博士論文

Analysis of Deep Learning from the Viewpoint of Model Structures
（モデル構造的見地からの深層学習解析）

大野 健太

Abstract

Machine learning (ML) models are designed to make full use of knowledge inherent in tasks. In Deep Learning (DL), which has become a significant movement in the ML community from the late 2000s, predictive models reflect the task knowledge to their *structures*, specifically computational units in computational graphs and the graphs' topology. For example, Convolutional Neural Network (CNN) and Graph Neural Network (GNN) are standard DL models for images and graph-structured data, respectively. They are intentionally designed to exploit the inductive bias of structures in data and show superior empirical performance. Many studies have clarified the strength of DL theoretically. However, their focus was mostly on Fully-connected Neural Networks (FNN), the most general architecture in DL, and less attention has been paid to the theoretical analysis of DL models with such specific structures.

In this study, we study how structures in DL models affect their theoretical characteristics, such as expressive power, optimization ability, and generalization ability. In particular, we pay attention to the following three structures: skip connections in CNNs, node aggregation operations of GNNs, and multi-scale GNNs.

First, we analyze the expressive power and generalization ability of CNNs with skip connections. Skip connections bypass intermediate computational units and directly feed the outputs of a unit to non-direct descendants. Residual Network (ResNet), a specific type of CNNs with skip connections, enabled us to stack more than 100 layers in image recognition tasks thanks to its special computational graph topology induced by skip connections. We theoretically analyze the effectiveness of skip connections from the viewpoint of *sparsity*. Specifically, we derive the estimation error rates of ResNet-type CNNs in non-parametric regression problems. As a corollary, we show the minimax optimality of *dense* ResNet-type CNNs in a particular setting, which has not been known for FNNs without unrealistic assumptions on model sparsity. The key of our analysis is to associate the expressive power of a ResNet-type CNNs with that of an FNN with special sparse structures that we coin a *block-sparse* FNN. Our result implies that skip connections in CNNs induce implicit sparse structures and promote their theoretical performance.

Next, we elucidate the expressive power of (single-scale) GNNs. Many GNN models have node aggregation operations in their architecture. These operations nicely reflect the inductive bias of graph learning tasks that neighboring nodes are highly correlated. However, they cause a side effect at the same time known as *over-smoothing*, in which deep GNNs with many node aggregations degenerate predictive performance. We investigate the expressive power of non-linear Graph Convolution Network (GCN), which is one of the most popular GNN models, with ReLU activation functions. We interpret the forward propagation of a GCN as discrete-time dynamics

and explain the over-smoothing as a distance to an invariant space that is "information-less" for node prediction tasks. We prove the convergence rate of a ReLU-GCN to the invariance space is the same as that of the corresponding linear GCN, a GCN without ReLU functions. It suggests that non-linearity in the model does not help to mitigate the over-smoothing problem of GNNs.

Finally, we develop optimization and learning theories for *multi-scale* GNNs. Multi-scale GNNs directly connect the output of intermediate layers to the final output using skip connections. They reflect the inductive bias that subgraphs with various radii contribute to the prediction of the center node. Our idea is to interpret a multi-scale GNN as an ensemble of single-scale GNNs and apply boosting theory. We derive the optimization and generalization bounds of a particular type of multi-scale GNNs in transductive learning settings under a weak-learning-type assumption, which is a standard condition in boosting theory that we can adequately train weak learners (i.e., a GNN with a specified scale). In particular, AdaBoost-like test error bounds are obtained, which advance our understanding of the role of skip connections in multi-scale GNNs as a countermeasure against the loss of expressive power caused by the over-smoothing.

Our analysis gives theoretical justifications for architectural choices of practically successful DL models and principled guidance for their further improvements.

Acknowledgment

Many people have supported my Ph.D. activities in various ways. I could not have accomplished it without their support. Since I cannot list them all, I only mention those to whom I am particularly thankful. However, I appreciate all people who have kindly encouraged and support my activities.

First and foremost, I would like to express my gratitude to my supervisor Dr. Taiji Suzuki, Associate Professor of the University of Tokyo, for his continuous help and encouragement during my Ph.D. activities. I am always impressed by his extraordinary knowledge and insightful suggestions. His professional attitude toward research has motivated me to achieve higher goals. I would appreciate Dr. Kenji Yamanishi, Professor of the University of Tokyo, for giving me guidance on my Ph.D. study. His critical comments rooted in his profound research experience have significantly improved my research work.

I thank my colleagues at the university. Especially, I thank Dr. Atsushi Nitanda and Dr. Sho Sonoda. The quality of my Ph.D. study has been greatly improved through many discussions with them. I also thank Maki Norikane for supporting me to do paperwork related to the Ph.D. activities fluently.

I conducted my Ph.D. study while working at Preferred Networks, Inc., Japan, as an engineer. I appreciate Toru Nishikawa, Chief Executive Officer, and Dr. Daisuke Okanohara, Chief Operating Officer, for kindly allowing me to pursue my career for the Ph.D. course. I also thank my colleagues at the company. I especially thank Dr. Kohei Hayashi, Shohei Hido, Dr. Masanori Koyama, Dr. Kentaro Minami, and Dr. Kazue Mizuno. My research activity in the doctoral course would not be possible without their support.

I thank anonymous reviewers and research collaborators for giving us insightful feedback and comments on my studies. I especially thank Dr. Masahiro Ikeda, Dr. Masaaki Imaizumi, Dr. Isao Ishikawa, Takeshi Teshima, and Dr. Koichi Tojo.

Finally, I am grateful for my family. My parents Etsuko Oono and Ryuichi Oono, gave me ample opportunity for education. I would like to express my deepest gratitude and love to my wife, Haru Negami Oono, whom I respect as a family member and a mathematical science researcher.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Structures in Deep Learning Models

Machine learning (ML) is a technology for having machines recognize, decide, and act intellectually by making full use of knowledge inherent in data. With the increase of data generated in the world, the importance of ML technology is rapidly growing as a way of exploiting data.

ML theory has shown several limitations of learning without prior information about data. For example, informally speaking, the no-free lunch theorem [Wolpert, 1996] showed that no ML models perform universally well for any data distribution. It implies that we need to narrow down data distributions that are likely to happen using prior information about the task and adopt ML models tailored to these distributions. Such theoretical limitations motivated us to explore methodologies for effectively reflecting biases of specific tasks (known as *inductive bias*) into ML models.

Deep Learning (DL) has become a great movement in the ML community since the late 2000s [Hinton et al., 2006, Salakhutdinov and Hinton, 2009]. For example, in 2012, DL methods won first place in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition, one of the largest image recognition competitions [Krizhevsky et al., 2012, 2017]. In speech recognition tasks, a seminal paper [Hinton et al., 2012] employed DL models and outperformed conventional methods.

DL models are often represented as a computational graph consisting of differentiable computation units. For example, Figure 1.1 is a computational of graph of Fully-connected Neural Network (FNN), also known as Multi-layer Perceptron (MLP), which is the most standard DL model. It represents the following computational procedures:

$$\begin{aligned}
h_1 &= \mathrm{Affine}(x; W_1, b_1) = W_1 x + b_1,\\
a_1 &= \mathrm{ReLU}(h_1),\\
h_2 &= \mathrm{Affine}(a_1; W_2, b_2) = W_2 a_1 + b_2,\\
p &= \mathrm{Softmax}(h_2),\\
l &= \mathrm{CrossEntropy}(p, t).
\end{aligned}$$

Here, Affine, ReLU, $\mathrm{Softmax}$, $\mathrm{CrossEntropy}$ are arithmetic operations, represented as computational units. Each operation is differentiable with respect to its input variables. Thanks to this property, we can train the whole model in an end-to-end manner using the combination of the backpropagation algorithm and stochastic gradient descent (see, e.g., Goodfellow et al. [2016] for the training algorithms of DL models.)

Most practically successful DL models are designed to reflect the inductive bias of problems to their *structures*, that is, the topology of the computational graph or computational units therein. For example, Convolutional Neural Network (CNN) is a popular DL model, tracing back the origin at least to the 1980s as Neocognitron [Fukushima and Miyake, 1982]. CNN is a de-facto model for analyzing grid-like objects such as sequences (1D), images represented by pixels (2D), and spatial objects represented by voxels (3D). The essence of modern CNN architectures comes from LeNet5 [LeCun et al., 1998], whose building blocks are a convolutional layer, pooling layer, and fully-connected layer. These layers are designed to reflect the inductive biases of grid-like objects. Specifically, the convolutional layer captures the translational invariance of the data, and the pooling layer captures the robustness of the image semantics to small perturbation by noise injection (see Section 2.1.1 for the mathematical definition of convolution operations).

Structures in DL models are critical factors of the practical success of DL. However, we would argue that there is a considerable gap between their empirical effectiveness and theoretical understanding of their role. On the one hand, researchers and engineers worldwide deeply analyze their models and tasks using their creativity every day and work out novel DL models with unique structures capturing the tasks' inductive bias. However, their justification is made mostly only through the empirical evaluation and is not examined via rigorous theory. On the other hand, many studies have clarified the strength of DL theoretically using various mathematical and statistical tools. However, since their main focus was mostly on FNN, the most general architecture in DL, they have limitations in understanding the power of practically successful DL models with specific structures. This situation motivates us to pose the following research question:

Research Question. *How do structures in DL models affect their theoretical characteristics?*

Filling the gap between practice and theory of DL is a critical issue we must address. In science and engineering, it often happens that theoretical justification lags behind practical usefulness. However, without theory, practitioners must rely on the tacit knowledge of domain experts. This problem is particularly critical for DL because DL is notoriously hard for tuning hyperparameters. Theory clarifies the essential enablers and limiting factors of the technology. It provides us trustful guidance on how to tame the technology and develop it further. The rigorous theory also makes the technology trustworthy because we can identify its potential pitfalls, even domain experts are not aware of. Therefore, we believe that the theoretical understanding of models' structures contributes to spreading the DL technology to the real-world.

There are numerous DL structures to date. Among others, we focus on the following three structures because they are empirically particularly important, but their theoretical understanding is insufficient: skip connections in CNNs, node aggregation operation of Graph Neural Networks (GNNs), and skip connections in multi-scale GNNs. We explain these structures and their research questions one by one in the following sections.

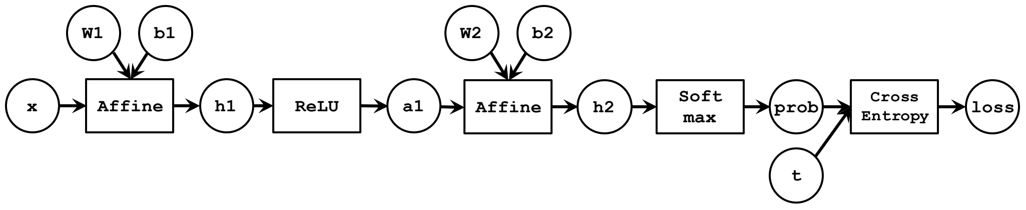


Figure 1.1: Computational graph of a two-layered FNN. Circles and rectangles in the figure represent variables and computational units, respectively.

1.2 Residual Networks (ResNets)

Training of deep CNNs has several difficulties, such as vanishing and exploding gradient problems. These problems limited the depth of classical CNNs to at most twenty layers. For example, AlexNet [Krizhevsky et al., 2012, 2017], an early CNN model that sparked the latest DL boom, had eleven layers. Activation normalization methods (such as Batch Normalization [Ioffe and Szegedy, 2015]) or weight initialization methods (such as the Xavier initialization [Glorot and Bengio, 2010] and the He initialization [He et al., 2015]) could mitigate these problems. However, it has been known that even though such methods enabled to converge the training of deep models, the performance was degraded compared to shallow counterparts [He et al., 2016]. See Section 2.1.2 for details about the vanishing and exploding gradient problems.

Residual Network (ResNet) [He et al., 2016] was designed to solve the performance degrade of deep CNNs. The architecture of ResNet is based on the hypothesis that it is easier for a model to learn residuals of the target function that are not accounted for in the upstream part of the model, rather than learning the whole function [He et al., 2016]. To realize this, ResNet consisted of subnetworks representing functions of the form $\mathrm{id} + F$ and concatenated them in series. Here, id is the identity function and F is a dense (and typically shallow) subnetwork called a *residual block*, or *resblock*. ResNet employed a skip connection which bypassed the network F to realize the identity function (Figure 1.2). This architectural trick enabled us to learn ResNets with more than 100 layers and won the ILSVRC competition in 2015. Inspired by the success of ResNet, many CNN models with skip connections have been proposed. For example, the Squeeze-and-Excitation Network (SENet) [Hu et al., 2018, 2020a], which is the winner of the ILSVRC competition in 2017, employed skip connections for better performance (see Section 2.1.4 for representative models of ResNet-type CNNs.)

Many studies have focused on understanding the success of ResNet-type CNNs theoretically and explained the practical effectiveness of skip connections, such as Veit et al. [2016], Zhou [2018], to name a few. However, to the best of our knowledge, few of them have given explanations from the viewpoint of statistical learning theory.

Problem 1. *Why do skip connections promote predictive performance of CNNs?*

In Chapter 3, we answer this question from the viewpoint of *sparsity*. We propose to hypothesize that the architecture of ResNet inherits a special type of sparse structures, we call a *block-sparse* structure and that this structure promotes theoretical superiority over classical CNNs.

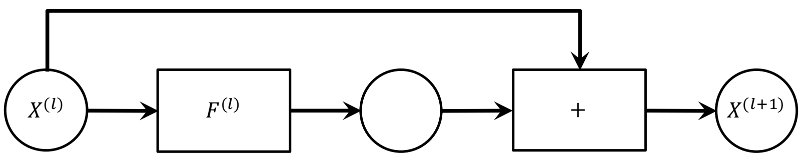


Figure 1.2: Computational graph of a resblock. $X^{(l)}$ represents the output of the l-th resblock and $F^{(l)}$ is the l-th resblock.

1.3 Graph Neural Networks (GNNs)

Graphs are a universal data structure representing relationships between entities. We can represent many data types as a graph, from small graphs such as chemical compounds to gigantic graphs such as social networks. Inspired by the success of DL, there has been a movement to apply DL models to structured data whose topology is more complex than that of image or sequential data. Applying ML models to graphs has several difficulties, such as node order invariance and inhomogeneous neighborhood structures (see Section 2.2.5). Graph Neural Networks (GNNs), pioneered by Gori et al. [2005], Scarselli et al. [2009], are a collective term of DL models for graph-structured data that solved these problems (Figure 1.3). GNNs have benefits in common with classical DL models such as FNNs and CNNs. For example, since a GNN consist of differentiable operators with respect to input variables and learnable parameters, we can train the whole model in an end-to-end manner. Also, similar to convolution operators in a CNN, GNNs can apply to graphs with different sizes. GNNs have been empirically successful for data analysis in various fields such as biochemistry [Duvenaud et al., 2015], computer vision [Yang et al., 2018], and knowledge graph analysis [Schlichtkrull et al., 2018].

Message Passing Neural Network (MPNN) [Gilmer et al., 2017] is a subtype of GNNs. Its architecture consists of interleaving of two types of operations: node *aggregation* and node *update* operations. Given representations for each node on a graph, each node collects representations of its neighboring nodes in the aggregation operation. Then, each node renews its representation using the collected information in the update operation. MPNN-type GNNs are popular due to its empirical performance and implementation simplicity. Among others, Graph Convolution Networks (GCN) [Kipf and Welling, 2017] is the most standard MPNN-type GNNs and has been applied in wide fields. See Section 2.3.2 for more details about MPNN-type GNNs.

1.4 Over-smoothing of GNNs

Node aggregation operations in GCNs resemble convolution operations of CNNs (see Section 2.4.3). However, as opposed to CNNs, deep GCNs (for example, more than ten layers) cannot perform better than shallow ones in practical situations [Kipf and Welling, 2017, Zhang et al., 2020]. Rong et al. [2020] reported that Graph Attention Network (GAT) [Veličković et al., 2018], another MPNN-type GNN variant, also suffered from the performance degradation.

Li et al. [2018b] attributed the performance degradation of a deep GCN to the degeneration of node representations, a phenomenon they coined *over-smoothing*. We can understand the over-smoothing phenomenon intuitively as follows. When we apply the aggregation operation, it

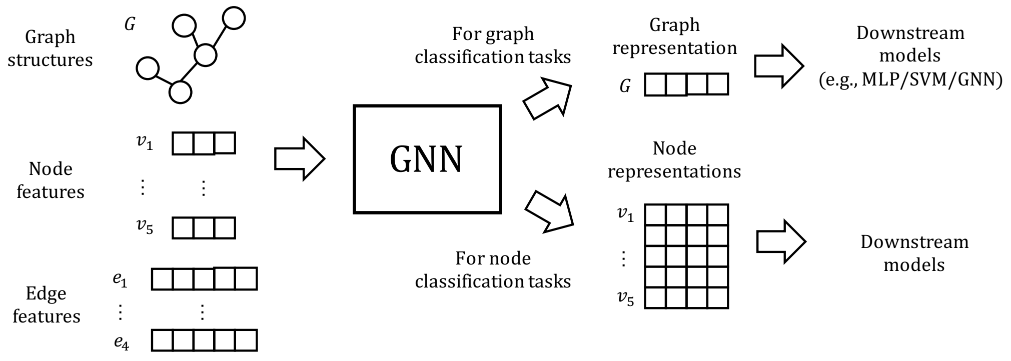


Figure 1.3: GNN overview.

smoothens node representations and makes them close to each other. Finally, node representations go indistinguishable when we repeat it many times. That is, node representations are over-smoothened. We can think of the over-smoothing phenomenon as a side effect of the architectural design of GNNs. For *linear* GNNs, that is, GNNs whose activation functions are the identity functions, several studies have proven that this intuition is correct [Li et al., 2018b, Zhang, 2019, Zhao and Akoglu, 2020] (see Section 2.5.3).

In classical DL models such as FNNs and CNNs, non-linearity introduced by activation functions can contribute to the theoretical and empirical performance. For example, an FNN with a single hidden layer with a sigmoid or ReLU activation function is a universal approximator for continuous functions, that is, it is sufficiently expressive so that it can arbitrarily approximate any continuous functions thanks to the non-linearity [Cybenko, 1989, Sonoda and Murata, 2017]. Given that, one may hope that non-linearity can be beneficial to GNNs, too. However, to the best of our knowledge, there has been little theoretical explanation for the over-smoothing of non-linear GNNs.

Problem 2. *Can non-linearity mitigate over-smoothing of GNNs caused by node aggregations?*

In Chapter 4, we answer this question negatively in a specific situation. We interpret the forward propagation of a GNN as a discrete-time dynamical system and characterize the over-smoothing as the convergence to an invariance space of the dynamics that is “information-less" for node prediction tasks (see Section 2.2.2 for the overview of graph ML tasks including node prediction tasks). We prove that the convergence rate to the invariant space for ReLU GCNs is the same as that for their linear counterparts, implying that ReLU GCNs are as vulnerable to over-smoothing as linear GCNs.

1.5 GNNs with Skip Connections

Over-smoothened GNNs are not appropriate for node prediction tasks because loss of their expressive power can cause under-fitting. As we saw in Section 1.2, skip connections enable us to stack many layers in the case of CNNs. Therefore, it is natural to think that we may solve the over-smoothing problem by introducing skip connections to GNNs. Several studies have shown that

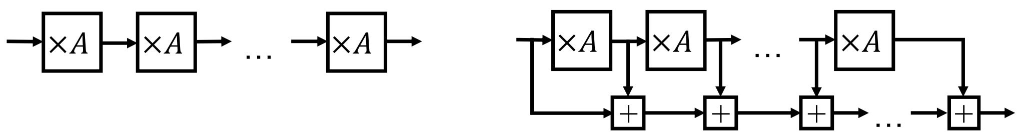


Figure 1.4: Single-scale GNN and mutli-scale GNN. $\times A$ and $+$ represents a node aggregation and an addition operations, respectively. We omit ndoe update operations for simplicity.

skip connections are practically useful to some extent. Among others, it is notable that in the paper of GCN, the authors reported that the performance degradation of deep GCNs is mitigated by skip connections (see Kipf and Welling [2017, Appendix B]). However, to the best of our knowledge, it has not been known for the theoretical explanations.

Problem 3. *What is the role of skip connections in GNNs? How do they affect the over-smoothing?*

In Chapter 5, we focus on *multi-scale* GNNs (e.g., Xu et al. [2018]) to answer this question. The idea of multi-scale GNNs is to exploit the inductive bias of tasks that subgraphs of various scales are beneficial for predicting nodes' properties. The most standard approach for realizing it is to connect intermediate outputs of a model to its final output directly using skip connections (Figure 1.4). For example, given a graph of size N, the following model is a simple example of multi-scale GNNs:

$$Z^{(l+1)} = \sigma(AZ^{(l)}W^{(l)}) \quad (l = 1, \ldots L)$$
$$Z = \sum_{l=1}^{L+1} Z^{(l)}.$$

Here, $Z^{(l)} \in \mathbb{R}^{N \times C}$ is concatenation of node representations at the l-th layer, where C is the dimension of node representations. $Z \in \mathbb{R}^{N \times C}$ is the final node representations. $A \in \mathbb{R}^{N \times N}$ is the (normalized) adjacency matrix of the graph (see Section 2.4). $W^{(l)} \in \mathbb{R}^{C \times C}$ is a learnable weight matrix at the l-th block. $\sigma : \mathbb{R} \to \mathbb{R}$ is an activation function, applied in an element-wise manner. We can realize the summation functions above using skip connections. See Section 2.6.2 for representative models of multi-scale GNNs.

Our analysis key is that we can interpret a multi-scale GNN as an ensemble of single-scale GNNs. This interpretation enables us to apply the boosting theory. We derive optimization and generalization bounds for multi-scale GNNs in node prediction tasks. These bounds give us insights when multi-scale structures induced by skip connections are effective as a countermeasure of the over-smoothing problem.

1.6 Organization of Dissertation

In Chapter 2, we introduce the backgrounds of the following chapters. First, we explain the architecture of ResNet to explain its design intent. Then, we provide representative variants of ResNet-type CNNs. Regarding the basics of GNNs, we first overview ML tasks on graphs, in particular, classification of graph types and task types to clarify problem settings on which we focus. Among others, we mainly focus on node prediction tasks on a graph and formulate it as semi-supervised learning problems (more specifically, transductive learning problems). After that, we showcase representative models of GNNs, including convolution-based GNNs and multi-scale GNNs. Finally, we briefly explain statistical learning theories of supervised learning problems for ResNet-type CNNs and transductive learning problems for GNNs.

In Chapter 3, we analyze the approximation and estimation errors in non-parametric regression problems using ResNet-type CNNs, toward the goal of clarifying the role of skip connections in ResNet-type CNNs. The analysis key is FNNs with special sparse structures, which we coin block-sparse FNNs, that have statistically favorable properties. By associating the expressive power of ResNet-type CNNs with that of block-sparse FNNs, we derive the approximation and estimation bounds of ResNet-type CNNs, which inherit the favorable properties of block-sparse FNNs. In particular, we show that, ResNet-type CNNs can achieve optimality in the sense of minimax optimal (ignoring log factors) without unrealistic constraints on sparsity when the true function is in the Hölder class. From this result, we hypothesize that the skip connections of ResNet-type CNNs implicitly induce special sparse structures, which contribute to their excellent performance.

In Chapter 4, we analyze the expressive power of ReLU GCNs in node classifications tasks to explain the over-soothing phenomenon when we introduce non-linearity to GNNs. Our theory associates the representation power with underlying graph spectra. The key idea is to interpret the ReLU function as a projection on the cone $\{X \geq 0\}$. This fact works well with the underlying graph Laplacian (see Section 2.4.1 for the basics of spectral graph theory). We prove that the convergence rate of the ReLU GCN to the invariant space that is information-less for node prediction tasks is the same as that of the linear counterparts. From this result, we suggest that in the case of ReLU GCNs, non-linear activation functions do not help mitigate the over-smoothing phenomena.

In Chapter 5, we study the optimization and generalization performance of multi-scale GNNs to investigate how skip connections affect the performance of GNNs. The key to the analysis is that by interpreting a multi-scale GNN as an ensemble of single-scale GNNs, we can train it using boosting algorithms. Under a weak learning condition (w.l.c.), which is a standard type of assumptions in the boosting theory, we derive an AdaBoost-like optimization and generalization performance bounds for multi-scale GNNs trained using boosting algorithms. In particular, when the model satisfies w.l.c. with "good" parameters, the performance of multi-scale GNNs is monotonically increasing as the depth (the number of node aggregations) increases, as opposed to the over-smoothing problem of single-scale GNNs we observe in Chapter 4. Our results advance the understanding of multi-scale structures induced by skip connections to counter the over-smoothing problems.

In the final chapter, we summarize each chapter and discuss future directions. We point out that analyzing continuous models, that is, the continuity limit of discrete models, is a promising approach for highlighting structures' characteristics. We conclude this dissertation by emphasizing the importance of unified theory for various architectures to understand the role of structures in DL models.

Chapter 3 is based on the study of Oono and Suzuki [2019], Chapter 4 is based on the study of Oono and Suzuki [2020a], and Chapter 5 is based on the study of Oono and Suzuki [2020b],

Table 1.1: Experiment Code URL

Chapter	URL
Chapter 4	`https://github.com/delta2323/gnn-asymptotics`
Chapter 5	`https://github.com/delta2323/GB-GNN`

respectively. The author also contributed to the work of Teshima et al. [2020]. Experiment code is available from the URLs in Table 1.1.

1.7 Notation

We denote the set of non-negative integers by $\mathbb{N}$, and the set of positive integers by $\mathbb{N}_+ := \{1, 2, \ldots\}$. For $M \in \mathbb{N}_+$, the set of positive integers less than or equal to M by $[M] := \{1, \ldots, M\}$. Let $a, b \in \mathbb{R}$ be scalars. We define $a \vee b := \max(a, b)$, $a \wedge b := \min(a, b)$ for $a, b \in \mathbb{R}$, respectively. We define the sign function by $\operatorname{sign}(a) = 1$ if $a \geq 0$ and -1 otherwise. For a proposition P, $\mathbf{1}\{P\}$ equals 1 when P is true and 0 otherwise. For $y \in \{0, 1\}$, we write $y^\sharp := 2y - 1 \in \{\pm 1\}$. For a sequence $\boldsymbol{w} = (w^{(1)}, \ldots, w^{(L)})$ and $l \leq l'$, we denote its subsequence from the l-th to l'-th elements by $\boldsymbol{w}[l : l'] := (w^{(l)}, \ldots, w^{(l')})$.

All vectors are column vectors. Let $u, v \in \mathbb{R}^N$ and $w \in \mathbb{R}^C$ be vectors and $X, Y \in \mathbb{R}^{N \times C}$ be matrices. $v \otimes w \in \mathbb{R}^{N \times C}$ denotes the Kronecker product of v and w defined by $(v \otimes w)_{nc} := v_n w_c$. $\operatorname{diag}(v) := (v_n \delta_{nm})_{n, m \in [N]} \in \mathbb{R}^{N \times N}$ denotes the diagonalization of v. We write $v \geq 0$ if and only if $v_n \geq 0$ for all $n \in [N]$. Similarly, for a matrix $X \in \mathbb{R}^{N \times C}$, we write $X \geq 0$ if and only if $X_{nc} \geq 0$ for all $n \in [N]$ and $c \in [C]$. We say such a vector and matrix is *non-negative*. $I_N \in \mathbb{R}^{N \times N}$ denotes the identity matrix of size N.

$\langle \cdot, \cdot \rangle$ denotes the inner product of vectors or matrices, depending on the context: $\langle u, v \rangle := u^\top v$ for vectors and $\langle X, Y \rangle := \operatorname{tr}(X^T Y)$ and matrices. For $p \geq 1$, we denote the p-norm of v by $\|v\|_p^p := \sum_{n=1}^N v_n^p$, $(2, p)$-norm of X by $\|X\|_{2,p}^p := \sum_{c=1}^C \left(\sum_{n=1}^N X_{nc}^2 \right)^{\frac{p}{2}}$, the Frobenius norm of X by $\|X\|_{\mathrm{F}} := \|X\|_{2,2}$. For normed spaces $(V, \| \cdot \|_V), (W, \| \cdot \|_W)$ and a linear operator $T : V \to W$ we denote the operator norm of T by $\|T\|_{\mathrm{op}} := \sup_{\|v\|_V = 1} \|Tv\|_W$. For a subset $V' \subset V$, we denote the restriction of T to V' by $P|_{V'}$.

For a tensor a, we define the positive part of a by $a_+ := a \vee 0$ where the maximum operation is performed in element-wise manner. Similarly the negative part of a is defined as $a_- := -a \vee 0$. Note that $a = a_+ - a_-$ holds for any tensor a. For two tensors a, b of the same size, we define the Hadamard product $a \otimes b$ by the element-wise multiplication of two tensors.

We employ the Landau notations $o(\cdot)$, $O(\cdot)$, $\Omega(\cdot)$, and $\Theta(\cdot)$ to describe asymptotic growth rates. $o_P(\cdot)$, $O_P(\cdot)$, $\Omega_P(\cdot)$, and $\Theta_P(\cdot)$ denote the corresponding O-notation in probability. $\tilde{o}(\cdot)$, $\tilde{O}(\cdot)$, $\tilde{o}_P(\cdot)$, and $\tilde{O}_P(\cdot)$ ignore logarithmic factors.

Tables 1.2–1.5 show the list of symbols used in special meanings in each chapter. We define other symbols when they appear for the first time.

Table 1.2: Notation Table (Common).

Symbol	Description
$\mathcal{X}$	Domain of feature vectors. e.g., $\mathcal{X} = \mathbb{R}^C$ ($C \in \mathbb{N}_+$).
$\mathcal{Y}$	Domain of target values. e.g., $\mathcal{Y} = \{0, 1\}$, $\mathbb{R}$.
$\widehat{\mathcal{Y}}$	Range of predictors. e.g., $\widehat{\mathcal{Y}} = \mathbb{R}$.
$f^\circ : \mathcal{X} \to \mathcal{Y}$	(Unknown) true function.
$\widehat{f} : \mathcal{X} \to \widehat{\mathcal{Y}}$	Estimator function.
$\widehat{Y} \in \widehat{\mathcal{Y}}^N$	Collection of predictions.
N (inductive setting)	Training sample size.
N (transductive setting)	Total sample size.
M (transductive setting)	Training sample size.
U (transductive setting)	Test sample size.
C, C', C_*^* etc.	Channel size.
L, L', L_*, L_*^* etc.	Depth.
D	Input dimension.
K	Filter size.
W, W^*, W_*^* etc.	Weight matrices.
$\widehat{\mathcal{R}}(f)$, $\widehat{\mathcal{L}}(f)$, $\widehat{\mathcal{R}}(Y)$, $\widehat{\mathcal{L}}(Y)$	Training Error of $f : \mathcal{X} \to \mathcal{Y}$ or $Y \in \widehat{\mathcal{Y}}^N$.
$\mathcal{R}(f)$, $\mathcal{L}(f)$, $\mathcal{R}(Y)$, $\mathcal{L}(Y)$	Test Error of $f : \mathcal{X} \to \mathcal{Y}$ or $Y \in \widehat{\mathcal{Y}}^N$.
$\mathcal{D}$	Training dataset.
$\mathcal{P}$	(Unknown) true distribution over $\mathcal{X} \times \mathcal{Y}$ in inductive settings.
$\mathcal{P}_{\mathcal{X}}$	Marginalized distribution of $\mathcal{P}$ over $\mathcal{X}$.
$\mathcal{L}^2(\mathcal{P}_{\mathcal{X}})$	Set of real-valued functions over $\mathcal{X}$ that is square-integrable with respect to $\mathcal{P}_{\mathcal{X}}$.

Table 1.3: Notation Table (Chapter 3).

Symbol	Description
$\boldsymbol{C} = (C_1, \ldots, C_L)$	Sequence of channels.
$\boldsymbol{W} = (W_1, \ldots, W_M)$	Sequence of weights.
$\boldsymbol{b} = (b_1, \ldots, b_M)$	Sequence of biases.
θ	Learnable parameters of an FNN or a CNN.
$\mathrm{FC}^{\sigma}_{W,b} : \mathbb{R}^{D'} \to \mathbb{R}^{D''}$	Fully connected layer with weight matrix W and bias b.
$\mathrm{Conv}^{\sigma}_{W,b} : \mathbb{R}^{D \times C'} \to \mathbb{R}^{D \times C''}$	Convolution layer with filter weight W and bias b.
$\mathrm{FC}^{\sigma}_{\boldsymbol{W},\boldsymbol{b}} : \mathbb{R}^{D'} \to \mathbb{R}^{D''}$	$= \mathrm{FC}^{\sigma}_{W_M,b_M} \circ \cdots \circ \mathrm{FC}^{\sigma}_{W_1,b_1}$
$\mathrm{Conv}^{\sigma}_{\boldsymbol{W},\boldsymbol{b}} : \mathbb{R}^{D \times C'} \to \mathbb{R}^{D \times C''}$	$= \mathrm{Conv}^{\sigma}_{W_M,b_M} \circ \cdots \circ \mathrm{Conv}^{\sigma}_{W_1,b_1}$
$\mathrm{FNN}^{\sigma}_{\theta} : \mathcal{X} \to \widehat{\mathcal{Y}}$	FNN with learnable parameter set θ and activation function σ.
$\mathrm{CNN}^{\sigma}_{\theta} : \mathcal{X} \to \widehat{\mathcal{Y}}$	ResNet-type CNN with θ and σ.
$\mathrm{mCNN}^{\sigma}_{\theta} : \mathcal{X} \to \widehat{\mathcal{Y}}$	Masked CNNs with θ and σ.
$\mathcal{F}^{(\mathrm{FNN})} = \mathcal{F}^{(\mathrm{FNN})}_{*,*,\ldots} \subset \{\mathcal{X} \to \mathcal{Y}\}$	Hypothesis space of FNNs with a specified architecture (asterisks denote architectural parameters).
$\mathcal{F}^{(\mathrm{CNN})} = \mathcal{F}^{(\mathrm{CNN})}_{*,*,\ldots} \subset \{\mathcal{X} \to \mathcal{Y}\}$	Hypothesis space of ResNet-CNNs.
$\mathcal{G} = \mathcal{G}_{*,*,\ldots} \subset \{\mathcal{X} \to \mathcal{Y}\}$	Hypothesis space consisting of masked CNNs.
$\mathcal{N}(\varepsilon, \mathcal{M}, d)$	(External) convering number.
$\|\cdot\|_{\beta}$	β-Hölder norm.

Table 1.4: Notation Table (Chapter 4).

Symbol	Description
$G = (V, E)$	Simple graph with node set V and edge set E.
$\tilde{G}$	Augmented graph of G
$A = A(G)$	Normalized adjacency matrix.
$D = D(G)$	Degree matrix.
$\Delta = \Delta(G)$	Unnormalized graph Laplacian.
$L = L(G)$	Normalized garph Laplacian.
$\tilde{A} = \tilde{A}(G)$	Augmented normalized adjacency matrix ($= A(\tilde{G})$.)
$\tilde{D} = \tilde{D}(G)$	Degree matrix for autmented graph $\tilde{G}$ ($= D(\tilde{G})$.)
X	Collection of feature verctors on nodes on a graph.
$\mathcal{M}$	The information-less invariant space. (see Chapter 4).
$d_{\mathcal{M}}(X)$	The distance from X to an invariant subspace $\mathcal{M}$ with respect to the Frobenius norm.
s, s^* etc.	Singular values of a weight matrix.
$G_{N,p}$	Erdos-Renyi graph of parameters N (node size) and p (edge probability).

Table 1.5: Notation Table (Chapter 5).

Symbol	Description
$\widehat{\mathcal{R}}$, $\mathcal{R}$	Training and test errors for δ-margin loss.
$\widehat{\mathcal{L}}$, $\mathcal{L}$	Training and test errors for sigmoid cross entropy loss.
$\mathcal{G}, \mathcal{G}^{(t)} \subset \{\mathcal{X} \to \mathcal{X}\}$	Set of aggregation functions at t-th iteration.
$\mathcal{B}, \mathcal{B}^{(t)} \subset \{\mathcal{X} \to \mathcal{Y}\}$	Set of transformation functions at t-th iteration.
$X^{(t)}$	Collection of representations on nodes at t-th iteration.
$\widehat{Y}^{(t)}$	Collection of predictions at the t-th iteration.
$\mathcal{F}^{(t)}$	Hypothesis space of weak learners at t-th iteration
$f^{(t)}$	Weak learner at t-th iteration.
α_t, β_t	Parameters for weak learning condition at t-th iteration.
ℓ_δ	δ-margin loss (0–1 loss when $\delta = 0$.)
ℓ_σ	Sigmoid cross entropy loss.
$\mathfrak{R}(\mathcal{F})$	(Unsymmetrized) Transductive Rademacher complexity of the fuction class $\mathcal{F}$.
$\overline{\mathfrak{R}}(\mathcal{F})$	Symmetrized Transductive Rademacher complexity.
$\widehat{\mathfrak{R}}_{\text{ind}}(\mathcal{F}; Z)$	Empirical Inductive Rademacher complexity of $\mathcal{F}$ conditioned on Z.

Chapter 2

Background

In this chapter, we describe the backgrounds of the following chapters. First, we introduce Residual Network (ResNet) and its variants (Section 2.1). Next, we overview graph ML tasks (Section 2.2) and Graph Neural Networks (GNN) as algorithms for graph ML tasks (Section 2.3). Then, we introduce two subtypes of GNNs: (1) convolution-based GNNs (Section 2.5), which is a subtype of MPNN-type GNNs, and (2) GNNs with skip connections (Section 2.6), which are further divided into two types (ResNet-type GNNs and multi-scale GNNs). Finally, we explain the problem formulation of statistical learning theory for supervised (inductive) and transductive learning settings (Section 2.7).

Chapter 3 is mainly related to Section 2.1 and 2.7, Chapter 4 is mainly related to Section 2.2, 2.3, and 2.5, and Chapter 5 is mainly related to Section 2.2, 2.3, 2.6, and 2.7.

2.1 ResNet-type CNNs

In this section, we explain the architecture of ResNet and the motivation of its design. We also explain examples of ResNet-type CNNs.

2.1.1 Convolution Operation

We define a convolution operation for images, i.e., the convolution operation for tensors with spatial dimension two[1]. We can define it for other spatial dimensions analogously.

Let $K_1, K_2 \in \mathbb{N}_+$ be a filter size for the first and second spatial dimensions, respectively, $C, C' \in \mathbb{N}_+$ be the input and output channel size. The convolution operator has a parameter $W \in \mathbb{R}^{K_1 \times K_2 \times C' \times C}$ called the weight tensor. The weight tensor is also referred to as a kernel in some literature. Accordingly, the filter size is also called the kernel size. $P_1^L, P_1^R, P_2^L, P_2^R \in \mathbb{N}_{\geq 0}$ be padding parameters and $S_1, S_2 \in \mathbb{N}$ be stride parameters. Usually, we treat the weight parameter as a parameter learned through training and kernel size, padding size, and stride size as hyperparameters. Given an input $x \in \mathbb{R}^{D_1 \times D_2 \times C}$ ($D_1, D_2 \in \mathbb{N}_+$), the output y of the convolution

[1]In some literature, especially in Mathematics, this operation is referred to as *correlation* or *cross-correlation*. However, we follow the convention of ML literature and call this operation *convolution*.

operation is an order-three tensor of size $O_1 \times O_2 \times C'$ (i.e., $y \in \mathbb{R}^{O_1 \times O_2 \times C'}$) where

$$O_i = \left\lfloor \frac{D_i + P_i^L + P_i^R - K_i}{S_i} \right\rfloor + 1 \quad (i = 1, 2)$$

so that $-P_i^L + K_i + S_i(o-1) \leq D_i + P_i^R$ for $o = 1, \ldots, O_i$. The component of the output y is calculated as

$$y_{o_1 o_2 c'} = \langle x_{s_1:e_1, s_2:e_2, 1:C}, W_{::c':} \rangle \quad (c' = 1, \ldots, C'), \tag{2.1}$$

where $s_i = -P_i + 1 + S_i(o_i - 1)$ and $e_i = -P_i + K_i + S_i(o_i - 1)$ for $i = 1, 2$ (we conventionally define $x_{ijk} = 0$ when either of i, j is non-positive.) Note that $x_{s_1:e_1, s_2:e_2, 1:C}$ is a subtensor of x that has the same size as $W_{::c'}$. A typical choice of hyperparameters is $S_i = 1$ and $P_i^L = P_i^R = \lfloor \frac{K_i+1}{2} \rfloor$ for $i = 1, 2$ so that the output size is the same as the input size (i.e., $D_i = O_i$). We call this configuration the *equal-padding* setting in this paper[2].

Remark 2.1. *Although the equal-padding setting is standard, we shall employ another configuration to simplify the analysis (especially in Chapter 3). Specifically, we set $S_i = 1$, $P_i^L = 0$ and $P_i^R = K_i - 1$ so that we add the zero-padding to only one side for each spatial dimension. We call it the one-side padding. Similarly to the equal-padding setting, the one-side padding does not change the spatial dimension of input tensors. Our analysis can extend to the equal-padding case (see Section 3.3.2 Remark 3.1). In addition, we consider 1-dimensional convolution in this study, especially analysis in Chapter 3. Our analysis can generalize to multi-dimensional convolution.*

Remark 2.2. *We can define typical pooling operations analogous to the convolution operation by replacing the inner product Equation (2.1) with a parameter-free function. Take the max-pooling for example, we set the the output channel size to be the same as the input one (i.e., $C = C'$) and use the maximum value of the subtensor:*

$$y_{o_1 o_2 c} = \max_{i_1 \in [s_1:e_1], i_2 \in [s_2:e_2]} x'_{i_1 i_2 c} \quad (c = 1, \ldots, C), \tag{2.2}$$

We typically choose the hyperparameter as $S_1 = K_1$ and $S_2 = K_2$. This choice reduces the output to approximately S_i times smaller than the input. We can define the average pooling by using the mean function in place of the max function.

2.1.2 Vanishing and Exploding Gradient Problems

Vanishing and exploding gradient problems occur when a CNN has many layers, and the scale of gradients become small (vanishing) or large (exploding). We describe the problems informally in this section. Let $W^{(l)} \in \mathbb{R}^{K^{(l)} \times C^{(l+1)} \times C^{(l)}}$ and $b^{(l)} \in \mathbb{R}^{C^{(l+1)}}$ be a weight matrix and a bias vector at the l-th layer, respectively. We denote $X^{(l)} \in \mathbb{R}^{D^{(l)} \times C^{(l)}}$ be the input of the l-th. Then, the convolution operator at the l-the layer is defined as

$$X^{(l+1)} := \sigma(L^{W^{(l)}}(X^{(l)}) + \mathbf{1} \otimes b^{(l)}).$$

[2] We do not know the common name for referring to this configuration. This naming is specific to this paper.

Here, $L^{W(l)}$ is the convolution operation defined by $W^{(l)}$ (see Section 2.1.1), $\sigma : \mathbb{R} \to \mathbb{R}$ is a scalar function, which is typically non-linear. Representative examples are the sigmoid function $\sigma(z) = (1 + \exp(-z))^{-1}$ or ReLU function [Krizhevsky et al., 2012, 2017] defined by $\sigma(z) = \max(z, 0)$. Since these operations are differentiable with respect to both input $X^{(l)}$ and weight $w^{(l)}$, we can train the model in an end-to-end manner using backpropagation.

Let $\mathcal{L}$ be a loss function, then the gradient of the loss function with respect to the intermediate value $X^{(l)}$ are related as

$$\frac{\partial \mathcal{L}}{\partial X^{(l)}} = \sigma'(Z^{(l)})(L^{W^{(l)}})^\top \frac{\partial \mathcal{L}}{\partial X^{(l+1)}},$$

where $Z^{(l)} = L^{W^{(l)}}(X^{(l)}) + \mathbf{1} \otimes b^{(l)}$ and $(L^{W^{(l)}})^\top$ is treated as a linear operator from $\mathbb{R}^{D^{(l+1)} \times C^{(l+1)}}$to $\mathbb{R}^{D^{(l)} \times C^{(l)}}$. We assume that the activation function σ is 1-Lipschitz continuous. Sigmoid and ReLU functions mentioned above satisfy this assumption. Then, roughly speaking, we have

$$\left\| \frac{\partial \mathcal{L}}{\partial X^{(l)}} \right\|_{\mathrm{F}} \approx \|L^{W^{(l)}}\|_{\mathrm{op}} \left\| \frac{\partial \mathcal{L}}{\partial X^{(l+1)}} \right\|_{\mathrm{F}}.$$

The *vanishing* gradient problem is a phenomenon that the norm $\|L^{W^{(l)}}\|_{\mathrm{op}}$ is smaller than 1 for most layer l. When it happens, the gradient scale gets smaller and smaller as we go back to layers close to the input by backpropagation. Then, since prediction errors are not propagated to input layers, we cannot train their parameters using gradient-based optimization methods such as stochastic gradient descent (SGD). On the contrary, the *exploding* gradient problem is a phenomenon the norm of $\|L^{W^{(l)}}\|$ is larger than 1 for most layer l. Since the scale of layers close to inputs is significantly large, training these layers is likely to be unstable.

From these observations, we see that the norm of convolution operations should be close to 1, and the scale of gradients is approximately the same for all layers for the stable training of deep CNNs.

2.1.3 Residual Networks

ResNet [He et al., 2016] is a CNN model consisting of computation units of the form $X^{(l+1)} = F^{(l)}(X^{(l)}) + X^{(l)}$. It corresponds to adding a skip connection for bypassing the block $F^{(l)}$, called *residual block*, or *resblock* (see Figure 1.2). In the original ResNet, $F^{(l)}$ is a two-layered shallow network consisting of two 3x3 convolution operations and the non-linearity. Let $\mathcal{L}$ be a loss function. Then, by direct computation, we have

$$\frac{\partial \mathcal{L}}{\partial X^{(l)}} = \left(1 + \frac{\partial F^{(l)}}{\partial X^{(l)}}\right) \frac{\partial \mathcal{L}}{\partial X^{(l+1)}}.$$

To avoid the vanishing and exploding gradient problems, we should control the scale of gradients through the scale of $F^{(l)}$. Empirically, Zaeemzadeh et al. [2020] confirmed that the norm convolution operations of trained deep ResNets are close to 1. They have proposed a training method which promoted the norm preservation in all layers and shown that it contributed to the empirical performance of ResNets. Another empirical evidence is given by Li et al. [2018a]. They visualized the loss landscapes of CNNs with and without skip connections and showed that skip connections

Table 2.1: ResNet-type CNNs examples in Section 2.1.4.

Model	Reference
Highway Network	Srivastava et al. [2015]
ResNet	He et al. [2016]
WideResNet	Zagoruyko and Komodakis [2016]
PyramidNet	Han et al. [2017]
Inception-ResNet	Szegedy et al. [2017]
ResNeXt	Xie et al. [2017]
DenseNet	Huang et al. [2017]
SENet	Hu et al. [2018, 2020a]
U-Net	Ronneberger et al. [2015]

prevented the landscape from being chaotic. The authors hypothesized that it helped to train deep models efficiently.

2.1.4 Representative Models

Inspired by the success of ResNet, many CNN architectures with residual connections have been proposed. Table 2.1 shows representative ResNet-type CNN models we explain in this section. Since the exploration of improved ResNet architectures is an active research area, it is beyond our ability to mention all architectures to date. This table is far from a comprehensive list of ResNet-type CNNs and may miss some popular models.

Highway Network Highway Network [Srivastava et al., 2015] was a pioneering CNN model with bypass connections. Although we categorize Highway Network as an example of ResNet-type CNN, we note that Srivastava et al. [2015] proposed Highway Network before ResNet.

While ResNet directly adds the previous layer's output using skip connection, Highway Network uses the gating mechanism to "skip" computation blocks. Specifically, the l-th computation block of Highway Network has the following form:

$$\alpha^{(l)} = T^{(l)}(Z^{(l)}, W^{(l)}) \tag{2.3}$$

$$Z^{(l+1)} = (1-\alpha^{(l)})Z^{(l)} + \alpha^{(l)}F^{(l)}(Z^{(l)}). \tag{2.4}$$

Here, $T^{(l)}$ is a function called a transform gate, and $F^{(l)}$ is a function corresponding to the l-th layer of a plain CNN. Typically these functions are modeled using shallow neural networks. On the one hand, when $\alpha^{(l)}$ equals to 0, the l-th block equals to the identity function. On the other hand, when $\alpha^{(l)}$ equals 1, the l-th block reduces to one layer of usual CNNs.

WideResNet In Zagoruyko and Komodakis [2016], the authors examined the architecture of ResNet and conducted extensive studies on their architecture variants. The authors hypothesized that the excessive depth of ResNet could be harmful to performance and proposed WideResNet, which increased channels of residual blocks and decreased the model depth. They showed that WideResNet outperformed very thin and deep counterparts in several benchmark image recognition tasks.

PyramidNet The original ResNet architecture keeps the channel size (i.e., input and output channel sizes are the same) in most residual blocks and doubles the channel size at a few residual blocks. Veit et al. [2016] pointed out that these exceptional blocks are the performance bottlenecks in the sense that the removal of these blocks has more effect on performance degradation than other blocks. PyramidNet [Han et al., 2017] addresses this problem by gradually increasing channel size block by block.

Inception-ResNet and ResNeXt Inception-ResNet [Szegedy et al., 2017] and ResNeXt [Xie et al., 2017] employed a sub-network called an *inception module* as a residual block. The typical inception block feeds the same input to convolution layers with various filter sizes (e.g., three layers with filter size one, three, and five, respectively), optionally along with the max-pooling layer. To further reduce the computational cost, filter-size-one convolution (usually called 1×1 convolution) layers are added to inception blocks to reduce channel size and hence computational cost.

GoogleNet [Szegedy et al., 2015] (also known as Inception-v1), which won the ILSVRC competition in 2014, first introduced the inception blocks to approximate and sparsified dense convolutions layers intending to reduce the computational cost. This design aligned with the intuition that the inception module processes visual information at various scales and aggregates them. GoogleNet has 12 times as fewer parameters as AlexNet [Krizhevsky et al., 2012, 2017], which won the ILSVRC competition in 2012.

Inception-ResNet used the inception block as a residual block. Its architecture is roughly comparable with Inception-v4 with residual connections. ResNeXt also employed the idea of using inception blocks as a residual block. The residual block of ResNeXt has another form equivalent to the inception-style architecture using grouped convolutions [Krizhevsky et al., 2012, 2017].

DenseNet DenseNet [Huang et al., 2017] consists of serialized computational blocks called *dense block*, connected by a transition layer. DenseNet maximizes the inter-dependency of layers within a dense block for efficient information propagation. The input of a layer in a dense block is the output of all previous layers in the same block. Schematically, the computation of the l-th dense block is modelled as

$$Z^{(l+1)} = F^{(l)}(Z^{(1)}, \ldots, Z^{(l-1)}), \tag{2.5}$$

where $F^{(l)}$ is a shallow network. We can optionally add 1x1 convolutions to reduce learnable parameters, known as a bottleneck structure.

SENet ResNet adds the output of a resblock directly to the output of the previous layer. It means that the importance of each output channel of the resblock is the same. Squeeze-and-Excitation

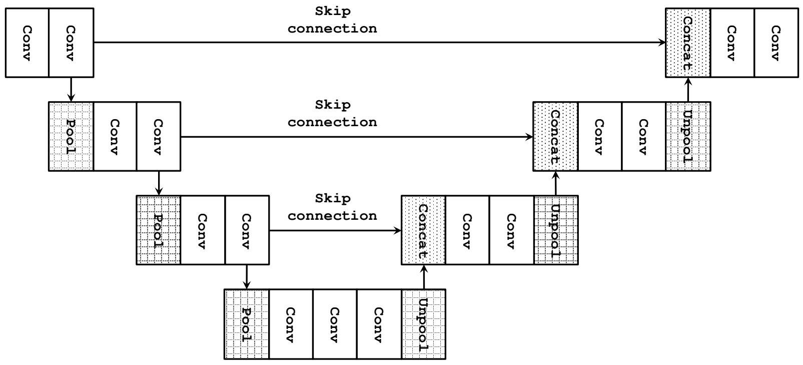


Figure 2.1: Scehmatic view of U-Net. Note that the number of stages and convolution layers are different from that of the original U-Net architecture in Ronneberger et al. [2015] for simplicity.

Network (SENet) [Hu et al., 2018, 2020a] computes the importance weight of each channel in a data dependent manner:

$$\alpha^{(l)} = G^{(l)}(Z^{(l)}), \tag{2.6}$$

$$Z^{(l+1)} = Z^{(l)} + \alpha^{(l)} \odot F^{(l)}(Z^{(l)}). \tag{2.7}$$

U-Net U-Net [Ronneberger et al., 2015] used CNNs with skip connections for tasks that require structured outputs such as segmentation tasks. Figure 2.1 shows the schematic view of U-Net. U-Net consists of an encoder and decoder, each of which has multiple stages. At stage transitions, the encoder applies a pooling operation to halves the spatial size and the decoder an unpooling operation to double the spatial size. Skip connections wire intermediate layers of the encoder and decoder at the same stage.

2.2 Machine Learning on Graphs

In this section, we introduce the basic terminologies of graphs and overview ML tasks on graphs.

2.2.1 Graph Types

In the simplest form, a *graph* is mathematically define as a pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set and $\mathcal{E}$ is a subset of $\mathcal{V} \times \mathcal{V}$. We call an element of $\mathcal{V}$ a *node* and an element of $\mathcal{E}$ an *edge*, respectively. For example, we can cast a compound's chemical formula to a graph whose nodes are atoms and edges are chemical bonds. In this study, we only consider a *finite* graph, that is, a graph with finite number of nodes: $N := |\mathcal{V}| < \infty$. An *ordering* σ of the node set $\mathcal{V}$ is a bijection $\sigma : V \to [N]$. By imposing some order on the node set $\mathcal{V}$, we can identify $\mathcal{V}$ with $[N]$.

Weighted Graph We can optionally associate each edge with a positive scalar value called *weight*. A graph with weights is called a *weighted graph*. Weights typically represent the edge's importance, cost, and length. By assuming some order on nodes, we can represent a weighted graph by the *adjacency matrix* $A \in \mathbb{R}^{N \times N}$ defined by

$$A_{ij} = \begin{cases} w((i,j)) & \text{if } (i,j) \in \mathcal{E}, \\ 0 & \text{if } (i,j) \notin \mathcal{E}, \end{cases} \tag{2.8}$$

where $w : \mathcal{E} \to \mathbb{R}_+$ is the weight function. Equivalently, we can think of w as a real-valued function on $\mathcal{V} \times \mathcal{V}$ such that $w(i,j) = 0$ if and only if $(i,j) \notin \mathcal{E}$. We can think of an unweighted graph as a weighted by assigning a constant weight (e.g., 1) to each edge.

Remark 2.3. *Weights can have non-positive value depending on the tasks. In this case, we usually require that $w((i,j)) = 0$ if and only if $(i,j) \notin \mathcal{E}$, that $w((i,j)) = 0$ and $(i,j) \notin \mathcal{E}$ have the same effect on the task. We imposed the positivity to weights due to the simplification of the presentation.*

Directed Graph A *directed graph* is a graph whose edges have a direction. The definition of a graph above is directed because, for an edge $(u,v) \in \mathcal{E}$, we can think the first element u (resp. second element v) as the head (resp. tail) of the edge (or the other way round). A graph that does not have direction is called an *undirected* graph. We can represent an undirected graph by requiring the set of edges $\mathcal{E}$ to satisfy that $(u,v) \in \mathcal{E}$ if and only if $(v,u) \in \mathcal{E}$ for any $u, v \in \mathcal{V}$. We can represent the adjacency matrix of an undirected graph with a symmetric matrix.

