BEYOND RANDOM MASKING: WHEN DROPOUT MEETS GRAPH CONVOLUTIONAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Convolutional Networks (GCNs) have emerged as powerful tools for learning on graph-structured data, yet the behavior of dropout in these models remains poorly understood. This paper presents a comprehensive theoretical analysis of dropout in GCNs, revealing its unique interactions with graph structure. We demonstrate that dropout in GCNs creates dimension-specific stochastic subgraphs, leading to a form of structural regularization not present in standard neural networks. Our analysis shows that dropout effects are inherently degree-dependent, resulting in adaptive regularization that considers the topological importance of nodes. We provide new insights into dropout's role in mitigating oversmoothing and derive novel generalization bounds that account for graph-specific dropout effects. Furthermore, we analyze the synergistic interaction between dropout and batch normalization in GCNs, uncovering a mechanism that enhances overall regularization. Our theoretical findings are validated through extensive experiments on both node-level and graph-level tasks across 14 datasets. Notably, GCN with dropout and batch normalization outperforms state-of-the-art methods on several benchmarks. This work bridges a critical gap in the theoretical understanding of regularization in GCNs and provides practical insights for designing more effective graph learning algorithms.

1 INTRODUCTION

The remarkable success of deep neural networks across various domains has been accompanied by the persistent challenge of overfitting, where models perform well on training data but fail to generalize to unseen examples. This issue has spurred the development of numerous regularization techniques, among which dropout has emerged as a particularly effective and widely adopted approach LeCun et al. (2015). Introduced by Srivastava et al. (2014), dropout addresses overfitting by randomly ”dropping out” a proportion of neurons during training, effectively creating an ensemble of subnetworks. This technique has proven highly successful in improving generalization and has become a standard tool in the deep learning toolkit. The effectiveness of dropout has prompted extensive theoretical analysis, with various perspectives offered to explain its regularization effects.

Some researchers have interpreted dropout as a form of model averaging (Baldi & Sadowski, 2013), while others have analyzed it through the lens of information theory (Achille & Soatto, 2018). Wager et al. (2013) provided insights into dropout's adaptive regularization properties, and Gal & Ghahramani (2016) established connections between dropout and Bayesian inference. These diverse theoretical frameworks have significantly enhanced our understanding of dropout's role in mitigating overfitting in traditional neural networks. However, as the field of deep learning has expanded to encompass more complex data structures, particularly graphs, new questions have arisen regarding the applicability and behavior of established techniques. Graph Neural Networks (GNNs), especially Graph Convolutional Networks (GCNs), have demonstrated remarkable performance on tasks involving graph-structured data (Kipf & Welling, 2017). Naturally, researchers and practitioners have applied dropout to GNNs, often observing beneficial effects on generalization (Hamilton et al., 2017).

Despite the widespread adoption of dropout in Graph Convolutional Networks (GCNs), our preliminary investigations have revealed intriguing discrepancies between its behavior in GCNs and its well-understood effects in traditional neural networks. These observations prompt a fundamental question: How does dropout uniquely interact with the graph structure in GCNs? In this paper, we present a comprehensive theoretical analysis of dropout in the context of GCNs. Our findings reveal that dropout in GCNs interacts with the underlying graph structure in ways that are fundamentally different from its operation in traditional neural networks. Specifically, we demonstrate that:

- Dropout in GCNs creates dimension-specific stochastic sub-graphs, leading to a unique form of structural regularization not present in standard neural networks.
- The effects of dropout are inherently degree-dependent, with differential impacts on nodes based on their connectivity, resulting in adaptive regularization that considers the topological importance of nodes in the graph.
- Dropout plays a crucial role in mitigating the oversmoothing problem in GCNs, though its effects are more nuanced than previously thought.
- The generalization bounds for GCNs with dropout exhibit a complex dependence on graph properties, diverging from traditional dropout theory.
- There exists a significant interplay between dropout and batch normalization in GCNs, revealing synergistic effects that enhance the overall regularization.

Our theoretical framework not only provides deeper insights into the mechanics of dropout in graph-structured data but also yields practical implications for the design and training of GCNs. We validate our theoretical findings through extensive experiments on both node-level and graph-level tasks, demonstrating the practical relevance of our analysis. This work bridges a critical gap in the theoretical understanding of regularization in GCNs and paves the way for more principled approaches to leveraging dropout in graph representation learning. Furthermore, we validate our theoretical findings through extensive experiments, demonstrating that GCNs incorporating our insights on dropout and batch normalization outperform several state-of-the-art methods on benchmark datasets, including Cora, CiteSeer, and PubMed. This practical success underscores the importance of our theoretical contributions and their potential to advance the field of graph representation learning.

2 RELATED WORK

Dropout in Neural Networks. Overfitting can be reduced by using dropout Hinton et al. (2012) to prevent complex co-adaptations on the training data. Since its inception, several variants have been proposed to enhance its effectiveness. DropConnect (Wan et al., 2013) generalizes dropout by randomly dropping connections rather than nodes. Gaussian dropout Srivastava et al. (2014) replaces the Bernoulli distribution with a Gaussian one for smoother regularization. Curriculum dropout (Morerio et al., 2017) adaptively adjusts the dropout rate during training. Theoretical interpretations of dropout have provided insights into its success. The model averaging perspective (Baldi & Sadowski, 2013) views dropout as an efficient way of approximately combining exponentially many different neural networks. The adaptive regularization interpretation (Wager et al., 2013) shows how dropout adjusts the regularization strength for each feature based on its importance. The Bayesian approximation view (Gal & Ghahramani, 2016) connects dropout to variational inference in Bayesian neural networks, providing a probabilistic framework for understanding its effects.

Regularization in Graph Neural Networks. Graph Neural Networks (GNNs), while powerful, are prone to overfitting and over-smoothing (Li et al., 2018). Various regularization techniques (Yang et al., 2021; Rong et al., 2019; Fang et al., 2023; Feng et al., 2020) have been proposed to address these issues. DropEdge (Rong et al., 2019) randomly removes edges from the input graph during training, reducing over-smoothing and improving generalization. Graph diffusion-based methods (Gasteiger et al., 2019) incorporate higher-order neighborhood information to enhance model robustness. Spectral-based approaches (Wu et al., 2019) leverage the graph spectrum to design effective regularization strategies. Empirical studies have shown that traditional dropout can be effective in GNNs (Hamilton et al., 2017), but its interaction with graph structure remains poorly understood. Some works have proposed adaptive dropout strategies for GNNs (Gao & Ji, 2019), but these are primarily heuristic approaches without comprehensive theoretical grounding.

Theoretical Frameworks for GNNs. Despite the empirical success of Graph Neural Networks (GNNs), establishing theories to explain their behaviors is still an evolving field. Recent works have made significant progress in understanding over-smoothing (Li et al., 2018; Zhao & Akoglu, 2019; Oono & Suzuki, 2019; Rong et al., 2020), interpretability (Ying et al., 2019; Luo et al., 2020; Vu & Thai, 2020; Yuan et al., 2020; 2021), expressiveness (Xu et al., 2018; Chen et al., 2019; Maron et al., 2018; Dehmamy et al., 2019; Feng et al., 2022), and generalization (Scarselli et al., 2018; Du et al., 2019; Verma & Zhang, 2019; Garg et al., 2020; Zhang et al., 2020; Oono & Suzuki, 2019; Lv, 2021; Liao et al., 2020; Esser et al., 2021; Cong et al., 2021). Our work aims to complement these existing theoretical frameworks by focusing on the practical aspects of dropout in GNNs, a widely used regularization technique that has not been thoroughly examined from a theoretical perspective. Previous works have provided valuable insights using classical techniques such as Vapnik-Chervonenkis dimension (Scarselli et al., 2018), Rademacher complexity (Lv, 2021; Garg et al., 2020), and algorithm stability (Verma & Zhang, 2019). Recent efforts (Oono & Suzuki, 2019; Esser et al., 2021) have also made strides in incorporating the transductive learning schema of GNNs into theoretical analyses. We bridge the gap between theoretical understanding and practical implementation of GNNs, offering insights into how dropout affects generalization and performance in graph-structured learning tasks.

3 THEORETICAL FRAMEWORK

In this section, we develop a rigorous mathematical framework to analyze the behavior of dropout in Graph Convolutional Networks (GCNs). We begin by establishing notations and definitions, then formalize the GCN model with dropout, and finally introduce key concepts that will be central to our analysis.

3.1 NOTATIONS AND DEFINITIONS

Notations. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ be an undirected graph with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges, where $\mathbf{X} \in \mathbb{R}^{n \times d_0}$ represents the node feature matrix with d_0 input features per node. We denote by $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ the adjacency matrix of $\mathcal{G}$, and by $\boldsymbol{D} = \text{diag}(deg_1, \ldots, deg_n)$ the degree matrix, where $deg_i = \sum_j A_{ij}$.

3.1.1 MATRIX DEFINITIONS AND GRAPH CONVOLUTIONAL NETWORKS (GCNS)

Definition 1 (Normalized Adjacency Matrix). *The normalized adjacency matrix $\tilde{\boldsymbol{A}}$ is defined as:*

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

Let $\boldsymbol{X} \in \mathbb{R}^{n \times d_0}$ be the input feature matrix, where d_0 is the number of input features per node.

