difference 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 outline three themes to tackle oversmoothing, and then separate all methods into six categories, followed by detailed reviews of representative methods, including their relation to the ATNPA, and discussion about their niche, strength, and weakness. The review not only draws in-depth understanding of existing methods in the field, but also shows a clear road map for future study.

1 Introduction

Graph Neural Networks (GNN) [1, 2, 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 and 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]. GNN models equipped with oversmoothing alleviation can in general accommodate more GNN layers and therefore allow nodes to have a larger receptive fields [9]. 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 [10, 11].

Oversmoothing can be described as a phenomenon that all node mebeddings, after deep GNN layers, become similar to each other. Several measures such as Dirichlet energy [12] and Mean Average Distances (MAD) have been proposed to quantify the extent of oversmoothing of a model [13]. In this paper, unless otherwise specified, Dirichlet energy will be used as the major measure of oversmoothing for analysis.

Preprint. Under review.

Majority theoretical works studying when oversmoothing occurs are based on dynamic systems. One important work [14] uses subspace theorem to analyze when GCN becomes oversmoothing asymptotically (*i.e.*, infinite layers behavior). This theory motivates many alleviation approaches, such as constrain-based and random edge dropping such as EGNN [15] and edge-drop [16]. Another work [12] uses perturbation theory to analyze how Dirichlet energy (energy of the dynamic system) behaves under different coupled systems [12]. This theory mostly serves as basis for dynamic-based alleviation approaches, such as graphCON [12] and G2-gating [11]. Recently, another theoretical work [17] suggests that graph attention network (GAT) [18] also suffers from oversmoothing. The first theory guides many constrain-based work and random edge dropping such as EGNN [15], edge-drop [16] while the second work mostly serves dynamic-based models such as graphCON, G2-gating [11].

Indeed, many works have been proposed to tackle oversmothing in GNNs, by using different types of design principles. For example, energy based approaches aim to increase initial energy or keep energy from exponential decay during propagation in GNNs. On the other hand, 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 [19] has compared several methods, and pointed out that some of the existing methods (such as GCNII [20], GraphCon [12]) 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 [19] 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.

In this paper, we propose to unify existing methods to the same form and point out their potential connection in tackling GNN oversmoothing. Our study not only provides a unified view, `ATNPA` to summarize all methods using common math formulations, but also separate them into six groups, by taking their unique designs into consideration. The survey outlines differences between methods in each groups, 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.

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 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.*

$$H^l \leftarrow A^{l-1} f_\theta(X); \tag{1}$$

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

Where Eq. (1) can be considered as a linear Graph Convolution as proposed by [21] and Eq. (2) is a common graph convolution propagation.

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 [22] and subspace theorem [14], normalization techniques have been proposed to alleviate the oversmoothing in graph neural networks.

Definition 2 (Normalization operator: $\texttt{NT}(\cdot)$). *We use $NT(\cdot)$ 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 $NT(\cdot)$ is the PairNorm method [23] 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.*

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

Note that $\cdot$ input for $\texttt{NT}(\cdot)$ 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) [12] or to reduce the variance of the learned embeddings [24].

Definition 3 (Layer aggregator: $\texttt{LA}(\cdot)$). *We use $LA(\cdot)$ 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 [25]. We use $\cup$ to denote the input is a union set of embeddings from all layers.*

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

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

A common choice of $\texttt{Aug}(\cdot)$ could be symmetric Laplacian, Laplacian, First-order Chebshev approximation (akin to GCN) following traditional spectral graph theory. Other choices include different random masking schemes such as random edge dropping [16] which is proven to be effective both empirically [16] and theoretically [14], and learnable attention matrix that has the same structure as A (examples include transformer architecture [26] and diffusivity in GRAND [27]).

2.1 Oversmoothing Definition

According to [14], 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 $\texttt{Aug}(A)$) *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.*,

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

Oversmoothing indicates that $D_M(H^l) \rightarrow 0$ exponentially converges, *w.r.t* the increase of layer value l.

Likewise, a node similarity measure μ is defined with two axioms [19]. $\exists c \forall i \in V$ 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 $\mu(H^l) \rightarrow 0$ when $l \rightarrow \infty$, 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 [14] with $\mu(\cdot)$ defined as $D_M(\cdot)$. The second definition is often used in diffusion-based system analysis such as [12] with μ as the Dirichlet energy of the system while the first definition is often used in GNN-backbone methods such as EGNN [15] and DropEdge [16].