Multigraph A *multigraph* is a graph that allows multiple edges between the same node pair. Mathematically, a (directd) multigraph is a three-tuple $(\mathcal{V}, \mathcal{E}, \phi)$, where $\mathcal{V}$ and $\mathcal{E}$ are the set of nodes and edges, respectively, and $\phi : \mathcal{E} \to \mathcal{V} \times \mathcal{V}$ is assignment of an edge to its head and tail. We can define an undirected multigraph analogously. When ϕ is injective, we can identify $\mathcal{E}$ with a subset of $\mathcal{V} \times \mathcal{V}$. Hence, the definition of a multigraph is reduced to the original definition of a graph, a graph with no multi edges (i.e., every node pair has at most one edge). A *simple graph* is an undirected graph that does not have multi-edges. *Knowledge graph* is an example of a multigraph where two entities (nodes) can have more than one relationship (an edge with an edge type) (see Section 2.2.4).

Graph with Features When we are only interested in the topology of a graph (i.e., how nodes are connected), the pair $(\mathcal{V}, \mathcal{E})$ is sufficient as a definition of a graph. Such a graph is *homogeneous* in the sense that all nodes and edges are the same type respectively and that they themselves do not have any further information. However, in typical ML tasks, we often have more information about graphs. For example, for chemical compounds, each node (atom) has information such as an atom type, 3D coordinates, and electric charge. Similarly, each edge is associated with the bond types (e.g., single/double/triple bonds) and geometric distances. Such information is often referred to as *node* or *edge features*, respectively.

One way to represent such information is to augment a graph with functions $\phi_{\mathcal{V}} : \mathcal{V} \to \mathcal{X}_{\mathcal{V}}$ and $\phi_{\mathcal{E}} : \mathcal{E} \to \mathcal{X}_{\mathcal{E}}$, where $\mathcal{X}_{\mathcal{V}}$ and $\mathcal{X}_{\mathcal{E}}$ are the domains of node and edge features. Node and edge features can be either discrete, continuous, or their mixture. Weighted graphs are a special type of graphs with scalar edge features (i.e., $\phi_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}$).

We can compactly represent node and edge features along with the graph topology using an order-three tensor $\mathcal{A}$. Suppose the node and features are vectors of size $D_{\mathcal{V}}$ and $D_{\mathcal{E}}$, respectively (i.e., $\mathcal{X}_{\mathcal{V}} = \mathbb{R}^{D_{\mathcal{V}}}$, $\mathcal{X}_{\mathcal{E}} = \mathbb{R}^{D_{\mathcal{E}}}$). Then, $\mathcal{A}$ is a tensor of size $N \times N \times (D_{\mathcal{V}} + D_{\mathcal{E}} + 1)$ defined by

$$\mathcal{A}_{ijk} = \begin{cases} \mathbf{1}[(i,j) \in \mathcal{E}] & \text{if } k = 1, \\ \phi_{\mathcal{V}}(i) & \text{if } i = j \text{ and } k = 2, \ldots D_{\mathcal{V}} + 1, \\ \phi_{\mathcal{E}}((i,j)) & \text{if } (i,j) \in \mathcal{E} \text{ and } k = D_{\mathcal{V}} + 2, \ldots, D_{\mathcal{V}} + D_{\mathcal{E}} + 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.9}$$

In the following, we give three particular types of graphs with features: homogeneous graph with node features, heterogeneous graph, and multiplex graphs.

Homogeneous Graph with Node Features When we have node features only, we can represent the graph data as a three-tuple $(\mathcal{V}, \mathcal{E}, X)$ where $\mathcal{V}$ and $\mathcal{E}$ are the node and edge sets and $X \in \mathbb{R}^{N \times D_{\mathcal{V}}}$ is a collection of node features. Homogeneous graphs with node features are the typical input of GNNs for homogeneous graphs (see Section 2.5).

Heterogeneous Graph A *heterogeneous* graph is a graph whose node is associated with one of K node types ($K \in \mathbb{N}_+$). It corresponds to setting the domain of node features as $\mathcal{X}_{\mathcal{V}} = [K]$ or one dimension of node features takes value in $[K]$. We can partition the node set using the node type: $\mathcal{V} = \bigsqcup_{k=1}^{K} \mathcal{V}_k$ where $\mathcal{V}_k$ is the set of nodes with type $k \in [K]$. Typically, edges satisfy some constraints that they connect specific pairs of node types. The DBLP dataset used in Wang et al. [2019b] is an example of a heterogeneous graph, with four node types: 14,328 papers, 4,057 authors, 20 conferences, and 8,789 terms. Paper-author, paper-conference, and paper-term relationships are represented as edges.

Multiplex Graph A *multiplex* graph is a graph with multiple (typically finite number of) adjacency matrices. For example, consider a graph of airports in which two airports are connected at the k-th adjacency matrix if an airline company with an index k has flights between them. Then, we can represent the flight information of whole companies as a single multiplex graph. By concatenating adjacency matrices, we can represent the graph as an adjacency tensor $\mathcal{A} \in \mathbb{R}^{N \times N \times K}$. We can alternatively represent a multiplex graph with K layers as a multigraph whose edges have one of K edge types.

Remark 2.4. *Logically, a homogeneous graph with node features is a special case of a heterogeneous graph. When we use the term heterogeneous graphs, we often implicitly assume that nodes with different node types have different attribute sets.*

Remark 2.5 (Note on terminology)**.** *There is no standard notion for a graph with features and their special cases to the best of our knowledge. We have mainly followed the definition of Hamilton [2020]. Different researchers can use the same terms in different ways. In Yao and Jiliang [2020], the authors used the term a heterogeneous graph to refer to a graph with the node feature $\phi_{\mathcal{V}}$ and edge feature $\phi_{\mathcal{E}}$ (the most general concept of graphs with features in our definition). In*

Table 2.2: Classification of ML tasks on graphs

Prediction task	Graph	Node	Link
Dataset	Set of (small) graphs	(Large) graph(s)	(Large) graph(s)
Data Point	Graph	Node	Node pair
Application examples	Molecule	Social network	User–item graph
	Scene graph	Biological network	Interaction network

the document of DGL[3], the authors define heterogeneous graphs by graphs that contain different types of nodes and edges. Therefore, they also employed the definition similar to that of Yao and Jiliang [2020].

Regarding the multiplex graph, some literature called it differently. such as a multidimensional, multilayer, For example, on the one hand, Yao and Jiliang [2020] referred to this type of graphs as multidimensional graphs. Another literature calls it a multi-relational graph. However, Hamilton [2020] used a multi-relational graph to indicate whose a graph with (finite) edge types.

2.2.2 Task Types

Supervised and Semi-supervised Learning

Typical ML tasks on graphs are classified into a graph prediction task, a node prediction task, and a link prediction task, depending on what each data point is in a graph or a set of graphs (Table 2.2). In a *graph prediction* task, a dataset consists of a set of (typically small) graphs. Each graph represents one data point. The objective of graph prediction tasks is to predict the properties of whole graphs. In a *node prediction* task, we are given a single (typically large) graph or a small number of graphs. Each data point is a node in a graph (or graphs). The task is to predict nodes' properties from the information of nodes and connection information of nodes. Similar to node prediction tasks, in a *link prediction* task, we are given a single graph or a small number of graphs. Each data point is represented as a pair of nodes. The task is to predict whether given two nodes likely have an edge between them. In this study, when we consider GNN models, we mainly focus on node prediction tasks on a graph.

Graph Generation

In a graph generation task, we want to generate a graph that has desired properties. Take drug discovery as an example. We want to find candidate molecules from existing databases and, at the same time, explore molecules that are not in the databases. Table 2.3 shows representative graph generative models classified by what is a base generative model and how we represent generated graphs. There is a wider variety of strategies compared to graph encoding models explained in

[3]`https://docs.dgl.ai/en/0.4.x/index.html` (Retrieved on October 11, 2020)

Table 2.3: Graph generative models. VAE: Variational Auto Encoder, RL: Reinforcement Learning, GAN: Generative Adversarial Network, SMILES: Simplified Molecular Input Line Entry System.

Model	Reference	Base model	Output
Chemical VAE	[Gómez-Bombarelli et al., 2018]	VAE	SMILES
VGAE	[Kipf and Welling, 2016]	VAE	Adjacency matrix
Graph VAE	[Simonovsky and Komodakis, 2018]	VAE	Adjacency matrix
Graphite	[Grover et al., 2019]	VAE	Adjacency matrix
Grammar VAE	[Kusner et al., 2017]	VAE	Syntax tree
SD-VAE	[Dai et al., 2018b]	VAE	Syntax tree
JT-VAE	[Jin et al., 2018]	VAE	Molecular tree
GraphRNN	[You et al., 2018]	Autoregressive	Nodes and edges
GRAN	[Liao et al., 2019a]	Autoregressive	Adjacency matrix
ORGAN	[Guimaraes et al., 2017]	RL + GAN	SMILES
ORGANIC	[Sanchez-Lengeling et al., 2017]	RL + GAN	SMILES
MolGAN	[De Cao and Kipf, 2018]	RL + GAN	SMILES
GraphNVP	[Madhawa et al., 2019]	Invertible flow	Adjacency matrix
GRF	[Honda et al., 2019]	Invertible flow	Adjacency matrix
MoFlow	[Zang and Wang, 2020]	Invertible flow	Adjacency matrix

Section 2.3.1. Graph generation tasks must overcome the discrete and inhomogeneous nature of graphs (see Section 2.2.5). These models mainly differ in how they overcome these difficulties.

Adversarial Attack and Defense on Graphs

Adversarial attack is to add intentional small noises to inputs to mislead a model to incorrect prediction. It has been reported that classical deep models, such as FNNs and CNNs are vulnerable to adversarial attack [Szegedy et al., 2014, Goodfellow et al., 2015]. GNNs also inherit this property [Dai et al., 2018a, Zügner et al., 2018].

One characteristic of adversarial attack and defense research is the variety of problem settings, depending on applications the task assumes. Jin et al. [2020] classified adversarial attack problems based on the following axes:

1. When attacks happen (before or after training)?
2. Which components in graphs attackers modify?
3. Whether the attackers' target is specified nodes or not (targeted or non-targeted)?
4. What attackers know about the victim models (black box, gray box, or white box)?

Also, it depends on tasks on how to define the "small noise" of a graph. For example, Dai et al. [2018a] defined it as a perturbation of a graph by adding or deleting a few edges to change classification results either at the node level or graph level.

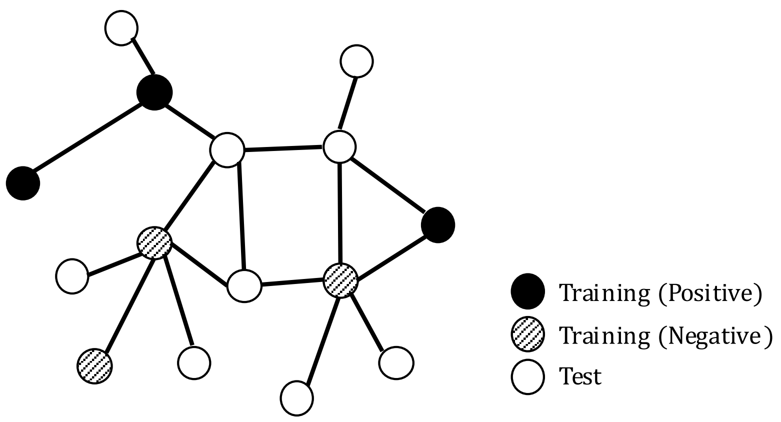


Figure 2.2: Transductive learning for a node classification task.

Similarly to the graph generation task (Section 2.2.2), attack and defence algorithms takes various strategies for overcomeing the discrete and inhomogeneous structures of graphs (Section 2.2.5). For example, Dai et al. [2018a] proposed three attack types: RL-based (Reinforcement Learning) approach (RL-S2V), gradient-based approach (GradArgmax), and evolutionary computation approach (GeneticAlg). Several strategies have been proposed regarding defense against attacks, such as adversarial training, graph purifying, and attention mechanism. See the comprehensive reviews [Sun et al., 2018, Xu et al., 2020b, Chen et al., 2020a, Jin et al., 2020] for representative attack and defense methods (e.g., Jin et al. [2020, Table 2, 5])[4]. Regarding implementation, DeepRobust [Li et al., 2020b, Xu et al., 2020a] provides a wide variety of off-the-shelf attack and defense methods on graphs.

2.2.3 Transductive Learning

When we treat a node as a data point, we can formulate a node prediction task on a graph as a *transductive learning* (Figure 2.2). Transductive learning is a type of semi-supervised learning. In a conventional supervised learning setting, the task is to build an ML model that can make predictions well to unseen (typically infinite) data points. Differently from it, in a transductive learning setting, the test sample (which typically has finite data points) is known in advance to training. The task is to make a model that predicts well on the given test sample. Therefore, we can use feature-label pairs of the training sample and feature vectors of the test sample. The ordinal supervised setting is sometimes referred to as *inductive* setting when we contrast it with a transductive setting. Similarly to the inductive setting, we have learning theory for a trusductive learning setting (see Section 2.7.6).

[4]We have retrieved the list of survey papers from the repository by Jin (`https://github.com/ChandlerBang/awesome-graph-attack-papers`) (Retrieved on October 24, 2020)

2.2.4 Application Fields

In this section, we discuss application fields of graph data analysis. Among others, we take social network analysis and chemo- and bio-informatics, and information retrieval, and computer vision as examples. We choose these fields as they have typical examples of task types (graph, node, and link prediction tasks) presented in Section 2.2.2.

Social Network Analysis

Social network analysis is a methodology in sociology in which we represent some social structures as graphs and analyze them. Since we deal with a single large graph (or a small number of graphs) are often interested in characteristics of nodes or edges in a graph(s), we often formulate ML tasks on such networks as a node prediction or link prediction task. From the viewpoint of graph theory, networks studied in social network analysis often have graph-theoretical characteristics such as scale-free-ness or small-world-ness.

Citation networks are a typical example of social networks, representing academic collaborations between researchers. A network represents a paper as a node and a citation relationship as an edge. The typical task is to predict a paper's genre from the citation relationships and word occurrence of the paper. This network type is the most standard benchmarks datasets for node classification tasks, such as Cora [McCallum et al., 2000, Sen et al., 2008], CiteSeer [Giles et al., 1998, Sen et al., 2008], and PubMed[Sen et al., 2008].

Social Networking Service (SNS) is another type of social network, in which each user attending to the service is represented as a node, and relations between users (e.g., friendship) are edges.

Chemoinformatics and Bioinformatics

The advance of measurement equipment has increased the chemistry and biological data and motivated their better use. To explain it more concretely, take drug discovery for an example. We want to know the properties of molecules that are the candidate of drug *in silico* (i.e., on computers). Such an analysis is sometimes called Quantitative Structure-Activity Relationship (QSAR). The process of drug discovery takes several steps. In the early drug discovery stage, we want to find candidates of drugs to target diseases, known as lead compounds. Such candidates are promoted to further analyses, such as clinical trials on animals or humans. High-throughput screening (HTS) is a technology for automatically assessing the *in vitro* activity of a massive number of compounds. Since this process generates a massive amount of data, it motivates the better use of such a dataset for constructing predictive models of molecular properties for selecting lead compounds within computers (thus, known as virtual screening).

Another example of a data explosion in biology is the emergence of the next-generation sequencer (NGS) for reading nucleic acid sequences. It is reported that the decrease speed of the cost of NGS has been exceeding Moore's law [5]. NGS has revolutionized the methodologies in Bioinformatics. For example, to analyze which genes are related to observable traits of living organisms

[5]K. A. Wetterstrand, The Cost of Sequencing a Human Genome, August 25, 2020 (Retrieved on October 29, 2020), `https://www.genome.gov/about-genomics/fact-sheets/Sequencing-Human-Genome-cost`

(called phenotypes), researchers classically selected a small number of genes and measured their expression levels. NGS has enabled us to measure the expression level comprehensively, leading to new methodology such as Genome-wide Association study (GWAS).

Investigation of molecules is the most standard application of the graph ML methodologies in chemistry and biology. The most standard representation of molecular graphs is to interpret each atom as a node and each covalent bond as an edge (Of course, this is not the representation of a molecule as a graph. See Section 2.2.5). In most cases, we are interested in a molecule's properties as a whole such as toxicity or solvency of a molecule. Therefore, a task on molecules is usually formulated as a graph prediction task.

Another example of networks in Biology is *interactomes*, which is a network of substances such as DNA, RNA, protein, and drugs in a body. For example, genes' expression is regularized with each other via proteins or genetic substances such as micro RNAs. Gene regulatory network is a typical example of the interactome that gives a holistic picture of gene interactions in a system. One way to represent to this network is to represent each gene as a node and an interaction between genes as an edge. Similarly, interactions between proteins are represented as a protein-protein interaction (PPI) network. Another example of gene regulatory networks is a heterogeneous graph (Section 2.2) whose nodes are genes and substances that mediate gene interactions.

An application of interactomes includes the discovery of signaling pathways and (side) effects of drugs. We can formulate such tasks as a link prediction task. For example, in Zitnik et al. [2018], the authors applied their proposed GNN model, Decagon, to protein-drug interaction networks, which are heterogeneous networks consisting of proteins and drugs to model polypharmacy side effects.

Information Retrieval

Knowledge bases are a collection of factual knowledge of interest. In natural language processing (NLP), relationships between words (technically called synonym sets or *synsets*) as a knowledge base. We can use it for various NLP tasks, such as logical reasoning, entity recognition, and sentiment analysis. For example, WordNet [Miller, 1995] is a lexical database consisting of approximately 117,000 synsets. In web science, information on the Internet is represented as a knowledge graph. Google's Knowledge Graph[6] is an example of commercialized knowledge graphs used for semantic web search.

Typically, a knowledge base represents a fact as a 3-tuple of the form (subject, predicate, object). Therefore, it is convenient that we represent knowledge base as a directed multigraph (Section 2.2), called *knowledge graph*, where nodes are subjects or objects and edges are predicates with relation types[7].

Computer Vision

Relationships between objects detected in an image are represented as a relational graph called a *scene graph* [Johnson et al., 2015, 2018]. Figure 2.3 represents a typical example of a scene graph.

[6]A. Singhal, Introducing the Knowledge Graph: things, not strings, May 16, 2012 (Retrieved on October 24, 2020) `https://blog.google/products/search/introducing-knowledge-graph-things-not/`,

[7]Schlichtkrull et al. [2018] called such a graph a *relational* graph.

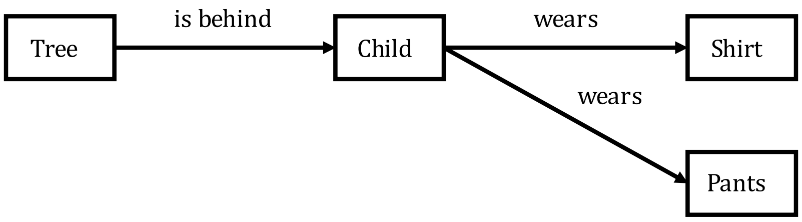


Figure 2.3: Scene graph example.

It is made from a child's picture in a shirt and pants standing in front of a tree. Scene graphs are used for image recognition tasks such as image search and caption generation.

Surfaces of 3D objects are represented as a triangulated mesh, forming a graph. We can think of polygon as a discretization of a manifold, informally speaking, a curved surface. Therefore, the application of deep models such as GNNs to such non-Euclidean data is often referred to as *geometric deep learning* [Bronstein et al., 2017].

2.2.5 Challenges

When we apply ML models to graph-structured data, we have several challenges that do not appear in grid-like data such as images or sequences, which are the main scope of classical DL models. Among others, we explain three challenges: order invariance/equivariance, inhomogeneous neighborhood structure, and data abstraction.

Order Invariance and Equivariance Roughly speaking, the *order-invariance* of a model is the characteristics that the model output is independent of the node-ordering of an input graph (Section 2.2). The concept of the order-invariance is typically used in graph prediction tasks (Section 2.2.2). Similarly, we call a model is *order-equivariant* if when we permute the node order of an input graph, the output of the model is also permuted consistently. When we discuss equivariance, we usually assume the ML model outputs a vector for each node in a graph. Therefore, we often use the concept of the order equivariance in node or edge prediction tasks.

We can formulate the invariance and equivariance of the function mathematically as follows (Figure 2.4). Let $\mathfrak{S}_N$ be the permutation group of order N. That is $\mathfrak{S}_N$ is the set of bijective function from $[N]$ to itself: $\mathfrak{S}_N := \{\sigma : [N] \to [N] \mid \sigma \text{ is bijective.}\}$. $\mathfrak{S}_N$ acts on the set of order-$(K+1)$ tensors $\mathbb{R}^{N^K \times C}$ by

$$\sigma \cdot a_{i_1 \cdots i_K c} := a_{\sigma(i_1)\cdots\sigma(i_K)c}.$$

For example, when the input tensor is a matrix (i.e., $K = 2$ and $C = 1$), the action of $\mathfrak{S}_N$ is the simultaneous permutation of rows and columns. We say a function $f : \mathbb{R}^{N^K \times C} \to \mathbb{R}^{C'}$ is *order invariant* when for any $\sigma \in \mathfrak{S}_N$, we have

$$f \circ \sigma = f.$$

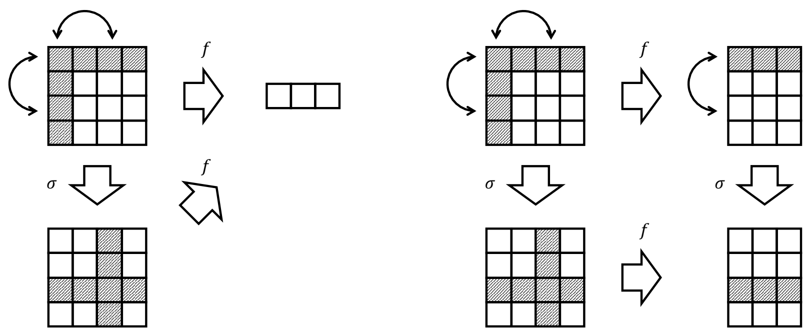


Figure 2.4: Invariance (left) and equivariance (right).

Similarly, we say a function $f : \mathbb{R}^{N^K \times C} \to \mathbb{R}^{N^{K'} \times C'}$ is *order equivariant* when for any $\sigma \in \mathfrak{S}_N$, we have

$$f \circ \sigma = \sigma \circ f.$$

Note that the order invariance is the special case of the order equivariance where $K = 0$.

A naïve idea to apply ordinal FNNs or CNNs to a graph with node features (Section 2.2.1) is to give an arbitrary order to the node-set of the graph and represent the data such as the adjacent matrix or an adjacent tensor. However, this idea fails to satisfy the requirement of the order invariance. For example, consider the graph prediction task (Section 2.2.2), where we assume that each data point is a graph of the same size N. By imposing a node order to an input graph, we can represent it is an adjacency tensor of size $M \times M \times N$ for some $M \in \mathbb{N}_+$. We can apply a 3D CNN, at least formally, to obtain the representation for a whole graph. Next, we consider another node order of the graph. We have another order-three tensor, which is different from the former tensor in general. Since the node ordering is not an intrinsic property of the graph data, we want ML models to output the same value to the two tensors made from different orders. However, we cannot ensure it because usual FNNs nor CNNs do not have such characteristics.

Inhomogeneous Neighborhood Structure Image and sequence data can be seen as a signal on a grid graph. They have a homogeneous structure that each node has the same neighborhood structures. For example, when we think of an image as a 2-dimensional grid graph whose nodes are pixels, each node has four adjacent nodes except nodes on borders. Such a homogeneous structure enables us to define the convolutional operations of a CNN (Section 2.1.1). However, arbitrary graphs do not have such structures. For example, each user in a social network has a different number of followers in general. Therefore this inhomogeneity prevents us from extending the operation of CNNs to arbitrary graphs.

Data Abstraction How to represent an object as a graph is not a trivial problem. When we represent an object as a graph, we lose some information. For example, if we represent a chemical compound as a graph whose nodes are atoms and whose edges are covalent bonds. Even if we introduce covalent bond types such as single/double/triple bonds, it loses some 3D structure information.

Although this is an important problem, we do not pursue this direction because we are mainly interested in how structures of ML models affect their theoretical performance. We assume that data points are represented as a graph using some preprocessing procedures.

2.3 GNNs Overview

In this section, we give an overview of GNN models. First, we explain the general role of GNNs, along with the list of representative models. Then, we give two unified formulations of GNNs: Message Passing Neural Network (MPNN) by Gilmer et al. [2017] and GraphNet by Battaglia et al. [2018]. Finally we briefly introduce pooling operations for GNNs.

2.3.1 Role of GNNs

Graph Neural Network, or GNN in short, pioneered by the work by Gori et al. [2005], Scarselli et al. [2009], is a collective term of DL models applied to graph-structured data. Typically, a GNN model gives representations of a whole graph or their components such as nodes and edges. By doing so, GNNs work as a (learnable) interface that converts graph-structured data to non-structured tensors and feeds them to ML models which do not assume structures (see Figure 1.3 in Section 1.3). To which components a GNN gives representations usually depends on task types we want to solve (Section 2.2.2). When we solve a graph (resp. node, edge) prediction task, typically the output of a GNN is a representation for a whole graph (resp. each node, each edge) that is order-invariant (resp. order-equivariant) (Section 2.2.5).

We can make graph and edge representations from node representations. Therefore, most GNNs are designed to produce node representations. A graph representation is obtained by aggregating all nodes' representations by some aggregation function. To make the graph representation order-invariant, we typically impose the aggregation function to be order-invariant such as the summation, aggregation, maximum, or concatenation function. Similarly, we can make edge representations by representations of end nodes. Of course, this is not the only way to make graph and edge representations. Several GNN models employ algorithms tailored to these representations.

Many GNN variants have been proposed depending on, which graph and task types (Section 2.2.2) a model deals with, and how the model overcomes the difficulties of graph ML tasks (Section 2.2.5). Table 2.4 shows the representative GNN models we feature in this paper.

2.3.2 MPNN-type GNNs

Message Passing Neural Network (MPNN) [Gilmer et al., 2017] is a unified framework of GNN models. The architecture of MPNN-type GNNs assumes that the data has *homophily*, an inductive bias that properties of neighbouring nodes are highly correlated. Therefore, their design intends that for predicting a node's properties, they make use of not only features of the node itself but also those of its neighboring nodes.

Many GNNs fall into this formulation, including GNN models proposed before, such as NFP (Neural Fingerprint) [Duvenaud et al., 2015], GGNN (Gated Graph Neural Network) [Li et al., 2016], and GCN (Graph Convolution Network) [Kipf and Welling, 2017]. The advantage of

Table 2.4: Representative GNN models.

Type	Model	Reference]
Early Model	–	[Gori et al., 2005]
	–	[Scarselli et al., 2009]
Graph Convolution	–	[Bruna et al., 2014]
(Section 2.5.2)	–	[Defferrard et al., 2016]
	NFP	[Duvenaud et al., 2015]
	GCN	[Kipf and Welling, 2017]
	SGC	[Wu et al., 2019a]
	GIN	[Xu et al., 2019]
	GAT	[Veličković et al., 2018]
ResNet-type GNN	GResNet	[Zhang, 2019]
(Section 2.6.1)	DeepGCN	[Li et al., 2019]
	DeeperGCN	[Li et al., 2020a]
	APPNP	[Klicpera et al., 2019]
	GCNII	[Chen et al., 2020b]
	CGNN	[Xhonneux et al., 2020]
Multi-scale GNN	JK-Net	[Xu et al., 2018]
(Section 2.6.2)	DCNN	[Atwood and Towsley, 2016]
	DCRNN	[Li et al., 2018c]
	MixHop	[Abu-El-Haija et al., 2019a]
	N-GCN	[Abu-El-Haija et al., 2019b]

MPNN-type GNNs is the simplicity of implementation. There are several graph libraries implementing GNNs such as PyTorch Geometric [Fey and Lenssen, 2019], DGL (Deep Graph Library) [Wang et al., 2019a], and Chainer Chemistry[8]. MPNN-type GNNs are the primary type implemented in these libraries. Another advantage is that it is computationally lighter than the method using the graph Fourier transform (see Section 2.4 for more details).

To make a node representation of a graph, an MPNN-type GNN iteratively conducts two operations: aggregation and update operations. In the aggregation operation, each node gathers information (called *message*) of neighboring nodes using the underlying graph structures. In the update operation, each node makes a new representation using its old representation and incoming messages. The general form of an L-layered MPNN-type GNN at the node i is as follows.

$$
\begin{aligned}
m_i^{(l+1)} &= \mathrm{AGGREGATE}(\{M^{(l)}(Z_i^{(l)}, Z_j^{(l)}, e_{i,j}) \mid j \in \mathcal{N}(v_i)\}), \\
Z_i^{(l+1)} &= \mathrm{UPDATE}(Z_i^{(l)}, m_i^{(l+1)}) \quad (l = 1, \ldots, L)
\end{aligned}
$$

with appropriate initialization of $h_i^{(1)}$ for each i. Here, $\mathcal{N}(i)$ is the set of nodes adjacent to i. Intuitively, the function $M^{(l)}$ incorporates the information of the neighbor node j to the node i as a message via an edge (i, j). The aggregation function $\mathrm{AGGREGATE}$ is a variadic function (i.e., a function which takes an arbitrary number of arguments) for collecting messages from neighboring nodes. Usually, we impose that the aggregate function is invariant under the permutation of any pair of arguments to obtain order-equivariant representations. In other words, $\mathrm{AGGREGATE}$ can be interpreted as a function of a multi-set. For example, the summation and mean functions are typical aggregation operations. The UPDATE function updates each node representation using the old representation $h_i^{(l)}$ and aggregated messages $m_i^{(l+1)}$. Typically, the update function is a learnable function common to all nodes, such as a multi-layer perceptron. Some models use LSTM (Long short-term memory) [Hochreiter and Schmidhuber, 1997] as an update function [Ishiguro et al., 2019]. By iterating these procedures, we obtain the node representations $Z_i^{(L+1)}$ for each node i.

We can use an MPNN-type GNN for graph prediction tasks by further aggregating node representations. This function is often called the readout function:

$$
y = \mathrm{READOUT}(\{Z_i^{(L+1)} \mid i \in \mathcal{V}\}).
$$

Similar to the aggregate function, we usually impose the readout function to be symmetric, such as the summation, mean, and maximum function.

We elaborate on a more specific MPNN-type GNNs, including GCN and its variants in Section 2.5 and 2.6.2.

2.3.3 GraphNet

GraphNet (GN) [Battaglia et al., 2018] is another unified formulation of GNNs. This formulation is different from MPNN (Section 2.3.2) in that a graph is allowed to have a global representation[9]

[8]`https://github.com/chainer/chainer-chemistry` (Retrieved on October 30, 2020)

[9]In Battaglia et al. [2018], the authors referred to the graph representation as a global *attribute*. Since we use the term graph/node/edge representations to indicate features associated with graph components, we employ this terminology for consistency.

that have the whole graph's global information. GN can utilize it to update node and edge representations. One computational unit, called a *GN block*, consists of edge, node, and global blocks. Given node representations $\{Z_i^v\}$ $(i \in \mathcal{V})$, edge representations $\{Z_k^e\}$ $(k \in \mathcal{E})$, and the global representation Z^u, the GN block makes the updated representations, denoted by $\tilde{Z}_i^v$, $\tilde{Z}_k^e$, and $\tilde{Z}^u$ respectively, as follows:

Edge Block

$$\tilde{Z}_k^e = \phi^e(Z_k^e, Z_{r_k}^v, Z_{s_k}^v, Z^u) \quad (k = (r_k, s_k) \in \mathcal{E}).$$

Node Block

$$\begin{aligned} \boldsymbol{e}_i &= \rho^{e\to v}(\{\tilde{Z}_k^e \mid k = (r_k, s_k) \in \mathcal{E}, r_k = i\}) \quad (i \in \mathcal{V}), \\ \tilde{Z}_i^v &= \phi^v(\boldsymbol{e}_i, Z_i^v, Z^u) \end{aligned}$$

Global Block

$$\begin{aligned} \boldsymbol{e} &= \rho^{e\to u}(\{\tilde{Z}_k \mid k \in \mathcal{E}\}), \\ \boldsymbol{v} &= \rho^{v\to u}(\{\tilde{Z}_i \mid i \in \mathcal{V}\}), \\ \tilde{Z}^u &= \phi^u(\boldsymbol{e}, \boldsymbol{v}, Z^u). \end{aligned}$$

Here, $\rho^{e\to v}$, $\rho^{e\to u}$, and $\rho^{v\to u}$ are aggregation functions for creating messages from edges to nodes, from edges to global, and from nodes to global, respectively. These functions are required to be order-invariant in order to make the GN block order-equivariant. ϕ^e, ϕ^v, and ϕ^u are update functions for renewing representations for edges, nodes, and global representations, respectively.

The MPNN formulation is the special case of GN, where we set ϕ^e, and ϕ^v does not depend on the global representation Z^u.

2.3.4 Pooling

Like CNNs, several works have proposed pooling operations for GNNs [Defferrard et al., 2016, Ying et al., 2018, Ma et al., 2019]. In either case, a pooling layer inputs a graph, optionally along with node and edge representations, and outputs a graph typically smaller than the original graph, associated with updated (node, edge) representations. Roughly speaking, pooling operations are categorized into two types depending on how to make the output graph: node coarsening and node sampling. On the one hand, in a node coarsening approach, a pooling operation partitions a set of nodes into groups and make a supernode per group. On the other hand, in a node sampling approach, a pooling operation selects a subset of nodes that are *informative* and discards the others.

Pooling operations are typically used for graph classification tasks since pooling operations reduce the number of nodes. By appropriating unpooling operations, we can use pooling operations for node prediction tasks, too. For example, Graph U-Net [Gao and Ji, 2019] proposed an unpooling operation (gUnpool) for restoring the original graph topology discarded by pooling operations.

2.4 Spectral Graph Theory

This section explains the spectral graph theory, particularly graph convolution operation via graph Fourier transform. Spectral graph theory studies various properties of graphs via their spectral information, such as the set of eigenpairs of adjacency matrices or graph Laplacians (Section 2.4.1). This section aims to introduce the graph convolution operator, which is a building block of convolution-based GNNs. We give a minimal explanation of the spectral graph theory needed for our purpose. Please refer to Chung and Graham [1997] for more details of the theory.

2.4.1 Definition

Let $G = (\mathcal{V}, \mathcal{E})$ be a simple graph (Section 2.2.1) with N nodes. We assume some order on the set of nodes and identify $\mathcal{V}$ with $[N]$. We denote its adjacency matrix by $A(G) \in \mathbb{R}^{N\times N}$. We define the *degree matrix* of G by

$$D(G)_{ij} = \begin{cases} \deg(i) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Here, $\deg$ is a node degree defined by $\deg(i) := |\{j \in \mathcal{V} \mid (i,j) \in \mathcal{E}\}|$. By definition, we have $D(\vec{G}) := \operatorname{diag} A\mathbf{1}$. We define the (unnormalized) graph Laplacian by $L(G) := D(G) - A(G)$. We define $D^{-\frac{1}{2}} \in \mathbb{R}^{N\times N}$ by

$$D(G)_{ij}^{-\frac{1}{2}} = \begin{cases} \deg(i)^{-\frac{1}{2}} & \text{if } i = j \text{ and } \deg(i) \neq 0 \\ 0 & \text{otherwise,} \end{cases}$$

and define the normalized adjacency matrix and the normalized Laplacian

$$\begin{aligned} \bar{A}(G) &:= D(G)^{-\frac{1}{2}} A(G) D(G)^{-\frac{1}{2}}, \\ \bar{L}(G) &:= D(G)^{-\frac{1}{2}} L(G) D(G)^{-\frac{1}{2}}. \end{aligned}$$

In particular, when G is connected, we have $\bar{L}(G) = I_N - \bar{A}(G)$. For GCN, we use the augmented version of the adjacency matrix and graph Laplacian. We define its augmented version $\tilde{G}$ by adding a single self-loop to each node for a graph G. By definition, we have $A(\tilde{G}) = A(G) + I_N$. Correspondingly, we define the *augmented normalized adjacency matrix* and *augmented normalized Laplacian* of G as the ones for augmented graph $\tilde{G}$. Specifically, the augmented normalized adjacency matrix $\tilde{A}(G)$ and augmented normalized Laplacian $\tilde{L}(G)$ is defined by

$$\begin{aligned} \tilde{A}(G) &:= \bar{A}(\tilde{G}) = (D(G) + I_N)^{-\frac{1}{2}} (A(G) + I_N)(D(G) + I_N)^{-\frac{1}{2}}, \\ \tilde{\Delta}(G) &:= \bar{\Delta}(\tilde{G}) = (D(G) + I_N)^{-\frac{1}{2}} (L(G) + I_N)(D(G) + I_N)^{-\frac{1}{2}}. \end{aligned}$$

If the underlying graph G is obvious from the context, we omit G and denote $A, \tilde{A}, \Delta$, and $\tilde{A}$, respectively for simplicity.

When G is connected, for $x \in \mathbb{R}^N$, we have

$$x^\top \bar{\Delta} x = \frac{1}{2} \sum_{i,j} a_{ij} \left(\frac{x_i}{\sqrt{d_i}} - \frac{x_j}{\sqrt{d_j}} \right)^2.$$

Note that this equation is valid for an unconneted graph G by treating the summand in the right hand side as 0 when either i or j is an isolated node. From this equation, we see that $\bar{\Delta}$ is positive semi-definite.

2.4.2 Graph Fourier Transform

We define the graph Fourier transform and inverse graph Fourier transform using the normalized adjacency matrix $\bar{A}$. We can define the graph Fourier transform using the augmented normalized adjacency matrix $\tilde{\Delta}$ analogously.

The eigenvalues of $\bar{A}$ is real because $\bar{A}$ is a symmetric matrix (since G is simple, hence undirected). Let $\lambda_1 \leq \cdots \leq \lambda_N$ be the eigenvalues of $\bar{\Delta}$, sorted by ascending order and $(e_i)_{i\in[N]}$ ($e_i \in \mathbb{R}^N$) be the basis consisting of the corresponding eigenvector associated with eigenvalues μ_i. We define the collection

$$U = \begin{bmatrix} e_1 \cdots e_N \end{bmatrix}^\top \in \mathbb{R}^{N\times N}. \tag{2.10}$$

Since $(e_i)_{i\in[N]}$ are orthonormal, U is an orthogonal matrix (i.e., $U \in O(N)$). We define the Fourier transform $\hat{\mathcal{F}} : \mathbb{R}^{N\times C} \to \mathbb{R}^{N\times C}$ and the inverse Fourier transform $\check{\mathcal{F}} : \mathbb{R}^{N\times C} \to \mathbb{R}^{N\times C}$ by $\hat{\mathcal{F}}(X) := UX$ and $\check{\mathcal{F}}(X) := U^\top X$, respectively. Since $U \in O(N)$, $\check{\mathcal{F}}$ is actually the inverse of $\hat{\mathcal{F}}$.

2.4.3 Graph Convolution Operator

The idea of spectral graph convolution is to make the filtering function learnable [Bruna et al., 2014]. For $X \in \mathbb{R}^{N\times C}$ and $W \in \mathbb{R}^N$, we define

$$\mathrm{Conv}_W(X) := \check{\mathcal{F}}(\hat{\mathcal{F}}(X) \odot W), \tag{2.11}$$

where $W = \begin{bmatrix} w_1 & \cdots & w_N \end{bmatrix}^\top$ is a matrix to be learned (Figure 2.5). If we decompose a signal $X \in \mathbb{R}^{N\times C}$ into frequency components as

$$X_{:c} = \sum_{i=1}^{N} a_{ic} e_i \tag{2.12}$$

where $a_{ic} = e_i^\top X_{:c} \in \mathbb{R}$, then, we have

$$[\mathrm{Conv}_W(X)]_{:c} := \sum_{i=1}^{N} w_i a_{ic} e_i. \tag{2.13}$$

The model proposed in Bruna et al. [2014] the cubic B-spline function to parameterize the learnable parameter W. Since Conv_W is differentiable with respect to the input X and parameter W, we can train the model in an end-to-end manner using backpropagation.

Although the operation successfully extends convolution operations from grid graphs to arbitrary ones, this model has two drawbacks. First, since the Fourier transform needs the eigenvalue decomposition to obtain the matrix U, the computational cost is prohibitive. Second, since the convolution operation has N learnable parameters, which is the same as the node size, we cannot apply it to graphs whose node is other than N.

To overcome this problem, we consider a different parametrization of the graph convolution operator. For a function $g : \mathbb{R} \to \mathbb{R}$, we define $g(\bar{A}) = U^\top g(\Lambda) U$ where $g(\Lambda) \in \mathbb{R}^{N\times N}$ is a diagonal matrix defined by $g(\Lambda)_{ii} = g(\lambda_i)$. Using the same decomposition as Equation (2.12), we have

$$[g(\bar{A})X]_{:c} = \sum_{i=1}^{N} g(\lambda_i) a_{ic} e_i. \tag{2.14}$$

Note that when g is a polynomial $g(\mu) = \sum_{k=1}^{K} b_k \mu^k$, then we have

$$g(\bar{A}) = \sum_{k=1}^{K} b_k \bar{A}^k, \tag{2.15}$$

Hence the notation the notation $g(\bar{A})$ is justified. By comparing and Equation (2.13) with Equation (2.14), we see that a filter $g(\bar{A})$ works as a convolution operator with weight $W = \begin{bmatrix} g(\lambda_1) & \cdots & g(\lambda_N) \end{bmatrix}^\top$.

This observation motivates us to parameterize the function g instead of W (i.e., $g = g_\theta$ for some parameter θ). When g is a polynomial, we can compute $g(\bar{\Delta})$ efficiently because there is no need for the eigenvalue decomposition, as we saw in Equation (2.15). Inspired by the approximation techniques of wavelet kernels by Chebyshev polynomials used in graph signal processing research [Hammond et al., 2011], Defferrard et al. [2016] used the weighted sum of Chebyshev polynomials with learnable weights.

Remark 2.6. *Since $\bar{A}$ and $\bar{\Delta}$ are related via $\bar{A} = I_N - \bar{\Delta}$, we can use a function of $\bar{\Delta}$ instead of $\bar{A}$ as a parameterized function g. This is a standard convention for GNNs using graph convolution operators (e.g., Kipf and Welling [2017]).*

2.5 Convolution-based GNNs

This section introduces the overview of convolution-based GNNs, which use (a function of) an adjacency matrix. First, we introduce a general form of convolution-based GNN and its instantiation, including GCN and its variants. Then, we explain the over-smoothing problem of linear convolution-based GNNs, which motivates the study of Chapter 4. We assume that input graphs are homogeneous graphs with node features (Section2.2) unless explicitly mentioned.

2.5.1 General Form

Figure 2.6 shows the graphical view of operations of GNNs using convolution operators as aggregation functions. Each layer has the following function form:

$$f^{(l)}(X) = F^{(l)}(g(\bar{A})X) : \mathbb{R}^{N\times C} \to \mathbb{R}^{N\times C'}. \tag{2.16}$$

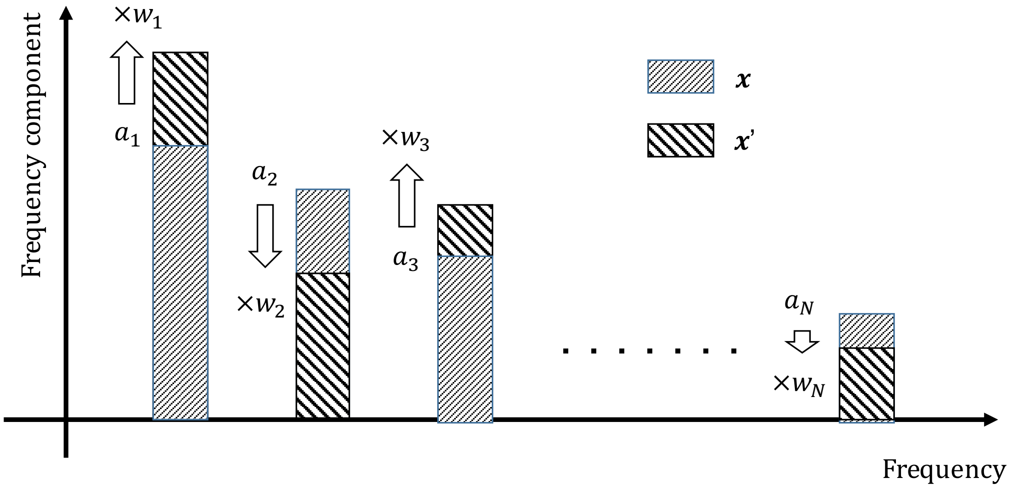


Figure 2.5: Interpretation of a graph neural network as graph signal processing. We denote the input by $x \in \mathbb{R}^N$ and the output by $x' = \mathrm{Conv}_W(x) := \mathcal{F}^{-1}(\mathcal{F}(x) \odot W) \in \mathbb{R}^N$ where $W = \begin{bmatrix} w_1 & \cdots & w_N \end{bmatrix}^\top \in \mathbb{R}^N$.

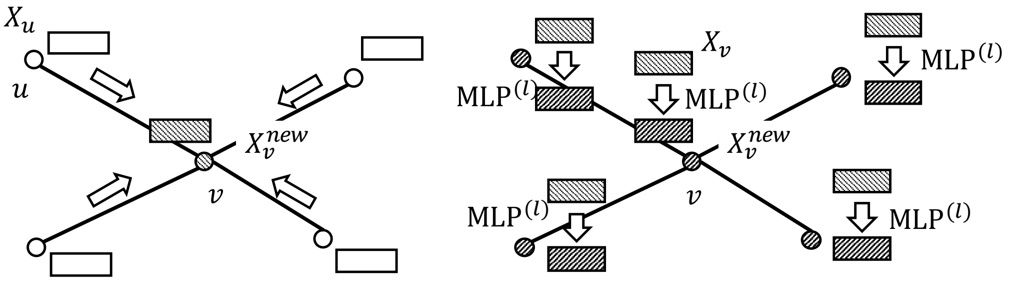


Figure 2.6: Schematic view of a convolution-based GNN $f^{(l)}(X) = \mathrm{MLP}^{(l)}(AX)$.

Here, $\bar{A}$ is the normalized adjacency matrix, $g : \mathbb{R} \to \mathbb{R}^N$ is a function, and $F^{(l)} : \mathbb{R}^C \to \mathbb{R}^{C'}$ is a function for updating node representations. We apply the same function $F^{(l)}$ to all nodes (i.e., the row-wise application to an input matrix). Both of $g^{(l)}$ and and $F^{(l)}$ can be learnable functions.

For example, GCN used the augmented normalized adjacency matrix $g(A) = \tilde{A}$ (Section 2.4.1). LanczosNet [Liao et al., 2019b] used the the polynomial of A or $\tilde{A}$ as $g(A)$. For the function $F^{(l)}$, GCN uses the learnable matrix $W^{(l)} \in \mathbb{R}^{C' \times C}$ followed by a non-linear transformation σ: $F^{(l)}(Z) = \sigma(ZW^{(l)})$. GIN (Graph Isomorphism Network) [Xu et al., 2019] extended GCN by replacing one-layer MLP with an MLP with a single hidden layer.

2.5.2 Variants

We show concrete examples of GNN models that use the graph convolution operations. In this section, we denote the collection of node representations on a graph G by $X \in \mathbb{R}^{N \times C^{(1)}}$ and the output of the l-th layer node representation by $Z^{(l)}\mathbb{R}^{N\times C^{(l)}}$ where $C^{(l)} \in \mathbb{N}_+$ is the channel size at the l-th layer.

GCN GCN (Graph Convolution Network) [Kipf and Welling, 2017] is one of the most standard GNNs. GCN iteratively computes node representations as follows:

$$\begin{aligned} Z^{(1)} &= X, \\ Z^{(l+1)} &= \sigma(\tilde{A}Z^{(l)}W^{(l)}). \end{aligned} \tag{2.17}$$

Here, $\tilde{A}$ is the augmented normalized adjacency matrix of the underlying graph and $W^{(l)} \in \mathbb{R}^{C^{(l+1)}\times C^{(l)}}$. We often omit the activation function of the last layer, in the same way as FNNs.

When a GCN is applied to a grid graph (say a two-dimensional grid graph), one layer of the GNN resembles a 3×3 convolution with fixed weight parameters, followed by a 1×1 convolution with learnable parameters and the non-linear activation function σ. This architecture is similar to the depth-wise separable convolution adopted by Xception [Chollet, 2017]. As we show in this and the following sections (Section 2.6.2), several GNN models have been proposed based on GCN's architecture.

We can think of GCN as a simplification of the model of Defferrard et al. [2016]. Despite of the simplification, GCN performed better than previous graph ML models, such as DeepWalk [Perozzi et al., 2014], or graph-based semi-supervised learning algorithms, such as Label Propagation Zhou et al. [2004], in node classification tasks on citation networks.

SGC The L-layered GCN is the following form

$$Z^{(L)} = \tilde{A}\sigma(\tilde{A}\cdots\sigma(\tilde{A}XW^{(1)})\cdots W^{(L-1)})W^{(L)}).$$

Wu et al. [2019a] questioned the role of the non-linearity σ and removed them from the model. They further concatenated all weights $W^{(l)}$, resulting in the following SGC (Simpifying Graph Convolution) model:

$$Z^{(L+1)} = \tilde{A}^L XW.$$

We introduce the non-linearity on top of it. For example, if we solve a node classification problem, we add the softmax function, which is non-linear.

GIN The original GCN employed the affine transformation plus ReLU non-linearity as the update function. GIN [Xu et al., 2019] replaced the update with an MLP applied to all nodes simultaneously:

$$\begin{aligned} Z^{(1)} &= X \\ Z^{(l+1)} &= \mathrm{MLP}^{(l)}(\tilde{A}Z^{(l)}) \end{aligned}$$

Theoretically, it is shown in Xu et al. [2019] that the expressive power of MPNN-type GNNs are bounded by 1-WL algorithm for solving the graph isomorphism problem. Using the universal approximation property of FNNs, Xu et al. [2019] also showed that GIN is approximately as powerful as the 1-WL algorithm. That is, it attains the theoretical upper bound of the expressive power of MPNN-type GNNs. In practice, the authors proposed to use an MLP with a single hidden layer.

GAT The node aggregation of a GCN multiplies the (augmented normalized) adjacency matrix to node representations (Equation (2.17)). It implicitly means that GCN sums up neighboring nodes' representations by assigning *fixed* weights to them. GAT (Graph Attention Network) [Veličković et al., 2018] determined the weights of neighboring nodes in a data-dependent manner by introducing the attention mechanism [Vaswani et al., 2017] to GNNs.

$$\begin{aligned} Z^{(l+1)} &= \sigma(A^{(l)}_{\mathrm{att}} Z^{(l)} W^{(l)}), \\ [A^{(l)}_{\mathrm{att}}]_{ij} &= \begin{cases} [\mathrm{softmax}_i(\alpha^{(l)}_i)]_j & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{if } (i,j) \notin \mathcal{E} \end{cases}, \\ \alpha^{(l)}_{ij} &= a^{(l)}(Z^{(l)}_i, Z^{(l)}_j), \end{aligned}$$

Here, softmax is a row-wise softmax function defined by

$$[\mathrm{softmax}_i(\alpha^{(l)}_i)]_j := \frac{\exp(\alpha^{(l)}_{ij})}{\sum_{k \in \mathcal{N}(i)} \exp(\alpha^{(l)}_{ik})},$$

$W^{(l)} \in \mathbb{R}^{C \times C'}$ is a learnable matrix, $a^{(l)} : \mathbb{R}^C \times \mathbb{R}^C \to \mathbb{R}$ is an learnable attention function that determines the relative importance of neighbor nodes, and $A^{(l)}_{\mathrm{att}} \in \mathbb{R}^{N \times N}$ is the resulting relative importance values. The (single-head) GAT used the following attention function:

$$a^{(l)}(z, z') := \mathrm{LeakyReLU}\left(\boldsymbol{a}^{(l)\top} \begin{bmatrix} W^{(l)}_{\mathrm{att}} z \\ W^{(l)}_{\mathrm{att}} z' \end{bmatrix} \right),$$

where $\boldsymbol{a}^{(l)} \in \mathbb{R}^{2C'}$ and $W^{(l)}_{\mathrm{att}} \in \mathbb{R}^{C' \times C}$ are learnable weight matrices, and $\mathrm{LeakyReLU} : \mathbb{R} \to \mathbb{R}$ is the LeakyReLU activation function defined by

$$\mathrm{LeakyReLU}(b) := \begin{cases} b & \text{if } b \geq 0, \\ -kb & \text{if } b < 0, \end{cases}$$

for some hyperparameter $k > 0$, applied in an element-wise manner.

Similar to the original paper [Vaswani et al., 2017], GAT employed multiple attention functions in one layer and concatenated their outputs (called the *multi-head* attention mechanism). While GAT concatenates all heads of multi-head attention, GaAN (Gated Attention Network) [Zhang et al., 2018] adaptively computed each head's weight and summed them up.

Several researches apply GNNs with attention mechanism with more complex graphs such as relational and heterogeneous graphs (Section 2.5.4) [Busbridge et al., 2019, Wang et al., 2019b]. The attention mechanism of GAT is *local* in the sense that the aggregation operation pulls representations from direct neighborhoods. Yun et al. [2019] and Choi et al. [2020] have introduced transformer-like global attention mechanisms to GNNs.

2.5.3 Over-smoothing of Linear GNNs

Although many GNN models perform well practically, it has been known that some of them do not perform well when we stack many layers to them. For example, in the paper of GCN [Gilmer et al., 2017], the authors reported that in node classification tasks on standard citation networks datasets (see Section 2.2.2), the prediction performance of GCNs degrades significantly when their depth increases to ten layers (See Appendix B of Gilmer et al. [2017]). Li et al. [2018b] visually showed that when we apply a GCN to the Karate dataset [Zachary, 1977] consisting of 34 nodes and 78 edges, the set of node representations shrank when we put more than three layers.

We can explain the over-smoothing of linear GNNs, that is, when their activation functions are the identity function, using the convergence of matrix powers. Li et al. [2018b], Zhang [2019], Zhao and Akoglu [2020] did a similar analysis. Let $P \in \mathbb{R}^{N\times N}$ be a symmetric matrix whose eigenvalues λ_i ($i \in [N]$) satisfies $-1 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{N-1} < \lambda_N = 1$. For example, the augmented normalized adjacency matrix (Section 2.4.1) satisfies this assumption. The (usual) normalized adjacency matrix also satisfies the assumption when the graph is not connected and does not have a bipartite component. Let $\mu := \min_{i\in[N-1]} |\lambda_i| (= \max(|\lambda_1|, |\lambda_{N-1}|))$ and e be the eigenvector associated with the largest eigenvalue λ_N, which is unique up to its sign. Then, for any $v \in \mathbb{R}^N$, we have

$$\|P^L v - ae\|_2 \leq e^{\mu L}\|v - ae\|_2 \tag{2.18}$$

where $a = v^\top e$. When P is the normalized adjacency matrix, e is proportional to half the degree vector: $e_i = \deg(i)^{1/2}$. Therefore, for two nodes i and j that have the same node degree, we have $v_i = v_j (= \deg(i)^{1/2})$. That means the initial vector v exponentially converges to a vector from which we cannot distinguish nodes with the same degree.

Intuitively, the aggregation function (corresponding to the multiplication by P) smoothens node representations and make them close to each other until node representations go indistinguishable when we repeat the function application. In Chapter 4, we consider the effect of non-linearity activation function σ and show that in the case of ReLU, the non-linearity does not help to mitigate the over-smoothing phenomena in the above sense.