Definition 2 (L-layer GCN). *An L-layer GCN is defined as a sequence of L graph convolutional layers, where the l-th layer ($l = 1, \ldots, L$) performs the following transformation:*

$$\boldsymbol{H}^{(l)} = \sigma(\tilde{\boldsymbol{A}} \boldsymbol{H}^{(l-1)} \boldsymbol{W}^{(l)}), \tag{2}$$

where $\boldsymbol{H}^{(l)} \in \mathbb{R}^{n \times d_l}$ is the feature matrix at layer l, $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$ is the weight matrix for layer l, $\sigma(\cdot)$ is a non-linear activation function, and $\boldsymbol{H}^{(0)} = \boldsymbol{X}$.

Definition 3 (Feature Energy). *The feature energy $E(\boldsymbol{H}^{(l)})$ of the node representations $\boldsymbol{H}^{(l)}$ at layer l is defined as:*

$$E(\boldsymbol{H}^{(l)}) = \frac{1}{2|\mathcal{E}|} \sum_{(i,j) \in \mathcal{E}} \|\boldsymbol{h}_i^{(l)} - \boldsymbol{h}_j^{(l)}\|^2,$$

where $\boldsymbol{h}_i^{(l)}$ denotes the i-th row of $\boldsymbol{H}^{(l)}$ and the representation of node i in the l-layer.

3.1.2 DROPOUT IN GCNS

Definition 4 (Dropout Mask). *For layer l, the dropout mask $\boldsymbol{M}^{(l)} \in \mathbb{R}^{n \times d_l}$ is a random matrix where each element $M_{ij}^{(l)}$ is drawn independently from a Bernoulli distribution:*

$$M_{ij}^{(l)} \sim \textit{Bernoulli}(1 - p) \tag{3}$$

where $p \in [0, 1]$ is the dropout probability.

We now formally define a GCN with dropout:

Definition 5 (GCN with Dropout). *For an L-layer GCN with dropout, the forward pass at layer l is defined as:*

$$\boldsymbol{H}^{(l)} = \frac{1}{1-p}\boldsymbol{M}^{(l)} \odot \sigma(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)}), \tag{4}$$

where $\odot$ denotes element-wise multiplication, and the factor $\frac{1}{1-p}$ is used to scale the outputs during training to match the expected value at inference time.

To elucidate the specific impact of dropout on embedding features, we introduce these concepts:

Definition 6 (Dimension-specific Sub-graph). *For each feature dimension j at layer l and iteration t, we define a stochastic sub-graph $\mathcal{G}_t^{(l,j)} = (\mathcal{V}, \mathcal{E}_t^{(l,j)})$, where:*

$$\mathcal{E}_t^{(l,j)} = \{(u,v) \in \mathcal{E} \mid M_{uj}^{(l,t)} \neq 0 \text{ and } M_{vj}^{(l,t)} \neq 0\},$$

Here, $M^{(l,t)}$ denotes the dropout mask for layer l at iteration t.

Definition 7 (Active Path). *A path $\mathcal{P} = (v_0, v_1, \ldots, v_k)$ in $\mathcal{G}$ is considered active for feature j at layer l and iteration t if and only if:*

$$\prod_{i=0}^{k-1} M_{v_i j}^{(l,t)} M_{v_{i+1} j}^{(l,t)} \neq 0.$$

Definition 8 (Feature-Topology Coupling Matrix). *For layer l at iteration t, we define the feature-topology coupling matrix $\boldsymbol{C}_t^{(l)} \in \mathbb{R}^{n\times n}$ as:*

$$\boldsymbol{C}_t^{(l)} = \tilde{\boldsymbol{A}} \odot (\boldsymbol{M}_t^{(l)}(\boldsymbol{M}_t^{(l)})^T),$$

where $\tilde{\boldsymbol{A}}$ is the normalized adjacency matrix, $\boldsymbol{M}_t^{(l)}$ is the dropout mask for layer l at iteration t, and $\odot$ denotes the Hadamard product.

This matrix $C_t^{(l)}$ captures how dropout affects both feature propagation and graph structure simultaneously.

Definition 9 (Effective Degree). *The effective degree $deg_i^{eff}(t)$ of node i at iteration t is defined as:*

$$deg_i^{eff}(t) = \sum_j (\boldsymbol{C}_t^{(l)})_{ij},$$

where $\boldsymbol{C}_t^{(l)}$ is the feature-topology coupling matrix defined earlier.

3.1.3 INTEGRATING BATCH NORMALIZATION (BN) AND DROPOUT IN GCNS

Definition 10 (GCN Layer with BN before activation). *For a layer l, the output is defined as:*

$$\boldsymbol{H}^{(l)} = \sigma(BN(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)})),$$

where σ is the activation function, such as ReLU, and BN is defined as:

$$BN(\boldsymbol{X}) = \gamma \odot \frac{\boldsymbol{X} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta.$$

Here, μ_B and σ_B^2 are the batch mean and variance, γ and β are learnable parameters, and ϵ is a small constant for numerical stability.

3.2 DIMENSION-SPECIFIC STOCHASTIC SUB-GRAPHS

Figure 1 shows how varying dropout rates impact the number of edges $\mathcal{E}_t$ in stochastic sub-graphs of a 2-layer GCN, defined by Equation 5, across the Cora and Citeseer datasets. We observe that higher dropout rates correlate with fewer edges in these sub-graphs. This variation demonstrates dropout's role in GCNs as a form of structural regularization, where dimension-specific stochastic sub-graphs are generated. Each feature dimension samples a different sub-graph from the original graph at each iteration. This mechanism provides a rich set of structural variations during training, potentially enhancing the model's ability to capture diverse graph patterns.

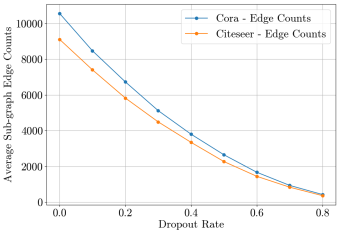

Figure 1: Sub-graph size.

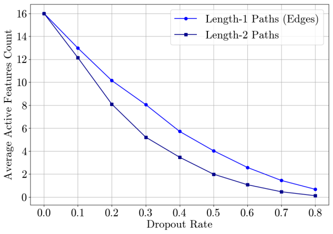

Figure 2: Active path on Cora.

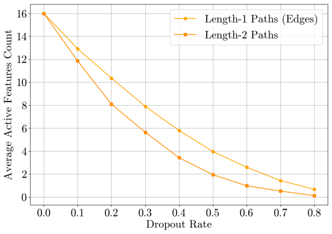

Figure 3: Active path on Citeseer.

Theorem 11 (Sub-graph Diversity)**.** *The expected number of distinct sub-graphs per iteration is:*

$$\mathbb{E}[|\mathcal{G}_t^{(l,j)} \mid j = 1, \ldots, d_l|] = d_l(1 - (1-p)^{2|\mathcal{E}|}),$$

where d_l is the number of features at layer l, p is the dropout probability, and $|\mathcal{E}|$ is the number of edges in the original graph (The complete proof is in the Appendix. A.1).

This theorem reveals that dropout in GCNs leads to a rich set of sub-graphs, providing a form of structural data augmentation unique to graph-based models. The diversity of these sub-graphs increases with both the dropout probability p and the number of features d_l. This suggests that higher-dimensional GCNs with moderate dropout rates can benefit from a wider range of structural variations during training, potentially leading to more robust and generalizable representations. Moreover, this mechanism allows the GCN to implicitly explore different graph structures without explicitly modifying the input graph. This could be particularly beneficial for tasks where the optimal graph structure is uncertain or where multiple relevant sub-structures exist within the data.

Theorem 12 (Expected Active Features per Path)**.** *For a path $\mathcal{P}$ of length k, the expected number of features for which it is active is:*

$$\mathbb{E}[\#\textit{active features for } \mathcal{P}] = d_l(1-p)^{k+1}.$$

This theorem demonstrates that while individual long paths are unlikely to be active for any given feature, the multi-dimensional nature of GCNs allows for effective long-range information flow through the ensemble effect across features. Figures 2 & 3 illustrate the behavior of active features along paths of length 1 and 2 within a 2-layer GCN equipped with 16 hidden dimensions, across varying dropout rates. Notably, at a dropout rate of 0.6, the average number of active features approaches zero. This characteristic also underscores the importance of multidimensional feature spaces in ensuring robust information transmission under feature dropout.

3.3 DEGREE-DEPENDENT NATURE OF DROPOUT EFFECTS

The interaction between dropout and the graph structure leads to a form of degree-dependent regularization in GCNs. This means that the effect of dropout varies based on the connectivity of each node, creating an adaptive regularization scheme that considers the topological importance of nodes in the graph.

Theorem 13 (Degree-Dependent Dropout Effect)**.** *The expected effective degree and its variance are given by:*

$$\mathbb{E}[deg_i^{\textit{eff}}(t)] = (1-p)^2 deg_i \quad \textit{and} \quad Var[deg_i^{\textit{eff}}(t)] = deg_i(1-p)^2(1-(1-p)^2), \tag{5}$$

where deg_i is the original degree of node i and p is the dropout probability).

This theorem highlights that dropout affects nodes differentially depending on their degree. High-degree nodes, typically more influential within the graph, exhibit less variation in their effective degree due to dropout, potentially resulting in more stable representations for these important nodes. This observation is empirically confirmed in the analysis of a 2-layer GCN presented in Figure 6. Consequently, the degree-dependent nature of dropout in GCNs results in adaptive regularization, where the regularization effect naturally adjusts to the local graph structure.

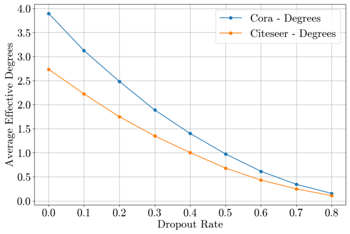

Figure 4: Effective degree.

