Stability and Generalization of Graph Convolutional Neural Networks

Saurabh Verma
Department of Computer Science
University of Minnesota, Twin Cities
verma076@umn.edu

Zhi-Li Zhang
Department of Computer Science
University of Minnesota, Twin Cities
zhzhang@cs.umn.edu

ABSTRACT

Inspired by convolutional neural networks on 1D and 2D data, graph convolutional neural networks (GCNNs) have been developed for various learning tasks on graph data, and have shown superior performance on real-world datasets. Despite their success, there is a dearth of *theoretical* explorations of GCNN models such as their generalization properties. In this paper, we take a first step towards developing a deeper theoretical understanding of GCNN models by analyzing the stability of single-layer GCNN models and deriving their generalization guarantees in a semi-supervised graph learning setting. In particular, we show that the algorithmic stability of a GCNN model depends upon the largest absolute eigenvalue of its graph convolution filter. Moreover, to ensure the uniform stability needed to provide strong generalization guarantees, the largest absolute eigenvalue must be independent of the graph size. Our results shed new insights on the design of new & improved graph convolution filters with guaranteed algorithmic stability. We evaluate the generalization gap and stability on various real-world graph datasets and show that the empirical results indeed support our theoretical findings. To the best of our knowledge, we are the first to study stability bounds on graph learning in a semi-supervised setting and derive generalization bounds for GCNN models.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Theory of computation** → *Graph algorithms analysis*; *Semi-supervised learning*.

KEYWORDS

Deep learning, graph convolutional neural networks, graph mining, stability, generalization guarantees

ACM Reference Format:
Saurabh Verma and Zhi-Li Zhang. 2019. Stability and Generalization of Graph Convolutional Neural Networks. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3292500.3330956

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330956

1 INTRODUCTION

Building upon the huge success of deep learning in computer vision (CV) and natural language processing (NLP), Graph Convolutional Neural Networks (GCNNs) [25] have recently been developed for tackling various learning tasks on graph-structured datasets. These models have shown superior performance on real-world datasets from various domains such as node labelling on social networks [26], link prediction in knowledge graphs [37] and molecular graph classification in quantum chemistry [19] . Due to the versatility of graph-structured data representation, GCNN models have been incorporated in many diverse applications, e.g., question-answer systems [39] in NLP and/or image semantic segmentation [36] in CV. While various versions of GCCN models have been proposed, there is a dearth of *theoretical* explorations of GCNN models ([46] is one of few exceptions which explores the *discriminant* power of GCNN models)—especially, in terms of their *generalization* properties and (*algorithmic*) *stability*. The latter is of particular import, as the stability of a learning algorithm plays a crucial role in generalization.

The generalization of a learning algorithm can be explored in several ways. One of the earliest and most popular approach is Vapnik–Chervonenkis (VC)-theory [6] which establishes generalization errors in terms VC-dimensions of a learning algorithm. Unfortunately, VC-theory is not applicable for learning algorithms with unbounded VC-dimensions such as neural networks. Another way to show generalization is to perform the Probably Approximately Correct (PAC) [23] analysis, which is generally difficult to do in practice. The third approach, which we adopt, relies on deriving stability bounds of a learning algorithm, often known as *algorithmic stability* [7]. The idea behind algorithmic stability is to understand how the learning function changes with small changes in the input data. Over the past decade, several definitions of algorithmic stability have been developed [1, 2, 7, 17, 32], including uniform stability, hypothesis stability, pointwise hypothesis stability, error stability and cross-validation stability, each yielding either a tight or loose bound on the generalization errors. For instance, learning algorithm based on Tikhonov regularization satisfy the *uniform stability* criterion (the strongest stability condition among all existing forms of stability), and thus are generalizable.

In this paper, we take a first step towards developing a deeper theoretical understanding of GCNN models by analyzing the (uniform) stability of GCNN models and thereby deriving their generalization guarantees. For simplicity of exposition, we focus on *single layer*GCNN models in a semi-supervised learning setting. The main result of this paper is that *(single layer) GCNN models with stable graph convolution filters can satisfy the strong notion of uniform stability and thus are generalizable.* More specifically, we show that the stability of a (single layer) GCNN model depends upon the *largest absolute eigenvalue* (the eigenvalue with the largest absolute value) of the graph filter it employs – or more generally, the largest *singular* value if the graph filter is *asymmetric* – and that the uniform stability criterion is met if the largest absolute eigenvalue (or singular value) is independent of the graph size, i.e., the number of nodes in the graph. As a consequence of our analysis, we establish that (appropriately) *normalized* graph convolution filters such as the symmetric normalized graph Laplacian or random walk based filters are all uniformly stable and thus are generalizable. In contrast, graph convolution filters based on the *unnormalized* graph Laplacian or adjacency matrix *do not* enjoy algorithmic stability, as their largest absolute eigenvalues grow as a function of the graph size. Empirical evaluations based on real world datasets support our theoretical findings: the generalization gap and weight parameters instability in case of unnormalized graph filters are *significantly higher* than those of the normalized filters. Our results shed new insights on the design of new & improved graph convolution filters with guaranteed algorithmic stability.

We remark that our GCNN generalization bounds obtained from algorithmic stability are non-asymptotic in nature, i.e., they do not assume any form of data distribution. Nor do they hinge upon the complexity of the hypothesis class, unlike the most uniform convergence bounds. We only assume that the activation & loss functions employed are Lipschitz continuous and smooth functions. These criteria are readily satisfied by several popular activation functions such as ELU (holds for $\alpha = 1$), Sigmoid and/or Tanh. To the best of our knowledge, we are the first to study stability bounds on graph learning in a semi-supervised setting and derive generalization bounds for GCCN models. Our analysis framework remains general enough and can be extended to theoretical stability analyses of GCCN models beyond a semi-supervised learning setting (where there is a single and fixed underlying graph structure) such as for the graph classification (where there are multiple graphs).

In summary, the major contributions of our paper are:

- We provide the first generalization bound on single layer GCNN models based on analysis of their algorithmic stability. We establish that GCNN models which employ graph filters with bounded eigenvalues that are independent of the graph size can satisfy the strong notion of uniform stability and thus are generalizable.
- Consequently, we demonstrate that many existing GCNN models that employ *normalized* graph filters satisfy the strong notion of uniform stability. We also justify the importance of employing batch-normalization in a GCNN architecture.
- Empirical evaluations of the generalization gap and stability using real-world datasets support our theoretical findings.

The paper is organized as follows. Section 2 reviews key generalization results for deep learning as well as regularized graphs and briefly discusses existing GCNN models. The main result is presented in Section 3 where we introduce the needed background and establish the GCNN generalization bounds step by step. In Section 4, we apply our results to existing graph convolution filters and GCNN architecture designs. In Section 5 we conduct empirical studies which complement our theoretical analysis. The paper is concluded in Section 6 with a brief discussion of future work.

2 RELATED WORK

Generalization Bounds on Deep Learning: Many theoretical studies have been devoted to understanding the representational power of neural networks by analyzing their capability as a universal function approximator as well as their depth efficiency [9, 13, 16, 31, 42]. In [13] the authors show that the number of hidden units in a shallow network has to grow exponentially (as opposed to a linear growth in a deep network) in order to represent the same function; thus depth yields much more compact representation of a function than having a wide-breadth. It is shown in [9] that convolutional neural networks with the ReLU activation function are universal function approximators with max pooling, but not with average pooling. The authors of [33] authors explore which complexity measure is more appropriate for explaining the generalization power of deep learning. The work most closest to ours is [22] where the authors derive upper bounds on the generalization errors for stochastic gradient methods. While also utilizing the notion of uniform stability [7], their analysis is concerned with the impact of SGD learning rates. More recently, through empirically evaluations on real-world datasets, it has been argued in [47] that the traditional measures of model complexity are not sufficient to explain the generalization ability of neural networks. Likely, in [24] several open-ended questions are posed regarding the (yet unexplained) generalization capability of neural networks, despite their possible algorithmic instability, non-robustness, and sharp minima.

Generalization Bounds on Regularized Graphs: Another line of work concerns with generalization bounds on regularized graphs in transductive settings [3, 5, 10, 41]. Of the most interest to ours is [5] where the authors provide theoretical guarantees for the generalization error based on Laplacian regularization, which are also derived based on the notion of algorithmic stability. Their generalization estimate is *inversely proportional* to the second smallest eigenvalue of the graph Laplacian. Unfortunately this estimate may be not yield desirable guarantee as the second smallest eigenvalue is dependent on both the graph structure and its size; it is in general difficult to remove this dependency via normalization. In contrast, our estimates are *directly proportional* to the largest absolute eigenvalue (or the largest singular value of an asymmetric graph filter), and can easily be made independent of the graph size by performing appropriate Laplacian normalization.

