Analysis of Graph Neural Networks with Theory of Markov Chains

Weichen Zhao
Chenguang Wang
Congying Han∗
Tiande Guo
University of Chinese Academy of Sciences (UCAS)
Key Laboratory of Big Data Mining and Knowledge Management, CAS
Beijing, China

Abstract
In this paper, we provide a theoretical tool for the interpretation and analysis of graph neural networks (GNNs). We use Markov chains on graphs to mathematically model the forward propagation processes of GNNs. The graph neural networks are divided into two classes of operator-consistent and operator-inconsistent based on whether the Markov chains are time-homogeneous. Based on this, we study over-smoothing which is an important problem in GNN research. We attribute the over-smoothing problem to the convergence of an arbitrary initial distribution to a stationary distribution. We prove the effectiveness of the previous methods for alleviating the over-smoothing problem. Further, we give the conclusion that operator-consistent GNN cannot avoid over-smoothing at an exponential rate in the Markovian sense. For operator-inconsistent GNN, we theoretically give a sufficient condition for avoiding over-smoothing. Based on this condition, we propose a regularization term which can be flexibly added to the training of the neural network. Finally, we design experiments to verify the effectiveness of this condition. Results show that our proposed sufficient condition not only improves the performance but also alleviates the over-smoothing phenomenon.

Keywords: Graph neural networks, Markov chains, Markov chains in random environments, over-smoothing, mixing time

1. Introduction
Graph neural networks (Kipf and Welling, 2016; Bruna et al., 2013; Defferrard et al., 2016; Veličković et al., 2018; Abu-El-Haija et al., 2018; Zhang et al., 2018; Lee et al., 2018; Klicpera et al., 2018) have achieved great success in processing graph data which is rich in information about the relationships between objects, and have been successfully applied to chemistry (Do et al., 2019; Kearnes et al., 2016; De Cao and Kipf, 2018; Gilmer et al., 2017), traffic prediction (Cui et al., 2019; Li et al., 2019 Kumar et al., 2019), knowledge graph (Park et al., 2019; Wang et al., 2019b), social network (Deng et al., 2019; Qiu et al., 2018), recommendation system (Ying et al., 2018) and other aspects.

∗. Corresponding author at: No.19A, Yuquan Road, Shijingshan District, Beijing.
As deep learning develops, the explanation from the mathematical perspective is highly valued. This facilitates a deeper understanding of deep neural networks and promotes its development. In recent years, the rapid development of graph neural network research has been accompanied by some problems that show more demand for mathematical explanations. Many scholars analyze GNNs using mathematical tools, including Weisfeiler-Lehman tests (Xu et al., 2018; Maron et al., 2019; Azizian et al., 2020), spectral analysis (Nt and Maehara, 2019; Li et al., 2018; Wu et al., 2019) and dynamical system (Oono and Suzuki, 2019). However, like other deep neural networks, theoretical analysis related to GNNs is still scarce. Different perspectives are needed to expand scholars’ understanding and promote the development of GNNs.

The deepening of the network has brought about changes in neural networks and caused a boom in deep learning. Unlike typical deep neural networks, there is an important problem in the training of graph neural networks that affects the further deepening of the network. Researchers have found that the performance of some graph neural networks decreases instead as the depth of the model increases. Most scholars attribute this anomaly to over-smoothing (Li et al., 2018), a phenomenon in which the feature vectors of different nodes tend to be consistent as the network deepens, leading to indistinguishable node representations. Many researchers have studied this problem and proposed some improvement methods (Li et al., 2018; Oono and Suzuki, 2019; Rong et al., 2019; Chen et al., 2020b, a; Huang et al., 2020; Cai and Wang, 2020; Yang et al., 2020; Chiang et al., 2019; Li et al., 2020). There are also articles explaining and analyzing the oversmoothing problem from the spectral analysis (Li et al., 2018) and dynamical system perspectives (Oono and Suzuki, 2019). However, there is still no unified theoretical framework to prove the effectiveness of these methods. In addition, the articles on related theoretical analysis have focused on specific models like graph convolution network (GCN) or graph attention network (GAT), and lack a comprehensive analysis and understanding of the general graph neural network.

**Contribution.**

- Noting the Markov property of the forward propagation process of GNNs, in this paper we try to develop a mathematical model to explain and analyze GNNs. Viewing the node set as a state space and the features of the nodes as the distribution over the state space, we model the forward message passing process of GNN as a discrete-time finite state Markov chain. We divide the message passing neural networks into two categories of operator-consistent models and operator-inconsistent models based on whether the message passing operators of each layer are consistent. Furthermore, we model the forward propagation process of the graph convolution model as a simple random walk on the graph and model the forward propagation process of the graph attention model as a time-inhomogeneous Markov chains on the graph. In addition, we model the stochastic method in GNN, DropEdge (Rong et al., 2019), as a random environment and use *Markov Chains in Random Environments* (MCRE) to study the GNN models which use the DropEdge method.

- We comprehensively study the over-smoothing problem of graph neural networks using the Markov chain model. We attribute the over-smoothing problem to the convergence of the probability distribution over the node set to the stationary distribu-
tion. Further, without being restricted to a specific GNN model, the over-smoothing problem of the general GNN is studied. On the one hand, we state that for the operator-consistent GNN, the node features cannot avoid over-smoothing, nor can they avoid over-smoothing at the exponential rate. On the other hand, we show that operator-inconsistent GNN model does not necessarily over-smooth and prove a sufficient condition for it to avoid over-smoothing. In addition, we interpret the methods for alleviating over-smoothing as different forms of lazy walk on the graph and prove the effectiveness of these methods. This part not only solve the important problem that limit the development of GNNs, but also is a successful case of using Markov chain to understand and analyze GNNs, demonstrating the potential of the Markov model to study other problems in GNNs.

- We designed experiments to verify our conclusions. We propose a regularization term based on it, which we call GNN-OI\(^1\) and can be plugged into existing GNN models by simply adding it on the original objective. GAT-OI can improve the performance as well as alleviate the over-smoothing problem. Also for GEN (Li et al., 2020), we conduct experiments on Appendix B and the results prove the sufficient promotion of GEN-OI on related graph tasks.

**Outline.** In Section 2, we introduce the graph neural networks, as well as over-smoothing, which is an important issue limiting the development of graph neural networks. In addition some definitions and conclusions in Markov chain theory are introduced. In Section 3, We model the GNN with the Markov chain on the graph in detail and model DropEdge (Rong et al., 2019) which is a common stochastic regularization method in GNNs with the random environment. In Sections 4, we study the over-smoothing problem based on the modeling in Section 3. Finally, in Section 5, we validate our conclusions through experiments on the real datasets.

**Notation.** Let \( G = (V, E) \) be a connected non-bipartite graph, where \( V := \{1, 2, \ldots, N\} \) is the node set, \( E \) is the edge set, \( N = |V| \) is the number of nodes and \( |A| \) denotes the number of elements in the set \( A \). If there are connected edges between nodes \( u, v \in V \), then denote by \( (u, v) \in E \). \( \text{deg}(u) \) denotes the degree of node \( u \in V \) and \( \mathcal{N}(v) \) denotes the neighbors of node \( v \). The corresponding adjacency matrix is \( A \) and the degree matrix is \( D \). \( \tilde{G} = (\tilde{V}, \tilde{E}) \) denotes the graph \( G \) added self-loop, the corresponding adjacency matrix is \( \tilde{A} \) and the degree matrix is \( \tilde{D} \).

Let \( \mathbb{R} \) be the set of real numbers, \( \mathbb{N} \) be the set of natural numbers, \( Z^+ \) be the set of positive integers, and \( F \) be the features dimensions of the nodes. Let \((\Omega, \mathcal{F}, P)\) be a probability space and \( E \) be the expectation operator on it. \( \| \cdot \| \) denotes the \( L^1 \) norm and \( \| \cdot \|_{TV} \) denotes the total variation distance. \( P^T \) denotes the transposed matrix of the square matrix \( P \). \( p(i, j) \) denotes the \( i \)th row \( j \)th column element of the matrix. \( P(i, \cdot) \) denotes the row vector formed by the \( i \)th row of the matrix \( P \) and \( P(\cdot, j) \) denotes the column vector formed by the \( j \)th column of the matrix \( P \).

1. 'OI' is taken from OPerator-Inconsistent. GNN-OI is the generic terms for various GNN models which can be specified such as GAT-OI and GEN-OI.
Let $P_{rw} := D^{-1}A$ be the transition matrix of simple random walk on the graph $\mathcal{G}$, $\tilde{P}_{rw} := \tilde{D}^{-1}\tilde{A}$ be the transition matrix of simple random walk on the graph $\tilde{\mathcal{G}}$, and let $P_{lazy} := (1 - \gamma)D^{-1}A + \gamma I$ for the transition matrix of the lazy walk on the graph $\mathcal{G}$.

2. Preliminaries

In this section we introduce the basic GNN model, the over-smoothing problem, and some definitions and conclusions in Markov chains theory.

2.1 Graph Neural Networks

**Graph convolution networks** are the most important class of GNN models at present. Researchers have been working to migrate the success of convolutional neural networks to the learning of graph data. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and the initial features of the nodes on it $H^{(0)} \in \mathbb{R}^{N \times F}$, Kipf and Welling (2016) improved neural network models on graphs (Bruna et al., 2013; Defferrard et al., 2016) and proposed the most widely studied and applied vanilla GCN

$$H^{(l)} = \sigma_{W^{(l)}}(P_{GCN}H^{(l-1)}) ,$$

where $H^{(l)} \in \mathbb{R}^{N \times F}$ is the node feature vector output by the $l$th hidden layer. $W^{(l)}$ is the parameter of the $l$th hidden layer. $P_{GCN} := \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ is the graph convolution operator.

**Remark 1.** Since the parameter $W^{(l)}$ is fixed during the forward propagation, we may consider writing the parameter $W^{(l)}$ together with the activation function as $\sigma_{W^{(l)}}$. This notation is also maintained in the writing of other models later on.

**Graph attention networks.** Inspired by the attention mechanism, many scholars have proposed attention-based graph neural network models (Veličković et al., 2018; Abu-El-Haija et al., 2018; Zhang et al., 2018; Lee et al., 2018). Among them, GAT (Veličković et al., 2018) is the most representative model. GAT establishes attention functions between nodes $u, v \in \mathcal{V}$ with connected edges $(u, v) \in \mathcal{E}$

$$\alpha^{(l)}_{u,v} = \frac{\exp(\phi^{(l)}(h^{(l-1)}_u, h^{(l-1)}_v))}{\sum_{k \in \mathcal{N}(u)} \exp(\phi^{(l)}(h^{(l-1)}_u, h^{(l-1)}_k))} ,$$

where $h^{(l)}_u \in \mathbb{R}^F$ is the embedding for node $u$ at the $l$-th layer and

$$\phi^{(l)}(h^{(l-1)}_u, h^{(l-1)}_v) := \text{LeakyReLU}^T(a[W^{(l)}h^{(l-1)}_u \parallel W^{(l)}h^{(l-1)}_v]) ,$$

where $a \in \mathbb{R}^{2F}$ and $W^{(l)}$ is the weight matrix. Then the GAT layer is defined as

$$h^{(l)}_u := \sigma_{W^{(l)}} \left( \sum_{v \in \mathcal{N}(u)} \alpha^{(l)}_{u,v} h^{(l-1)}_v \right) .$$

Written in matrix form

$$H^{(l)} = \sigma_{W^{(l)}}(P_{att}^{(l)}H^{(l-1)}) ,$$
where $P_{\text{att}}^{(l)} \in \mathbb{R}^{N \times N}$ is the attention matrix satisfying $P_{\text{att}}^{(l)}(u, v) = \alpha_{u,v}^{(l)}$ if $v \in \mathcal{N}(u)$ otherwise $P_{\text{att}}^{(l)}(u, v) = 0$ and $\sum_{v=1}^{N} P_{\text{att}}^{(l)}(u, v) = 1$.

**Message Passing Neural Network** (MPNN) is a GNN model proposed by Gilmer et al. (2017). MPNN is a general framework for current GNN models, and most GNN models can be unified under this framework. It describes the GNN uniformly as a process that the information of the neighbors of a node $u \in \mathcal{V}$ in a graph is passed as messages along an edge and aggregated at the node $u$, i.e.

$$h_{u}^{(l)} = \text{UP}^{(l)} \left( h_{u}^{(l-1)}, \text{MSG}^{(l)}(h_{v}^{(l-1)}, v \in \mathcal{N}(u)) \right),$$

where $h_{u}^{(l)}$ denotes the features of node $u$ output from the $l$th hidden layer of the model, $\text{UP}^{(l)}$ and $\text{SG}^{(l)}$ denote the update and message passing functions of the $l$th layer, respectively.

GNN such as GCN and GAT can be written in this form, differing only in the update and message passing functions.

### 2.2 Over-smoothing

When neural networks are getting deeper and deeper, a phenomenon that the GCN had better experimental results in the shallow layer case, and instead did not work well as the number of layers in the network increased was found. The researchers found that this is due to the fact that during the GCN training process, the hidden layer representation of each node tend to converge to the same value as the number of layers increases. This phenomenon is called over-smoothing. This problem affects the deepening of GNN layers and limits the further development of GNNs. The main current methods to alleviate over-smoothing are residual connections method (Kipf and Welling, 2016; Chiang et al., 2019), personalized propagation of neural predictions (PPNP) (Klicpera et al., 2018), and the DropEdge method (Rong et al., 2019; Huang et al., 2020).