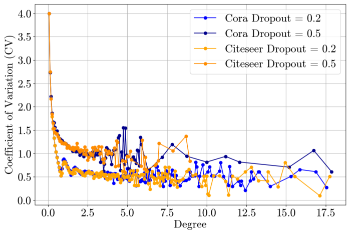

Figure 5: Effective CV vs degree.

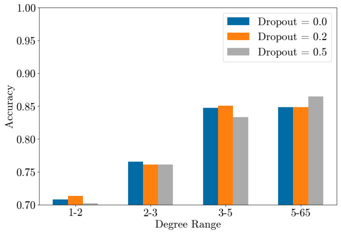

Figure 6: Accuracy on Cora.

Corollary 14 (Relative Stability of High-Degree Nodes)**.** *The coefficient of variation of the effective degree, defined as* $CV[deg_i^{eff}(t)] = \sqrt{Var[deg_i^{eff}(t)]}/\mathbb{E}[deg_i^{eff}(t)]$*, decreases with increasing node degree:*

$$CV[deg_i^{eff}(t)] = \frac{\sqrt{1-(1-p)^2}}{\sqrt{deg_i}(1-p)}.$$

This corollary further confirms that high-degree nodes experience relatively less variation in their effective degree due to dropout. Figure 5 illustrates that the CV decreases as node degree increases. This degree-dependent effect distinguishes dropout in GCNs from its application in standard neural networks and suggests that the optimal dropout strategy for GCNs may need to consider the graph structure explicitly.

3.4 ROLE OF DROPOUT IN OVERSMOOTHING

Oversmoothing is a well-known issue in GCNs, where node representations become indistinguishable as the number of layers increases. Our analysis reveals that dropout plays a crucial role in this context, though its effects are more nuanced than previously thought.

Theorem 15 (Dropout and Feature Energy)**.** *For a GCN with dropout probability* p*, the expected feature energy at layer* l *is bounded by:*

$$\mathbb{E}[E(\boldsymbol{H}^{(l)})] \leq \frac{deg_{\max}}{|\mathcal{E}|}(\frac{1}{1-p})^l ||\tilde{\boldsymbol{A}}||_2^{2l} \prod_{i=1}^{l} ||\boldsymbol{W}^{(i)}||_2^2 ||\boldsymbol{X}||_F^2 \tag{6}$$

where $E(\boldsymbol{X})$ *is the energy of the input features and* $\boldsymbol{W}^{(i)}$ *are the weight matrices (The complete proof is in the Appendix.A.2).*

The derived bound demonstrates how dropout affects feature energy through the interplay of network depth (l), graph structure (through $deg_{\max}$ and $\tilde{\boldsymbol{A}}$), and weight properties ($||\boldsymbol{W}^{(i)}||_2$). Note that this analysis only provides an upper bound; the absence of a lower bound in this derivation is due to limitations in bounding certain terms. We will later show that when considering batch normalization, we can establish the existence of a lower bound, providing a more complete characterization of feature energy behavior. Additionally, we explored how dropout modulates the weight matrices in a 2-layer GCN, with a particular focus on its effects on the spectral norm, as detailed in Appendix A.5. Building on this, we further analyze three key metrics to understand how dropout influences feature representations, as depicted in Figure 9. From the left side of Figure 9, the Frobenius norm of features remains relatively stable whether dropout is applied or not, suggesting that dropout's effects are not simply uniformly scaling all features. The middle of Figure 9 shows that dropout consistently doubles the average pairwise distance between nodes, aiding in maintaining more distinctive node representations. Most notably, the right side of Figure 9 demonstrates that dropout significantly increases feature energy. The substantial rise in feature energy, compared to the moderate changes in Frobenius norm and pairwise distances, provides strong evidence that dropout enhances discriminative power between connected nodes, explaining its effectiveness in preventing oversmoothing.

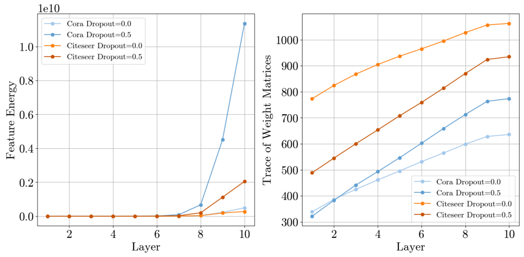

Figure 7: Feature energy vs dropout rates.

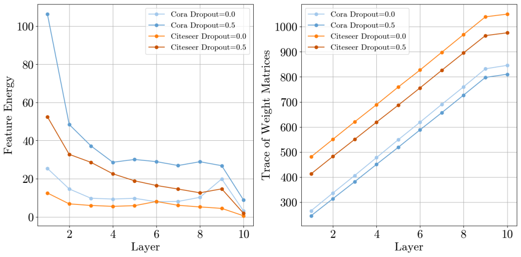

Figure 8: BN feature energy vs dropout rates.

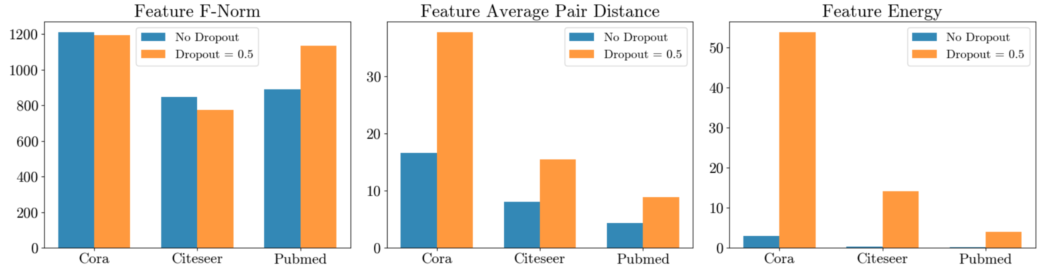

Figure 9: Effect of dropout on feature F-norm, average pair distance, and feature energy.

3.5 GENERALIZATION BOUNDS WITH GRAPH-SPECIFIC DROPOUT EFFECTS

The unique properties of dropout in GCNs, such as the creation of stochastic sub-graphs and degree-dependent effects, influence how these models generalize to unseen data. Our analysis provides novel generalization bounds that explicitly account for these graph-specific dropout effects, offering insights into how dropout interacts with graph structure to influence the model's generalization capabilities.

Theorem 16 (Generalization Bound for L-Layer GCN with Dropout)**.** *For an L-layer GCN F with dropout probability p, with probability at least $1-\delta$ over the training examples, the following generalization bound holds:*

$$\mathbb{E}_D[L(F(\boldsymbol{x}))]-\mathbb{E}_S[L(F(\boldsymbol{x}))] \leq O\left(\sqrt{\frac{\log(1/\delta)}{n}}\sum_{l=1}^{L} L_{loss}\cdot L_l\cdot\sqrt{\frac{p}{1-p}}\|\sigma(\tilde{\boldsymbol{A}}H^{(l-1)}\boldsymbol{W}^{(l)})\|_F\right), \tag{7}$$

where $\mathbb{E}_D$ is the expectation over the data distribution. $\mathbb{E}_S$ is the expectation over the training samples. L is the loss function with Lipschitz constant L_{loss}. $L_l = \prod_{i=l+1}^{L}(\|\boldsymbol{W}^{(i)}\|\cdot\|\tilde{\boldsymbol{A}}\|)$ is the Lipschitz constant from layer l to output. $\|\boldsymbol{W}^{(i)}\|$ is the spectral norm (largest singular value) of the weight matrix at layer i. $\|\tilde{\boldsymbol{A}}\|$ is the spectral norm of the normalized adjacency matrix. n is the number of training samples. p is the dropout probability. The bound reflects how the network's stability depends on the Lipschitz constant of the loss function L_{loss}, the layer-wise Lipschitz constants L_l capturing weight and graph effects, the magnitude of feature activations $\|\sigma(\tilde{\boldsymbol{A}}H^{(l-1)}\boldsymbol{W}^{(l)})\|_F$, the dropout rate p through the term $\sqrt{\frac{p}{1-p}}$ (The complete proof is in the Appendix.A.3).

This generalization bound reveals how dropout affects GCNs' learning capabilities and presents several practical insights: First, network depth plays a crucial role. As signals propagate through layers, the effects of weights and graph structure accumulate multiplicatively. This suggests that deeper GCNs might need more careful regularization, as small perturbations could amplify through the network. Second, the graph structure naturally influences how information flows through the network. The way we normalize our adjacency matrix (typically ensuring its norm is at most 1) provides a built-in stabilizing effect. However, graphs with different connectivity patterns might require different dropout strategies. Third, looking at each layer individually, we see that both network weights and feature magnitudes matter. Some layers might process more important features than others, suggesting that a one-size-fits-all dropout rate might not be optimal. Instead, adapting dropout rates based on layer-specific characteristics could be more effective. Finally, there's an inherent trade-off in choosing dropout rates. Higher dropout rates provide stronger regularization but also introduce more noise in the training process. Our bound helps explain this balance mathematically, suggesting why moderate dropout rates often work best in practice.

3.6 INTERACTION OF DROPOUT AND BATCH NORMALIZATION IN GCNS

While dropout provides a powerful regularization mechanism for GCNs, its degree-dependent nature can lead to uneven regularization across nodes. Batch Normalization (BN) offers a complementary approach that can potentially address this issue and enhance the benefits of dropout. Our analysis reveals how the combination of dropout and BN creates a synergistic regularization effect that is sensitive to both graph structure and feature distributions.