2.2 Oversmoothing Measures

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

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

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:

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

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 Model Expressive Power

Model expressive power and oversmoothing have intrinsic connections. Under this assumption, models with strong express power may be more resilient to oversmoothing. Therefore, enhancing model expressive power seems to be also helpful for oversmoothing alleviation.

In the graph neural network domain, existing research on model expressive power is primarily focused on comparing GNNs' separation ability with Weisfeiler-Lehman (WL) graph isomorphism test [28]. Theoretical analysis for existing studies is built upon two important concepts: (1) multi-set; or (2) permutation invariant injective aggregation function [29] [30]. For the former, the neighbor set along with the center node is formed as a multi-set, on which the expressive power is then defined accordingly. For the latter, aggregation function should be injective so that non-isomorphic graphs can be mapped to unique embeddings and permutation invariant property guarantees that isomorphic graphs (which can be obtained by permutation group action) can be mapped to embeddings belonging to the same equivalent class.

While graph isomorphism test focuses on graph-level classification, we can consider node-level classification as a local subgraph differentiation test. For a graph G_i, we use $\mathbb{P}(G_i)$ to denote the set of all graphs isomorphic to G_i. For one L layer GNN $F_\theta(\cdot) : G \to R^d$, if two graphs G_1, G_2 expanded from two target nodes (*i.e.* each graph includes target node and its L hop neighbors) are non-isomorphic, we can define an ε_{MEP} score in Eq. (10) to quantify model expressive power over node separation ability:

$$\varepsilon_{\text{base}}(G_i) = \sup_{G\in\mathbb{P}(G_i)} ||F_\theta(G_i) - F_\theta(G)||_2^2 \tag{9}$$

$$\varepsilon_{\text{MEP}} = \frac{||F_\theta(G_1) - F_\theta(G_2)||_2^2}{\max(\varepsilon_{\text{base}}(G_1), \varepsilon_{\text{base}}(G_2)) + \epsilon} \tag{10}$$

where ϵ is a scalar ensuring that the denominator does not equal to zero, which happens when $F_\theta(\cdot)$ is permutation invariant.

Overall, Eq. (10) defines that for non-isomorphic graphs G_1 and G_2, the larger the ε_{MEP} score of the model, the better its expressive power is in separating G_1 and G_2. Meanwhile, because $F_\theta(G_1)$ denotes embedding of graph G_1, oversmoothing would imply that $F_\theta(G_1) \approx F_\theta(G_2)$, resulting in a small ε_{MEP} score for non-isomorphic graphs.

Assuming a model has an infinitely strong power of differentiating node up to L-hops away, it implies that the model can avoid oversmoothing up to L-layers. This seems to suggest that oversmoothing is a side effect brought by the model lacking expressive power in separating nodes (including node and its local neighborhood). A recent study [31] provides a positive evidence towards this hypothesis by showing that increasing the node expressive power through local structure aware GNN can help alleviate oversmoothing issue.

Methods	Category	Energy (Rewiring)	Energy (Nomalization)	Energy (Coefficient)	Energy (Initialization)	Decoupling	Dynamics	Dense/Residual
ResGCN [32]	residual-based			✓				✓
APPNP [33]	residual-based			✓		✓		✓
GCNII [20]	residual-based			✓				✓
GEN [34]	residual-/energy-based		✓					✓
EGNN [15]	residual-/energy-based			✓	✓			✓
GroupNorm [35]	residual-/energy-based		✓					✓
G2-gating [11]	residual-/energy-/diffusion-based			✓			✓	✓
JKnet [25]	dense-based							✓
DAGNN [36]	dense-based					✓		✓
DCGCN [37]	dense-based							✓
MixHop [38]	dense-based							✓
DropEdge [16]	random-mask based	✓			✓			
DropConnect [39]	random-mask based	✓	✓		✓			
DropMessage [40]	residual-/random-mask based	✓	✓		✓			✓
PairNorm [23]	energy-based		✓					
NodeNorm [24]	energy-based		✓					
GRAND [27]	diffusion-based	✓			✓		✓	
GraphCon [12]	diffusion-based						✓	
ACMP [41]	diffusion-based	✓	✓	✓			✓	
Neural Sheaf Diffusion [42]	diffusion-based	✓					✓	
GraphT [26]	transformer-based	✓			✓			✓
GraphiT [43]	transformer-based	✓			✓			✓

Table 1: A summary of representative methods *w.r.t* their categorization and properties in tackling oversmoothing.