2.5.4 GCN variants for Heterogeneous Graphs

Although we mainly focus on GNN models that take homogeneous graphs with node features. Several research, especially in the field of information retrieval (Section 2.2), extended the plain GCN model so that it can apply to heterogeneous graphs. Table 2.5 shows representative heterogeneous GNN models (Section 2.2). See also the repository[10] by Ji that curates resources about the representation learning on heterogeneous graphs.

2.6 GNNs with Skip Connections

This section explains the two types of GNN models with skip connections. The first one is a ResNet-type GNNs, in which skip connects middle layers. The second one is a multi-scale GNNs,

[10] `https://github.com/Jhy1993/Representation-Learning-on-Heterogeneous-Graph` (Retrieved on October 30, 2020)

Table 2.5: GNN models for relational and heterogeneous graphs

Model	Reference	Base Models	Graph Type
R-GCN	[Schlichtkrull et al., 2018]	GCN	Relational
RGAT	[Busbridge et al., 2019]	GAT	Relational
GTN	[Yun et al., 2019]	GCN + Transformer	Heterogeneous
HAN	[Wang et al., 2019b]	GAT	Heterogeneous
HetGNN	[Zhang et al., 2019]	Bi-LSTM + Attention	Heterogeneous
GEM	[Liu et al., 2018]	Graph Conv. + Residual + Attention	Heterogeneous

which connects middle layers to the final layer.

2.6.1 ResNet-type GNNs

GResNet Zhang [2019] evaluated the empirical performance of GCNs with four architectural variants of skip connection patterns.

$$Z^{(l+1)} = \sigma(\tilde{A}Z^{(l)}W^{(l)} + Z^{(l)}), \qquad \text{(GResNet (naïve))}$$
$$Z^{(l+1)} = \sigma(\tilde{A}(Z^{(l)}W^{(l)} + Z^{(l)})), \qquad \text{(GResNet graph-naïve))}$$
$$Z^{(l+1)} = \sigma(\tilde{A}Z^{(l)}W^{(l)} + Z^{(1)}), \qquad \text{(GResNet (raw))}$$
$$Z^{(l+1)} = \sigma(\tilde{A}(Z^{(l)}W^{(l)} + Z^{(1)})), \qquad \text{(GResNet (graph-raw))}$$

with the node representation initialization $Z^{(1)} = X$. It is worth noting that, according to Zhang [2019], the raw-residual pattern achieved better overall performance than the naive residual patterns.

DeepGCN and DeeperGCN DeepGCN [Li et al., 2019] successfully trained GCN-based GNN with 56 layers using skip connections and dilated convolution using in point cloud semantic segmentation tasks. Since point clouds do not have graph structures *a priori*, they create a k-nearest neighbor graph based on their geometric distances. In their follow-up work [Li et al., 2020a], the authors proposed DeeperGCN, an improved architecture of DeepGCN, which had the message normalization layer and reported that its prediction performance increased as layer size increases up to 112 layers in experiments on the Open Graph Benchmark [Hu et al., 2020b] dataset.

APPNP APPNP (Approximate Personalized Propagation of Neural Predictions) [Klicpera et al., 2019] is a GNN model that aggregates representations using Personalized Page Rank [Page et al., 1999] (also known as Markov Chain with restarts):

$$\begin{aligned} Z^{(1)} &= \mathrm{MLP}(X), \\ Z^{(l+1)} &= (1-\alpha)\tilde{A}Z^{(l)} + \alpha Z^{(1)}. \end{aligned} \tag{2.19}$$

$\alpha \in [0, 1]$ is a hyperparameter corresponding to the teleportation probability of the Personalized Page Rank algorithm. One notable point of APPNP is that it separates node aggregation operations from non-linear operations and does not introduce non-linearity between node aggregations. It is different from GCN-like models, which interleave node aggregation operations $Z \mapsto \tilde{A}Z$ and non-linear operations $Z \mapsto \sigma(ZW)$.

CGNN CGNN (Continuous Graph Neural Network) [Xhonneux et al., 2020] is an extension of APPNP to a continuous-time scheme. Similarly to NODE (Neural Ordinal Differential Equation) [Chen et al., 2018c], CGCN evolves the node representation using the ordinal differential equation (ODE) with learnable parameters to update node representations. Let $Z_t \in \mathbb{R}^{N\times C}$ be a collection of node representations at time $t \in [0, T]$ $(T > 0)$. Z_t evolves by the ordinal differential equation:

$$\begin{aligned} Z_0 &= \mathrm{MLP}(X) \\ \dot{Z}_t &= -(I - \tilde{A})Z_t + Z_0 \\ &= -\tilde{\Delta}Z_t + Z_0. \end{aligned}$$

This ODE has the following analytical solution:

$$Z_t = -\tilde{\Delta}^{-1}(e^{\tilde{\Delta}t} - I)Z_0 + e^{-\tilde{\Delta}t}Z_0. \tag{2.20}$$

We can think CGNN as the continuous version of APPNP as follows[11]. For when Δt is small, by Taylor extension.

$$\begin{aligned} Z_{t+\Delta t} &\approx Z_t + \Delta t\dot{Z} \\ &= Z_t + \Delta t(-\tilde{\Delta})Z_t + Z_0 \\ &= (1 - \Delta t\tilde{\Delta})Z_t + \Delta t Z_0. \end{aligned} \tag{2.21}$$

By formally assigning $\Delta t = 1$ to Equation (2.21), we have the same formula as APPNP with the correspondence $Z^{(l+1)} \leftarrow Z_t$.

GCNII GCNII (Graph Convolutional Network via Initial residual and Identity mapping) [Chen et al., 2020b]) adds two types of residual mechanisms to node aggregations and the linear transformation common to all nodes separately. They proposed the following two variants, which differ in the order of skip connections:

$$Z^{(l+1)} = \sigma\left(((1-\alpha_l)\tilde{A}Z^{(l)} + \alpha_l Z^{(1)})((1-\beta_l)I + \beta_l W^{(l)})\right) \tag{GCNII}$$

$$Z^{(l+1)} = \sigma\left((1-\alpha_l)\tilde{A}Z^{(l)}((1-\beta_l)I + \beta_l W_1^{(l)}) + \alpha_l Z^{(l)}((1-\beta_l)I + \beta_l W_2^{(l)})\right) \tag{GCNII*}$$

with initialization $Z^{(1)} = X$. $W^{(l)}$, $W_1^{(l)}$, $W_2^{(l)}$ are learnable weight parameters and α_l, β_l are hypereparameters at the l-th layer. Similarly to GResNet (raw) and APPNP, they employed initial residual connections.

[11] In the original paper [Xhonneux et al., 2020], the authors employed another derivation.

2.6.2 Multi-scale GNNs

Multi-scale GNNs are designed to use the inductive bias of a problem that the information of subgraphs at various scales are useful for prediction by connecting outputs of intermediate layers to the final output directly using skip connections. Hence, the general form of a multi-scale GNN is as follows:

$$\begin{aligned} Z^{(l)} &= f^{(l)}(\tilde{A}, Z^{(l-1)}), \quad (l = 1, \ldots, L) \\ Z &= F(Z^{(1)}, \ldots, Z^{(L)}), \end{aligned}$$

where $f^{(l)}$ and F are learnable functions. Intuitively, $Z^{(l)}$ carries the information of subgraphs with radius l. Hence, the function F use the multi-scale information via $Z^{(l)}$.

We can think of Z as the ensemble of the sub-architectures $f^{(l)}$, especially when F is the summation function. This interpretation motivates us to adopt boosting theory to multi-scale GNNs. We analyze a particular type of multi-scale GNNs from this perspective in Chapter 5.

DCNN and DCRNN DCNN [Atwood and Towsley, 2016] is a pioneering work of multi-scale GNN, which has the following architecutre:

$$\begin{aligned} Z^{(l+1)} &= \sigma(W^{(l)} \odot PX), \\ Z &= \mathop{\Big\|}_{l=1}^{L+1} Z^{(l)} \end{aligned}$$

where $||$ is the concatenation operator along the channel axis and $P = D^{-1}A$ is the transition matrix. DCRNN [Li et al., 2018c] used the summation function instead of concatenation for aggregating middle representations:

$$\begin{aligned} Z^{(l+1)} &= \sigma(PXW^{(l)}), \\ Z &= \sum_{l=1}^{L+1} Z^{(l)}. \end{aligned}$$

JK-Net JK-Net (Jumping Knowledge Network) [Xu et al., 2018] is a first multi-scale GNN model that was designed to overcome the over-smoothing problem by aggregating the subgraph information at various scale.

$$\begin{aligned} Z^{(1)} &= X \\ Z^{(l+1)} &= \sigma(\tilde{A}Z^{(l)}W^{(l)}) \quad (l = 0, \ldots, L-1) \\ Z &= \mathrm{AGGREGATE}(Z^{(1)}, \ldots, Z^{(L+1)}). \end{aligned}$$

Here, AGGREGATE is an aggregation function such as concatenation, summation, and maximum operations.

Mixhop and N-GCN MixHop [Abu-El-Haija et al., 2019a] make use of multi-scale information at one layer. Specifically, one layer of MixHop is a concatenation of GCN-like transformation using A^k in place of A:

$$Z^{(l+1)} = \mathop{\Big\Vert}_{k=1}^{K_l} \sigma(A^j Z^{(l)} W_j^{(l)}).$$

One layer of MixHop can be think of concatenation of GCN layer with different power of an adjacency matrix. N-GCN [Abu-El-Haija et al., 2019b] is a GNN concatenation of GCNs with different power of the adjacency matrix.

Multi-scale Structure of Input Residual GNNs We can interpret that GNN models with initial residual connections, such as GResNet, APPNP, and CGCN (Section 2.6.1), has implicit multi-scale structures. To explain this, let us take APPNP as an example. The fixed point $Z^{(\infty)}$ of the update function of APPNP (Equation (2.19)) is

$$\begin{aligned} & Z^{(\infty)} = (1-\alpha)AZ^{(\infty)} + \alpha Z^{(1)} \\ \iff & Z^{(\infty)} = \alpha(I - (1-\alpha)A)^{-1} Z^{(1)}. \end{aligned}$$

Therefore, under the condition $\|A\|_{\mathrm{op}} < (1-\alpha)^{-1}$, we expand the inverse and obtain

$$Z^{(\infty)} = \sum_{l=0}^{\infty} \alpha(1-\alpha)^l A^l Z^{(1)}.$$

Therefore, we can interpret a deep APPNP as a multi-scale GNN whose mixture ratio of representations is fixed. This interpretation is one reason that motivates us to analyze multi-scale GNNs instead of ResNet-type GNNs in this study (especially, in Chapter 5).

2.7 Statistical Learning Theory

In this section, we overview the problem formulation of machine learning tasks for both inductive and transductive settings (Section 5.C.1) from the viewpoint of statistical learning theory. We explain general strategies for evaluating generalization gaps. On the way, we introduce analysis tools such as the Rademacher complexity and the covering number. We also briefly explain how to obtain a faster rate. The concept of minimax optimality is also introduced as an optimality condition of training algorithms.

2.7.1 Inductive Learning Setting

Let $\mathcal{X}$ and $\mathcal{Y}$ be measurable spaces, representing the space of features and target values, respectively. Let $N \in \mathbb{N}_+$ be the sample size and $\mathcal{D} = ((x_1, y_1), \ldots, (x_N, y_N)) \in (\mathcal{X} \times \mathcal{Y})^N$ be the training sample of size N. We assume that $\mathcal{D}$ is independently and identically sampled from some probability distribution $\mathcal{P}$ on $\mathcal{X} \times \mathcal{Y}$, which is unknown to training algorithms: $(x_i, y_i) \overset{\text{i.i.d.}}{\sim} \mathcal{P}$ (or equivalently $\mathcal{D} \sim \mathcal{P}^{\otimes N}$). For example, in Chapter 3, we assume a (unknown) true function

$f^\circ : \mathcal{X} \to \mathcal{Y}$ and the target value is generated by the formula $Y = f^\circ(X) + \xi$. Here, X is the random variable on $\mathcal{X}$, and ξ is a random noise independent of X, such as Gaussian noise with a constant variance. We shall impose some smoothness on the true function f° such as the Hölder or Barron class.

Let $\widehat{\mathcal{Y}}$ be a measurable space representing the range of ML models. We set the collection of possible models $\mathcal{F} \subset \{\mathcal{X} \to \widehat{\mathcal{Y}}\}$, called the *hypothesis space*. For example, consider an architecture (e.g., FNN, CNN, GNN) with specified architectural parameters (e.g., width, depth, channel size, and the maximum norm of parameters). We can define $\mathcal{F}$ by the set of functions that the architecture can represent by appropriately choosing learnable parameters. We can think of a training algorithm as an assignment $\mathcal{A}$ from the training dataset $\mathcal{D}$ from a function in $\mathcal{F}$, called an estimator. The training algorithm $\mathcal{A}$ can be deterministic ($\mathcal{A} : \mathcal{X} \times \mathcal{Y} \to \mathcal{F}$) or stochastic ($\mathcal{A} : (\mathcal{X} \times \mathcal{Y}) \times \Omega \to \mathcal{F}$ for some probability space Ω. The Empirical Risk Minimization (ERM) (Section 2.7.2) and Stochastic Gradient Descent (SGD) are typical examples of deterministic and stochastic training algorithms, respectively. Note that $\hat{f}$ is a random variable for the training set $\mathcal{D}$ and the algorithm $\mathcal{A}$ (if it is stochastic). We denote the training algorithm's output as $\hat{f}$ and do not write the dependence on them for simplicity.

An inductive learning task aims to find an estimator $\hat{f}$ that can make an accurate prediction to unseen data points. Typically, we define a loss function $\ell : \widehat{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}$ and evaluate how bad a predictor $f : \mathcal{X} \to \widehat{\mathcal{Y}}$ is using the test error $\mathcal{R}_\ell$ defined by

$$\mathcal{R}_\ell(f)(= \mathcal{R}_\ell(f, \mathcal{P})) := \mathbb{E}_{(x,y)\sim\mathcal{P}}[\ell(f(x), y)].$$

When the loss function ℓ is obvious, we omit the subscript ℓ and write as $\mathcal{R}_\ell = \mathcal{R}$.

2.7.2 ERM estimator

The Empirical Risk Minimization (ERM) estimator is one of the most standard estimators. We cannot directly minimize the test error $\mathcal{R}$ because the underlying distribution $\mathcal{P}$ is unknown. Instead, we use a quantity computable from the training set as a proxy of the test error. We define *training error* $\mathcal{R}_\ell$ of the loss function ℓ by

$$\widehat{\mathcal{R}}_\ell(f) = \frac{1}{N}\sum_{n=1}^{N} \ell(f(x_n), y_n),$$

where, again $f : \mathcal{X} \to \widehat{\mathcal{Y}}$. The ERM estimator is the function that minimizes the training error:

$$\hat{f} \in \inf_{f\in\mathcal{F}} \widehat{\mathcal{R}}_\ell(f). \tag{2.22}$$

Similar to the test error, we omit the subscript ℓ if it is obvious from the context: $\mathcal{R}_\ell = \mathcal{R}$. Note that $\hat{f}(x_i)$ can depend not only on i but also j ($j \neq i$) because $\hat{f}$ can depend on j via the training sample $\mathcal{D}$. In Chapter 3, we consider the test error bound of the ERM estimator (strictly speaking, the variant of the ERM estimator whose output is clipped to a bounded range). If more than one function achieves the infimum, we pick an arbitrary one among the minimizers (either deterministically or stochastically). We can check that results of this study hold independent of the tie-breaking.

Remark 2.7. *In this study, we consider problem settings in which a function that achieves the infimum of Equation (2.22) exists. For example the following setting satisfies the assumption: $\mathcal{X}$ is a compact set, $\mathcal{F}$ is continuously parameterized by a parameter on a compact set, and the loss function ℓ is continuous with respect to the first variable.*

2.7.3 Uniform Bound via Model Complexity

Decomposition of Test Error We consider the ERM estimator (Section 2.7.2) as an estimator $\hat{f}$ in this section. The standard approach to evaluate the test error is to decompose it as follows and evaluate each term:

$$\mathcal{R}(\hat{f}) = [\mathcal{R}(\hat{f}) - \mathcal{R}(f^*)] + [\mathcal{R}(f^*) - \mathcal{R}(f^\circ)] + \mathcal{R}(f^\circ). \tag{2.23}$$

Here, f^* is the minimizer of the test error among $\mathcal{F}$, whose existence we assume for simplicity of explanation: $f^* \in \inf_{f\in\mathcal{F}} \mathcal{R}(f)$. Each term in Equation (2.23) is called the *excess risk* and *approximation error*, respectively. We can further decompose the first term as follows:

$$\begin{aligned}\mathcal{R}(\hat{f}) - \mathcal{R}(f^*) &= [\mathcal{R}(\hat{f}) - \widehat{\mathcal{R}}(\hat{f})] + [\widehat{\mathcal{R}}(\hat{f}) - \widehat{\mathcal{R}}(f^*)] + [\widehat{\mathcal{R}}(f^*) - \mathcal{R}(f^*)] \\ &\le [\mathcal{R}(\hat{f}) - \widehat{\mathcal{R}}(\hat{f})] + [\widehat{\mathcal{R}}(f^*) - \mathcal{R}(f^*)] \\ &\le 2 \sup_{f\in\mathcal{F}} |\widehat{\mathcal{R}}(f) - \mathcal{R}(f)| =: 2\Delta(\mathcal{F}). \end{aligned} \tag{2.24}$$

In the first inequality above, we used the inequality $\widehat{\mathcal{R}}(\hat{f}) \le \widehat{\mathcal{R}}(f^*)$, which comes from the fact that $\hat{f}$ is the minimizer of the training error. We assume that $\Delta(\mathcal{F})$ is a random variable (i.e., it is measurable with respect to the underlying probability distribution) in the remaining discussion of this section.

Therefore, the problem of bounding the test error is reduced to the evaluation of the quantity $\Delta(\mathcal{F})$. *Model complexity*, which measures the "size" of a hypothesis space $\mathcal{F}$, is a standard tool upper bound this quantity. In the next two paragraph, we give two examples of model complexity: the Rademacher complexity and covering number.

Rademacher Complexity We evaluate the expected value of $\Delta(\mathcal{F})$ and deviation of $\Delta(\mathcal{F})$ from the expected value, respectively. Using the technique known as *symmetrization*, we obtain the following inequality:

Fact 2.1 (Symmetrization)**.**

$$\mathbb{E}_{\mathcal{D}\sim\mathcal{P}^{\otimes N}}[\Delta(\mathcal{F})] \le 2\mathbb{E}_{\mathcal{D}\sim\mathcal{P}^{\otimes N}}\left[\sup_{f\in\mathcal{F}} \mathbb{E}_{(\sigma_i)_{i\in[N]}}\left[\frac{1}{N}\left|\sum_{i=1}^{N}\sigma_i l(f(x_i), y_i)\right|\right]\right].$$

Here, σ_i's are independent random variables defined by $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = 1/2$, which are also independent of the training dataset $\mathcal{D}$.

For notational simplicity, we denote $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$ and $z_i = (x_i, y_i)$ for $i \in [N]$. Motivated by the above inequality, we define the *(inductive) Rademacher complexity* $\overline{\mathfrak{R}}_{\mathrm{ind}}(\mathcal{G})$ for $\mathcal{G} \subset \{\mathcal{Z} \to \mathbb{R}\}$ by

$$\overline{\mathfrak{R}}_{\mathrm{ind}}(\mathcal{G}) := \mathbb{E}_{\mathcal{D} \sim \mathcal{P}^{\otimes N}} \left[\sup_{g \in \mathcal{G}} \mathbb{E}_{(\sigma_i)_{i \in [N]}} \left[\frac{1}{N} \left| \sum_{i=1}^{N} \sigma_i g(z_i) \right| \right] \right],$$

where, again σ_i's are the Rademacher variables[12]. In conclusion, the expected value of the right hand side of Equation (2.24) is bounded by the Rademacher complexity of $l \circ \mathcal{F} := \{\mathcal{Z} \ni z = (x, y) \mapsto l(f(x), y) \mid f \in \mathcal{F}\}$.

Regarding the variance of $\Delta(\mathcal{F})$, we can evaluate it using the following *uniform* law of large numbers (e.g., Giné and Nickl [2015, Theorem 3.4.5], Wainwright [2019, Chapter 4]):

Fact 2.2 (Uniform Law of Large Numbers)**.** *Suppose the loss function l is bounded to $[0, 1]$, there exists a universal constant $C > 0$ such that for any $\delta > 0$, with probability at least $1 - \delta$, we have*

$$\Delta(\mathcal{F}) \leq C \left(\overline{\mathfrak{R}}_{\mathrm{ind}}(l \circ \mathcal{F}) + \sqrt{\frac{\log(1/\delta)}{N}} \right).$$

It is often easier to evaluate the Rademacher complexity than the quantity $\Delta(\mathcal{F})$. For example, by the Talagrand's contraction lemma (e.g., Ledoux and Talagrand [2013]), when the loss function l is Lipschitz-bounded, we can compute the Rademacher complexity of $l \circ \mathcal{F}$ using that of $\mathcal{F}$. In addition, it is known that taking the convex hull of a hypothesis class does not change the Rademacher complexity's value.

Covering Number Covering number is another model complexity for measuring hypothesis spaces. Let be $(\mathcal{M}_0, d)$ a metric space. For $\mathcal{M} \subset \mathcal{M}_0$ and $\varepsilon > 0$, we define the *(external) covering number* $\mathcal{N}(\varepsilon, \mathcal{M}, d)$ by

$$\mathcal{N}(\varepsilon, \mathcal{M}, d) := \min \left\{ J \in \mathbb{N} \mid \exists g_1, \ldots, g_J \in \mathcal{M}_0 \text{ s.t. } \mathcal{M} \subset \cup_{j=1}^{J} \mathcal{B}_d(g_j, \varepsilon) \right\}.$$

Here, $\mathcal{B}_d(g, \varepsilon) := \{g' \in \mathcal{F} \mid d(g, g') \leq \varepsilon\}$ is the ε-ball of $\mathcal{M}$ centered at f with respect to d. The logarithm of the covering number is called the *metric entropy*. The technique known as the *chaining* argument, also known as Dudley's entropy integral [Dudley, 1967], relate the quantity $\Delta(\mathcal{F})$ or the Rademacher complexity $\overline{\mathfrak{R}}_{\mathrm{ind}}(l \circ \mathcal{F})$ with the covering number of $l \circ \mathcal{F}$ (see, e.g., Giné and Nickl [2015, Theorem 3.5.1], Wainwright [2019, Chapter 5]):

Fact 2.3 (Dudley's Entropy Integral)**.** *Let $\mathcal{G} \subset \{\mathcal{Z} \to \mathbb{R}\}$. Suppose $\mathcal{G}$ is countable and $0 \in \mathcal{G}$, then, there exists a universal constant $C > 0$ such that the following inequality holds:*

$$\overline{\mathfrak{R}}_{\mathrm{ind}}(\mathcal{G}) \leq \frac{C}{\sqrt{N}} \mathbb{E}_{\mathcal{D} \sim \mathcal{P}^{\otimes N}} \left[\int_{\alpha}^{\infty} \sqrt{\log \mathcal{N}(\varepsilon, \mathcal{G}, \| \cdot \|_N)} \mathrm{d}\varepsilon \right].$$

Here, we define $\|g\|_N^2 := \frac{1}{N} \sum_{i=1}^{N} g^2(z_i)$ for $g \in \mathcal{G}$.

[12] In this study, we call the *symmetrized* inductive Rademacher complexity (Definition 5.4). In Chapter 5, we consider its *unsymmetrized* variant that does not take the absolute value in the definition (see Definition 5.5).

Bias-Variance Trade-off There is a trade-off between the approximation error (bias) and the model complexity (variance), such as the Rademacher complexity and covering number. On the one hand, when $\mathcal{F}$ is large in terms of model complexity, the approximation error is small. However, the model complexity is large. This situation corresponds to over-fitting. On the other hand, when $\mathcal{F}$ is small, the converse is true, that is, the under-fitting occurs. Therefore, to get the best test error bound, we need to control the size of $\mathcal{F}$ appropriately to balance the approximation error and model complexity. When $\mathcal{F}$ is realized by DL models, it corresponds to determining architectural parameters of the models.

2.7.4 Fast Rate

Although we can obtain test error bounds by the procedure in the previous section (Section 2.7.3), these bounds sometimes fail to be optimal, for example, in the sense of the minimax optimality we explain later (Section 2.7.5). There are several strategies for obtaining tighter test error bounds, which is often called the *fast* rate.

The *local* Rademacher complexity [Mendelson, 2002, Bartlett et al., 2005, Koltchinskii, 2006] is the most standard approach of deriving fast rates. We have evaluated the difference of training and test errors uniformly for all $f \in \mathcal{F}$ in Equation (2.24). In that sense, the bound is *uniform* for the hypothesis class. This evaluation could make the bound loose. The idea of local Rademacher complexity is that when an estimator is likely to be near the optimal one (e.g., the Bayes classifier in the classification problem), we can effectively shrink the hypothesis class by putting higher weights to hypotheses close to the optimal.

For a more specfic situation, we can use a more direct approach. For example, Schmidt-Hieber [2020] derived a fast test error bound using Bernstein's inequality when the loss function is the squared loss and the noise is the Gaussian distribution. We employ this strategy in Chapter 3 to derive bounds for ResNet-type CNNs.

2.7.5 Minimax Optimality

We have seen a general strategy for deriving test error bounds. The natural question is whether the obtained bounds are optimal in some sense. *minimax optimality* gives a standard criterion of optimality. Informally, the minimax optimal rate is the best test error bound achieved in the worst situation. We are given a collection $\mathfrak{P}$ of probabilities on $\mathcal{X} \times \mathcal{Y}$ from which the true data distribution is drawn. Recall that we define a test error $\mathcal{R}(\hat{f}, \mathcal{P})$ for a true distribution $\mathcal{P} \in \mathfrak{P}$ and an estimator $\hat{f}$ (Section 2.7.1). We say a function $\phi : \mathbb{N}_+ \to \mathbb{R}$ is the *(asymptotically) minimax optimal rate* if

$$\sup_{\mathcal{P}\in\mathfrak{P}} \inf_{\hat{f}} \mathcal{R}(\hat{f}, \mathcal{P}) = \Theta(\phi(N)). \tag{2.25}$$

Here, $\hat{f} = \hat{f}(\mathcal{D}) : \mathcal{X} \to \widehat{\mathcal{Y}}$ runs all (mesurable) estimators made by N i.i.d. data points drawn from $\mathcal{P}$[13].

[13]We can define minimax optimality in more general settings. However, the above definition is sufficient for our purpose.

For example, consider the problem setting in Section 2.7.1. When true function f° is a D-variate β-Hölder function on a compact domain, it is known that the minimax estimation test error rate is $\phi(N) = O_P(N^{-\frac{2\beta}{2\beta+D}})$ (e.g., [Tsybakov, 2008]). Several methods, such as linear estimators and kernel methods, can achieve the minimax optimal rate in this setting. For deep learning, it is known that FNNs can achieve the minimax optimal test error rate up to logarithmic factors [Yarotsky, 2018, Schmidt-Hieber, 2020].

2.7.6 Transductive Learning Setting

In this section, we formulate a learning problem in transductive settings (Section 5.C.1). The task we consider in Chapter 5 falls into this formulation. We continue to denote the space of feature vectors, target values, and output of predictors by $\mathcal{X}$, $\mathcal{Y}$, and $\widehat{\mathcal{Y}}$, respectively.

Let N be the sample size and $\mathcal{V}$ be the sample of size N. We identify V with $[N]$. Let $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ be the feature-label pair of the data point $i \in [N]$. Differently from inductive settings (Section 2.7.1), we do not assume the underlying distribution over $\mathcal{X} \times \mathcal{Y}$ from which we assume to draw the sample. Let $\mathcal{V}_{\text{train}}, \mathcal{V}_{\text{test}} \subset \mathcal{V}$ be the partition of the sample into the training and test samples. That is, $\mathcal{V}_{\text{train}}$ and $\mathcal{V}_{\text{test}}$ satisfy $\mathcal{V}_{\text{train}} \cap \mathcal{V}_{\text{test}} = \emptyset$ and $\mathcal{V}_{\text{train}} \cup \mathcal{V}_{\text{test}} = \mathcal{V}$. We fix one partition tentatively. Later, we consider various partitions of $\mathcal{V}$ into $\mathcal{V}_{\text{train}}$ and $\mathcal{V}_{\text{test}}$.

Next, we set a hypothesis space. In a transductive learning setting, it is often the case that the model's prediction for the data point i depends not only on x_i but also the feature vector x_j for other data point j $(j \neq i)$ directly. For example, we see from the one layer transformation of a GCN (Equation (2.17)) that it updates the representation for the node i from those of nodes adjacent to i. Therefore, a model for a transductive learning is a function F as $F : \mathcal{X}^N \to \widehat{\mathcal{Y}}^N$. In addition, we assume that samples are given *a priori* and the goal of transductive learning is to complete predictions for the sample. That means we do not have to consider any point $x \in \mathcal{X}$ not included in the dataset. Therefore, considering $F : \mathcal{X}^N \to \widehat{\mathcal{Y}}^N$ is equivalent to considering $F(X) \in \widehat{\mathcal{Y}}$ where $X = (x_1, \ldots, x_N)$. In conclusion, it is appropriate to set the hypothesis space $\mathcal{F}$ as a subset of $\widehat{\mathcal{Y}}^N$.

A training algorithm $\mathcal{A}$ takes feature-label pairs of the training sample and features of the test sample and outputs the collection of predictions for the whole sample. That is, a (deterministic) training algorithm is a mapping of the form:

$$\mathcal{A} : ((x_i)_{i \in \mathcal{V}}, (y_i)_{i \in \mathcal{V}_{\text{train}}}) \mapsto \widehat{Y} \in \widehat{\mathcal{Y}}^N.$$

Similar to the inductive case, we can also consider the stochastic version of the training algorithm.

For a loss function $\ell : \widehat{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}$, we define the training error $\widehat{\mathcal{R}} = \widehat{\mathcal{R}}_\ell$ and test error $\mathcal{R} = \mathcal{R}_\ell$ by

$$\widehat{\mathcal{R}}(\widehat{Y}) = \frac{1}{M} \sum_{i \in \mathcal{V}_{\text{train}}} \ell(\hat{y}_i, y_i),$$
$$\mathcal{R}(\widehat{Y}) = \frac{1}{U} \sum_{i \in \mathcal{V}_{\text{test}}} \ell(\hat{y}_i, y_i).$$

Here, $\widehat{Y} \in \widehat{\mathcal{Y}}^N$ is an estimator, $M = |\mathcal{V}_{\text{train}}|$ is the training sample size, and $U = |\mathcal{V}_{\text{test}}|$ is the test sample size.

We want to formalize learning guarantees as upper bounds of $\mathcal{R}(\widehat{Y})$. However, since we do not assume any relationship between training and test sets so far, it is hopeless that we can obtain any meaningful guarantee. We employ the problem setting proposed in Vapnik [1982] (also employed in El-Yaniv and Pechyony [2009].) We define the generalization gap as the discrepancy between the training and test errors in terms of the random partition of the full sample into training and test datasets. More precisely, for a fixed $M \in \mathbb{N}_+$, we create a training set by uniformly randomly drawing M sample points *without* replacement from $\mathcal{V}$. We treat the remaining U sample points as a test set. We think of training and test errors as random variables with respect to the random partition of $\mathcal{V}$. Sampling without replacement causes dependency between the training and test samples. El-Yaniv and Pechyony [2009] derived the generalization gap for this setting using the concentration inequalities for the sampling without replacement. Similarly to the inductive case, we can obtain faster rates using the localized version of transductive Rademacher complexity [Tolstikhin et al., 2014].

Chapter 3

Approximation and Non-parametric Estimation Analysis of ResNet-type Convolutional Neural Networks

In this chapter, we explore the role of skip connections in Convolutional Neural Networks (CNNs) from the viewpoint of statistical learning theory. CNNs have been shown to achieve optimal approximation and estimation error rates (in minimax sense) in several function classes. However, previous analyzed optimal CNNs are unrealistically wide and difficult to obtain via optimization due to sparse constraints in important function classes, including the Hölder class. We show a ResNet-type CNN can attain the minimax optimal error rates in these classes in more plausible situations – it can be dense, and its width, channel size, and filter size are constant with respect to sample size. The key idea is that we can replicate the learning ability of Fully-connected Neural Networks (FNNs) by tailored CNNs, as long as the FNNs have *block-sparse* structures. Our theory is general in a sense that we can automatically translate any approximation rate achieved by block-sparse FNNs into that by CNNs. As an application, we derive approximation and estimation error rates of the aformentioned type of CNNs for the Barron and Hölder classes with the same strategy.

3.1 Introduction

Convolutional Neural Network (CNN) is one of the most popular architectures in deep learning research, with various applications such as computer vision [Krizhevsky et al., 2012, 2017], natural language processing [Wu et al., 2016], and sequence analysis in bioinformatics [Alipanahi et al., 2015, Zhou and Troyanskaya, 2015]. Despite practical popularity, theoretical justification for the power of CNNs is still scarce from the viewpoint of statistical learning theory.

For fully-connected Neural Networks (FNNs), there is a lot of existing work, dating back to the 80's, for theoretical explanation regarding their *approximation* ability [Cybenko, 1989, Barron, 1993, Lu et al., 2017, Yarotsky, 2017, Lee et al., 2017, Petersen and Voigtlaender, 2018b] and *generalization* power [Barron, 1994, Arora et al., 2018, Suzuki, 2018]. See also surveys of earlier work by Pinkus [2005] and Kainen et al. [2013]. Although less common compared to

FNNs, recently, statistical learning theories for CNNs have been studied both about approximation ability [Zhou, 2018, Yarotsky, 2018, Petersen and Voigtlaender, 2018a] and generalization power [Zhou and Feng, 2018]. Among others, Petersen and Voigtlaender [2018a] showed any function realizable by an FNN is representable with an (equivariant) CNN that has the same order of parameters. This fact means virtually any approximation and estimation error rates achieved by FNNs can be achieved by CNNs, too. In particular, because FNNs are optimal in minimax sense [Tsybakov, 2008, Giné and Nickl, 2015] for several important function classes such as the Hölder class [Yarotsky, 2017, Schmidt-Hieber, 2020], CNNs are also minimax optimal for these classes.

However, the optimal CNN obtained by the result of [Petersen and Voigtlaender, 2018b] can be unrealistically *wide*: for D variate β-Hölder case (see Definition 3.4), its depth is $O(\log N)$, while its channel size is as large as $O(N^{\frac{D}{2\beta+D}})$ where N is sample size. To the best of our knowledge, no CNNs that achieve the minimax optimal rate in important function classes, including the Hölder class, can keep the number of units per layer constant with respect to N. Thanks to recent techniques such as skip connections [He et al., 2016, Huang et al., 2018], sophisticated initialization schemes [He et al., 2015, Chen et al., 2018b], and normalization methods [Ioffe and Szegedy, 2015, Miyato et al., 2018], architectures that are considerably deep and moderate channel size and width have become feasible. Therefore, we would argue that there are growing demands for theories which can accommodate such constant-size architectures.

The other issue is impractical *sparsity* constraints imposed on neural networks. Existing literature [Schmidt-Hieber, 2020, Suzuki, 2019, Imaizumi and Fukumizu, 2019] proved the minimax optimal property of FNNs for several function classes. However, they picked an estimator from a set of functions realizable by FNNs with a given number of non-zero parameters. For example, Schmidt-Hieber [2020] constructed an optimal FNN that has depth $O(\log N)$, width $O(N^\alpha)$, and $O(N^\alpha \log N)$ non-zero parameters when the true function is D variate β-Hölder. Here, N is the sample size and $\alpha = \frac{D}{2\beta+D}$. It means the ratio of non-zero parameters (i.e., the number of non-zero parameters divided by the number of all parameters) is $\tilde{O}(N^{-\alpha})$. To obtain such neural networks, we need to consider impractical combinatorial problems such as L_0 norm optimization. Although we can obtain minimax optimal CNNs using the equivalence of CNNs and FNNs explained before, these CNNs have the same order of sparsity, too.

In this chapter, we show that ResNet-type (Residual Network) CNNs [He et al., 2016] with ReLU activation functions can achieve minimax optimal approximation and estimation error rates, even they have more plausible architectures. Specifically, the optimal CNNs can be dense and have constant width, channel size, and filter size against the sample size.

Our strategy is to emulate FNNs by constructing tailored ResNet-type CNNs in a similar spirit to Zhou [2018] and Petersen and Voigtlaender [2018a]. The unique point of our method is to pay attention to a *block-sparse* structure of an FNN, which roughly means a linear combination of multiple possibly dense FNNs. We first prove that if an FNN is block-sparse with M blocks, we can realize the FNN with a ResNet-type CNN with $O(M)$ additional parameters (Theorem 3.1). In particular, if blocks in the FNN are dense, which is often true in typical settings, the increase of parameters in number is negligible. Block-sparseness decreases the model complexity coming from the combinatorial sparsity patterns and promotes better bounds, a technique utilized in approximation and learning theories of FNNs implicitly or explicitly in previous studies [Yarotsky, 2018, Bölcskei et al., 2019].

Using this theorem, we next prove that the order of approximation rate of CNNs is the same as that of FNNs, and hence we show that the CNNs can achieve the same estimation error rate as the FNNs, without sparse structures in general (Theorem 3.2). Although our primary interest is the Hölder class, this result is general in the sense that it is not restricted to a specific function class, as long as we can approximate it using block-sparse FNNs.

To demonstrate the broad applicability of our methods, we derive approximation and estimation errors for two types of function classes with the same strategy: the Barron class (of parameter $s = 2$, see Definition 3.3) and Hölder class. We prove, as corollaries, that our CNNs can achieve the approximation error of order $\tilde{O}(M^{-\frac{D+2}{2D}})$ for the Barron class (Corollary 3.2) and $\tilde{O}(M^{-\frac{\beta}{D}})$ for the β-Hölder class (Corollary 3.4) and the estimation error of order $\tilde{O}_P(N^{-\frac{D+2}{2(D+1)}})$ for the Barron class (Corollary 3.3) and $\tilde{O}_P(N^{-\frac{2\beta}{2\beta+D}})$ for the β-Hölder class (Corollary 3.5) Here, M is the number of parameters (we used M, which is same as the number of blocks, to indicate the parameter count because it will turn out that CNNs have $\Omega(M)$ blocks for these cases), N is the sample size, and D is the input dimension. These rates are same as the ones for FNNs ever known in existing literature. An important consequence of our theory is that the ResNet-type CNN can achieve the minimax optimal estimation error (up to logarithmic factors) for the Hölder class even if it can be dense, and its width, filter size, and channel size are constant against sample size. This fact is in contrast to existing work, where optimal FNNs or CNNs are inevitably sparse and have width or channel size going to infinity as $N \to \infty$.

Finally, we prove minimax optimal CNNs can have constant-depth residual blocks for the Hölder case, if we introduce signal scaling mechanisms to CNNs (see Definition 3.5 for the definition and Theorem 3.4 for the statements).

In summary, the contributions in this chapter are as follows:

- We develop general approximation theories for CNNs via ResNet-type architectures. If we can approximate a function with a block-sparse FNN with M dense blocks, we can approximate the function with a ResNet-type CNN at the same rate, too (Theorem 3.1). The CNN is dense in general and is not assumed to have unrealistic sparse structures.

- We derive the upper bound of the estimation error of ResNet-type CNNs (Theorem 3.2). It gives a sufficient condition to obtain the same estimation error rate as that of FNNs (Corollary 3.1).

- We apply our theory to the Barron and Hölder classes and derive the approximation (Corollary 3.2 and 3.4) and estimation (Corollary3.3 and 3.5) error rates, which are identical to those for FNNs, even if the CNNs are dense and have constant width, channel size, and filter size with respect to sample size. This rate is minimax optimal for the Hölder case.

- For the Hölder case, the optimal CNNs can additionally have constant-depth residual blocks if we introduce scaling mechanism to skip connections (Theorem 3.3 and 3.4).

Table 3.1: Comparison of CNN architectures. Function type: The function type CNNs can approximate. “(Block-sparse) FNNs" means any function (blocks-sparse) FNNs can realize. Channel size: the number of channles needed to approximate a β-Hölder function with accuracy ε measured by the sup norm. Sparsity: the ratio of non-zero parameters of optimal FNNs when true function is β-Hölder (N is the sample size).

	Zhou [2018]	Petersen & Voigtlaender [2018a]	Ours
CNN type	Conventional	Conventional	ResNet
Function type	Barron ($s = 2$)	FNNs	Block-sparse FNNs
Channel size	N.A.	$\tilde{O}(\varepsilon^{-\frac{D}{\beta}})$	$O(1)$
Sparsity	N.A.	$\tilde{O}(N^{-\frac{D}{2\beta+D}})$	$O(1)$

3.2 Related Work

In Table 3.1, we highlight differences in CNN architectures between our work and work done by Zhou [2018] and by Petersen and Voigtlaender [2018a], which established approximation theories of CNNs via FNNs.

First and foremost, Zhou only considered a specific function class — the Barron class — as a target function class, although we can apply their method to any function class realizable by a 2-layered ReLU FNN (i.e., a ReLU FNN with a single hidden layer). Regarding architectures, they considered CNNs with a single channel and whose width is “linearly increasing" [Zhou, 2018] layer by layer. For regression or classification problems, it is rare to use such an architecture. Besides, since they did not bound the norm of parameters in approximating CNNs, we cannot derive the estimation error from their result.

Petersen and Voigtlaender [2018a] fully utilized a group invariance structure of underlying input spaces to construct CNNs. Such a structure makes theoretical analysis easier, especially for investigating the equivariance properties of CNNs because it enables us to incorporate mathematical tools such as group theory, Fourier analysis, and representation theory [Cohen et al., 2018]. Although their results are quite general in a sense that we can apply it to any function that can be approximated by FNNs, their assumption on group structures excludes the padding convolution layer, a popular type of convolution operations. Secondly, if we simply combine their result with the approximation result of Yarotsky [2017], the CNN which optimally approximates β-Hölder function by the accuracy ε (with respect to the sup-norm) has $\tilde{O}(\varepsilon^{-\frac{D}{\beta}})$ channels, which grows as $\varepsilon \to 0$ (D is the input dimension). Finally, the ratio of non-zero parameters of optimal CNNs is $\tilde{O}(N^{-\frac{D}{2\beta+D}})$. That means the optimal CNNs gets incredibly sparse as the sample size N increases. One of the reasons for the large channel size and sparse structure is that their construction was not aware of the sparse internal structure of approximating FNNs, which motivates us to consider special structures of FNNs, the block-sparse structure.

As opposed to these two studies, we employ padding- and ResNet-type CNNs which have multiple channels, fixed-sized filters, and constant width. Like Petersen and Voigtlaender [2018a], we can apply our result to any function, as long as FNNs to be approximated are block-sparse, including the Barron and Hölder cases. If we apply our theorem to these classes, we can show that the optimal CNNs can achieve the same approximation and estimation rates as FNNs, while they are dense and the number of channels is independent of the sample size.

Finite-width neural networks have been studied in earlier work [Lu et al., 2017, Perekrestenko et al., 2018, Fan et al., 2018]. However, they only derived approximation abilities. For finite-width networks, it is far from trivial to derive optimal estimation error rates from approximation results: if a network approximates a true function more accurately while restricting its capacity per layer, the neural network inevitably gets deeper. Then, the model complexity of networks explodes typically exponentially as their depth increases, which makes difficult to derive optimal estimation bounds. We overcome this problem by sophisticated evaluation of model complexity using parameter rescaling techniques (see Section 3.5.1).

Due to its practical success, theoretical analysis for ResNet has been explored recently [Lin and Jegelka, 2018, Lu et al., 2018, Nitanda and Suzuki, 2018, Huang et al., 2018]. From the viewpoint of statistical learning theory, Nitanda and Suzuki [2018] and Huang et al. [2018] investigated generalization power of ResNet from the perspective of boosting interpretation. However, they did not derive precise estimation error rates for concrete function classes. To the best of our knowledge, our theory is the first work to provide the estimation error rate of CNN classes that can accommodate the ResNet-type ones.

We import the approximation theories for FNNs, especially ones for the Barron and Hölder classes. Originally Barron [1993] considered the Barron class with a parameter $s = 1$ and an activation function σ satisfying $\sigma(z) \to 1$ as $z \to \infty$ and $\sigma(z) \to 0$ as $z \to -\infty$. Using this result, Lee et al. [2017] proved that the composition of n Barron functions with $s = 1$ can be approximated by an FNN with $n+1$ layers. Klusowski and Barron [2018] studied its approximation theory with $s = 2$ and proved that 2-layered ReLU FNNs with M hidden units can approximate functions of this class with the order of $\tilde{O}(M^{-\frac{D+2}{2D}})$. Yarotsky [2017] proved FNNs with S non-zero parameters can approximate D variate β-Hölder continuous functions with the order of $\tilde{O}(S^{-\frac{\beta}{D}})$. Using this bound, Schmidt-Hieber [2020] proved that the estimation error of the ERM estimator is $\tilde{O}(N^{-\frac{2\beta}{2\beta+D}})$, which is minimax optimal up to logarithmic factors (see, e.g., Tsybakov [2008]).

3.3 Problem Settings

3.3.1 Empirical Risk Minimization

We consider a regression task in this chapter. Let X be a $[-1,1]^D$-valued random variable with an unknown probability distribution $\mathcal{P}_X$ and ξ be an independent random noise drawn from the Gaussian distribution with an unknown variance σ^2 ($\sigma > 0$): $\xi \sim \mathcal{N}(0, \sigma^2)$. Let f° be an unknown deterministic function $f^\circ : [-1,1]^D \to \mathbb{R}$ (we will characterize f° rigorously later). We define a random variable Y by $Y := f^\circ(X) + \xi$. We denote the joint distribution of (X, Y) by $\mathcal{P}$. Suppose we are given a dataset $\mathcal{D} = ((x_1, y_1), \ldots, (x_N, y_N))$ independently and identically sampled from the distribution $\mathcal{P}$, we want to estimate the true function f° from $\mathcal{D}$.

We evaluate the performance of an estimator by the squared error. For a measurable function $f : [-1, 1]^D \to \mathbb{R}$, we define the *empirical error* of f by $\hat{\mathcal{R}}_{\mathcal{D}}(f) := \frac{1}{N}\sum_{n=1}^{N}(y_n - f(x_n))^2$ and the *estimation error* by $\mathcal{R}(f) := \mathbb{E}_{X,Y}\left[(f(X) - Y)^2\right]$. Given a subset $\mathcal{F}$ of measurable functions from $[-1, 1]^D$ to $\mathbb{R}$, we consider the *clipped empirical risk minimization (ERM) estimator* $\hat{f}$ of $\mathcal{F}$ that satisfies

$$\hat{f} := \mathrm{clip}[f_{\min}] \quad \text{where } f_{\min} \in \operatorname*{arg\,min}_{f\in\mathcal{F}} \hat{\mathcal{R}}_{\mathcal{D}}(\mathrm{clip}[f]).$$

Here, clip is a clipping operator defined by $\mathrm{clip}[f] := (f \vee -\|f^\circ\|_\infty) \wedge \|f^\circ\|_\infty$. For a measurable function $f : [-1, 1]^D \to \mathbb{R}$, we define the L_2-norm (weighted by $\mathcal{P}_X$) and the sup norm of f by $\|f\|_{\mathcal{L}^2(\mathcal{P}_X)} := \left(\int_{[-1,1]^D} f^2(x)\mathrm{d}\mathcal{P}_X(x)\right)^{\frac{1}{2}}$ and $\|f\|_\infty := \sup_{x\in[-1,1]^D} |f(x)|$, respectively. Let $\mathcal{L}^2(\mathcal{P}_X)$ be the set of measurable functions f such that $\|f\|_{\mathcal{L}^2(\mathcal{P}_X)} < \infty$ with the norm $\|\cdot\|_{\mathcal{L}^2(\mathcal{P}_X)}$. The task is to estimate the *approximation* error $\inf_{f\in\mathcal{F}} \|f - f^\circ\|_\infty$ and the *estimation* error of the clipped ERM estimator: $\mathcal{R}(\hat{f}) - \mathcal{R}(f^\circ)$. Note that the estimation error is a random variable with respect the choice of the training dataset $\mathcal{D}$. By the definition of $\mathcal{R}$ and the independence of X and ξ, the estimation error equals to $\|\hat{f} - f^\circ\|^2_{\mathcal{L}^2(P_X)}$.

3.3.2 Convolutional Neural Networks

In this section, we define CNNs used in this chapter. Let $K, C, C' \in \mathbb{N}_+$ be a filter size, input channel size, and output channel size, respectively. For a filter $w = (w_{n,j,i})_{n\in[K],j\in[C'],i\in[C]} \in \mathbb{R}^{K\times C'\times C}$, we define the *one-sided padding and stride-one convolution*[1] by w as an order-4 tensor $L^w_D = ((L^w_D)^{\beta,j}_{\alpha,i}) \in \mathbb{R}^{D\times D\times C'\times C}$ defined by

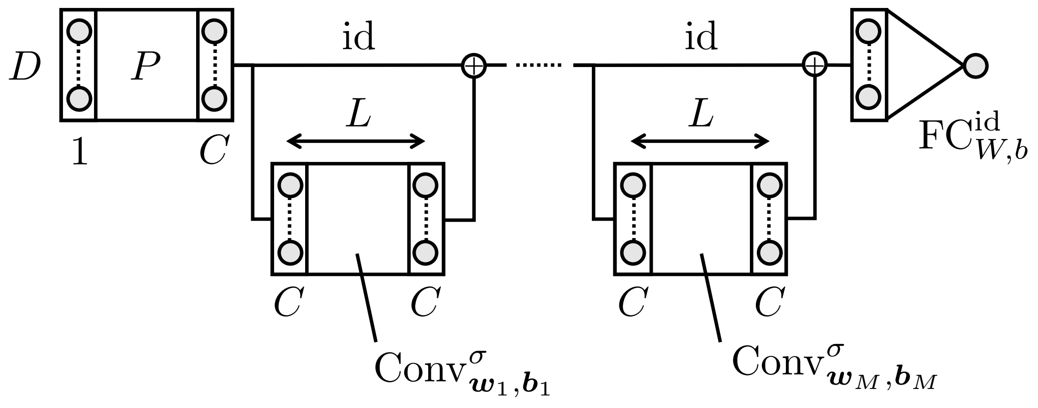
$$(L^w_D)^{\beta,j}_{\alpha,i} := \begin{cases} w_{(\alpha-\beta+1),j,i} & \text{if } 0 \le \alpha - \beta \le K - 1, \\ 0 & \text{otherwise.} \end{cases}$$

Here, i (resp. j) runs through 1 to C (resp. C') and α and β through 1 to D. Since we fix the input dimension D throughout the paper, we omit the subscript D and write as L^w if it is obvious from the context. We can interpret L^w as a linear mapping from $\mathbb{R}^{D\times C}$ to $\mathbb{R}^{D\times C'}$. Specifically, for $x = (x_{\alpha,i})_{\alpha,i} \in \mathbb{R}^{D\times C}$, we define $(y_{\beta,j})_{\beta,j} = L^w(x) \in \mathbb{R}^{D\times C'}$ by

$$y_{\beta,j} := \sum_{i,\alpha} (L^w)^{\beta,j}_{\alpha,i}\, x_{\alpha,i}.$$

Next, we define building blocks of CNNs: convolutional layers and fully-connected layers. Let $K, C, C' \in \mathbb{N}_+$. For a weight tensor $w \in \mathbb{R}^{K\times C'\times C}$, a bias vector $b \in \mathbb{R}^{C'}$, and an activation function $\sigma : \mathbb{R} \to \mathbb{R}$, we define the *convolutional layer* $\mathrm{Conv}^\sigma_{w,b} : \mathbb{R}^{D\times C} \to \mathbb{R}^{D\times C'}$ by $\mathrm{Conv}^\sigma_{w,b}(x) := \sigma(L^w(x) - \mathbf{1}_D \otimes b)$, where $\mathbf{1}_D$ is a D dimensional vector consisting of 1's, $\otimes$ is the

[1] we discuss the difference of one-sided padding and two-sided padding in Remark 3.1.



D
P
1
C
id
L
C
C
$\mathrm{Conv}^{\sigma}_{\boldsymbol{w}_1,\boldsymbol{b}_1}$
id
L
C
C
$\mathrm{Conv}^{\sigma}_{\boldsymbol{w}_M,\boldsymbol{b}_M}$
$\mathrm{FC}^{\mathrm{id}}_{W,b}$

Figure 3.1: Schematic view of a ResNet-type CNN. Variables are as in Definition 3.1.

outer product of vectors, and σ is applied in element-wise manner. Similarly, let $W \in \mathbb{R}^{C' \times DC}$, $b \in \mathbb{R}^{C'}$, and $\sigma : \mathbb{R} \to \mathbb{R}$, we define the *fully-connected layer* $\mathrm{FC}^{\sigma}_{W,b} : \mathbb{R}^{D \times C} \to \mathbb{R}^{C'}$ by $\mathrm{FC}^{\sigma}_{W,b}(a) = \sigma(W \mathrm{vec}(a) - b)$. Here, $\mathrm{vec}(\cdot)$ is the vectorization operator that flattens a matrix into a vector.

Finally, we define the ResNet-type CNN as a sequential concatenation of one convolution block, M residual blocks, and one fully-connected layer. Figure 3.1 is the schematic view of the CNN we adopt.

Definition 3.1 (Convolutional Neural Networks (CNNs))**.** *Let $M, L, C, K \in \mathbb{N}_+$, which will be the number of residual blocks and depth, channel size, and filter size of blocks, respectively. For $m \in [M]$ and $l \in [L]$, let $w^{(l)}_m \in \mathbb{R}^{K \times C \times C}$ and $b^{(l)}_m \in \mathbb{R}^C$ be a weight tensor and bias of the l-th layer of the m-th block in the convolution part, respectively. Finally, let $W \in \mathbb{R}^{DC \times 1}$ and $b \in \mathbb{R}$ be a weight matrix and a bias for the fully-connected layer part, respectively. For $\boldsymbol{\theta} := ((w^{(l)}_m)_{m,l}, (b^{(l)}_m)_{m,l}, W, b)$ and an activation function $\sigma : \mathbb{R} \to \mathbb{R}$, we define $\mathrm{CNN}^{\sigma}_{\boldsymbol{\theta}} : \mathbb{R}^D \to \mathbb{R}^D$, the CNN constructed from $\boldsymbol{\theta}$, by*

$$\mathrm{CNN}^{\sigma}_{\boldsymbol{\theta}} := \mathrm{FC}^{\mathrm{id}}_{W,b} \circ (\mathrm{Conv}^{\sigma}_{\boldsymbol{w}_M,\boldsymbol{b}_M} + \mathrm{id}) \circ \cdots \circ (\mathrm{Conv}^{\sigma}_{\boldsymbol{w}_1,\boldsymbol{b}_1} + \mathrm{id}) \circ P,$$

where $\mathrm{Conv}^{\sigma}_{\boldsymbol{w}_m,\boldsymbol{b}_m} := \mathrm{Conv}^{\sigma}_{w^{(L)}_m,b^{(L)}_m} \circ \cdots \circ \mathrm{Conv}^{\sigma}_{w^{(1)}_m,b^{(1)}_m}$, $\mathrm{id} : \mathbb{R}^{D \times C} \to \mathbb{R}^{D \times C}$ is the identity function, and $P : \mathbb{R}^D \to \mathbb{R}^{D \times C}; x \mapsto \begin{bmatrix} x & 0 & \cdots & 0 \end{bmatrix}$ is a padding operation that adds zeros to align the number of channels[2].