Theorem 17 (Layer-wise Energy Lower Bound for GCN)**.** *For an L-layer Graph Convolutional Network with dropout rate p, batch normalization parameters $\{\beta_d^{(l)}, \gamma_d^{(l)}\}_{d=1}^{d_l}$ at each layer l, with probability at least $(1-\delta)^L$, the expected feature energy at each layer l satisfies:*

$$E(\boldsymbol{H}^{(l)}) \geq \frac{p\,deg_{\min}}{2|\mathcal{E}|(1-p)} \sum_{d=1}^{d_l} \Phi(\beta_d^{(l)}/\gamma_d^{(l)}) \cdot (\beta_d^{(l)})^2$$

where $l = 1, 2, ..., L$ indicates the layer, $deg_{\min}$ is the minimum degree in the graph, $|\mathcal{E}|$ is the total number of edges, Φ is the standard normal CDF and $\beta_d^{(l)}, \gamma_d^{(l)}$ are the BN parameters for dimension d at layer l (The complete proof is in the Appendix.A.4).

Our theoretical bound reveals the synergistic interaction between dropout and batch normalization in GCNs, establishing a refined form of regularization. The energy preservation term $\frac{p}{1-p}$ from dropout combines with the BN-induced bound $\sum_{d=1}^{d_l} \Phi(\beta_d^{(l)}/\gamma_d^{(l)}) \cdot (\beta_d^{(l)})^2$ to maintain non-vanishing feature energy. This interaction is empirically validated in Figures 7 & 8, which demonstrate how batch normalization effectively moderates the energy amplification caused by dropout. These findings suggest that the joint application of dropout and batch normalization in GCNs creates a specialized mechanism particularly suited for graph-structured data.

4 EXPERIMENTS

To validate our theoretical analysis, we conducted extensive experiments on a variety of datasets, considering both node-level and graph-level tasks. We implemented dropout technique on several popular GNN architectures: GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018), and GatedGCN (Bresson & Laurent, 2017). For each model, we compared the performance with and without dropout. Our code is available at `https://anonymous.4open.science/r/dropout-theory`.

4.1 DATASETS AND SETUP

Datasets. For node-level tasks, we used 10 datasets: Cora, CiteSeer, PubMed (Sen et al., 2008), ogbn-arxiv, ogbn-products (Hu et al., 2020), Amazon-Computer, Amazon-Photo, Coauthor-CS, Coauthor-Physics (Shchur et al., 2018), and WikiCS (Mernyei & Cangea, 2020). Cora, CiteSeer, and PubMed are citation networks, evaluated using the semi-supervised setting and data splits from Kipf & Welling (2017). Computer and Photo (Shchur et al., 2018) are co-purchase networks. CS and Physics (Shchur et al., 2018) are co-authorship networks. We used the standard 60%/20%/20% training/validation/test splits and accuracy as the evaluation metric (Chen et al., 2022; Shirzad et al., 2023; Deng et al., 2024). For WikiCS, we adopted the official splits and metrics (Mernyei & Cangea, 2020). For large-scale graphs, we included ogbn-arxiv and ogbn-products with 0.16M to 2.4M nodes, using OGB's standard evaluation settings (Hu et al., 2020).

For graph-level tasks, we used MNIST, CIFAR10 (Dwivedi et al., 2023), and two Peptides datasets (functional and structural) (Dwivedi et al., 2022). MNIST and CIFAR10 are graph versions of their image classification counterparts, constructed using 8-nearest neighbor graphs of SLIC superpixels.

Table 1: Node classification results (%). The baseline results are taken from Deng et al. (2024); Wu et al. (2023). The top $\mathbf{1^{st}}$, $\mathbf{2^{nd}}$ and $\mathbf{3^{rd}}$ results are highlighted. ”dp” denotes dropout.

	Cora	CiteSeer	PubMed	Computer	Photo	CS	Physics	WikiCS	ogbn-arxiv	ogbn-products
# nodes	2,708	3,327	19,717	13,752	7,650	18,333	34,493	11,701	169,343	2,449,029
# edges	5,278	4,732	44,324	245,861	119,081	81,894	247,962	216,123	1,166,243	61,859,140
Metric	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑
GCNII	**85.19** ± 0.26	**73.20** ± 0.83	80.32 ± 0.44	91.04 ± 0.41	94.30 ± 0.20	92.22 ± 0.14	95.97 ± 0.11	78.68 ± 0.55	72.74 ± 0.31	79.42 ± 0.36
GPRGNN	83.17 ± 0.78	71.86 ± 0.67	79.75 ± 0.38	89.32 ± 0.29	94.49 ± 0.14	95.13 ± 0.09	96.85 ± 0.08	78.12 ± 0.23	71.10 ± 0.12	79.76 ± 0.59
APPNP	83.32 ± 0.55	71.78 ± 0.46	80.14 ± 0.22	90.18 ± 0.17	94.32 ± 0.14	94.49 ± 0.07	96.54 ± 0.07	78.87 ± 0.11	72.34 ± 0.24	78.84 ± 0.09
tGNN	82.97 ± 0.68	71.74 ± 0.49	**80.67** ± 0.34	83.40 ± 1.33	89.92 ± 0.72	92.85 ± 0.48	96.24 ± 0.24	71.49 ± 1.05	72.88 ± 0.26	81.79 ± 0.54
GraphGPS	82.84 ± 1.03	72.73 ± 1.23	79.94 ± 0.26	91.19 ± 0.54	95.06 ± 0.13	93.93 ± 0.12	97.12 ± 0.19	78.66 ± 0.49	70.97 ± 0.41	OOM
NAGphormer	82.12 ± 1.18	71.47 ± 1.30	79.73 ± 0.28	91.22 ± 0.14	95.49 ± 0.11	**95.75** ± 0.09	**97.34** ± 0.03	77.16 ± 0.72	70.13 ± 0.55	73.55 ± 0.21
Exphormer	82.77 ± 1.38	71.63 ± 1.19	79.46 ± 0.35	91.47 ± 0.17	95.35 ± 0.22	94.93 ± 0.01	96.89 ± 0.09	78.54 ± 0.49	72.44 ± 0.28	OOM
GOAT	83.18 ± 1.27	71.99 ± 1.26	79.13 ± 0.38	90.96 ± 0.90	92.96 ± 1.48	94.21 ± 0.38	96.24 ± 0.24	77.00 ± 0.77	72.41 ± 0.40	**82.00** ± 0.43
NodeFormer	82.20 ± 0.90	**72.50** ± 1.10	79.90 ± 1.00	86.98 ± 0.62	93.46 ± 0.35	95.64 ± 0.22	96.45 ± 0.28	74.73 ± 0.94	59.90 ± 0.42	73.96 ± 0.30
SGFormer	**84.50** ± 0.80	72.60 ± 0.20	80.30 ± 0.60	92.42 ± 0.66	**95.58** ± 0.36	**95.71** ± 0.24	96.75 ± 0.26	80.05 ± 0.46	72.63 ± 0.13	81.54 ± 0.43
Polynormer	83.25 ± 0.93	72.31 ± 0.78	79.24 ± 0.43	**93.68** ± 0.21	**96.46** ± 0.26	95.53 ± 0.16	**97.27** ± 0.08	80.10 ± 0.67	**73.46** ± 0.16	**83.82** ± 0.11
GCN	**85.22** ± 0.66	**73.24** ± 0.63	**81.08** ± 1.16	**93.15** ± 0.34	95.03 ± 0.24	94.41 ± 0.13	97.07 ± 0.04	**80.14** ± 0.52	**73.13** ± 0.27	81.87 ± 0.41
Dirichlet energy	7.403	0.437	0.452	8.020	3.765	20.241	8.966	735.876	8.021	7.771
GCN w/o dp	83.18 ± 1.22	70.48 ± 0.45	79.40 ± 1.02	90.60 ± 0.84	94.10 ± 0.15	94.30 ± 0.22	96.92 ± 0.05	77.61 ± 1.34	72.05 ± 0.23	77.50 ± 0.37
Dirichlet energy	2.951	0.170	0.114	0.592	1.793	3.980	0.318	264.230	1.231	1.745
GCN w/o BN	84.97 ± 0.73	72.97 ± 0.86	80.94 ± 0.87	92.39 ± 0.18	94.38 ± 0.13	93.46 ± 0.24	96.76 ± 0.06	79.00 ± 0.48	71.93 ± 0.18	79.37 ± 0.42
SAGE	84.14 ± 0.63	71.62 ± 0.29	77.86 ± 0.79	92.65 ± 0.21	**95.71** ± 0.20	**95.90** ± 0.09	**97.20** ± 0.10	**80.29** ± 0.97	72.72 ± 0.13	**82.69** ± 0.28
SAGE w/o dp	83.06 ± 0.80	69.68 ± 0.82	76.40 ± 1.48	90.17 ± 0.60	94.90 ± 0.17	95.80 ± 0.08	97.06 ± 0.06	78.84 ± 1.17	71.37 ± 0.31	79.82 ± 0.22
SAGE w/o BN	83.89 ± 0.67	71.39 ± 0.75	77.26 ± 1.02	92.54 ± 0.24	95.51 ± 0.23	94.87 ± 0.15	97.03 ± 0.03	79.50 ± 0.93	71.52 ± 0.17	80.91 ± 0.35
GAT	83.92 ± 1.29	72.00 ± 0.91	**80.48** ± 0.99	**93.47** ± 0.27	95.53 ± 0.16	94.49 ± 0.17	96.73 ± 0.10	**80.21** ± 0.68	**72.83** ± 0.19	80.05 ± 0.34
GAT w/o dp	82.58 ± 1.47	71.08 ± 0.42	79.28 ± 0.58	92.94 ± 0.30	93.88 ± 0.16	94.30 ± 0.14	96.42 ± 0.08	78.67 ± 0.40	71.52 ± 0.41	77.87 ± 0.25
GAT w/o BN	83.76 ± 1.32	71.82 ± 0.83	80.43 ± 1.03	92.16 ± 0.26	95.05 ± 0.49	93.33 ± 0.26	96.57 ± 0.20	79.49 ± 0.62	71.68 ± 0.36	78.21 ± 0.32

We follow all evaluation protocols suggested by Dwivedi et al. (2023). Peptides-func involves classifying graphs into 10 functional classes, while Peptides-struct regresses 11 structural properties. All evaluations followed the protocols in (Dwivedi et al., 2022).