3 ATNPA: A Unified View of Oversmoothing Alleviation

In this session, we first outline message propagation process commonly used in GNN learning (Sec 3.1), then summarize main themes to tackle oversmoothing (Sec 3.2), which help lay the foundation for ATNPA, the unified view and categorization (Sec 3.3). Secs 3.4 and beyond 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.

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.

Residual Connection in message passing scheme can be defined in Eq. (11) where α, β can be hyperparamter constant but can also be learned.

$$H^l \leftarrow F_\theta(H^{l-1}, A) + \alpha H^{l-1} + \beta H^1 \tag{11}$$

Dense connection is defined in Eq. (12). Being dense, it implies that embeddings at the current layer H^l aggregate information from all preceding layers, including $l-1, l-2, \dots$ and so on.

$$H^l \leftarrow \mathtt{LA}(\cup_{i=1}^{l} H^i): \quad H^l = \mathtt{F}_\theta(H^{l-1}, A) \tag{12}$$

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, diffusion-based 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 graph diffusion system (assuming a static graph) can be defined in Eq. (13), 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.

$$H'' \leftarrow \mathtt{F}_\theta(H, H', H'', A, t) \tag{13}$$

Discretizing Eq. (13) with different discretization schemes induces a recurrence relation similar to residual-based or dense-based connection but with a more complex structure. An example of discretization could be

$$(H^l)' \leftarrow (H^{l-1})' + \beta(\sigma(F_\theta(A, H^{l-1})) - \gamma H^{l-1} - \alpha(H^{l-1})') \tag{14}$$

$$H^l \leftarrow H^{l-1} + \beta(H^l)' \tag{15}$$

The principle behind the diffusion-based 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 [27], etc.*

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 1 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. Random-mask based methods provide a simple solution to alleviate the oversmoothing issue with GNN-backbone. The analysis [14] has shown that with a relatively less dense graph, GCN is less likely to suffer information loss (*i.e.*, oversmoothing). Therefore, DropEdge [16] randomly reducing the density of the graph in the beginning naturally alleviates oversmoothing. Similarly, EGNN uses orthogonal weight initialization [15] to ensure each layer's initial energy is upper bounded at the starting point of training. Both methods are consistent with the analysis [14] 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 [12]), coefficient(*e.g.*, G2-gating [11]), and learned features (*e.g.*, PairNorm [23]) 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 [19] 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.

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 how system evolves and avoid the fixed point convergence exponentially. Accordingly, 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 [12]. 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 [21] and DAGNN [36]), 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 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 seemingly 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.

$$\text{Augmentation:} \quad \tilde{A} \leftarrow \texttt{Aug}(X, A) \tag{16}$$

$$\text{Transformation:} \quad H_c^l \leftarrow \texttt{F}_\theta(H^{l-1}, \tilde{A}) \tag{17}$$

$$\text{Normalization:} \quad H_c^l \leftarrow \texttt{NT}(H_c^l) \tag{18}$$

$$\text{Propagation:} \quad H^l \leftarrow \texttt{NT}(\texttt{Update}(H_c^l, H^{l-1}, H^1)) \tag{19}$$

$$\text{Aggregation:} \quad H^l \leftarrow \texttt{LA}(\cup_{i=1}^l H^i) \tag{20}$$

The ATNPA unified view, defined from Eq. (16) to Eq. (20), outlines an abstract-level framework majority GNN methods follow, with all operators being defined in previous sections. In the following, we categorize all methods into six categorizes, and review each category in details in the succeeding subsections.

3.3.1 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.

Residual-based: Residual-based models explicitly add skip- or residual-connection to the ATNPA's Propagation step at Eq. (19). Examples include APPNP [33], ResGCN [32], GCNII, GEN [34], EGNN, G2-gating, etc). Initial works, such as ResGCN, GCNII, are inspired by residual connection in the computer vision field [44]. Later, EGNN and G2-gating focus on improving the coefficient of each residual component under the principle of preserving Dirichlet energy among layers.

Dense-based: Dense-based methods explicitly aggregate all layer embeddings into the final embeddings, which is reflected in ATNPA's Aggregation step at Eq. (20). Examples include JKnet [25], DCGCN [37], MixHop [38], Scattering GCN [45] and DAGNN [36].