Graph Convolution Neural Networks: Coming from graph signal processing [38] domain, GCNN is defined as the problem of learning filter parameters in the graph Fourier transform [8]. Since then rapid progress has been made and GCNN model have improved in many aspects [4, 14, 15, 25, 30, 35, 45]. For instance in [30] parameterize graph filters using residual Laplacian matrix and in [40] authors used simply polynomial of adjacency matrix. Random walk and quantum walk based graph convolutions are also been proposed recently [14, 35, 48]. Similarly, graph convolutional operation has been generalized with the graph capsule notion in [45]. The authors of [21, 44] have also applied graph convolution to large graphs. Message passing neural networks (MPNNs) are also been developed [11, 18, 19, 29] which can be viewed as GCNN model since the notion of graph convolution operation remains the same. MPNNs can also be break into two step process where edge features are updated though message passing and then node features are updates using the information encoded in its nearby edges. This is similar to Embedding belief propagation message passing algorithm proposed in [11]. Several attempts have also been made to convert graph into regular grid structure for straight forwardly applying standard 2D or 1D CNNs [34, 43]. A very tangential approach was taken in [27] where authors design covariant neural network based on group theory for computing graph representation.

3 STABILITY AND GENERALIZATION GUARANTEES FOR GCNNS

To derive generalization guarantees of GCNNs based on algorithmic stability analysis, we adopt the strategy devised in [7]. It relies on bounding the output difference of a loss function due to a single data point perturbation. As stated earlier, there exist several different notions of algorithmic stability [7, 32]. In this paper, we focus on the strong notion of *uniform stability* (see Definition 1).

3.1 Graph Convolution Neural Networks

Notations: Let $G = (V, E, \mathbf{A})$ be a graph where V is the vertex set, E the edge set and $\mathbf{A}$ the adjacency matrix, with $N = |V|$ the graph size. We define the standard graph Laplacian $\mathbf{L} \in \mathbb{R}^{N\times N}$ as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the degree matrix. We define a graph filter, $g(\mathbf{L}) \in \mathbb{R}^{N\times N}$ as a function of the graph Laplacian $\mathbf{L}$ or a normalized (using $\mathbf{D}$) version of it. Let $\mathbf{U\Lambda U}^T$ be the eigen decomposition of L, with $\mathbf{\Lambda} = diag[\lambda_i]$ the diagonal matrix of $\mathbf{L}$'s eigenvalues. Then $g(\mathbf{L}) = \mathbf{U}g(\mathbf{\Lambda})\mathbf{U}^T$, and its eigenvalues $\lambda_i^{(g)} = \{g(\lambda_i), 1 \le i \le N\}$. We define $\lambda_G^{\max} = \max_i \{|\lambda_i^{(g)}|\}$, referred to as the *largest absolute eigenvalue*[1] of the graph filter $g(\mathbf{L})$. Let m is the number of training samples depending on N as $m \le N$.

Let $\mathbf{X} \in \mathbb{R}^{N\times D}$ be a node feature matrix (D is the input dimension) and $\theta \in \mathbb{R}^D$ be the learning parameters. With a slight abuse of notation, we will represent both a node (index) in a graph G and its feature values by $\mathbf{x} \in \mathbb{R}^D$. $\mathcal{N}(\mathbf{x})$ denotes a set of the neighbor indices at most 1−hop distance away from node $\mathbf{x}$ (including $\mathbf{x}$). Here the 1−hop distance neighbors are determined using the $g(\mathbf{L})$ filter matrix. Finally, $G_{\mathbf{x}}$ represents the ego-graph extracted at node $\mathbf{x}$ from G.

Single Layer GCNN (Full Graph View): Output function of a single layer GCNN model – on all graph nodes together – can be written in a compact matrix form as follows,

$$f(\mathbf{X}, \theta) = \sigma\Big(g(\mathbf{L})\mathbf{X}\theta\Big) \tag{1}$$

where $g(\mathbf{L})$ is a graph filter. Some commonly used graph filters are a linear function of $\mathbf{A}$ as $g(\mathbf{L}) = \mathbf{A} + \mathbf{I}$ [46] (here $\mathbf{I}$ is the identity matrix) or a Chebyshev polynomial of $\mathbf{L}$ [12].

[1]This definition is valid for a symmetric graph filter $g(\mathbf{L})$, or the matrix is normal. More generally, $\lambda_G^{\max}$ is defined as the largest singular value of $g(\mathbf{L})$.

Single Layer GCNN (Ego-Graph View): We will work with the notion of ego-graph for each node (extracted from G) as it contains the *complete* information needed for computing the output of a single layer GCNN model. We can re-write the Equation (1) for a single node prediction as,

$$f(\mathbf{x}, \theta) = \sigma\Big(\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j^T\theta\Big) \tag{2}$$

where $e_{\cdot j} \in \mathbb{R} = [g(\mathbf{L})]_{\cdot j}$ is the weighted edge (value) between node $\mathbf{x}$ and its neighbor $\mathbf{x}_j$, $j \in \mathcal{N}(\mathbf{x})$ if and only $e_{\cdot j} \neq 0$. The size of an ego-graph depends upon $g(\mathbf{L})$. We assume that the filters are localized to the 1−hop neighbors, but our analysis is applicable to k−hop neighbors. For further notational clarity, we will consider the case $D = 1$, and thus $f(\mathbf{x}, \theta_S) = \sigma\Big(\sum_{j\in\mathcal{N}(\mathbf{x})} e_{\cdot j}\mathbf{x}_j\theta_S\Big)$. Our analysis holds for the general D−dimensional case.

3.2 Main Result

The main result of the paper is stated in Theorem 1, which provides a bound on the generalization gap for single layer GCNN models. This gap is defined as the difference between the generalization error $R(\cdot)$ and empirical error $R_{emp}(\cdot)$ (see definitions in Section 3.3).

Theorem 1. *[**GCNN Generalization Gap**] Let A_S be a single layer GCNN model equipped with the graph convolution filter $g(L)$, and trained on a dataset S using the SGD algorithm for T iterations. Let the loss & activation functions be Lipschitz-continuous and smooth. Then the following expected generalization gap holds with probability at least $1 - \delta$, with $\delta \in (0, 1)$,*

$$\mathbf{E}_{SGD}[R(A_S) - R_{emp}(A_S)] \le \frac{1}{m}\mathcal{O}\big((\lambda_G^{\max})^{2T}\big) + \Big(\mathcal{O}\big((\lambda_G^{\max})^{2T}\big) + M\Big)\sqrt{\frac{\log\frac{1}{\delta}}{2m}}$$

where the expectation $\mathbf{E}_{SGD}$ is taken over the randomness inherent in SGD, m is the number of training samples and M a constant depending on the loss function.

Remarks: Theorem 1 establishes a key connection between the generalization gap and the graph filter eigenvalues. A GCNN model is uniformly stable if the bound converges to zero as $m \to \infty$. In particular, we see that if $\lambda_G^{\max}$ is independent of the graph size, the generalization gap decays at the rate of $\mathcal{O}(\frac{1}{\sqrt{m}})$, yielding the tightest bound possible. Theorem 1 sheds light on the design of stable graph filters with generalization guarantees.

Proof Strategy: We need to tackle several technical challenges in order to obtain the generalization bound in Theorem 1.

(1) **Analyzing GCNN Stability w.r.t. Graph Convolution**: We analyze the stability of a graph convolution function under the single data perturbation. For this purpose, we separately bound the difference on weight parameters from the graph convolution operation in the GCNN output function.
(2) **Analyzing GCNN Stability w.r.t. SGD algorithm**: GCNNs employ the randomized stochastic gradient descent algorithm (SGD) for optimizing the weight parameters. Thus, we need to bound the difference in the expected value over

the learned weight parameters under single data perturbation and establish stability bounds. For this, we analyze the uniform stability of SGD in the context of GCNNs. We adopt the same strategy as in [22] to obtain uniform stability of GCNN models, but with fewer assumptions compared with the general case [22].

3.3 Preliminaries

Basic Setup: Let $\mathcal{X}$ and $\mathcal{Y}$ be a a subset of a Hilbert space and define $Z = \mathcal{X} \times \mathcal{Y}$. We define $\mathcal{X}$ as the input space and $\mathcal{Y}$ as the output space. Let $\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y} \subset R$ and S be a training set $S = \{\mathbf{z}_1 = (\mathbf{x}_1, y_1), \mathbf{z}_2 = (\mathbf{x}_2, y_2), ..., \mathbf{z}_m = (\mathbf{x}_m, y_m)\}$. We introduce two more notations below:

Removing i^{th} data point in the set S is represented as,

$$S^{\setminus i} = \{\mathbf{z}_1, ..., \mathbf{z}_{i-1}, \mathbf{z}_{i+1}, ..., \mathbf{z}_m\}$$

Replacing the i^{th} data point in S by $\mathbf{z}_i^{'}$ is represented as,

$$S^i = \{\mathbf{z}_1, ..., \mathbf{z}_{i-1}, \mathbf{z}_i^{'}, \mathbf{z}_{i+1}, ..., \mathbf{z}_m\}$$

General Data Sampling Process: Let $\mathcal{D}$ denote an unknown distribution from which $\{\mathbf{z}_1, ..., \mathbf{z}_m\}$ data points are sampled to form a training set S. Throughout the paper, we assume all samples (including the replacement sample) are i.i.d. unless mentioned otherwise. Let $\mathbf{E}_S[f]$ denote the expectation of the function f when m samples are drawn from $\mathcal{D}$ to form the training set S. Likewise, let $\mathbf{E}_z[f]$ denote the expectation of the function f when $\mathbf{z}$ is sampled according to $\mathcal{D}$.