Baselines. Our main focus lies on the following prevalent GNNs and transformer models from Polynormer (Deng et al., 2024): GCN (Kipf & Welling, 2017), SAGE (Hamilton et al., 2017), GAT Veličković et al. (2018), GCNII (Chen et al., 2020), (Veličković et al., 2018), APPNP (Gasteiger et al., 2018), GPRGNN (Chien et al., 2020), SGFormer (Wu et al., 2023), Polynormer (Deng et al., 2024), GOAT (Kong et al., 2023), NodeFormer (Wu et al., 2022), NAGphormer (Chen et al., 2022), GTDwivedi & Bresson (2020), SAN Kreuzer et al. (2021), MGT Ngo et al. (2023), DRew Gutteridge et al. (2023), Graph-MLPMixer He et al. (2023), GRIT Ma et al. (2023) , GraphGPS (Rampášek et al., 2022), Exphormer (Shirzad et al., 2023), CKGCN (Ma et al., 2024), GRED (Ding et al., 2024), Graph Mamba Behrouz & Hashemi (2024). We report the performance results of baselines primarily from (Deng et al., 2024), with the remaining obtained from their respective original papers or official leaderboards whenever possible, as those results are obtained by well-tuned models.

Experimental Setup. We implemented all models using the PyTorch Geometric library (Fey & Lenssen, 2019). The experiments are conducted on a single workstation with 8 RTX 3090 GPUs. For node-level tasks, we adhered to the training protocols specified in (Deng et al., 2024), employing BN and adjusting the dropout rate between 0.1 and 0.7. In graph-level tasks, we followed the experimental settings established by Tönshoff et al. (2023), utilizing BN with a consistent dropout rate of 0.2. All experiments were run with 5 different random seeds, and we report the mean accuracy and standard deviation. To ensure generalizability, we used Dirichlet energy (Cai & Wang, 2020) as an oversmoothing metric, which is proportional to our feature energy (see appendix).

4.2 NODE-LEVEL CLASSIFICATION RESULTS

The node-level classification results in Table 1 not only align with our theoretical predictions but also showcase the remarkable effectiveness of dropout. Notably, GCN with dropout and batch normalization outperforms state-of-the-art methods on several benchmarks, including Cora, CiteSeer, and PubMed. This superior performance underscores the practical significance of our theoretical insights. Consistently across all datasets, models employing dropout outperform their counterparts without it, validating our analysis that dropout provides beneficial regularization in GNNs, distinct from its effects in standard neural networks. The varying levels of improvement observed across different datasets support our theory of degree-dependent dropout effects that adapt to the graph structure. Furthermore, the consistent increase in Dirichlet energy when using dropout provides em-

Table 2: Graph classification results on two peptide datasets from LRGB (Dwivedi et al., 2022).

Model	Peptides-func	Peptides-struct
# graphs	15,535	15,535
Avg. # nodes	150.9	150.9
Avg. # edges	307.3	307.3
Metric	AP ↑	MAE ↓
GT	0.6326 ± 0.0126	0.2529 ± 0.0016
SAN+RWSE	0.6439 ± 0.0075	0.2545 ± 0.0012
GraphGPS	0.6535 ± 0.0041	0.2500 ± 0.0012
MGT+WavePE	0.6817 ± 0.0064	0.2453 ± 0.0025
DRew	0.7150 ± 0.0044	0.2536 ± 0.0015
Exphormer	0.6527 ± 0.0043	0.2481 ± 0.0007
Graph-MLPMixer	0.6970 ± 0.0080	0.2475 ± 0.0015
GRIT	0.6988 ± 0.0082	0.2460 ± 0.0012
CKGCN	0.6952 ± 0.0068	0.2477 ± 0.0019
GRED	0.7085 ± 0.0027	0.2503 ± 0.0019
Graph Mamba	0.6972 ± 0.0100	0.2477 ± 0.0019
GCN	0.7015 ± 0.0021	**0.2437** ± 0.0012
Dirichlet energy	9.649	6.121
GCN w/o dp	0.6484 ± 0.0034	0.2541 ± 0.0026
Dirichlet energy	6.488	3.725

Table 3: Graph classification results on two image datasets from (Dwivedi et al., 2023).

Model	MNIST	CIFAR10
# graphs	70,000	60,000
Avg. # nodes	70.6	117.6
Avg. # edges	564.5	941.1
Metric	Accuracy ↑	Accuracy ↑
GT	90.831 ± 0.161	59.753 ± 0.293
SAN+RWSE	-	-
GraphGPS	98.051 ± 0.126	72.298 ± 0.356
MGT+WavePE	-	-
DRew	-	-
Exphormer	98.550 ± 0.039	74.696 ± 0.125
Graph-MLPMixer	97.422 ± 0.110	73.961 ± 0.330
GRIT	98.108 ± 0.111	76.468 ± 0.881
CKGCN	98.423 ± 0.155	72.785 ± 0.436
GRED	98.383 ± 0.012	76.853 ± 0.185
Graph Mamba	98.392 ± 0.183	74.563 ± 0.379
GatedGCN	**98.783** ± 0.122	**78.231** ± 0.274
Dirichlet energy	20.920	25.121
GatedGCN w/o dp	98.235 ± 0.136	71.384 ± 0.397
Dirichlet energy	14.242	13.587

pirical evidence for our theoretical insight into dropout's crucial role in mitigating oversmoothing in GCNs, particularly evident in larger graphs. The complementary roles of dropout and batch normalization are demonstrated by the performance drop when either is removed, supporting our analysis of their synergistic interaction in GCNs.

4.3 GRAPH-LEVEL CLASSIFICATION RESULTS

Our graph-level classification results, presented in Tables 2 and 3, further validate the broad applicability of our theoretical framework. First, compared to recent SOTA models, we observe that simply tuning dropout enables GNNs to achieve SOTA performance on three datasets and is competitive with the best single-model results on the remaining dataset. Second, the significant accuracy improvements on graph-level tasks such as Peptides-func and CIFAR10 highlight that our insights extend beyond node classification. The varying degrees of improvement across different graph datasets are consistent with our theory that dropout provides adaptive regularization tailored to graph properties. Third, the consistent increase in Dirichlet energy when using dropout supports our theoretical analysis of dropout's role in preserving feature diversity.

These results robustly validate our theory, showing that dropout in GCNs produces dimension-specific stochastic sub-graphs, has degree-dependent effects, mitigates oversmoothing, and offers topology-aware regularization. Combined with batch normalization, dropout enhances GCN performance on graph-level tasks, affirming the relevance and utility of our framework and suggesting directions for improving GNN architectures.

5 CONCLUSIONS

Our comprehensive theoretical analysis of dropout in GCNs has unveiled complex interactions between regularization, graph structure, and model performance that challenge traditional understanding. These insights not only deepen our understanding of how dropout functions in graph-structured data but also open new avenues for research and development in graph representation learning. Our findings suggest the need to reimagine regularization techniques for graph-based models, explore adaptive and structure-aware dropout strategies, and carefully balance local and global information in GCN architectures. Furthermore, the observed synergies between dropout and batch normalization point towards more holistic approaches to regularization in GNNs. As we move forward, this work lays a foundation for developing more robust and effective graph learning algorithms, with potential applications in dynamic graphs, large-scale graph sampling, and adversarial robustness. Ultimately, this research contributes to bridging the gap between the empirical success of GNNs and their theoretical foundations, paving the way for designing graph learning models.

REFERENCES

Alessandro Achille and Stefano Soatto. Information dropout: Learning optimal representations through noisy computation. *IEEE transactions on pattern analysis and machine intelligence*, 40 (12):2897–2905, 2018.

Pierre Baldi and Peter J Sadowski. Understanding dropout. *Advances in neural information processing systems*, 26, 2013.

Ali Behrouz and Farnoosh Hashemi. Graph mamba: Towards learning on graphs with state space models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 119–130, 2024.

Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

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

Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. Nagphormer: A tokenized graph transformer for node classification in large graphs. In *The Eleventh International Conference on Learning Representations*, 2022.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pp. 1725–1735. PMLR, 2020.

Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. *Advances in neural information processing systems*, 32, 2019.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2020.

Weilin Cong, Morteza Ramezani, and Mehrdad Mahdavi. On provable benefits of depth in training graph convolutional networks. *Advances in Neural Information Processing Systems*, 34:9936–9949, 2021.

Nima Dehmamy, Albert-László Barabási, and Rose Yu. Understanding the representation power of graph neural networks in learning graph topology. *Advances in Neural Information Processing Systems*, 32, 2019.

Chenhui Deng, Zichao Yue, and Zhiru Zhang. Polynormer: Polynomial-expressive graph transformer in linear time. *arXiv preprint arXiv:2403.01232*, 2024.

Yuhui Ding, Antonio Orvieto, Bobby He, and Thomas Hofmann. Recurrent distance filtering for graph representation learning. In *Forty-first International Conference on Machine Learning*, 2024.

Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *Advances in neural information processing systems*, 32, 2019.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Vijay Prakash Dwivedi, Ladislav Rampášek, Mikhail Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. *arXiv preprint arXiv:2206.08164*, 2022.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

Pascal Esser, Leena Chennuru Vankadara, and Debarghya Ghoshdastidar. Learning theory can (sometimes) explain generalisation in graph neural networks. *Advances in Neural Information Processing Systems*, 34:27043–27056, 2021.

Taoran Fang, Zhiqing Xiao, Chunping Wang, Jiarong Xu, Xuan Yang, and Yang Yang. Dropmessage: Unifying random dropping for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 4267–4275, 2023.

Jiarui Feng, Yixin Chen, Fuhai Li, Anindya Sarkar, and Muhan Zhang. How powerful are k-hop message passing graph neural networks. *Advances in Neural Information Processing Systems*, 35: 4776–4790, 2022.

Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33:22092–22103, 2020.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pp. 2083–2092. PMLR, 2019.

Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pp. 3419–3430. PMLR, 2020.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. *Advances in neural information processing systems*, 32, 2019.

Benjamin Gutteridge, Xiaowen Dong, Michael M Bronstein, and Francesco Di Giovanni. Drew: Dynamically rewired message passing with delay. In *International Conference on Machine Learning*, pp. 12252–12267. PMLR, 2023.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Xiaoxin He, Bryan Hooi, Thomas Laurent, Adam Perold, Yann LeCun, and Xavier Bresson. A generalization of vit/mlp-mixer to graphs. In *International Conference on Machine Learning*, pp. 12724–12745. PMLR, 2023.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arxiv 2012. *arXiv preprint arXiv:1207.0580*, 2012.

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. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=SJU4ayYgl`.

Kezhi Kong, Jiuhai Chen, John Kirchenbauer, Renkun Ni, C. Bayan Bruss, and Tom Goldstein. GOAT: A global transformer on large-scale graphs. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17375–17390. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/kong23a.html`.

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Renjie Liao, Raquel Urtasun, and Richard Zemel. A pac-bayesian approach to generalization bounds for graph neural networks. *arXiv preprint arXiv:2012.07690*, 2020.

Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33:19620–19631, 2020.

Shaogao Lv. Generalization bounds for graph convolutional neural networks via rademacher complexity. *arXiv preprint arXiv:2102.10234*, 2021.

Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. *arXiv preprint arXiv:2305.17589*, 2023.

Liheng Ma, Soumyasundar Pal, Yitian Zhang, Jiaming Zhou, Yingxue Zhang, and Mark Coates. Ckgconv: General graph convolution with continuous kernels. *arXiv preprint arXiv:2404.13604*, 2024.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

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

Pietro Morerio, Jacopo Cavazza, Riccardo Volpi, René Vidal, and Vittorio Murino. Curriculum dropout. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3544–3552, 2017.

Nhat Khang Ngo, Truong Son Hy, and Risi Kondor. Multiresolution graph transformers and wavelet positional encoding for learning long-range and hierarchical structures. *The Journal of Chemical Physics*, 159(3), 2023.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *arXiv preprint arXiv:2205.12454*, 2022.

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

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Hkx1qkrKPr`.

Franco Scarselli, Ah Chung Tsoi, and Markus Hagenbuchner. The vapnik–chervonenkis dimension of graph and recursive neural networks. *Neural Networks*, 108:248–259, 2018.

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.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. *arXiv preprint arXiv:2303.06147*, 2023.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Jan Tönshoff, Martin Ritzert, Eran Rosenbluth, and Martin Grohe. Where did the gap go? reassessing the long-range graph benchmark. *arXiv preprint arXiv:2309.00367*, 2023.

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

Saurabh Verma and Zhi-Li Zhang. Stability and generalization of graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1539–1548, 2019.

Minh Vu and My T Thai. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. *Advances in neural information processing systems*, 33:12225–12235, 2020.

Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. *Advances in neural information processing systems*, 26, 2013.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pp. 1058–1066. PMLR, 2013.

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

Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in Neural Information Processing Systems*, 35:27387–27401, 2022.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Simplifying and empowering transformers for large-graph representations. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=R4xpvDTWkV`.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

Han Yang, Kaili Ma, and James Cheng. Rethinking graph regularization for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4573–4581, 2021.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 430–438, 2020.

Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *International conference on machine learning*, pp. 12241–12252. PMLR, 2021.

Shuai Zhang, Meng Wang, Sijia Liu, Pin-Yu Chen, and Jinjun Xiong. Fast learning of graph neural networks with guaranteed generalizability: one-hidden-layer case. In *International Conference on Machine Learning*, pp. 11268–11277. PMLR, 2020.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223*, 2019.

A APPENDIX

A.1 PROOF OF THEOREM 11

Proof. Let's approach this proof:

Step 1: For a single feature j, the probability that an edge is present in the sub-graph $G_t^{(l,j)}$ is $(1-p)^2$, as both endpoints need to retain this feature.

Step 2: The probability that an edge is not present in $G_t^{(l,j)}$ is $1-(1-p)^2 = p(2-p)$.

Step 3: For a sub-graph to be identical to the original graph, all edges must be present. The probability of this is: $((1-p)^2)^{|\mathcal{E}|} = (1-p)^{2|\mathcal{E}|}$.

Step 4: Therefore, the probability that $G_t^{(l,j)}$ is different from the original graph (i.e., unique) is $1-(1-p)^{2|\mathcal{E}|}$.

Step 5: Define an indicator random variable X_j for each feature j:

$$X_j = \begin{cases} 1 & \text{if } G_t^{(l,j)} \text{ is unique} \\ 0 & \text{otherwise} \end{cases}.$$

Step 6: We have:

$$P(X_j = 1) = 1-(1-p)^{2|\mathcal{E}|}][P(X_j = 0) = (1-p)^{2|\mathcal{E}|}.$$

Step 7: The expected value of X_j is:

$$\mathbb{E}[X_j] = 1 \cdot P(X_j = 1) + 0 \cdot P(X_j = 0) = 1-(1-p)^{2|\mathcal{E}|}.$$

Step 8: The total number of unique sub-graphs is $\sum_{j=1}^{d_l} X_j$. By the linearity of expectation:

$$\mathbb{E}[|G_t^{(l,j)} \mid j = 1,\dots,d_l|] = \mathbb{E}[\sum_{j=1}^{d_l} X_j] = \sum_{j=1}^{d_l} \mathbb{E}[X_j] = d_l(1-(1-p)^{2|\mathcal{E}|}).$$

This completes the proof. □

A.2 PROOF OF THEOREM 15

Proof. We start with the definition of feature energy:

$$E(\boldsymbol{H}^{(l)}) = \frac{1}{2|\mathcal{E}|} \sum_{i,j\in\mathcal{E}} \|\boldsymbol{h}_i^{(l)} - \boldsymbol{h}_j^{(l)}\|_2^2$$

Step 1: Taking the expectation:

$$\mathbb{E}[E(\boldsymbol{H}^{(l)})] = \frac{1}{2|\mathcal{E}|} \sum_{i,j\in\mathcal{E}} \mathbb{E}[\|\boldsymbol{h}_i^{(l)} - \boldsymbol{h}_j^{(l)}\|_2^2].$$

.

Step 2: Since $\sum_{(i,j)\in\mathcal{E}}[\|\boldsymbol{h}_i\|^2 + \|\boldsymbol{h}_j\|^2] = 2\sum_i deg_i \|\boldsymbol{h}_i\|^2$:

$$\begin{aligned}
\frac{1}{2|\mathcal{E}|} \sum_{i,j\in\mathcal{E}} \mathbb{E}[\|\boldsymbol{h}_i^{(l)} - \boldsymbol{h}_j^{(l)}\|_2^2] &= \frac{1}{2|\mathcal{E}|} \sum_{i,j\in\mathcal{E}} \mathbb{E}[\|\frac{1}{1-p}\boldsymbol{M}_i^{(l)} \odot \boldsymbol{z}_i^{(l)} - \frac{1}{1-p}\boldsymbol{M}_j^{(l)} \odot \boldsymbol{z}_j^{(l)}\|_2^2] \\
&= \frac{1}{2|\mathcal{E}|(1-p)^2} \sum_{i,j\in\mathcal{E}} \mathbb{E}[\|\boldsymbol{M}_i^{(l)} \odot \boldsymbol{z}_i^{(l)} - \boldsymbol{M}_j^{(l)} \odot \boldsymbol{z}_j^{(l)}\|_2^2] \\
&= \frac{1}{2|\mathcal{E}|(1-p)^2} \sum_{i,j\in\mathcal{E}} [(1-p)(\|\boldsymbol{z}_i^{(l)}\|_2^2 + \|\boldsymbol{z}_j^{(l)}\|_2^2) - 2(1-p)^2(\boldsymbol{z}_i^{(l)})^T \boldsymbol{z}_j^{(l)}] \\
&= \frac{1}{1-p}\frac{1}{|\mathcal{E}|} \sum_i deg_i \|\boldsymbol{z}_i^{(l)}\|_2^2 - \frac{1}{|\mathcal{E}|}\text{Tr}(\boldsymbol{Z}^T \boldsymbol{A}\boldsymbol{Z})
\end{aligned}$$

where $\boldsymbol{z}_i = \sigma(\sum_k \tilde{\boldsymbol{A}}_{ik}\boldsymbol{h}_k^{(l-1)}\boldsymbol{W}^{(l)})$.