Random-mask based: Random-mask based models randomly mask or drop edges/nodes of the original graph, corresponding to changes in ATNPA's Augmentation step at Eq. (16), and then use resulted stochastic graph for propagation. Examples include DropEdge, Drop-connect [39], DropMessage [40].

Energy-based: Energy-based 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. (18)

and Propagation step at Eq. (19). Examples include EGNN, PairNorm, NodeNorm [24], GroupNorm [35], G2-gating.

Diffusion-based: Diffusion-based 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. (16) and Propagation step at Eq. (19). Examples include GraphCon, GRAND [27], ACMP [41], Neural Sheaf Diffusion [42].

Transformer-based: 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 [46] categorizes transformer-type models into three types: (1) Graph auxiliary Type (GA) such as GraphTrans [47] and GraphBert [48], (2) positional encoder type (PE) such as Graphormer [49], and (3) improved attention matrix from graph (AT) such as GraphiT [43] and graphT [26]. 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.

3.4 Residual-based Methods

Early example of residual-based deep GNN method is APPNP [33] and GCNII [20], which are inspired from image field residual architecture [44]. APPNP can be summarized as (assuming f_θ as a one-layer MLP):

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

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

GCNII can be summarized as:

$$\tilde{A} \leftarrow \mathtt{Aug}(A): \quad H^1 \leftarrow \sigma(AXW^1) \tag{23}$$

$$H^l \leftarrow \sigma(\tilde{A}(\alpha H^{l-1} + (1-\alpha)H^1)(\beta I + (1-\beta)W^l)) \tag{24}$$

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

$$H^l \leftarrow \mathtt{F}_\theta(H^{l-1} + H_c^l): \quad H_c^l \leftarrow s \cdot \|H^{l-1}\|_2 \frac{H_c^l}{\|H_c^l\|_2} \tag{25}$$

where the regularization is inspired from batch normalization [22] and it empirically works very well.

EGNN [15] uses a slightly more complex structure that includes both skip connection and residual connection:

$$H^l \leftarrow \sigma(((1-c_{min})AH^{l-1} + \alpha H^{l-1} + \beta H^1)W^l) \tag{26}$$

where $\alpha + \beta = c_{min}$ and c_{min} is a positive hyperparamter chosen to satify 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 diffusion-based methods.

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

$$H^l \leftarrow H^{l-1} + F_\theta(H^{l-1}, A) \tag{27}$$

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(Residual Connection): Note that all above methods fit into ATNPA's unified view by making changes to the Normalization (Eq. 18) and Propagation (Eq. 19) 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.

3.5 Dense-based Methods

Existing dense-based methods include JKnet, DAGNN, and DCGCN [37]. 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:

$$H^L \leftarrow \sum_{i=1}^{L} c_i H^i : \quad H^l \leftarrow \sigma(AH^{l-1}W^l) \;\&\; H^1 \leftarrow X \tag{28}$$

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. (28). For DAGNN, final embeddings can be summarized as (assuming one layer MLP in the beginning):

$$H^L \leftarrow \sum_{i=1}^{L} c_i H^i : \quad H^l \leftarrow AH^{l-1} \;\&\; H^1 \leftarrow \sigma(XW) \tag{29}$$

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 [50]. We can see that both methods fit into ATNPA's unified view in Transformation (Eq. 17) and Aggregation (Eq. 20), 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:

$$H^l \leftarrow F_\theta(\cup_{i=1}^{l-1} H^l, A) \tag{30}$$

which is still a variant of ATNPA's Aggregation (Eq. 20) with a slight difference in the order of aggregation before convolution instead of after convolution.

Discussion(Dense Connection): 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 [21]. 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.

3.6 Random-mask based 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 oversmoothing alleviation [16], 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

$$\tilde{A} \leftarrow \texttt{Aug}(A) : \quad \texttt{Aug}(A) = \texttt{Bern}(p) \odot A \tag{31}$$

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 [14] that indicates less connected graph leading to slow convergence of oversmoothing state. The random-mask modification fits into ATNPA's Augmentation step (Eq. 16).

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

$$H^l \leftarrow A\tilde{H}^{l-1}W : \;\; \tilde{H}^{l-1} = H^{l-1} \odot Bern(p) \tag{32}$$

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(Dropping): 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 [16]. 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.

3.7 Energy-based 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.

Compared with GCNII randomly searching coefficient α, β for each residual component, EGNN [15] 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 [11] 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 [23] 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 [19].

GroupNorm [35] 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.