Graph Node Sampling Process: At first it may not be clear on how to describe the sampling procedure of nodes from a graph G in the context of GCNNs for performing semi-supervised learning. For our purpose, we consider *ego-graphs* formed by the 1−hops neighbors at each node as a single data point. This ego-graph is *necessary and sufficient* to compute the single layer GCNN output as shown in Equation (2). We assume node data points are sampled in an i.i.d. fashion by first choosing a node $\mathbf{x}$ and then extracting its neighbors from G to form an ego-graph.

Generalization Error: Let A_S be a learning algorithm trained on dataset S. A_S is defined as a function from Z^m to $(\mathcal{Y})^X$. For GCNNs, we set $A_S = f(\mathbf{x}, \theta_S)$. Then generalization error or risk $R(A_S)$ with respect to a loss function $\ell : \mathbf{Z}^m \times \mathbf{Z} \to \mathbb{R}$ is defined as,

$$R(A_S) := \mathbf{E}_z[\ell(A_S, \mathbf{z})] = \int \ell(A_S, \mathbf{z})p(\mathbf{z})d\mathbf{z}.$$

Empirical Error: Empirical risk $R_{emp}(A_S)$ is defined as,

$$R_{emp}(A_S) := \frac{1}{m} \sum_{j=1}^{m} \ell(A_S, \mathbf{z}_j).$$

Generalization Gap: When A_S is a randomized algorithm, we consider the expected generalization gap as shown below,

$$\epsilon_{\text{gen}} := \mathbf{E}_A[R(A_S) - R_{emp}(A_S)].$$

Here the expectation $\mathbf{E}_A$ is taken over the inherent randomness of A_S. For instance, most learning algorithms employ Stochastic Gradient descent (SGD) to learn the weight parameters. SGD introduces randomness due to the random order it uses to choose samples for batch processing. In our analysis, we only consider randomness in A_S due to SGD and ignore the randomness introduced by parameter initialization. Hence, we will replace $\mathbf{E}_A$ with $\mathbf{E}_{\text{SGD}}$.

Uniform Stability of Randomized Algorithm: For a randomized algorithm, uniform stability is defined as follows,

Definition 1. *[**Uniform Stability**] A randomized learning algorithm A_S is β_m−uniformly stable with respect to a loss function ℓ, if it satisfies,*

$$\sup_{S,z} |\mathbf{E}_A[\ell(A_S, \mathbf{z})] - \mathbf{E}_A[\ell(A_{S^{\setminus i}}, \mathbf{z})]| \le \beta_m$$

For our convenience, we will work with the following definition of uniform stability,

$$\sup_{S,z} |\mathbf{E}_A[\ell(A_S, \mathbf{z})] - \mathbf{E}_A[\ell(A_{S^i}, \mathbf{z})]| \le 2\beta_m$$

which follows immediately from the fact that,

$$\sup_{S,z} |\mathbf{E}_A[\ell(A_S, \mathbf{z})] - \mathbf{E}_A[\ell(A_{S^i}, \mathbf{z})]| \le \Big(\sup_{S,z} |\mathbf{E}_A[\ell(A_S, \mathbf{z})] - \mathbf{E}_A[\ell(A_{S^{\setminus i}}, \mathbf{z})]|\Big) + \Big(\sup_{S,z} |\mathbf{E}_A[\ell(A_{S^i}, \mathbf{z})] - \mathbf{E}_A[\ell(A_{S^{\setminus i}}, \mathbf{z})]|\Big)$$

Remarks: Uniform stability imposes an upper bound on the difference in losses due to a removal (or change) of a single data point from the set (of size m) for all possible combinations of S, z. Here, β_m is a function of m (the number of training samples). Note that there is a subtle difference between Definition 1 above and the uniform stability of randomized algorithms defined in [17] (see Definition 13 in [17]). The authors in [17] are concerned with random elements associated with the cost function such as those induced by bootstrapping, bagging or initialization process. However, we focus on the randomness due to the learning procedure, i.e., SGD.

Stability Guarantees: A randomized learning algorithm with uniform stability yields the following bound on generalization gap:

Theorem 2. *[**Stability Guarantees**] A uniform stable randomized algorithm (A_S, β_m) with a bounded loss function $0 \le \ell(A_S, \mathbf{z}) \le M$, satisfies following generalization bound with probability at-least $1 - \delta$, over the random draw of S,$\mathbf{z}$ with $\delta \in (0, 1)$,*

$$\mathbf{E}_A[R(A_S) - R_{emp}(A_S)] \le 2\beta_m + \big(4m\beta_m + M\big)\sqrt{\frac{\log \frac{1}{\delta}}{2m}}.$$

Proof: The proof for Theorem 2 mirrors that of Theorem 12 (shown in [7] for *deterministic* learning algorithms). For the sake of completeness, we include the proof in Appendix based on our definition of uniform stability $:= \sup_{S,z} |\mathbf{E}_A[\ell(A_S, \mathbf{z})] - \mathbf{E}_A[\ell(A_{S^i}, \mathbf{z})]| \le 2\beta_m$.

Remarks: The generalization bound is meaningful if the bound converges to 0 as $m \to \infty$. This occurs when β_m decays faster than $\mathcal{O}(\frac{1}{\sqrt{m}})$; otherwise the generalization gap does not approach to zero as $m \to \infty$. Furthermore, generalization gap produces tightest bounds when β_m decays at $\mathcal{O}(\frac{1}{m})$ which is the most stable state possible for a learning algorithm.

$\sigma-$Lipschitz Continuous and Smooth Activation Function: Our bounds hold for all activation functions which are Lipschitz-continuous and smooth. An activation function $\sigma(x)$ is Lipschitz-continuous if $|\nabla\sigma(x)| \le \alpha_\sigma$, or equivalently, $|\sigma(x)-\sigma(y)| \le \alpha_\sigma|x-y|$. We further require $\sigma(x)$ to be smooth, namely, $|\nabla\sigma(x)-\nabla\sigma(y)| \le \nu_\sigma|x-y|$. This assumption is more strict but necessary for establishing the strong notion of uniform stability. Some common activation functions satisfying the above conditions are ELU (with $\alpha = 1$), Sigmoid, and Tanh.

$\ell-$Lipschitz Continuous and Smooth Loss Function: We also assume that the loss function is Lipschitz-continuous and smooth,

$$\left|\ell(f(\cdot),y) - \ell(f^{'}(\cdot),y)\right| \le \alpha_\ell\left|f(\cdot) - f^{'}(\cdot))\right|,$$
$$\text{and } \left|\nabla\ell(f(\cdot),y) - \nabla\ell(f^{'}(\cdot),y)\right| \le \nu_\ell\left|\nabla f(\cdot) - \nabla f^{'}(\cdot)\right|.$$

Unlike in [22], we define Lipschitz-continuity with respect to the function argument rather than the weight parameters, a relatively weak assumption.

3.4 Uniform Stability of GCNN Models

The crux of our main result relies on showing that GCNN models are uniformly stable as stated in Theorem 3 below.

Theorem 3. *[**GCNN Uniform Stability**] Let the loss & activation be Lipschitz-continuous and smooth functions. Then a single layer GCNN model trained using the SGD algorithm for T iterations is β_m-uniformly stable, where*

$$\beta_m \le \left(\eta\alpha_\ell\alpha_\sigma\nu_\ell(\lambda_G^{\max})^2\sum_{t=1}^{T}\left(1+\eta\nu_\ell\nu_\sigma(\lambda_G^{\max})^2\right)^{t-1}\right)/m.$$

Remarks: Plugging the bound on β_m in Theorem 2 yields the main result of our paper.

Before we proceed to prove this theorem, we first explain what is meant by training a single layer GCNN using SGD on datasets S and S^i which differ in one data point, following the same line of reasoning as in [22]. Let $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_t, \ldots, \mathbf{z}_T\}$ be a sequence of samples, where $\mathbf{z}_t$ is an i.i.d. sample drawn from S at the t^{th} iteration of SGD during a training run of the GCCN[2]. Training the same GCCN using SGD on S^i means that we supply the same sample sequence to the GCCN except that if $\mathbf{z}_t = (\mathbf{x}_i, y_i)$ for some t ($1 \le t \le T$), we replace it with $\mathbf{z}_t^{'} = (\mathbf{x}_i', y_i')$, where i is the (node) index at which S and S^i differ. We denote this sample sequence by Z'. Let $\{\theta_{S,0}, \theta_{S,1}, \ldots, \theta_{S,T}\}$ and $\{\theta_{S^i,0}, \theta_{S^i,1}, \ldots, \theta_{S^i,T}\}$ denote the corresponding sequences of the weight parameters learned by running SGD on S and S^i, respectively. Since the parameter initialization is kept same, $\theta_{S,0} = \theta_{S^i,0}$. In addition, if k is the first time that the sample sequences Z and Z' differ, then $\theta_{S,t} = \theta_{S^i,t}$ at each step t before k, and at the k^{th} and subsequent steps, $\theta_{S,t}$ and $\theta_{S^i,t}$ diverge. The key in establishing the uniform stability of a GCNN model is to bound the difference in losses when training the GCNN using SGD on S *vs.* S^i. As stated earlier in the proof strategy, we proceed in two steps.