**Residual connections method** is proposed based on an intuitive analysis of the over-smoothing problem. From a qualitative perspective, the over-smoothing problem is that as the network is stacked, the model forgets the initial input features and only updates the features based on the structure of the graph data. It is natural to think that the problem of the model forgetting the initial features can be alleviated by reminding the network what its previous features are. Many methods have been proposed based on such intuitive analysis. The simplest one, Kipf and Welling (2016) propose to add residual connections to graph convolutional networks

$$H^{(l)} = \sigma_{W^{(l)}} \left( P_{\text{GCN}} H^{(l-1)} \right) + H^{(l-1)},$$

The node features of the $l$th hidden layer are directly added to the node features of the previous layer $H^{(l-1)}$ to remind the network not to forget the previous features. However, Chiang et al. (2019) argues that residual connectivity ignores the structure of the graph and should be considered to reflect more the influence of the weights of different neighboring nodes. So this work gives more weight to the features from the previous layer in the message
passing of each GCN layer by improving the graph convolution operator

$$H^{(l)} = \sigma_{W^{(l)}} \left( (P_{\text{GCN}} + I) H^{(l-1)} \right). \quad (3)$$

**Personalized propagation of neural predictions.** The well-known graph neural network PPNP (Klicpera et al., 2018) has also been experimentally proven to alleviate the over-smoothing. The innovation of PPNP is to decouple node information embedding and node feature propagation in GNN models. It simplifies the model by not learning the parameters during model propagation, and becomes one of the GNN models that are widely studied and applied (Bojchevski et al. 2020).

PPNP’s node information embedding is implemented by a MLP

$$H^{(0)} = f_W(X).$$

Inspired by personalized PageRank (PPR), the node feature propagation process is

$$H^{(l)} = \sigma \left( (1 - \alpha) \tilde{D}^{-1} \tilde{A} H^{(l-1)} + \alpha H^{(0)} \right), \quad (4)$$

where $\alpha \in (0, 1)$ is the teleport (or restart) probability.

**DropEdge** is a method proposed by Rong et al. (2019) to alleviate the over-smoothing problem. Its simple idea and powerful generalization in GNN models have made this method widely used. And it becomes a method for dealing with large-scale graph data. The idea of DropEdge is to randomly drop some edges in the original graph $\tilde{G} = (\tilde{V}, \tilde{E})$ at each layer. The specific operation is to randomly let some elements 1 of the adjacency matrix $A$ become 0

$$A^{(l)}_{\text{drop}} = A - A^{(l)}_{\text{drop}},$$

where $A^{(l)}_{\text{drop}}$ is the adjacency matrix formed by the expansion of a random subset $\tilde{E}'$ of $\tilde{E}$.

Although these three methods have been experimentally verified to alleviate GCN over-smoothing, their nature lacks explanation. Residual connections method only discusses the intuitive understanding, and PPNP is experimentally found to alleviate over-smoothing. DropEdge, as a widely used stochastic regularization method, needs more mathematical explanation. In addition, the effectiveness of these three methods needs to be proved theoretically.

### 2.3 Results in Markov Chains

Since the graph has a finite node set and the forward propagation process of graph neural networks is time-discrete, we focus on the results related to discrete-time Markov chains in finite state. This section introduces some conclusions of Markov chains, Markov chains in random environments (Cogburn, 1980, 1990; Nawrotzki, 1982), which has only been developed in the last three decades, and mixing time (Levin and Peres, 2017), which is an important tool to characterize the transitions of Markov chains. The proofs of some conclusions in this section will be added in the Appendix A.
Lemma 2 (Dobrushin’s inequality) Let $\mu$ and $\nu$ be probability distributions on a finite state space $E$ and $P$ be a transition matrix, then
\[
\|\mu P - \nu P\| \leq C(P) \|\mu - \nu\|,
\]
where
\[
C(P) := \frac{1}{2} \sup_{i,j} \sum_{k \in E} |p(i,k) - p(j,k)|
\]
is called the Dobrushin contraction coefficient of the transition matrix $P$ and $\| \cdot \|$ denotes the $L^1$ norm.

Theorem 3 If one of the following two conditions is satisfied

1. $P$ is irreducible and aperiodic.
2. $C(P) < 1$.

Then there exist stationary distribution $\pi$, constants $\alpha \in (0, 1)$ and $C > 0$ such that
\[
\max_{i \in E} \|P^n(i, \cdot) - \pi\| \leq C \alpha^n,
\]
where $P(i, \cdot)$ denotes the row vector consisting of the $i$th row of the transition matrix $P$.

Compared to the time-homogeneous chain, it is much more difficult to describe the transfer of an arbitrary initial distribution according to a time-inhomogeneous chain. The limiting case that an arbitrary initial distribution transfer according to a time-inhomogeneous chain was discussed by Bowerman et al. (1977); Huang et al. (1976). The following Dobrushin-Isaacson-Madsen theorem gives a sufficient condition for the existence of stationary distributions in the limiting sense of time-inhomogeneous Markov chains.

Theorem 4 (Dobrushin-Isaacson-Madsen theorem) Let $\bar{X} = \{X_n, n \in T\}$ be a time-inhomogeneous Markov chain on a finite state space $E$ with transition matrix $P^{(n)}$. If the following (1), (2) and (3A) or (3B) are satisfied

1. There exists a stationary distribution $\pi^{(n)}$ when $P^{(n)}$ is treated as a transition matrix of a time-homogeneous chain;
2. $\sum_{n} \|\pi^{(n)} - \pi^{(n+1)}\| < \infty$;
3A (Isaacson-Madsen condition) For any probability distribution $\mu, \nu$ on $E$ and positive integer $k$
\[
\|(\mu - \nu)P^{(k)} \cdots P^{(n)}\| \rightarrow 0, \quad n \rightarrow \infty.
\]
3B (Dobrushin condition) For any positive integers $k$
\[
C(P^{(k)} \cdots P^{(n)}) \rightarrow 0, \quad n \rightarrow \infty.
\]

Then there exists a probability measure $\pi$ on $E$ such that
(1) \[ \| \pi^{(n)} - \pi \| \to 0, \quad n \to \infty; \]

(2) Let the initial distribution be \( \mu_0 \) and the distribution of the chain \( \tilde{X} \) at step \( n \) be 
\[ \mu_n := \mu_{n-1} P^{(n)} \]
then for any initial distribution \( \mu_0 \), we have 
\[ \| \mu_n - \pi \| \to 0, \quad n \to \infty, \]
where \( \| \cdot \| \) denotes the \( L^1 \) norm.

Although Markov chains are widely used in real-world engineering, some practical problems have more complex environmental factors that affect the transition of the original Markov chain. We need a more advanced mathematical tool to deal with such problems. If these complex environmental factors are described as a stochastic process that affects the transition function of the original chain, the original chain may lose the Markov property as a result, so researchers Cogburn (1980, 1990), Nawrotzki (1982) and Orey (1991) developed a theory of Markov chains in random environments (MCRE) to deal with these problems.

Let \((\Omega, \mathcal{F}, \mathbb{P})\) be a probability space, \((E, \mathcal{E}), (Y, \mathcal{Y})\) be two state spaces, \(T_1 = \mathbb{N} = \{0, 1, 2, \ldots \}, T_2 = \mathbb{Z}^+ = \{1, 2, \ldots \}\) are two sets of times. In the following we first introduce the random Markov kernel that couples the original chain to the random environment.

**Definition 5 (random Markov kernel)** Let 
\[ p(\cdot; \cdot, \cdot) : Y \times E \times E \mapsto [0, 1]. \] If

(1) For every fixed \( \theta \in Y \), \( p(\theta; \cdot, \cdot) \in MK(E, \mathcal{E}) \), where \( MK(E, \mathcal{E}) \) is the set of all Markov kernels on \((E, \mathcal{E})\).

(2) For every fixed \( A \in \mathcal{E} \), \( p(\cdot; \cdot, A) \) on \( Y \times X \) is measurable.

Then \( p \) is said to be a random Markov kernel. The set of all random Markov kernels on \((E, \mathcal{E})\) is noted as \( RMK(E, \mathcal{E}) \).

In particular, if \( E \) is a finite set, then any random Markov kernel \( p(\theta; i, A) \) is given by a random transition matrix
\[ P(\theta) := (p(\theta; i, j), i, j \in E), \]
where \( p(\theta; i, j) := p(\theta; i, \{j\}) \), and
\[ p(\theta; i, A) = \sum_{j \in A} p(\theta; i, j). \]

In the following we give the definition of a Markov chain in a positive time-dependent random environment with finite state space.

**Definition 6 (Markov chains in random environments)** Let \( \tilde{X} = \{X_n, n \in T_1\} \) and \( \tilde{\xi} = \{\xi_n, n \in T_2\} \) be two random sequences defined on the probability space \((\Omega, \mathcal{F}, \mathbb{P})\) taking values in the finite sets \( E \) and \( Y \), respectively. \( p(\cdot; \cdot, \cdot) \) is a random Markov kernel. If for any \( n \geq 2 \) and any \( i_0, \ldots, i_n \in E \), we have
\[ P(X_n = i_n \mid X_0 = i_0, \ldots, X_{n-1} = i_{n-1}, \tilde{\xi}) = p(\xi_{n-1}; i_{n-1}, i_n). \]

Then we say that \((\tilde{X}, \tilde{\xi})\) is a Markov chain in a positive time-dependent random environment, which is abbreviated as Markov chain in a random environment in this paper, and is denoted as MCRE.
If $\tilde{\xi} = \{\xi_n, n \in T_2\}$ is not stochastic, then the MCRE degenerates to a classical Markov chain. The MCRE is the Markov chain whose transition probability is influenced by a random environment.

**Remark 7** In general, under the definition of Cogburn (1980, 1990), Markov chains in positive time-dependent random environments and Markov chains in random environments are two different concepts. The Markov chain in a positive time-dependent random environment is not necessarily a MCRE, and a MCRE is not necessarily a Markov chain in a positive time-dependent random environment. Without causing confusion, we use the term MCRE to express Markov chains in positive time-dependent random environments for the convenience of illustration in this paper.

We are also interested in the rate that the initial distribution over state space converge to the stationary distribution. In Markov chains theory, the mixing time is used to denote the time required for a certain probability distribution to converge to the stationary distribution.

**Definition 8 (Mixing Time)** Let $\tilde{X} = \{X_n, n \in T\}$ be a time-homogeneous Markov chain on a finite state space $E$ with transition matrix $P$ and stationary distribution $\pi$. The mixing time is defined by

$$t_{\text{mix}}(\epsilon) := \min \{t : d(t) \leq \epsilon\},$$

where

$$d(t) := \max_{i \in E} \|P^t(i, \cdot) - \pi\|_{TV}.$$  

Following are the properties of total variational distance.

**Proposition 9** Let $\mu$ and $\nu$ be two probability distributions on a finite set $E$. Then

$$\|\mu - \nu\|_{TV} = \frac{1}{2}\|\mu - \nu\| = \frac{1}{2} \sum_{i \in E} |\mu(i) - \nu(i)|.$$  

**Proposition 10** Let $P$ be the transition matrix of a Markov chain with state space $E$ and let $\mu$ and $\nu$ be any two distributions on $E$. Then

$$\|\mu P - \nu P\|_{TV} \leq \|\mu - \nu\|_{TV}.$$  

This in particular shows that $\|\mu P^{t+1} - \pi\|_{TV} \leq \|\mu P^t - \pi\|_{TV}$, that is, advancing the chain can only move it closer to stationarity.

### 3. Markov chain modeling for GNNs

This section introduces the Markov chain model for GNN. In Section 3.1, we describe the Message Passing Neural Network (MPNN) with a discrete-time finite Markov chain on the graph. And we divide the MPNN into two classes. More specifically, in 3.2, we describe GCN with the simple random walk on the graph. In 3.3, we describe the graph attention model with the time-inhomogeneous chain on the graph. In 3.4, the DropEdge+GCN model is described by a random walk on graph in a random environment.
3.1 Message passing framework

In this subsection we model the message forward propagation process of MPNN as a Markov chain on the graph and divide it into two categories of models.

Recalling the message passing framework equation (1), since the node features of the $l$th layer are obtained from the node features of the $l-1$th layer only, independent of the previous $l-2$ layers, the message passing process can be described as a Markov chain on the graph.

We take the node set $\mathcal{V}$ as the state space and construct the family of transition matrices \( \{P^{(1)}, P^{(2)}, \ldots, P^{(l)}, \ldots\} \) according to MSG\(^{(l)}\) and UP\(^{(l)}\). Consider the features on the nodes $H^{(l)}$ as the distribution on the node set $\mathcal{V}$. Then the message passing process at $l$th layer in the MPNN is a one-step transition process of the distribution $H^{(l-1)}$ according to the one-step transition matrix $P^{(l)}$. In this way, we model the forward process of the message passing model as a Markov Chain $\vec{V}$ with state space $\mathcal{V}$ and initial distribution $H^{(0)}$, transferring according to the family of transition matrices $\{P^{(1)}, P^{(2)}, \ldots, P^{(l)}, \ldots\}$. Then we can use Markov chains theory to study the message passing model.