We say a *linear* convolutional layer or a *linear* CNN when the activation function σ is the identity function and a *ReLU* convolution layer or a *ReLU* CNN when σ is ReLU, which is defined by $\mathrm{ReLU}(x) := x \vee 0$. We call $\mathrm{Conv}^{\sigma}_{\boldsymbol{w}_m,\boldsymbol{b}_m}$ $(m > 0)$ and id in the above definition the m-th *residual block* and skip connection, respectively. We say $\boldsymbol{\theta}$ is *compatible* with (C, K) when each component of $\boldsymbol{\theta}$ satisfies the aforementioned dimension conditions.

[2]Although $\mathrm{CNN}^{\sigma}_{\boldsymbol{\theta}}$ in this definition has a fully-connected layer, we refer to a stack of convolutional layers both with or without the final fully-connect layer as a CNN.

For the number of blocks M, depth of residual blocks L, channel size C, filter size K, and norm parameters for convolution layers $B^{(\mathrm{conv})} > 0$ and for a fully-connected layer $B^{(\mathrm{fc})} > 0$, we define $\mathcal{F}^{(\mathrm{CNN})}_{M,L,C,K,B^{(\mathrm{conv})},B^{(\mathrm{fc})}}$, the hypothesis class consisting of ReLU CNNs as

$$\mathcal{F}^{(\mathrm{CNN})}_{M,L,C,K,B^{(\mathrm{conv})},B^{(\mathrm{fc})}} = \left\{ \mathrm{CNN}^{\mathrm{ReLU}}_{\boldsymbol{\theta}} \;\middle|\; \begin{array}{l} \mathrm{CNN}^{\mathrm{ReLU}}_{\boldsymbol{\theta}} \text{ has } M \text{ residual blocks,} \\ \text{depth of each residual block is } L, \\ \boldsymbol{\theta} \text{ is compatible with } (C, K), \\ \max_{m,l} \|w^{(l)}_m\|_\infty \vee \|b^{(l)}_m\|_\infty \le B^{(\mathrm{conv})}, \\ \|W\|_\infty \vee \|b\|_\infty \le B^{(\mathrm{fc})} \end{array} \right\}.$$

Here, the domain of CNNs is restricted to $[-1, 1]^D$. Note that we impose norm constraints to the convolution and fully-connected part separately. We emphasize that we do not impose any sparse constraints (e.g., restricting the number of non-zero parameters in a CNN to some fixed value) on CNNs, as opposed to previous literature [Yarotsky, 2017, Schmidt-Hieber, 2020, Imaizumi and Fukumizu, 2019].

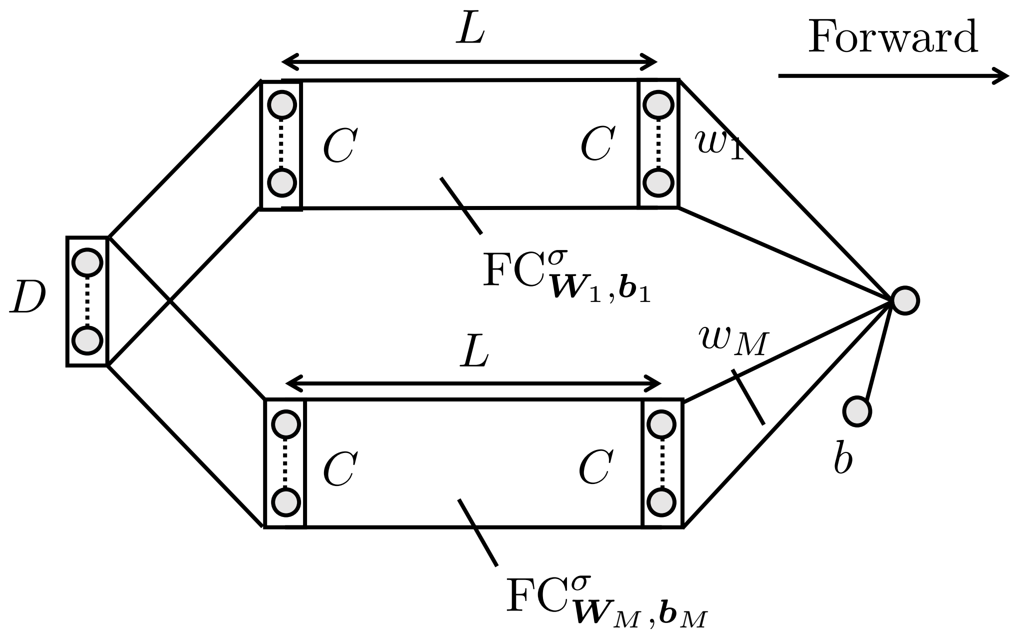
Remark 3.1 (One-sided padding vs. Equal-padding). *In this study, we adopted one-sided padding, which is not used so often practically, in order to make proofs simple. However, with slight modifications, all statements are true for equally-padded convolutions, a widely employed padding style which adds (approximately) same numbers of zeros to both ends of an input signal, with the exception that the filter size K is restricted to $K \le \lfloor \frac{D}{2} \rfloor$ instead of $K \le D$.*

Remark 3.2 (Difference between Original ResNet and Ours). *There are several differences between the CNN in this chapter and the original ResNet [He et al., 2016], aside from the number of layers. The most critical one is that our CNN does not have pooling nor Batch Normalization layers [Ioffe and Szegedy, 2015]. We will consider a scaling scheme simpler than Batch Normalization to derive optimality of CNNs with constant-depth residual blocks (see Definition 3.5). It is left for future research whether our result can extend to the ResNet-type CNNs with pooling or other scaling layers such as Batch Normalization.*

3.3.3 Block-sparse Fully-connected Neural Networks

In this section, we mathematically define FNNs we consider in this chapter, in parallel with the CNN case. Our FNN, which we coin a *block-sparse* FNN, consists of M possibly dense FNNs (blocks) concatenated in parallel, followed by a single fully-connected layer. We sketch the architecture of a block-sparse FNN in Figure 3.2.

Definition 3.2 (Fully-connected Neural Networks (FNNs)). *Let $M, L, C \in \mathbb{N}_+$ be the number of blocks in an FNN, the depth and width of blocks, respectively. Let $W^{(l)}_m \in \mathbb{R}^{C\times C}$ and $b^{(l)}_m \in \mathbb{R}^C$ be a weight matrix and a bias of the l-th layer of the m-th block for $m \in [M]$ and $l \in [L]$, with the exception that $W^{(1)}_m \in \mathbb{R}^{C\times D}$. Let $w_m \in \mathbb{R}^C$ be a weight (sub)vector of the final fully-connected layer corresponding to the m-th block and $b \in \mathbb{R}$ be a bias for the fully-connected layer. For $\boldsymbol{\theta} = ((W^{(l)}_m)_{m,l}, (b^{(l)}_m)_{m,l}, (w_m)_m, b)$ and an activation function $\sigma : \mathbb{R} \to \mathbb{R}$, we define*



L
Forward
C
C
w_1
$\mathrm{FC}^{\sigma}_{\boldsymbol{W}_1,\boldsymbol{b}_1}$
D
w_M
L
C
C
b
$\mathrm{FC}^{\sigma}_{\boldsymbol{W}_M,\boldsymbol{b}_M}$

Figure 3.2: Schematic view of a block-sparse FNN. Variables are as in Definition 3.2.

$\mathrm{FNN}^{\sigma}_{\boldsymbol{\theta}} : \mathbb{R}^D \to \mathbb{R}$, *the block-sparse FNN constructed from* $\boldsymbol{\theta}$, *by*

$$\mathrm{FNN}^{\sigma}_{\boldsymbol{\theta}} := \sum_{m=1}^{M} w_m^{\top} \mathrm{FC}^{\sigma}_{\boldsymbol{W}_m,\boldsymbol{b}_m}(\cdot) - b,$$

where $\mathrm{FC}^{\sigma}_{\boldsymbol{W}_m,\boldsymbol{b}_m} := \mathrm{FC}^{\sigma}_{W_m^{(L)},b_m^{(L)}} \circ \cdots \circ \mathrm{FC}^{\sigma}_{W_m^{(1)},b_m^{(1)}}$.

We say $\boldsymbol{\theta}$ is *compatible* with C when each component of $\boldsymbol{\theta}$ matches the dimension conditions determined by the width parameter C, as we did in the CNN case. When $L = 1$, a block-sparse FNN is a 2-layered neural network with $C' := MC$ hidden units of the form $f(x) = \sum_{c=1}^{C'} b_c \sigma(a_c^{\top} x - t_c) - b$ where $a_c \in \mathbb{R}^D$ and $b_c, t_c, b \in \mathbb{R}$.

For the number of blocks M, depth L and width C of blocks, and norm parameters for the block part $B^{(\mathrm{bs})} > 0$ and for the final layer $B^{(\mathrm{fin})} > 0$, we define $\mathcal{F}^{(\mathrm{FNN})}_{M,L,C,B^{(\mathrm{bs})},B^{(\mathrm{fin})}}$, the set of functions realizable by FNNs as

$$\mathcal{F}^{(\mathrm{FNN})}_{M,L,C,B^{(\mathrm{bs})},B^{(\mathrm{fin})}} = \left\{ \mathrm{FNN}^{\mathrm{ReLU}}_{\boldsymbol{\theta}} \;\middle|\; \begin{array}{l} \mathrm{FNN}^{\mathrm{ReLU}}_{\boldsymbol{\theta}} \text{ has } M \text{ blocks,} \\ \text{depth of each block is } L, \\ \boldsymbol{\theta} \text{ is compatible with } C, \\ \max_{m,l} \|W_m^{(l)}\|_{\infty} \vee \|b_m^{(l)}\|_{\infty} \le B^{(\mathrm{bs})}, \\ \max_m \|w_m\|_{\infty} \vee |b| \le B^{(\mathrm{fin})}. \end{array} \right\},$$

where the domain is again restricted to $[-1, 1]^D$.

3.4 Main Theorems

With the preparation in previous sections, we state our main results of this paper. We only describe statements of theorems and corollaries in the main article. All complete proofs are deferred to the supplemental material.

3.4.1 Approximation

Our first theorem claims that any block-sparse FNN with M blocks is realizable by a ResNet-type CNN with fixed-sized channels and filters by adding $O(M)$ parameters.

Theorem 3.1. *Let $M, L, C \in \mathbb{N}_+$, $K \in \{2, \ldots D\}$ and $L_0 := \left\lceil \frac{D-1}{K-1} \right\rceil$. Then, there exist $L' \leq L + L_0$, $C' \leq 4C$, and $K' \leq K$ such that, for any $B^{(\mathrm{bs})}, B^{(\mathrm{fin})} > 0$, any FNN in $\mathcal{F}^{(\mathrm{FNN})}_{M,L,C,B^{(\mathrm{bs})},B^{(\mathrm{fin})}}$ can be realized by a CNN in $\mathcal{F}^{(\mathrm{CNN})}_{M,L',C',K',B^{(\mathrm{conv})},B^{(\mathrm{fc})}}$. Here, $B^{(\mathrm{conv})} = B^{(\mathrm{bs})}$ and $B^{(\mathrm{fc})} = B^{(\mathrm{fin})}(1 \vee (B^{(bs)})^{-1})$.*

In particular, if we can approximate a function with a block-sparse FNN with $O(M)$ parameters, we can approximate the function with a ResNet-type CNN at the same rate, too. By the definition of $\mathcal{F}^{(\mathrm{CNN})}_{M,L',C',K',B^{(\mathrm{conv})}}$, the CNN emulating the block-sparse FNN is dense and does not have sparse structures in general.

3.4.2 Estimation

Our second theorem bounds the estimation error of the clipped ERM estimator. We denote $\mathcal{F}^{(\mathrm{FNN})} = \mathcal{F}^{(\mathrm{FNN})}_{M,L,C,B^{(\mathrm{bs})},B^{(\mathrm{fin})}}$ and $\mathcal{F}^{(\mathrm{CNN})} = \mathcal{F}^{(\mathrm{CNN})}_{M,L',C',K',B^{(\mathrm{conv})},B^{(\mathrm{fc})}}$ in short.

Theorem 3.2. *Let $f^\circ : \mathbb{R}^D \to \mathbb{R}$ be a measurable function and $B^{(\mathrm{bs})}, B^{(\mathrm{fin})} > 0$. Let M, L, C, K, and L_0 as in Theorem 3.1. Suppose $L', C', K', B^{(\mathrm{conv})}$ and $B^{(\mathrm{fc})}$ satisfy $\mathcal{F}^{(\mathrm{FNN})} \subset \mathcal{F}^{(\mathrm{CNN})}$ (their existence is ensured by Theorem 3.1). Suppose that the covering nubmer of $\mathcal{F}^{(\mathrm{CNN})}$ is larger than 2. Then, the clipped ERM estimator $\hat{f}$ of $\mathcal{F} := \{\mathrm{clip}[f] \mid f \in \mathcal{F}^{(\mathrm{CNN})}\}$ satisfies*

$$\mathbb{E}_{\mathcal{D}} \|\hat{f} - f^\circ\|^2_{\mathcal{L}^2(\mathcal{P}_X)} \leq C_0 \left(\inf_f \|f - f^\circ\|^2_\infty + \frac{\tilde{F}^2}{N} \Lambda_2 \log(2\Lambda_1 B N) \right). \tag{3.1}$$

Here, f ranges over $\mathcal{F}^{(\mathrm{FNN})}$, $C_0 > 0$ is a universal constant, $\tilde{F} := \frac{\|f^\circ\|_\infty}{\sigma} \vee \frac{1}{2}$, and $B := B^{(\mathrm{conv})} \vee B^{(\mathrm{fc})}$. $\Lambda_1 = \Lambda_1(\mathcal{F}^{(\mathrm{CNN})})$ and $\Lambda_2 = \Lambda_2(\mathcal{F}^{(\mathrm{CNN})})$ are defined by

$$\Lambda_1 := (2M+3)C'D(1 \vee B^{(\mathrm{fc})})(1 \vee B^{(\mathrm{conv})})\varrho\varrho^+,$$
$$\Lambda_2 := ML'\left(C'^2 K' + C'\right) + C'D + 1,$$

where $\varrho := (1+\rho)^M$, $\varrho^+ := 1 + ML'\rho^+$, $\rho := (C'K'B^{(\mathrm{conv})})^{L'}$, and $\rho^+ := (1 \vee C'K'B^{(\mathrm{conv})})^{L'}$.

The first term of (3.1) is the approximation error achieved by $\mathcal{F}^{(\mathrm{FNN})}$. On the other hand, the second term of (3.1) represents the model complexity of $\mathcal{F}^{(\mathrm{CNN})}$ since Λ_1 and Λ_2 are determined by the architectural parameters of $\mathcal{F}^{(\mathrm{CNN})}$ — Λ_1 corresponds to the Lipschitz constant of a function realized by a CNN and Λ_2 is the number of parameters, including zeros, of a CNN. There is a trade-off between these two terms. Using appropriately chosen M to balance them, we can evaluate the order of estimation error with respect to the sample size N.

Corollary 3.1. *Under the same assumptions as Theorem 3.2, suppose further* $\log \Lambda_1 B = \tilde{O}(1)$ *as a function of* M*. If* $\inf_{f \in \mathcal{F}^{(\mathrm{FNN})}} \|f - f^\circ\|_\infty^2 = \tilde{O}(M^{-\gamma_1})$ *and* $\Lambda_2 = \tilde{O}(M^{\gamma_2})$ *for some constants* $\gamma_1, \gamma_2 > 0$ *independent of* M*, then, the clipped ERM estimator* $\hat{f}$ *of* $\mathcal{F}$ *achieves the estimation error* $\|f^\circ - \hat{f}\|_{\mathcal{L}_2(\mathcal{P}_X)}^2 = \tilde{O}_P(N^{-\frac{2\gamma_1}{2\gamma_1 + \gamma_2}})$.

3.5 Applications

3.5.1 Barron Class

The Barron class is an example of the function class that can be approximated by block-sparse FNNs. We employ the definition of Barron functions used in Klusowski and Barron [2018].

Definition 3.3 (Barron class). *We call a measurable function* $f^\circ : [-1, 1]^D \to \mathbb{R}$ *a Barron function of a parameter* $s > 0$ *if* f° *admits the Fourier representation (i.e.,* $f^\circ(x) = \check{\mathcal{F}}\mathcal{F}[f^\circ]$*) and* $\int_{\mathbb{R}^D} \|w\|_2^s |\mathcal{F}[f^\circ](w)| \,\mathrm{d}w < \infty$*. Here,* $\mathcal{F}$ *and* $\check{\mathcal{F}}$ *are the Fourier and inverse Fourier transformation, respectively.*

Klusowski and Barron [2018] studied approximation of the Barron function f° with the parameter $s = 2$ by a linear combination of M ridge functions (i.e., a 2-layered ReLU FNN). Specifically, they showed that there exists a function f_M of the form

$$f_M := f^\circ(0) + \nabla f^{\circ\top}(0) x + \frac{1}{M} \sum_{m=1}^{M} b_m (a_m^\top x - t_m)_+ \tag{3.2}$$

with $|b_m| \le 1$, $\|a_m\|_1 = 1$, and $|t_m| \le 1$, such that $\|f^\circ - f_M\|_\infty = \tilde{O}(M^{-(\frac{1}{2} + \frac{1}{D})})$. Using this approximator f_M, we can derive the same approximation order using CNNs by applying Theorem 3.1 with $L = 1$ and $C = 1$.

Corollary 3.2. *Let* $f^\circ : [-1, 1]^D \to \mathbb{R}$ *be a Barron function with the parameter* $s = 2$ *such that* $f^\circ(0) = 0$ *and* $\nabla f^\circ(0) = \mathbf{0}_D$*. Then, for any* $K \in \{2, \ldots, D\}$*, there exists a CNN* $f^{(\mathrm{CNN})}$ *with* M *residual blocks, each of which has depth* $O(1)$ *and at most* 4 *channels, and whose filter size is at most* K*, such that* $\|f^\circ - f^{(\mathrm{CNN})}\|_\infty = \tilde{O}(M^{-(\frac{1}{2} + \frac{1}{D})})$.

Note that this rate is same as the one obtained for FNNs [Klusowski and Barron, 2018].

We have one design choice when we apply Corollary 3.1 in order to derive the estimation error: how to set $B^{(\mathrm{bs})}$ and $B^{(\mathrm{fin})}$? Looking at the definition of f_M, a naive choice would be $B^{(\mathrm{bs})} := 1$ and $B^{(\mathrm{fin})} := M^{-1}$. However, this cannot satisfy the assumption on Λ_1 of Corollary 3.1, due to the term $\varrho = (1+\rho)^M$. We want the logarithm of Λ_1 to be $\tilde{O}(1)$ as a function of M. In order to do that, we change the *relative scale* between parameters in the block-sparse part and the fully-connected part using the homogeneous property of the ReLU function: $\mathrm{ReLU}(ax) = a\mathrm{ReLU}(x)$ for $a > 0$. The rescaling operation enables us to choose $B^{(\mathrm{bs})} := M^{-1}$ and $B^{(\mathrm{fin})} = 1$ to meet the assumption of Corollary 3.1. By setting $\gamma_1 = \frac{1}{2} + \frac{1}{D}$ and $\gamma_2 = 1$, we obtain the desired estimation error.

Corollary 3.3. *Let $f^\circ : [-1,1]^D \to \mathbb{R}$ be a Barron function with the parameter $s = 2$ such that $f^\circ(0) = 0$ and $\nabla f^\circ(0) = \mathbf{0}_D$. Let $K \in \{2, \ldots, D\}$. There exist the number of residual blocks $M = O(N^{\frac{D}{2+2D}})$, depth of each residual block $L = O(1)$, channel size $C = O(1)$, and norm bounds $B^{(\mathrm{conv})}, B^{(\mathrm{fc})} > 0$ such that for sufficiently large N, the clipped ERM estimator $\hat{f}$ of $\{\mathrm{clip}[f] \mid f \in \mathcal{F}^{(\mathrm{CNN})}_{M,L,C,K,B^{(\mathrm{conv})},B^{(\mathrm{fc})}}\}$ achieves the estimation error $\|f^\circ - \hat{f}\|^2_{\mathcal{L}_2(\mathcal{P}_X)} = \tilde{O}_P(N^{-\frac{D+2}{2(D+1)}})$.*

3.5.2 Hölder Class

We next consider the approximation and error rates of CNNs when the true function f° is a Hölder function.

Definition 3.4 (Hölder class)**.** *Let $\beta > 0$. A function $f^\circ : [-1,1]^D \to \mathbb{R}$ is called a β-Hölder function if*

$$\|f^\circ\|_\beta := \sum_{0 \le |\alpha| < \lfloor \beta \rfloor} \|\partial^\alpha f^\circ\|_\infty + \sum_{|\alpha| = \lfloor \beta \rfloor} \sup_{x \neq y} \frac{|\partial^\alpha f^\circ(x) - \partial^\alpha f^\circ(y)|}{|x-y|^{\beta - \lfloor \beta \rfloor}} < \infty.$$

Here, $\alpha = (\alpha_1, \ldots, \alpha_D)$ is a multi-index. That is, $\partial^\alpha f := \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \cdots \partial x_D^{\alpha_D}}$ and $|\alpha| := \sum_{d=1}^D \alpha_d$.

Yarotsky [2017] showed that FNNs with S non-zero parameters can approximate any D variate β-Hölder function with the order of $\tilde{O}(S^{-\frac{\beta}{D}})$. Schmidt-Hieber [2020] also proved a similar statement using a different construction method. They only specified the width[3], depth, and non-zero parameter counts of the approximating FNN and did not write in detail how non-zero parameters are distributed in the statements explicitly (see Theorem 1 of Yarotsky [2017] and Theorem 5 of Schmidt-Hieber [2020]). However, if we carefully look at their proofs, we find that we can transform the FNNs they constructed into block-sparse ones (see Lemma 3.7 of the supplemental material). Therefore, we can apply Theorem 3.1 to these FNNs. To meet the assumption of Corollary 3.1, we again rescale the parameters of the FNNs, as we did in the Barron-class case, so that $\log \Lambda_1 = \tilde{O}(1)$. We can derive the approximation and estimation errors by setting $\gamma_1 = \frac{\beta}{D}$ and $\gamma_2 = 1$.

Corollary 3.4. *Let $\beta > 0$ and $f^\circ : [-1,1]^D \to \mathbb{R}$ be a β-Hölder function. Then, for any $K \in \{2, \ldots, D\}$, there exists a CNN $f^{(\mathrm{CNN})}$ with $O(M)$ residual blocks, each of which has depth $O(\log M)$ and $O(1)$ channels, and whose filter size is at most K, such that $\|f^\circ - f^{(\mathrm{CNN})}\|_\infty = \tilde{O}(M^{-\frac{\beta}{D}})$.*

[3]Yarotsky [2017] didn't specified the width of FNNs.

Corollary 3.5. *Let $\beta > 0$ and $f^\circ : [-1,1]^D \to \mathbb{R}$ be a β-Hölder function. For any $K \in \{2, \ldots, D\}$, there exist the number of residual blocks $M = O(N^{\frac{D}{2\beta+D}})$, depth of each residual block $L = O(\log N)$, channel size $C = O(1)$, and norm bounds $B^{(\mathrm{conv})}, B^{(\mathrm{fc})} > 0$ such that for sufficiently large N, the clipped ERM estimator $\hat{f}$ of $\{\mathrm{clip}[f] \mid f \in \mathcal{F}^{(\mathrm{CNN})}_{M,L,C,K,B^{(\mathrm{conv})},B^{(\mathrm{fc})}}\}$ achieves the estimation error $\|f^\circ - \hat{f}\|^2_{\mathcal{L}_2(\mathcal{P}_X)} = \tilde{O}_P(N^{-\frac{2\beta}{2\beta+D}})$.*

Since the estimation error rate of the β-Hölder class is $O_P(N^{-\frac{2\beta}{2\beta+D}})$ (see, e.g., Tsybakov [2008]), Corollary 3.5 implies that our CNN can achieve the minimax optimal rate up to logarithmic factors even though it can be dense and its width D, channel size C, and filter size K are constant with respect to the sample size N.

3.6 Discussion

Optimal CNNs with Constant-depth Blocks

In the previous section, we proved the optimality of dense and narrow ResNet-type CNNs for the Hölder class. However, the constructed CNN can have residual blocks whose depth is as large as $O(\log N)$. Such an architecture is different from practically successful ResNets because they usually have relatively shallow (e.g., 2- or 3-layered) networks as residual blocks. We hypothesize that the essence of the problem resides in the difference of scales between skip connections and residual blocks. Therefore, we consider another type of CNNs that admits scaling schemes of intermediate signals in order to overcome this problem. Among others, we consider the simplest scaling method, which zeros out some channels in skip connections.

Definition 3.5 (Masked CNNs)**.** *Let $M, L, C, K \in \mathbb{N}_+$. Let $w_m^{(l)} \in \mathbb{R}^{K\times C\times C}$, $b_m^{(l)} \in \mathbb{R}^C$, $W \in \mathbb{R}^{DC\times 1}$ and $b \in \mathbb{R}$ be parameters of CNNs for $m \in [M]$ and $l \in [L]$. Let $z_m = (z_{m,1}, \ldots, z_{m,C}) \in \{0,1\}^C$ be a mask for the m-th skip connection. For $\boldsymbol{\theta} := ((w_m^{(l)})_{m,l}, (b_m^{(l)})_{m,l}, W, b, (z_m)_m)$ and an activation function $\sigma : \mathbb{R} \to \mathbb{R}$, we define $\mathrm{mCNN}^\sigma_{\boldsymbol{\theta}} : \mathbb{R}^D \to \mathbb{R}^D$, the masked CNN constructed from $\boldsymbol{\theta}$, by*

$$\mathrm{mCNN}^\sigma_{\boldsymbol{\theta}} := \mathrm{FC}^{\mathrm{id}}_{W,b} \circ (\mathrm{Conv}^\sigma_{\boldsymbol{w}_M,\boldsymbol{b}_M} + J_M) \circ \cdots \circ (\mathrm{Conv}^\sigma_{\boldsymbol{w}_1,\boldsymbol{b}_1} + J_1) \circ P,$$

where $J_m : \mathbb{R}^{D\times C} \to \mathbb{R}^{D\times C}$ is a channel wise mask operation defined by $[x_1 \ \cdots \ x_C] \mapsto [z_{m,1}x_1 \ \cdots \ z_{m,C}x_C]$.

By definition, plain ResNet-type CNNs in Definition 3.1 are a special case of masked CNNs. Note that we do not restrict the number of non-zero mask elements. Therefore, although masks take discrete values, we can obtain approximated ERM estimators via sparse optimization techniques. We say $\boldsymbol{\theta}$ is compatible with (C, K) when $\boldsymbol{\theta}$ satisfies the dimension conditions as we did in Definition 3.1. We define $\mathcal{G}_{M,L,C,K,B^{(\mathrm{conv})},B^{(\mathrm{fc})}}$ by

$$\mathcal{G}_{M,L,C,K,B^{(\mathrm{conv})},B^{(\mathrm{fc})}} = \left\{ \mathrm{mCNN}^{\mathrm{ReLU}}_{\boldsymbol{\theta}} \;\middle|\; \begin{array}{l} \mathrm{mCNN}^{\mathrm{ReLU}}_{\boldsymbol{\theta}} \text{ has } M \text{ residual blocks,} \\ \text{depth of each residual block is } L, \\ \boldsymbol{\theta} \text{ is compatible with } (C, K), \\ \max_{m,l} \|w^{(l)}_m\|_\infty \vee \|b^{(l)}_m\|_\infty \le B^{(\mathrm{conv})}, \\ \|W\|_\infty \vee \|b\|_\infty \le B^{(\mathrm{fc})} \end{array} \right\}.$$

In the above definition, we treat the mask pattern $z = (z_m)_m$ as learnable parameters. We can also treat z as fixed during training and search for best z as architecture search. The following theorems show that masked CNNs can approximate and estimate any Hölder function optimally even if the depth of residual blocks is specified *a priori*. We treat L as a constant against M in the theorems.

Theorem 3.3. *Let $f^\circ : [-1, 1]^D \to \mathbb{R}$ be a β-Hölder function. For any $K \in \{2, \ldots, D\}$ and $L \in \mathbb{N}_+$, there exists a CNN $f^{(\mathrm{CNN})}$ with $O(M \log M)$ residual blocks, each of which has depth L and $O(1)$ channels, and whose filter size is at most K, such that $\|f^\circ - f^{(\mathrm{CNN})}\|_\infty = \tilde{O}(M^{-\frac{\beta}{D}})$.*

Theorem 3.4. *Let $f^\circ : [-1, 1]^D \to \mathbb{R}$ be a β-Hölder function. For any $K \in \{2, \ldots, D\}$ and $L \in \mathbb{N}_+$, there exist the number of residual blocks $\tilde{M} = O(N^{\frac{D}{2\beta+D}} \log N)$, channel size $C = O(1)$, and norm bounds $B^{(\mathrm{conv})}, B^{(\mathrm{fc})} > 0$ such that for sufficiently large N, the clipped ERM estimator $\hat{f}$ of $\{\mathrm{clip}[f] \mid f \in \mathcal{G}_{\tilde{M},L,C,K,B^{(\mathrm{conv})},B^{(\mathrm{fc})}}\}$ achieves the estimation error $\|f^\circ - \hat{f}\|^2_{\mathcal{L}_2(\mathcal{P}_X)} = \tilde{O}_P(N^{-\frac{2\beta}{2\beta+D}})$.*

3.7 Chapter Conclusion

In this chapter, we established new approximation and statistical learning theories for ResNet-type CNNs by utilizing the block-sparse structure of FNNs to understand the role of skip connections in CNNs. We proved that any block-sparse FNN with M blocks is realizable by a CNN that has $O(M)$ additional parameters. Then, we derived the approximation and estimation error rates for CNNs from those for block-sparse FNNs. Our theory is general in a sense that it does not depend on a specific function class, as long as we can approximate it with block-sparse FNNs. Using this theory, we derived approximation and error rates for the Barron and Hölder classes in almost the same manner and showed that the estimation error of CNNs is same as that of FNNs, even if CNNs are dense and have constant channel size, filter size, and width with respect to the sample size. We can additionally make the depth of residual blocks constant if we allowed skip connections to have scaling schemes. The key techniques were careful evaluations of the Lipschitz constant and non-trivial weight parameter rescaling of NNs.

One of the interesting open questions is the role of the weight rescaling. We critically use the homogeneous property of the ReLU to change the relative scale between the block-sparse and fully-connected part, if it were not for this property, the estimation error rate would be worse. The general theory for rescaling, not restricted to the Barron nor Hölder classes would be beneficial for deeper understanding of the relationship between the approximation and estimation capabilities of FNNs and CNNs.

Another question is when the approximation and estimation error rates of CNNs can *exceed* that of FNNs. We can derive the same rates as FNNs essentially because we can realize block-sparse FNNs using CNNs that have the same order of parameters (see Theorem 3.1). If we can find some special structures of FNNs – like repetition, then, the CNNs might need fewer parameters and can achieve better estimation error rate. Note that there is no hope for enhancement for the Hölder case since the estimation rate using FNNs is already minimax optimal (up to logarithmic factors). It is left for future research which function classes and constraints of FNNs, like block-sparseness, we should choose.

3.A Proofs

3.A.1 Definitions of General CNNs and FNNs

We prove Theorem 3.1 and Theorem 3.2 in more general form. Specifically, we allow CNNs to have residual blocks with different depth and each residual block to have varying numbers of channels and filter sizes. Similarly, FNNs can have blocks with different depth, and the width of a block can be non-constant.

Definition 3.6 (Convolutional Neural Networks (CNNs))**.** *Let $M \in \mathbb{N}_+$ and $L_m \in \mathbb{N}_+$, which will be the number of residual blocks and the depth of m-th block, respectively. Let $C_m^{(l)}, K_m^{(l)}$ be the channel size and filter size of the l-th layer of the m-th block for $m \in [M]$ and $l \in [L_m]$. We assume $C_1^{(L_1)} = \cdots = C_M^{(L_M)}$ and denote it by $C^{(0)}$. Let $w_m^{(l)} \in \mathbb{R}^{K_m^{(l)} \times C_m^{(l)} \times C_m^{(l-1)}}$ and $b_m^{(l)} \in \mathbb{R}$ be the weight tensors and biases of l-th layer of the m-th block in the convolution part, respectively. Here $C_m^{(0)}$ is defined as $C^{(0)}$. Finally, let $W \in \mathbb{R}^{D \times C_0^{(L_0)}}$ and $b \in \mathbb{R}$ be the weight matrix and the bias for the fully-connected layer part, respectively. For $\boldsymbol{\theta} := ((w_m^{(l)})_{m,l}, (b_m^{(l)})_{m,l}, W, b)$ and an activation function $\sigma : \mathbb{R} \to \mathbb{R}$, we define $\mathrm{CNN}_{\boldsymbol{\theta}}^{\sigma} : \mathbb{R}^D \to \mathbb{R}^D$, the CNN constructed from $\boldsymbol{\theta}$, by*

$$\mathrm{CNN}_{\boldsymbol{\theta}}^{\sigma} := \mathrm{FC}_{W,b}^{\mathrm{id}} \circ (\mathrm{Conv}_{\boldsymbol{w}_M,\boldsymbol{b}_M}^{\sigma} + \mathrm{id}) \circ \cdots \circ (\mathrm{Conv}_{\boldsymbol{w}_1,\boldsymbol{b}_1}^{\sigma} + \mathrm{id}) \circ P,$$

where $\mathrm{Conv}_{\boldsymbol{w}_m,\boldsymbol{b}_m}^{\sigma} := \mathrm{Conv}_{w_m^{(L_m)},b_m^{(L_m)}}^{\sigma} \circ \cdots \circ \mathrm{Conv}_{w_m^{(1)},b_m^{(1)}}^{\sigma}$, $\mathrm{id} : \mathbb{R}^{D \times C^{(0)}} \to \mathbb{R}^{D \times C^{(0)}}$ is the identity function, and $P : \mathbb{R}^D \to \mathbb{R}^{D \times C^{(0)}}; x \mapsto \begin{bmatrix} x & 0 & \cdots & 0 \end{bmatrix}$ is a padding operation that adds zeros to align the number of channels.

Definition 3.7 (Fully-connected Neural Networks (FNNs))**.** *Let $M \in \mathbb{N}_+$ be the number of blocks in an FNN. Let $\boldsymbol{D}_m = (D_m^{(1)}, \ldots, D_m^{(L_m)}) \in \mathbb{N}_+^{L_m}$ be the sequence of intermediate dimensions of the m-th block, where $L_m \in \mathbb{N}_+$ is the depth of the m-th block for $m \in [M]$. Let $W_m^{(l)} \in \mathbb{R}^{D_m^{(l)} \times D_m^{(l-1)}}$ and $b_m^{(l)} \in \mathbb{R}^{D_m^{(l)}}$ be the weight matrix and the bias of the l-th layer of m-th block (with the convention $D_m^{(0)} = D$). Let $w_m \in \mathbb{R}^{D_m^{(L_m)}}$ be the weight (sub)vector of the final fully-connected layer corresponding to the m-th block and $b \in \mathbb{R}$ be the bias for the last layer. For $\boldsymbol{\theta} = ((W_m^{(l)})_{m,l}, (b_m^{(l)})_{m,l}, (w_m)_m, b)$ and an activation function $\sigma : \mathbb{R} \to \mathbb{R}$, we define $\mathrm{FNN}_{\boldsymbol{\theta}}^{\sigma} : \mathbb{R}^D \to \mathbb{R}$, the block-sparse FNN constructed from $\boldsymbol{\theta}$, by*

$$\mathrm{FNN}_{\boldsymbol{\theta}}^{\sigma} := \sum_{m=1}^{M} w_m^{\top} \mathrm{FC}_{\boldsymbol{W}_m,\boldsymbol{b}_m}^{\sigma}(\cdot) - b,$$

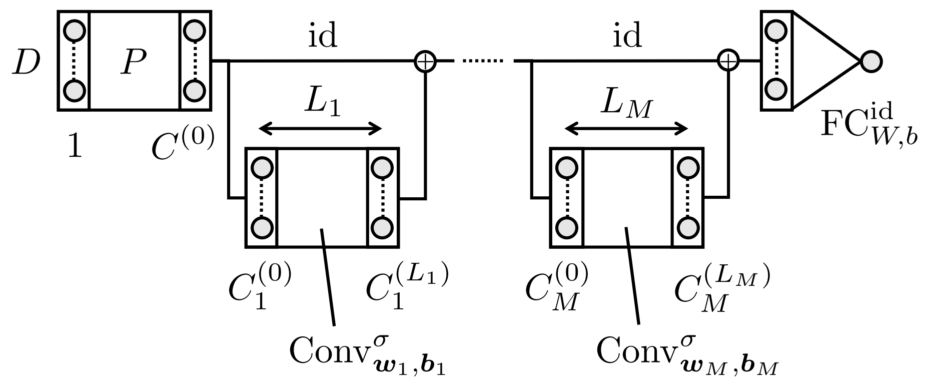

D
P
1
$C^{(0)}$
id
L_1
$C_1^{(0)}$
$C_1^{(L_1)}$
$\mathrm{Conv}^{\sigma}_{\boldsymbol{w}_1,\boldsymbol{b}_1}$
id
L_M
$C_M^{(0)}$
$C_M^{(L_M)}$
$\mathrm{Conv}^{\sigma}_{\boldsymbol{w}_M,\boldsymbol{b}_M}$
$\mathrm{FC}^{\mathrm{id}}_{W,b}$

Figure 3.3: Schematic view of a general ResNet-type CNN. Variables are as in Definition 3.6.

where $\mathrm{FC}^{\sigma}_{\boldsymbol{W}_m,\boldsymbol{b}_m} := \mathrm{FC}^{\sigma}_{W_m^{(L_m)},b_m^{(L_m)}} \circ \cdots \mathrm{FC}^{\sigma}_{W_m^{(1)},b_m^{(1)}}$.

Figure 3.3 shows the schematic view of a ResNet-type CNNs defined in Definition 3.6 and Figure 3.4 shows that of Definition 3.7. Definition 3.6 is reduced to Definition 3.1 by setting $L_m = L$, $\boldsymbol{C} = (C)_{m,l}$ and $\boldsymbol{K} = (K)_{m,l}$. Similarly, Definition 3.2 is a special case of Definition 3.7 where $L_m = L$ and $\boldsymbol{D} = (C)_{m,l}$. Correspondingly, we denote the set of functions realizable by CNNs and FNNs by $\mathcal{F}^{(\mathrm{CNN})}_{\boldsymbol{C},\boldsymbol{K},B^{(\mathrm{conv})},B^{(\mathrm{fc})}}$ and $\mathcal{F}^{(\mathrm{FNN})}_{\boldsymbol{D},B^{(\mathrm{bs})},B^{(\mathrm{fin})}}$, respectively [4].

3.A.2 Proof of Theorem 3.1

We restate Theorem 3.1 in more general form. Note that Theorem 3.1 is a special case of Theorem 3.5 where width, depth, channel sizes and filter sizes are same among blocks.

Theorem 3.5. *Let* $M \in \mathbb{N}_+$, $K \in \{2, \ldots D\}$, *and* $L_0 := \left\lceil \frac{D-1}{K-1} \right\rceil$. *Let* $L_m, D_m^{(l)} \in \mathbb{N}_+$ *and* $\boldsymbol{D} = (D_m^{(l)})_{m,l}$ *for* $m \in [M]$ *and* $l \in [L_m]$. *Then, there exist* $L'_m \in \mathbb{N}_+$, $\boldsymbol{C} = (C_m^{(l)})_{m,l}$, *and* $\boldsymbol{K} = (K_m^{(l)})_{m,l}$ $(m \in [M], l \in [L'_m])$ *satisfying the following properties:*

1. $L'_m \leq L_m + L_0$ $(\forall m \in [M])$,
2. $\max_{l \in [L'_m]} C_m^{(l)} \leq 4 \max_{l \in [L_m]} D_m^{(l)}$ $(\forall m \in [M])$, *and*
3. $\max_{l \in [L'_m]} K_m^{(l)} \leq K$ $(\forall m \in [M], \forall l \in [L'_m])$

such that for any $B^{(\mathrm{bs})}, B^{(\mathrm{fin})} > 0$, *any FNN in* $\mathcal{F}^{(\mathrm{FNN})}_{\boldsymbol{D},B^{(\mathrm{bs})},B^{(\mathrm{fin})}}$ *can be realized by a CNN in* $\mathcal{F}^{(\mathrm{CNN})}_{\boldsymbol{C},\boldsymbol{K},B^{(\mathrm{conv})},B^{(\mathrm{fc})}}$. *Here,* $B^{(\mathrm{conv})} = B^{(\mathrm{bs})}$ *and* $B^{(\mathrm{fc})} = B^{(\mathrm{fin})}(1 \vee (B^{(\mathrm{bs})})^{-1})$. *Further, if* $L_1 = \cdots = L_M$, *then we can choose* L'_m *to be a same value.*

[4]Note that information of M and L_m are included in $\boldsymbol{C}$, $\boldsymbol{K}$, and $\boldsymbol{D}$. Therefore, we do not have to put them as subscripts

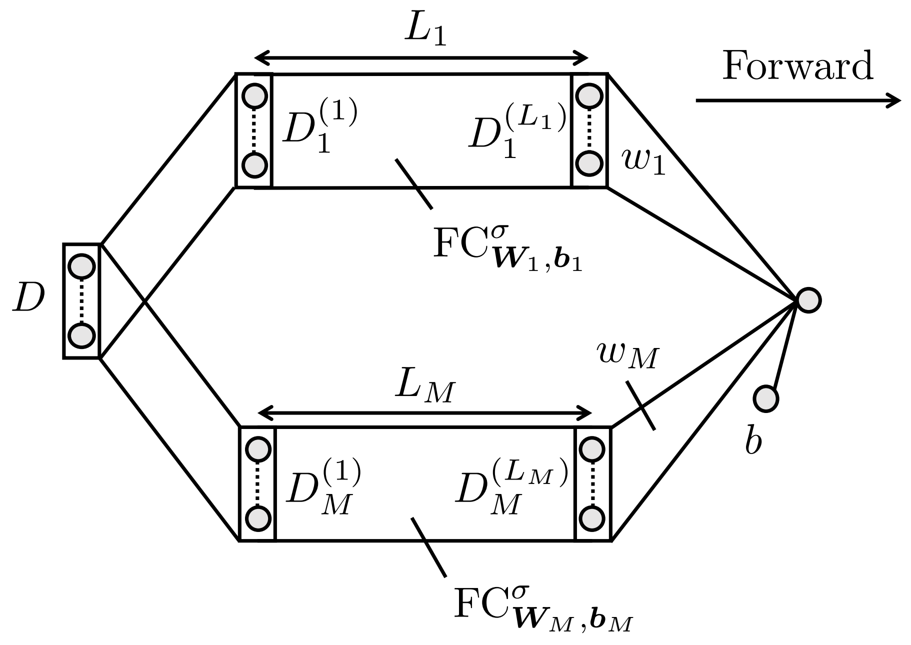

L_1
Forward
$D_1^{(1)}$
$D_1^{(L_1)}$
w_1
$\mathrm{FC}^{\sigma}_{\boldsymbol{W}_1,\boldsymbol{b}_1}$
D
w_M
L_M
b
$D_M^{(1)}$
$D_M^{(L_M)}$
$\mathrm{FC}^{\sigma}_{\boldsymbol{W}_M,\boldsymbol{b}_M}$

Figure 3.4: Schematic view of a general block-sparse FNN. Variables are as in Definition 3.7.

Remark 3.3. *For $K \leq K'$, we can embed $\mathbb{R}^K$ into $\mathbb{R}^{K'}$ by inserting zeros: $w = (w_1, \ldots, w_K) \mapsto w' = (w_1, \ldots, w_K, 0, \ldots, 0)$. It is easy to show $L^w = L^{w'}$. Using this equality, we can expand a size-K filter to size-K'. Furthermore, we can arbitrary increase the number of output channels of a convolution layer by adding filters consisting of zeros. Therefore, although properties 2 and 3 allow $C_m^{(l)}$ and $K_m^{(l)}$ to be different values, we can choose $C_m^{(l)}$ and $K_m^{(l)}$ so that inequalities in properties 2 and 3 are actually equals by adding filters consisting of zeros. In particular, when $D_m^{(l)}$'s are same value, we can choose $C_m^{(l)}$ to be same.*

Proof Overview

For $f^{(\mathrm{FNN})} \in \mathcal{F}^{(\mathrm{FNN})}$, we realize a CNN $f^{(\mathrm{CNN})}$ using M residual blocks by “serializing" blocks in the FNN and converting them into convolution layers.

First we multiply the channel size by three using the first padding operation. We will use the first channel for storing the original input signal for feeding to downstream blocks and the second and third ones for accumulating properly scaled outputs of each blocks, that is, $\sum_{m=1}^{m'} w_m^\top \mathrm{FC}^{\mathrm{ReLU}}_{\boldsymbol{W}_m,\boldsymbol{b}_m}(x)$ where w_m is the weight of the final fully-connected layer corresponding to the m-th block.

For $m = 1, \ldots, M$, we create the m-th residual block from the m-th block of $f^{(\mathrm{FNN})}$. First, we show that for any $a \in \mathbb{R}^D$ and $t \in \mathbb{R}$, there exists L_0-layered 4-channel ReLU CNN with $O(D)$ parameters whose first output coordinate equals to a ridge function $x \mapsto (a^\top x - t)_+$ (Lemma 3.1 and Lemma 3.2). Since the first layer of m-th block is concatenation of C hinge functions, it is realizable by a $4C$-channel ReLU CNN with L_0-layers.

For the l-th layer of the m-th block ($m \in [M], l = 2, \ldots, L'_m$), we prepare C size-1 filters made from the weight parameters of the corresponding layer of the FNN. Observing that the convolution operation with size-1 filter is equivalent to a dimension-wise affine transformation, the first coordinate of the output of l-th layer of the CNN is inductively same as that of the m-th block of the FNN. After computing the m-th block FNN using convolutions, we add its output to the accumulating channel in the skip connection.

Finally, we pick the first coordinate of the accumulating channel and subtract the bias term using the final affine transformation.

Decomposition of Affine Transformation

The following lemma shows that any affine transformation is realizable with a $\left\lceil \frac{D-1}{K-1} \right\rceil$-layered linear conventional CNN (without the final fully-connect layer).

Lemma 3.1. *Let $a \in \mathbb{R}^D$, $t \in \mathbb{R}$, $K \in \{2, \ldots, D-1\}$, and $L_0 := \left\lceil \frac{D-1}{K-1} \right\rceil$. Then, there exists*

$$w^{(l)} \in \begin{cases} \mathbb{R}^{K\times 2\times 1} & \textit{(for } l = 1\textit{)} \\ \mathbb{R}^{K\times 2\times 2} & \textit{(for } l = 2, \ldots, L_0 - 1\textit{)} \\ \mathbb{R}^{K\times 1\times 2} & \textit{(for } l = L_0\textit{)} \end{cases}$$

and $b \in \mathbb{R}$ such that

1. $\max_{l\in[L_o]} \|w_m\|_\infty = \|a\|_\infty$, $\max_{l\in[L_0]} \|b^{(l)}\|_\infty = |t|$, *and*

2. $\mathrm{Conv}^{\mathrm{id}}_{\boldsymbol{w},\boldsymbol{b}} : \mathbb{R}^D \to \mathbb{R}^D$ *satisfies* $\mathrm{Conv}^{\mathrm{id}}_{\boldsymbol{w},\boldsymbol{b}}(x) = a^\top x - t$ *for any $x \in [-1, 1]^D$.*

Proof. First, observe that the convolutional layer constructed from $u = \begin{bmatrix} u_1 & \ldots & u_K \end{bmatrix}^\top \in \mathbb{R}^{K\times 1\times 1}$ takes the inner product with the first K elements of the input signal: $L^u(x) = \sum_{k=1}^K u_k x_k$. In particular, $u = \begin{bmatrix} 0 & \ldots & 0 & 1 \end{bmatrix}^\top \in \mathbb{R}^{K\times 1\times 1}$ works as the "left-translation" by $K-1$. Therefore, we should define $\boldsymbol{w}$ so that it takes the inner product with the K left-most elements in the first channel and shift the input signal by $K-1$ with the second channel. Specifically, we define $\boldsymbol{w} = (w^{(1)}, \ldots, w^{(L_0)})$ by

$$(w^{(1)})_{:,1,:} = \begin{bmatrix} a_1 \\ \vdots \\ a_K \end{bmatrix}, \qquad (w^{(1)})_{:,2,:} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix},$$

$$(w^{(l)})_{:,1,:} = \begin{bmatrix} 0 & a_{(l-1)K+1} \\ \vdots & \vdots \\ 0 & a_{lK} \end{bmatrix}, \quad (w^{(l)})_{:,2,:} = \begin{bmatrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 1 & 0 \end{bmatrix},$$

$$(w^{(L_0)})_{:,1,:} = \begin{bmatrix} 0 & a_{(L_0-1)K+1} \\ \vdots & \vdots \\ 0 & a_D \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix}.$$

We set $\boldsymbol{b} := (\underbrace{0, \ldots, 0}_{L_0 - 1 \text{ times}}, t)$. Then, $\boldsymbol{w}$ and $\boldsymbol{b}$ satisfy the condition of the lemma. □

Transformation of a Linear CNN into a ReLU CNN

The following lemma shows that we can convert any linear CNN to a ReLU CNN that has approximately 4 times larger parameters. This type of lemma is also found in Petersen and Voigtlaender [2018b, Lemma 2.3]. The constructed network resembles to a CNN with CReLU activation [Shang et al., 2016].

Lemma 3.2. *Let $\boldsymbol{C} = (C^{(1)}, \ldots, C^{(L)}) \in \mathbb{N}_+^L$ be channel sizes $\boldsymbol{K} = (K^{(1)}, \ldots, K^{(L)}) \in \mathbb{N}_+^L$ be filter sizes. Let $w^{(l)} \in \mathbb{R}^{K^{(l)} \times C^l \times C^{(l)}}$ and $b^{(l)} \in \mathbb{R}^{(l)}$. Consider the linear convolution layers constructed from $\boldsymbol{w}$ and $\boldsymbol{b}$: $f_{\mathrm{id}} := \mathrm{Conv}^{\mathrm{id}}_{\boldsymbol{w},\boldsymbol{b}} : \mathbb{R}^D \to \mathbb{R}^{D \times C^{(L)}} \mathbb{N}_+^L$ where $\boldsymbol{w} = (w^{(l)})_l$ and $\boldsymbol{b} = (b^{(l)})_l$. Then, there exists a pair $\tilde{\boldsymbol{w}} = (\tilde{w}^{(l)})_{l \in [L]}, \tilde{\boldsymbol{b}} = (\tilde{b}^{(l)})_{l \in [L]}$ where $\tilde{w}^{(l)} \in \mathbb{R}^{K^{(l)} \times 2C^{(l)} \times 2C^{(l-1)}}$ and $\tilde{b}^{(l)} \in \mathbb{R}^{2C^{(l)}}$ such that*

1. $\max_{l \in [L]} \|\tilde{w}^{(l)}\|_\infty = \max_{l \in [L]} \|w^{(l)}\|_\infty$, $\max_{l \in [L]} \|\tilde{b}^{(l)}\|_\infty = \max_{l \in [L]} \|b^{(l)}\|_\infty$, *and*

2. $f_{\mathrm{ReLU}} := \mathrm{Conv}^{\mathrm{ReLU}}_{\tilde{\boldsymbol{w}},\tilde{\boldsymbol{b}}} : \mathbb{R}^D \to \mathbb{R}^{D \times 2C^{(L)}}$, *satisfies* $f_{\mathrm{ReLU}}(\cdot) = (f_{\mathrm{id}}(\cdot)_+, f_{\mathrm{id}}(\cdot)_-)$.

Proof. We define $\tilde{\boldsymbol{w}}$ and $\tilde{\boldsymbol{b}}$ as follows:

$$\begin{aligned} (\tilde{w}^{(1)})_{k,:,:} &= \begin{bmatrix} (w^{(1)})_{k,:,:} \\ -(w^{(1)})_{k,:,:} \end{bmatrix} \text{ for } k = 1, \ldots, K^{(1)}, \\ (\tilde{w}^{(l)})_{k,:,:} &= \begin{bmatrix} (w^{(l)})_{k,:,:} & -(w^{(l)})_{k,:,:} \\ -(w^{(l)})_{k,:,:} & (w^{(l)})_{k,:,:} \end{bmatrix} \text{ for } k = 1, \cdots K^{(l)}, \\ \tilde{b}^{(l)} &= \begin{bmatrix} b^{(l)} \\ -b^{(l)} \end{bmatrix} \end{aligned}$$

By definition, a pair $(\tilde{\boldsymbol{w}}, \tilde{\boldsymbol{b}})$ satisfies the conditions (1) and (2). For any $x \in \mathbb{R}^D$, we set $y^{(l)} := \mathrm{Conv}^{\mathrm{id}}_{\boldsymbol{w}[1:l],\boldsymbol{b}[1:l]}(x) \in \mathbb{R}^{C^{(l)} \times D}$. We will prove

$$\mathrm{Conv}^{\mathrm{ReLU}}_{\tilde{\boldsymbol{w}}[1:l],\tilde{\boldsymbol{b}}[1:l]}(x) = \begin{bmatrix} y^{(l)}_+ & y^{(l)}_- \end{bmatrix}^\top \tag{3.3}$$

for $l = 1, \dots, L$ by induction. Note that we obtain $f_{\mathrm{ReLU}}(\cdot) = (f_{\mathrm{id}+}(\cdot), f_{\mathrm{id}-}(\cdot))$ by setting $l = L$. For $l = 1$, by definition of $\tilde{w}^{(1)}$ we have,

$$(\tilde{w}^{(1)})_{\alpha,:,:} x^{\beta,:} = \begin{bmatrix} (w^{(1)})_{\alpha,:,:} x^{\beta,:} \\ -(w^{(1)})_{\alpha,:,:} x^{\beta,:} \end{bmatrix}$$

for any $\alpha, \beta \in [D]$. Summing them up and using the definition of $\tilde{b}^{(1)}$ yield

$$[L^{\tilde{w}^{(1)}}(x) - \mathbf{1}_D \otimes \tilde{b}^{(1)}]^\top = \begin{bmatrix} L^{w^{(1)}}(x) - \mathbf{1}_D \otimes b^{(1)} \\ -\left(L^{w^{(1)}}(x) - \mathbf{1}_D \otimes b^{(1)}\right) \end{bmatrix}^\top$$

Suppose (3.3) holds up to l ($l < L$), by the definition of $\tilde{w}^{(l+1)}$,

$$\begin{aligned}
(\tilde{w}^{(l+1)})_{\alpha,:,:} \begin{bmatrix} (y_+^{(l)})^{\beta,:} \\ (y_-^{(l)})^{\beta,:} \end{bmatrix} &= \begin{bmatrix} (w^{(l+1)})_{\alpha,:,:} & -(w^{(l+1)})_{\alpha,:,:} \\ -(w^{(l+1)})_{\alpha,:,:} & (w^{(l+1)})_{\alpha,:,:} \end{bmatrix} \begin{bmatrix} (y_+^{(l)})^{\beta,:} \\ (y_-^{(l)})^{\beta,:} \end{bmatrix} \\
&= \begin{bmatrix} (w^{(l+1)})_{\alpha,:,:} \left((y_+^{(l)})^{\beta,:} - (y_-^{(l)})^{\beta,:}\right) \\ -(w^{(l+1)})_{\alpha,:,:} \left((y_+^{(l)})^{\beta,:} - (y_-^{(l)})^{\beta,:}\right) \end{bmatrix} \\
&= \begin{bmatrix} (w^{(l+1)})_{\alpha,:,:} (y^{(l)})^{\beta,:} \\ -(w^{(l+1)})_{\alpha,:,:} (y^{(l)})^{\beta,:} \end{bmatrix}
\end{aligned}$$

for any $\alpha, \beta \in [D]$. Again, by taking the summation and using the definition of $\tilde{b}^{(l+1)}$, we get

$$[L^{\tilde{w}^{(l+1)}}([y_+^{(l)}, y_-^{(l)}]) - \mathbf{1}_D \otimes \tilde{b}^{(1)}]^\top = \begin{bmatrix} L^{w^{(l+1)}}(y^{(l)}) - \mathbf{1}_D \otimes b^{(l+1)} \\ -\left(L^{w^{(l+1)}}(y^{(l)}) - \mathbf{1}_D \otimes b^{(l+1)}\right) \end{bmatrix}^\top .$$

By applying ReLU, we get

$$\mathrm{Conv}^{\mathrm{ReLU}}_{\tilde{w}^{(l+1)}, \tilde{b}^{(l+1)}} \left([y_+^{(l)}, y_-^{(l)}]\right) = \mathrm{ReLU}\left([y^{(l+1)}, -y^{(l+1)}]\right). \tag{3.4}$$

By using the induction hypothesis, we get

$$\begin{aligned}
\mathrm{Conv}^{\mathrm{ReLU}}_{\tilde{\boldsymbol{w}}[1:(l+1)], \tilde{\boldsymbol{b}}[1:(l+1)]}(x) &= \mathrm{Conv}^{\mathrm{ReLU}}_{\tilde{w}^{(l+1)}, \tilde{b}^{(l+1)}} \left([y_+^{(l)}, y_-^{(l)}]\right) \\
&= \mathrm{ReLU}\left([y^{(l+1)}, -y^{(l+1)}]\right) \\
&= [y_+^{(l+1)}, -y_-^{(l+1)}]
\end{aligned}$$

Therefore, the claim holds for $l + 1$. By induction, the claim holds for L, which is what we want to prove. $\square$

Concatenation of CNNs

We can concatenate two CNNs with the same depths and filter sizes in parallel. Although it is almost trivial, we state it formally as a proposition. In the following proposition, $C^{(0)}$ and $C'^{(0)}$ are not necessarily one.