[2] One way to generate the sample sequence is to choose a node index i_t uniformly at random from the set $\{1, \ldots, m\}$ at each step t. Alternatively, one can first choose a random permutation of $\{1, \ldots, m\}$ and then process the samples accordingly. Our analysis holds for both cases.

Proof Part I (Single Layer GCNN Bound): We first bound the expected loss by separating the factors due to the graph convolution operation *vs.* the expected difference in the filter weight parameters learned via SGD on two datasets S and S^i.

Let θ_S and θ_{S^i} represent the final GCNN filter weights learned on training set S and S^i respectively. Define $\Delta\theta = \theta_S - \theta_{S^i}$. Using the facts that the loss are Lipschitz continuous and also $|\mathbf{E}[x]| \le \mathbf{E}[|x|]$, we have,

$$|\mathbf{E}_{\text{SGD}}[\ell(A_S,y) - \ell(A_{S^i},y)]| \le \alpha_\ell\mathbf{E}_{\text{SGD}}[|f(\mathbf{x},\theta_S) - f(\mathbf{x},\theta_{S^i})|]$$
$$\le \alpha_\ell\mathbf{E}_{\text{SGD}}\left[\left|\sigma\Big(\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\theta_S\Big) - \sigma\Big(\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\theta_{S^i}\Big)\right|\right]$$

Since activation function is also $\sigma-$Lipschitz continuous,

$$\begin{aligned}
&\le \alpha_\ell\mathbf{E}_{\text{SGD}}\left[\Big|\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\theta_S - \sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\theta_{S^i}\Big|\right]\\
&\le \alpha_\ell\mathbf{E}_{\text{SGD}}\left[\Big|\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j(\theta_S - \theta_{S^i})\Big|\right]\\
&\le \alpha_\ell\mathbf{E}_{\text{SGD}}\left[\Big|\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} (e_{\cdot j}\mathbf{x}_j)\Big|\left(\left|\theta_S - \theta_{S^i}\right|\right)\right]\\
&\le \alpha_\ell\Big|\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} (e_{\cdot j}\mathbf{x}_j)\Big|\left(\mathbf{E}_{\text{SGD}}\left[|\Delta\theta|\right]\right)\\
&\le \alpha_\ell\mathbf{g}_\lambda\mathbf{E}_{\text{SGD}}\left[|\Delta\theta|\right]
\end{aligned} \tag{3}$$

where $\mathbf{g}_\lambda$ is defined as $\mathbf{g}_\lambda := \sup_{\mathbf{x}} \left|\sum_{j\in\mathcal{N}(\mathbf{x})} e_{\cdot j}\mathbf{x}_j\right|$. We will bound $\mathbf{g}_\lambda$ in terms of the largest absolute eigenvalue of the graph convolution filter $g(\mathbf{L})$ later. Note that $\sum_{j\in\mathcal{N}(\mathbf{x})} e_{\cdot j}\mathbf{x}_j$ is nothing but a graph convolution operation. As such, reducing $\mathbf{g}_\lambda$ will be the contributing factor in improving the generalization performance.

Proof Part II (SGD Based Bounds For GCNN Weights): What remains is to bound $\mathbf{E}_{\text{SGD}}[|\Delta\theta|]$ due to the randomness inherent in SGD. This is proved through a series of three lemmas. We first note that on a given training set S, a GCNN minimizes the following objective function,

$$\min_{\theta} \quad \mathcal{L}(f(\mathbf{x},\theta_S),y) = \frac{1}{m}\sum_{i=1}^{m}\ell(f(\mathbf{x},\theta_S),y_i) \tag{4}$$

For this, at each iteration t, SGD performs the following update:

$$\theta_{S,t+1} = \theta_{S,t} - \eta\nabla\ell(f(\mathbf{x}_{i_t},\theta_{S,t}),y_{i_t}) \tag{5}$$

where $\eta > 0$ is the learning rate.

Given two sequences of the weight parameters, $\{\theta_{S,0}, \theta_{S,1}, \ldots, \theta_{S,T}\}$ and $\{\theta_{S^i,0}, \theta_{S^i,1}, \ldots, \theta_{S^i,T}\}$, learned by the GCCN running SGD on S and S^i, respectively, we first find a bound on $\Delta\theta_t := |\theta_{S,t} - \theta_{S^i,t}|$ at each iteration step t of SGD.

There are two scenarios to consider 1) At step t, SGD picks a sample $\mathbf{z}_t = (\mathbf{x}, y)$ which is identical in Z and Z', and occurs with probability $(m-1)/m$. From Equation (5), we have $|\Delta\theta_{t+1}| \le |\Delta\theta_t| + \eta|\nabla\ell(f(\mathbf{x},\theta_{S,t}),y) - \ell(f(\mathbf{x},\theta_{S,t}),y)|$. We bound this term in Lemma 1 below 2) At step t, SGD picks the only samples that

Z and Z' differ, $\mathbf{z}_t = (\mathbf{x}_i, y_i)$ and $\mathbf{z}'_t = (\mathbf{x}'_i, y'_i)$ which occurs with probability $1/m$. Then $|\Delta\theta_{t+1}| \leq |\Delta\theta_t| + \eta|\nabla\ell(f(\mathbf{x}_i, \theta_{S,t}), y_i) - \ell(f(\mathbf{x}'_i, \theta_{S,t}), y'_i)|$. We bound the second term in Lemma 2 below.

Lemma 1. ***[GCNN Same Sample Loss Stability Bound]*** *The loss-derivative bound difference of (single-layer) GCNN models trained with SGD algorithm for T iterations on two training datasets S and S^i respectively, with respect to the same sample is given by,*

$$\left|\nabla\ell(f(\mathbf{x}, \theta_{S,t}), y) - \nabla\ell(f(\mathbf{x}, \theta_{S^i,t}), y)\right| \leq \nu_\ell \nu_\sigma \mathbf{g}_\lambda^2 |\Delta\theta_t|.$$

Proof: The first order derivative of a single-layer the GCNN output function, $f(\mathbf{x}, \theta) = \sigma(\sum_{j\in\mathcal{N}} e_{\cdot j}\mathbf{x}_j\theta)$, is given by,

$$\frac{\partial f(\mathbf{x}, \theta)}{\partial\theta} = \sigma'\Big(\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\theta\Big)\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j, \tag{6}$$

where $\nabla\sigma(\cdot)$ is the first order derivative of the activation function.

Using Equation (6) and the fact that the loss function is Lipschitz continuous and smooth, we have,

$$\begin{aligned}
&\left|\nabla\ell(f(\mathbf{x}, \theta_{S,t}), y) - \nabla\ell(f(\mathbf{x}, \theta_{S^i,t}), y)\right| \leq \\
&\qquad\qquad \nu_\ell\left|\nabla f(\mathbf{x}, \theta_{S,t}) - \nabla f(\mathbf{x}, \theta_{S^i,t})\right| \\
&\leq \nu_\ell\Big|\nabla\sigma\Big(\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\theta_{S,t}\Big)\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j - \\
&\qquad\qquad \nabla\sigma\Big(\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\theta_{S^i,t}\Big)\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\Big| \\
&\leq \nu_\ell\Big(\Big|\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\Big|\Big)\Big|\nabla\sigma\Big(\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\theta_{S,t}\Big) - \nabla\sigma\Big(\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\theta_{S^i,t}\Big)\Big|
\end{aligned}$$

Since the activation function is Lipschitz continuous and smooth, and plugging $\Big|\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\Big| \leq \mathbf{g}_\lambda$, we get,

$$\begin{aligned}
&\leq \nu_\ell\nu_\sigma\mathbf{g}_\lambda\Big|\Big(\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\theta_{S,t}\Big) - \Big(\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\theta_{S^i,t}\Big)\Big| \\
&\leq \nu_\ell\nu_\sigma\mathbf{g}_\lambda\Big(\Big|\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\Big|\Big)|\theta_{S,t} - \theta_{S^i,t}| \\
&\leq \nu_\ell\nu_\sigma\mathbf{g}_\lambda^2|\Delta\theta_t|
\end{aligned}$$

This completes the proof of Lemma 1.

Note: Without the $\sigma-$smooth assumption, it would not be possible to derive the above bound in terms of $|\Delta\theta_t|$ which is necessary for showing the uniform stability. Unfortunately, this constraint excludes RELU activation from our analysis.