If the message passing operator is consistent at each layer, i.e., for all $l \geq 1$, MSG\(^{(l)}\) and UP\(^{(l)}\) are the same, then the transition matrices $P^{(l)}$, $\forall l \geq 1$ are the same and we can model it with a time-homogeneous Markov chain. Otherwise, the message passing operator is inconsistent, and we can model it with a time-inhomogeneous Markov chain.

We classify GNNs into two categories, operator-consistent GNN and operator-inconsistent GNN, based on whether the message passing operators of each layer of the model are consistent. Specifically, we use GCN and GAT as representatives of operator-consistent GNN and operator-inconsistent GNN, respectively, which are discussed in Sections 3.2 and 3.3.

3.2 Graph convolution model

In this section, we discuss the equivalence between GCN and a simple random walk on the graph. Considering $\vec{V}$ is a simple random walk on $\tilde{\mathcal{G}}$ with transition matrix

$$\tilde{P} = \tilde{D}^{-\frac{1}{2}} \tilde{A}.$$

The graph convolution operator can be written as

$$P_{\text{GCN}} = \tilde{D}^{-\frac{1}{2}} \tilde{P}^T \tilde{D}^{\frac{1}{2}}.$$

Thus the message passing of the $l$ th layer of GCN is

$$H^{(l)} = P_{\text{GCN}} H^{(l-1)} = (P_{\text{GCN}})^l H^{(0)} = \tilde{D}^{-\frac{1}{2}} \left( \tilde{P}^T \right)^l \tilde{D}^{\frac{1}{2}} H^{(0)}.$$

Both sides multiply left $\tilde{D}^{\frac{1}{2}}$ at the same time

$$\tilde{D}^{\frac{1}{2}} H^{(l)} = \left( \tilde{P}^T \right)^l \tilde{D}^{\frac{1}{2}} H^{(0)}.$$
Let \( X^{(l)} = \left( \tilde{D}_l H^{(l)} \right)^T \in \mathbb{R}^{F \times N} \), then for both sides transposed simultaneously we have
\[
X^{(l)} = X^{(l-1)} \tilde{P}.
\]
(6)

This is exactly the simple random walk \( \tilde{V}_{rw} \) with initial distribution \( X^{(0)} \) on the graph \( \tilde{G} \).

**Remark 11** Similar to Li et al. (2018), in this paper's discussion of graph neural networks, we omit the nonlinear activation function \( \sigma \) between layers in graph convolutional networks. In fact, according to the spectral analysis of Wu et al. (2019) and the experimental results, the GNN with omitted nonlinear activation function is not different from the GNN with added activation function in terms of performance. In this paper, we call this expression in the form of equation (5) the message passing of the model.

Combining the above discussion, we model the most basic model in GNN, vanilla GCN, as simple random walk on \( \tilde{G} \) which is the simplest Markov chain on the graph. The graph convolution models (Kipf and Welling, 2016; Atwood and Towsley, 2016; Simonovsky and Komodakis, 2017; Pham et al., 2017) are constructed by designing graph convolution operators and then stacking the graph convolution layers layer by layer. We can follow the above discussion to describe the graph convolution operator as a one-step transition matrix and construct a time-homogeneous Markov chain on the graph \( G \) to complete the Markov chain modeling of the graph convolution model. Accordingly, we can study the graph convolution model with the help of studying the time-homogeneous Markov chain.

### 3.3 Graph attention model

In this section, we discuss the equivalence between GAT and a time-inhomogeneous Markov chain on the graph. Since \( P^{(l)}_{att}(u, v) \geq 0 \) and \( \sum_{v=1}^{N} P^{(l)}_{att}(u, v) = 1 \), \( P^{(l)}_{att} \) is a transition matrix. We consider a weighted graph \( G^{(l)}_{att} = (V^{(l)}_{att}, E^{(l)}_{att}) \) with edges \( (u, v) \in E^{(l)}_{att} \) weights
\[
w^{(l)}(u, v) := \exp \left( \text{LeakyReLU}(a^T W^{(l)} h^{(l-1)}_u \| W^{(l)} h^{(l-1)}_v) \right).
\]

The weighted degree of node \( u \in G^{(l)}_{att} \) is
\[
deg(u) := \sum_{k \in N(u)} \exp \left( \text{LeakyReLU}(a^T W^{(l)} h^{(l-1)}_u \| W^{(l)} h^{(l-1)}_v) \right).
\]

Based on this definition, \( P^{(l)}_{att} \) is the transition matrix of simple random walk on the weighted graph \( G^{(l)}_{att} \).

After analyzing the GAT operator \( P^{(l)}_{att} \) for each layer, we return to GAT. Unlike the GCN model, the message transition matrices \( P^{(l)}_{att} \) for each layer of GAT are inconsistent. It is natural to use a time-inhomogeneous Markov chain \( V^{(l)}_{att} \) with the state space \( V \) and a family of transition matrices
\[
\{ P^{(1)}_{att}, P^{(2)}_{att}, \ldots, P^{(l)}_{att}, \ldots \}
\]
to model GAT. In Section 4.4, we will discuss the properties of GAT message passing using the relevant conclusions of time-inhomogeneous Markov chains.
3.4 DropEdge method

Recalling the DropEdge method introduced in Section 2.2, in this subsection we model it as a random environment. Specifically, we connect DropEdge+GCN and a random walk in a random environment on the graph.

Consider the stochastic process
\[ \vec{\xi} = (\xi_1, \xi_2, \ldots, \xi_l, \ldots) = (\Theta^{(1)}, \Theta^{(2)}, \ldots, \Theta^{(l)}, \ldots) \]
with the set of time parameters \( T_2 = \mathbb{Z}^+ \), and taking values in \( \vec{\Xi} := \Xi_1 \times \Xi_2 \times \cdots \times \Xi_l \times \cdots \), where \( \Xi_i \subset \{0, 1\}^{N \times N} \), \( \forall i \geq 1 \). \( \Theta^{(l)}, \ l \geq 1 \) are the random adjacency matrices that are independent and have the same distribution. If \( (u, v) \in \tilde{E} \), its elements \( \theta^{(l)}(u, v) \) satisfy
\[
\begin{align*}
\mathbb{P}(\theta^{(l)}(u, v)) &= \begin{cases} 
1 - \frac{1}{|\tilde{E}|} & \theta^{(l)}(u, v) = 1 \\
\frac{1}{|\tilde{E}|} & \theta^{(l)}(u, v) = 0 
\end{cases}
\end{align*}
\]
is a Bernoulli random variable. It indicates that each edge \( (u, v) \) is dropped with a uniform probability \( \frac{1}{|\tilde{E}|} \). We use the random environment \( \vec{\xi} = (\Theta^{(1)}, \Theta^{(2)}, \ldots, \Theta^{(l)}, \ldots) \) to model \( (A_{\text{drop}}^{(1)}, A_{\text{drop}}^{(2)}, \ldots, A_{\text{drop}}^{(l)}, \ldots) \).

We next model DropEdge+GCN. Since \( H^{(l)} \) is only related to \( H^{(l-1)} \) and \( A_{\text{drop}}^{(l)} \), and we have described GCN as a simple random walk on the graph, according to the definition of MCRE in Section 2.3, we describe the DropEdge+GCN as a simple random walk on the graph in the random environment.

The message passing of DropEdge+GCN at the lth layer is
\[
H^{(l)} = \left( D_{\text{drop}}^{(l)} \right)^{-1} A_{\text{drop}}^{(l)} H^{(l-1)},
\]
where the degree matrix after DropEdge is \( D_{\text{drop}}^{(l)} := \text{diag}(\zeta_1^{(l)}, \ldots, \zeta_N^{(l)}) \), the degree of node \( u \)
\[
\zeta_u^{(l)} := \sum_{v \in \mathcal{N}(u)} \theta^{(l)}(u, v) \tag{7}
\]
is a random variable. Since \( \theta^{(l)}(u, v) \) is a Bernoulli random variable, the random variable \( \zeta_u^{(l)} \) follows a binomial distribution with parameters \( |\mathcal{N}(u)| = \deg(u) \) and \( 1 - \frac{1}{|\tilde{E}|} \), i.e.
\[
\zeta_u^{(l)} \sim B(\deg(u), 1 - \frac{1}{|\tilde{E}|}).
\]

Let \( \vec{V} = (V_0, V_1, \ldots, V_l, \ldots) \) be the original chain. \( V_i, i = 0, 1, \ldots \) is random variables taking values in \( \mathcal{V} \). Consider the random transition matrix
\[
P(\Theta^{(l)}) := D_{\Theta^{(l)}}^{-1} \Theta^{(l)},
\]

---

2. DropEdge+GCN denotes the GCN model using DropEdge method
where $\tilde{D}_{\Theta(l)} = D_{\text{drop}}^{(l)}$. Since $\forall l \geq 2$ and $\forall v_0, v_1, \ldots, v_l \in \mathcal{V}$,

$$P(V_l = v_l|V_0 = v_0, \ldots, V_{l-1} = v_{l-1}, \tilde{\xi}) = p(\Theta^{(l)}; v_{l-1}, v_l) = \frac{\theta^{(l)}(v_{l-1}, v_l)}{\zeta_{v_{l-1}}},$$

$(\tilde{\mathcal{V}}, \tilde{\xi})$ is a MCRE.

To summarize the discussion, we describe the DropEdge method as a stochastic process $\tilde{\xi}$. And specifically, we model DropEdge+GCN as a random walk in a random environment on the graph $(\tilde{\mathcal{V}}, \tilde{\xi})$ with random transition matrices $\{P(\Theta^{(l)})\}_{l \geq 1}$. DropEdge applied to different GNN models means MCRE with the same random environment but different original chains. Although the specific details are not same, the modeling methods are the same. Next, in Section 4.2, we discuss in depth the effectiveness of DropEdge method to alleviate the over-smoothing problem.

4. Markov analysis of the over-smoothing problem

In this section, we attribute the over-smoothing problem to the convergence of an arbitrary initial distribution to a stationary distribution. In Section 4.2, we demonstrate the effectiveness of previous methods to alleviate the over-smoothing problem by analyzing the lazy walk on the graph. Furthermore, we point out that these methods still can not avoid over-smoothing at the exponential rate. In Section 4.3, we show that the over-smoothing phenomenon is inherent to the operator-consistent GNN model, and that it is neither possible to avoid over-smoothing nor to avoid over-smoothing at the exponential rate in the Markovian sense. In Section 4.4, we prove the conclusion that the operator-inconsistent GNN models can avoid over-smoothing under certain conditions, and we give a sufficient condition.

4.1 Cause of over-smoothing

In Section 3.1 we show that a Markov Chain $\tilde{\mathcal{V}}$ can be used to describe the message passing framework. Meanwhile, the feature vectors of the nodes can be described as distributions over the node set. The forward propagation process of node features is the transfer process of the distribution on the node set $\mathcal{V}$. Thus over-smoothing problem is the process of the feature distribution converging to the stationary distribution. In the following, we take vanilla GCN as an example to specifically analyze why the node representations are over-smoothing.

For a simple random walk on the graph $\tilde{\mathcal{G}}$, for all $v \in \tilde{\mathcal{V}}$

$$\sum_{u \in \tilde{\mathcal{V}}} \frac{\text{deg}(u)}{\text{deg}(v)} \tilde{p}_{\text{rw}}(u, v) = \sum_{(u, v) \in \tilde{\mathcal{E}}} \frac{\text{deg}(u)}{\text{deg}(v)} = \text{deg}(v),$$

where $\tilde{p}_{\text{rw}}(u, v) = \frac{1}{\text{deg}(u)}$ if $(u, v) \in \tilde{\mathcal{E}}$. To get a probability, we simply normalize by $\sum_{v \in \tilde{\mathcal{V}}} \text{deg}(v) = 2|\tilde{\mathcal{E}}|$. Then the probability measure

$$\pi(u) = \frac{\text{deg}(u)}{2|\tilde{\mathcal{E}}|} \quad \forall u \in \tilde{\mathcal{V}}$$
is always a stationary for the walk. Write in matrix form
\[ \pi = \pi \tilde{P}_{tw}. \]
Recall Section 3.2 we use a simple random walk on the graph to describe the vanilla GCN. Noting
\[ X^{(l)} = (\tilde{D}^{1/2} H^{(l)})^T, \]
the message passing of the \( l \) th GCN hidden layer is
\[ X^{(l)} = X^{(l-1)} \tilde{P}_{tw}. \]
Since simple random walks on connected graphs are irreducible and aperiodic Markov chains,
\[ X^{(l)}(k, \cdot) \to \pi, \quad l \to \infty, \]
where \( X^{(l)}(k, \cdot), \quad k = 1, 2, \ldots, F \) is the distribution over the node set \( \tilde{V} \) consisting of the \( k \)th component of each node feature, \( \pi \) is the unique stationary distribution of \( \tilde{P}_{tw} \), thus for all \( u \in \tilde{V} \)
\[ x^{(l)}(1, u) = x^{(l)}(2, u) = \cdots = x^{(l)}(F, u) = \pi(u), \]
where \( \pi(u) \) is the \( u \)th component of \( \pi \).

From the above discussion, the over-smoothing problem is due to the fact that the features on the nodes tend to be stationary distribution with the move of the Markov chain, resulting in the convergence of the each node feature vector. In the next subsection we will analyze the methods that alleviate the over-smoothing problem in detail.