Proposition 3.1. *Let $\boldsymbol{C} = (C^{(l)})_{l\in[L]}$, $\boldsymbol{C}' = (C'^{(l)})_{l\in[L]}$, and $\boldsymbol{K} = (K^{(l)})_{l\in[L]} \in \mathbb{N}_+^L$. Let $w^{(l)} \in \mathbb{R}^{K^{(l)}\times C^{(l)}\times C^{(l-1)}}$, $b \in \mathbb{R}^{C^{(l)}}$ and denote $\boldsymbol{w} = (w^{(l)})_l$ and $\boldsymbol{b} = (b^{(l)})_l$. We define $\boldsymbol{w}'$ and $\boldsymbol{b}'$ in the same way, with the exception that $C^{(l)}$ is replaced with $C'^{(l)}$. We define $\tilde{\boldsymbol{w}} = (\tilde{w}^{(1)}, \ldots, \tilde{w}^{(L)})$ and $\tilde{\boldsymbol{b}} = (\tilde{b}^{(1)}, \ldots, \tilde{b}^{(L)})$ by*

$$(\tilde{w}^{(l)})_{k,:,:} := \begin{bmatrix} w^{(l)} & 0 \\ 0 & w'^{(l)} \end{bmatrix} \in \mathbb{R}^{(C^{(l)}+C'^{(l)})\times(C^{(l-1)}+C'^{(l-1)})}$$

$$\tilde{b}^{(l)} := \begin{bmatrix} b^{(l)} \\ b'^{(l)} \end{bmatrix} \in \mathbb{R}^{(C^{(l)}+C'^{(l)})}$$

for $l \in [L]$ and $k \in [K^{(l)}]$. Then, we have,

$$\mathrm{Conv}^{\sigma}_{\tilde{\boldsymbol{w}},\tilde{\boldsymbol{b}}}(\begin{bmatrix} x & x' \end{bmatrix}) = \begin{bmatrix} \mathrm{Conv}^{\sigma}_{\boldsymbol{w},\boldsymbol{b}}(x) & \mathrm{Conv}^{\sigma}_{\boldsymbol{w}',\boldsymbol{b}'}(x') \end{bmatrix}$$

for any $x, x' \in \mathbb{R}^{D\times C^{(0)}}$ and any $\sigma : \mathbb{R} \to \mathbb{R}$. □

Note that by the definition of $\|\cdot\|_0$ and $\|\cdot\|_\infty$, we have

$$\max_{l\in[L]} \|\tilde{w}^{(l)}\|_\infty = \max_{l\in[L]} \|w^{(l)}\|_\infty \vee \|w'^{(l)}\|_\infty,$$
$$\max_{l\in[L]} \|\tilde{b}^{(l)}\|_\infty = \max_{l\in[L]} \|b^{(l)}\|_\infty \vee \|b'^{(l)}\|_\infty.$$

Proof of Theorem 3.5

By the definition of $\mathcal{F}^{(\mathrm{FNN})}_{\boldsymbol{D},B^{(\mathrm{bs})},B^{(\mathrm{fin})}}$, there exists a 4-tuple $\boldsymbol{\theta} = ((W_m^{(l)})_{m,l}, (b_m^{(l)})_{m,l}, (w_m)_m, b)$ compatible with $(D_m^{(l)})_{m,l}$ ($m \in [M]$ and $l \in [L_m]$) such that

$$\max_{m\in[M],l\in[L_m]} (\|W_m^{(l)}\|_\infty \vee \|b_m^{(l)}\|_\infty) \le B^{(\mathrm{bs})},$$
$$\max_{m\in[M]} \|w_m\|_\infty \vee |b| \le B^{(\mathrm{fin})},$$

and $f^{(\mathrm{FNN})} = \mathrm{FNN}^{\mathrm{ReLU}}_{\boldsymbol{\theta}}$. We will construct the desired CNN consisting of M residual blocks, whose m-th residual block is made from the ingredients of the corresponding m-th block in $f^{(\mathrm{FNN})}$ (specifically, $\boldsymbol{W}_m := (W_m^{(l)})_{l\in[L_m]}$, $\boldsymbol{b}_m := (b_m^{(l)})_{l\in[L_m]}$, and w_m).

Padding Block We prepare the padding operation P that multiply the channel size by 3 (i.e., we set $C^{(0)} = 3$).

$\boldsymbol{m = 1, \ldots, M}$ **Blocks** For fixed $m \in [M]$, we first create a CNN realizing $\mathrm{FC}^{\mathrm{ReLU}}_{\boldsymbol{W}_m, \boldsymbol{b}_m}$. We treat the first layer (i.e. $l = 1$) of $\mathrm{FC}^{\mathrm{ReLU}}_{\boldsymbol{W}_m, \boldsymbol{b}_m}$ as concatenation of $D_m^{(1)}$ hinge functions $\mathbb{R}^D \ni x \mapsto f_d(x) := ((W_m^{(1)})_d x - b_m^{(1)})_+$ for $d \in [D_m^{(1)}]$. Here, $(W_m^{(1)})_d \in \mathbb{R}^{1 \times D}$ is the d-th row of the matrix $W_m^{(1)} \in \mathbb{R}^{D_m^{(1)} \times D}$. We apply Lemma 3.1 and Lemma 3.2 and obtain ReLU CNNs realizing the hinge functions. By combining them in parallel using Proposition 3.1, we have a learnable parameter $\boldsymbol{\theta}_m^{(1)}$ such that the ReLU CNN $\mathrm{Conv}^{\mathrm{ReLU}}_{\boldsymbol{\theta}_m^{(1)}} : \mathbb{R}^{D \times 2} \to \mathbb{R}^{D \times 2D_m^{(1)}}$ constructed from $\boldsymbol{\theta}_m^{(1)}$ satisfies

$$\mathrm{Conv}^{\mathrm{ReLU}}_{\boldsymbol{\theta}_m^{(1)}}(\begin{bmatrix} x & x' \end{bmatrix}^\top)_1 = \begin{bmatrix} f_1(x) & * & \cdots & f_{D_m^{(1)}}(x) & * \end{bmatrix}^\top.$$

Since we double the channel size in the $m = 0$ part, the skip connection has 2 channels. Therefore, we made $\mathrm{Conv}^{\mathrm{ReLU}}_{\boldsymbol{\theta}_m^{(1)}}$ so that it has 2 input channels and neglects the input signals coming from the second one. This is possible by adding filters consisting of zeros appropriately.

Next, for l-th layer ($l = 2, \ldots, L_m$), we prepare size-1 filters $w_m^{(2)} \in \mathbb{R}_m^{1 \times D_m^{(2)} \times 2D^{(1)}}$ for $l = 2$ and $w_m^{(l)} \in \mathbb{R}^{1 \times D_m^{(l)} \times 2D_m^{(l-1)}}$ for $l = 3, \ldots, D_m^{(L_m)}$ defined by

$$(w_m^{(l)})_{1,:,:} := \begin{cases} W_m^{(2)} \otimes \begin{bmatrix} 1 & 0 \end{bmatrix} & \text{if } l = 2 \\ W_m^{(l)} & \text{if } l = 3, \ldots, D_m^{(L_m)}, \end{cases}$$

where $\otimes$ is the Kronecker product of matrices. Intuitively, the $l = 2$ layer will pick all odd indices of the output of $\mathrm{Conv}^{\mathrm{ReLU}}_{\boldsymbol{\theta}_m^{(1)}}$ and apply the fully-connected layer. We construct CNNs from $\theta_m^{(l)} := (w_m^{(l)}, b_m^{(l)})$ ($l \geq 2$) and concatenate them along with $\mathrm{Conv}^{\mathrm{ReLU}}_{\boldsymbol{\theta}_m^{(1)}}$:

$$\mathrm{Conv}_m := \mathrm{Conv}^{\mathrm{ReLU}}_{\theta_m^{(L_m)}} \circ \cdots \circ \mathrm{Conv}^{\mathrm{ReLU}}_{\theta_m^{(2)}} \circ \mathrm{Conv}^{\mathrm{ReLU}}_{\boldsymbol{\theta}_m^{(1)}}.$$

Note that $\mathrm{Conv}^{\mathrm{ReLU}}_{\theta_m^{(l)}}$ ($l \geq 2$) just rearranges parameters of $\mathrm{FC}^{\mathrm{ReLU}}_{\boldsymbol{W}_m, \boldsymbol{b}_m}$. The output dimension of Conv_m is either $\mathbb{R}^{D \times 2D_m^{(L_m)}}$ (if $L_m = 1$) or $\mathbb{R}^{D \times D_m^{(L_m)}}$ (if $L_m \geq 2$)., We denote the output channel size (either $2D_m^{(L_m)}$ or $D_m^{(L_m)}$) by $D_m^{(\mathrm{out})}$. By the inductive calculation, we have

$$\mathrm{Conv}_m(x)_1 = \begin{cases} \mathrm{FC}^{\mathrm{ReLU}}_{\boldsymbol{W}_m, \boldsymbol{b}_m}(x) \otimes \begin{bmatrix} 1 & 0 \end{bmatrix} & \text{if } L_m = 1 \\ \mathrm{FC}^{\mathrm{ReLU}}_{\boldsymbol{W}_m, \boldsymbol{b}_m}(x) & \text{if } L_m \geq 2 \end{cases}.$$

By definition, Conv_m has $L_0 + L_m - 1$ layers and at most $4D_m^{(1)} \vee \max_{l=2,\ldots L_m} D_m^{(l)} \leq 4 \max_{l \in [L_m]} D_m^{(l)}$ channels. The ∞-norm of its parameters does not exceed that of parameters in $\mathrm{FC}^{\mathrm{ReLU}}_{\boldsymbol{W}_m, \boldsymbol{b}_m}$.

Next, we consider the filter $\tilde{w}_m \in \mathbb{R}^{1\times 3\times D_m^{(\mathrm{out})}}$ defined by

$$(\tilde{w}_m)_{1,:,:} = \frac{B^{(\mathrm{bs})}}{B^{(\mathrm{fin})}} \begin{cases} \begin{bmatrix} 0 & \cdots & 0 \\ & w_m \otimes \begin{bmatrix} 0 & 1 \end{bmatrix} & \\ & -w_m \otimes \begin{bmatrix} 0 & 1 \end{bmatrix} & \end{bmatrix} & \text{if } L_m = 1 \\ \begin{bmatrix} 0 & \cdots & 0 \\ & w_m & \\ & -w_m & \end{bmatrix} & \text{if } L_m \geq 2 \end{cases},$$

Then, $\mathrm{Conv}'_m := \mathrm{Conv}^{\mathrm{ReLU}}_{\tilde{w}_m,0}$ adds the output of m-th residual block, weighted by w_m, to the second channel in the skip connections, while keeping the first channel intact. Note that the final layer of each residual block does not have the ReLU activation. By definition, Conv'_m has $D_m^{(L_m)}$ parameters.

Given Conv_m and Conv'_m for each $m \in [M]$, we construct a CNN realizing $\mathrm{FNN}^{\mathrm{ReLU}}_{\boldsymbol{\theta}}$. Let $f^{(\mathrm{conv})} : \mathbb{R}^D \to \mathbb{R}^{D\times 3}$ be the sequential interleaving concatenation of Conv_m and Conv'_m, that is,

$$f^{(\mathrm{conv})} := (\mathrm{Conv}'_M \circ \mathrm{Conv}_M + I) \circ \cdots \quad \circ (\mathrm{Conv}'_1 \circ \mathrm{Conv}_1 + I) \circ P.$$

Then, we have

$$f^{(\mathrm{conv})}_{1,:} = \begin{bmatrix} 0 & z_1 & z_2 \end{bmatrix} \in \mathbb{R}^3$$

where $z_1 = \frac{B^{(\mathrm{bs})}}{B^{(\mathrm{fin})}} \sum_{m=1}^M \left(w_m^\top \mathrm{FC}^{\mathrm{ReLU}}_{\boldsymbol{W}_m, \boldsymbol{b}_m} \right)_+$ and $z_2 = \frac{B^{(\mathrm{bs})}}{B^{(\mathrm{fin})}} \sum_{m=1}^M \left(w_m^\top \mathrm{FC}^{\mathrm{ReLU}}_{\boldsymbol{W}_m, \boldsymbol{b}_m} \right)_-$.

Final Fully-connected Layer Finally, we set

$$w := \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \frac{B^{(\mathrm{fin})}}{B^{(\mathrm{bs})}} & 0 & \cdots & 0 \\ -\frac{B^{(\mathrm{fin})}}{B^{(\mathrm{bs})}} & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{D\times 3}$$

and put $\mathrm{FC}^{\mathrm{id}}_{w,b}$ on top of $f^{(\mathrm{conv})}$ to pick the first coordinate of $f^{(\mathrm{conv})}$ and subtract the bias term. By definition, $f^{(\mathrm{CNN})} := \mathrm{FC}^{\mathrm{id}}_{w,b} \circ f^{(\mathrm{conv})}$ satisfies $f^{(\mathrm{CNN})} = f^{(\mathrm{FNN})}$.

Property Check We will check $f^{(\mathrm{FNN})}$ satisfies the desired properties. **(Property 1)**: Since Conv_m and Conv'_m has $L_0 + L_m - 1$ and 1 layers, respectively, the $m(\geq 1)$-th residual block of $f^{(\mathrm{CNN})}$ has $L'_m = L_0 + L_m$ layers. In particular, if L_m's are same, we can choose L'_m to be the same value $L_0 + L_m$. **(Property 2)**: Conv_m has at most $4 \max_{l\in[L_m]} D_m^{(l)}$ channels and Conv'_m has at most 2 channels, respectively. Therefore, the channel size of the m-th block is at most $4 \max_{l\in[L_m]} D_m^{(l)}$. **(Property 3)**: Since each filter of Conv_m and Conv'_m is at most K, the filter size of CNN is also at most K. **(Properties on $B^{(\mathrm{conv})}$ and $B^{(\mathrm{fin})}$)**: Parameters of $f^{(\mathrm{conv})}$

are either 0, or parameters of $\mathrm{FC}^{\mathrm{ReLU}}_{\boldsymbol{W}_m, \boldsymbol{W}_m}$, whose absolute value is bounded by $B^{(\mathrm{bs})}$ or $\frac{B^{(\mathrm{bs})}}{B^{(\mathrm{fin})}} w_m$. Since we have $\|w_m\|_\infty \leq B^{(\mathrm{fin})}$, the ∞-norm of parameters in $f^{(\mathrm{CNN})}$ is bounded by $B^{(\mathrm{bs})}$. The parameters of the final fully-connected layer $\mathrm{FC}_{w,b}$ is either $\frac{B^{(\mathrm{fin})}}{B^{(\mathrm{bs})}}$, 0, or b, therefore their norm is bounded by $\frac{B^{(\mathrm{fin})}}{B^{(\mathrm{bs})}} \vee B^{(\mathrm{fin})}$. □

As discussed in the beginning of this section, Theorem 3.1 is the special case of Theorem 3.5.

Remark 3.4. *Another way to construct a CNN which is identical (as a function) to a given FNN is as follows. First, we use a "rotation" convolution with D filters, each of which has a size D, to serialize all input signals to channels of a single input dimension. Then, apply size-1 convolution layers, whose l-th layer consisting of appropriately arranged weight parameters of the l-th layer of the FNN. This is essentially what Petersen and Voigtlaender [2018a] did to prove the existence of a CNN equivalent to a given FNN. To restrict the size of filters to K, we should further replace the the first convolution layer with $O(D/K)$ convolution layers with size-K filters. We can show essentially same statement using this construction method.*

3.A.3 Proof of Theorem 3.2

Same as Theorem 3.1, we restate Theorem 3.2 in more general form. We denote $\mathcal{F}^{(\mathrm{CNN})} := \mathcal{F}^{(\mathrm{CNN})}_{\boldsymbol{C},\boldsymbol{K},B^{(\mathrm{conv})},B^{(\mathrm{fc})}}$ and $\mathcal{F}^{(\mathrm{FNN})} := \mathcal{F}^{(\mathrm{FNN})}_{\boldsymbol{D},B^{(\mathrm{bs})},B^{(\mathrm{fin})}}$ in shorthand.

Theorem 3.6. *Let $f^\circ : \mathbb{R}^D \to \mathbb{R}$ be a measurable function and $B^{(\mathrm{bs})}, B^{(\mathrm{fin})} > 0$. Let M, K, L_0, L_m, and $\boldsymbol{D}$ as in Theorem 3.5. Suppose $L'_m, \boldsymbol{C}, \boldsymbol{K}, B^{(\mathrm{conv})}$ and $B^{(\mathrm{fc})}$ satisfy $\mathcal{F}^{(\mathrm{FNN})} \subset \mathcal{F}^{(\mathrm{CNN})}$ for $B^{(\mathrm{bs})}$ and $B^{(\mathrm{fin})}$ (their existence is ensured for any $B^{(\mathrm{bs})}$ and $B^{(\mathrm{fin})}$ by Theorem 3.5). Suppose that the covering nubmer of $\mathcal{F}^{(\mathrm{CNN})}$ is larger than 3. Then, the clipped ERM estimator $\hat{f}$ in $\mathcal{F} := \{\mathrm{clip}[f] \mid f \in \mathcal{F}^{(\mathrm{CNN})}\}$ satisfies*

$$\mathbb{E}_{\mathcal{D}} \|\hat{f} - f^\circ\|^2_{\mathcal{L}^2(\mathcal{P}_X)} \leq C \left(\inf_f \|f - f^\circ\|^2_\infty + \frac{\tilde{F}^2}{N} \Lambda_2 \log(2\Lambda_1 B N) \right). \tag{3.5}$$

Here, f ranges over $\mathcal{F}^{(\mathrm{FNN})}$, $C_0 > 0$ is a universal constant, $\tilde{F} := \frac{\|f^\circ\|_\infty}{\sigma} \vee \frac{1}{2}$, and $B = B^{(\mathrm{conv})} \vee B^{(\mathrm{fc})}$. $\Lambda_1 = \Lambda_1(\mathcal{F}^{(\mathrm{CNN})})$ and $\Lambda_2 = \Lambda_2(\mathcal{F}^{(\mathrm{CNN})})$ are defined by

$$\Lambda_1 := (2M+3) C^{(0)} D (1 \vee B^{(\mathrm{fc})})(1 \vee B^{(\mathrm{conv})}) \varrho \varrho^+$$
$$\Lambda_2 := \sum_{m=1}^{M} \sum_{l=1}^{L'_m} \left(C_m^{(l-1)} C_m^{(l)} K_m^{(l)} + C_m^{(l)} \right) + C^{(0)} D + 1,$$

where $\varrho = \prod_{m=1}^M (1 + \rho_m)$, $\varrho^+ = 1 + \sum_{m=1}^M L'_m \rho_m^+$, $\rho_m := \prod_{l=1}^{L'_m} C_m^{(l-1)} K_m^{(l)} B^{(\mathrm{conv})}$ and $\rho_m^+ := \prod_{l=1}^{L'_m} (1 \vee C_m^{(l-1)} K_m^{(l)} B^{(\mathrm{conv})})$.

Again, Theorem 3.2 is a special case of Theorem 3.6 where width, depth, channel sizes and filter sizes are same among blocks. Note that the definitions of Λ_1, Λ_2, ρ, ρ^+, ϱ, and ϱ^+ in Theorem 3.2 and Theorem 3.6 are consistent by this specialization.

Proof Overview

We relate the approximation error of Theorem 3.2 with the estimation error using the covering number of the hypothesis class $\mathcal{F}^{(\mathrm{CNN})}$. Although there are several theorems of this type, we employ the one in Schmidt-Hieber [2020] due to its convenient form (Lemma 3.5). We can prove that the logarithm of the covering number is upper bounded by $\Lambda_2 \log((B^{(\mathrm{conv})} \vee B^{(\mathrm{fc})})\Lambda_1/\varepsilon)$ (Lemma 3.4) using the similar techniques to the one in Schmidt-Hieber [2020]. Theorem 3.2 is the immediate consequence of these two lemmas.

To prove Corollary 3.1, we set $M = O(N^\alpha)$ for some $\alpha > 0$. Then, under the assumption of the corolarry, we have $\|f^\circ - \hat{f}\|^2_{\mathcal{L}^2(\mathcal{P}_x)} = \tilde{O}\left(\max\left(N^{-2\alpha\gamma_1}, N^{\alpha\gamma_2 - 1}\right)\right)$ from Theorem 3.2. The order of the right hand side with respect to N is minimized when $\alpha = \frac{1}{2\gamma_1+\gamma_2}$. By substituting α, we can prove Corollary 3.1.

Covering Number of CNNs

The goal of this section is to prove Lemma 3.4, stated in Section 3.A.3, that evaluates the covering number of the set of functions realized by CNNs.

Bounds for convolutional layers

We assume $w, w' \in \mathbb{R}^{K\times J\times I}$, $b, b' \in \mathbb{R}$, and $x \in \mathbb{R}^{D\times I}$ unless specified. We have in mind that the activation function σ is either the ReLU function or the identity function id. But the following proposition holds for any 1-Lipschitz function such that $\sigma(0) = 0$. Remember that we can treat L^w as a linear operator from $\mathbb{R}^{D\times I}$ to $\mathbb{R}^{D\times J}$. We endow $\mathbb{R}^{D\times I}$ and $\mathbb{R}^{D\times J}$ with the sup norm and denote the operator norm L^w by $\|L^w\|_{\mathrm{op}}$.

Proposition 3.2. *It holds that* $\|L^w\|_{\mathrm{op}} \leq IK\|w\|_\infty$.

Proof. Write $w = (w_{kji})_{k\in[K],j\in[J],i\in[I]}$, $L^w = ((L^w)^{\beta,j}_{\alpha,i})_{\alpha,\beta\in[D],j\in[J],i\in[I]}$. For any $x = (x_{\alpha,i})_{\alpha\in[D],i\in[I]} \in \mathbb{R}^{D\times I}$, the sup norm of $y := (y_{\beta j})_{\beta\in[D] j\in[J]} = L^w(x)$ is evaluated as follows:

$$\begin{aligned}
\|y\|_\infty = \max_{\beta,j} |y_{\beta,j}| &\leq \max_{\beta,j} \sum_{\alpha,i} |(L^w)^{\beta,j}_{\alpha,i}||x_{\alpha,i}| \\
&\leq \max_{\beta,j} \sum_{\alpha,i} |(L^w)^{\beta,j}_{\alpha,i}|\|x\|_\infty \\
&= \max_{\beta,j} \sum_{\alpha,i} |w_{(\alpha-\beta+1),j,i}|\|x\|_\infty \\
&\leq IK\|w\|_\infty\|x\|_\infty
\end{aligned}$$

□

Proposition 3.3. *It holds that* $\|\mathrm{Conv}^\sigma_{w,b}(x)\|_\infty \leq \|L^w\|_{\mathrm{op}}\|x\|_\infty + |b|$.

Proof.

$$\begin{aligned}\|\mathrm{Conv}^\sigma_{w,b}(x)\|_\infty &\le \|\sigma(L^w(x) - \mathbf{1}_D \otimes b)\|_\infty \\ &\le \|L^w(x) - \mathbf{1}_D \otimes b\|_\infty \\ &\le \|L^w(x)\|_\infty + \|\mathbf{1}_D \otimes b\|_\infty \\ &\le \|L^w\|_{\mathrm{op}}\|x\|_\infty + |b|.\end{aligned}$$

□

Proposition 3.4. *The Lipschitz constant of* $\mathrm{Conv}^\sigma_{w,b}$ *is bounded by* $\|L^w\|_{\mathrm{op}}$.

Proof. For any $x, x' \in \mathbb{R}^{D\times I}$,

$$\begin{aligned}\|\mathrm{Conv}^\sigma_{w,b}(x) - \mathrm{Conv}^\sigma_{w,b}(x')\|_\infty &= \|\sigma\left(L^w(x) - \mathbf{1}_D \otimes b\right) - \sigma\left(L^w(x') - \mathbf{1}_D \otimes b\right)\|_\infty \\ &\le \|\left(L^w(x) - \mathbf{1}_D \otimes b\right) - \left(L^w(x') - \mathbf{1}_D \otimes b\right)\|_\infty \\ &\le \|L^w(x - x')\|_\infty \\ &\le \|L^w\|_{\mathrm{op}}\|x - x'\|_\infty.\end{aligned}$$

Note that the first inequality holds because the ReLU function is 1-Lipschitz. □

Proposition 3.5. *It holds that* $\|\mathrm{Conv}^\sigma_{w,b}(x) - \mathrm{Conv}^\sigma_{w',b'}(x)\| \le \|L^{w-w'}\|_{\mathrm{op}}\|x\|_\infty + |b - b'|$.

Proof.

$$\begin{aligned}\|\mathrm{Conv}^\sigma_{w,b}(x) - \mathrm{Conv}^\sigma_{w',b'}(x)\|_\infty &= \|\sigma(L^w(x) - \mathbf{1}_D \otimes b) - \sigma(L^{w'}(x) - \mathbf{1}_D \otimes b')\|_\infty \\ &\le \|(L^w(x) - \mathbf{1}_D \otimes b) - (L^{w'}(x) - \mathbf{1}_D \otimes b')\| \\ &= \|L^w(x) - L^{w'}(x)\| + \|\mathbf{1}_D \otimes (b - b')\|_\infty \\ &\le \|L^{w-w'}\|_{\mathrm{op}}\|x\|_\infty + |b - b'|\end{aligned}$$

□

Bounds for fully-connected layers

In the following propositions in this subsection, we assume $W, W' \in \mathbb{R}^{DC\times C'}$, $b, b' \in \mathbb{R}^{C'}$, and $x \in \mathbb{R}^{D\times C}$. Again, these propositions hold for any 1-Lipschitz function $\sigma : \mathbb{R} \to \mathbb{R}$ such that $\sigma(0) = 0$. But $\sigma = \mathrm{ReLU}$ or id is enough for us.

Proposition 3.6. *It holds that* $\|\mathrm{FC}^\sigma_{W,b}(x)\|_\infty \le \|W\|_0\|W\|_\infty\|x\|_\infty + \|b\|_\infty$.

Proof.

$$\begin{aligned}\|\mathrm{FC}^\sigma_{W,b}(x)\|_\infty &\le \|W\mathrm{vec}(x) - b\|_\infty \\ &\le \|W\mathrm{vec}(x)\|_\infty + \|b\|_\infty \\ &\le \max_j \sum_{\alpha,i} \left|W_{\alpha,i,j}x^{\alpha,i}\right| + \|b\|_\infty.\end{aligned}$$

The number of non-zero summand in the summation is at most $\|W\|_0$ and each summand is bounded by $\|W\|_\infty\|x\|_\infty$ Therefore, we have $\|\mathrm{FC}^\sigma_{W,b}(x)\|_\infty \le \|W\|_0\|W\|_\infty\|x\|_\infty + \|b\|_\infty$. □

Proposition 3.7. *The Lipschitz constant of* $\mathrm{FC}^{\sigma}_{W,b}$ *is bounded by* $\|W\|_0\|W\|_\infty$.

Proof. For any $x, x' \in \mathbb{R}^{D\times C}$,

$$\begin{aligned}\|\mathrm{FC}^{\sigma}_{W,b}(x) - \mathrm{FC}^{\sigma}_{W,b}(x')\|_\infty &\le \|(W\mathrm{vec}(x) - b) - (W\mathrm{vec}(x') - b)\|_\infty \\ &\le \|W(\mathrm{vec}(x) - \mathrm{vec}(x'))\|\infty \\ &\le \|W\|_0\|W\|_\infty\|\mathrm{vec}(x) - \mathrm{vec}(x')\|_\infty.\end{aligned}$$

□

Proposition 3.8. *It holds that* $\|\mathrm{FC}^{\sigma}_{W,b}(x) - \mathrm{FC}^{\sigma}_{W',b'}(x)\|_\infty \le (\|W\|_0 + \|W'\|_0)\|W - W'\|_\infty\|x\|_\infty + \|b - b'\|_\infty$.

Proof.

$$\begin{aligned}\|\mathrm{FC}^{\sigma}_{W,b}(x) - \mathrm{FC}^{\sigma}_{W',b'}(x)\|_\infty &\le \|(W\mathrm{vec}(x) - b) - (W'\mathrm{vec}(x) - b')\|_\infty \\ &= \|((W - W')\mathrm{vec}(x) - (b - b')\|_\infty \\ &\le \|(W - W')\mathrm{vec}(x)| + \|b - b'\|_\infty \\ &\le \|W - W'\|_0\|W - W'\|_\infty\|x\|_\infty + \|b - b'\|_\infty \\ &\le (\|W\|_0 + \|W'\|_0)\|W - W'\|_\infty\|x\|_\infty + \|b - b'\|_\infty\end{aligned}$$

□

Bounds for residual blocks

In this section, we denote the architecture of CNNs by $\boldsymbol{C} = (C^{(l)})_{l\in[L]} \in \mathbb{N}_+^L$ and $\boldsymbol{K} = (K^{(l)})_{l\in[L]} \in \mathbb{N}_+^L$ and the norm constraint on the convolution part by $B^{(\mathrm{conv})}$ ($C^{(0)}$ need not equal to 1 in this section). Let $w^{(l)}, w'^{(l)} \in \mathbb{R}^{K^{(l)}\times C^{(l)}\times C^{(l-1)}}$ and $b^{(l)}, b'^{(l)} \in \mathbb{R}$. We denote $\boldsymbol{w} := (w^{(l)})_{l\in[L]}$, $\boldsymbol{b} := (b^{(l)})_{l\in[L]}$, $\boldsymbol{w}' := (w'^{(l)})_{l\in[L]}$, and $\boldsymbol{b} := (b^{(l)})_{l\in[L]}$.

For $1 \le l \le l' \le L$, we denote $\rho(l, l') := \prod_{i=l}^{l'}(C^{(i-1)}K^{(i)}B^{(\mathrm{conv})})$ and $\rho^+(l, l') := \prod_{i=l}^{l'} 1 \vee (C^{(i-1)}K^{(i)}B^{(\mathrm{conv})})$.

Proposition 3.9. *Let* $l \in [L]$. *We assume* $\max_{l\in[L]} \|w^{(l)}\|_\infty \vee \|b^{(l)}\|_\infty \le B^{(\mathrm{conv})}$. *Then, for any* $x \in [-1, 1]^{D\times C^{(0)}}$, *we have* $\|\mathrm{Conv}^{\sigma}_{\boldsymbol{w}[1:l],\boldsymbol{b}[1:l]}(x)\|_\infty \le \rho(1, l)\|x\|_\infty + B^{(\mathrm{conv})}l\rho^+(1, l)$.

Proof. We write in shorthand as $C_{[s:t]} := \mathrm{Conv}^{\sigma}_{\boldsymbol{w}[s:t],\boldsymbol{b}[s:t]}$. Using Proposition 3.3 recursively, we get

$$\begin{aligned}\|C_{[1:l]}(x)\|_\infty &\le \|L^{w^{(l)}}\|_{\mathrm{op}}\|C_{[1:l-1]}(x)\|_\infty + \|b^{(l)}\|_\infty \\ &\cdots \\ &\le \|x\|_\infty \prod_{i=1}^{l} \|L^{w^{(i)}}\|_{\mathrm{op}} + \sum_{i=2}^{l} \|b^{(i-1)}\|_\infty \prod_{j=i}^{l} \|L^{w^{(j)}}\|_{\mathrm{op}} + \|b^{(l)}\|_\infty.\end{aligned}$$

By Proposition 3.2 and assumptions $\|w^{(i)}\|_\infty \leq B^{(\mathrm{conv})}$ and $\|b^{(i)}\|_\infty \leq B^{(\mathrm{conv})}$, it is further bounded by

$$\begin{aligned}&\|x\|_\infty \prod_{i=1}^{l}(C^{(i-1)}K^{(i)}B^{(\mathrm{conv})}) + B^{(\mathrm{conv})}\sum_{i=2}^{l}\prod_{j=i}^{l}(C^{(j-1)}K^{(j)}B^{(\mathrm{conv})}) + B^{(\mathrm{conv})} \\ &\leq \rho(1,l)\|x\|_\infty + B^{(\mathrm{conv})}l\rho^{+}(1,l)\end{aligned}$$

□

Proposition 3.10. *Let* $\varepsilon > 0$*, suppose* $\max_{l\in[L]}\|w^{(l)} - w'^{(l)}\|_\infty \leq \varepsilon$ *and* $\max_{l\in[L]}\|b^{(l)} - b'^{(l)}\|_\infty \leq \varepsilon$*, then* $\|C_{[1:L]} - C'_{[1:L]}(x)\|_\infty \leq (L\rho(1,L)\|x\|_\infty + (1\vee B^{(\mathrm{conv})})L^2\rho^{+}(1,L))\varepsilon$ *for any* $x \in \mathbb{R}^{D\times C^{(0)}}$.

Proof. For any $l \in [L]$, we have

$$\begin{aligned}&\left|C'_{[l+1:L]}\circ(C_l - C'_l)\circ C_{[1:l-1]}(x)\right| \\ &\leq \|C'_{[l+1:L]}\circ(C_l - C'_l)\circ C_{[1:l-1]}(x)\|_\infty \\ &\leq \rho(l+1,L)\left\|(C_l - C'_l)\circ C_{[1:l-1]}(x)\right\|_\infty \quad \text{(by Proposition 3.2 and 3.4)} \\ &\leq \rho(l+1,L)\left(\rho(l,l)\|C_{[1:l-1]}\|_\infty\varepsilon + \varepsilon\right) \quad \text{(by Proposition 3.2 and 3.5)} \\ &\leq \rho(l+1,L)\left(\rho(l,l)(\rho(1,l-1)\|x\|_\infty + B^{(\mathrm{conv})}(l-1)\rho_{+}(1,l-1)) + 1\right)\varepsilon \\ &\quad \text{(by Proposition 3.9)} \\ &= \left(\rho(1,L)\|x\|_\infty + (1\vee B^{(\mathrm{conv})})L\rho_{+}(1,L)\right)\varepsilon \end{aligned} \tag{3.6}$$

Therefore,

$$\begin{aligned}\|C_{[1:L]}(x) - C'_{[1:L]}(x)\|_\infty &\leq \sum_{l=1}^{L}\|C_{[l+1:L]}\circ(C_l - C'_l)\circ C_{[1:l-1]}(x)\|_\infty \\ &\leq (L\rho(1,L)\|x\|_\infty + (1\vee B^{(\mathrm{conv})})L^2\rho^{+}(1,L))\varepsilon\end{aligned}$$

□

Putting them all

Let $M, L_m, C_m^{(l)}, K_m^{(l)} \in \mathbb{N}_+$, $\boldsymbol{C} := (C_m^{(l)})_{m,l}$, and $\boldsymbol{K} := (K_m^{(l)})_{m,l}$ for $m \in [M]$ and $l \in [L_m]$. Let $\boldsymbol{\theta} = ((w_m^{(l)})_{m,l}, (b_m^{(l)})_{m,l}, W, b)$ and $\boldsymbol{\theta}' = ((w'^{(l)}_m)_{m,l}, (b'^{(l)}_m)_{m,l}, W', b')$ be tuples compatible with $(\boldsymbol{C},\boldsymbol{K})$ such that $\mathrm{CNN}^{\mathrm{ReLU}}_{\boldsymbol{\theta}}, \mathrm{CNN}^{\mathrm{ReLU}}_{\boldsymbol{\theta}'} \in \mathcal{F}^{(\mathrm{CNN})}_{\boldsymbol{C},\boldsymbol{K},B^{(\mathrm{conv})},B^{(\mathrm{fc})}}$ for some $B^{(\mathrm{conv})}, B^{(\mathrm{fc})} > 0$. We denote the l-th convolution layer of the m-th block by $C_m^{(l)}$ and the m-th residual block of by C_m:

$$C_m^{(l)} := \begin{cases}\mathrm{Conv}^{\mathrm{id}}_{w_m^{(l)}} & (\text{if } l = L_m) \\ \mathrm{Conv}^{\mathrm{ReLU}}_{w_m^{(l)}} & (\text{otherwise})\end{cases}$$

$$C_m := C_m^{(L_m)} \circ \cdots \circ C_m^{(1)}.$$

Also, we denote by $C_{[m:m']}$ the subnetwork of $\mathrm{Conv}_{\boldsymbol{\theta}}^{\mathrm{ReLU}}$ between the m-th and m'-th block. That is,

$$C_{[m:m']} := \begin{cases} (C_{m'} + I) \circ \cdots \circ (C_m + I) & (\text{if } m \geq 1) \\ (C_{m'} + I) \circ \cdots \circ (C_1 + I) \circ P & (\text{if } m = 0) \end{cases}$$

for $m, m' = 0, \ldots, M$. We define $C'^{(l)}_m$, C'_m and $C'_{[m:m']}$ similarly for $\boldsymbol{\theta}'$.

Proposition 3.11. *For $m \in [M]$ and $x \in [-1,1]^D$, we have $\|C_{[0:m]}(x)\|_\infty \leq (1 \vee B^{(\mathrm{conv})})\varrho_m \varrho_m^+$. Here, $\varrho_m = \left(\prod_{i=1}^m (1+\rho_i)\right)$ and $\varrho_m^+ = \left(1 + \sum_{i=1}^m L_i \rho_i^+\right)$ (ρ_m and ρ_m^+ are constants defined in Theorem 3.6).*

Proof. By using Proposition 3.9 inductively, we have

$$\begin{aligned} \|C_{[0:m]}(x)\|_\infty &\leq \|C_m(C_{[0:m-1]}(x)) + C_{[0:m-1]}(x)\|_\infty \\ &\leq \|(1+\rho_m)C_{[0:m-1]}(x) + B^{(\mathrm{conv})} L_m \rho_{+m})\|_\infty \\ &\leq (1+\rho_m)\|C_{[0:m-1]}(x)\|_\infty + B^{(\mathrm{conv})} L_m \rho_m^+ \\ &\cdots \\ &\leq \|P(x)\|_\infty \prod_{i=1}^m (1+\rho_i) + B^{(\mathrm{conv})} \sum_{i=1}^m L_i \rho_i^+ \prod_{j=i+1}^m (1+\rho_j) \\ &\leq \prod_{i=1}^m (1+\rho_i) + B^{(\mathrm{conv})} \sum_{i=1}^m L_i \rho_i^+ \prod_{j=i+1}^m (1+\rho_j) \\ &\leq (1 \vee B^{(\mathrm{conv})}) \varrho_m \varrho_m^+. \end{aligned}$$

□

Lemma 3.3. *Let $\varepsilon > 0$. Suppose $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ are within distance ε, that is, $\max_{m,l} \|w_m^{(l)} - w'^{(l)}_m\|_\infty \leq \varepsilon$, $\|b_m^{(l)} - b'^{(l)}_m\|_\infty \leq \varepsilon$, $\|W - W'\|_\infty \leq \varepsilon$, and $\|b - b'\|_\infty \leq \varepsilon$. Then, $\|\mathrm{CNN}_{\boldsymbol{\theta}}^{\mathrm{ReLU}} - \mathrm{CNN}_{\boldsymbol{\theta}'}^{\mathrm{ReLU}}\|_\infty \leq \Lambda_1 \varepsilon$ where Λ_1 is the constant defined in Theorem 3.6.*

Proof. For any $x \in [-1,1]^D$, we have

$$\begin{aligned} &\left|\mathrm{CNN}_{\boldsymbol{\theta}}^{\mathrm{ReLU}}(x) - \mathrm{CNN}_{\boldsymbol{\theta}'}^{\mathrm{ReLU}}(x)\right| \\ &= \left|\mathrm{FC}_{W,b}^{\mathrm{id}} \circ C_{[0:M]}(x) - \mathrm{FC}_{W',b'}^{\mathrm{id}} \circ C'_{[0:M]}(x)\right| \\ &= \left|\left(\mathrm{FC}_{W,b}^{\mathrm{id}} - \mathrm{FC}_{W',b'}^{\mathrm{id}}\right) \circ C_{[0:M]}(x)\right| + \sum_{m=1}^M \left|\mathrm{FC}_{W',b'}^{\mathrm{id}} \circ C_{[m+1:M]} \circ \left(C_m - C'_m\right) \circ C'_{[0:m-1]}(x)\right|. \end{aligned} \tag{3.7}$$

We will bound each term of (3.7). By Proposition 3.8 and Proposition 3.11,

$$\left|\left(\mathrm{FC}_{W,b}^{\mathrm{id}} - \mathrm{FC}_{W',b'}^{\mathrm{id}}\right) \circ C_{[0:M]}(x)\right|$$

$$\begin{aligned}
&\leq (\|W\|_0 + \|W'\|_0)\|W - W'\|_\infty \|C_{[0:M]}(x)\|_\infty + \|b - b'\|_\infty \\
&\leq 2C_0^{(L_0)} D \|C_{[0:M]}(x)\|_\infty \varepsilon + \varepsilon \\
&\leq 2C_0^{(L_0)} D (1 \vee B^{(\mathrm{conv})}) \varrho_M \varrho_M^+ \varepsilon + \varepsilon \\
&\leq 3C_0^{(L_0)} D (1 \vee B^{(\mathrm{conv})}) \varrho_M \varrho_M^+ \varepsilon. \qquad (3.8)
\end{aligned}$$

On the other hand, for $m \in [M]$,

$$\begin{aligned}
&\left|\mathrm{FC}^{\mathrm{id}}_{W',b'} \circ C'_{[m+1:M]} \circ (C_m - C'_m) \circ C_{[0:m-1]}(x)\right| \\
&\leq \|W'\|_0 \|W'\|_\infty \|C'_{[m+1:M]} \circ (C_m - C'_m) \circ C_{[1:m-1]}(x)\|_\infty \quad \text{(by Proposition 3.7)} \\
&\leq C_0^{(L_0)} D B^{(\mathrm{fc})} \|C'_{[m+1:M]} \circ (C_m - C'_m) \circ C_{[0:m-1]}(x)\|_\infty \\
&\leq C_0^{(L_0)} D B^{(\mathrm{fc})} \left(\prod_{i=m+1}^{M} \rho_i\right) \left\|(C_m - C'_m) \circ C_{[0:m-1]}(x)\right\|_\infty \quad \text{(by Proposition 3.2 and 3.4)} \\
&\leq C_0^{(L_0)} D B^{(\mathrm{fc})} \left(\prod_{i=m+1}^{M} \rho_i\right) \left(\rho_m \|C_{[0:m-1]}\|_\infty \varepsilon + \varepsilon\right) \quad \text{(by Proposition 3.2 and 3.5)} \\
&\leq C_0^{(L_0)} D B^{(\mathrm{fc})} \left(\prod_{i=m+1}^{M} \rho_i\right) \left(\rho_m (1 \vee B^{(\mathrm{conv})}) \varrho_{m-1} \varrho_{m-1}^+ + 1\right) \varepsilon \quad \text{(by Proposition 3.9)} \\
&\leq 2C_0^{(L_0)} D B^{(\mathrm{fc})} (1 \vee B^{(\mathrm{conv})}) \varrho_M \varrho_M^+ \varepsilon \qquad (3.9)
\end{aligned}$$

By applying (3.8) and (3.9) to (3.7), we have

$$\begin{aligned}
&|\mathrm{CNN}^{\mathrm{ReLU}}_{\boldsymbol{\theta}}(x) - \mathrm{CNN}^{\mathrm{ReLU}}_{\boldsymbol{\theta'}}(x)| \\
&\leq 3C_0^{(L_0)} D (1 \vee B^{(\mathrm{conv})}) \varrho_M \varrho_M^+ \varepsilon + 2MC_0^{(L_0)} D B^{(\mathrm{fc})} (1 \vee B^{(\mathrm{conv})}) \varrho_M \varrho_M^+ \varepsilon \\
&\leq (2M+3) C_0^{(L_0)} D (1 \vee B^{(\mathrm{fc})})(1 \vee B^{(\mathrm{conv})}) \varrho_M \varrho_M^+ \varepsilon \\
&= \Lambda_1 \varepsilon.
\end{aligned}$$

□

Bounds for covering number of CNNs

For a metric space $(\mathcal{M}_0, d)$ and $\varepsilon > 0$, we denote the (external) covering number of $\mathcal{M} \subset \mathcal{M}_0$ by $\mathcal{N}(\varepsilon, \mathcal{M}, d)$: $\mathcal{N}(\varepsilon, \mathcal{M}, d) := \inf\{N \in \mathbb{N} \mid \exists f_1, \ldots, f_N \in \mathcal{M}_0 \text{ s.t. } \forall f \in \mathcal{M}, \exists n \in [N] \text{ s.t. } d(f, f_n) \leq \varepsilon\}$.

Lemma 3.4. *Let* $B := B^{(\mathrm{conv})} \vee B^{(\mathrm{fc})}$. *For* $\varepsilon > 0$, *we have* $\mathcal{N}(\varepsilon, \mathcal{F}^{(\mathrm{CNN})}, \|\cdot\|_\infty) \leq \left(2B\Lambda_1\varepsilon^{-1}\right)^{\Lambda_2}$.

Proof. The idea of the proof is same as that of Lemma 5 of Schmidt-Hieber [2020]. We divide the interval of each parameter range ($[-B^{(\mathrm{conv})}, B^{(\mathrm{conv})}]$ or $[-B^{(\mathrm{fc})}, B^{(\mathrm{fc})}]$) into bins with width $\Lambda_1^{-1}\varepsilon$ (i.e., $2B^{(\mathrm{conv})}\Lambda_1\varepsilon^{-1}$ or $2B^{(\mathrm{fc})}\Lambda_1\varepsilon^{-1}$ bins for each interval). If $f, f' \in \mathcal{F}^{(\mathrm{CNN})}$ can be realized by parameters such that every pair of corresponding parameters are in a same bin, then, $\|f - f'\|_\infty \leq \varepsilon$ by Lemma 3.3. We make a subset $\mathcal{F}_0$ of $\mathcal{F}^{(\mathrm{CNN})}$ by picking up every combination of bins for Λ_2 parameters. Then, for each $f \in \mathcal{F}^{(\mathrm{CNN})}$, there exists $f_0 \in \mathcal{F}_0$ such that $\|f - f_0\|_\infty \leq \varepsilon$. There are at most $2B\Lambda_1\varepsilon^{-1}$ choices of bins for each parameter. Therefore, the cardinality of $\mathcal{F}_0$ is at most $\left(2B\Lambda_1\varepsilon^{-1}\right)^{\Lambda_2}$. □

Proofs of Theorem 3.2 and Corollary 3.1

We use the lemma in Schmidt-Hieber [2020] to bound the estimation error of the clipped ERM estimator $\hat{f}$. Since our problem setting is slightly different from one in the paper, we restate the statement.

Lemma 3.5 (cf. Schmidt-Hieber [2020] Lemma 4)**.** *Let $\mathcal{F}$ be a family of measurable functions from $[-1,1]^D$ to $\mathbb{R}$. Let $\hat{f}$ be the clipped ERM estimator of the regression problem described in Section 3.3.1. Suppose the covering number of $\mathcal{F}$ satisfies $\mathcal{N}(N^{-1}, \mathcal{F}, \|\cdot\|_\infty) \geq 3$. Then,*

$$\mathbb{E}_{\mathcal{D}}\|f^\circ - \hat{f}\|^2_{\mathcal{L}^2(\mathcal{P}_X)} \leq C\left(\inf_{f\in\mathcal{F}}\|f - f^\circ\|^2_{\mathcal{L}^2(\mathcal{P}_X)} + \log\mathcal{N}\left(\frac{1}{N}, \mathcal{F}, \|\cdot\|_\infty\right)\frac{\tilde{F}^2}{N}\right),$$

where $C > 0$ is a universal constant, $\tilde{F} := \frac{R_{\mathcal{F}}}{\sigma} \vee \frac{\|f^\circ\|_\infty}{\sigma} \vee \frac{1}{2}$ and $R_{\mathcal{F}} := \sup\{\|f\|_\infty \mid f \in \mathcal{F}\}$.

Proof. Basically, we convert our problem setting so that it fits to the assumptions of Schmidt-Hieber [2020, Lemma 4] and apply the lemma to it. For $f : [-1,1]^D \to [-\sigma\tilde{F}, \sigma\tilde{F}]$, we define $A[f] : [0,1]^D \to [0, 2\tilde{F}]$ by $A[f](x') := \frac{1}{\sigma}f(2x'-1) + \tilde{F}$. Let $\hat{f}_1$ be the (non-clipped) ERM etimator of $\mathcal{F}$. We define $X' := \frac{1}{2}(X+1)$, $f'^\circ := A[f^\circ]$, $Y' := f'^\circ(X) + \xi'$, $\mathcal{F}' := \{A[f] \mid f \in \mathcal{F}\}$, $\hat{f}'_1 := A[\hat{f}_1]$, and $\mathcal{D}' := ((x'_n, y'_n))_{n\in[N]}$ where $x'_n := \frac{1}{2}(x_n + 1)$ and $y'_n := f'^\circ(x'_n) + \frac{1}{\sigma}(y_n - f^\circ(x_n))$. Then, the probability that $\mathcal{D}'$ is drawn from $\mathcal{P}'^{\otimes N}$ is same as the probability that $\mathcal{D}$ is drawn from $\mathcal{P}^{\otimes N}$ where $\mathcal{P}'$ is the joint distribution of (X', Y'). Also, we can show that $\hat{f}'$ is the ERM estimator of the regression problem $Y' = f'^\circ + \xi'$ using the dataset $\mathcal{D}'$: $\hat{f}'_1 \in \arg\min_{f'\in\mathcal{F}'}\hat{\mathcal{R}}_{\mathcal{D}'}(f')$. We apply Schmidt-Hieber [2020, Lemma 4] with $n \leftarrow N$, $d \leftarrow D$, $\varepsilon \leftarrow 1$, $\delta \leftarrow \frac{1}{N}$, $\Delta_n \leftarrow 0$, $\mathcal{F}' \leftarrow \mathcal{F}$, $F \leftarrow 2\tilde{F}$, $\hat{f} \leftarrow \hat{f}'_1$ and use the fact that the estimation error of the clipped ERM estimator is no worse than that of the ERM estimator, that is, $\|f^\circ - \hat{f}\|^2_{\mathcal{L}^2(\mathcal{P}_X)} \leq \|f^\circ - \hat{f}_1\|^2_{\mathcal{L}^2(\mathcal{P}_X)}$ to conclude. □

Proof of Theorem 3.6. By Lemma 3.4, we have $\log\mathcal{N} := \log\mathcal{N}(N^{-1}, \mathcal{F}^{(\mathrm{CNN})}, \|\cdot\|_\infty) \leq \Lambda_2\log(2B\Lambda_1 N)$, where $B = B^{(\mathrm{conv})} \vee B^{(\mathrm{fc})}$. Therefore, by Lemma 3.5,

$$\begin{aligned}\|f^\circ - \hat{f}\|^2_{\mathcal{L}^2(\mathcal{P}_X)} &\leq C_0\left(\inf_{f\in\mathcal{F}}\|f - f^\circ\|^2_{\mathcal{L}^2(\mathcal{P}_X)} + \log\mathcal{N}\frac{\tilde{F}^2}{N}\right)\\ &\leq C_1\left(\inf_{f\in\mathcal{F}^{(\mathrm{FNN})}}\|f - f^\circ\|^2_\infty + \frac{\tilde{F}^2}{N}\Lambda_2\log(2B\Lambda_1 N)\right),\end{aligned}$$

where $C_0, C_1 > 0$ are universal constants. We used in the last inequality the fact $\|\mathrm{clip}[f] - f^\circ\|_{\mathcal{L}^2(\mathcal{P}_X)} \leq \|\mathrm{clip}[f] - f^\circ\|_\infty \leq \|f - f^\circ\|_\infty$ any $f \in \mathcal{F}^{(\mathrm{CNN})}$ and the assumption $\mathcal{F}^{(\mathrm{FNN})} \subset \mathcal{F}^{(\mathrm{CNN})}$. □

As discussed in the beginning of this section, Theorem 3.2 is the special case of Theorem 3.6.

Proof of Corollay 3.1. We only care the order with respect to N in the O-notation. Set $M = \lfloor N^{\alpha} \rfloor$ for $\alpha > 0$. Using the assumptions of the corollary, the estimation error is

$$\|f^{\circ} - \hat{f}\|^2_{\mathcal{L}^2(\mathcal{P}_x)} = \tilde{O}\left(\max\left(N^{-2\alpha\gamma_1}, N^{\alpha\gamma_2 - 1}\right)\right)$$

by Theorem 3.2. The order of the right hand side with respect to N is minimized when $\alpha = \frac{1}{2\gamma_1 + \gamma_2}$. By substituting α, we can derive Corollary 3.1. □

3.A.4 Proofs of Corollary 3.2 and Corollary 3.3

By Klusowski and Barron [2018, Theorem 2], for each $M \in \mathbb{N}_+$, there exists

$$f^{(\mathrm{FNN})} := \frac{1}{M}\sum_{m=1}^{M} b_m (a_m^{\top} x - t_m)_+ = \sum_{m=1}^{M} b_m \left(\frac{a_m^{\top}}{M} x - \frac{t_m}{M}\right)_+$$

with $|b_m| \leq 1$, $\|a_m\|_1 = 1$, and $|t_m| \leq 1$ such that $\|f^{\circ} - f^{(\mathrm{FNN})}\|_{\infty} \leq C v_{f^{\circ}} \sqrt{\log M + D} M^{-\frac{1}{2} - \frac{1}{D}}$ where $v_{f^{\circ}} := \int_{\mathbb{R}^D} \|w\|_2^s |\mathcal{F}[f^{\circ}](w)| \, \mathrm{d}w$ and $C > 0$ is a universal constant. We set $L_m \leftarrow 1$, $D_m^{(1)} \leftarrow 1$, $B^{(\mathrm{bs})} \leftarrow \frac{1}{M}$, $B^{(\mathrm{fin})} \leftarrow 1$ ($m \in [M]$) in the Theorem 3.5, then, we have $f^{(\mathrm{FNN})} \in \mathcal{F}^{(\mathrm{FNN})}_{\boldsymbol{D}_1, B^{(\mathrm{bs})}, B^{(\mathrm{fin})}}$. By applying Theorem 3.5, there exists a CNN $f^{(\mathrm{CNN})} \in \mathcal{F}^{(\mathrm{CNN})}_{\boldsymbol{C}, \boldsymbol{K}, B^{(\mathrm{conv})}, B^{(\mathrm{fc})}}$ such that $f^{(\mathrm{FNN})} = f^{(\mathrm{CNN})}$. Here, $\boldsymbol{C} = (C_m^{(1)})_m$ with $C_m^{(1)} = 4$, $\boldsymbol{K} = (K_m^{(1)})_m$ with $K_m^{(1)} = K$, $B^{(\mathrm{conv})} = \frac{1}{M}$, and $B^{(\mathrm{fc})} = M$. This proves Corollary 3.2.