Lemma 2. ***[GCNN Different Sample Loss Stability Bound]*** *The loss-derivative bound difference of (single-layer) GCNN models trained with SGD algorithm for T iterations on two training datasets S and S^i respectively, with respect to the different samples is given by,*

$$\left|\nabla\ell(f(\mathbf{x}_i, \theta_{S,t}), y_i) - \nabla\ell(f(\mathbf{x}'_i, \theta_{S^i,t}), y'_i)\right| \leq 2\nu_\ell\alpha_\sigma\mathbf{g}_\lambda.$$

Proof: Again using Equation (6) and the fact that the loss & activation function is Lipschitz continuous and smooth, and for any a, b, $|a-b| \leq |a| + |b|$, we have,

$$\begin{aligned}
&\left|\nabla\ell(f(\mathbf{x}, \theta_{S,t}), y) - \nabla\ell(f(\mathbf{x}', \theta_{S^i,t}), y')\right| \leq \\
&\qquad\qquad \nu_\ell\left|\nabla f(\mathbf{x}, \theta_{S,t}) - \nabla f(\mathbf{x}', \theta_{S^i,t})\right| \\
&\leq \nu_\ell\Big|\nabla\sigma\Big(\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\theta_{S,t}\Big)\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j - \nabla\sigma\Big(\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x}')}} e_{\cdot j}\mathbf{x}'_j\theta_{S^i,t}\Big)\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x}')}} e_{\cdot j}\mathbf{x}'_j\Big| \\
&\leq \nu_\ell\Big|\nabla\sigma\Big(\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\theta_{S,t}\Big)\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x})}} e_{\cdot j}\mathbf{x}_j\Big| + \\
&\quad \nu_\ell\Big|\nabla\sigma\Big(\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x}')}} e_{\cdot j}\mathbf{x}'_j\theta_{S^i,t}\Big)\sum_{\substack{j\in\\ \mathcal{N}(\mathbf{x}')}} e_{\cdot j}\mathbf{x}'_j\Big|
\end{aligned} \tag{7}$$

Using the fact that the first order derivative is bounded,

$\leq 2\nu_\ell\alpha_\sigma\mathbf{g}_\lambda$

This completes the proof of Lemma 2.

Summing over all iteration steps, and taking expectations over all possible sample sequences Z, Z' from S and S^i, we have

Lemma 3. ***[GCNN SGD Stability Bound]*** *Let the loss & activation functions be Lipschitz-continuous and smooth. Let $\theta_{S,T}$ and $\theta_{S^i,T}$ denote the graph filter parameters of (single-layer) GCNN models trained using SGD for T iterations on two training datasets S and S^i, respectively. Then the expected difference in the filter parameters is bounded by,*

$$\mathbf{E}_{SGD}\left[|\Delta\theta_{S,T} - \theta_{S^i,T}|\right] \leq \frac{2\eta\nu_\ell\alpha_\sigma\mathbf{g}_\lambda}{m}\sum_{t=1}^{T}\left(1 + \eta\nu_\ell\nu_\sigma\mathbf{g}_\lambda^2\right)^{t-1}$$

Proof: From Equation (5) and taking into account the probabilities of the two scenarios considered in Lemma 1 and Lemma 2 at step t, we have,

$$\begin{aligned}
&\mathbf{E}_{\text{SGD}}\left[|\Delta\theta_{t+1}|\right] \leq \Big(1 - \frac{1}{m}\Big)\mathbf{E}_{\text{SGD}}\Big[\Big|\Big(\theta_{S,t} - \eta\nabla\ell(f(\mathbf{x}, \theta_{S,t}), y)\Big) - \\
&\quad \Big(\theta_{S^i,t} - \eta\nabla\ell(f(\mathbf{x}, \theta_{S^i,t}), y)\Big)\Big|\Big] + \Big(\frac{1}{m}\Big)\mathbf{E}_{\text{SGD}}\Big[\Big|\Big(\theta_{S,t} - \\
&\quad \eta\nabla\ell(f(\mathbf{x}', \theta_{S,t}), y')\Big) - \Big(\theta_{S^i,t} - \eta\nabla\ell(f(\mathbf{x}'', \theta_{S^i,t}), y'')\Big)\Big|\Big] \\
&\leq \Big(1 - \frac{1}{m}\Big)\mathbf{E}_{\text{SGD}}\left[|\Delta\theta_t|\right] + \Big(1 - \frac{1}{m}\Big)\eta\mathbf{E}_{\text{SGD}}\Big[\Big|\nabla\ell(f(\mathbf{x}, \theta_{S,t}), y) - \\
&\quad \nabla\ell(f(\mathbf{x}, \theta_{S^i,t}), y)\Big|\Big] + \Big(\frac{1}{m}\Big)\mathbf{E}_{\text{SGD}}\left[|\Delta\theta_t|\right] + \\
&\quad \Big(\frac{1}{m}\Big)\eta\mathbf{E}_{\text{SGD}}\Big[\Big|\nabla\ell(f(\mathbf{x}', \theta_{S,t}), y') - \nabla\ell(f(\mathbf{x}'', \theta_{S^i,t}), y'')\Big|\Big] \\
&= \mathbf{E}_{\text{SGD}}\left[|\Delta\theta_t|\right] + \\
&\quad \Big(1 - \frac{1}{m}\Big)\eta\mathbf{E}_{\text{SGD}}\Big[\Big|\nabla\ell(f(\mathbf{x}, \theta_{S,t}), y) - \nabla\ell(f(\mathbf{x}, \theta_{S^i,t}), y)\Big|\Big] + \\
&\quad \Big(\frac{1}{m}\Big)\eta\mathbf{E}_{\text{SGD}}\Big[\Big|\Big(\nabla\ell(f(\mathbf{x}', \theta_{S,t}), y')\Big) - \Big(\nabla\ell(f(\mathbf{x}'', \theta_{S^i,t}), y'')\Big)\Big|\Big].
\end{aligned} \tag{8}$$

Plugging the bounds in Lemma 1 and Lemma 2 into Equation (8), we have,

$$\begin{aligned}\mathbf{E}_{\text{SGD}}\left[|\Delta\theta_{t+1}|\right] &\leq \mathbf{E}_{\text{SGD}}\left[|\Delta\theta_t|\right] + \left(1-\frac{1}{m}\right)\eta\nu_\ell\nu_\sigma \mathbf{g}_\lambda^2 \mathbf{E}_{\text{SGD}}[|\theta_t|] \\ &\quad + \left(\frac{1}{m}\right)2\eta\nu_\ell\alpha_\sigma\mathbf{g}_\lambda \\ &= \left(1 + (1-\frac{1}{m})\eta\nu_\ell\nu_\sigma\mathbf{g}_\lambda^2\right)\mathbf{E}_{\text{SGD}}[|\theta_t|] + \frac{2\eta\nu_\ell\alpha_\sigma\mathbf{g}_\lambda}{m} \\ &\leq \left(1 + \eta\nu_\ell\nu_\sigma\mathbf{g}_\lambda^2\right)\mathbf{E}_{\text{SGD}}[|\theta_t|] + \frac{2\eta\nu_\ell\alpha_\sigma\mathbf{g}_\lambda}{m}.\end{aligned}$$

Lastly, solving the $\mathbf{E}_{\text{SGD}}\left[|\Delta\theta_t|\right]$ first order recursion yields,

$$\mathbf{E}_{\text{SGD}}\left[|\Delta\theta_T|\right] \leq \frac{2\eta\nu_\ell\alpha_\sigma\mathbf{g}_\lambda}{m}\sum_{t=1}^{T}\left(1+\eta\nu_\ell\nu_\sigma\mathbf{g}_\lambda^2\right)^{t-1}$$

This completes the proof of Lemma 3.

Bound on $\mathbf{g}_\lambda$: We now bound $\mathbf{g}_\lambda$ in terms of the largest absolute eigenvalue of the graph filter matrix $g(\mathbf{L})$. We first note that at each node $\mathbf{x}$, the ego-graph $G_\mathbf{x}$ ego-graph can be represented as a submatrix of $g(\mathbf{L})$. Let $g_\mathbf{x}(\mathbf{L}) \in \mathbb{R}^{q\times q}$ be the submatrix of $g(\mathbf{L})$ whose row and column indices are from the set $\{j \in \mathcal{N}(\mathbf{x})\}$. The ego-graph size is $q = |\mathcal{N}(\mathbf{x})|$. We use $\mathbf{h}_\mathbf{x} \in \mathbb{R}^q$ to denote the graph signals (node features) on the ego-graph $G_\mathbf{x}$. Without loss of generality, we will assume that node $\mathbf{x}$ is represented by index 0 in $G_\mathbf{x}$. Thus, we can compute $\sum_{j\in\mathcal{N}(\mathbf{x})} e_{\cdot j}\mathbf{x}_j = [g_\mathbf{x}(\mathbf{L})\mathbf{h}_\mathbf{x}]_0$, a scalar value. Here $[\cdot]_0 \in \mathbb{R}$ represents the value of a vector at index 0, i.e., corresponding to node $\mathbf{x}$. Then the following holds (assuming the graph signals are normalized, i.e., $\|\mathbf{h}_\mathbf{x}\|_2 = 1$),

$$|[g_\mathbf{x}(\mathbf{L})\mathbf{h}_\mathbf{x}]_0| \leq \|g_\mathbf{x}(\mathbf{L})\mathbf{h}_\mathbf{x}\|_1 \leq \|\|g_\mathbf{x}(\mathbf{L})\|_2\|\mathbf{h}_\mathbf{x}\|_2 = \lambda_{G_\mathbf{x}}^{\max} \tag{9}$$

where the second inequality follows from Cauchy–Schwarz Inequality, and $\|M\|_2 = \sup_{\|x\|_2=1}\|Mx\|_2 = \sigma_{max}(M)$ is the matrix operator norm and $\sigma_{max}(M)$ is the largest singular value of matrix M. For a normal matrix M (such as a symmetric graph filter $g(L)$), $\sigma_{max}(M) = \max|\lambda(M)|$, the largest absolute eigenvalue of M.