Step 3: Since $deg_i \leq deg_{max}$ for all i:

$$\frac{1}{|\mathcal{E}|}\sum_i deg_i \|\boldsymbol{z}_i\|_2^2 \leq \frac{deg_{\max}}{|\mathcal{E}|}\sum_i \|\boldsymbol{z}_i\|_2^2 = \frac{deg_{\max}}{|\mathcal{E}|}\|\boldsymbol{Z}\|_F^2.$$

Step 4: By ReLU non-negative homogeneity and submultiplicative property:

$$||\boldsymbol{Z}^{(l)}||_F^2 \leq ||\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)}||_F^2 \leq ||\boldsymbol{W}^{(l)}||_2^2||\tilde{\boldsymbol{A}}||_2^2||\boldsymbol{H}^{(l-1)}||_F^2$$

Step 5: By dropout scaling with probability p:

$$||\boldsymbol{H}^{(l-1)}||_F^2 = \frac{1}{1-p}||\boldsymbol{Z}^{(l-1)}||_F^2$$

Step 6: By applying steps 4-5 recursively:

$$||\boldsymbol{Z}^{(l)}||_F^2 \leq (\frac{1}{1-p})^{l-1}||\tilde{\boldsymbol{A}}||_2^{2l}\prod_{i=1}^{l}||\boldsymbol{W}^{(i)}||_2^2||\boldsymbol{X}||_F^2$$

Step 7: Combining all inequalities:

$$\mathbb{E}[E(\boldsymbol{H}^{(l)})] \leq \frac{deg_{\max}}{|\mathcal{E}|}(\frac{1}{1-p})^{l}||\tilde{\boldsymbol{A}}||_2^{2l}\prod_{i=1}^{l}||\boldsymbol{W}^{(i)}||_2^2||\boldsymbol{X}||_F^2$$

□

A.3 PROOF OF THEOREM 16

Proof. The proof proceeds in several steps:

Step 1: Dropout Effect as Perturbation. Consider layer l with dropout probability p_l. The effect of dropout is a perturbation $\delta^{(l)}$:

$$\delta^{(l)} = \frac{1}{1-p_l}M^{(l)} \odot \sigma(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)}) - \sigma(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)}), \tag{8}$$

where $M^{(l)}$ has elements drawn from Bernoulli$(1-p_l)$.

Step 2: Perturbation Propagation. Let $F_l(x)$ denote the network output with dropout applied up to layer l. Define:

$$L_l = (\prod_{i=l+1}^{L}\|\boldsymbol{W}^{(i)}\| \cdot \|\tilde{\boldsymbol{A}}\|) \cdot (\|\tilde{\boldsymbol{A}}\|^l \prod_{i=1}^{l}\|\boldsymbol{W}^{(i)}\|) \cdot \|\boldsymbol{H}^{(0)}\| \tag{9}$$

By the properties of operator norms and composition:

$$\|F_l(x) - F_{l-1}(x)\| \leq L_l\|\delta^{(l)}\| \tag{10}$$

Step 3: Bounding Perturbation Magnitude. For the perturbation magnitude:

$$\begin{aligned}\mathbb{E}[\|\delta^{(l)}\|^2] &= \mathbb{E}[\|\frac{1}{1-p_l}M^{(l)} \odot \sigma(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)}) - \sigma(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)})\|^2] & (11)\\ &= \frac{p_l}{1-p_l}\|\sigma(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)})\|_F^2 & (12)\end{aligned}$$

where we use $\mathbb{E}[(\boldsymbol{M}^{(l)})^2] = \mathbb{E}[M^{(l)}] = 1 - p_l$.

Step 4: Loss Stability. By the Lipschitz property of the loss function:

$$\begin{aligned}\mathbb{E}[|L(F_l(x)) - L(F_{l-1}(x))|] &\leq L_{loss} \cdot \mathbb{E}[\|F_l(x) - F_{l-1}(x)\|] & (13)\\ &\leq L_{loss} \cdot L_l \cdot \mathbb{E}[\|\delta^{(l)}\|] & (14)\\ &\leq L_{loss} \cdot L_l \cdot \sqrt{\frac{p_l}{1-p_l}}\|\sigma(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)})\|_F & (15)\end{aligned}$$

where we used Jensen's inequality in the last step.

Step 5: Layer Aggregation. The total expected change in loss:

$$\mathbb{E}[|L(F(x)) - L(F_{\text{no dropout}}(x))|] \leq \sum_{l=1}^{L} R_l \tag{16}$$

where

$$R_l = L_{loss} \cdot L_l \cdot \sqrt{\frac{p_l}{1-p_l}} \|\sigma(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)})\|_F \tag{17}$$

Step 6: Concentration Bound. Let $f(S) = \mathbb{E}_D[L(F(x))] - \mathbb{E}_S[L(F(x))]$ where S is training set. When changing one example in S to S', the maximum change is:

$$\|f(S) - f(S')\| \leq \frac{2}{n} \sum_{l=1}^{L} R_l \tag{18}$$

where

$$R_l = L_{loss} \cdot L_l \cdot \sqrt{\frac{p}{1-p}} \|\sigma(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)})\|_F \tag{19}$$

By McDiarmid's inequality:

$$P(\mathbb{E}_D[L(F(x))] - \mathbb{E}_S[L(F(x))] > \epsilon) \leq \exp\left(-\frac{2n\epsilon^2}{4(\sum_{l=1}^{L} R_l)^2}\right) \tag{20}$$

Set probability to δ:

$$\exp\left(-\frac{2n\epsilon^2}{4(\sum_{l=1}^{L} R_l)^2}\right) = \delta \tag{21}$$

Solve for ϵ:

$$-\frac{2n\epsilon^2}{4(\sum_{l=1}^{L} R_l)^2} = \ln(\delta) \tag{22}$$

$$\epsilon^2 = \frac{2(\sum_{l=1}^{L} R_l)^2 \ln(1/\delta)}{n} \tag{23}$$

$$\epsilon = O\left(\sqrt{\frac{\ln(1/\delta)}{n}}\right) \sum_{l=1}^{L} R_l \tag{24}$$

Therefore, with probability at least $1 - \delta$:

$$\mathbb{E}_D[L(F(x))] - \mathbb{E}_S[L(F(x))] \leq O\left(\sqrt{\frac{\ln(1/\delta)}{n}}\right) \sum_{l=1}^{L} L_{loss} \cdot L_l \cdot \sqrt{\frac{p}{1-p}} \|\sigma(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)})\|_F \tag{25}$$

□

A.4 PROOF OF THEOREM 17

Proof. **Step 1:** Start with feature energy and node representation:

$$E(\boldsymbol{H}^{(l)}) = \frac{1}{2|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} \|\boldsymbol{h}_i^{(l)} - \boldsymbol{h}_j^{(l)}\|^2$$

$$\boldsymbol{h}_i^{(l)} = \frac{1}{1-p} \boldsymbol{M}_i^{(l)} \odot \boldsymbol{z}_i^{(l)}$$

where $\boldsymbol{z}_i^{(l)} \in \mathbb{R}^{d_l}$ and $\boldsymbol{z}_i^{(l)} = \sigma(\mathrm{BN}(\sum_k \tilde{\boldsymbol{A}}_{ik}\boldsymbol{h}_k^{(l-1)}\boldsymbol{W}^{(l)}))$

Step 2: For the BN output before ReLU at layer l, for each feature dimension $d \in \{1, ..., d_l\}$:

$$(\boldsymbol{Y}^{(l)})_{:,d} = \mathrm{BN}((\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)})_{:,d}) = \gamma_d^{(l)} \frac{(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)})_{:,d} - \mu_d^{(l)}}{\sqrt{(\sigma_d^{(l)})^2 + \epsilon}} + \beta_d^{(l)}$$

Step 3: For ReLU activation z = max(0, y) at layer l, for each dimension d:

$$\mathbb{E}[(z_d^{(l)})^2] \geq \Phi(\beta_d^{(l)}/\gamma_d^{(l)}) \cdot (\beta_d^{(l)})^2$$

where Φ is the standard normal CDF.

Step 4: Using the BN-induced bound:

$$\begin{aligned}\|\boldsymbol{z}_i^{(l)}\|^2 &= \sum_{d=1}^{d_l} (z_i^{(l)})_d^2 \\ &\geq \sum_{d=1}^{d_l} \Phi(\beta_d^{(l)}/\gamma_d^{(l)}) \cdot (\beta_d^{(l)})^2 > 0\end{aligned}$$

Step 5: For feature energy with merged terms:

$$\begin{aligned}E(\boldsymbol{H}^{(l)}) &= \frac{1}{2|\mathcal{E}|}\sum_{(i,j)\in\mathcal{E}} [\frac{1}{1-p}(\|\boldsymbol{z}_i^{(l)}\|^2 + \|\boldsymbol{z}_j^{(l)}\|^2) - 2(\boldsymbol{z}_i^{(l)})^T\boldsymbol{z}_j^{(l)}] \\ &\geq \frac{1}{2|\mathcal{E}|}\sum_{(i,j)\in\mathcal{E}} [\frac{1}{1-p}(\|\boldsymbol{z}_i^{(l)}\|^2 + \|\boldsymbol{z}_j^{(l)}\|^2) - (\|\boldsymbol{z}_i^{(l)}\|^2 + \|\boldsymbol{z}_j^{(l)}\|^2)] \\ &= \frac{1}{2|\mathcal{E}|}\sum_{(i,j)\in\mathcal{E}} (\frac{1}{1-p} - 1)(\|\boldsymbol{z}_i^{(l)}\|^2 + \|\boldsymbol{z}_j^{(l)}\|^2) \\ &= \frac{p}{1-p}\frac{1}{2|\mathcal{E}|}\sum_{(i,j)\in\mathcal{E}} (\|\boldsymbol{z}_i^{(l)}\|^2 + \|\boldsymbol{z}_j^{(l)}\|^2) \\ &= \frac{p}{1-p}\frac{1}{2|\mathcal{E}|}\sum_i deg_i\|\boldsymbol{z}_i^{(l)}\|^2 \\ &\geq \frac{p\,deg_{\min}}{1-p}\frac{1}{2|\mathcal{E}|}\|\boldsymbol{Z}^{(l)}\|_F^2\end{aligned}$$