With these evaluations, we have $\Lambda_1 = O(M^3)$ (note that since $B^{(\mathrm{conv})} = \frac{1}{M}$, we have $\prod_{m=0}^{M}(1 + \rho_m) = O(1)$). In addition, $B^{(\mathrm{conv})}$ is $O(1)$ and $B^{(\mathrm{fc})}$ is $O(M)$. Therefore, we have $\log \Lambda_1 B = \tilde{O}(1)$. Since $\Lambda_2 = O(M)$, we can use Corollary 3.1 with $\gamma_1 = \frac{1}{2} + \frac{1}{D}$, $\gamma_2 = 1$. □

3.A.5 Proofs of Corollary 3.4 and Corollary 3.5

We first prove the scaling property of the FNN class.

Lemma 3.6. *Let $M \in \mathbb{N}_+$, $L_m \in \mathbb{N}_+$, and $D_m^{(l)} \in \mathbb{N}_+$ for $m \in [M]$ and $l \in [L_m]$. Let $B^{(\mathrm{bs})}, B^{(\mathrm{fin})} > 0$. Then, for any $k \geq 1$, we have $\mathcal{F}^{(\mathrm{FNN})}_{\boldsymbol{D}, B^{(\mathrm{bs})}, B^{(\mathrm{fin})}} \subset \mathcal{F}^{(\mathrm{FNN})}_{\boldsymbol{D}, k^{-1}B^{(\mathrm{bs})}, k^{L}B^{(\mathrm{fin})}}$ where $L := \max_{m \in [M]} L_m$ is the maximum depth of the blocks.*

Proof. Let $\boldsymbol{\theta} = ((W_m^{(l)})_{m,l}, (b_m^{(l)})_{m,l}, (w_m)_m, b)$ be the parameter of an FNN and suppose that $\mathrm{FNN}^{\mathrm{ReLU}}_{\boldsymbol{\theta}} \in \mathcal{F}^{(\mathrm{FNN})}_{\boldsymbol{D}, B^{(\mathrm{bs})}, B^{(\mathrm{fin})}}$. We define $\boldsymbol{\theta}' := ((W'^{(l)}_m)_{m,l}, (b'^{(l)}_m)_{m,l}, (w'_m)_m, b')$ by

$$W'^{(l)}_m := k^{-\frac{L}{L_m}} W_m^{(l)}, \quad b'^{(l)}_m := k^{-l\frac{L}{L_m}} b_m^{(l)}, \quad w'_m := k^L w_m, \quad b' := b.$$

Since $k \geq 1$, we have $\mathrm{FNN}^{\mathrm{ReLU}}_{\boldsymbol{\theta}'} \in \mathcal{F}^{(\mathrm{FNN})}_{\boldsymbol{D}, k^{-1}B^{(\mathrm{bs})}, k^{L}B^{(\mathrm{fin})}}$. Also, by the homogeneous property of the ReLU function (i.e., $\mathrm{ReLU}(ax) = a\mathrm{ReLU}(x)$ for $a > 0$), we have $\mathrm{FNN}^{\mathrm{ReLU}}_{\boldsymbol{\theta}} = \mathrm{FNN}^{\mathrm{ReLU}}_{\boldsymbol{\theta}'}$. □

Next, we prove the existence of a block-sparse FNN with constant-width blocks that optimally approximates a given β-Hölder function. It is almost same as the proof appeared in Schmidt-Hieber [2020]. However, we need to construct the FNN so that it has a block-sparse structure.

Lemma 3.7 (cf. Schmidt-Hieber [2020] Theorem 5)**.** *Let $\beta > 0$, $M \in \mathbb{N}_+$ and $f^\circ : [-1,1]^D \to \mathbb{R}$ be a β-Hölder function. Then, there exists $D' = O(1)$, $L' = O(\log M)$, and a block-sparse FNN $f^{(\mathrm{FNN})} \in \mathcal{F}^{(\mathrm{FNN})}_{\boldsymbol{D},1,2M\|f^\circ\|_\beta}$ such that $\|f^\circ - f^{(\mathrm{FNN})}\|_\infty = \tilde{O}(M^{-\frac{\beta}{D}})$. Here, we set $L_m := L'$ and $D_m^{(l)} := D'$ for all $m \in [M]$ and $l \in [L_m]$ and define $\boldsymbol{D} := (D_m^{(l)})_{m,l}$.*

Proof. First, we prove the lemma when the domain of f° is $[0,1]^D$. Let M' be the largest interger satisfying $(M'+1)^D \leq M$. Let $\Gamma(M') = \left(\frac{\mathbb{Z}}{M'}\right)^D \cap [0,1]^D = \{\frac{m'}{M'} \mid m' \in \{0,\ldots,M'\}^D\}$ be the set of lattice points in $[0,1]^D$[5]. Note that the cardinality of $\Gamma(M')$ is $(M'+1)^D$. Let $P_a^\beta f^\circ$ be the Taylor expansion of f° up to order $\lfloor \beta \rfloor$ at $a \in [0,1]^D$:

$$(P_a^\beta f^\circ)(x) = \sum_{0 \leq |\alpha| < \beta} \frac{(\partial^\alpha f^\circ)(a)}{\alpha!}(x-a)^\alpha.$$

For $a \in [0,1]^D$, we define a hat-shaped function $H_a : [0,1]^D \to [0,1]$ by

$$H_a(x) := \prod_{j=1}^D ({M'}^{-1} - |x_j - a_j|_+).$$

Note that we have $\sum_{a \in \Gamma(M')} H_a(x) = 1$, i.e., they are a partition of unity. Let $P^\beta f^\circ$ be the weighted sum of the Taylor expansions at lattice points of $\Gamma(M')$:

$$(P^\beta f^\circ)(x) := {M'}^D \sum_{a \in D(M')} (P_a^\beta f^\circ)(x) H_a(x).$$

By Schmidt-Hieber [2020, Lemma B.1], we have

$$\|P^\beta f^\circ - f^\circ\|_\infty \leq \|f^\circ\|_\beta {M'}^{-\beta}.$$

Let m be an interger specified later and set $L^* := (m+5)\lceil \log_2 D \rceil$. By the proof of Schmidt-Hieber [2020, Lemma B.2], for any $a \in \Gamma(M')$, there exists an FNN $\mathrm{Hat}_a : [0,1]^D \to [0,1]$ whose depth and width are at most $2 + L^*$ and $6D$, respectively and whose parameters have sup-norm 1, such that

$$\|\mathrm{Hat}_a - H_a\|_\infty \leq 3^D 2^{-m}.$$

Next, let $B := 2\|f^\circ\|_\beta$ and $C_{D,\beta}$ be the number of distinct D-variate monomials of degree up to $\lfloor \beta \rfloor$. By Schmidt-Hieber [2020, Equation (7.11)], for any $a \in \Gamma(M)$, there exists an FNN

[5] Schmidt-Hieber [2020] used $\boldsymbol{D}(M')$ to denote this set of lattice points. We used different character to avoid notational conflict.

$Q_a : [0,1]^D \to [0,1]$ [6] whose depth and width are $1 + L^*$ and $6DC_{D,\beta}$ respectively and whose parameters have sup-norm 1, such that

$$\left\| Q_a - \left(\frac{P_a^\beta f^\circ}{B} + \frac{1}{2} \right) \right\|_\infty \leq 3^D 2^{-m}.$$

Thirdly, by Schmidt-Hieber [2020, Lemma A.2], there exists an FNN $\mathrm{Mult} : [0,1]^2 \to [0,1]$, whose depth and width are $m+4$ and 6, respectively and whose parameters have sup-norm 1 such that

$$|\mathrm{Mult}(x,y) - xy| \leq 2^{-m}$$

for any $x, y \in [0,1]$. For each $a \in \Gamma(M')$, we combine Hat_a and Q_a using Mult and constitute a block of the block-sparse FNN corresponding to $a \in \Gamma(M)$ by $\mathrm{FC}_a := \mathrm{Mult}(Q_a(\cdot), \mathrm{Hat}_a(\cdot))$. Then, we have

$$\left\| \mathrm{FC}_a - \left(\frac{P_a^\beta f^\circ}{B} + \frac{1}{2} \right) H_a \right\|_\infty \leq 2^{-m} + 3^D 2^{-m} + 3^D 2^{-m} \leq 3^{D+1} 2^{-m}.$$

We define $f^{(\mathrm{FNN})}(x) := \sum_{a \in \Gamma(M)} (B{M'}^D \mathrm{FC}_a(x)) - \frac{B}{2}$. By construction, $f^{(\mathrm{FNN})}$ is a block-sparse FNN with $(M'+1)^D (\leq M)$ blocks each of which has depth and width at most $L' := 2 + L^* + (m+4)$ and $D' := 6(C_{D,\beta}+1)D$, respectively. The norms of the block-sparse part and the finally fully-connected layer are 1 and $B{M'}^D (\leq BM)$, respectively. In addition, we have

$$\begin{aligned}
&|f^{(\mathrm{FNN})}(x) - (P^\beta f^\circ)(x)| \\
&\leq \sum_{a \in \Gamma(M)} B{M'}^D \left| \mathrm{FC}_a(x) - \left(\frac{(P_a^\beta f^\circ)(x)}{B} + \frac{1}{2} \right) H_a(x) \right| + \frac{B}{2} \left| 1 - {M'}^D \sum_{a \in \Gamma(M')} H_a(x) \right| \\
&\leq (M'+1)^D \times B{M'}^D 3^{D+1} 2^{-m} \\
&\leq 3^{D+1} 2^{-m} BM^2
\end{aligned}$$

for any $x \in [0,1]^D$. Therefore,

$$\begin{aligned}
|f^{(\mathrm{FNN})}(x) - f^\circ(x)| &\leq |f^{(\mathrm{FNN})} - (P^\beta f^\circ)(x)| + |(P^\beta f^\circ)(x) - f^\circ(x)| \\
&\leq 3^{D+1} 2^{-m} BM^2 + \|f^\circ\|_\beta {M'}^{-\beta} \\
&\leq 2 \cdot 3^{D+1} 2^{-m} \|f^\circ\|_\beta M^2 + \|f^\circ\|_\beta M^{-\frac{\beta}{D}}.
\end{aligned}$$

We set $m = \lceil \log_2 M^{2+\frac{\beta}{D}} \rceil$, then, we have $L' = O(\log M)$, $D' = O(1)$, and

$$\|f^{(\mathrm{FNN})} - f^\circ\| \leq \|f^\circ\|_\beta (2 \cdot 3^{D+1} + 2^\beta) M^{-\frac{\beta}{D}}.$$

[6] We prepare Q_a for each $a \in \Gamma(M)$ as opposed to the original proof of Schmidt-Hieber [2020], in which Q_a's shared the layers the except the final one and were collectively denoted by Q_1.

By the defnition of $f^{(\mathrm{FNN})}$ we have $f^{(\mathrm{FNN})} \in \mathcal{F}^{(\mathrm{FNN})}_{\boldsymbol{D},1,2\|f^\circ\|_\beta M}$.

When the domain of f° is $[-1,1]^D$, we should add the function $x \mapsto \frac{1}{2}(x+1) = \frac{1}{2}(x+1)_+ - \frac{1}{2}(-x-1)_+$ as a first layer of each block to fit the range into $[0,1]^D$. Specifically, suppose the first layer of m-th block in $f^{(\mathrm{FNN})}$ is $x \mapsto \mathrm{ReLU}(Wx-b)$, then the first two layers become $x \mapsto \mathrm{ReLU}(\begin{bmatrix}\frac{1}{2}(x+1) & -\frac{1}{2}(x+1)\end{bmatrix})$ and $\begin{bmatrix}y_1 & y_2\end{bmatrix} \mapsto \mathrm{ReLU}(Wy_1 - Wy_2 - b)$, respectively. Since this transformation does not change the maximum sup norm of parameters in the block-sparse and the order of L' and D', the resulting FNN still belongs to $\mathcal{F}^{(\mathrm{FNN})}_{\boldsymbol{D},1,2\|f^\circ\|M}$. □

Proofs of Corollary 3.4 and Corollary 3.5. In this proof, we only care the dependence on M in the O-notation. Let $\tilde{M} := 2\|f^\circ\|_\beta M$. By Lemma 3.7, there exists $f^{(\mathrm{FNN})} \in \mathcal{F}^{(\mathrm{FNN})}_{\boldsymbol{D},1,\tilde{M}}$ such that $\|f^{(\mathrm{FNN})} - f^\circ\|_\infty = O(M^{-\frac{\beta}{D}})$ (L', D', and $\boldsymbol{D}$ as in Lemma 3.7). Let $k := 16D'K(M^{\frac{1}{L'}} \wedge 1)^{-1} = 16D'K(e^{\frac{1}{C'}} \wedge 1)^{-1} \geq 1$ where C' is a constant such that $L' = C' \log M$. Using Lemma 3.6, there exists $\tilde{f}^{(\mathrm{FNN})} \in \mathcal{F}^{(\mathrm{FNN})}_{\boldsymbol{D},k^{-1},k^{L'}\tilde{M}}$ such that $\tilde{f}^{(\mathrm{FNN})} = f^{(\mathrm{FNN})}$. We apply Theorem 3.5 to $\mathcal{F}^{(\mathrm{FNN})}_{\boldsymbol{D},k^{-1},k^{L'}\tilde{M}}$ and find $f^{(\mathrm{CNN})} \in \mathcal{F}^{(\mathrm{CNN})}_{\boldsymbol{C},\boldsymbol{K},B^{(\mathrm{conv})},B^{(\mathrm{fc})}}$ such that $L \leq M(L'+L_0)$, $\boldsymbol{C} := (C_m^{(l)})_{m\in[M],l\in[L_m]}$ with $C_m^{(l)} \leq 4D'$, $\boldsymbol{K} := (K_m^{(l)})_{m\in[M],l\in[L_m]}$ with $K_m^{(l)} \leq K$, $B^{(\mathrm{conv})} = k^{-1}$, $B^{(\mathrm{fc})} = k^{L'}(k \vee 1)\tilde{M} = k^{L'+1}\tilde{M}$, and $f^{(\mathrm{CNN})} = \tilde{f}^{(\mathrm{FNN})}$. This proves Corollary 3.4 (note that by definition, we have $B^{(\mathrm{conv})} = k^{-1} = O(1)$ and $\log B^{(\mathrm{fc})} = (L'+1)k + \log(\tilde{M}) = O(\log M)$).

By the definition of k and the bound on $C_m^{(l)}$ and $K_m^{(l)}$, we have $C_m^{(l-1)}K_m^{(l)}k^{-1} \leq \frac{1}{4}M^{-\frac{1}{L'}}$. Therefore, we have $\rho_m \leq \prod_{l=1}^{L'}(C_m^{(l-1)}K_m^{(l)}k^{-1}) \leq M^{-1}$ and hence $\prod_{m=0}^{M}(1+\rho_m) = O(1)$. Since $C_m^{(l-1)}K_m^{(l)}k^{-1} \leq \frac{1}{2}$ for sufficiently large M, we have $\rho_m^+ = 1$ for sufficiently large M. In addition, we have $\log(B^{(\mathrm{conv})} \vee B^{(\mathrm{fc})}) = \tilde{O}(1)$. Combining them, we have $\log \Lambda_1 = \tilde{O}(1)$ and hence $\log \Lambda_1(B^{(\mathrm{conv})} \vee B^{(\mathrm{fc})}) = \tilde{O}(1)$. For Λ_2, we can bound it by $\Lambda_2 = O(M \log M)$ using bounds for $C_m^{(l)}$, $K_m^{(l)}$ and L'. Therefore, we can apply Corollary 3.2 with $\gamma_1 = \frac{\beta}{D}$, $\gamma_2 = 1$ and obtain the desired estimation error. Since we have $M = O(N^{\frac{1}{2\gamma_1+\gamma_2}})$ by the proof of Corollary 3.1, we can derive the bounds for L_m with respect to N. □

3.A.6 Proofs of Theorem 3.3 and Theorem 3.4

Lemma 3.8. *Let $L, L', C', K' \in \mathbb{N}_+$ and $B > 0$. Suppose we can realize $f + \mathrm{id} : \mathbb{R}^{D\times C'} \to \mathbb{R}^{D\times C'}$ with a residual block with an skip connection whose depth, channel size, and filter size are L', C', and K', respectively and whose parameter norm is bounded by B. Let $S_0 = \lceil \frac{L'}{L} \rceil$. Then, there exist $S = 2S_0 - 1$ functions $\tilde{f}_1, \ldots, \tilde{f}_S : \mathbb{R}^{D\times 3C'} \to \mathbb{R}^{D\times 3C'}$ and S masks $z_1, \ldots, z_S \in \{0,1\}^{3C'}$, such that f_s is realizable by a residual block whose depth, channel size, filter size, and parameter norm bound are L, $3C'$, K', and B, respectively and $\tilde{f} := (\tilde{f}_S + J_S) \circ \cdots \circ (\tilde{f}_1 + J_1) : \mathbb{R}^{D\times 3C'} \to \mathbb{R}^{D\times 3C'}$ satisfies $\tilde{f}(\begin{bmatrix}x & 0 & 0\end{bmatrix}) = \begin{bmatrix}f(x) & 0 & 0\end{bmatrix}$. Here J_s is a channel-wise mask operation made from z_s.*

Proof. We divide the residual block representing f into S_0 CNNs with depth at most L and denote them sequentially by $g_1, \ldots, g_{S_0}$ so that $f = g_{S_0} \circ \cdots \circ g_1$. We define $\tilde{g}_s : \mathbb{R}^{D\times 3C'} \to \mathbb{R}^{D\times 3C'}$

($s \in [S_0]$) from g_s by

$$\tilde{g}_s([x_1\ x_2\ x_3]) = \begin{cases} [0\ y_1\ 0] \text{ (if } s = 1) \\ [0\ y_3\ 0] \text{ (if } s \neq 1, S_0 \text{ and odd)} \\ [0\ 0\ y_2] \text{ (if } s \neq 1, S_0 \text{ and even)} \\ [y_3\ 0\ 0] \text{ (if } s = S_0 \text{ and odd)} \\ [y_2\ 0\ 0] \text{ (if } s = S_0 \text{ and even)} \end{cases},$$

where $y_i = g_s(x_i)$ $(i = 1, 2, 3)$. Note that we can construct $\tilde{g}_s$ by a residual block with depth L, channel size $3C'$, filter size K', and parameter norm B. Next, we define u_s ($s \in [S_0 - 1]$) by

$$u_s = \begin{cases} \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^\top & \text{(if } s\text{: odd)} \\ \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}^\top & \text{(if } s\text{: even)} \end{cases}$$

Then, we define $\tilde{f} := (\tilde{g}_{S_0} + \mathrm{id}) \circ (0 + J'_{S_0-1}) \circ (\tilde{g}_{S_0-1} + \mathrm{id}) \circ (0 + J'_1) \circ (\tilde{f}_1 + \mathrm{id})$ where J'_s is a channel-wise mask constructed from u_s and $0 : \mathbb{R}^{D\times 3C'} \to \mathbb{R}^{D\times 3C'}$ is a constant zero function, which is obviously representable by a residual block. By definition, $\tilde{f}$ is realizable by S residual blocks with channel-wise masking skip connections and satisfy the conditions on the depth, channel size, filter size, and norm bound. □

Proof of Theorem 3.3. The first part of the proof is same as that of Corollary 3.4, except that we define k using L instead of L': $k = 16D'K(M^{\frac{1}{L}} \wedge 1)^{-1}$. Here, D' is a constant satisfying $D' = O(1)$ as a function of M. Then, there exists a CNN $\tilde{f}^{(\mathrm{CNN})} \in \mathcal{F}^{(\mathrm{CNN})}_{M,L',C',K',B^{(\mathrm{conv})},B^{(\mathrm{fin})}}$ such that $\|\tilde{f}^{(\mathrm{CNN})} - f^\circ\| = O(M^{-\frac{\beta}{D}})$. Parameter of the set of CNNs satisfy $L' = O(\log M)$ $C' \leq 4D'$, $K' \leq K$, $B^{(\mathrm{conv})} = k^{-1}$, and $B^{(\mathrm{fc})} = 2\|f^\circ\|_\beta k^{L'} M$. We apply Lemma 3.8 to each residual block of $\tilde{f}^{(\mathrm{CNN})}$. Then, there exists $f^{(\mathrm{CNN})} \in \mathcal{G}_{\tilde{M},L,C,K,B^{(\mathrm{conv})},B^{(\mathrm{fin})}}$ such that $f^{(\mathrm{CNN})} = \tilde{f}^{(\mathrm{CNN})}$ and $\tilde{M} = M\lceil \frac{L'}{L} \rceil$, $C \leq 3C'$, $K' \leq K$, $B^{(\mathrm{conv})} = k^{-1}$, and $B^{(\mathrm{fin})} = 2\|f^\circ\|_\beta k^{L'+1} M$. □

Before going to the proof of Theorem 3.4, we first note that the definitions of Λ_1 and Λ_2 in Theorem 3.2 are valid even if we replace $\mathcal{F}^{(\mathrm{CNN})}_{\tilde{M},L,C,K,B^{(\mathrm{conv})},B^{(\mathrm{fin})}}$ with $\mathcal{G} = \mathcal{G}_{\tilde{M},L,C,K,B^{(\mathrm{conv})},B^{(\mathrm{fin})}}$.

Lemma 3.9. *Let $\tilde{M}, L, C, K \in \mathbb{N}_+$ and $B^{(\mathrm{conv})}, B^{(\mathrm{fin})}, \varepsilon > 0$. Set $B = B^{(\mathrm{conv})} \vee B^{(\mathrm{fin})}$. Then, the covering number of $\mathcal{G}$ with respect to the sup-norm $\mathcal{N}(\varepsilon, \mathcal{G}, \|\cdot\|_\infty)$ is bounded by $(2B\Lambda_1\varepsilon^{-1})^{\Lambda_2} \cdot 2^{C\tilde{M}L}$, where $\Lambda_1 = \Lambda_1(\mathcal{G})$ and $\Lambda_2 = \Lambda_2(\mathcal{G})$ are ones defined in Theorem 3.2, except that $\mathcal{F}^{(\mathrm{CNN})}$ is replaced with $\mathcal{G}$.*

Proof. First we note that we can apply same inequalities in Section 3.A.3 – 3.A.3 and Proposition 3.11 to CNNs in $\mathcal{G}$. Therefore, if two masked CNNs $f, g \in \mathcal{G}$ have same masking patterns in skip connections and distance of each pair of corresponding parameters in residual blocks is at most ε, then, we can show $\|f - g\|_\infty \leq \Lambda_1\varepsilon$ in the same way as Lemma 3.3. Therefore, by the same argument of Lemma 3.4, the covering number of the subset of $\mathcal{G}$ consisting of CNNs with a specific masking pattern is bounded by $(2B\Lambda_1\varepsilon^{-1})^{\Lambda_2}$. Since each CNN in $\mathcal{G}$ has $C\tilde{M}L$ parameters in skip connections which take values in $\{0,1\}$, there are $2^{C\tilde{M}L}$ masking patterns. Therefore, we have $\mathcal{N}(\varepsilon, \mathcal{G}, \|\cdot\|_\infty) \leq (2B\Lambda_1\varepsilon^{-1})^{\Lambda_2} \cdot 2^{C\tilde{M}L}$. □

The strategy for the proof of Theorem 3.4 is almost same as the proofs for Theorem 3.6 and Corollary 3.5, except that we should replace $\Lambda_2 \log(2B\Lambda_1 N)$ in (3.5) with $\Lambda_2 \log(2B\Lambda_1 N) + C\tilde{M}L\log 2$ (and Λ_1 and Λ_2 are defined via $\mathcal{G}$ instead of $\mathcal{F}^{(\mathrm{CNN})}$). However, the second term is at most as same order (upto logarithmic factors) as the first one in our situation. Therefore, we can derive the same estimation error rate.

Proof of Theorem 3.4. Take $\mathcal{G}$ as in the proof of Theorem 3.3. Let $\log\mathcal{N} := \log\mathcal{N}(N^{-1}, \mathcal{G}, \|\cdot\|_\infty)$. By Lemma 3.5, we have

$$\|f^\circ - \hat{f}\|^2_{\mathcal{L}^2(\mathcal{P}_X)} \leq C_0 \left(\inf_{f \in \mathcal{F}^{(\mathrm{FNN})}} \|f - f^\circ\|^2_\infty + \frac{\tilde{F}^2}{N} \left(\Lambda_2 \log(2B\Lambda_1 N) + C\tilde{M}L\log 2 \right) \right),$$

where $C_0 > 0$ is a universal constant. The first term in the outer-most parenthesis is $O(M^{-\frac{\beta}{D}})$ by Lemma 3.7. We will evaluate the order of the second term. First, we have $\Lambda_2 = O(\tilde{M}) = \tilde{O}(M)$ by the definition of Λ_2. By the definition of k, we have $\rho \leq M^{-1}$ and $\rho^+ = 1$ for sufficiently large M therefore, $\varrho = O(1)$ and $\varrho^+ = O(M)$ for sufficiently large M. Again, by the definition of k, we have $B^{(\mathrm{conv})} = O(1)$ and $B^{(\mathrm{fc})} = O(M)$. Therefore, we have $\Lambda_1 = O(M^3)$ and $B = O(M)$ and hence $\Lambda_2 \log(2B\Lambda_1 N) = \tilde{O}(MN)$. On the other hand, since $C = O(1), \tilde{M} = \tilde{O}(M), L = O(1)$, we have $C\tilde{M}L\log 2 = \tilde{O}(M)$.

Therefore, by setting $M = \lfloor N^\alpha \rfloor$ for $\alpha > 0$, the estimation error is

$$\|f^\circ - \hat{f}\|^2_{\mathcal{L}^2(\mathcal{P}_x)} = \tilde{O}\left(\max\left(N^{-2\alpha\gamma_1}, N^{\alpha\gamma_2 - 1}\right)\right),$$

where $\gamma_1 = \frac{\beta}{D}$ and $\gamma_2 = 1$. The order of the right hand side with respect to N is minimized when $\alpha = \frac{1}{2\gamma_1 + \gamma_2}$. By substituting α, we can derive the theorem. □

Chapter 4

Over-smoothing of Non-linear Graph Neural Networks

In this chapter, we focus on node aggregation operations in Graph Neural Networks (GNNs). GNNs are a promising deep learning approach for analyzing graph-structured data. However, it is known that they do not improve (or sometimes worsen) their predictive performance as we pile up many layers and add non-lineality. To tackle this problem, we investigate the expressive power of GNNs via their asymptotic behaviors as the layer size tends to infinity. Our strategy is to generalize the forward propagation of a GNN as a specific discrete-time dynamical system. In the case of Graph Convolutional Networks (GCN), which is a popular GNN variant, we show that when its weights satisfy the conditions determined by the spectra of the (augmented) normalized Laplacian, its output exponentially approaches the set of signals that carry information of the connected components and node degrees only for distinguishing nodes. Our theory enables us to relate the expressive power of GCNs with the topological information of the underlying graphs inherent in the graph spectra. To demonstrate this, we characterize the asymptotic behavior of GCNs on the Erdős – Rényi graph. We show that when the Erdős – Rényi graph is sufficiently dense and large, a broad range of GCNs on it suffers from the “information loss" in the limit of infinite layers with high probability. Based on the theory, we provide a principled guideline for weight normalization of GNNs. We experimentally confirm that the proposed weight scaling enhances the predictive performance of GCNs in real data. Code is available at `https://github.com/delta2323/gnn-asymptotics`.

4.1 Introduction

Motivated by the success of Deep Learning (DL), several attempts have been made to apply DL models to non-Euclidean data, particularly, graph-structured data such as chemical compounds, social networks, and polygons. Recently, *Graph Neural Networks* (GNNs) [Duvenaud et al., 2015, Li et al., 2016, Gilmer et al., 2017, Hamilton et al., 2017, Kipf and Welling, 2017, Nguyen et al., 2017, Schlichtkrull et al., 2018, Battaglia et al., 2018, Xu et al., 2019, Wu et al., 2019a] have emerged as a promising approach. However, despite their practical popularity, theoretical research of GNNs has not been explored extensively.

The characterization of DL model *expressive power*, i.e., to identify what function classes DL models can (approximately) represent, is a fundamental question in theoretical research of DL. Many studies have been conducted for Fully Connected Neural Networks (FNNs) [Cybenko, 1989, Hornik, 1991, Hornik et al., 1989, Barron, 1993, Mhaskar, 1993, Sonoda and Murata, 2017, Yarotsky, 2017] and Convolutional Neural Networks (CNNs) [Petersen and Voigtlaender, 2018a, Zhou, 2018, Oono and Suzuki, 2019]. For such models, we have theoretical and empirical justification that deep and non-linear architectures can enhance representation power [Telgarsky, 2016, Chen et al., 2018b, Zhou and Feng, 2018]. However, for GNNs, several papers have reported that node representations go indistinguishable (known as *over-smoothing*) and prediction performances severely degrade when we stack many layers [Kipf and Welling, 2017, Wu et al., 2019b, Li et al., 2018b]. Besides, Wu et al. [2019a] reported that GNNs achieved comparable performance even if they removed intermediate non-linear functions. These studies posed a question about the current architecture and made us aware of the need for the theoretical analysis of the GNN expressive power.

In this chapter, we investigate the expressive power of GNNs by analyzing their asymptotic behaviors as the layer size goes to infinity. Our theory gives new theoretical conditions under which neither layer stacking nor non-linearity contributes to improving expressive power. We consider a specific dynamics that includes a transition defining a Markov process and the forward propagation of a Graph Convolutional Network (GCN) [Kipf and Welling, 2017], which is one of the most popular GNN variants, as special cases. We prove that under certain conditions, the dynamics exponentially approaches a subspace that is invariant under the dynamics. In the case of GCN, the invariant space is a set of signals that correspond to the lowest frequency of graph spectra and that have “no information" other than connected components and node degrees for a node classification task whose goal is to predict the nodes' properties in a graph. The rate of the distance between the output and the invariant space is $O((s\lambda)^L)$ where s is the maximum singular values of weights, λ is typically a quantity determined by the spectra of the (augmented) normalized Laplacian, and L is the layer size. See Sections 4.4.1 (general case) and 4.5 (GCN case) for precise statements.

We can interpret our theorem as the generalization of the well-known property that if a finite and discrete Markov process is irreducible and aperiodic, it exponentially converges to a unique equilibrium and the eigenvalues of its transition matrix determine the convergence rate (see, e.g., Chung and Graham [1997]). Different from the Markov process case, which is linear, the existence of intermediate non-linear functions complicates the analysis. We overcame this problem by leveraging the combination of the ReLU activation function [Krizhevsky et al., 2012, 2017] and the positivity of eigenvectors of the Laplacian associated with the smallest positive eigenvalues.

Our theory enables us to investigate asymptotic behaviors of GNNs via the spectral distribution of the underlying graphs. To demonstrate this, we take GCNs defined on the Erdős – Rényi graph $G_{N,p}$, which has N nodes and each edge appears independently with probability p, for an example. We prove that if $\frac{\log N}{pN} = o(1)$ as a function of N, any GCN whose weights have maximum singular values at most $C\sqrt{\frac{Np}{\log(N/\varepsilon)}}$ approaches the “information-less" invariant space with probability at least $1-\varepsilon$, where C is a universal constant. Intuitively, if the graph on which we define GNNs is sufficiently dense, graph-convolution operations mix signals on nodes fast and hence the feature maps lose information for distinguishing nodes quickly.

Our contributions in this chapter are as follows:

- We relate asymptotic behaviors of GNNs with the topological information of underlying graphs via the spectral distribution of the (augmented) normalized Laplacian.
- We prove that if the weights of a GCN satisfy conditions determined by the graph spectra, the output of the GCN carries no information other than the node degrees and connected components for discriminating nodes when the layer size goes to infinity (Theorems 4.1, 4.2).
- We apply our theory to Erdős – Rényi graphs as an example and show that when the graph is sufficiently dense and large, many GCNs suffer from the information loss (Theorem 4.3).
- We propose a principled guideline for weight normalization of GNNs and empirically confirm it using real data.

4.2 Related Work

MPNN-type GNNs Since many GNN variants have been proposed, there are several unified formulations of GNNs [Gilmer et al., 2017, Battaglia et al., 2018]. Our approach is the closest to the formulation of Message Passing Neural Network (MPNN) [Gilmer et al., 2017], which unified GNNs in terms of the update and readout operations. Many GNNs fall into this formulation such as Duvenaud et al. [2015], Li et al. [2016], and Veličković et al. [2018]. Among others, GCN [Kipf and Welling, 2017] is an important application of our theory because it is one of the most widely used GNNs. In addition, GCNs are interesting from a theoretical research perspective because, in addition to an MPNN-type GNN, we can interpret GCNs as a simplification of spectral-type GNNs [Henaff et al., 2015, Defferrard et al., 2016], that make use of the graph Laplacian.

Our approach, which considers the asymptotic behaviors GNNs as the layer size goes to infinity, is similar to Scarselli et al. [2009], one of the earliest works about GNNs. They obtained node representations by iterating message passing between nodes until convergence. Their formulation is general in that we can use any local aggregation operation as long as it is a *contraction map*. Our theory differs from theirs in that we proved that the output of a GNN approaches a certain space even if the local aggregation function is not necessarily a contraction map.

Expressive Power of GNNs Several studies have focused on theoretical analysis and the improvement of GNN expressive power. For example, Xu et al. [2019] proved that GNNs are no more powerful than the Weisfeiler – Lehman (WL) isomorphism test [Weisfeiler and A.A., 1968] and proposed a Graph Isomorphism Network (GIN), that is approximately as powerful as the WL test. Although they experimentally showed that GIN has improved accuracy in supervised learning tasks, their analysis was restricted to the graph isomorphism problem. Xu et al. [2018] analyzed the non-asymptotic properties of GCNs through the lens of random walk theory. They proved the limitations of GCNs in expander-like graphs and proposed a Jumping Knowledge Network (JK-Net) to address the issue. To handle the non-linearity, they linearized networks by a randomization assumption [Choromanska et al., 2015]. We take a different strategy and make use of the interpretation of ReLU as a projection onto a cone. Recently, NT and Maehara [2019] showed that a GCN approximately works as a low-pass filter plus an MLP in a certain setting. Although they analyzed finite-depth GCNs, our theory has similar spirits with theirs because our "information-less" space corresponds to the lowest frequency of a graph Laplacian. Another point is that they imposed assumptions that input signals consist of low-frequent true signals and high-frequent noise, whereas we need not such an assumption.

Role of Deep and Non-linear Structures For ordinal DL models such as FNNs and CNNs, we have both theoretical and empirical justification of deep and non-linear architectures for enhancing of the expressive power (e.g., Telgarsky [2016], Petersen and Voigtlaender [2018a], Oono and Suzuki [2019]). In contrast, several studies have witnessed severe performance degradation when stacking many layers on GNNs [Kipf and Welling, 2017, Wu et al., 2019b]. Li et al. [2018b] reported that feature vectors on nodes in a graph go indistinguishable as we increase layers in several tasks. They named this phenomenon *over-smoothing*. Regarding non-linearity, Wu et al. [2019a] empirically showed that GNNs achieve comparable performance even if we omit intermediate non-linearity. These observations gave us questions about the current models of deep GNNs in terms of their expressive power. Several studies gave theoretical explanations of the over-smoothing phenomena for *linear* GNNs [Li et al., 2018b, Zhang, 2019, Zhao and Akoglu, 2020]. We can think of our theory as an extension of their results to non-linear GNNs.

4.3 Problem Settings

Although we are mainly interested in GCNs, we develop our theory more generally using dynamical systems. We will specialize to the GCNs in Section 4.5.

For $N, C, H_l \in \mathbb{N}_+$ ($l \in \mathbb{N}_+$), let $P \in \mathbb{R}^{N\times N}$ be a symmetric matrix and $W_{lh} \in \mathbb{R}^{C\times C}$ for $l \in \mathbb{N}_+$ and $h \in [H_l]$. We define $f_l : \mathbb{R}^{N\times C} \to \mathbb{R}^{N\times C}$ by $f_l(X) := \mathrm{MLP}_l(PX)$. Here, $\mathrm{MLP}_l : \mathbb{R}^{N\times C} \to \mathbb{R}^{N\times C}$ is the l-th multi-layer perceptron common to all nodes [Xu et al., 2019] and is defined by $\mathrm{MLP}_l(X) := \sigma(\cdots\sigma(\sigma(X)W_{l1})W_{l2}\cdots W_{lH_l})$, where $\sigma : \mathbb{R}^{N\times C} \to \mathbb{R}^{N\times C}$ is an element-wise ReLU function [Krizhevsky et al., 2012, 2017] defined by $\sigma(X)_{nc} := \max(X_{nc}, 0)$ for $n \in [N], c \in [C]$. We consider the dynamics $X^{(l+1)} := f_l(X^{(l)})$ with some initial value $X^{(0)} \in \mathbb{R}^{N\times C}$. We are interested in the asymptotic behavior of $X^{(l)}$ as $l \to \infty$.

For $M \le N$, let U be a M-dimensional subspace of $\mathbb{R}^N$. We assume that U and P satisfy the following properties that generalize the situation where U is the eigenspace associated with the smallest eigenvalue of a (normalized) graph Laplacian Δ (that is, zero) and P is a polynomial of Δ.

Assumption 4.1. *U has an orthonormal basis $(e_m)_{m\in[M]}$ that consists of non-negative vectors.*

Assumption 4.2. *U is invariant under P, i.e., if $u \in U$, then $Pu \in U$.*

We endow $\mathbb{R}^N$ with the ordinal inner product and denote the orthogonal complement of U by $U^\perp := \{u \in \mathbb{R}^N \mid \langle u, v\rangle = 0, \forall v \in U\}$. By the symmetry of P, we can show that $U^\perp$ is invariant under P, too as we see in the following proposition (see Section 4.A.1 for the proof).

Proposition 4.1. *Let $P \in \mathbb{R}^{N\times N}$ be a symmetric matrix, treated as a linear operator $P : \mathbb{R}^N \to \mathbb{R}^N$. If a subspace $U \subset \mathbb{R}^N$ is invariant under P (i.e., if $u \in U$, then $Pu \in U$), then, $U^\perp$ is invariant under P, too.*

Therefore, we can regard $P|_{U^\perp}$ as a linear mapping from $U^\perp$ to itself: $P|_{U^\perp} : U^\perp \to U^\perp$. We denote the operator norm of $P|_{U^\perp}$ by λ. When U is the eigenspace associated with the smallest eigenvalue of Δ and P is $g(\Delta)$ where g is a polynomial, then, λ corresponds to $\lambda = \sup_\mu |g(\mu)|$ where sup ranges over all eigenvalues except the smallest one.

4.4 Main Results

4.4.1 Convergence of Dynamical System

We define the subspace $\mathcal{M}$ of $\mathbb{R}^{N\times C}$ by $\mathcal{M} := U \otimes \mathbb{R}^C = \{\sum_{m=1}^M e_m \otimes w_m \mid w_m \in \mathbb{R}^C\}$ where $(e_m)_{m\in[M]}$ is the orthonormal basis of U appeared in Assumption 4.1. For $X \in \mathbb{R}^{N\times C}$, we denote the distance between X and $\mathcal{M}$ by $d_\mathcal{M}(X) := \inf\{\|X - Y\|_\mathrm{F} \mid Y \in \mathcal{M}\}$. We denote the maximum singular value of W_{lh} by s_{lh} and set $s_l := \prod_{h=1}^{H_l} s_{lh}$. With these preparations, we introduce the main theorem of the paper.

Theorem 4.1. *Under Assumptions 4.1 and 4.2, we have $d_\mathcal{M}(f_l(X)) \le s_l \lambda d_\mathcal{M}(X)$ for any $l \in \mathbb{N}_+$ and $X \in \mathbb{R}^{N\times C}$.*

The proof key is that the non-linear operation σ decreases the distance $d_\mathcal{M}$, that is, $d_\mathcal{M}(\sigma(X)) \le d_\mathcal{M}(X)$. We use the non-negativity of e_m to prove this claim. See Appendix 4.A.2 for the complete proof. We discuss the strictness of Theorem 4.1 in Section 4.4.3.

By setting $d_\mathcal{M}(X) = 0$, this theorem implies that $\mathcal{M}$ is invariant under f_l. In addition, if the maximum value of singular values are small, $X^{(l)}$ asymptotically approaches $\mathcal{M}$ in the sense of Johnson [1973] for any initial value $X^{(0)}$. That is, the following corollaries hold under Assumptions 4.1 and 4.2.

Corollary 4.1. *$\mathcal{M}$ is invariant under f_l for any $l \in \mathbb{N}_+$, that is, if $X \in \mathcal{M}$, then we have $f_l(X) \in \mathcal{M}$.*

Corollary 4.2. *Let $s := \sup_{l\in\mathbb{N}_+} s_l$. We have $d_\mathcal{M}(X^{(l)}) = O((s\lambda)^l)$. In particular, if $s\lambda < 1$, then X_l exponentially approaches $\mathcal{M}$ as $l \to \infty$ for any initial value $X^{(0)}$.*

Suppose the operator norm of $P|_U : U \to U$ is no larger than λ, then, under the assumption of $s\lambda < 1$, $X^{(l)}$ converges to 0, the trivial fixed point (see Proposition 4.2 below) Therefore, we are mainly interested in the case where the operator norm of $P|_U$ is strictly larger than λ (see Proposition 4.1). Finally, we restate Theorem 4.1 specialized to the situation where U is the direct sum of eigenspaces associated with the largest M eigenvalues of P. Note that the eigenvalues of P is real since P is symmetric.

Corollary 4.3. *Let $\lambda_1 \le \cdots \le \lambda_N$ be the eigenvalue of P, sorted in ascending order. Suppose the multiplicity of the largest eigenvalue λ_N is $M(\le N)$, i.e., $\lambda_{N-M} < \lambda_{N-M+1} = \cdots = \lambda_N$. We define $\lambda := \max_{n\in[N-M]} |\lambda_n|$ and U by the eigenspace associated with λ_N. Then, we have $d_\mathcal{M}(X^{(l+1)}) \le s_l \lambda d_\mathcal{M}(X^{(l)})$.*

4.4.2 Convergence to Trivial Fixed Point

Let $P \in \mathbb{R}^{N \times N}$ be a symmetric matrix, $W_l \in \mathbb{R}^{C \times C}$, s_l be the maximum singular value of W_l for $l \in \mathbb{N}_+$. We define $f_l : \mathbb{R}^{N \times C} \to \mathbb{R}^{N \times C}$ by $f_l(X) := \sigma(PXW_l)$ where σ is the element-wise ReLU function. The following proposition shows that the dynamical system converges to a trivial fixed point when weight scales are too small. See Appendix 4.A.3 for the proof.

Proposition 4.2. *Suppose that the operator norm of P is no larger than λ, then we have $\|f_l(X)\|_{\mathrm{F}} \leq s_l \lambda \|X\|_{\mathrm{F}}$ for any $l \in \mathbb{N}_+$. In particular, let $s := \sup_{l \in \mathbb{N}_+} s_l$. If $s\lambda < 1$, then, X_l exponentially approaches 0 as $l \to \infty$.*

4.4.3 Strictness of Main Theorem

Theorem 4.1 implies that if $s\lambda \leq 1$, then, one-step transition f_l does not increase the distance to $\mathcal{M}$. In this section, we first prove that this theorem is strict in the sense that, there exists a situation in which $s_l \lambda > 1$ holds and the distance $d_{\mathcal{M}}$ increases by one-step transition f_l at some point X. See Appendix 4.A.4 for the proof of the propositions in this section.

Set $N \leftarrow 2$, $C \leftarrow 1$, and $M \leftarrow 1$ in Section 4.3. For $\mu, \lambda > 0$, we set

$$P \leftarrow \begin{bmatrix} \mu & 0 \\ 0 & \lambda \end{bmatrix}, e \leftarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix}, U \leftarrow \left\{ \begin{bmatrix} x \\ y \end{bmatrix} \mid y = 0 \right\}.$$

Then, by definition, we can check that the 3-tuple (P, e, U) satisfies the Assumptions 4.1 and 4.2. Set $\mathcal{M} := U \otimes \mathbb{R} = U$ and choose $W \in \mathbb{R}$ so that $W > \lambda^{-1}$. Finally define $f : \mathbb{R}^{N \times C} \to \mathbb{R}^{N \times C}$ by $f(X) := \sigma(PXW)$ where σ is the element-wise ReLU function.

Proposition 4.3. *We have $d_{\mathcal{M}}(f(X)) > d_{\mathcal{M}}(X)$ for any $X = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top \in \mathbb{R}^2$ such that $x_2 > 0$.*

Next, we prove the non-strictness of Theorem 4.1 in the sense that there exists a situation in which $s_l \lambda > 1$ holds and the distance $d_{\mathcal{M}}$ uniformly decreases by f_l. Again, we set Set $N \leftarrow 2$, $C \leftarrow 1$, and $M \leftarrow 1$. Let $\lambda \in (1, 2)$ and set

$$P \leftarrow \frac{\lambda}{2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, e \leftarrow \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, U \leftarrow \left\{ \begin{bmatrix} x \\ y \end{bmatrix} \mid x = y \right\}$$

Then, we can directly show that 3-tuple (P, e, U) satisfies the Assumptions 4.1 and 4.2. Set $W \leftarrow 1$.

Proposition 4.4. *We have $W\lambda > 1$ and $d_{\mathcal{M}}(f_l(X)) < d_{\mathcal{M}}(X)$ for all $X \in \mathbb{R}^2$.*

We have shown that the non-negativity of e (Assumption 4.1) is *not* a redundant condition in Section 4.6.1.

4.4.4 Relation to Markov Process

It is known that any Markov process on finite states converges to a unique distribution (*equilibrium*) if it is irreducible and aperiodic (see e.g., Norris [1998]). Theorem 4.1 includes this proposition as a special case with $M = 1$, $C = 1$, and $W_l = 1$ for all $l \in \mathbb{N}_+$. This is essentially the direct consequence of Perron – Frobenius' theorem (see e.g., Meyer [2000]).

Let $S := \{1, \ldots, N\}$ be a finite discrete state space. Consider a Markov process on S characterized by a symmetric transition matrix $P = (p_{ij})_{i,j \in [N]} \in \mathbb{R}^{N \times N}$ such that $P \geq 0$ and $P\mathbf{1} = \mathbf{1}$ where $\mathbf{1}$ is the all-one vector. We interpret p_{ij} as the transition probability from a state i to j. We associate P with a graph $G_P = (V_P, E_P)$ by $V_P = [N]$ and $(i, j) \in E_P$ if and only if $p_{ij} > 0$. Since P is symmetric, we can regard G_P as an undirected graph. We assume P is irreducible and aperiodic [1]. Perron – Frobenius' theorem (see, e.g., Meyer [2000]) implies that P satisfy the assumption of Corollary 4.3 with $M = 1$.

Fact 4.1 (Perron – Frobenius). *Let the eigenvalues of P be $\lambda_1 \leq \cdots \leq \lambda_N$. Then, we have $-1 < \lambda_1$, $\lambda_{N-1} < 1$, and $\lambda_N = 1$. Further, there exists unique vector $e \in \mathbb{R}^N$ such that $e \geq 0$, $\|e\| = 1$, and e is the eigenvector for the eivenvalue 1.*

Corollary 4.4. *Let $\lambda := \max_{n=1,\ldots,N-1} |\lambda_n| (< 1)$ and $\mathcal{M} := \{e \otimes w \mid w \in \mathbb{R}^C\}$. If $s\lambda < 1$, then, for any initial valueX_1, X_l exponentially approaches $\mathcal{M}$ as $l \to \infty$.*

If we set $C = 1$ and $W_l = 1$ for all $l \in \mathbb{N}_+$, then, we can inductively show that $X_l \geq 0$ for any $l \geq 2$. Therefore, we can interpret X_l as a measure on S. Suppose further that we take the initial value X_1 as $X_1 \geq 0$ and $X_1^\top \mathbf{1} = 1$ so that we can interpret X_1 as a probability distribution on S. Then, we can inductively show that $X_l \geq 0$, $X_l^\top \mathbf{1} = 1$ (i.e., X_l is a probability distribution on S), and $X_{l+1} = \sigma(PX_lW_l) = PX_l$ for all $l \in \mathbb{N}_+$. In conclusion, the corollary is reduced to the fact that if a finite and discrete Markov process is irreducible and aperiodic, any initial probability distribution converges exponentially to an equibrilium. In addition, the the rate λ corresponds to the *mixing time* of the Markov process.

4.5 Application to Graph Neural Networks

4.5.1 GCNs

We formulate a GCN [Kipf and Welling, 2017] without readout operations [Gilmer et al., 2017] using the dynamical system in the previous section and derive a sufficient condition in terms of the spectra of underlying graphs in which layer stacking nor non-linearity are not helpful for node classification.

Let $G = (V, E)$ be an undirected graph where V is a set of nodes and E is a set of edges. We denote the number of nodes in G by $N = |V|$ and identify V with $[N]$ by fixing an order of V. We associate a C dimensional signal to each node. X in the previous section corresponds to concatenation of the signals. GCNs iteratively update signals on V using the connection information and weights.

Let $A := (\mathbf{1}_{\{(i,j) \in E\}})_{i,j \in [N]} \in \mathbb{R}^{N \times N}$ be the adjacency matrix and $D := \mathrm{diag}(\deg(i)_{i \in [N]}) \in \mathbb{R}^{N \times N}$ be the degree matrix of G where $\deg(i) := |\{j \in V \mid (i, j) \in E\}|$ is the degree of node

[1]A symmetric matrix A is called *irreducible* if and only if G_A is connected. We say a graph G is *aperiodic* if the greatest common divisor of length of all loops in G is 1. A symmetric matrix A is aperiodic if the graph G_A induced by A is aperiodic.

i. Let $\tilde{A} := A + I_N$, $\tilde{D} := D + I_N$ be the adjacent and degree matrix of graph G augmented with self-loops. We define the *augmented* normalized Laplacian [Wu et al., 2019a] of G by $\tilde{\Delta} := I_N - \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ and set $P := I_N - \tilde{\Delta}$. Let $L, C \in \mathbb{N}_+$ be the layer and channel sizes, respectively. For weights $W_l \in \mathbb{R}^{C\times C}$ ($l \in [L]$), we define a GCN associated with G by $f = f_L \circ \cdots \circ f_1$ where $f_l : \mathbb{R}^{N\times C} \to \mathbb{R}^{N\times C}$ is defined by $f_l(X) := \sigma(PXW_l)$[2]. We are interested in the asymptotic behavior of the output $X^{(L)}$ of the GCN as $L \to \infty$.

Suppose G has M connected components and let $V = V_1 \sqcup \cdots \sqcup V_M$ be the decomposition of the node set V into connected components. We denote an indicator vector of the m-th connected component by $u_m := (\mathbf{1}_{\{n\in V_m\}})_{n\in[N]} \in \mathbb{R}^N$. The following proposition shows that GCN satisfies the assumption of Corollay 4.3 (see Appendix 4.A.6 for proof).

Proposition 4.5. *Let $\lambda_1 \le \cdots \le \lambda_N$ be the eigenvalue of P sorted in ascending order. Then, we have $-1 < \lambda_1$, $\lambda_{N-M} < 1$, and $\lambda_{N-M+1} = \cdots = \lambda_N = 1$. In particular, we have $\lambda := \max_{n=1,\ldots,N-M} |\lambda_n| < 1$. Further, $e_m := \tilde{D}^{\frac{1}{2}}u_m$ for $m \in [M]$ are the basis of the eigenspace associated with the eigenvalue* 1.

Theorem 4.2. *For any initial value $X^{(0)}$, the output of l-th layer $X^{(l)}$ satisfies $d_{\mathcal{M}}(X^{(l)}) \le (s\lambda)^l d_{\mathcal{M}}(X^{(0)})$. In particular, $d_{\mathcal{M}}(X^{(l)})$ exponentially converges to* 0 *when $s\lambda < 1$.*

In the context of node classification tasks, we can interpret this corollary as the "information loss" of GCNs in the limit of infinite layers. For two nodes $i, j \in V$, we denote $i \sim j$ if nodes i and j are in a same connected component and their degrees are identica. For any $X \in \mathcal{M}$, if $i \sim j$, then, we have $X_i = X_j$, that is, the column vectors of X corresponding to nodes i and j are identical. It means that we cannot distinguish these nodes using X. In this sense, $\mathcal{M}$ only has information about connected components and node degrees and we can interpret this theorem as the exponential information loss of GCNs in terms of the layer size. Similarly to the discussion in the previous section, $X^{(l)}$ converges to the trivial fixed point 0 when $s < 1$ (remember $\lambda_N = 1$). An interesting point is that even if $s \ge 1$, $X^{(l)}$ can suffer from this information loss.

We note that the rate $s\lambda$ in Theorem 4.2 depends on the spectra of the augmented normalized Laplacian, which is determined by the topology of the underlying graph G. Hence, our result explicitly relates the topological information of graphs and asymptotic behaviors of GNNs.

4.5.2 Asymptotic Behavior of GCNs on Erdős – Rényi Graphs

Theorem 4.2 gives us a way to characterize the asymptotic behaviors of GCNs via the spectral distributions of the underlying graphs. To demonstrate this, we consider an Erdős – Rényi graph $G_{N,p}$ [Erdös and Rényi, 1959, Gilbert, 1959], which is a random graph that has N nodes and whose edges between two distinct nodes appear independently with probability $p \in [0, 1]$, as an example. First, consider a (non-random) graph G with M connected components. Let $0 = \tilde{\mu}_1 = \cdots = \tilde{\mu}_M < \tilde{\mu}_{M+1} \le \cdots \le \tilde{\mu}_N < 2$ be eigenvalues of the augmented normalized Laplacian of G (see, Proposition 4.5) and set $\lambda := \min_{m=M+1,\ldots,N} |1 - \tilde{\mu}_m| (< 1)$. By Theorem 4.2, the output of GCN "loses information" as the layer size goes to infinity when the largest singular values of

[2]Following the original paper [Kipf and Welling, 2017], we use one-layer MLPs (i.e., $H_l = 1$ for all $l \in \mathbb{N}_+$.). However, our result holds for the multi-layer case

weights are strictly smaller than λ^{-1}. Therefore, the closer the positive eigenvalues μ_m are to 1, the broader range of GCNs satisfies the assumption of Theorem 4.2.

For an Erdős – Rényi graph $G_{N,p}$, Chung and Radcliffe [2011] showed that when $\frac{\log N}{Np} = o(1)$, the eigenvalues of the (usual) normalized Laplacian except for the smallest one converge to 1 with high probability (see Theorem 2 therein)[3]. We can interpret this theorem as the convergence of Erdős-Rényi graphs to the complete graph in terms of graph spectra. We can prove that the augmented normalized Laplacian behaves similarly (Lemma 4.6). By combining this fact with the discussion in the previous paragraph, we obtain the asymptotic behavior of GCNs on the Erdős – Rényi graph. See Appendix 4.A.5 for the complete proof.