Lemma 4. *[Ego-Graph Eigenvalue Bound] Let $G = (V, E)$ be a (un)directed graph with (either symmetric or non-negative) weighted adjacency matrix $g(\mathbf{L})$ and $\lambda_G^{\max}$ be the maximum absolute eigenvalue of $g(\mathbf{L})$. Let $G_\mathbf{x}$ be the ego-graph of a node $\mathbf{x} \in V$ with corresponding maximum absolute eigenvalue $\lambda_{G_\mathbf{x}}^{\max}$. Then the following eigenvalue (singular value) bound holds $\forall\mathbf{x}$,*

$$\lambda_{G_\mathbf{x}}^{\max} \leq \lambda_G^{\max}$$

Proof: Notice that $g_\mathbf{x}(\mathbf{L})$ is the adjacency matrix of $G_\mathbf{x}$ which also happens to be the principal submatrix of $g(\mathbf{L})$. As a result, above bound holds from the eigenvalue interlacing theorem for normal/Hermitian matrices and their principal submatrices [20, 28].

Finally, plugging $\mathbf{g}_\lambda \leq \lambda_G^{\max}$ and Lemma 3 into Equation (3) yields the following remaining result,

$$\begin{aligned}2\beta_m &\leq \alpha_\ell\lambda_G^{\max}\mathbf{E}_{\text{SGD}}\left[|\Delta\theta|\right] \\ \beta_m &\leq \frac{\eta\alpha_\ell\alpha_\sigma\nu_\ell(\lambda_G^{\max})^2\sum_{t=1}^{T}\left(1+\eta\nu_\ell\nu_\sigma(\lambda_G^{\max})^2\right)^{t-1}}{m} \\ \beta_m &\leq \frac{1}{m}\mathcal{O}\left((\lambda_G^{\max})^{2T}\right) \qquad \forall T \geq 1\end{aligned}$$

This completes the full proof of Theorem 3.

4 REVISITING GRAPH CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE

In this section, we discuss the implication of our results in designing graph convolution filters and revisit the importance of employing batch-normalization layers in GCNN network.

Unnormalized Graph Filters: One of the most popular graph convolution filters is $g(\mathbf{L}) = \mathbf{A} + \mathbf{I}$ [46]. The eigen spectrum of the unnormalized $\mathbf{A}$ is bounded by $\mathcal{O}(N)$. This is concerning as now $\mathbf{g}_\lambda$ is bounded by $\mathcal{O}(N)$ and as m becomes close to N, β_m tend towards $\mathcal{O}(N^c)$ complexity with $c \geq 0$. As a result, the generalization gap of such a GCNN model is not guaranteed to converge.

Normalized Graph Filters: Numerical instabilities with the unnormalized adjacency matrix have already been suspected in [25]. Therefore, the symmetric normalized graph filter has been adopted: $g(\mathbf{L}) = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} + \mathbf{I}$. The eigen spectrum of $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ is bounded between $[-1, 1]$. As a result, such a GCNN model is uniformly stable (assuming that the graph features are also normalized appropriately, e.g., $\|\mathbf{x}\|_2 = 1$).

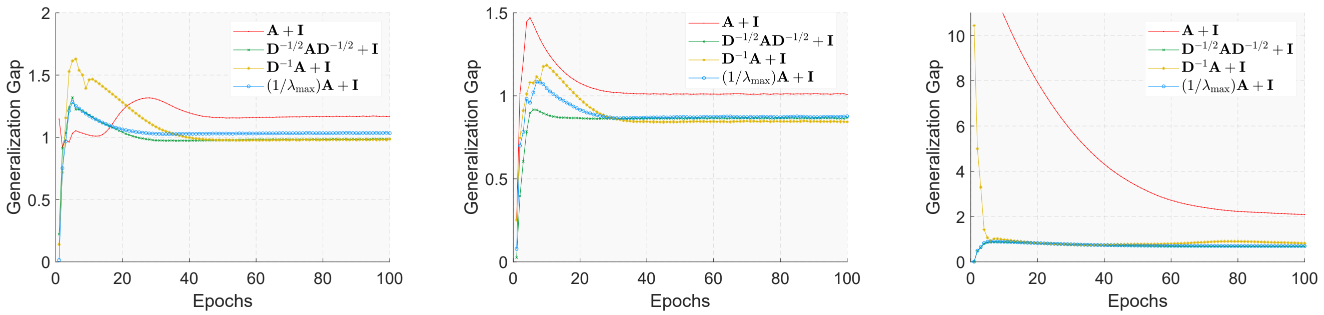
Random Walk Graph Filters: Another graph filter that has been widely used is based on random walks: $g(\mathbf{L}) = \mathbf{D}^{-1}\mathbf{A} + \mathbf{I}$ [35]. The eigenvalues of $\mathbf{D}^{-1}\mathbf{A}$ are spread out in the interval $[0, 2]$ and thus such a GCNN model is uniformly stable.

Importance of Batch-Normalization in GCNN: Recall that $\mathbf{g}_\lambda = \sup_\mathbf{x}\left|\sum_{j\in\mathcal{N}(\mathbf{x})} e_{\cdot j}\mathbf{x}_j\right|$ and notice that in Equation (9), we assume that the graph signals are normalized in order to bound $\mathbf{g}_\lambda$. This can easily be accomplished by normalizing features during data pre-processing phase for a single layer GCNN. However, for a multi-layer GCNN, the intermediate feature outputs are not guaranteed to be normalized. Thus to ensure stability, it is crucial to employ batch-normalization layers in GCNN models. This has already been reported in [46] as an important factor for keeping the GCNN outputs stable.

5 EXPERIMENTAL EVALUATION

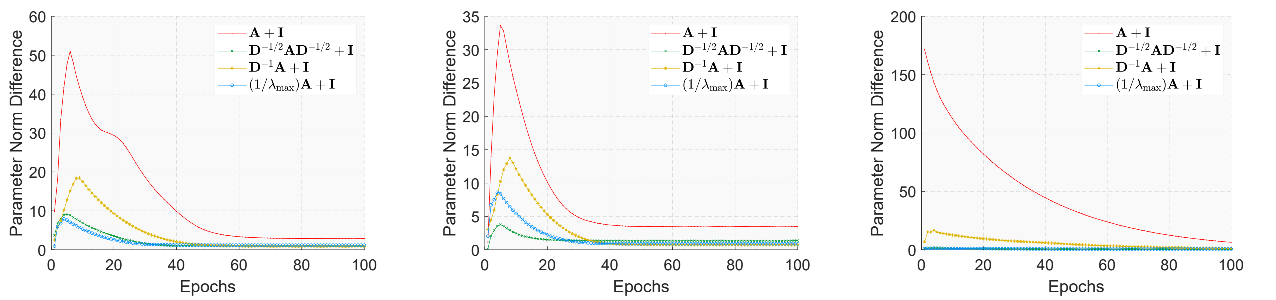
In this section, we empirically evaluate the effect of graph filters on the GCNN stability bounds using four different GCNN filters. We employ three citation network datasets: Citeseer, Cora and Pubmed (see [25] for details about the datasets).

Experimental Setup: We extract 1−hop ego-graphs of each node in a given dataset to create samples and normalize the node graph features such that $\|\mathbf{x}\|_2 = 1$ in the data pre-processing step. We run the SGD algorithm with a fixed learning rate $\eta = 1$ with the batch size equal to 1 for 100 epochs on all datasets. We employ ELU (set $\alpha = 1$) as the activation function and cross-entropy as the loss function.



a. Generlization Gap on Citeseer Dataset **b. Generlization Gap on Cora Dataset** **c. Generlization Gap on Pubmed Dataset**

Figure 1. The above figures show the generalziation gap for three datasets. The generlization gap is measured with respect to the loss function, i.e., |(training error − test error)|. In this experiment, the cross-entropy loss is used.



a. Parameter $L2-$Norm Diff on Citeseer **b. Parameter $L2-$Norm Diff on Cora** **c. Parameter $L2-$Norm Diff on Pubmed**

Figure 2. The above figures show the divergence in weight parameters of a single layer GCNN measured using $L2-$norm on the three datasets. We surgically alter one sample point at index $i = 0$ in the training set S to generate S^i and run the SGD algorithm.