### 4.2 Lazy walk analysis of methods to alleviate over-smoothing

In this subsection, we use the lazy walk on the graph to uniformly model the methods that alleviate the over-smoothing problem including the residual connections method (Kipf and Welling, 2016; Chiang et al., 2019), personalized propagation of neural predictions (PPNP)(Klicpera et al., 2018), and the DropEdge method (Rong et al., 2019; Huang et al., 2020) as lazy walks on the graph. Finally, we proof the effectiveness of these methods.

**Residual Connections Method.** Essentially, consider the simplification of the model in Remark 11. If we omit the nonlinear activation function in the forward propagation process of the graph neural network and focus only on the node message passing process, then the models of equation (2) and (3) are the same. The operator \( P_{GCN} + I \) is normalized
\[ P_{res} := \frac{1}{2} P_{GCN} + \frac{1}{2} I. \]
Then the message passing for the residual connections method is
\[ H^{(l)} = \left( \frac{1}{2} P_{GCN} + \frac{1}{2} I \right) H^{(l-1)} = P_{res} H^{(l-1)}. \]
According to the analysis of Wang et al. (2019a), we describe \( P_{res} \) as the transition matrix of a lazy walk with parameter \( \gamma = \frac{1}{2} \). Thus residual connections method can be regarded as a lazy walk on the graph.
PPNP. We consider the lazy walk with the parameter $\alpha$ on $\tilde{G}$ with transition matrix

$$P_{\text{lazy}} = (1 - \alpha)\tilde{D}^{-1}\tilde{A} + \alpha I.$$ (8)

This is formally similar to the PPNP messaging (equation (4)). Intuitively, equation (8) indicates that the original random walk has a $1 - \alpha$ probability of continuing to walk, and a $\alpha$ probability of staying in place. This is exactly the idea of personalized PageRank.

Combining the above discussion, we can use a lazy walk on $\tilde{G}$ with parameter $\alpha$ to describe the node feature propagation in PPNP. As can be seen from the message passing expression (8), PPNP is a more general method than the residual connections method. In particular, if $\alpha = \frac{1}{2}$, formally PPNP becomes the residual connections method.

DropEdge. Unlike residual connections method and PPNP, DropEdge method explicitly have nothing to do with the form of lazy walk. However, we will prove that message passing of the DropEdge+GCN model is a lazy walk on the graph.

In Section 3.4 we modeled DropEdge+GCN as a MCRE $(\vec{V}, \vec{\xi})$ on the graph. The following theorem illustrates that the original chain $\vec{V}$ is a lazy walk on the graph.

**Theorem 12** Let $(\vec{V}, \vec{\xi})$ be the MCRE that describes the DropEdge+GCN model in Section 3.4.

$$\vec{\xi} = (\Theta^{(1)}, \Theta^{(2)}, \ldots, \Theta^{(l)}, \ldots)$$

is a random environment with independent identical distribution. Then the original chain $\vec{V}$ is a time-homogeneous Markov chain with a transition matrix

$$P_{\text{drop}} := (I - \Gamma)\tilde{D}^{-1}\tilde{A} + \Gamma,$$ (9)

where

$$\Gamma := \text{diag}\left( \frac{1}{|E|\deg(1)}, \frac{1}{|E|\deg(2)}, \ldots, \frac{1}{|E|\deg(N)} \right)$$

is a diagonal matrix.

**Proof.** Let the distribution of $\xi_1$ be $\mu := P \circ (\xi_1)^{-1}$, then the distribution of $\vec{\xi}$ is $\nu = \mu^{Z^+}$. Given any path $(V_0 = v_0, V_1 = v_1, \ldots, V_l = v_l)$, $v_i \in V$, $i = 0, 1, \ldots, l$ of $(\vec{V}, \vec{\xi})$,

$$P(V_0 = v_0, \ldots, V_l = v_l) = \int_{\xi} \delta_{v_0 v_0} p(\Theta^{(1)}; v_0, v_1) p(\Theta^{(2)}; v_1, v_2) \cdots p(\Theta^{(l)}; v_l-1, v_l) \nu(d\vec{\xi})$$

$$= \delta_{v_0 v_0} \int_{\xi} p(\Theta^{(1)}; v_0, v_1) \mu(d\Theta^{(1)}) \cdots \int_{\xi} p(\Theta^{(l)}; v_{l-1}, v_l) \mu(d\Theta^{(l)})$$

$$= \delta_{v_0 v_0} \mathbb{E} \left[ \frac{\theta^{(1)}(v_0, v_1)}{\zeta^{(1)}_{v_0}} \right] \mathbb{E} \left[ \frac{\theta^{(2)}(v_1, v_2)}{\zeta^{(2)}_{v_1}} \right] \cdots \mathbb{E} \left[ \frac{\theta^{(l)}(v_{l-1}, v_l)}{\zeta^{(l)}_{v_{l-1}}} \right],$$

where the notations are all defined in Section 3.4, and the random variables $\zeta_{v_{l-1}}^{(l)}$ are defined as equation (7). Since $\Theta^{(0)}, \Theta^{(1)}, \ldots, \Theta^{(l)}, \ldots$ are independently identical distribution, $\vec{V}$ is a time-homogeneous Markov chain with the transition matrix

$$\mathbb{E}[P(\Theta^{(1)})] = \cdots = \mathbb{E}[P(\Theta^{(l)})] = \cdots.$$
The following computes the transition matrix $E[P(\Theta(l))]$, i.e., the expectation of $\frac{\theta(l)(u,v)}{\zeta_u}$, for all $(u, v) \in E$. Since

$$
\zeta_u^{(l)} \sim B(\text{deg}(u), 1 - \frac{1}{|E|}),
$$

for $k = 1, 2, \ldots, \text{deg}(u)$,

$$
P\left(\frac{\theta(l)(u,v)}{\zeta_u^{(l)}} = \frac{1}{k}\right) = P(\theta(l)(u,v) = 1, \zeta_u^{(l)} = k)
$$

$$
= P(\theta(l)(u,v) = 1)P(\zeta_u^{(l)} = k|\theta(l)(u,v) = 1)
$$

$$
= P(\theta(l)(u,v) = 1)P(\zeta_u^{(l)} - \theta(l)(u,v) = k - 1|\theta(l)(u,v) = 1)
$$

$$
= (1 - \frac{1}{|E|})^{C_{\text{deg}(u)-1}} \left(1 - \frac{1}{|E|}\right)^{k-1} \left(\frac{1}{|E|}\right)^{(\text{deg}(u)-1)-(k-1)}
$$

$$
= C_{\text{deg}(u)-1}^{k-1} \left(1 - \frac{1}{|E|}\right)^{k} \left(\frac{1}{|E|}\right)^{\text{deg}(u)-k};
$$

$$
P\left(\frac{\theta(l)(u,v)}{\zeta_u^{(l)}} = 0\right) = 1 - \sum_{k=1}^{\text{deg}(u)} P\left(\frac{\theta(l)(u,v)}{\zeta_u^{(l)}} = \frac{1}{k}\right),
$$

where $C_n^m := \frac{n!}{m!(n-m)!}$ denotes the combinatorial number, satisfying the formula

$$
\frac{1}{n} \cdot C_n^m = \frac{1}{m} \cdot C_{n-1}^{m-1}.
$$

The probability distribution of $\frac{\theta(l)(u,v)}{\zeta_u}$ is obtained, and its expectation is calculated below

$$
E\left[\frac{\theta(l)(u,v)}{\zeta_u^{(l)}}\right] = \sum_{k=1}^{\text{deg}(u)} C_{\text{deg}(u)-1}^{k-1} \left(1 - \frac{1}{|E|}\right)^{k} \left(\frac{1}{|E|}\right)^{\text{deg}(u)-k} \cdot \frac{1}{k}
$$

$$
= \sum_{k=1}^{\text{deg}(u)} \left(1 - \frac{1}{|E|}\right)^{k} \left(\frac{1}{|E|}\right)^{\text{deg}(u)-k} \cdot \frac{1}{\text{deg}(u)} \cdot C_{\text{deg}(u)-1}^{k-1}
$$

$$
= \sum_{k=1}^{\text{deg}(u)} \left(1 - \frac{1}{|E|}\right)^{k} \left(\frac{1}{|E|}\right)^{\text{deg}(u)-k} \cdot \frac{1}{\text{deg}(u)} \cdot C_{\text{deg}(u)}^{k}
$$

$$
= \frac{1}{\text{deg}(u)} \cdot \left(\sum_{k=1}^{\text{deg}(u)} C_{\text{deg}(u)}^{k} \left(1 - \frac{1}{|E|}\right)^{k} \left(\frac{1}{|E|}\right)^{\text{deg}(u)-k}\right)
$$

$$
= \frac{1}{\text{deg}(u)} \cdot \left(1 - C_{\text{deg}(u)}^{0} \left(\frac{1}{|E|}\right)^{\text{deg}(u)}\right)
$$

$$
= \left(1 - \frac{1}{|E|^{|\text{deg}(u)|}}\right) \bar{p}_{\text{rw}}(u,v),
$$

16
where \( \tilde{p}_{rw}(u,v) = \frac{1}{\deg(u)} \) is the \( u \)th row, \( v \)th column element of \( \tilde{P}_{rw} := \tilde{D}^{-1}\tilde{A} \). So the transition matrix of the original chain \( \tilde{V} \) is

\[
P_{\text{drop}} := E[\tilde{P}(\Theta(l))] = (I - \Gamma)\tilde{D}^{-1}\tilde{A} + \Gamma,
\]

where

\[
\Gamma := \text{diag}\left(\frac{1}{|E|\deg(1)}, \frac{1}{|E|\deg(2)}, \ldots, \frac{1}{|E|\deg(N)}\right)
\]
is a diagonal matrix. The element \( \frac{1}{|E|\deg(u)} \) denotes the probability that all edges connected to the node \( u \) are dropped.

The conclusion of Theorem 12 tells us that the DropEdge+GCN model is a lazy walk on the graph. Equation (9) is intuitively equivalent to that the message of node \( u \) passes with the probability \( 1 - \frac{1}{|E|\deg(u)} \), and stays in node \( u \) with probability \( \frac{1}{|E|\deg(u)} \).

In fact, equation (9) is a more generalized form of lazy walk on the graph. In particular, if \( G \) is a regular graph, i.e., a graph with the same degree of each node. Then the matrix \( \Gamma := \text{diag}\left(\frac{1}{|E|\deg(1)}, \frac{1}{|E|\deg(2)}, \ldots, \frac{1}{|E|\deg(N)}\right) \) degenerates to the constant \( \frac{1}{|E|\deg(u)} \). Then the message passing of the DropEdge+GCN model is a lazy walk with the parameter \( \frac{1}{|E|\deg(u)} \) on the graph.

We have already modeled residual connections method, PPNP and DropEdge method uniformly as lazy walk on graphs, and these three methods have gradually generalized lazy walks from special to general. Next we will use the mixing time theory as a mathematical tool to analyze the property of the lazy walk on the graph. This is used to illustrate the effectiveness of these methods to alleviate the over-smoothing problem.

The following theorem illustrates that starting from the any initial distribution, the lazy walk on the graph moves to the stationary distribution more slowly than the simple random walk.

**Theorem 13** Let

\[
P_{rw} := D^{-1}A
\]

be the transition matrix of a simple random walk \( \tilde{V}_{rw} \) on the graph \( G = (V, E) \),

\[
P_{\text{lasy}} := (1 - \gamma)D^{-1}A + \gamma I
\]

be the transition matrix of the lazy walk \( \tilde{V}_{lasy} \) on the graph \( G = (V, E) \). If \( \tilde{V}_{rw} \) and \( \tilde{V}_{lasy} \) move from any distribution on \( V \), then there is the following conclusion:

(1) \( \tilde{V} \) and \( \tilde{V}_{lasy} \) have the same stationary distribution \( \pi \), where

\[
\pi(u) = \frac{\deg(u)}{2|E|}, \quad \forall u \in V.
\]

(2) For all \( l > 0 \),

\[
\max_{u \in V} \|P_{lasy}^l(u, \cdot) - \pi\|_{TV} \geq \max_{u \in V} \|P_{rw}^l(u, \cdot) - \pi\|_{TV},
\]

where \( \| \cdot \|_{TV} \) is the total variation distance defined in Section 2. \( P(u, \cdot) \) is the row vector corresponding to the \( u \)th row of the matrix \( P \). 

17
Proof.

(1) For $V_{rw}$, for all $v \in V$, since
\begin{align*}
\sum_{u \in V} \pi(u) p_{rw}(u, v) &= \sum_{(u,v) \in E} \frac{\deg(u)}{2|E|} \frac{1}{\deg(u)} \\
&= \frac{\deg(v)}{2|E|} \\
&= \pi(v),
\end{align*}
$\pi$ is a stationary distribution of $V_{rw}$. For $V_{lazy}$, $\forall v \in V$, since
\begin{align*}
\sum_{u \in V} \pi(u) p_{lazy}(u, v) &= \sum_{(u,v) \in E} \frac{\deg(u)(1 - \gamma)}{2|E|} \frac{1}{\deg(u)} + \frac{\deg(v)}{2|E|} \gamma \\
&= \frac{\deg(v)}{2|E|} (1 - \gamma) + \frac{\deg(v)}{2|E|} \gamma \\
&= \frac{\deg(v)}{2|E|} \\
&= \pi(v),
\end{align*}
$\pi$ also is a stationary distribution of $V_{lazy}$.