Discussion(Normalization): We note that both normalization and coefficient computation approaches fit into ATNPA's unified view in Normalization (Eq. 18) and Propagation (Eq. 19). 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.

3.8 Diffusion-based Methods

Diffusion-based 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) [27]. 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 [27] 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. 33) related to node features and edges (Eq. 34):

$$\texttt{Aug}(X) \leftarrow \sigma(\frac{(KX)^T(QX)}{d_k}) \tag{33}$$

$$\texttt{Aug}(X, A) \leftarrow (\texttt{Aug}(X) > \epsilon) \odot A \tag{34}$$

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. (33) is the diffusion variant and Eq. (34) is rewired variants for GRAND. With both discretization, the key component fits into ATNPA's unified view in Augmentation Eq. (16).

Similar to GRAND, ACMP [41] modifies the graph diffusion equation to a particle interaction system. It generalizes GRAND's Aug() in Eq. (34) 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 [12] leverages a graph dynamic system of non-linear ODEs:

$$Y' \leftarrow F_\theta(A, H, t) - \gamma H - \alpha Y \tag{35}$$

$$H' \leftarrow Y \tag{36}$$

where H' is the first order derivative with respect to time t (a default setting at physics) and Y' is equivalent to H''. After discretization, t is essentially equivalent to layer l in GNN backbones. Following IMEX (implicit-explicit) time discretization [51], GraphCon obtains a new recurrence:

$$\begin{aligned} Y^n \leftarrow Y^{n-1} + \Delta(t)(\sigma(F_\theta(A, H^{n-1}, t^{n-1})) \\ - \gamma H^{n-1} - \alpha Y^{n-1}) \end{aligned} \tag{37}$$

$$H^n \leftarrow H^{n-1} + \Delta(t) Y^n \tag{38}$$

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

Discussion:(Continuous System) 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 [42]. The augmentation to Sheaf Diffusion, similar to GCN augmentation, constructs a continuous differential equation as:

$$(H^t)' \leftarrow -\sigma(\texttt{Aug}_\theta(A, H^t) W_1^t H^t W_2^t) \tag{39}$$

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. (39) becomes

$$(H^t) \leftarrow H^{t-1} - \sigma(\texttt{Aug}_\theta(A, H^{t-1}) W_1^{t-1} H^{t-1} W_2^{t-1}) \tag{40}$$

Discussion(Neural Sheaf Diffusion): 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.

3.9 Transformer-based Methods

Transformer has shown superior performance in long-term relation learning [52]. 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 [53]. 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 [10], 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:

$$X \leftarrow \texttt{Transformer}(X, \tilde{A}) : \quad \tilde{A} \leftarrow \texttt{Aug}(A) \tag{41}$$

Discussion (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 (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. (42) for GraphT and Eq. (43) for GraphiT, which share a striking similarity to attention-based diffusivity structures, such as GRAND and ACMP.

$$\texttt{Aug}(X, A) \leftarrow \sigma(\frac{(XQ)(XK)^T}{d_k} \odot A) \tag{42}$$

$$\texttt{Aug}(X, A) \leftarrow \sigma(\frac{(XQ)(XK)^T}{d_k} \odot \kappa(A)) \tag{43}$$

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 (AT): The main difference between GraphT (Eq. 42) and GRAND (Eq. 34) 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.

4 Conclusion

In this paper, we reviewed and analyzed existing 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 such 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. Such a 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.

References

[1] M. Gori, G. Monfardini, and F. Scarselli, “A new model for learning in graph domains,” in *IEEE International Joint Conference on Neural Networks (IJCNN)*, vol. 2, pp. 729–734 vol. 2, 2005.

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

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

[4] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

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

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

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

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

[9] U. Alon and E. Yahav, “On the bottleneck of graph neural networks and its practical implications,” in *International Conference on Learning Representations (ICLR)*, 2021.

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

[11] T. K. Rusch, B. P. Chamberlain, M. W. Mahoney, M. M. Bronstein, and S. Mishra, “Gradient gating for deep multi-rate learning on graphs,” in *International Conference on Learning Representations*, 2023.

[12] T. K. Rusch, B. P. Chamberlain, J. R. Rowbottom, S. Mishra, and M. M. Bronstein, “Graph-coupled oscillator networks,” in *International Conference on Machine Learning*, 2022.

[13] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, “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.

[14] K. Oono and T. Suzuki, “Graph neural networks exponentially lose expressive power for node classification,” in *International Conference on Learning Representations (ICLR)*, 2020.