Measuring Generalization Gap: In this experiment, we quantitatively measure the generalization gap defined as the absolute difference between the training and test errors. From Figure 1, it is clear that the unnormalized graph convolution filters such as $g(\mathbf{L}) = \mathbf{A}+\mathbf{I}$ show a significantly higher generalization gap than the normalized ones such as $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ or random walk $g(\mathbf{L}) = \mathbf{D}^{-1}\mathbf{A} + \mathbf{I}$ based graph filters. The results hold consistently across the three datasets. We note that the generalization gap becomes constant after a certain number of iterations. While this phenomenon is not reflected in our bounds, it can plausibly be explained by considering the variable bounding parameters (as a function of SGD iterations). This hints at the pessimistic nature of our bounds.

Measuring GCNN Learned Filter-Parameters Stability Based On SGD Optimizer: In this experiment, we evaluate the difference between learned weight parameters of two single layer GCNN models trained on datasets S and S^i which differ precisely in one sample point. We generate S^i by surgically altering one sample point in S at the node index $i = 0$. For this experiment, we initialize the GCNN models on both datasets with the same parameters and random seeds, and then run the SGD algorithm. After each epoch, we measure the $L2-$norm difference between the weight parameters of the respective models. From Figure 2, it is evident that for the unnormalized graph convolution filters, the weight parameters tend to deviate by a large amount and therefore the network is less stable. While for the normalized graph filters the norm difference converges quickly to a fixed value. These empirical observations are reinforced by our stability bounds. However, the decreasing trend in the norm difference after a certain number of iterations before convergence, remains unexplained, due to the pessimistic nature of our bounds.

6 CONCLUSION AND FUTURE WORK

We have taken the first steps towards establishing a deeper theoretical understanding of GCNN models by analyzing their stability and establishing their generalization guarantees. More specifically, we have shown that the algorithmic stability of GCNN models depends upon the largest absolute eigenvalue of graph convolution filters. To ensure uniform stability and thereby generalization guarantees, the largest absolute eigenvalue must be independent of the graph size. Our results shed new insights on the design of new & improved graph convolution filters with guaranteed algorithmic stability. Furthermore, applying our results to existing GCNN models, we provide a theoretical justification for the importance of employing the batch-normalization process in a GCNN architecture. We have also conducted empirical evaluations based on real world datasets which support our theoretical findings. To the best of our knowledge, we are the first to study stability bounds on graph learning in a semi-supervised setting and derive generalization bounds for GCNN models.

As part of our ongoing and future work, we will extend our analysis to multi-layer GCNN models. For a multi-layer GCNN, we need to bound the difference in weights at each layer according to the back-propagation algorithm. Therefore the main technical challenge is to study the stability of the full fledged back-propagation algorithm. Furthermore, we plan to study the stability and generalization properties of non-localized convolutional filters designed based on rational polynomials of the graph Laplacian. We also plan to generalize our analysis framework beyond semi-supervised learning to provide generalization guarantees in learning settings where multiple graphs are present, e.g., for graph classification.

7 ACKNOWLEDGMENTS

The research was supported in part by US DoD DTRA grant HDTRA1-14-1-0040, and NSF grants CNS 1618339, CNS 1617729, CNS 1814322 and CNS183677.

REFERENCES

[1] Shivani Agarwal and Partha Niyogi. 2005. Stability and generalization of bipartite ranking algorithms. In *International Conference on Computational Learning Theory*. Springer, 32–47.
[2] Shivani Agarwal and Partha Niyogi. 2009. Generalization bounds for ranking algorithms via algorithmic stability. *Journal of Machine Learning Research* 10, Feb (2009), 441–474.
[3] Rie K Ando and Tong Zhang. 2007. Learning on graph with Laplacian regularization. In *Advances in neural information processing systems*. 25–32.
[4] James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*. 1993–2001.
[5] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. 2004. Regularization and semi-supervised learning on large graphs. In *International Conference on Computational Learning Theory*. Springer, 624–638.
[6] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. 1989. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM (JACM)* 36, 4 (1989), 929–965.
[7] Olivier Bousquet and André Elisseeff. 2002. Stability and generalization. *Journal of Machine Learning Research* 2, Mar (2002), 499–526.
[8] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).
[9] Nadav Cohen and Amnon Shashua. 2016. Convolutional rectifier networks as generalized tensor decompositions. In *International Conference on Machine Learning*. 955–963.
[10] Corinna Cortes, Mehryar Mohri, Dmitry Pechyony, and Ashish Rastogi. 2008. Stability of transductive regression algorithms. In *Proceedings of the 25th international conference on Machine learning*. ACM, 176–183.
[11] Hanjun Dai, Bo Dai, and Le Song. 2016. Discriminative embeddings of latent variable models for structured data. In *International Conference on Machine Learning*. 2702–2711.
[12] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*. 3837–3845.
[13] Olivier Delalleau and Yoshua Bengio. 2011. Shallow vs. deep sum-product networks. In *Advances in Neural Information Processing Systems*. 666–674.
[14] Stefan Dernbach, Arman Mohseni-Kabir, Siddharth Pal, and Don Towsley. 2018. Quantum Walk Neural Networks for Graph-Structured Data. In *International Workshop on Complex Networks and their Applications*. Springer, 182–193.
[15] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*. 2224–2232.
[16] Ronen Eldan and Ohad Shamir. 2016. The power of depth for feedforward neural networks. In *Conference on Learning Theory*. 907–940.
[17] Andre Elisseeff, Theodoros Evgeniou, and Massimiliano Pontil. 2005. Stability of randomized learning algorithms. *Journal of Machine Learning Research* 6, Jan (2005), 55–79.
[18] Alberto García-Durán and Mathias Niepert. 2017. Learning Graph Representations with Embedding Propagation. *arXiv preprint arXiv:1710.03059* (2017).
[19] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212* (2017).
[20] Willem H Haemers. 1995. Interlacing eigenvalues and graphs. *Linear Algebra and its applications* 226 (1995), 593–616.
[21] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.
[22] Moritz Hardt, Benjamin Recht, and Yoram Singer. 2015. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240* (2015).
[23] David Haussler. 1990. *Probably approximately correct learning*. University of California, Santa Cruz, Computer Research Laboratory.
[24] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. 2017. Generalization in deep learning. *arXiv preprint arXiv:1710.05468* (2017).
[25] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[26] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).
[27] Risi Kondor, Hy Truong Son, Horace Pan, Brandon Anderson, and Shubhendu Trivedi. 2018. Covariant Compositional Networks For Learning Graphs. *arXiv preprint arXiv:1801.02144* (2018).
[28] Thomas J Laffey and Helena Šmigoc. 2008. Spectra of principal submatrices of nonnegative matrices. *Linear Algebra Appl.* 428, 1 (2008), 230–238.
[29] Tao Lei, Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2017. Deriving neural architectures from sequence and graph kernels. *arXiv preprint arXiv:1705.09037* (2017).
[30] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. 2018. Adaptive Graph Convolutional Neural Networks. *arXiv preprint arXiv:1801.03226* (2018).
[31] Hrushikesh N Mhaskar and Tomaso Poggio. 2016. Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications* 14, 06 (2016), 829–848.
[32] Sayan Mukherjee, Partha Niyogi, Tomaso Poggio, and Ryan Rifkin. 2006. Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics* 25, 1-3 (2006), 161–193.
[33] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. 2017. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*. 5947–5956.
[34] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *Proceedings of the 33rd annual international conference on machine learning. ACM*.
[35] Gilles Puy, Srdan Kitic, and Patrick Pérez. 2017. Unifying local and non-local signal processing with graph CNNs. *arXiv preprint arXiv:1702.07759* (2017).
[36] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 2017. 3d graph neural networks for rgbd semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5199–5208.
[37] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*. Springer, 593–607.
[38] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine* 30, 3 (2013), 83–98.
[39] Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *arXiv preprint arXiv:1809.02040* (2018).
[40] Felipe Petroski Such, Shagan Sah, Miguel Alexander Dominguez, Suhas Pillai, Chao Zhang, Andrew Michael, Nathan D Cahill, and Raymond Ptucha. 2017. Robust spatial filtering with graph convolutional neural networks. *IEEE Journal of Selected Topics in Signal Processing* 11, 6 (2017), 884–896.
[41] Shiliang Sun, Zakria Hussain, and John Shawe-Taylor. 2014. Manifold-preserving graph reduction for sparse semi-supervised learning. *Neurocomputing* 124 (2014), 13–21.
[42] Matus Telgarsky. 2016. Benefits of depth in neural networks. *arXiv preprint arXiv:1602.04485* (2016).
[43] Antoine J-P Tixier, Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. 2018. Graph Classification with 2D Convolutional Neural Networks. (2018).
[44] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep graph infomax. *arXiv preprint arXiv:1809.10341* (2018).
[45] Saurabh Verma and Zhi-Li Zhang. 2018. Graph Capsule Convolutional Neural Networks. *arXiv preprint arXiv:1805.08090* (2018).
[46] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks? *arXiv preprint arXiv:1810.00826* (2018).