(2) Notice the intuitive definition of $P^l_{lazy}$,
\begin{align*}
P^l_{lazy} = \sum_{i=0}^{l} C_i^l \gamma^{l-i} (1 - \gamma)^i P^i_{rw}.
\end{align*}
That is, each step transition matrix $P_{lazy}$ is a identity matrix $I$ with independent probability $\gamma$. And for all $i < l$, by the proposition 10 we have
\begin{align*}
\max_{u \in V} \|P^i_{rw}(u, \cdot) - \pi\|_{TV} \geq \max_{u \in V} \|P^i_{rw}(u, \cdot) - \pi\|_{TV}.
\end{align*}
Then
\begin{align*}
\max_{u \in V} \|P^l_{lazy}(u, \cdot) - \pi\|_{TV} = \max_{u \in V} \| \sum_{i=0}^{l} C_i^l \gamma^{l-i} (1 - \gamma)^i P^i_{rw}(u, \cdot) - \pi\|_{TV} \\
= \max_{u \in V} \| \sum_{i=0}^{l} C_i^l \gamma^{l-i} (1 - \gamma)^i P^i_{rw}(u, \cdot) - \pi \|_{TV} \\
= \max_{u \in V} \| \sum_{i=0}^{l} C_i^l \gamma^{l-i} (1 - \gamma)^i \| \pi_{TV} \\
= \max_{u \in V} \| \sum_{i=0}^{l} C_i^l \gamma^{l-i} (1 - \gamma)^i \| P^l_{rw}(u, \cdot) - \pi_{TV} \\
\geq \sum_{i=0}^{l} C_i^l \gamma^{l-i} (1 - \gamma)^i \max_{u \in V} \| P^l_{rw}(u, \cdot) - \pi_{TV} \\
= \max_{u \in V} \| P^l_{rw}(u, \cdot) - \pi_{TV} \\
\end{align*}
Notice the definition of mixing time in Section 2

\[ t_{\text{mix}}(\epsilon) := \min \{ t : d(t) \leq \epsilon \}, \quad d(t) := \max_{u \in V} \| P^t(u, \cdot) - \pi \|_{TV}. \]

Theorem 13 shows that the mixing time for a lazy walk on the graph moving to the stationary distribution is longer than that for a simple random walk. This result is very intuitive: the time required for a lazy walk to converge to the stationary distribution is naturally larger than that of a random walk.

Back to the over-smoothing problem in GCN, in Section 4.1, we interpret over-smoothing as the convergence of the probability distribution to the stationary distribution with the move of the Markov chain. Theorem 13 shows that the residual connections method, PPNP and DropEdge method which can be modeled as lazy walks on the graph can indeed slow down the convergence of the GCN model to over-smoothing.

We have shown that methods based on the lazy walk can slow down the rate of GCN model over-smoothing. There is another problem that these methods avoid the over-smoothing. The answer is no. The following Theorem 14 shows that for a lazy walk on the graph, the probability distribution on the node set still converges to the stationary distribution, and the rate of convergence is exponential as with a simple random walk on the graph.

**Theorem 14** Let \( L := I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \) be the normalized Laplacian matrix of the graph \( G \), \( \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N \) be the eigenvalues of \( L \), \( \phi_1, \phi_2, \cdots, \phi_N \) are the corresponding eigenvectors of the corresponding eigenvalues. Then for any initial distribution \( \mu : V \to \mathbb{R} \), \( l > 0 \),

\[
\begin{align*}
\mu P^l_{\text{rw}} &= \pi + \sum_{k=2}^{N} (1 - \lambda_k)^l a_k \phi_k D^{\frac{1}{2}}, \\
\mu P^l_{\text{lazy}} &= \pi + \sum_{k=2}^{N} (1 - (1 - \gamma)\lambda_k)^l a_k \phi_k D^{\frac{1}{2}},
\end{align*}
\]

where \( a_k, k = 1, 2, \ldots, N \) are the coordinates of the vector \( \mu D^{-\frac{1}{2}} \) on the basis \( (\phi_1, \phi_2, \cdots, \phi_N) \), i.e. \( \mu D^{-\frac{1}{2}} = \sum_{k=1}^{N} a_k \phi_k \). Further, by the Frobenius-Perron Theorem, we have \( 0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N \leq 2 \), so \( \mu P^l \) and \( \mu P^l_{\text{lazy}} \) both converge to \( \pi \) with \( l \) exponentially.

**Proof.**

\[
\begin{align*}
P^l_{\text{rw}} &= (D^{-1}A)^l \\
&= D^{-\frac{1}{2}} (D^{-\frac{1}{2}}AD^{-\frac{1}{2}})^l D^{\frac{1}{2}} \\
&= D^{-\frac{1}{2}} (I - L)^l D^{\frac{1}{2}}.
\end{align*}
\]

Then

\[
\begin{align*}
\mu P^l_{\text{rw}} &= \mu D^{-\frac{1}{2}} (I - L)^l D^{\frac{1}{2}} \\
&= \sum_{k=1}^{N} (1 - \lambda_k)^l a_k \phi_k D^{\frac{1}{2}} \\
&= \pi + \sum_{k=2}^{N} (1 - \lambda_k)^l a_k \phi_k D^{\frac{1}{2}},
\end{align*}
\]
where $\pi = a_1\phi_1D^{\frac{1}{2}}$ is concluded from Chung and Graham (1997). On the other hand, the corresponding normalized Laplacian matrix of $P_{\text{lazy}}$ is

$$L_{\text{lazy}} = I - D_{\text{lazy}}^{-\frac{1}{2}}D_{\text{lazy}}^{-\frac{1}{2}}A_{\text{lazy}}D_{\text{lazy}}^{-\frac{1}{2}}A_{\text{lazy}}D_{\text{lazy}}^{-\frac{1}{2}} = I - D_{\text{lazy}}^{-\frac{1}{2}}(1 - \gamma)A + \gamma D)D_{\text{lazy}}^{-\frac{1}{2}} = I - (1 - \gamma)D^{-\frac{1}{2}}AD^{-\frac{1}{2}} - \gamma I = (1 - \gamma)L.$$  

Therefore, the eigenvalue of $L_{\text{lazy}}$ is $0 = (1 - \gamma)\lambda_1 \leq (1 - \gamma)\lambda_2 \leq \cdots \leq (1 - \gamma)\lambda_N < 2$, and the eigenvector remains $\phi_1, \phi_2, \cdots, \phi_N$. In the same way as $\mu_{\text{P}}^{\text{rw}}$, there are

$$\mu_{\text{P}}^{\text{P}}_{\text{lazy}} = \pi + \sum_{k=2}^{N}(1 - (1 - \gamma)\lambda_k)^{\frac{1}{2}}a_k\phi_k D_{\text{lazy}}^{\frac{1}{2}}.$$  

In this subsection, we use Theorem 13 to show the effectiveness of the residual connections method, PPNP and DropEdge method to alleviate the over-smoothing problem. It is further shown by Theorem 14 that these methods can neither make the GCN model avoid over-smoothing nor can they make the GCN model avoid over-smoothing at the exponential rate. Then there are problems, can the operator-consistent GNN model avoid over-smoothing? Or can it avoid over-smoothing at the exponential rate? We will discuss these two questions in detail in the next section.

### 4.3 Conclusion of the operator-consistent GNN model

In Section 4.2, we study several major methods to alleviate over-smoothing, model them uniformly as lazy walks on the graph, and rigorously demonstrate the effectiveness of these methods to alleviate the over-smoothing problem using the mixing time theory of Markov chains. However, we also point out that these methods cannot avoid the GCN model from converging to over-smoothing, but only reduce the rate at which the model converges to over-smoothing, and this rate remains exponential. In this section, we discuss following two core issues

- Can operator-consistent GNN models avoid over-smoothing?
- If over-smoothing cannot be avoided, can operator-consistent GNN models avoid over-smoothing at the exponential rate?

The answer to both questions is no. The following Theorem 15 answers the first question by showing that the stationary distribution must exist for the transition matrix of time-homogeneous random walk on the graph. See Appendix A for proof.

**Theorem 15** Let $P$ be a transition matrix of random walk on the connected, non-bipartite graph $G$. Then there exists a unique probability distribution $\pi$ over $V$ that satisfies

$$\pi = \pi P.$$
Corollary 16 Under the conditions of Theorem 15, the transition matrix \( P = (p(u, v), u, v \in \mathcal{V}) \) of the random walk on the graph can be expressed formally as

\[
p(u, v) = \frac{r(u, v)}{\sum_{z \in \mathcal{N}(u)} r(u, z)},
\]

where \( r(u, v) > 0 \). Then its unique stationary distribution is

\[
\pi(u) = \frac{\text{deg}(u)}{\sum_{k \in \mathcal{V}} \text{deg}(k)},
\]

where \( \text{deg}(u) = \sum_{z \in \mathcal{N}(u)} r(u, z) \).

Proof. For all \( v \in \mathcal{V} \), since

\[
\sum_{u \in \mathcal{V}} \pi(u)p(u, v) = \sum_{(u, v) \in \mathcal{E}} \frac{\text{deg}(u)}{\sum_{k \in \mathcal{V}} \text{deg}(k)} \frac{r(u, v)}{\sum_{z \in \mathcal{N}(u)} r(u, z)} = \frac{\text{deg}(v)}{\sum_{k \in \mathcal{V}} \text{deg}(k)} = \pi(v),
\]

\( \pi \) is a stationary distribution of \( P \).

Theorem 15 shows that there must be a stationary distribution for the message passing operator on the graph. Then distribution on \( \mathcal{V} \) starting from any initial distribution will converge to a stationary distribution \( \pi \). According to the discussion of the stationary distribution and over-smoothing problem in Section 4.1, for an operator-consistent GNN, regardless of the initial input, the features of the nodes over-smoothing as the GNN propagates forward. Combining the above discussion, the operator-consistent GNN cannot avoid over-smoothing.

The following Theorem 17 answers the second question by showing that the time-homogeneous random walk on the graph will converge at the exponential rate to its stationary distributions \( \pi \). See Appendix A for proof.

Theorem 17 Under the condition of Theorem 15, there exist constants \( \alpha \in (0, 1) \) and \( C > 0 \) such that

\[
\max_{u \in \mathcal{V}} \| P^t(u, \cdot) - \pi \|_{TV} \leq C \alpha^t,
\]

where \( P(u, \cdot) \) denotes the row vector consisting of the \( i \)th row of the transition matrix \( P \).

Notice the definition of mixing time in Section 2

\[
t_{mix}(\epsilon) := \min\{t : d(t) \leq \epsilon\}, \quad d(t) := \max_{u \in \mathcal{V}} \| P^t(u, \cdot) - \pi \|_{TV}.
\]
Returning to the GNN model, Theorem 17 shows that under the mixing time theory, the rate at which the operator-consistent GNN model converges to over-smoothing is exponential. So as long as the message passing operators of each layer of the GNN model are consistent, the GNN model cannot avoid over-smoothing at the exponential rate.

In Markovian sense, we give the general conclusion on the problem of operator-consistent GNN model over-smoothing. The operator-consistent GNN model can neither avoid over-smoothing nor can it avoid over-smoothing at the exponential rate. For operator-consistent GNN models, we can only alleviate but not avoid over-smoothing. However, is there a similar conclusion for operator-inconsistent GNN models? We discuss this in detail in Section 4.4.

### 4.4 Conclusion of the operator-inconsistent GNN models

In this subsection, we give a sufficient condition for the operator-inconsistent GNN models to avoid over-smoothing. In addition, experimental verification is performed in Section 5.

Previously, Wang et al. (2019a) discussed the over-smoothing problem of GAT. And they concluded that GAT will over-smooth. Similar to our modeling process, they view the GAT operator \( P^{(l)}_{\text{att}} \) at each layer as a transition matrix of a random walk on the graph. However, they ignore the fact that the complete forward propagation process of GAT is a time-inhomogeneous random walk on the graph. Their proof that the GAT will over-smooth relies on the stationary distribution \( \pi^{(l)} \) of the GAT operator \( P^{(l)}_{\text{att}} \) being consistent for each layer, i.e.

\[
\pi^{(1)} = \pi^{(2)} = \cdots = \pi^{(l)} = \cdots.
\]

However, by Corollary 16, for the GAT operator \( P^{(l)}_{\text{att}} \)

\[
\pi^{(l)}(u) = \frac{\text{deg}^{(l)}(u)}{\sum_{v \in V} \text{deg}^{(l)}(v)} \quad \forall u \in V,
\]

where

\[
\text{deg}^{(l)}(u) := \sum_{k \in N(u)} \exp(\phi^{(l)}(h_u^{(l-1)}, h_k^{(l-1)}))
\]

is the weighted degree of \( u \), and

\[
\phi^{(l)}(h_u^{(l-1)}, h_k^{(l-1)}) := \text{LeakyReLU}(a^T [W^{(l)} h_u^{(l-1)} \parallel W^{(l)} h_k^{(l-1)}]).
\]

Since the weighted degree of \( u \) is different between each layer, then

\[
\pi^{(1)} \neq \pi^{(2)} \neq \cdots \neq \pi^{(l)} \neq \cdots.
\]

The proof of Wang et al. (2019a) is flawed. Thus the conclusion that GAT will over-smoothing in the Markovian sense is also incorrect.