Then with BN bound:

$$E(\boldsymbol{H}^{(l)}) \geq \frac{p\,deg_{\min}}{1-p}\frac{1}{2|\mathcal{E}|}\sum_{d=1}^{d_l} \Phi(\beta_d^{(l)}/\gamma_d^{(l)}) \cdot (\beta_d^{(l)})^2$$

□

A.5 EFFECT OF DROPOUT ON MAX SINGULAR VALUES OF THE WEIGHT MATRICES

We analyze why dropout leads to larger weight matrices in terms of spectral norm $\|\boldsymbol{W}\|_2$. Consider the gradient update for weights $\boldsymbol{W}^2$ between layers:

$$\frac{\partial L}{\partial \boldsymbol{W}^2} = (\tilde{\boldsymbol{A}}\boldsymbol{H}^1_{drop})^\top \times \frac{\partial L}{\partial \boldsymbol{H}^2} = (\tilde{\boldsymbol{A}}(\boldsymbol{H}^1 \odot \boldsymbol{M}^1)/(1-p))^\top \times \frac{\partial L}{\partial \boldsymbol{H}^2} \quad (26)$$

where p is the dropout rate and M^1 is the dropout mask. This leads to weight updates:

$$\Delta\boldsymbol{W}^2 = -\eta(\tilde{\boldsymbol{A}}\boldsymbol{H}^1_{drop})^\top \times \frac{\partial L}{\partial \boldsymbol{H}^2} = -\eta(\tilde{\boldsymbol{A}}(\boldsymbol{H}^1 \odot \boldsymbol{M}^1)/(1-p))^\top \times \frac{\partial L}{\partial \boldsymbol{H}^2} \quad (27)$$

The $1/(1-p)$ scaling factor in dropout has two key effects: 1) For surviving features (where $M^1_{ij} = 1$), the gradient is amplified by $1/(1-p)$. This leads to larger updates for these weights during training. 2) During each iteration, different subsets of features survive, but their gradients are consistently scaled up. Over many iterations, this accumulates to larger weight values despite the unbiased expectation maintained by dropout. Specifically, with dropout rate p when $p = 0.5$, surviving gradients are doubled. This amplification effect compounds over training iterations. While dropout maintains unbiased expected values during forward propagation, the consistent gradient scaling during backward propagation leads to systematically larger weight magnitudes. Empirically, we observe that higher dropout rates correlate with larger spectral norms $\|\boldsymbol{W}\|_2^2$ (as shown in Figure 10), supporting this theoretical analysis. The increased weight magnitudes directly contribute to higher Dirichlet energy $E(\boldsymbol{H}^2)$ during inference, as:

$$E(\boldsymbol{H}^2) = \sum_{(i,j)\in\mathcal{E}} \|\boldsymbol{h}_i^2 - \boldsymbol{h}_j^2\|_2^2/2|\mathcal{E}| \tag{28}$$

where larger weights produce more distinctive features between connected nodes, helping mitigate oversmoothing.

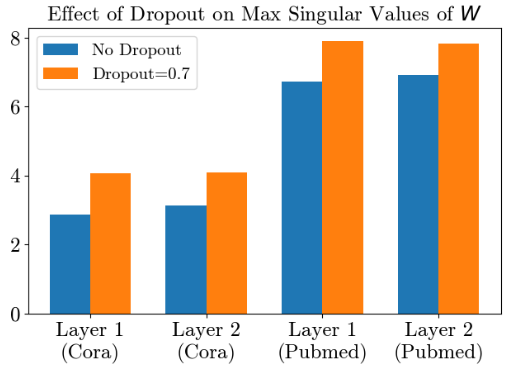

Figure 10: Effect of dropout on max singular values of the weight matrices.

A.6 ADDITIONAL EXPERIMENTAL RESULTS AND DISCUSSION ON DROPOUT VARIANTS

Different methods apply masks at various stages of graph neural network training:

DropNode (Feng et al., 2020):

$$\boldsymbol{M}_d = \tilde{\boldsymbol{A}}((\boldsymbol{M}_{node} \odot \boldsymbol{H}^{(l-1)})\boldsymbol{W}^{(l)})_d$$

where $\boldsymbol{M}_{node}$ is a node-wise mask applied to all dimensions.

DropEdge Rong et al. (2019):

$$\boldsymbol{M}_d = (\boldsymbol{M}_{edge} \odot \tilde{\boldsymbol{A}})(\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)})_d$$

where $\boldsymbol{M}_{edge}$ is a single mask for the adjacency matrix.

DropMessage Fang et al. (2023):

$$\boldsymbol{M}_d = \tilde{\boldsymbol{A}}(\boldsymbol{M}_{msg_d} \odot (\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)}))_d$$

where $\boldsymbol{M}_{msg_d}$ is a dimension-specific message mask.

Dropout (Srivastava et al., 2014):

$$\boldsymbol{M}_d = \boldsymbol{M}_{feat_d} \odot \tilde{\boldsymbol{A}}(\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)})_d$$

where $\boldsymbol{M}_{feat_d}$ is a dimension-specific feature mask.

Additionally, these methods exhibit different subgraph formation and degree-dependent effects:

DropNode:

$$\mathcal{G}_t = (\mathcal{V} \setminus \mathcal{V}_{dropped}, \mathcal{E} \setminus \{(i,j)|i \in \mathcal{V}_{dropped} \text{ or } j \in \mathcal{V}_{dropped}\}), \mathcal{V}_{dropped} = \{i|\boldsymbol{M}_{node_i} = 0\}$$

$$\mathbb{E}[deg_i^{eff}(t)] = deg_i \prod_{j \in \mathcal{N}(i)} (1-p)$$

DropEdge:

$$\mathcal{G}_t = (\mathcal{V}, \mathcal{E} \setminus \mathcal{E}_{dropped}), \mathcal{E}_{dropped} = \{(i,j)|\boldsymbol{M}_{edge_{ij}} = 0\}$$

$$\mathbb{E}[deg_i^{eff}(t)] = (1-p)deg_i$$

DropMessage:

$$\mathcal{G}_t^d = (\mathcal{V}, \mathcal{E}_t^d), \mathcal{E}_t^d = \{(i,j) \in \mathcal{E}|\boldsymbol{M}_{msg_{d_{ij}}} \neq 0\}$$

$$\mathbb{E}[deg_i^{eff}(t)] = (1-p)deg_i$$

Dropout:

$$\mathcal{G}_t^d = (\mathcal{V}, \mathcal{E}_t^d), \mathcal{E}_t^d = \{(i,j) \in \mathcal{E}|\mathbf{M}_{feat_{d_i}} \neq 0 \text{ and } \boldsymbol{M}_{feat_{d_j}} \neq 0\}$$

$$\mathbb{E}[deg_i^{eff}(t)] = (1-p)^2 deg_i$$

Overall, dropout's quadratic degree-dependent effect makes it particularly effective by providing natural adaptive regularization at hub nodes, where over-mixing of features is most problematic. While other methods also provide degree-dependent regularization, they either lack dimension-specific patterns (DropNode, DropEdge) or do not provide sufficiently strong control at high-degree nodes (DropMessage).

To further explore the practical impact of these different regularization techniques, we conducted hyperparameter tuning for DropEdge, DropNode, and DropMessage on the Cora, Citeseer, and Pubmed datasets. The results, summarized in Table 4, demonstrate that while these methods yield comparable performance, traditional dropout generally performs best.

Table 4: Experimental results of different regularization methods on Cora, Citeseer, and PubMed.

	Cora (GCN)	**CiteSeer (GCN)**	**PubMed (GCN)**	**Cora (SAGE)**	**CiteSeer (SAGE)**	**PubMed (SAGE)**	**Cora (GAT)**	**CiteSeer (GAT)**	**PubMed (GAT)**
GNN	83.18 ± 1.22	70.48 ± 0.45	79.40 ± 1.02	83.06 ± 0.80	69.68 ± 0.82	76.40 ± 1.48	82.58 ± 1.47	71.08 ± 0.42	79.28 ± 0.58
GNN+Dropout	**85.22 ± 0.66**	**73.24 ± 0.63**	**81.08 ± 1.16**	**84.14 ± 0.63**	71.62 ± 0.29	77.86 ± 0.79	**83.92 ± 1.29**	**72.00 ± 0.91**	**80.48 ± 0.99**
GNN+DropEdge	84.88 ± 0.68	72.96 ± 0.38	80.42 ± 1.15	83.10 ± 0.51	71.72 ± 0.92	77.88 ± 1.31	83.44 ± 0.78	71.60 ± 1.14	79.82 ± 0.68
GNN+DropNode	84.92 ± 0.52	73.08 ± 0.39	80.60 ± 0.49	83.42 ± 0.58	**71.92 ± 0.65**	78.06 ± 1.09	83.80 ± 0.97	71.30 ± 0.87	79.50 ± 0.68
GNN+DropMessage	84.78 ± 0.58	73.12 ± 1.19	80.92 ± 0.88	83.18 ± 0.62	71.22 ± 1.34	**78.20 ± 0.80**	83.46 ± 1.06	71.38 ± 1.12	79.36 ± 1.22