Theorem 4.3. *Consider a GCN on the Erdős-Rényi graph $G_{N,p}$ such that $\frac{\log N}{Np} = o(1)$ as a function of N. For any $\varepsilon > 0$, if the supremum s of the maximum singular values of weights in the GCN satisfies $s < s_0 := \frac{1}{7}\sqrt{\frac{Np-p+1}{\log(4N/\varepsilon)}}$, then, for sufficiently large N, the GCN satisfies the condition of Theorem 4.2 with probability at least $1 - \varepsilon$.*

Theorem 4.3 requires that an underlying graph is not extremely sparse. For example, suppose the node size is $N = 20,000$, which is the approximately the maximum node size of datasets we use in experiments, and the edge probability is $p = \log N/N$. Then, each node has the order of $Np \approx 4.3$ adjacent nodes.

Under the condition of Theorem 4.3, the upper bound $s_0 \to \infty$ as $N \to \infty$. It means that if the graph is sufficiently large and not extremely sparse, most GCNs suffer from the information loss. For the dependence on the edge probability p, s_0 is an increasing function of p, which means the denser a graph is, the more quickly graph convolution operations mix signals on nodes and move them close to each other.

Theorem 4.3 implies that GNNs perform poorly on dense NNs. More aggressively, we can hypothesize that the sparsity of practically available graphs is one of the reasons for the success of GNNs in node classification tasks. To confirm this hypothesis, we artificially add edges to citation networks to make them dense in the experiments and observe the failure of GNNs as expected (see Section 4.6.3).

4.5.3 GCNs Defined by Normalized Laplacian

In Section 4.5.1, we defined P using the augmented normalized Laplacian $\tilde{\Delta}$ by $P = I_N - \tilde{\Delta}$. We can alternatively use the usual normalized Laplacian Δ instead of the augmented one to define P and apply the theory developed in Section 4.3. We write the normalized Laplacian version as $P_\Delta := I_N - \Delta$. The only obstacle is that the smallest eigenvalue λ_1 of P_Δ can be equal to -1, while that of P is strictly larger than -1 (see, Proposition 4.5). This corresponds to that fact the largest eigenvalue of $\tilde{\Delta}$ is strictly smaller than 2, while that for Δ can be 2. It is known that the largest eigenvalue of Δ is 2 if and only if the graph has a non-trivial bipartite connected component (see, e.g., Chung and Graham [1997]). Therefore, we can develop a theory using the normalized Laplacian instead of the augmented one in parallel for such a graph G.

In Section 4.5.2, we characterized the asymptotic behavior of GCN defined by the augmented normalized Laplacian via its spectral distribution (see also Lemma 4.6 of Appendix 4.A.5). We can

[3]Chung et al. [2004] and Coja-Oghlan [2007] proved similar theorems.

derive a similar claim for GCN defined via the normalized Laplacian using the original theorem for the normalized Laplacian in Chung and Radcliffe [2011] (Theorem 7 therein). The normalized Laplacian version of GCN is advantegeous over the one made from the augmented one because we know its spectral distribution for broader range of random graphs. For example, Chung and Radcliffe [2011] proved the convergence of the spectral distribution of the normalized Laplacian for Chung-Lu's model [Chung and Lu, 2002], which includes power law graphs as a special case (see, Chung and Radcliffe [2011, Theorem 4]).

4.5.4 Over-smoothing in Link Prediction Tasks

Although our primary focus is node prediction tasks, our theory has implications for link prediction tasks (Section 2.2.2), too. Roughly speaking, if a link prediction model predicts edges' existence via a GCN's output, the model suffers from the over-smoothing in the same way as node prediction models.

Let us formalize our link prediction task. Suppose we are given a graph $G = (V, E)$ and a collection of node features $X \in \mathbb{R}^{N \times C}$. The edge set E contains edges for training only. The goal is to output $Z \in \{0, 1\}^{N \times N}$ where Z_{ij} indicates the existence or non-existence of an edge between the node pair i and j. Let $X^{(l)} \in \mathbb{R}^{N \times C}$ be a set of node representations at the l-th layer of a ReLU GCN (Section 4.5.1). We consider a link prediction model of the form $Z_{ij} = f(X_i^{(L)}, X_j^{(L)})$, where $f : \mathbb{R}^C \times \mathbb{R}^C \to \{0, 1\}$ is some (possibly learnable) function.

We define the "information-less" invariant space $\mathcal{M}$ in the same way as Section 4.5.1. Let $X \in \mathcal{M}$ and consider two node pairs (i, j) and (i', j'). By the discussion similar to Section 4.5.1, if $i \sim i'$ and $j \sim j'$, we have $(X_i, X_j) = (X_{i'}, X_{j'})$. In this sense, if a link prediction model computes the existence of an edge from representations of the node pair of the edge, it cannot extract any information for any link prediction task from the invariant space $\mathcal{M}$. Theorem 4.2 implies that a link prediction model of this type suffers from the over-smoothing problem in the same condition as the corresponding node prediction model.

4.5.5 Remark on Previous Study about Over-smoothing for Non-linear GNNs

The old version (version 2) of the preprint of Luan et al. [2019][4] formulated a theorem that explains the over-smoothing of non-linear GNNs. Specifically, Theorem 1 of the paper claims that if a graph does not have a bipartite component and the input distribution is continuous, the rank of the output of a GCN converges to the number of connected components of the underlying graph as the layer size goes to infinity almost surely. However, it is not true in general as we give a counterexample.

We restate Theorem 1 of Luan et al. [2019]. Let G be a simple undirected graph with N nodes and k connected components such that it does not have a bipartite component. Let $L = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2} \in \mathbb{R}^{N \times N}$ be the augmented normalized Laplacian of G. Let $F \in \mathbb{N}_+$ and $W_n \in \mathbb{R}^{F \times F}$ be the weight of the n-th layer for $n \in \mathbb{N}_+$. For the input $X \in \mathbb{R}^{N \times F}$, we define the output $Y_n \in \mathbb{R}^{N \times F}$ of the n-th layer of a GCN by $Y_n = \sigma(L \cdots \sigma(LXW_0) \cdots W_n)$ where σ is the ReLU function. We assume the input X is drawn from a continuous distribution on $\mathbb{R}^{N \times F}$. Then, the

[4]`https://arxiv.org/abs/1906.02174v2` (Retrieved on December, 2nd, 2020)

theorem claims that we have $\lim_{n\to\infty} \operatorname{rank}(Y_n) = k$ almost surely with respect to the distribution of X.

We construct a conterexample. Consider a graph G consisting of $N = 4$ nodes whose adjacency matrix is

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

Note that G is connected (i.e., $k = 1$) and is not bipartite. We make a GCN with $F = 3$ channels and whose weight matrices are $W_n = I_3$ (the identity matrix of size 3) for all $n \in \mathbb{N}$. For the distribution of the input X, we consider an absolutely continuous distribution with respect to the Lebesgue measure on $\mathbb{R}^{4\times 3}$ such that $P(X \geq 0) > 0$ (here, $X \geq 0$ means the element-wise comparison). For example, the standard Gaussian distribution satisfies the condition.

Since $L \geq 0$, we have $Y_n = L^n X$ if $X \geq 0$. Let $L = P^\top \Lambda P$ be the diagonalization of L where $P \in O(4)$ is an orthogonal matrix of size 4. Since $\operatorname{rank}(L) = 3$, we have $\operatorname{rank}(\Lambda^n) = 3$ for any n (we can assume that $\Lambda_{44} = 0$ without loss of generality). Therefore, under the condition $X \geq 0$, we have

$$\begin{aligned} \operatorname{rank}(Y_n) = 3 &\iff \operatorname{rank}(P^\top \Lambda^n P X) = 3 \\ &\iff X \in \{P^{-1} \begin{bmatrix} B & v \end{bmatrix}^\top \mid B \in \mathbb{R}^{3\times 3} \text{ is invertible}, v \in \mathbb{R}^3\}. \end{aligned}$$

Note that the last condition is independent of n. Since the set of invertible matrices is dense in the set of all matrices of the same size (with respect to the standard topology of the Euclidean space), we have $P(\{\operatorname{rank}(Y_n) = 3 \text{ for all } n \in \mathbb{N}\}) > 0$. Therefore, we have $\lim_{n\to\infty} \operatorname{rank}(Y_n) = 3$ with a non-zero probability. $\square$

4.6 Experiments

4.6.1 Synthesis Data: One-step Transition

We numerically investigate how the transition $f(X) := \sigma(PXW)$ changes inputs using the vector field $V(X) := f(X) - X$[5]. For this purpose, we set $N = 2$, $M = 1$, and $C = 1$. Let $\lambda_1 \leq \lambda_2$ be the eigenvalues of P. We choose W as $|\lambda_2|^{-1} \leq W < |\lambda_1|^{-1}$ so that Theorem 4.1 is applicable but is not reduced to the trivial situation (see, Appendix 4.4.2). We choose the eigenvector $e \in \mathbb{R}^2$ associated with λ_2 in two ways as described below. See Appendix 4.B.1 for the concrete values of P, e, and W. Figure 4.1 shows the visualization of V. First, we choose the non-negative eigenvector e so that it satisfies Assumption 4.1 (**Case 1**). We see that the transition function f uniformly decreases the distance from $\mathcal{M}$. This is consistent with the consequence of Theorem 4.1. Next, we choose the eigenvector $e = \begin{bmatrix} e_1 & e_2 \end{bmatrix}^\top$ such that the signs of e_1 and e_2 differ (**Case 2**), which violates Assumption 4.1. We see that $\mathcal{M}$ is not invariant under f and f does not uniformly decrease the distance from $\mathcal{M}$. Therefore, we cannot remove the non-negativity assumption from Theorem 4.1.

[5]Since we consider the one-step transition only, we omit the subscript l from f_l, X_l, and W_l.

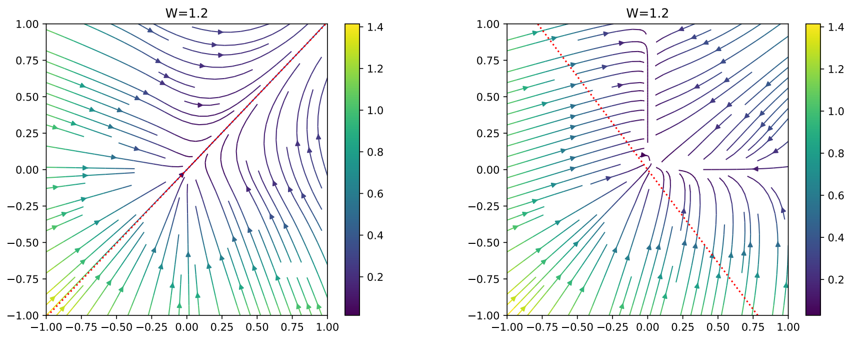


Figure 4.1: Visualization of vector field $V(X) := f(X) - X$ induced by the one-step transition. Color maps indicate the absolute value $|V(X)|$ at the point X. Dotted lines are the subspace $\mathcal{M}$. Left: **Case 1**. Right: **Case 2**. Best view in color.

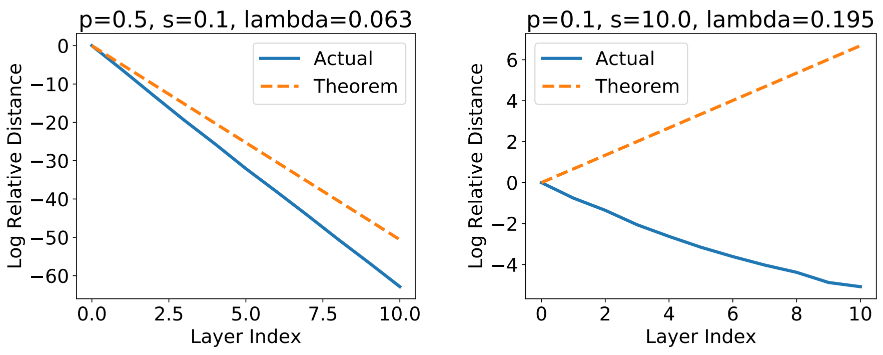


Figure 4.2: Distances to the invariant space $\mathcal{M}$ and their upper bounds. Solid lines are the log relative distance defined by $y(l) = \log(d_{\mathcal{M}}(X^{(l)})/d_{\mathcal{M}}(X^{(0)}))$ and dotted lines are upper bound $y(l) = l\log(s\lambda)$, where $X^{(0)}$ is the input signal and $X^{(l)}$ is the output of the l-th layer.

4.6.2 Synthesis Data: Distance to Invariant Space

We evaluate the distance to the invariant space $\mathcal{M}$ using synthesis data. We randomly generate an Erdős – Rényi graph, a GCN on it, and an input signal $X^{(0)}$. We compute the distance between the l-th intermediate output $X^{(l)}$ and the invariant space $\mathcal{M}$ for various edge probability p and maximum singular value s. Figure 4.2 plots the logarithm of the relative distance $y(l) = \log \frac{d_{\mathcal{M}}(X^{(l)})}{d_{\mathcal{M}}(X^{(0)})}$ with respect to the layer index l. From Theorem 4.1, we know that it is upper bounded by $y(l) = l\log(s\lambda)$. We see that this bound well approximates the actual value when $s\lambda$ is small. On the other hand, it is loose for large $s\lambda$. We leave tighter bounds for $d_{\mathcal{M}}$ in such a case for future research.

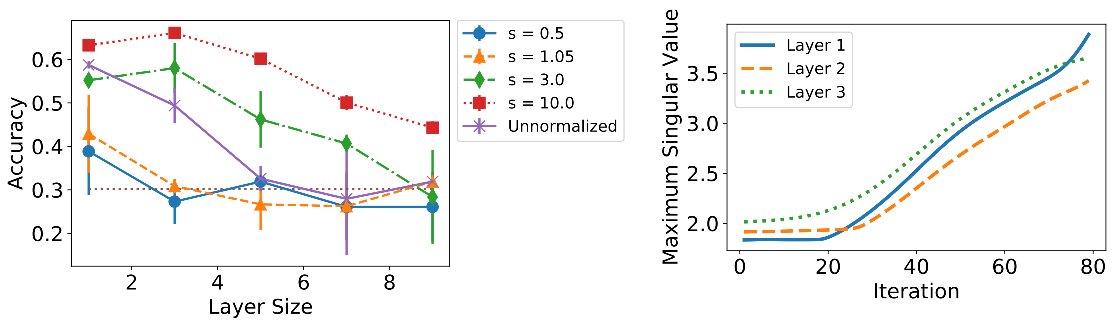


Figure 4.3: Node prediction results on Noisy Cora. Left: Effect of the maximum singular values on weights on model performance. The horizontal dotted line indicates the chance rate (30.2%). The error bar is the standard deviation of 3 trials. Right: Transition of maximum singular values during training. See Appendix 4.C.3 for results using other datasets. Best view in color.

4.6.3 Real Data: Effect of Maximum Singular Values on Performance

Theorem 4.2 implies that if s is smaller than the threshold λ^{-1}, we cannot expect deep GCN to achieve good prediction accuracy. Conversely, if we can successfully train the model, s should avoid the region $s \leq \lambda^{-1}$. We empirically confirm these hypotheses using real datasets.

We use Cora, CiteSeer, and PubMed [Sen et al., 2008], which are standard citation network datasets. The task is to classify the genre of papers using word occurrences and citation relationships. We regard each paper as a node and citation relationship as an edge. Due to space constraints, we focus on Cora in the main article. See Appendix 4.B.3 and 4.C.3 for the other datasets. The discussion in Section 4.5.2 implies that Theorem 4.2 can support a wide range of GCNs when the underlying graph is relatively dense. However, the citation networks are too sparse to examine the aforementioned hypotheses — Theorem 4.2 gives a non-trivial result only when $1 \leq s < \lambda^{-1} \approx 1 + 3.62 \times 10^{-3}$. To circumvent this, we make *noisy versions* of citation networks by randomly adding edges to graphs. Through this manipulation, we can increase the value of λ^{-1} to 1.11.

Figure 4.3 (left) shows the accuracy for the test dataset in terms of the maximum singular values and the number of graph convolution layers. We can observe that when GCNs whose maximum singular value s is out of the region $s < \lambda^{-1}$ outperform those inside the region in almost all configurations. Furthermore, the accuracy of GCNs with $s = 10$ are better than those without normalization (*unnormalized*). Figure 4.3 (right) shows the transition of the maximum singular values of the weights during training when we use a three-layered GCN. We can observe that the maximum singular value s does not shrink to the region $s \leq \lambda^{-1}$. In addition, when the layer size is small and predictive accuracy is high, GCNs gradually increase s from the initial value and avoid the region. In conclusion, the experiment results are consistent with the theorems.

4.6.4 Real Data: Effect of Signal Component Perpendicular To Invariant Space

We can decompose the output X of a model as $X = X_0 + X_1$ ($X_0 \in \mathcal{M}$, $X_1 \in \mathcal{M}^{\perp}$). According to the theory, X_0 has limited information for node classification. We hypothesize that

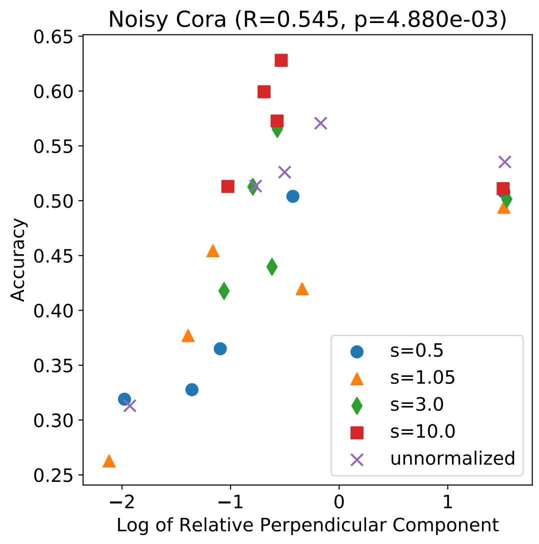


Figure 4.4: The logarithm of the relative perpendicular component $\log t(X)$ and prediction accuracy on Noisy Cora.

the model emphasizes the perpendicular component X_1 to perform good predictions. To quantitatively evaluate it, we define the relative magnitude of the perpendicular component of the output X by $t(X) := X_1/X_0$. Figure 4.4 compares this quantity and the prediction accuracy on the noisy version of Cora (see Appendix 4.C.4 for other datasets). We observe that these two quantities are correlated ($R = 0.545$). If we remove GCNs have only one layer (corresponding to right points in the figure), the correlation coefficient is 0.827. This result does not contradict to the hypothesis above [6].

4.7 Discussion

Applicability to GNNs on Sparse Graphs We have theoretically and empirically shown that when the underlying graph is sufficiently dense and large, the threshold λ^{-1} is large (Theorem 4.2 and Section 4.6.3), which means many graph CNNs are eligible. However, real-world graphs are not often dense, which means that Theorem 4.2 is applicable to very limited GCNs. In addition, Coja-Oghlan [2007] theoretically proved that if the expected average degree of $G_{N,p}$ is bounded, the smallest positive eigenvalue of the normalized Laplacian of $G_{N,p}$ is $o(1)$ with high probability. The asymptotic behaviors of GNNs on sparse graphs are left for future research.

Remedy for Over-smoothing Based on our theory, we can propose several techniques for mitigating the over-smoothing phenomena. One idea is to (randomly) sample edges in an underlying graph. The sparsity of practically available graphs could be a factor in the success of GNNs.

[6]We cannot conclude that large perpendicular components are essential for good performance, since the maximum singular value s is correlated to the accuracy, too.

Assuming this hypothesis is correct, there is a possibility that we can relive the effect of over-smoothing by sparsification. Since we can never restore the information in pruned edges if we remove them permanently, random edge sampling could work better as FastGCN [Chen et al., 2018a] and GraphSAGE [Hamilton et al., 2017] do. Another idea is to scale node representations (i.e., intermediate or final output of GNNs) appropriately so that they keep away from the invariant space $\mathcal{M}$. Our proposed weight scaling mechanism takes this strategy. Recently, Zhao and Akoglu [2020] has proposed PairNorm to alleviate the over-smoothing phenomena. Although the scaling target is different – they rescaled signals whereas we normalized weights – theirs and ours have similar spirits.

GNNs with Large Weights Our theory suggests that the maximum singular values of weights in a GCN should not be smaller than a threshold λ^{-1} because it suffers from information loss for node classification. On the other hand, if the scale of weights are very large, the model complexity of the function class represented by GNNs increases, which may cause large generalization errors. Therefore, from a statistical learning theory perspective, we conjecture that the GNNs with too-large weights perform poorly, too. A trade-off should exist between the expressive power and model complexity and there should be a “sweet spot" on the weight scale that balances the two.

Relation to Double Descent Phenomena Belkin et al. [2019] pointed out that modern deep models often have *double descent* risk curves: when a model is under-parameterized, a classical bias-variance trade-off occurs. However, once the model has a large capacity and perfectly fits the training data, the test error decreases as we increase the number of parameters. To the best of our knowledge, no literature reported the double descent phenomena for GNNs (it is consistent with the picture of the classical U-shaped risk curve in the previous paragraph). It is known that double descent phenomena do not occur in some situations, especially depending on regularization types. For example, while Belkin et al. [2019] employed the interpolating hypothesis with the minimum norm, Mei and Montanari [2019] found that the double descent was alleviated or disappeared when they used Ridge-type regularization techniques. Therefore, one can hypothesize the over-smoothing is a cause or consequence of regularization that is more like a Ridge-type rather than minimum-norm inductive bias.

Limitations in GNN Architectures Our analysis is limited to GNNs with the ReLU activation function because we implicitly use the property that ReLU is a projection onto the cone $\{X \geq 0\}$ (Appendix 4.A.2, Lemma 4.3). This fact enables the ReLU function to get along with the non-negativity of eigenvectors associated with the largest eigenvalues. Therefore, it is far from trivial to extend our results to other activation functions such as the sigmoid function or Leaky ReLU [Maas et al., 2013]. Another point is that our formulation considers the update operation [Gilmer et al., 2017] of GNNs only and does not take readout operations into account. In particular, we cannot directly apply our theory to graph classification tasks in which each sample is a graph.

Over-smoothing of Residual GNNs Considering the correspondence of GNNs and Markov processes (see Appendix 4.4.4), one can imagine that residual links do not contribute to alleviating the over-smoothing phenomena because adding residual connections to a GNN corresponds to converting a Markov process to its lazy version. When a Markov process converges to a stable distribution, the corresponding lazy process also converges eventually under certain conditions. It implies that residual links might not be helpful. However, Li et al. [2019] reported that GNNs with as many as 56 layers performed well if they added residual connections. Considering that, the situation could be more complicated than our intuitions. The analysis of the role of residual connections in GNNs is a promising direction for future research.

4.8 Chapter Conclusion

In this chapter, to understand the over-smoothing problem, an empirically observed phenomena that deep non-linear GNNs do not perform well, we analyzed their asymptotic behaviors by interpreting them as a dynamical system that includes GCN and Markov process as special cases. We gave theoretical conditions under which GCNs suffer from the information loss in the limit of infinite layers. Our theory directly related the expressive power of GNNs and topological information of the underlying graphs via spectra of the Laplacian. It enabled us to leverage spectral and random graph theory to analyze the expressive power of GNNs. To demonstrate this, we considered GCN on the Erdős – Rényi graph as an example and showed that when the underlying graph is sufficiently dense and large, a wide range of GCNs on the graph suffer from information loss. Based on the theory, we gave a principled guideline for how to determine the scale of weights of GNNs and empirically showed that the weight normalization implied by our theory performed well in real datasets. One promising direction of research is to analyze the optimization and statistical properties such as the generalization power [Verma and Zhang, 2019] of GNNs via spectral and random graph theories.

4.A Proofs

4.A.1 Proof of Proposition 4.1

Proof. For any $u \in U^\perp$ and $v \in U$, by symmetry of P, we have

$$\langle Pu, v\rangle = (Pu)^\top v = u^\top P^\top v = u^\top P v = \langle u, Pv\rangle.$$

Since U is an invariant space of P, we have $Pv \in U$. Hence, we have $\langle u, Pv\rangle = 0$ because $u \in U^\perp$. We obtain $Pu \in U^\perp$ by the definition of $U^\perp$. □

4.A.2 Proof of Theorem 4.1

As we wrote in the main article, it is enough to show the following lemmas (definition of miscellaneous variables are as in Section 4.3). Remember that $\lambda = \sup_{n\in[N-M]} |\lambda_n|$ and s_{lh} is the maximum singular value of W_{lh}

Lemma 4.1. *For any $X \in \mathbb{R}^{N\times C}$, we have $d_{\mathcal{M}}(PX) \leq \lambda d_{\mathcal{M}}(X)$.*

Lemma 4.2. *For any $X \in \mathbb{R}^{N\times C}$, we have $d_{\mathcal{M}}(XW_{lh}) \leq s_{lh} d_{\mathcal{M}}(X)$.*

Lemma 4.3. *For any $X \in \mathbb{R}^{N\times C}$, we have $d_{\mathcal{M}}(\sigma(X)) \leq d_{\mathcal{M}}(X)$.*

Proof of Lemma 4.1. Since P is a symmetric linear operator on $U^{\perp}$, we can choose the orthonormal basis $(e_m)_{m=M+1,\ldots,N}$ of $U^{\perp}$ consisting of the eigenvalue of $P|_{U^{\perp}}$. Let λ_m be the eigenvalue of P to which e_m is associated ($m = M+1, \ldots, N$). Note that since the operator norm of $P|_{U^{\perp}}$ is λ, we have $|\lambda_m| \leq \lambda$ for all $m = M+1, \ldots, N$. Since $(e_m)_{m\in[N]}$ forms the orthonormal basis of $\mathbb{R}^N$, we can uniquely write $X \in \mathbb{R}^{N\times C}$ as $X = \sum_{m=1}^{N} e_m \otimes w_m$ for some $w_m \in \mathbb{R}^C$. Then, we have $d_{\mathcal{M}}^2(X) = \sum_{m=M+1}^{N} \|w_m\|^2$ where $\|\cdot\|$ is the 2-norm of a vector. On the other hand, we have

$$\begin{aligned}
PX &= \sum_{m=1}^{N} Pe_m \otimes w_m \\
&= \sum_{m=1}^{M} Pe_m \otimes w_m + \sum_{m=M+1}^{N} Pe_m \otimes w_m \\
&= \sum_{m=1}^{M} Pe_m \otimes w_m + \sum_{m=M+1}^{N} e_m \otimes (\lambda_m w_m)
\end{aligned}$$

Since U is invariant under P, for any $m \in [M]$, we can write Pe_m as a linear combination of $e_n (n \in [M])$. Therefore, we have $d_{\mathcal{M}}^2(PX) = \sum_{m=M+1}^{N} \|\lambda_m w_m\|^2$. Then, we obtain the desired inequality as follows:

$$\begin{aligned}
d_{\mathcal{M}}^2(PX) &= \sum_{m=M+1}^{N} \|\lambda_m w_m\|^2 \\
&\leq \lambda^2 \sum_{m=M+1}^{N} \|w_m\|^2 \\
&\leq \lambda^2 \sum_{m=M+1}^{N} \|w_m\|^2 \\
&= \lambda^2 d_{\mathcal{M}}^2(X).
\end{aligned}$$

□

Proof of Lemma 4.2. Using the same decomposition of X as the proof in Lemma 4.1, we have

$$\begin{aligned}
XW_{lh} &= \sum_{m=1}^{N} e_m \otimes (W_{lh}^{\top} w_m) \\
&= \sum_{m=1}^{M} e_m \otimes (W_{lh}^{\top} w_m) + \sum_{m=M+1}^{N} e_m \otimes (W_{lh}^{\top} w_m).
\end{aligned}$$

Therefore, we have

$$\begin{aligned} d_{\mathcal{M}}^2(XW_{lh}) &= \sum_{m=M+1}^{N} \|W_{lh}^\top w_m\|^2 \\ &\leq s_{lh}^2 \sum_{m=M+1}^{N} \|w_m\|^2 \\ &= s_{lh}^2 d_{\mathcal{M}}^2(X). \end{aligned}$$

$\square$

Proof of Lemma 4.3. We choose $(e_m)_{m=N-M+1,\ldots,N}$ as in the proof of Lemma 4.1. We denote $X = (X_{nc})_{n\in[N],c\in[C]}$ and $e_n = (e_{mn})_{m\in[N]}$, respectively. Let $(e'_c)_{c\in[C]}$ be the standard basis of $\mathbb{R}^C$. Then, $(e_n \otimes e'_c)_{n\in[N],c\in[C]}$ is the orthonormal basis of $\mathbb{R}^{N\times C}$, endowed with the standard inner product as a Euclid space. Therefore, we can decompose X as $X = \sum_{n=1}^N \sum_{c=1}^C a_{nc} e_n \otimes e'_c$ where $a_{nc} = \langle X, e_n \otimes e'_c\rangle = \sum_{m=1}^N X_{mc} e_{mn}$. Then, we have $d_{\mathcal{M}}^2(X) = \sum_{n=M+1}^N \| \sum_{c=1}^C a_{nc} e'_c\|^2$, which is further transformed as

$$\begin{aligned} d_{\mathcal{M}}^2(X) &= \sum_{n=M+1}^{N} \left\| \sum_{c=1}^{C} a_{nc} e'_c \right\|^2 \\ &= \sum_{n=M+1}^{N} \sum_{c=1}^{C} a_{nc}^2 \\ &= \sum_{c=1}^{C} \left(\sum_{n=1}^{N} a_{nc}^2 - \sum_{n=1}^{M} a_{nc}^2 \right) \\ &= \sum_{c=1}^{C} \left(\|X_{\cdot c}\|^2 - \sum_{n=1}^{M} \langle X_{\cdot c}, e_n\rangle^2 \right), \end{aligned}$$

where $X_{\cdot c}$ is the c-th column vector of X. Similarly, we have

$$d_{\mathcal{M}}^2(\sigma(X)) = \sum_{c=1}^{C} \left(\|X_{\cdot c}^+\|^2 - \sum_{n=1}^{M} \langle X_{\cdot c}^+, e_n\rangle^2 \right),$$

where we denote $\sigma(X) = (X_{nc}^+)_{n\in[N],c\in[C]}$ in shorthand. Therefore, the inequality follow from the following lemma. $\square$

Lemma 4.4. *Let $x \in \mathbb{R}^N$ and $v_1, \ldots, v_M \in \mathbb{R}^N$ be orthonormal vectors (i.e., $\langle v_m, v_n\rangle = \delta_{mn}$) satisfying $v_m \geq 0$ for all $m \in [M]$. Then, we have $\|x\|^2 - \sum_{m=1}^M \langle x, v_m\rangle^2 \geq \|x^+\|^2 - \sum_{m=1}^M \langle x^+, v_m\rangle^2$ where $x^+ := \max(x, 0)$ for $x \in \mathbb{R}$.*

Proof. The value $\|y\|^2 - \sum_{m=1}^M \langle y, u_m\rangle^2$ is invariant under simultaneous coordinate permutation of y and u_m's. Therefore, we can assume without loss of generality that the coordinate of x are sorted: $x_1 \leq \ldots \leq x_L < 0 \leq x_{L+1} \leq \cdots \leq x_N$ for some $L \leq N$. Then, we have

$$\|x\|^2 - \|x^+\|^2 = \sum_{n=1}^{L} x_n^2. \tag{4.1}$$

When $L = 0$, the sum in the right hand side is treated as 0. On the other hand, writing as $v_m = (v_{nm})_{n \in [N]}$, direct calculation shows

$$\sum_{m=1}^{M} \langle x, v_m \rangle^2 - \langle x^+, v_m \rangle^2 = \sum_{m=1}^{M} \left(\left(\sum_{n=1}^{L} x_n v_{nm} \right)^2 - 2 \sum_{n=1}^{L} \sum_{l=L+1}^{N} x_n x_l v_{nm} v_{lm} \right). \tag{4.2}$$

Let $I_m := \{n \in [N] \mid v_{nm} > 0\}$ be the support of v_m for $m \in [M]$. We note that if $m \neq m' \in [M]$, we have $I_m \cap I_{m'} = \emptyset$ since if there existed $n \in I_m \cap I_{m'}$, we have

$$0 = \langle v_m, v_{m'} \rangle \geq v_{nm} v_{nm'} > 0,$$

which is contradictory. Therefore,

$$\begin{aligned}
\sum_{m=1}^{N} \left(\sum_{n=1}^{L} x_n v_{nm} \right)^2 &= \sum_{m=1}^{N} \left(\sum_{n \in I_m \cap [L]} x_n v_{nm} \right)^2 \\
&\leq \sum_{m=1}^{N} \left(\sum_{n \in I_m \cap [L]} x_n^2 \right) \left(\sum_{n \in I_m \cap [L]} v_{nm}^2 \right) \quad (\because \text{Cauchy–Schwarz inequality}) \\
&\leq \sum_{m=1}^{N} \left(\sum_{n \in I_m \cap [L]} x_n^2 \right) \quad (\because \|v_m\|^2 = 1) \\
&\leq \sum_{n=1}^{L} x_n^2.
\end{aligned} \tag{4.3}$$

We used the fact that I_m's are disjoint and $v_{nm} = 0$ if $n \notin \cup_m I_m$ in the first equality above. Further, we have $x_n x_l v_{nm} v_{lm} \leq 0$ for $1 \leq n \leq L$ and $L + 1 \leq l \leq N$ by the definition of L and non-negativity of v_m. By combining (4.1), (4.2), and (4.3), we have

$$\sum_{m=1}^{M} \langle x, v_m \rangle^2 - \langle x^+, v_m \rangle^2 \leq \sum_{n=1}^{L} x_n^2 = \|x\|^2 - \|x^+\|^2.$$

$\square$

Proof of Theorem 4.1. By Lemma 4.1, 4.2, and 4.3, we have

$$d_{\mathcal{M}}(f_l(X)) = d_{\mathcal{M}}(\underbrace{\sigma(\cdots \sigma(\sigma(}_{H \text{ times}} PX) W_{l1}) W_{l2} \cdots W_{lH_l}))$$

$$
\begin{aligned}
&\leq d_{\mathcal{M}}(\underbrace{\sigma(\cdots\sigma(\sigma(}_{H-1\text{ times}}PX)W_{l1})W_{l2}\cdots)W_{lH_l}) \\
&\leq s_{lH_l-1} d_{\mathcal{M}}(\underbrace{\sigma(\cdots\sigma(\sigma(}_{H-1\text{ times}}PX)W_{l1})W_{l2}\cdots)W_{lH_l-1})) \\
&\cdots \\
&\leq \left(\prod_{h=1}^{H_l} s_{lh}\right) d_{\mathcal{M}}(PX) \\
&\leq s_l d_{\mathcal{M}}(PX) \\
&\leq s_l \lambda d_{\mathcal{M}}(X).
\end{aligned}
$$

□

4.A.3 Proof of Proposition 4.2

Proof. Since λ is the operator norm of $P|_{U^\perp}$, the assumption implies that the operator norm of P itself is no less than λ. Therefore, we have $\|PXW_l\|_{\mathrm{F}} \leq \lambda\|XW_l\|_{\mathrm{F}} \leq s_l\lambda\|X\|_{\mathrm{F}}$. On the other hand, since $\sigma(x)^2 \leq x^2$ for any $x \in \mathbb{R}$, we have $\|\sigma(X)\|_{\mathrm{F}} \leq \|X\|_F$ for any $X \in \mathbb{R}^{N\times C}$. Combining the two, we have $\|f_l(X)\|_{\mathrm{F}} \leq \|PXW_l\|_{\mathrm{F}} \leq s_l\lambda\|X\|_{\mathrm{F}}$. □

4.A.4 Proofs of Proposition 4.3 and Proposition 4.4

Proof of Proposition 4.3. By definition, we have $d_{\mathcal{M}}(X) = |x_2|$. On the other hand, direct calculation shows that $f_l(X) = \begin{bmatrix}(W\mu X_1)^+ & (W\lambda X_2)^+\end{bmatrix}^\top$ and $d_{\mathcal{M}}(f_l(X)) = (W\lambda X_2)^+$ where $x^+ := \max(x, 0)$ for $x \in \mathbb{R}$. Since $W > \lambda^{-1}$ and $x_2 > 0$, we have $d_{\mathcal{M}}(f_l(X)) > d_{\mathcal{M}}(X)$. □

Proof of Proposition 4.4. First, note that $e' := \frac{1}{\sqrt{2}}\begin{bmatrix}1 & -1\end{bmatrix}^\top$ is the eigenvector of P associated to λ: $Pe' = \lambda e'$. For $X = ae + be'$ ($a, b > 0$), the distance between X and $\mathcal{M}$ is $d_{\mathcal{M}}(X) = |b|$. On the other hand, by direct computation, we have

$$
f(X) = \sigma(PXW) = \begin{cases} \begin{bmatrix}0 & \frac{\lambda b}{\sqrt{2}}\end{bmatrix}^\top & \text{(if } b \geq 0\text{)}, \\ \begin{bmatrix}\frac{-\lambda b}{\sqrt{2}} & 0\end{bmatrix}^\top & \text{(if } b < 0\text{)}. \end{cases}
$$

Therefore, the distance between $f(X)$ and $\mathcal{M}$ is $d_{\mathcal{M}}(f(X)) = \lambda|b|/2$. Since $\lambda < 2$, we have $d_{\mathcal{M}}(f(X)) < d_{\mathcal{M}}(X)$ for any $X \in \mathbb{R}^2$. □

4.A.5 Proof of Theorem 4.3

We follow the proof of Theorem 2 of Chung and Radcliffe [2011]. The idea is to relate the spectral distribution of the normalized Laplacian with that of its expected version. Since we can compute the latter one explicitly for the Erdős-Rényi graph, we can derive the convergence of spectra. We employ this technique and derive similar conclusion for the augmented normalized Laplacian.

First, we consider genral random graphs not restricted to Erdős-Rényi graphs. Let $N \in \mathbb{N}_+$, and $P = (p_{ij})_{i,j\in[N]}$ be a non-negative symmetric matrix (meaning that $p_{ij} \geq 0$ for any $i, j \in [N]$). Let G be an undirected random graph with N nodes such that an edge between i and j is independently present with probability p_{ij}. Let A and D be the adjacency and the degree matrices of G, respectively (that is, $A_{ij} \sim \mathrm{Ber}(p_{ij})$, i.i.d.). Define the expected node degree of node i by $t_i := \sum_{j=1}^N p_{ij}$. Let $\tilde{A} := A + I_N$, $\tilde{D} := D + I_N$ and define $\bar{A} := \mathbb{E}[\tilde{A}] = P + I_N$ and $\bar{D} := \mathbb{E}[\tilde{D}] = \mathrm{diag}(t_1, \ldots, t_N) + I_N$ correspondingly. We define the augmented normalized Laplacian $\tilde{\Delta}$ of G by $\tilde{\Delta} := I_N - \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ and its expected version by $\bar{\Delta} := I_N - \bar{D}^{-\frac{1}{2}}\bar{A}\bar{D}^{-\frac{1}{2}}$ [7]. For a symmetric matrix $X \in \mathbb{R}^N$, we define its eigenvalues, sorted in ascending order by $\lambda_1(X) \leq \cdots \leq \lambda_N(X)$ and its operator norm by $\|X\| = \max_{n\in[N]} |\lambda_n(X)|$.

Lemma 4.5 (Ref. Chung and Radcliffe [2011] Theorem 2)**.** *Let $\delta := \min_{n\in[N]} t_n$ be the minimum expected degree of G. Set $k(\varepsilon) := 3(1 + \log(4/\varepsilon))$. Then, for any $\varepsilon > 0$, if $\delta + 1 > k(\varepsilon)\log N$, we have*

$$\max_{n\in[N]} \left|\lambda_n(\tilde{\Delta}) - \lambda_n(\bar{\Delta})\right| \leq 4\sqrt{\frac{3\log(4N/\varepsilon)}{\delta+1}}$$

with probability at least $1-\varepsilon$.

Proof. By Weyl's theorem, we have $\max_{n\in[N]} \left|\lambda_n(\tilde{\Delta}) - \lambda_n(\bar{\Delta})\right| \leq \|\tilde{\Delta} - \bar{\Delta}\|$. Therefore, it is enough to bound $\|\tilde{\Delta} - \bar{\Delta}\|$. Let $C := I_N - \bar{D}^{-\frac{1}{2}}\tilde{A}\bar{D}$. By the triangular inequality, we have $\|\tilde{\Delta} - \bar{\Delta}\| \leq \|\tilde{\Delta} - C\| + \|C - \bar{\Delta}\|$. We will bound these terms respectively.

First, we bound $\|C - \bar{\Delta}\|$. Direct calculation shows $C - \bar{\Delta} = -\bar{D}^{-\frac{1}{2}}(A - P)\bar{D}^{-\frac{1}{2}}$. Let $E^{ij} \in \mathbb{R}^{N\times N}$ be a matrix defined by

$$(E^{ij})_{kl} = \begin{cases} 1 & \text{if } (i = k \text{ and } i = l) \text{ or } (i = l \text{ and } j = k), \\ 0 & \text{otherwise.} \end{cases}$$

We define the random variable Y_{ij} by

$$Y_{ij} := \frac{A_{ij} - p_{ij}}{\sqrt{t_i + 1}\sqrt{t_j + 1}} E^{ij}.$$

Then, Y_{ij}'s are independent and we have $C - \bar{\Delta} = \sum_{i,j=1}^N Y_{ij}$. To apply Theorem 5 of Chung and Radcliffe [2011] to Y_{ij}'s, we bound $\|Y_{ij} - \mathbb{E}[Y_{ij}]\|$ and $\|\sum_{i,j=1}^N \mathbb{E}[Y_{ij}^2]\|$. First, we have

$$\|Y_{ij} - \mathbb{E}[Y_{ij}]\| = \|Y_{ij}\| \leq \frac{\|E^{ij}\|}{\sqrt{t_i + 1}\sqrt{t_j + 1}} \leq (\delta + 1)^{-1}.$$

[7] Note that $\mathbb{E}[\tilde{\Delta}] \neq \bar{\Delta}$ in general due to the dependence between $\tilde{A}$ and $\tilde{D}$.

Since

$$\mathbb{E}[Y_{ij}^2] = \frac{p_{ij} - p_{ij}^2}{(t_i+1)(t_j+1)} \begin{cases} E^{ii} + E^{jj} & (\text{if } i \neq j), \\ E^{ii} & (\text{if } i = j), \end{cases}$$

we have

$$\begin{aligned} \left\| \sum_{i,j=1}^N \mathbb{E}[Y_{ij}^2] \right\| &= \left\| \sum_{i,j=1}^N \frac{p_{ij} - p_{ij}^2}{(t_i+1)(t_j+1)} E^{ii} \right\| \\ &= \max_{i \in [N]} \left(\sum_{j=1}^N \frac{p_{ij} - p_{ij}^2}{(t_i+1)(t_j+1)} \right) \\ &\leq \max_{i \in [N]} \left(\sum_{j=1}^N \frac{p_{ij}}{(t_i+1)(t_j+1)} \right) \\ &\leq (\delta+1)^{-1}. \end{aligned}$$

By letting $a \leftarrow \sqrt{\frac{3\log(4N/\varepsilon)}{\delta+1}}$, $M \leftarrow (\delta+1)^{-1}$, $v^2 \leftarrow (\delta+1)^{-1}$ and applying Theorem 5 of Chung and Radcliffe [2011], we have

$$\begin{aligned} \Pr(\|C - \bar{\Delta}\| > a) &\leq 2N \exp\left(-\frac{a^2}{2(\delta+1)^{-1} + 2(\delta+1)^{-1}a/3} \right) \\ &\leq 2N \exp\left(-\frac{3\log(4N/\varepsilon)}{2(1+a/3)} \right). \end{aligned}$$

By the definition of $k(\varepsilon)$, we have $a < 1$ if $\delta + 1 > k(\varepsilon)\log n$. For such δ, we have

$$\begin{aligned} \Pr(\|C - \bar{\Delta}\| > a) &\leq 2N \exp\left(-\frac{3\log(4N/\varepsilon)}{2(1+a/3)} \right) \\ &\leq 2N \exp\left(-\log(4N/\varepsilon)\right) \quad (\because a < 1) \\ &= \frac{\varepsilon}{2}. \end{aligned} \tag{4.4}$$

Next, we bound $\|\tilde{\Delta} - C\|$. First, since $a < 1$, by Chernoff bound (see, e.g. Angluin and Valiant [1979], Hagerup and Rüb [1990])), we have

$$\begin{aligned} \Pr(|d_i - t_i| > a(t_i+1)) &\leq 2\exp\left(-\frac{a^2(t_i+1)}{3} \right) \\ &\leq 2\exp\left(-\frac{a^2(\delta+1)}{3} \right) \\ &= \frac{\varepsilon}{2N}. \end{aligned}$$

Therefore, if $|d_i - t_i| \leq a(t_i + 1)$, then we have

$$\begin{aligned}\left|\sqrt{\frac{d_i+1}{t_i+1}} - 1\right| &\leq \left|\frac{d_i+1}{t_i+1} - 1\right| \quad (\because |\sqrt{x} - 1| \leq |x-1| \text{ for } x \geq 0)\\ &= \left|\frac{d_i - t_i}{t_i+1}\right| \\ &\leq a.\end{aligned}$$

Therefore, by union bound, we have

$$\|\bar{D}^{-\frac{1}{2}}\tilde{D}^{\frac{1}{2}} - I_N\| = \max_{i\in[N]}\left|\sqrt{\frac{d_i+1}{t_i+1}} - 1\right| \leq a$$

with probability at least $1 - \varepsilon/2$. Further, since the eigenvalue of the augmented normalized Laplacian is in $[0, 2]$ by the proof of Proposition 4.5, we have $\|I_N - \tilde{\Delta}\| \leq 1$. By combining them, we have

$$\begin{aligned}\|\tilde{\Delta} - C\| &= \|(\bar{D}^{-\frac{1}{2}}\tilde{D}^{\frac{1}{2}} - I_N)(I_N - \tilde{\Delta})\tilde{D}^{\frac{1}{2}}\bar{D}^{-\frac{1}{2}} + (I_N - \tilde{\Delta})(I - \tilde{D}^{\frac{1}{2}}\bar{D}^{-\frac{1}{2}})\| \\ &\leq \|(\bar{D}^{-\frac{1}{2}}\tilde{D}^{\frac{1}{2}} - I_N\|\|\tilde{D}^{\frac{1}{2}}\bar{D}^{-\frac{1}{2}}\| + \|I - \tilde{D}^{\frac{1}{2}}\bar{D}^{-\frac{1}{2}}\| \\ &\leq a(a+1) + a. \end{aligned} \tag{4.5}$$

From (4.4) and (4.5), we have

$$\begin{aligned}\|\tilde{\Delta} - \bar{\Delta}\| &\leq \|\tilde{\Delta} - C\| + \|C - \bar{\Delta}\| \\ &\leq a + a(a+1) + a \\ &\leq a^2 + 3a \\ &\leq 4a \quad (\because a < 1)\end{aligned}$$

with probability at least $1 - \varepsilon$ by union bound. □

Let $N \in \mathbb{N}_+$ and $p > 0$. In the case of the Erdős-Rényi graph $G_{N,p}$, we should set $P = p(J_N - I_N)$ where $J_N \in \mathbb{R}^{N\times N}$ are the all-one matrix. Then, we have $\bar{A} = pJ_N + (1-p)I_N$, $\bar{D} = (Np - p + 1)I_N$, and $\bar{\Delta} = \frac{p}{Np-p+1}(NI_N - J_N)$. Since the eigenvalue of J_N is N (with multiplicity 1) and 0 (with multiplicity $N-1$), the eigenvalue of $\bar{\Delta}$ is 0 (with multiplicity 1) and $\frac{Np}{Np-p+1}$ (with multiplicity $N-1$). For $G_{N,p}$, δ is the expected average degree $(N-1)p$. Hence, we have the following lemma from Lemma 4.5:

Lemma 4.6. *Let $\tilde{\Delta}$ be its augmented normalized Laplacian of the Erdős-Rényi graph $G_{N,p}$. For any $\varepsilon > 0$, if $\frac{Np-p+1}{\log N} > k(\varepsilon) := 3(1 + \log(4/\varepsilon))$, then, with probability at least $1 - \varepsilon$, we have*

$$\max_{i=2,\ldots,N}\left|\lambda_i(\tilde{\Delta}) - \frac{Np}{Np-p+1}\right| \leq 4\sqrt{\frac{3\log(4N/\varepsilon)}{Np-p+1}}.$$

Corollary 4.5. *Consider GCN on $G_{N,p}$. Let W_l be the weight of the l-th layer of GCN and s_l be the maximum singular value of W_l for $l \in \mathbb{N}_+$. Set $s := \sup_{l\in\mathbb{N}_+}$. Let $\varepsilon > 0$. We define $k(\varepsilon) := 3(1 + \log(4/\varepsilon))$ and $l(N,p,\varepsilon) = \frac{1-p}{Np-p+1} + 4\sqrt{\frac{3\log(4N/\varepsilon)}{Np-p+1}}$. If $\frac{Np-p+1}{\log N} > k(\varepsilon)$ and $s \leq l(N,\varepsilon)^{-1}$, then, GCN on $G_{N,p}$ satisfies the assumption of Theorem 4.2 with probability at least $1-\varepsilon$.*

Proof of Theorem 4.3. Since $\frac{\log N}{Np} = o(1)$, for fixed ε, we have

$$\frac{Np-p+1}{\log N} > \frac{Np}{\log N} > k(\varepsilon)$$

for sufficiently large N. Further, $Np \to \infty$ as $N \to \infty$ when $\frac{\log N}{Np} = o(1)$. Therefore, we have

$$\frac{(1-p)^2}{Np-p+1} \leq \frac{1}{Np} \leq (7-4\sqrt{3})^2 \log\left(\frac{4N}{\varepsilon}\right)$$

for sufficiently large N. Hence.

$$\frac{1-p}{Np-p+1} \leq (7-4\sqrt{3})\sqrt{\frac{\log(4N/\varepsilon)}{Np-p+1}}.$$

Therefore, we have $l(N,p,\varepsilon) \leq 7\sqrt{\frac{\log(4N/\varepsilon)}{Np-p+1}}$. Therefore, if $s \leq \frac{1}{7}\sqrt{\frac{Np-p+1}{\log(4N/\varepsilon)}}$, then we have $s \leq l(N,p,\varepsilon)^{-1}$. □

4.A.6 Proof of Proposition 4.5

Proof. Let $\tilde{\mu}_1 \leq \cdots \leq \tilde{\mu}_N$ be the eigenvalue of the augmented normalized Laplacian $\tilde{\Delta}$, sorted in ascending order. Since $P = I_N - \tilde{\Delta}$, it is enough to show $\tilde{\mu}_1 = \cdots = \tilde{\mu}_M = 0$, $\tilde{\mu}_{M+1} > 0$, and $\tilde{\mu}_N < 2$. For the first two, the statements are equivalent to that $\tilde{\Delta}$ is positive semi-definite and that the multiplicity of the eigenvalue 0 is same as the number of connected components [8]. This is well-known for Laplacian or its normalized version (see, e.g., Chung and Graham [1997]) and the proof for $\tilde{\Delta}$ is similar. By direct calculation, we have

$$x^\top \tilde{\Delta} x = \frac{1}{2}\sum_{i,j=1}^{N} a_{ij}\left(\frac{x_i}{\sqrt{d_i+1}} - \frac{x_j}{\sqrt{d_j+1}}\right)^2$$

for any $x = \begin{bmatrix} x_1 & \cdots & x_N \end{bmatrix}^\top \in \mathbb{R}^N$. Therefore, $\tilde{\Delta}$ is positive semi-definite and hence $\tilde{\mu}_1 \geq 0$.

Suppose temporally that G is connected. If $x \in \mathbb{R}^N$ is an eigenvector associated to 0, then, by the aformentioned calculation, $\frac{x_i}{\sqrt{d_i+1}}$ and $\frac{x_j}{\sqrt{d_j+1}}$ must be same for all pairs (i,j) such that $a_{ij} > 0$. However, since G is connected, $\frac{x_i}{\sqrt{d_i+1}}$ must be same value for all $i \in [N]$. That means the multiplicity of the eigenvalue 0 is 1 and any eigenvector associated to 0 must be proportional

[8]The former statement is identical to Lemma 1 and latter one is the extension of Lemma 2 of Wu et al. [2019a].

to $\tilde{D}^{\frac{1}{2}}\mathbf{1}$. Now, suppose G has M connected components $V_1, \ldots, V_M$. Let $\tilde{\Delta}_m$ be the augmented normalized Laplacians corresponding to each connected component V_m for $m \in [M]$. By the aformentioned discussion, $\tilde{\Delta}_m$ has the eigenvalue 0 with multiplicity 1. Since $\tilde{\Delta}$ is the direct sum of $\tilde{\Delta}'_m s$, the eigenvalue of $\tilde{\Delta}$ is the union of those for $\tilde{\Delta}_m$'s. Therefore, $\tilde{\Delta}$ has the eigenvalue 0 with multiplicity M and $e_m = \tilde{D}^{\frac{1}{2}}\mathbf{1}_m$'s are the orthogonal basis of the eigenspace.

Finally, we prove $\tilde{\mu}_N < 2$. Let μ_N be the largest eigenvalue of the normalized Laplacian $\Delta = D^{-\frac{1}{2}}(D-A)D^{-\frac{1}{2}}$, where $D^{-\frac{1}{2}} \in \mathbb{R}^{N\times N}$ is the diagonal matrix defined by

$$D_{ii}^{-\frac{1}{2}} = \begin{cases} \deg(i)^{-\frac{1}{2}} & (\text{if } \deg(i) \neq 0) \\ 0 & (\text{if } \deg(i) = 0) \end{cases}.$$

Note that $D^{-\frac{1}{2}}D^{\frac{1}{2}}$ nor $D^{\frac{1}{2}}D^{-\frac{1}{2}}$ are not equal to the identity matrix I_N in general. However, we have

$$L = D^{\frac{1}{2}}D^{-\frac{1}{2}}LD^{-\frac{1}{2}}D^{\frac{1}{2}} \tag{4.6}$$

where $L = D - A$ is the (unnormalized) Laplacian. Therefore, we have

$$\begin{aligned}
\tilde{\mu}_N &= \max_{x\neq 0} \frac{x^\top \tilde{\Delta} x}{\|x\|} \\
&= \max_{x\neq 0} \frac{x^\top \tilde{D}^{-\frac{1}{2}} L \tilde{D}^{-\frac{1}{2}} x}{\|x\|} \\
&= \max_{x\neq 0} \frac{x^\top \tilde{D}^{-\frac{1}{2}} D^{\frac{1}{2}} D^{-\frac{1}{2}} L D^{-\frac{1}{2}} D^{\frac{1}{2}} \tilde{D}^{-\frac{1}{2}} x}{\|x\|} \qquad (\because (4.6)) \\
&= \max_{x\neq 0} \frac{(D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x)^\top \Delta (D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x)}{\|x\|} \\
&= \max_{\substack{x\neq 0 \\ D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x\neq 0}} \frac{(D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x)^\top \Delta (D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x)}{\|x\|} \\
&= \max_{\substack{x\neq 0 \\ D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x\neq 0}} \frac{(D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x)^\top \Delta (D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x)}{\|D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x\|} \frac{\|D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x\|}{\|x\|} \\
&\leq \max_{\substack{x\neq 0 \\ D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x\neq 0}} \frac{(D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x)^\top \Delta (D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x)}{\|D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x\|} \max_{\substack{x\neq 0 \\ D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x\neq 0}} \frac{\|D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x\|}{\|x\|} \\
&\leq \max_{y\neq 0} \frac{y^\top \Delta y}{\|y\|} \max_{x\neq 0} \frac{\|D^{\frac{1}{2}}\tilde{D}^{-\frac{1}{2}}x\|}{\|x\|} \\
&= \mu_N \max_{n\in[N]} \left(\frac{\deg(i)}{\deg(i)+1}\right)^{\frac{1}{2}} \\
&\leq \mu_N.
\end{aligned}$$

Therefore, we have $\tilde{\mu}_N \leq \mu_N$[9]. Since $\max_{i\in[N]}\left(\frac{\deg(i)}{\deg(i)+1}\right)^{\frac{1}{2}} < 1$, the equality $\tilde{\mu}_N = \mu_N$ holds if and only if $\mu_N = 0$, that is, G has N connected components. On the other hand, it is known that $\mu_N \leq 2$ and the equality holds if and only if G has non-trivial bipartite graph as a connected component (see, e.g., Chung and Graham [1997]). Therefore, $\tilde{\mu}_N = \mu_N$ and $\mu_N = 2$ does not hold simultaneously and we obtain $\mu_N < 2$. □

4.B Experiment Settings

4.B.1 Experiments in Section 4.6.1

We set the eigenvalue of P to $\lambda_1 = 0.5$ and $\lambda_2 = 1.0$ and randomly generated P until the eigenvector e associated to λ_2 satisfies the condition of each case described in the main article. We set $W = 1.2$ and used the following values for each case as P and e.