We have modeled the graph attention model in Section 3.3 as a time-inhomogeneous Markov chain \( \tilde{V}_{\text{att}} \) with finite states on the graph. Then by the Dobrushin-Isaacson-Madsen theorem (Theorem 4), considering the GAT operator \( \{ P^{(l)}_{\text{att}} \} \) as a family of transition matrices can only satisfy the condition (1), which neither guarantees the condition (2) nor
the Isaacson-Madsen condition or the Dobrushin condition. So the time-inhomogeneous Markov chain $\vec{V}_{att}$ does not necessarily have a stationary distribution.

In Section 4.1, we interpreted over-smoothing problem of the GNN model as the convergence of the node feature distribution to a stationary distribution with the move of the Markov chain on the graph. Since the time-inhomogeneous Markov chain $\vec{V}_{att}$ does not necessarily have a stationary distribution in the limit sense, this suggests that GAT does not necessarily oversmooth.

We give the conclusion that the operator-consistent GNN model cannot avoid over-smoothing at the exponential rate in Section 4.3. GAT is used as example to show that operator-inconsistent GNNs do not necessarily oversmooth. To further refine the discussion of this problem, we propose a necessary condition for the existence of the stationary distribution (limiting sense) for the time-inhomogeneous Markov chain. And a sufficient condition is developed to ensure that GAT avoid over-smoothing.

**Theorem 18** Let $\vec{X} = \{X_n, n \in T\}$ be a time-inhomogeneous Markov chain on a finite state space $E$, and write its $n$th-step transition matrix as $P^{(n)}$, satisfying that, $P^{(n)}$ is irreducible and aperiodic, there exists a unique stationary distribution $\pi^{(n)}$ as the time-homogeneous transition matrix, and $C(P^{(n)}) < 1$. Let the initial distribution be $\mu_0$ and the distribution of the chain $\vec{X}$ at step $n$ be $\mu_n := \mu_{n-1}P^{(n)}$. Then the necessary condition for there exists a probability measure $\pi$ on $E$ such that $\|\mu_n - \pi\| \to 0, \ n \to \infty$ is

$$\|\pi^{(n)} - \pi\| \to 0, \ n \to \infty.$$  

**Proof.**

Suppose there exists a probability measure $\pi$ on $E$ such that $\|\mu_n - \pi\| \to 0, \ n \to \infty$. The following conclusion

$$\|\pi^{(n)} - \mu_{n-1}\| \to 0, \ n \to \infty$$

is proved by contradiction. If for any $N \in \mathbb{N}^+$, there exists $\delta > 0$, when $n > N$, all have

$$\|\pi^{(n)} - \mu_{n-1}\| > \delta.$$ 

Then by the triangle inequality and the Dobrushin inequality (Lemma 2)

$$\|\mu_n - \mu_{n-1}\| = \|((\pi^{(n)} - \mu_{n-1}) - (\pi^{(n)} - \mu_n))\|
\geq \|\pi^{(n)} - \mu_{n-1}\| - \|\pi^{(n)} - \mu_n\|
= \|\pi^{(n)} - \mu_{n-1}\| - \|\pi^{(n)}P^{(n)} - \mu_{n-1}P^{(n)}\|
\geq \|\pi^{(n)} - \mu_{n-1}\| - C(P^{(n)})\|\pi^{(n)} - \mu_{n-1}\|
= (1 - C(P^{(n)}))\|\pi^{(n)} - \mu_{n-1}\|
> (1 - C(P^{(n)}))\delta.$$ 

And since $C(P^{(n)}) < 1$, then for any $N \in \mathbb{N}^+$, there exists $(1 - C(P^{(n)}))\delta > 0$, for all $n > N$,

$$\|\mu_n - \mu_{n-1}\| > (1 - C(P^{(n)}))\delta.$$
By Cauchy’s convergence test, it is contradictory to
\[ \|\mu_n - \pi\| \to 0, \ n \to \infty. \]
Thus for any \( \epsilon > 0 \), there exists \( N_1 \in \mathbb{N}^+ \), and when \( n > N_1 \),
\[ \|\pi^{(n)} - \mu_{n-1}\| < \frac{\epsilon}{2}. \]
Since \( \|\mu_n - \pi\| \to 0, \ n \to \infty \), there exists \( N_2 \in \mathbb{N}^+ \), for all \( n > N_2 \),
\[ \|\mu_{n-1} - \pi\| < \frac{\epsilon}{2}. \]
Taking \( N = \max\{N_1, N_2\} \), when \( n > N \), we have
\[
\|\pi^{(n)} - \pi\| = \|\pi^{(n)} - \mu_{n-1}\| + \|\mu_{n-1} - \pi\|
\leq \|\pi^{(n)} - \mu_{n-1}\| + \|\mu_{n-1} - \pi\|
< \epsilon.
\]
Then
\[ \|\pi^{(n)} - \pi\| \to 0, \ n \to \infty. \]

This necessary condition is very intuitive. In the limit sense, the transition of \( \mu_{n-1} \) satisfies
\[ \lim_{n \to \infty} \mu_{n-1}P^{(n)} = \lim_{n \to \infty} \mu_n = \lim_{n \to \infty} \mu_{n-1} = \pi. \]
On the other hand, for any \( n > 0 \), \( \pi^{(n)} \) is the only solution of the equation
\[ \mu = \mu P^{(n)}. \]
Naturally,
\[ \lim_{n \to \infty} \pi^{(n)} = \lim_{n \to \infty} \mu_{n-1} = \pi. \]

According to Theorem 18, we propose the following corollary which give a sufficient condition to ensure that GAT avoid over-smoothing in Markovian sense.

**Corollary 19 (Sufficient condition for GAT to avoid over-smoothing)** Let \( h^{(l)}_u \) be the \( l \)th hidden layer feature of node \( u \in \mathcal{V} \) in GAT, then a sufficient condition for GAT to avoid over-smoothing is that there exists a hyperparameter \( \delta > 0 \) such that for any \( l \geq 2 \), satisfying
\[ \|h^{(l-1)}_u - h^{(l)}_u\| > \delta. \] (10)

**Proof.** By Corollary 16, for the GAT operator \( P^{(l)}_{\text{att}} \),
\[ \pi^{(l)}(u) = \frac{\deg^{(l)}(u)}{\sum_{v \in \mathcal{V}} \deg^{(l)}(v)} \forall u \in \mathcal{V}, \]
where \( \text{deg}^{(l)}(u) := \sum_{k \in \mathcal{N}(u)} \exp(\phi^{(l)}(h_{u}^{(l-1)}, h_{k}^{(l-1)})) \) is the weighted degree of \( u \), where

\[
\phi^{(l)}(h_{u}^{(l-1)}, h_{k}^{(l-1)}) := \text{LeakyReLU}(a^T[W^{(l)} h_{u}^{(l-1)}||W^{(l)} h_{k}^{(l-1)}]).
\]

Since \( \mathcal{G} \) is connected, non-bipartite graph,

\[
C(P_{\text{att}}^{(l)}) < 1.
\]

By Theorem 18, the sufficient condition for that there is no probability measure \( \pi \) on \( E \) such that

\[
\|\mu_n - \pi\| \to 0, \quad n \to \infty
\]

is

\[
\|\pi^{(n)} - \pi\| \to 0, \quad n \to \infty.
\]

By the Cauchy’s convergence test, it is equivalent to the existence of \( \delta_{\pi} > 0 \) such that for any \( l \geq 1 \), satisfying

\[
\|\pi^{(l)} - \pi^{(l+1)}\| > \delta_{\pi}.
\]

Let \( D^{(l)} := \sum_{u \in \mathcal{V}} \text{deg}^{(l)}(u) \), \( D_{\text{min}} = \min\{D^{(l)}, D^{(l+1)}\} \),

\[
\|\pi^{(l)} - \pi^{(l+1)}\| = \sum_{u \in \mathcal{V}} \left| \frac{\text{deg}^{(l)}(u)}{D^{(l)}} - \frac{\text{deg}^{(l+1)}(u)}{D^{(l+1)}} \right|
\]

\[
> \left| \frac{\text{deg}^{(l)}(u)}{D^{(l)}} - \frac{\text{deg}^{(l+1)}(u)}{D^{(l+1)}} \right|
\]

\[
> \frac{1}{D_{\text{min}}} \left| \text{deg}^{(l)}(u) - \text{deg}^{(l+1)}(u) \right|
\]

and

\[
\text{deg}^{(l)}(u) := \sum_{k \in \mathcal{N}(u)} \exp(\text{LeakyReLU}(a^T[W^{(l)} h_{u}^{(l-1)}||W^{(l)} h_{k}^{(l-1)}])).
\]

Then if there exists \( \delta > 0 \) such that for any \( l \geq 2 \), satisfying

\[
\|h_{u}^{(l-1)} - h_{u}^{(l)}\| > \delta,
\]

there must exist \( \delta_{\pi} > 0 \) such that for any \( l \geq 1 \), satisfying

\[
\|\pi^{(l)} - \pi^{(l+1)}\| > \delta_{\pi}.
\]

This sufficient condition is intuitive. The essence of over-smoothing is that the node features converge with the propagation of the network. By Cauchy’s convergence test, the condition (10) exactly avoid features \( h_{u}^{(l)} \) of the node \( u \) from converging as network deepens.
5. Experiments

In this section, we experimentally verify the correctness of the theoretical analysis. Since we do not aim to refresh State of the Arts (SOTA), we verify that the sufficient conditions in Section 4.4 can indeed avoid over-smoothing and improve the performance of GAT, while keeping the other hyperparameters (network structure, learning rate, dropout, epoch, etc.) the same (note that this is not necessarily the optimal hyperparameter). We also conduct experiments on GEN-SoftMax (Li et al., 2020) and we leave this part in Appendix B.

5.1 Setup

In this section we briefly introduce the settings in our experiments; see Appendix C for more specific settings.

**Variant of sufficient condition.** Notice that the sufficient condition in the Corollary 19 is in the form of inequality, which is not conducive to conducting experiments. In the concrete implementation, let $h_u^{(l)}$ be the $l$th hidden layer feature on node $u \in \mathcal{V}$. We normalize the distance of the node features between two adjacent layers and then let it equal to a given hyperparameter threshold $T \in (0,1)$, i.e., for the GNN model with $n$ layers:

$$\left( \frac{1}{n} \sum_{l=0}^{n-1} (\| \text{Sigmoid}(h_u^{(l)}) - \text{Sigmoid}(h_u^{(l+1)}) \| - T) \right)^2 = 0. \quad (11)$$

Since there must exist $\delta > 0$ that satisfies $T > \delta$, equation (10) can be satisfied. For a detailed discussion of the threshold, we put it in Appendix C.

**Datasets.** In terms of datasets, we follow the datasets used in the original work of GAT (Velićković et al., 2018). We use three standard benchmark datasets - Cora, Citeseer, and Pubmed (Sen et al., 2008), covering the basic transductive learning tasks.

**Implementation Details.** For the specific implementation, we refer to the open-source code of vanilla GAT, and models with different layers share the same settings: We use the Adam SGD optimizer (Kingma and Ba, 2015) with learning rate 0.01, the hidden dimension is 64, each GAT layer has 8 heads and the amount of training epoch is 500. All experiments are conducted on a single Nvidia Tesla v100.

5.2 Results of GAT

To keep statistical confidence, we repeat all experiments 5 times and record the mean value and standard deviation. Results shown in Table 1 demonstrate that almost on each dataset and number of layers, GAT-OI will gain a improvement in the performance. Specifically, on Cora and Citeseer, GAT’s performance begins to decrease drastically when layer numbers surpass 6 and 5 but GAT-OI can relieve this trend in some way. On Pubmed, vanilla GAT’s performance has a gradual decline, however, GAT-OI’s performance keeps competitive for all layer numbers.
Table 1: Results of GAT

| datasets | model       | #layers |
|----------|-------------|---------|
|          |             | 3       | 4       | 5       | 6       | 7       | 8       |
| Cora     | GAT         | 0.7773±0.0054 | 0.7602±0.0166 | 0.4821±0.3021 | 0.2774±0.2542 | 0.1672±0.0780 | 0.0958±0.0059 |
|          | GAT-OI      | 0.7884±0.0157 | 0.7872±0.0127 | 0.7648±0.0077 | 0.6454±0.2508 | 0.3244±0.2465 | 0.1678±0.0756 |
| Citeseer | GAT         | 0.6643±0.0063 | 0.6541±0.0076 | 0.3472±0.2582 | 0.2474±0.1947 | 0.1768±0.0064 | 0.1902±0.0598 |
|          | GAT-OI      | 0.6678±0.0157 | 0.6692±0.0072 | 0.6208±0.0380 | 0.2706±0.1884 | 0.1915±0.0200 | 0.1864±0.0299 |
| Pubmed   | GAT         | 0.7616±0.0115 | 0.7534±0.0114 | 0.7653±0.0072 | 0.7468±0.0084 | 0.7468±0.0045 | 0.7076±0.0112 |
|          | GAT-OI      | 0.7673±0.0064 | 0.7659±0.0123 | 0.7684±0.0063 | 0.7664±0.0092 | 0.7596±0.0107 | 0.7618±0.0114 |

5.3 Verification of avoiding over-smoothing

In this subsection, we further show that the sufficient conditions in Section 4.4 not only improve the performance of the model but also do avoid the over-smoothing problem. Since the neural network is a black-box model, we cannot explicitly compute the stationary distribution of the graph neural network when it is over-smoothed. Therefore we measure the degree of over-smoothing by calculating the standard deviation of each node’s representation at each layer. A lower value implies more severe over-smoothing. Results shown in Fig. 1-3 demonstrate that the node representations obtained from GAT-OI are more diverse than those from GAT, which means the remission of over-smoothing. It’s also interesting that there is an accordance between the performance and over-smoothing, for example on Cora dataset, the performance would have a huge decrease when the number of layers is larger than 5, Fig. 1 shows the over-smoothing phenomenon is severe at the same time. Also on Pubmed dataset, the performance is relatively stable and the corresponding Fig. 3 shows that the model trained on this dataset suffers from over-smoothing lightly. These results enlighten us that over-smoothing may be caused by various objective reasons, e.g. the property of the dataset, and GAT-OI can relieve this negative effect to some extent.