[47] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530* (2016).

[48] Zhihong Zhang, Dongdong Chen, Jianjia Wang, Lu Bai, and Edwin R Hancock. 2019. Quantum-based subgraph convolutional neural networks. *Pattern Recognition* 88 (2019), 38–49.

8 APPENDICES

Proof of Theorem 2: To derive generalization bounds for uniform stable randomized algorithms, we utilize McDiarmid's concentration inequality. Let $\mathbf{X}$ be a random variable set and $f : \mathbf{X}^m \to R$, then the inequality is given as,

$$\begin{aligned} &if \sup_{x_1,..,x_i,..,x_m,x_i'} |f(x_1,..,x_i,..,x_m) - f(x_1,..,x_i',..,x_m)| \leq c_i \\ &= \sup_{x_1,..,x_i,..,x_m,x_i'} |f_S - f_{S^i}| \leq c_i \quad ,\forall i \\ &\implies P\Big(f(S) - \mathbf{E}_S[f(S)] \geq \epsilon\Big) \leq e^{-\frac{2\epsilon^2}{\sum_{i=1}^m c_i^2}} \end{aligned} \tag{10}$$

We will derive some expressions that would be helpful to compute variables needed for applying McDiarmid's inequality.

Since the samples are i.i.d., we have

$$\begin{aligned} \mathbf{E}_S[\ell(A_S, \mathbf{z})] &= \int \ell\big(A(\mathbf{z}_1, ..., \mathbf{z}_m), \mathbf{z}\big)p(\mathbf{z}_1, ..., \mathbf{z}_m)d\mathbf{z}_1...d\mathbf{z}_m \\ &= \int \ell\big(A(\mathbf{z}_1, ..., \mathbf{z}_m), \mathbf{z}\big)p(\mathbf{z}_1)...p(\mathbf{z}_m)d\mathbf{z}_1...d\mathbf{z}_m \end{aligned} \tag{11}$$

Using Equation 11 and renaming the variables, one can show that

$$\begin{aligned} \mathbf{E}_S[\ell(A_S, \mathbf{z}_j)] &= \int \ell\big(A(\mathbf{z}_1, .., \mathbf{z}_j, .., \mathbf{z}_m), \mathbf{z}_j\big)\times \\ &\qquad p(\mathbf{z}_1, .., \mathbf{z}_j, .., \mathbf{z}_m)d\mathbf{z}_1...d\mathbf{z}_m \\ &= \int \ell\big(A(\mathbf{z}_1, .., \mathbf{z}_j, .., \mathbf{z}_m), \mathbf{z}_j\big)p(\mathbf{z}_1)..p(\mathbf{z}_j)..p(\mathbf{z}_m)d\mathbf{z}_1...d\mathbf{z}_m \\ &= \int \ell\big(A(\mathbf{z}_1, .., \mathbf{z}_i', .., \mathbf{z}_m), \mathbf{z}_i'\big)p(\mathbf{z}_1)..p(\mathbf{z}_i')..p(\mathbf{z}_m)d\mathbf{z}_1..d\mathbf{z}_i'..d\mathbf{z}_m \\ &= \int \ell\big(A(\mathbf{z}_1, .., \mathbf{z}_i', .., \mathbf{z}_m), \mathbf{z}_i'\big)p(\mathbf{z}_1, .., \mathbf{z}_i', .., \mathbf{z}_m)d\mathbf{z}_1..d\mathbf{z}_i'..d\mathbf{z}_m\times \\ &\qquad \int p(\mathbf{z}_i)d\mathbf{z}_i \\ &= \int \ell\big(A(\mathbf{z}_1, .., \mathbf{z}_i', .., \mathbf{z}_m), \mathbf{z}_i'\big)p(\mathbf{z}_1, .., \mathbf{z}_i, \mathbf{z}_i', .., \mathbf{z}_m)d\mathbf{z}_1...d\mathbf{z}_m d\mathbf{z}_i' \\ &= \mathbf{E}_{S,z_i'}[\ell(A_{S^i}, \mathbf{z}_i')] \end{aligned} \tag{12}$$

Using Equation 12 and $\beta-$uniform stability, we obtain

$$\begin{aligned} &\mathbf{E}_S[\mathbf{E}_A[R(A)] - \mathbf{E}_A[R_{emp}(A)]] = \mathbf{E}_S[\mathbf{E}_z[\mathbf{E}_A[\ell(A_S, \mathbf{z})]]] - \\ &\qquad \frac{1}{m}\sum_{j=1}^m \mathbf{E}_S[\mathbf{E}_A[\ell(A_S, \mathbf{z}_j)]] \\ &= \mathbf{E}_S[\mathbf{E}_z[\mathbf{E}_A[\ell(A_S, \mathbf{z})]]] - \mathbf{E}_S[\mathbf{E}_A[\ell(A_S, \mathbf{z}_j)]] \\ &= \mathbf{E}_{S,z_i'}[\mathbf{E}_A[\ell(A_S, \mathbf{z}_i')]] - \mathbf{E}_{S,z_i'}[\mathbf{E}_A[\ell(A_{S^i}, \mathbf{z}_i')]] \\ &= \mathbf{E}_{S,z_i'}[\mathbf{E}_A[\ell(A_S, \mathbf{z}_i') - \ell(A_{S^i}, \mathbf{z}_i')]] \\ &\leq \mathbf{E}_{S,z_i'}[\mathbf{E}_A[|\ell(A_S, \mathbf{z}_i') - \ell(A_{S^i}, \mathbf{z}_i')|]] \\ &\leq 2\beta \end{aligned} \tag{13}$$

$$\begin{aligned} &|\mathbf{E}_A[R(A_S) - R(A_{S^i})]| = |\mathbf{E}_z[\mathbf{E}_A[\ell(A_S, \mathbf{z})]] - \mathbf{E}_z[\mathbf{E}_A[\ell(A_{S^i}, \mathbf{z})]]| \\ &= |\mathbf{E}_z[\mathbf{E}_A[\ell(A_S, \mathbf{z})] - \mathbf{E}_A[\ell(A_{S^i}, \mathbf{z})]]| \\ &\leq \mathbf{E}_z[\mathbf{E}_A[|\ell(A_S, \mathbf{z})] - \mathbf{E}_A[\ell(A_{S^i}, \mathbf{z})|]] \\ &\leq \mathbf{E}_z[\beta] = 2\beta \end{aligned} \tag{14}$$

$$\begin{aligned} &|\mathbf{E}_A[R_{emp}(A_S)] - R_{emp}(A_{S^i})]| \leq \\ &\qquad |\frac{1}{m}\sum_{j=1, j\neq i}^m (\mathbf{E}_A[\ell(A_S, \mathbf{z}_j) - \ell(A_{S^i}, \mathbf{z}_j)])| + \\ &\qquad |\frac{1}{m}(\mathbf{E}_A[\ell(A_S, \mathbf{z}_i) - \ell(A_{S^i}, \mathbf{z}_i')])| \end{aligned} \tag{15}$$

$$\leq 2\frac{(m-1)}{m}2\beta + \frac{M}{m}$$

$$\leq 2\beta + \frac{M}{m}$$

Let $K_S := R(A_S) - R_{emp}(A_S)$.

Using Equation 14 and Equation 15, we have

$$\begin{aligned} |\mathbf{E}_A[K_S] - \mathbf{E}_A[K_{S^i}]| &= \Big|\mathbf{E}_A[\big(R(A_S) - R_{emp}(A_S)\big)] \\ &\qquad - \mathbf{E}_A[\big(R(A_{S^i}) - R_{emp}(A_{S^i})\big)]\Big| \\ &\leq \Big|\mathbf{E}_A[R(A_S)] - \mathbf{E}_A[R(A_{S^i})]\Big| + \Big|\mathbf{E}_A[R_{emp}(A_S)] \\ &\qquad - \mathbf{E}_A[R_{emp}(A_{S^i})]\Big| \\ &\leq 2\beta + (2\beta + \frac{M}{m}) \\ &\leq 4\beta + \frac{M}{m} \end{aligned} \tag{16}$$

Applying McDiarmid's concentration inequality,

$$P\Bigg(\mathbf{E}_A[K_S] - \mathbf{E}_S[\mathbf{E}_A[K_S]] \geq \epsilon\Bigg) \leq \underbrace{e^{-\frac{2\epsilon^2}{m(4\beta + \frac{M}{m})^2}}}_{\delta}$$

$$P\Bigg(\mathbf{E}_A[K_S] \leq 2\beta + (4m\beta + M)\sqrt{\frac{\log\frac{1}{\delta}}{2m}}\Bigg) \geq 1 - \delta$$

This complete the proof of Theorem 2.