[15] K. Zhou, X. Huang, D. Zha, R. Chen, L. Li, S.-H. Choi, and X. Hu, “Dirichlet energy constrained learning for deep graph neural networks,” *Advances in neural information processing systems*, 2021.

[16] Y. Rong, W. Huang, T. Xu, and J. Huang, “Dropedge: Towards deep graph convolutional networks on node classification,” in *International Conference on Learning Representations*, 2020.

[17] X. Wu, A. Ajorlou, Z. Wu, and A. Jadbabaie, “Demystifying oversmoothing in attention-based graph neural networks,” in *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[18] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in *International Conference on Learning Representations*, 2018.

[19] T. K. Rusch, M. M. Bronstein, and S. Mishra, “A survey on oversmoothing in graph neural networks,” *arXiv:2303.10993*, 2023.

[20] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, “Simple and deep graph convolutional networks,” in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 1725–1735, PMLR, 13–18 Jul 2020.

[21] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, “Simplifying graph convolutional networks,” in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 6861–6871, PMLR, 09–15 Jun 2019.

[22] S. Ioffe and C. Szegedy, "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, p. 448–456, JMLR.org, 2015.

[23] L. Zhao and L. Akoglu, "Pairnorm: Tackling oversmoothing in gnns," in *International Conference on Learning Representations (ICLR)*, 2020.

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

[25] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 5453–5462, PMLR, 10–15 Jul 2018.

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

[27] B. P. Chamberlain, J. Rowbottom, M. I. Gorinova, S. D. Webb, E. Rossi, and M. M. Bronstein, "GRAND: Graph neural diffusion," in *The Symbiosis of Deep Learning and Differential Equations*, 2021.

[28] A. Leman, "The reduction of a graph to canonical form and the algebra which appears therein," 2018.

[29] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *International Conference on Learning Representations*, 2019.

[30] Z. Chen, S. Villar, L. Chen, and J. Bruna, "On the equivalence between graph isomorphism testing and function approximation with gnns," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[31] A. Wijesinghe and Q. Wang, "A new perspective on "how graph neural networks go beyond weisfeiler-lehman?"," in *International Conference on Learning Representations*, 2022.

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

[33] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *International Conference on Learning Representations*, 2018.

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

[35] K. Zhou, X. Huang, Y. Li, D. Zha, R. Chen, and X. Hu, "Towards deeper graph neural networks with differentiable group normalization," in *Advances in neural information processing systems*, 2020.

[36] M. Liu, H. Gao, and S. Ji, "Towards deeper graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, 2020.

[37] Z. Guo1, Y. Zhang, Z. Teng, and W. Lu, "Densely connected graph convolutional networks for graph-to-sequence learning," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 297–312, 2019.

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

[39] A. Hasanzadeh, E. Hajiramezanali, S. Boluki, M. Zhou, N. Duffield, K. Narayanan, and X. Qian, "Bayesian graph neural networks with adaptive connection sampling," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 4094–4104, PMLR, 13–18 Jul 2020.

[40] T. Fang, Z. Xiao, C. Wang, J. Xu, X. Yang, and Y. Yang, "Dropmessage: Unifying random dropping for graph neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, p. 4267–4275, June 2023.

[41] Y. Wang, K. Yi, X. Liu, Y. G. Wang, and S. Jin, "ACMP: Allen-cahn message passing with attractive and repulsive forces for graph neural networks," in *The Eleventh International Conference on Learning Representations*, 2023.

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

[43] G. Mialon, D. Chen, M. Selosse, and J. Mairal, "Graphit: Encoding graph structure in transformers," *arXiv:2106.05667*, 2021.

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

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

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

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

[48] J. Zhang, H. Zhang, C. Xia, and L. Sun, "Graph-bert: Only attention is needed for learning graph representations," *arXiv preprint arXiv:2001.05140*, 2020.

[49] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu, "Do transformers really perform badly for graph representation?," in *Advances in Neural Information Processing Systems* (A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds.), 2021.

[50] J. Gasteiger, S. Weißenberger, and S. Günnemann, "Diffusion improves graph learning," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[51] E. Hairer, S. Norsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, vol. 8. 01 1993.

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

[53] S. Luan, C. Hua, Q. Lu, J. Zhu, M. Zhao, S. Zhang, X.-W. Chang, and D. Precup, "Is heterophily a real nightmare for graph neural networks to do node classification?," *arXiv:2109.05641*, 2021.