![Figure 1](image1.png)

Figure 1: Measurement of over-smoothing of GAT on Cora. Orange curve indicates the results of GAT-OI and blue curve indicates the results of vanilla GAT

![Figure 2](image2.png)

Figure 2: Measurement of over-smoothing of GAT on Citeseer. Orange curve indicates the results of GAT-OI and blue curve indicates the results of vanilla GAT
Figure 3: Measurement of over-smoothing of GAT on Pubmed. Orange curve indicates the results of GAT-OI and blue curve indicates the results of vanilla GAT.

6. Conclusion

This article provides a theoretical tool for explaining and analyzing GNNs by modeling the forward propagation process of GNNs as Markov chains on graphs. We model GCN as a simple random walk on the graph, GAT as a time-inhomogeneous Markov chain on the graph and the GNN using the DropEdge method as the Markov chain in random environment. We connect Markov chains with GNNs in the hope that we can solve some problems in the field of GNNs through the study of Markov chains, and inspire more scholars to try to analyze from the Markov perspective when studying GNNs, as well as to design high-performance GNN models by the guidance of Markov chains theory.

We study the over-smoothing which is an important problem limiting the development of GNNs. We attribute the over-smoothing problem to the convergence of the probability distribution over the node set to the stationary distribution. Using results from the study of Markov chains, we prove a series of important conclusions containing the effectiveness of the methods to alleviate the over-smoothing problem, the inability of operator-consistent GNNs to avoid over-smoothing at an exponential rate, and the sufficient condition for operator-inconsistent GNNs to avoid over-smoothing in the Markovian sense. Finally, according to the experiments we designed, the proposed sufficient condition can indeed improve the performance of operator-inconsistent GNNs by solving the over-smoothing problem.

Acknowledgments

This paper is supported by National key research and development program of China (2021YFA1000403). And it is supported by the National Natural Science Foundation of China (Nos.11991022, U19B2040), and the Strategic Priority Research Program of Chinese Academy of Sciences (No.XDA27000000) and the Fundamental Research Funds for the Central Universities.
Appendix A. Additional proofs

In this appendix we prove the following conclusions. Some of these proofs refer to books (Levin and Peres, 2017; Gong and Qian, 2004; Hu, 2005)

Lemma 2 Let $\mu$ and $\nu$ be probability distributions on a finite state space $E$ and $P$ be a transition matrix, then

$$\|\mu P - \nu P\| \leq C(P) \|\mu - \nu\|,$$

where

$$C(P) := \frac{1}{2} \sup_{i,j} \sum_{k \in E} |p(i,k) - p(j,k)|.$$

Proof Denote the positive and negative parts of $a$ as $a^+$ and $a^-$ respectively. Then denote

$$\rho^+_k = (\mu(k) - \nu(k))^+ / \frac{1}{2} \|\mu - \nu\|, \quad \rho^-_k = (\mu(k) - \nu(k))^- / \frac{1}{2} \|\mu - \nu\|.$$

Since

$$\sum_k [(\mu(k) - \nu(k))^+ - (\mu(k) - \nu(k))^-] = \sum_k (\mu(k) - \nu(k)) = 0,$$

$$\sum_k [(\mu(k) - \nu(k))^+ + (\mu(k) - \nu(k))^+] = \sum_k |\mu(k) - \nu(k)| = \|\mu - \nu\|,$$

then

$$\sum_k \rho^+_k = \sum_k \rho^-_k = \frac{2}{\|\mu - \nu\|} \sum_k (\mu(k) - \nu(k))^+ = 1$$

and

$$\mu(k) - \nu(k) = (\mu(k) - \nu(k))^+ - (\mu(k) - \nu(k))^- = \frac{1}{2} (\rho^+_k - \rho^-_k) \|\mu - \nu\|.$$

Thus

$$\|\mu P - \nu P\| = \sum_k \left| \sum_i p(i,k) \cdot (\mu(i) - \nu(i)) \right|$$

$$= \frac{1}{2} \|\mu - \nu\| \sum_k \left| \sum_i p(i,k) \cdot (\rho^+_i - \rho^-_i) \right|$$

$$= \frac{1}{2} \|\mu - \nu\| \sum_k \left| \sum_i \sum_j p(i,k) - p(j,k) \cdot (\rho^+_i \rho^-_j) \right|$$

$$\leq \|\mu - \nu\| \sum_i \sum_j (\rho^+_i \rho^-_j) \sum_k \left( \frac{1}{2} |p(i,k) - p(j,k)| \right)$$

$$\leq C(P) \|\mu - \nu\|.$$

Theorem 3 If one of the following two conditions is satisfied
(1) $P$ is irreducible and aperiodic.

(2) $C(P) < 1$.

Then there exist stationary distribution $\pi$, constants $\alpha \in (0, 1)$ and $C > 0$ such that

$$\max_{i \in E} \|P^n(i, \cdot) - \pi\| \leq C \alpha^n,$$

where $P(i, \cdot)$ denotes the row vector consisting of the $i$th row of the transition matrix $P$ and $\| \cdot \|$ denotes the $L^1$ norm.

**Proof**

(1) If $P$ is irreducible and aperiodic.

Since $P$ is irreducible and aperiodic, there exists an $r$ such that $P^r$ has strictly positive entries. Let $\Pi$ be the matrix with $N$ rows, each of which is the row vector $\pi$. For sufficiently small $\delta > 0$, we have

$$p^r(i, j) \geq \delta \pi(j), \quad \forall i, j \in E.$$

Let $\theta = 1 - \delta$. The equation

$$P^r = (1 - \theta)\Pi + \theta Q$$

(A1)

defines a stochastic matrix $Q$.

It is a straightforward computation to check that $M \Pi = \Pi$ for any stochastic matrix $M$ and that $\Pi M = \Pi$ for any matrix $M$ such that $\pi M = \pi$.

Next, we use induction to demonstrate that

$$P^{rk} = (1 - \theta^k)\Pi + \theta^k Q^k$$

(A2)

for $k \geq 1$. If $k = 1$, this holds by (A1). Assuming that (A2) holds for $k = n$,

$$P^{r(n+1)} = P^n P^r = [(1 - \theta^n)\Pi + \theta^n Q^n]P^r.$$

Distributing and expanding $P^r$ in the second term (using (A1)) gives

$$P^{r(n+1)} = [1 - \theta^n]\Pi P^r + (1 - \theta)\theta^n Q^n \Pi + \theta^{n+1} Q^n Q.$$

Using that $\Pi P^r = \Pi$ and $Q^n \Pi = \Pi$ shows that

$$P^{r(n+1)} = [1 - \theta^n]\Pi + \theta^{n+1} Q^{n+1}.$$

This establishes (A2) for $k = n + 1$ (assuming it holds for $k = n$), and hence it holds for all $k$.

Multiplying by $P^l$ and rearranging terms now yields

$$P^{rk+l} - \Pi = \theta^k(Q^k P^l - \Pi).$$

(A3)
To complete the proof, sum the absolute values of the elements in row $i_0$ on both sides of (A3). On the right, the second factor is at most the largest possible total variation distance between distributions, which is 1. Hence for any $i_0$ we have

$$\|P^{r_k+1}(i_0, \cdot) - \pi\| \leq \theta^k.$$

Taking $\alpha = \theta^{1/r}$ and $C = 1/\theta$ finishes the proof.

(2) If $C(P) < 1$.

For all probability distributions $\mu$ and $\nu$ on a finite state space $E$. Using Dobrushin’s inequality (lemma 2) yields

$$\| \sum_i \mu(i) P^n(i, \cdot) - \sum_i \nu(i) P^n(i, \cdot) \| = \| \sum_{i,j} P^{n-1}(i,j) P(j, \cdot) (\mu(i) - \nu(i)) \|
\leq C(P) \| \sum_i p_{n-1}(i,j) \mu(i) - \sum_i p_{n-1}(i,j) \nu(i) \|
\leq \ldots
\leq (C(P))^n \| \mu - \nu \|.$$

By Banach fixed-point theorem (contraction mapping principle), there exists a probability distribution $\pi$, for all probability distribution $\mu$

$$\| \sum_i \mu(i) P^n(i, \cdot) - \pi \| \leq (C(P))^n \| \mu - \nu \|.$$

Taking $\mu(i) = p(k,i)$, $\alpha = C(P)$ and $C = 2$. Then for all $k \in E$

$$\sum_i |p^n(k,i) - \pi(i)| = \|P^n(k, \cdot) - \pi\| \leq C \alpha^n.$$

Theorem 4 Let $\vec{X} = \{X_n, n \in T\}$ be a time-inhomogeneous Markov chain on a finite state space $E$ with transition matrix $P^{(n)}$. If the following (1), (2) and (3A) or (3B) are satisfied

(1) There exists a stationary distribution $\pi^{(n)}$ when $P^{(n)}$ is treated as a transition matrix of a time-homogeneous chain;

(2) $\sum_n \| \pi^{(n)} - \pi^{(n+1)} \| < \infty$;

(3A) Isaacson-Madsen condition: for any probability distribution $\mu$, $\nu$ on $E$ and positive integer $k$

$$\| (\mu - \nu) P^{(k)} \ldots P^{(n)} \| \to 0, \quad n \to \infty.$$
(3B) Dobrushin condition: for any positive integers \(k\)

\[ C(P^{(k)} \cdots P^{(n)}) \to 0, \quad n \to \infty. \]

Then there exists a probability measure \(\pi\) on \(E\) such that

1. \[\|\pi^{(n)} - \pi\| \to 0, \quad n \to \infty.\]

2. Let the initial distribution be \(\mu_0\) and the distribution of the chain \(\bar{X}\) at step \(n\) be \(\mu_n := \mu_{n-1}P^{(n)}\), then for any initial distribution \(\mu_0\), we have

\[\|\mu_n - \pi\| \to 0, \quad n \to \infty.\]

**Proof** Let \(\mathcal{L}(E)\) be the set of all real-valued functions on \(E\). For \(f, g \in \mathcal{L}(E)\), define metric

\[d(f, g) := \|f - g\|.\]

Then \(\mathcal{L}(E)\) is the complete metric space.

First of all, from condition (2), using the triangle inequality, \(\{\pi^{(n)}\}\) is the Cauchy column in \(\mathcal{L}(E)\), so conclusion (1) is correct.

To prove conclusion (2). Using the triangle inequality and the Dobrushin’s inequality, we get

\[
\|\mu_0P^{(1)} \cdots P^{(n)} - \pi\| \leq \|(\mu_0P^{(1)} \cdots P^{(k-1)} - \pi)P^{(k)} \cdots P^{(n)}\| + \|\pi P^{(k)} \cdots P^{(n)} - \pi\| = A(4)
\]

The first term of the equation (A4) tends to 0 when the condition (3A) or (3B) is satisfied. Using the stationary distribution condition \(\pi^{(n)} = \pi^{(n)}P^{(n)}\) for the second term, we deduce that for \(k \geq N\) we have

\[
\|\pi P^{(k)} \cdots P^{(n)} - \pi\| = \|((\pi - \pi^{(k)})P^{(k)} \cdots P^{(n)} + (\pi^{(k)}P^{(k)} \cdots P^{(n)} - \pi))
\]

\[
= \|((\pi - \pi^{(k)})P^{(k)} \cdots P^{(n)} + (\pi^{(k)}P^{(k+1)} \cdots P^{(n)} - \pi))
\]

\[
= \|((\pi - \pi^{(k)})P^{(k)} \cdots P^{(n)} + \sum_{j=1}^{n-k} (\pi^{(k+j-1)} - \pi^{(k+j)})P^{(k+j)} \cdots P^{(n)} + (\pi^{(n)} - \pi))
\]

\[
\geq \sup_{n \geq N} \|((\pi^{(n)} - \pi^{(n+1)}\| + \sum_{n \geq N} \|((\pi^{(n)}) - \pi^{(n+1)})\| + \sup_{n \geq N} \|((\pi^{(n)} - \pi))\| \to 0.
\]

**Proposition 9** Let \(\mu\) and \(\nu\) be two probability distributions on a finite set \(E\). Then

\[
\|\mu - \nu\|_{TV} = \frac{1}{2}\|\mu - \nu\| = \frac{1}{2}\sum_{i \in E} |\mu(i) - \nu(i)|
\]

32
Proposition 10 Let $P$ be the transition matrix of a Markov chain with state space $E$ and let $\mu$ and $\nu$ be any two distributions on $E$. Then
\[ \|\mu P - \nu P\|_{TV} \leq \|\mu - \nu\|_{TV}. \]
This in particular shows that $\|\mu P^t + 1 - \pi\|_{TV} \leq \|\mu P^t - \pi\|_{TV}$, that is, advancing the chain can only move it closer to stationarity.

Proof The conclusion holds from Dobrushin’s inequality (lemma 2) and proposition 9.

Theorem 15 Let $P$ is a transition matrix of random walk on the connected, non-bipartite graph $G$, Then there exists a unique probability distribution $\pi$ over $V$ that satisfies
\[ \pi = \pi P. \]

Proof Since $G$ is a connected graph and the transition matrix $P = (p(u,v), u,v \in V)$ satisfies
\[ p(u,v) > 0, \ (u,v) \in E, \]
for any node $u,z \in V$, there exists $n \in \mathbb{Z}^+$ that satisfies
\[ p^n(u,z) > 0. \]
Thus $P$ is irreducible.

We consider the period of any $u \in V$. Then since $G$ is a non-bipartite graph, $T(u) = 1$. Thus $P$ is aperiodic.

Then there exists a unique stationary distribution $\pi$ of $P$.

Theorem 17 Under the condition of Theorem 15, there exist constants $\alpha \in (0,1)$ and $C > 0$ such that
\[ \max_{u \in V} \|P^t(u, \cdot) - \pi\|_{TV} \leq C\alpha^n, \]
where $P(u, \cdot)$ denotes the row vector consisting of the $i$th row of the transition matrix $P$.

Proof Since $P$ is irreducible and aperiodic. Then by the exponential convergence theorem (Theorem 3), there exist constants $\alpha \in (0,1)$ and $C > 0$ such that
\[ \max_{u \in V} \|P^t(u, \cdot) - \pi\|_{TV} \leq C\alpha^n. \]
Appendix B. Analysis of GEN-SoftMax

In this appendix, we analyze Generalized Aggregation Networks (GEN-SoftMax) (Li et al., 2020), which is a operator-inconsistent GNN model proposed for training deeper GNNs.

In order to be able to train deeper GNN models, Li et al. (2020) propose a new message passing method between nodes $u$ and $v$

$$
\lambda_{u,v}^{(l)} := \frac{\exp(\beta m_{u,v}^{(l-1)})}{\sum_{k \in \mathcal{N}(u)} \exp(\beta m_{u,k}^{(l-1)})},
$$

where $\beta$ is inverse temperature and

$$
m_{u,v}^{(l)} := \text{ReLU}(h_v + I(h_{(l)}(u,v) \cdot h_{(l)}(u,v))) + \epsilon \quad v \in \mathcal{N}(u),
$$

where $I(\cdot)$ is an indicator function being 1 when edge features exist otherwise 0, $\epsilon$ is a small positive constant chosen as $10^{-7}$. Then the definition of message passing in GEN-SoftMax is

$$
h_{(l)}^{(l)} := \sum_{v \in \mathcal{N}(u)} \lambda_{u,v}^{(l)} h_{v}^{(l-1)}.
$$

Write in matrix form

$$
H^{(l)} = P_{\text{GEN}}^{(l)} H^{(l-1)},
$$

where $P_{\text{GEN}}^{(l)} \in \mathbb{R}^{N \times N}$ satisfies $P_{\text{GEN}}^{(l)}(u,v) = \lambda_{u,v}^{(l)}$ if $v \in \mathcal{N}(u)$ otherwise $P_{\text{GEN}}^{(l)}(u,v) = 0$, and $\sum_{v=1}^{N} P_{\text{GEN}}^{(l)}(u,v) = 1$.

Similar to the discussion of GAT in Section 3.3, we can model GEN-SoftMax with $\mathcal{V}$ on a time-inhomogeneous Markov chain $\vec{V}_{\text{GEN}}$ with a family of transition matrices of

$$
\left\{ P_{\text{GEN}}^{(1)}, P_{\text{GEN}}^{(2)}, \ldots, P_{\text{GEN}}^{(l)}, \ldots \right\}.
$$

According to the discussion in 4.4, GEN-SoftMax does not necessarily oversmooth.

Similar to Corollary 19, we have the following sufficient condition to ensure that GEN-SoftMax avoids over-smoothing.

**Corollary 20 (Sufficient condition for GEN-SoftMax to avoid over-smoothing)**

Let $h_{u}^{(l)}$ be the $l$th layer hidden layer feature on node $u \in \mathcal{V}$ in GEN-SoftMax, then a sufficient condition for GEN-SoftMax to avoid over-smoothing in the Markovian sense is that there exists a hyperparameter $\delta > 0$ such that for any $l \geq 1$, satisfying

$$
\|h_{u}^{(l)} - h_{u}^{(l+1)}\| > \delta. \quad (A7)
$$

**Experimental Results.** Similarly in Section 5, we conduct experiments of GEN (Li et al., 2020) on OGB (Hu et al., 2020) dataset. In Table 2, results show that there is an improvement in each dataset and each layer compared with the original model when adding our proposed regularization term. Due to the various tricks during the implementation of GEN, which may relieve over-smoothing, it’s not clear whether the performance improvement is due to our approach, so we don’t demonstrate the degree of over-smoothing here.
Table 2: GEN’s performance on OGB datasets

| Datasets          | Model      | 7 #layers          | 14 #layers         | 28 #layers         | 56 #layers         |
|-------------------|------------|--------------------|--------------------|--------------------|--------------------|
| ogbn-arxiv        | GEN        | 0.7140(±0.0003)    | 0.7198(±0.0007)    | 0.7192(±0.0016)    | -                  |
|                   | GEN-OI     | 0.7181(±0.0006)    | 0.7204(±0.0014)    | 0.7220(±0.0008)    | -                  |
| ogbg-molhiv       | GEN        | 0.7858(±0.0117)    | 0.77567(±0.0019)   | 0.7641(±0.0058)    | 0.7696(±0.0075)    |
|                   | GEN-OI     | 0.7872(±0.0083)    | 0.7838(±0.0024)    | 0.7835(±0.0010)    | 0.7795(±0.0027)    |
| ogbg-ppa          | GEN        | 0.7554(±0.0073)    | 0.7631(±0.0065)    | 0.7712(±0.0071)    | -                  |
|                   | GEN-OI     | 0.7600(±0.0062)    | 0.7700(±0.0081)    | 0.7800(±0.0037)    | -                  |

Appendix C. Experimental details

In this appendix, we add more details on the experiments. Table 3 shows the basic information of each dataset used in our experiments.

Table 3: Summary of the statistics and data split of datasets.

| Dataset         | (Avg.) Nodes | (Avg.) Edges | Features | Class | Train(#/%) | Val(#/%) | Test(#/%) |
|-----------------|--------------|--------------|----------|-------|------------|----------|-----------|
| Cora            | 2708(1 graph)| 5429         | 1433     | 7     | 140        | 500      | 1000      |
| Citeseer        | 3327(1 graph)| 4732         | 3703     | 6     | 120        | 500      | 1000      |
| Pubmed          | 19717(1 graph)| 44338       | 500      | 3     | 60         | 500      | 1000      |
| ogbn-arxiv      | 169,343(1 graph)| 1,166,243  | 128      | 40    | 0.54       | 0.18     | 0.28      |
| ogbg-molhiv     | 25.5(41,127 graph)| 27.5       | 9        | 2     | 0.8        | 0.1      | 0.1       |
| ogbg-ppa        | 243.4(158,100 graph)| 2,266.1   | 7        | 37    | 0.49       | 0.29     | 0.22      |

Table 4 demonstrates the configuration of GNN models, actually, we keep the same setting in the corresponding paper, the only difference is we add the extra proposed regularization term in the optimization objective.

Table 4: Training configuration

| Model | Dataset   | Hidden. | LR.   | Dropout | Epoch | Block | GCN | Agg.   | β       |
|-------|-----------|---------|-------|---------|-------|-------|-----|--------|---------|
| GAT   | Cora      | 64      | 1e-2  | 0.5     | 500   | -     | -   | -      | -       |
|       | Citeseer  | 64      | 1e-2  | 0.5     | 500   | -     | -   | -      | -       |
|       | Pubmed    | 64      | 1e-2  | 0.5     | 500   | -     | -   | -      | -       |
| GEN   | ogbn-arxiv| 256     | 1e-4  | 0.2     | 300   | Res+  | softmax_sg  | 1e-1  |
|       | ogbg-molhiv| 128    | 1e-3  | 0.5     | 500   | Res+  | softmax  | 1     |
|       | ogbg-ppa  | 128     | 1e-2  | 0.5     | 200   | Res+  | softmax_sg  | 1e-2  |

In Table 5, we show the detailed selection of threshold $T$ in equation (11).
Table 5: Selection of threshold $T$ on different layer numbers

| datasets     | model | #layers |
|--------------|-------|---------|
|              |       | 3 4 5 6 7 8 |
| Cora         | GAT-  | 1 0.5 0.6 0.8 1 0.8 |
| Citeseer     | GAT-  | 0.3 0.3 1 0.4 0.2 0.1 |
| Pubmed       | GAT-  | 0.3 0.2 0.2 0.8 0.5 0.4 |
| ogbn-arxiv   | GEN-  | 0.3 0.1 (0.1)0.3 - |
| ogbg-molhiv  | GEN-  | 0.7 0.1 0.5 1 |
| ogbg-ppa     | GEN-  | 0.1 0.3 0.1 - |

References

Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alexander A Alemi. Watch your step: Learning node embeddings via graph attention. *Advances in Neural Information Processing Systems*, 31:9180–9190, 2018.

James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in neural information processing systems*, pages 1993–2001, 2016.

Waiss Azizian et al. Expressive power of invariant and equivariant graph neural networks. In *International Conference on Learning Representations*, 2020.

Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rozemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2464–2473, 2020.

Bruce Bowerman, HT David, and Dean Isaacscon. The convergence of cesaro averages for certain nonstationary markov chains. *Stochastic processes and their applications*, 5(3):221–230, 1977.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

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

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pages 1725–1735. PMLR, 2020b.
Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Clustering: An efficient algorithm for training deep and large graph convolutional networks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 257–266, 2019.

Fan RK Chung and Fan Chung Graham. Spectral graph theory. Number 92. American Mathematical Soc., 1997.

Robert Cogburn. Markov chains in random environments: the case of markovian environments. The Annals of Probability, 8(5):908–916, 1980.

Robert Cogburn. On direct convergence and periodicity for transition probabilities of markov chains in random environments. The Annals of probability, pages 642–654, 1990.

Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Y. H. Wang. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. IEEE Transactions on Intelligent Transportation Systems, PP(99), 2019.

Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. arXiv preprint arXiv:1805.11973, 2018.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems, 29:3844–3852, 2016.

Songgaojun Deng, Huzefa Rangwala, and Yue Ning. Learning dynamic context graphs for predicting social events. In the 25th ACM SIGKDD International Conference, 2019.

Kien Do, Truyen Tran, and Svetha Venkatesh. Graph transformation policy network for chemical reaction prediction. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 750–760, 2019.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In International conference on machine learning, pages 1263–1272. PMLR, 2017.

Guanglu Gong and Minping Qian. Tutorial on applied stochastic processes and stochastic models in algorithms and intelligent computing (in Chinese). Tsinghua University Press, 2004.

Dihe Hu. Stochastic process theory - foundations, theory, applications (in Chinese). Wuhan University Press, 2005.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. arXiv preprint arXiv:2005.00687, 2020.

Cheng-Chi Huang, Dean Isaacsen, and B Vinograde. The rate of convergence of certain non-homogeneous markov chains. Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete, 35(2):141–146, 1976.
Wenbing Huang, Yu Rong, Tingyang Xu, Fuchun Sun, and Junzhou Huang. Tackling over-smoothing for general graph convolutional networks. *arXiv preprint arXiv:2008.09864*, 2020.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Combining neural networks with personalized pagerank for classification on graphs. In *International Conference on Learning Representations*, 2018.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *the 25th ACM SIGKDD International Conference*, 2019.

John Boaz Lee, Ryan Rossi, and Xiangnan Kong. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1666–1674, 2018.

David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcnn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020.

Jia Li, Zhichao Han, Hong Cheng, Jiao Su, Pengyun Wang, Jianfeng Zhang, and Lujia Pan. Predicting path failure in time-evolving graphs. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1279–1289, 2019.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in Neural Information Processing Systems*, 32:2156–2167, 2019.

Kurt Nawrotzki. Finite markov chains in stationary random environments. *The Annals of Probability*, 10(4):1041–1046, 1982.

Hoang Nt and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.
Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

Steven Orey. Markov chains with stochastically stationary transition probabilities. *The Annals of Probability*, 19(3):907–928, 1991.

Namyong Park, Andrey Kan, Xin Luna Dong, Tong Zhao, and Christos Faloutsos. Estimating node importance in knowledge graphs using graph neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 596–606, 2019.

Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Column networks for collective classification. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. Deep-inf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2110–2119, 2018.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

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

Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Guangtao Wang, Rex Ying, Jing Huang, and Jure Leskovec. Improving graph attention networks with large margin-based constraints. *arXiv preprint arXiv:1910.11945*, 2019a.

Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 968–977, 2019b.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.
Chaoqi Yang, Ruijie Wang, Shuochao Yao, Shengzhong Liu, and Tarek Abdelzaher. Revisiting over-smoothing in deep gcns. *arXiv preprint arXiv:2003.13663*, 2020.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, 2018.