Case 1

$$P = \begin{bmatrix} 0.7469915 & 0.2499819 \\ 0.2499819 & 0.7530085 \end{bmatrix}, \quad e = \begin{bmatrix} 0.7028392 \\ -0.71134876 \end{bmatrix}.$$

Case 2

$$P = \begin{bmatrix} 0.6899574 & -0.2426827 \\ -0.2426827 & 0.8100426 \end{bmatrix}, \quad e = \begin{bmatrix} 0.61637234 \\ -0.78745485 \end{bmatrix}.$$

4.B.2 Experiments in Section 4.6.2

We randomly generated an Erdős – Rényi graph $G_{N,p}$ with $N = 1000$ and randomly generated a one-of-K hot vector for each node and embed it to a C-dimensional vector using a random matrix whose elements were randomly sampled from the standard Gaussian distribution. Here, $K = 10$ and $C = 32$. We treated the resulting single as the input signal $X^{(0)} \in \mathbb{R}^{N\times C}$. We constructed a GCN with $L = 10$ layers and C channels. All parameters were i.i.d. sampled from the Gaussian distribution whose standard deviation is same as the one used in LeCun et al. [2012][10] and multiplied a scalar to each weight matrix so that the largest singular value equals to a specified value s. We used three configurations $(p, s) = (0.1, 0.1), (0.5, 1.0), (0.5, 10.0)$. λ of the generated GCNs are 0.063, 0.197, 0.194, respectively. See Appendix 4.6.2 for the results of other configurations of (p, s).

[9]Theorem 1 of Wu et al. [2019a] showed that this inequality strictly holds when G is simple and connected. We do not require this assumption.

[10]This is the default initialization method for weight matrices in Chainer and Chainer Chemistry.

Table 4.1: Dataset specifications for original citation networks. The threshold λ^{-1} in the table indicates the upper bound of Corollary 4.2.

	#Node	#Edge	#Class	Chance Rate	Threshold λ^{-1}
Cora	2708	5429	6	30.2%	$1 + 3.62 \times 10^{-3}$
CiteSeer	3312	4732	7	21.1%	$1 + 1.25 \times 10^{-3}$
PubMed	19717	44338	3	39.9%	$1 + 9.57 \times 10^{-3}$

4.B.3 Experiments in Section 4.6.3

Dataset

We used the Cora [McCallum et al., 2000, Sen et al., 2008], CiteSeer [Giles et al., 1998, Sen et al., 2008], and PubMed[Sen et al., 2008] datasets for experiments. We obtained the preprocessed dataset from the code repository of Kipf and Welling [2017][11]. Table 4.1 summarizes specifications of datasets and their noisy version (explained in the next section).

The Cora dataset is a citation network dataset consisting of 2708 papers and 5429 links. Each paper is represented as the occurence of 1433 unique words and is associated to one of 7 genres (Case Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning, Theory). The graph made from the citation links has 78 connected components and the smallest positive eigenvalue of the augmented Normalized Laplacian is approximately $\tilde{\mu} = 3.62 \times 10^{-3}$. Therefore, the upper bound of Theorem 4.2 is $\lambda^{-1} = (1 - \tilde{\mu})^{-1} \approx 1 + 3.62 \times 10^{-3}$. 818 out of 2708 samples are labelled as “Probabilistic Methods", which is the largest proportion. Therefore, the chance rate is $818/2708 = 30.2\%$.

The CiteSeer dataset is a citation network dataset consisting of 3312 papers and 4732 links. Each paper is represented as the occurence of 3703 unique words and is associated to one of 6 genres (Agents, AI, DB, IR, ML, HCI). The graph made from the citation links has 438 connected components and the smallest positive eigenvalue of the augmented Normalized Laplacian is approximately $\tilde{\mu} = 1.25 \times 10^{-3}$. Therefore, the upper bound of Theorem 4.2 is $\lambda^{-1} = (1 - \tilde{\mu})^{-1} \approx 1 + 1.25 \times 10^{-3}$. 701 out of 2708 samples are labelled as “IR", which is the largest proportion. Therefore, the chance rate is $701/3312 = 21.1\%$.

The PubMed dataset is a citation network dataset consisting of 19717 papers and 44338 links. Each paper is represented as the occurence of 500 unique words and is associated to one of 3 genres (“Diabetes Mellitus, Experimental", “Diabetes Mellitus Type 1", “Diabetes Mellitus Type 2"). The graph made from the citation links has 438 connected components and the smallest positive eigenvalue of the augmented Normalized Laplacian is approximately $\tilde{\mu} = 9.48 \times 10^{-3}$. Therefore, the upper bound of Theorem 4.2 is $\lambda^{-1} = (1 - \tilde{\mu})^{-1} \approx 1 + 9.57 \times 10^{-3}$. 7875 out of 19717 samples are labelled as “Diabetes Mellitus Type 2", which is the largest proportion. Therefore, the chance rate is $7875/19717 = 39.9\%$.

Table 4.2: Dataset specifications for noisy citation networks. The threshold λ^{-1} in the table indicates the upper bound of Corollary 4.2.

	Original Dataset	#Edge Added	Threshold λ^{-1}
Noisy Cora 2500	Cora	2495	1.11
Noisy Cora 5000	Cora	4988	1.15
Noisy CiteSeer	CiteSeer	4991	1.13
Noisy PubMed	PubMed	24993	1.17

Noisy Citation Networks

We summarize the properties of noisy citation networks in Table 4.2.

We created two datasets from the Cora dataset: Noisy Cora 2500 and Noisy Cora 5000. *Noisy Cora 2500* is made from the Cora dataset by uniformly randomly adding 2500 edges, respectively. Since some random edges are overlapped with existing edges, the number of newly-added edges is 2495 in total. We only changed the underlying graph from the Cora dataset and did not change word occurences (feature vectors) and genres (labels). The underlying graph of the Noisy Cora dataset has two connected components and the smallest positive eigenvalue is $\tilde{\mu} \approx 9.62 \times 10^{-2}$. Therefore, the threshold of the maximum singular values of in Theorem 4.2 has been increased to $\lambda^{-1} = (1-\tilde{\mu})^{-1} \approx 1.11$. Similarly, *Noisy Cora 5000* was made by adding 5000 edges uniformaly randomly. The number of newly added edges is 4988 and the graph is connected (i.e., it has only 1 connected component). $\tilde{\mu}$ and λ are $\tilde{\mu} \approx 1.32 \times 10^{-1}$ and $\lambda = (1-\tilde{\mu})^{-1} \approx 1.15$, respectively.

We made the noisy version of CiteSeer (*Noisy CiteSeer*) and PubMed (*Noisy PubMed*), in the similar way, by adding 5000 and 25000 edges uniformly randomly to the datasets. Since some random edges were overlapped with existing edges, 4991 and 24993 edges are newly added, respectively. This manipulation reduced the number of connected component of the graph to 3. $\tilde{\mu}$ is approximately 1.11×10^{-1} (Noisy CiteSeer) and 1.43×10^{-1} (Noisy PubMed) and $\lambda^{-1} = (1-\tilde{\mu})^{-1}$ is approximately 1.13 (Noisy CiteSeer) and 1.17 (Noisy PubMed), respectively. Figure 4.5 (right) shows the spectral distribution of the augmented normalized Laplacian For comparison, we show in Figure 4.5 (left) the spectral distribution of the normalized Laplacian for these datasets[12].

Model Architecture

We used a GCN consisting of a single node embedding layer, one to nine graph convolution layers, and a readout operation [Gilmer et al., 2017], which is a linear transformation common to all nodes in our case. We applied softmax function to the output of GCN. The output dimension of GCN is same as the number of classes (i.e., seven for Noisy Cora 2500/5000, six for Noisy CiteSeer, and three for Noisy PubMed). We treated the number of units in each graph convolution layer as a hyperparameter. Optionally, we specified the maximum singular values s of graph convolution layers. The choice of s is either 0.5 (smaller than 1), s_1 (in the interval $\{1 \leq s < \lambda^{-1}\}$), 3 and 10

[11] `https://github.com/tkipf/gcn` (Retrieved on December, 2nd, 2020)

[12] Due to computational resource problems, we cannot compute the spectral distributions for PubMed and Noisy Pubmed.

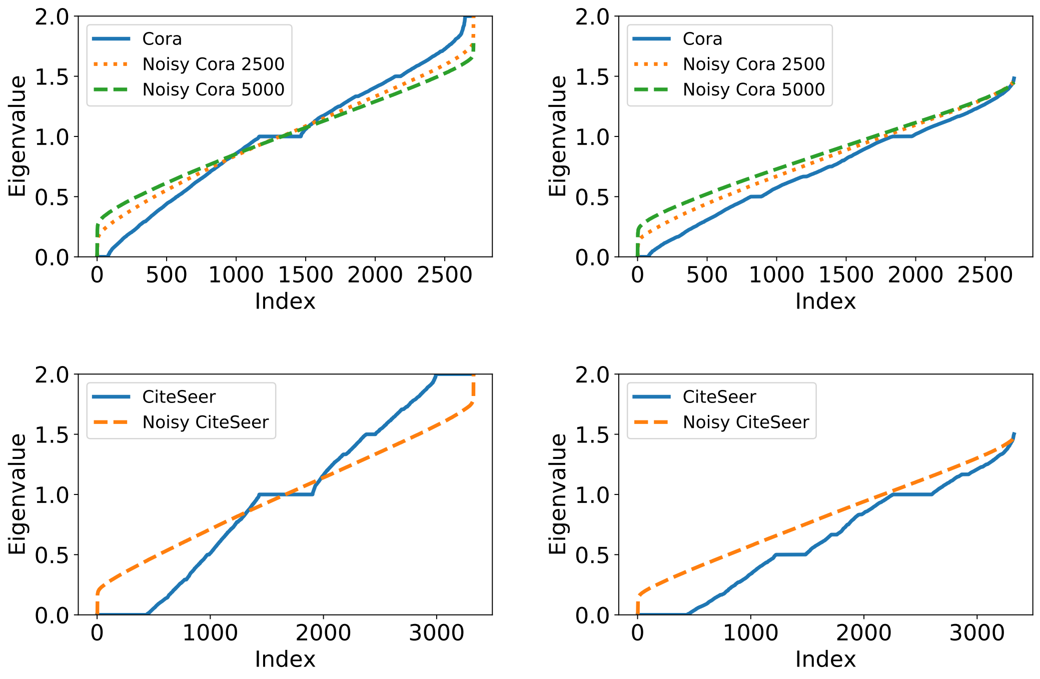


Figure 4.5: Spectral distribution of Laplacian for the citation network datasets. Left: normalized Laplacian. Right: augmented normalized Laplacian. Top: Cora and Noisy Cora (2500, 5000). Bottom: CiteSeer and Noisy CiteSeer.

(larger than λ^{-1}). We used $s_1 = 1.05$ for Noisy Cora 2500, Noisy CiteSeer, and Noisy PubMed, and $s_1 = 1.1$ for Noisy Cora 5000 and Noisy CiteSeer so that s_1 is not close to the edges of the the interval $\{1 \leq s < \lambda^{-1}\}$.

Performance Evaluation Procedure

We split all nodes in a graph (either Noisy Cora 2500/5000 or Noisy CiteSeer) into training, validation, and test sets. Data split is the same as the one done by Kipf and Welling [2017]. This is a transductive learning [Pan and Yang, 2010] setting because we can use node properties of the validation and test data during training. We trained the model three times for each choice of hyperparemeters using the training set and defined the objective function as the average accuracy on the validation set. We chose the combination of hyperparameters that achieves the best value of objective function. We evaluate the accuracy of the test dataset three times using the chosen combination of hyperparameters and computed their average and the standard deviation.

Training

At initialization, we sampled parameters from the i.i.d. Gaussian distribution. If the scale of maximum singular values s was specified, we subsequently scaled weight matrices of graph con-

Table 4.3: Hyperparameters of the experiment in Section 4.6.3. $X \sim \mathrm{LogUnif}[10^a, 10^b]$ denotes the random variable $\log_{10} X$ obeys the uniform distribution over $[a, b]$. "Learning rate" corresponds to α when "Optimization algorithm" is Adam [Kingma and Ba, 2015].

Name	Value
Unit size	$\{10, 20, \ldots, 500\}$
Epoch	$\{10, 20, \ldots, 100\}$
Optimization algorithm	$\{\mathrm{SGD}, \mathrm{MomentumSGD}, \mathrm{Adam}\}$
Learning rate	$\mathrm{LogUnif}[10^{-5}, 10^{-2}]$

volution layers so that their maximum singular values were normalized to s. The loss function was defined as the sum of the cross entropy loss for all training nodes. We train the model using the one of gradient-based optimization methods described in Table 4.3.

Hyperprameters

Table 4.3 shows the set of hyperparameters from which we chose. Since we compute the representations of all nodes at once at each iteration, each epoch consists of 1 iteration. We employ Tree-structured Parzen Estimator [Bergstra et al., 2011] for hyperparameter optimization.

Implementation

We used Chainer Chemistry[13], which is an extension library for the deep learning framework Chainer [Tokui et al., 2015, 2019], to implement GCNs and Optuna [Akiba et al., 2019] for hyperparameter tuning. We conducted experiments in a signel machine which has 2 Intel(R) Xeon(R) Gold 6136 CPU@3.00GHz (24 cores), 192 GB memory (DDR4), and 3 GPGPUs (NVIDIA Tesla V100). Our implementation achieved 68.1% with Dropout [Srivastava et al., 2014] (2 graph convolution layers) and 64.2% without Dropout (1 graph convolution layer) on the test dataset. These are slightly worse than the accuracy reported in Kipf and Welling [2017], but are still comparable with it.

4.B.4 Experiments in Section 4.6.4

The experiment settings are almost same as the experiment in Section 4.6.3. The only difference is that we did not train the node embedding layer, which we put before convolution layers of a GCN, while we did in Section 4.6.3. This is because we wanted to see the the effect of convolution operations on the perpendicular component of signals, while we interested in the prediction accuracy in real training settings in the previous experiment.

[13]`https://github.com/pfnet-research/chainer-chemistry` (Retrieved on December, 2nd, 2020)

4.C Additional Experiment Results

4.C.1 Experiments in Section 4.6.1

Figure 4.6 (**Case 1**) and Figure 4.7 (**Case 2**) show the vector field V for various W. Parameters other than W are same as experiments in Section 4.6.1 (detail values are available in Section 4.B.1).

4.C.2 Experiments in Section 4.6.2

Figure 4.8 shows the relative log distance of signals and their upper bound for various edge probability p and the maximum singular value s. Note that we generate a new graph for each configuration of (p, s). Therefore, different configurations may have different graphs and hence different λ even they have a same edge probability p in common.

4.C.3 Experiments in Section 4.6.3

Predictive Accuracy Figure 4.9 shows the comparison of predictive performance in terms the maximum singular value and layer size when the dataset is Noisy Cora 5000 (left) and Noisy Citeseer (right), respectively. Concrete values are available in Table 4.4.

Transition of Maximum Singular Values Figure 4.10 – 4.13 show the transition of weight of graph convolution layers during training when the dataset is Noisy Cora 2500, Noisy Cora 5000, and Noisy CiteSeer, respectively. We note that the result of 3-layered GCN from the Noisy Cora 2500 is identical to Figure 4.3 (right) of the main article.

4.C.4 Experiments in Section 4.6.4

Figure 4.14 shows the logarithm of relative perpendicular component and prediction accuracy on Noisy Cora, Noisy CiteSeer, and Noisy PubMed datasets. We use Pearson R as a correlation coefficient. If GCNs have only one layer, it has more large relative perpendicular components (corresponding to right points in the figures) than GCNs which have other number of layers. The correlation between the logarithm of relative perpendicular components and prediction accuracies are $0.827 (p = 6.890 \times 10^{-6})$ for Noisy Cora, $0.524 (p = 1.771 \times 10^{-2})$ for Noisy CiteSeer, and $0.679 (p = 1.002 \times 10^{-3})$ for Noisy PubMed, if we treat the one-layer case as outliers and remove them.

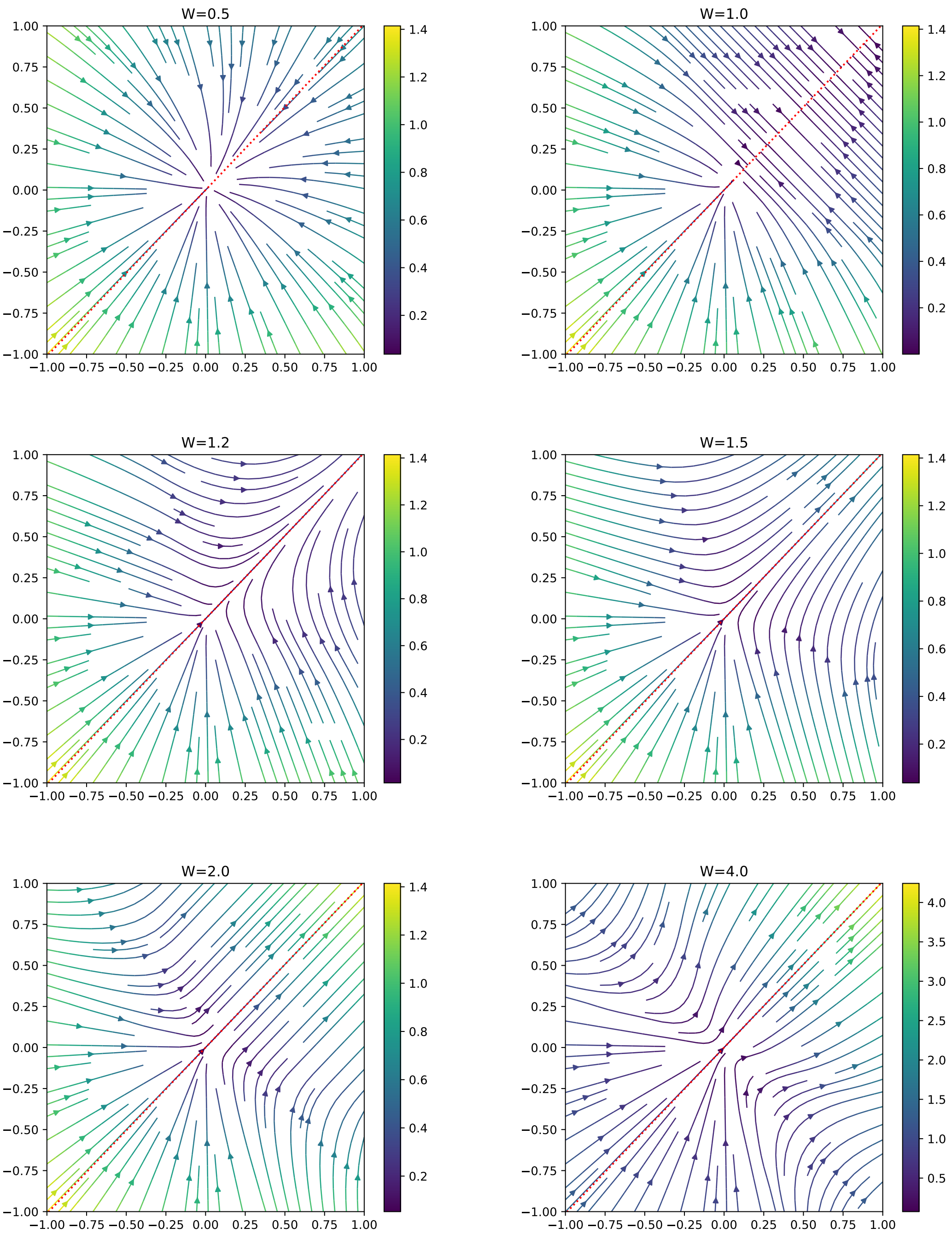


Figure 4.6: Vector field V for various weights W for **Case 1**. Top left: $W = 0.5$. Top right: $W = 1.0$. Middle left: $W = 1.2$ (same as Figure 4.1 in the main article). Middle right: $W = 1.5$. Bottom left: $W = 2.0$. Bottom right: $W = 4.0$. Best view in color.

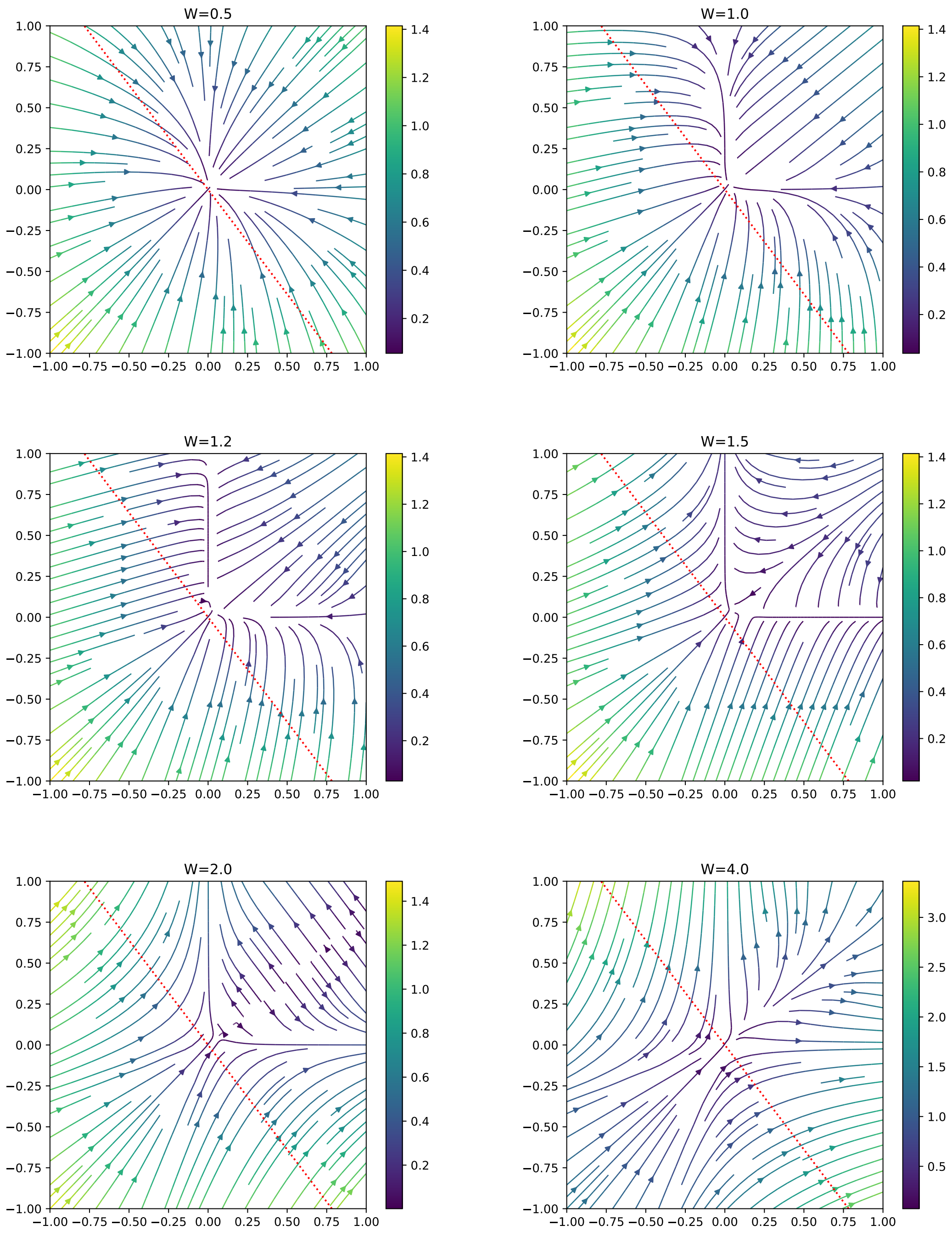


Figure 4.7: Vector field V for various weights W for **Case 2**. Top left: $W = 0.5$. Top right: $W = 1.0$. Middle left: $W = 1.2$ (same as Figure 4.1 in the main article). Middle right: $W = 1.5$. Bottom left: $W = 2.0$. Bottom right: $W = 4.0$. Best view in color.

Table 4.4: Accuracy in terms of maximum singular value of weights and layer size. “U" in the right most column indicates the accuracy of GCN without weight normalization.

Noisy Cora 2500

	Maximum Singular Value				
Depth	1	1.05	3	10	U
1	0.389 ± 0.101	0.429 ± 0.090	0.552 ± 0.014	0.632 ± 0.007	0.587 ± 0.008
3	0.273 ± 0.051	0.309 ± 0.017	0.580 ± 0.058	0.661 ± 0.003	0.494 ± 0.041
5	0.319 ± 0.000	0.267 ± 0.059	0.462 ± 0.065	0.602 ± 0.004	0.326 ± 0.029
7	0.261 ± 0.076	0.262 ± 0.080	0.407 ± 0.021	0.501 ± 0.017	0.279 ± 0.129
9	0.261 ± 0.080	0.319 ± 0.000	0.284 ± 0.109	0.443 ± 0.014	0.319 ± 0.000

Noisy Cora 5000

	Maximum Singular Value				
Depth	1	1.1	3	10	U
1	0.301 ± 0.080	0.333 ± 0.099	0.557 ± 0.004	0.561 ± 0.019	0.555 ± 0.016
3	0.245 ± 0.066	0.247 ± 0.076	0.370 ± 0.041	0.587 ± 0.009	0.286 ± 0.066
5	0.274 ± 0.048	0.237 ± 0.070	0.257 ± 0.076	0.535 ± 0.031	0.319 ± 0.000
7	0.263 ± 0.080	0.297 ± 0.031	0.260 ± 0.074	0.339 ± 0.060	0.319 ± 0.000
9	0.262 ± 0.081	0.258 ± 0.064	0.262 ± 0.080	0.261 ± 0.082	0.318 ± 0.002

Noisy CiteSeer

	Maximum Singular Value				
Depth	0.5	1.1	3	10	U
1	0.461 ± 0.018	0.467 ± 0.012	0.490 ± 0.016	0.494 ± 0.006	0.495 ± 0.009
3	0.438 ± 0.027	0.436 ± 0.010	0.450 ± 0.019	0.462 ± 0.007	0.417 ± 0.061
5	0.285 ± 0.008	0.371 ± 0.016	0.373 ± 0.011	0.425 ± 0.007	0.380 ± 0.024
7	0.213 ± 0.006	0.282 ± 0.011	0.309 ± 0.012	0.385 ± 0.007	0.308 ± 0.012
9	0.182 ± 0.005	0.242 ± 0.030	0.303 ± 0.021	0.325 ± 0.003	0.229 ± 0.033

Noisy Pubmed

	Maximum Singular Value				
Depth	0.5	1.1	3	10	U
1	0.488 ± 0.039	0.636 ± 0.006	0.641 ± 0.010	0.632 ± 0.002	0.631 ± 0.010
3	0.442 ± 0.027	0.426 ± 0.026	0.658 ± 0.004	0.661 ± 0.005	0.631 ± 0.013
5	0.431 ± 0.033	0.431 ± 0.034	0.561 ± 0.083	0.641 ± 0.004	0.424 ± 0.093
7	0.428 ± 0.032	0.443 ± 0.051	0.449 ± 0.035	0.619 ± 0.011	0.440 ± 0.041
9	0.413 ± 0.009	0.438 ± 0.039	0.539 ± 0.052	0.569 ± 0.042	0.473 ± 0.031

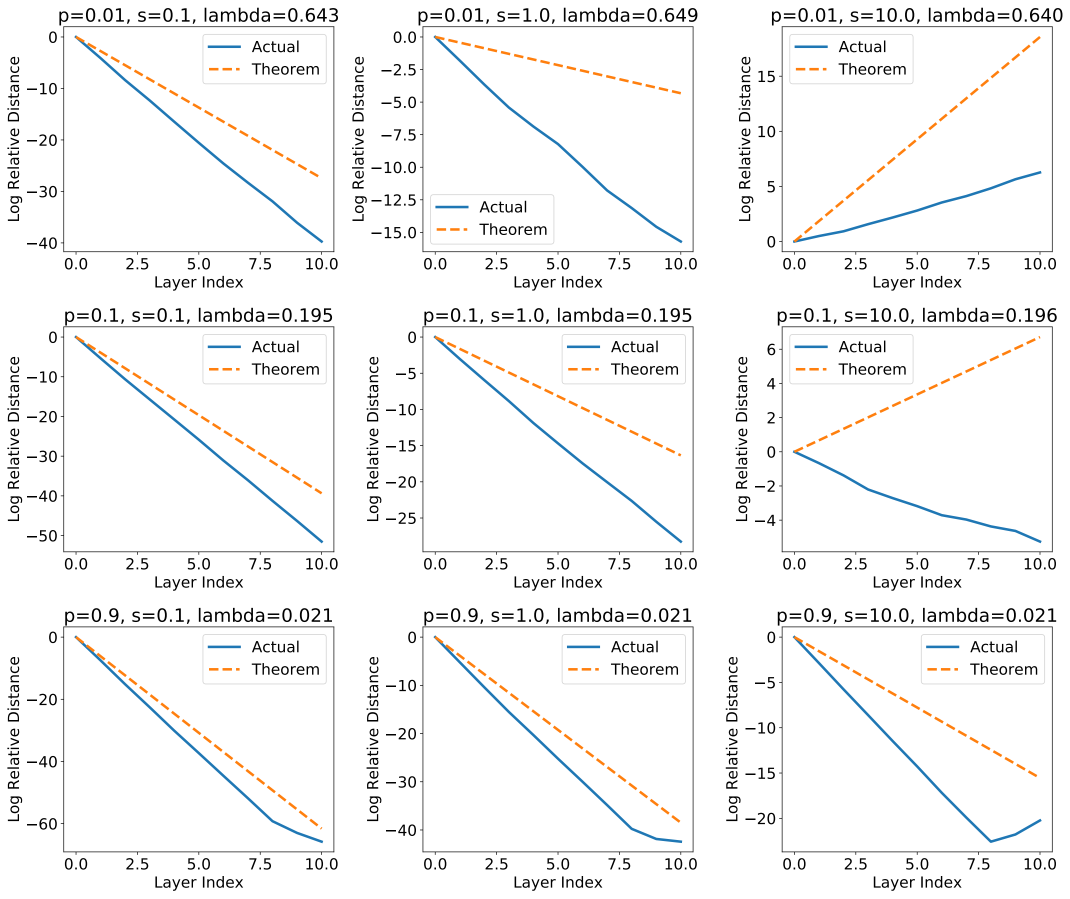


Figure 4.8: The distance $d_{\mathcal{M}}$ to the invariant space $\mathcal{M}$ and the upper bound inferred by Theorem 4.1. The edge probability p takes $0.01(\mathrm{top}), 0.1, 0.9(\mathrm{bottom})$ and the maximum singular value s takes $0.1(\mathrm{left}), 1.0, 10(\mathrm{right})$. Blue lines are the log relative distance defined by $y(l) := \log \frac{d_{\mathcal{M}}(X^{(l)})}{d_{\mathcal{M}}(X^{(0)})}$ and orange dotted lines are upper bound $y(l) := l \log(s\lambda)$, where $X^{(0)}$ is the input signal and $X^{(l)}$ is the output of the l-th layer. Best view in color.

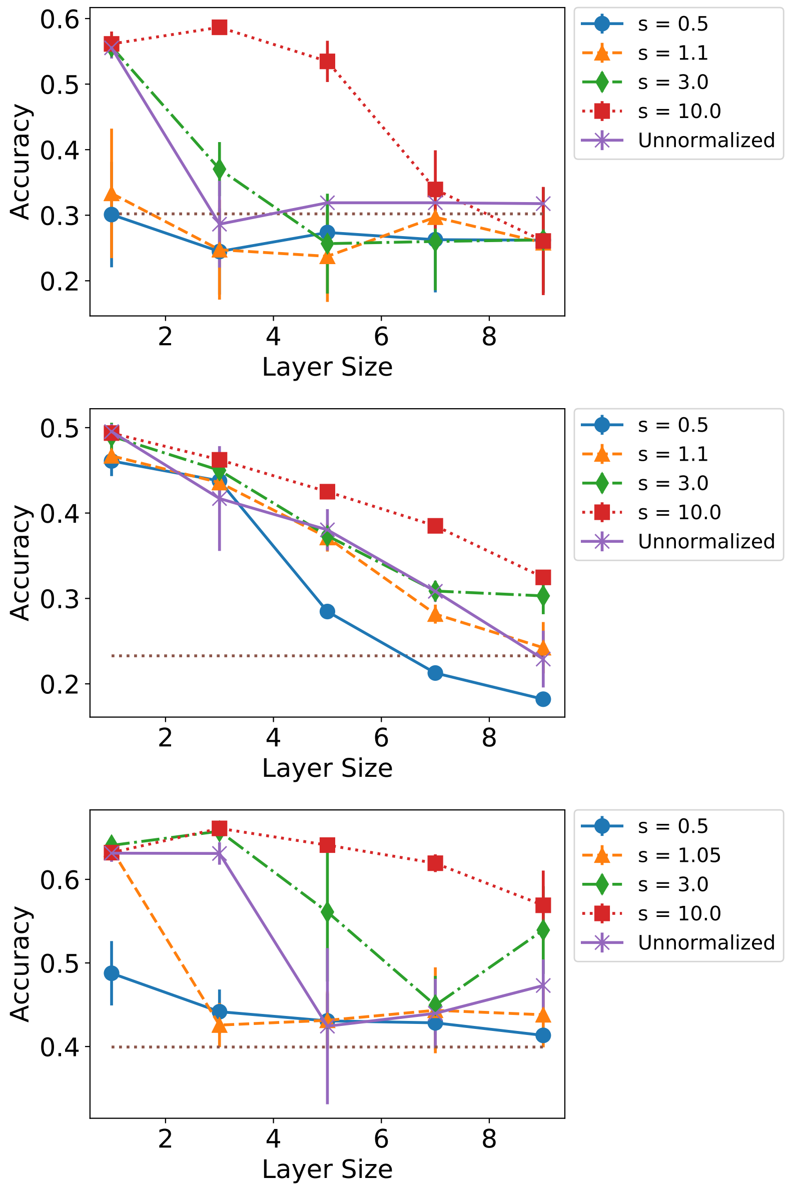


Figure 4.9: Effect of the maximum singular values of weights on predictive performance. Horizontal dotted lines indicate the chance rates (30.2% for Noisy Cora 5000, 21.2% for Noisy CiteSeer, and 39.9% for Noisy PubMed). The error bar is the standard deviation of 3 trials. Left: Noisy Cora 5000. Right: Noisy CiteSeer. Bottom: Noisy Pubmed. Best view in color.

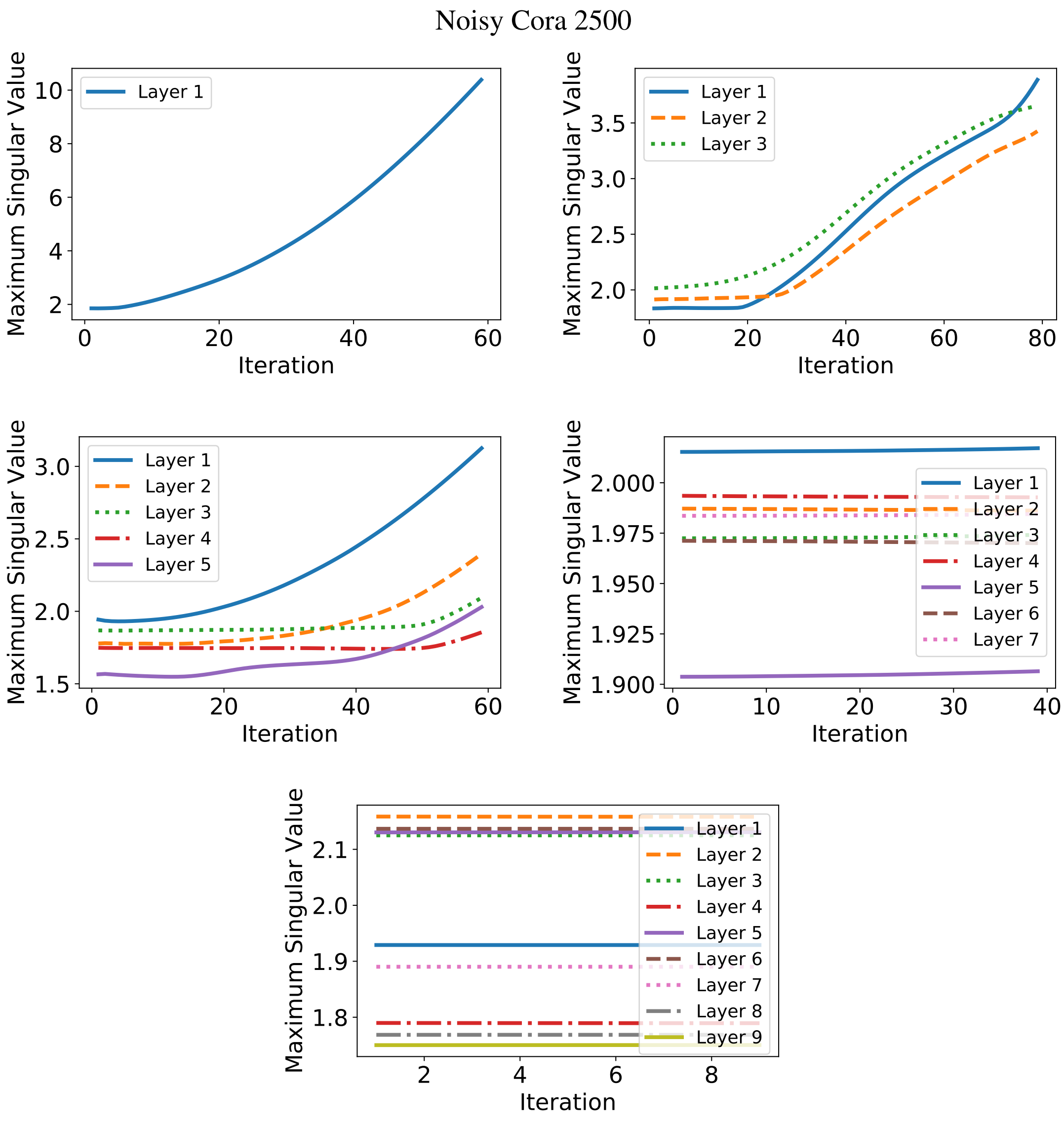


Figure 4.10: Transition of maximum singular values of GCN during training using Noisy Cora 2500. Top left: 1 layer. Top right: 5 layers. Bottom left: 7 layers. Bottom right: 9 layers.

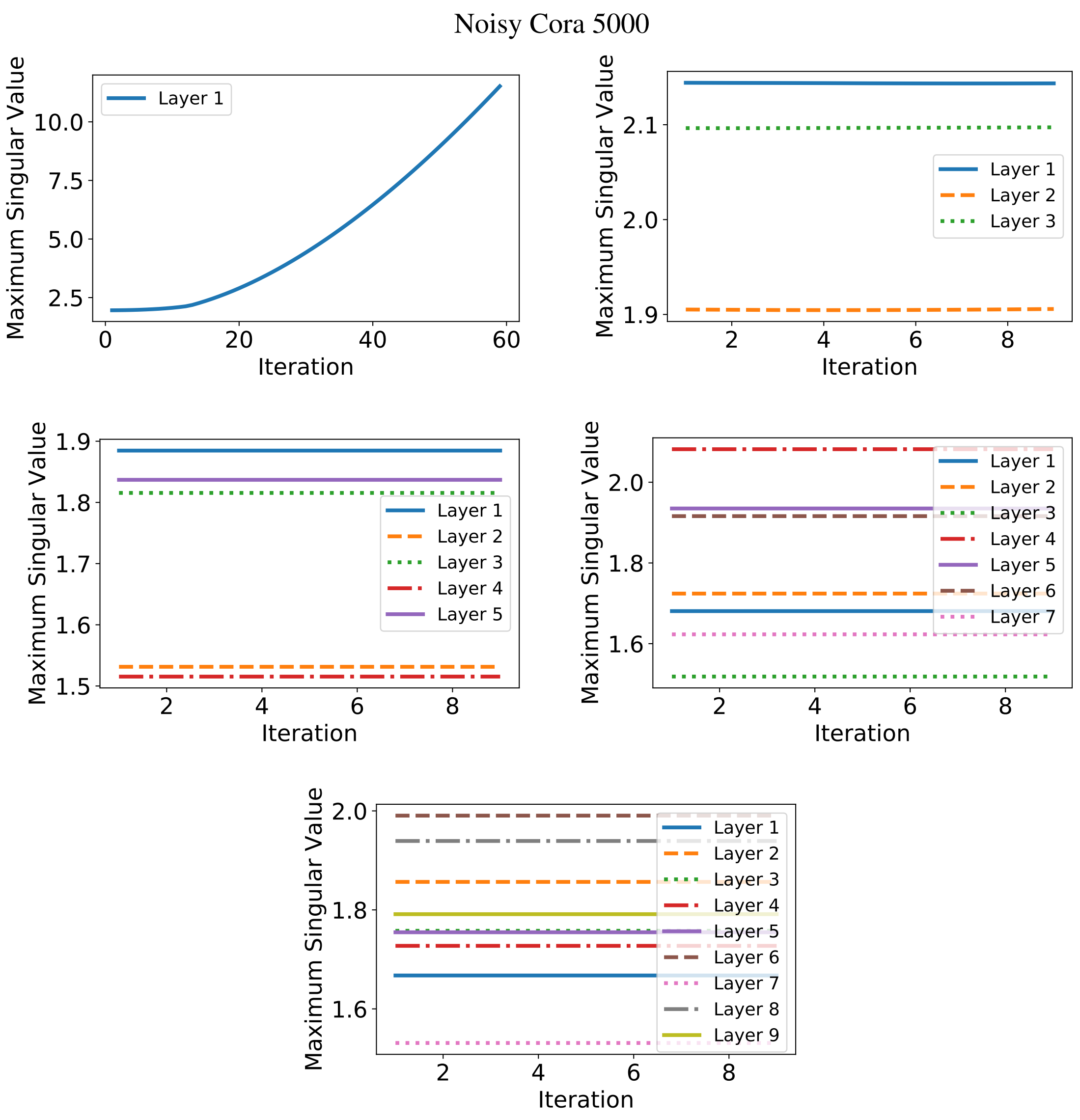


Figure 4.11: Transition of maximum singular values of GCN during training using Noisy Cora 5000. Top left: 1 layer. Top right: 3 layers. Middle left: 5 layers. Middle right: 7 layers. Bottom: 9 layers.

Noisy CiteSeer

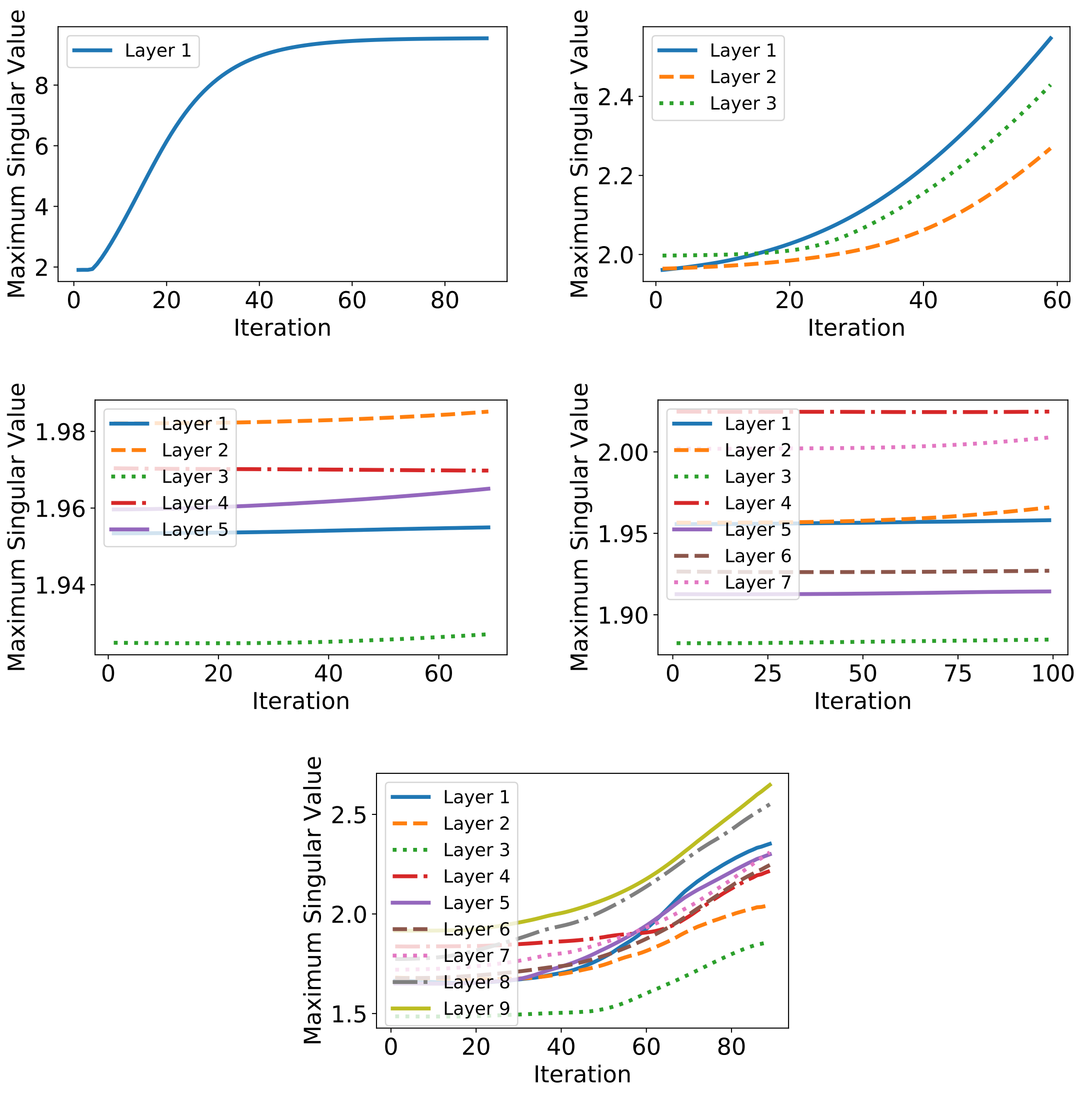


Figure 4.12: Transition of maximum singular values of GCN during training using Noisy CiteSeer. Top left: 1 layer. Top right: 3 layers. Middle left: 5 layers. Middle right: 7 layers. Bottom: 9 layers.

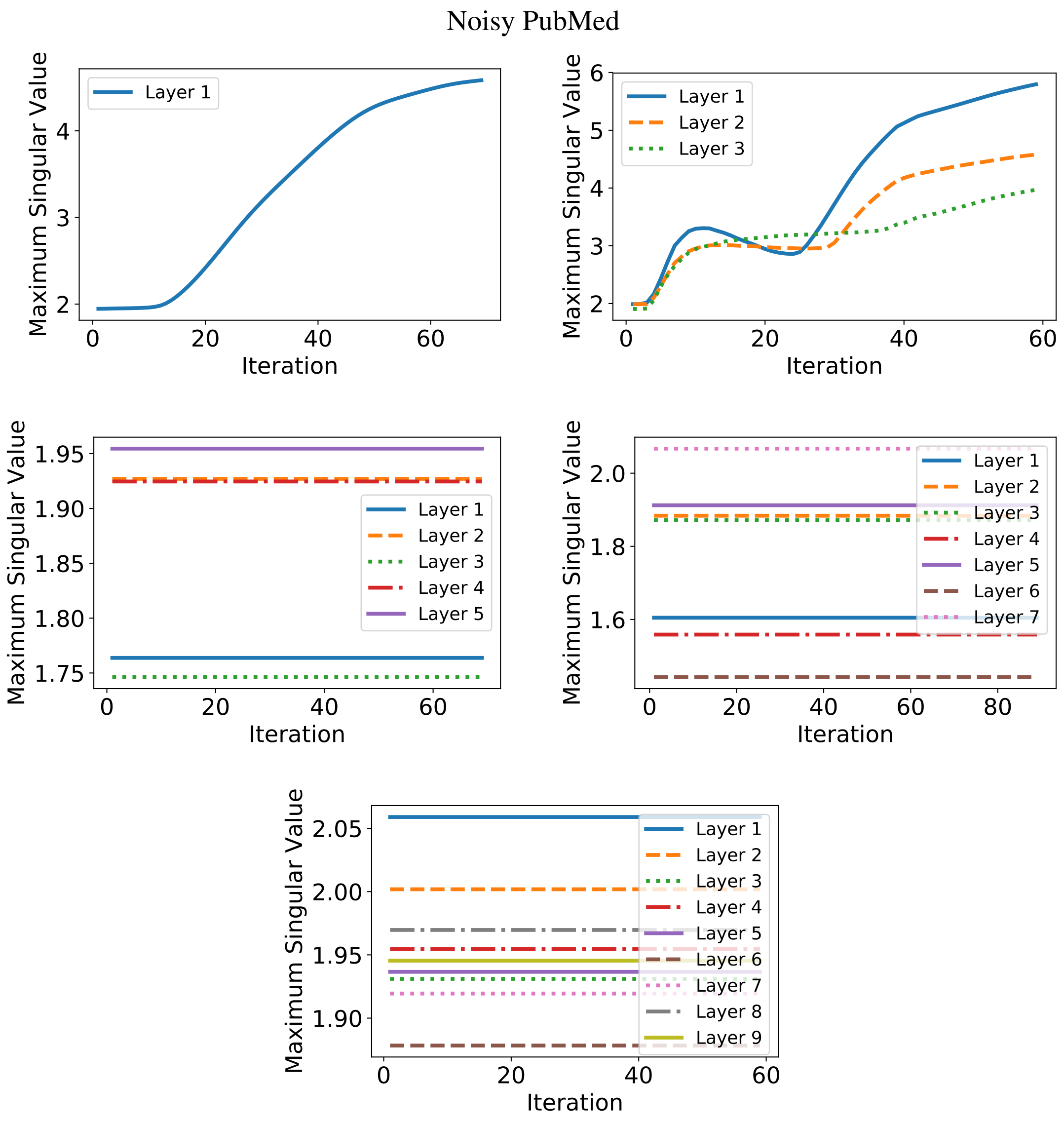


Figure 4.13: Transition of maximum singular values of GCN during training using Noisy PubMed. Top left: 1 layer. Top right: 3 layers. Middle left: 5 layers. Middle right: 7 layers. Bottom: 9 layers.

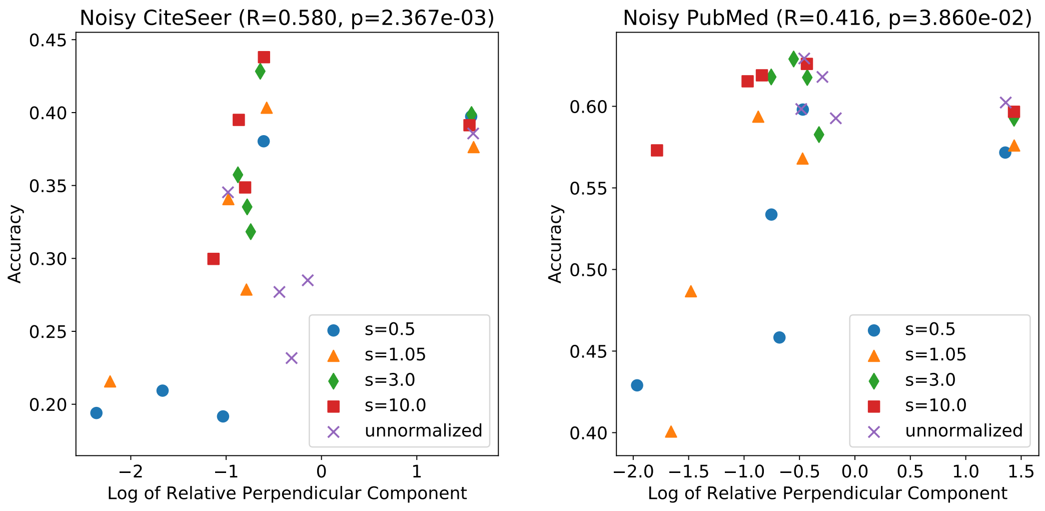


Figure 4.14: Logarithm of relative perpendicular component and prediction accuracy. Left: Noisy CiteSeer. Right: Noisy PubMed. p in the title represents the p-value for the Pearson R coefficients.

Chapter 5

Optimization and Generalization Analysis of Multi-scale Graph Neural Networks through Gradient Boosting

In this chapter, we elucidate how skip connections affect the performance of GNNs. As we have seen in the previous chapter, Graph Neural Networks (GNNs) are difficult to make themselves deeper due to the over-smoothing problem. Multi-scale GNNs are a promising approach for mitigating the over-smoothing problem. However, there is little explanation of why it works empirically from the viewpoint of learning theory. In this chapter, we derive the optimization and generalization guarantees of transductive learning algorithms that include multi-scale GNNs. Using the boosting theory, we prove the convergence of the training error under weak learning-type conditions. By combining it with generalization gap bounds in terms of transductive Rademacher complexity, we show that a test error bound of a specific type of multi-scale GNNs that decreases corresponding to the number of node aggregations under some conditions. Our results clarifies when multi-scale structures of GNNs are effective against the over-smoothing problem. We apply boosting algorithms to the training of multi-scale GNNs for real-world node prediction tasks. We confirm that its performance is comparable to existing GNNs, and the practical behaviors are consistent with theoretical observations. Code is available at `https://github.com/delta2323/GB-GNN`.

5.1 Introduction

Graph Neural Networks (GNNs) [Gori et al., 2005, Scarselli et al., 2009] are an emerging deep learning model for analyzing graph structured-data. They have achieved state-of-the-art performances in node prediction tasks on a graph in various fields such as biochemistry [Duvenaud et al., 2015], computer vision [Yang et al., 2018], and knowledge graph analysis [Schlichtkrull et al., 2018]. While they are promising, the current design of GNNs suffers from *over-smoothing* [Li et al., 2018b, Oono and Suzuki, 2020a], as we have seen in Chapter 4. Several studies suspected that this is the cause of the performance degradation of deep GNNs and devised methods to mitigate it [Rong et al., 2020, Zhao and Akoglu, 2020]. Among others, multi-scale GNNs [Liao et al., 2019b, Nguyen et al., 2017, Xu et al., 2018] are a promising approach as a solution for the over-smoothing problem. These models are designed to combine the subgraph information at various scales, for example, by bypassing the output of the middle layers of a GNN to the final layer.

Although multi-scale GNNs empirically have resolved the over-smoothing problem to some extent, little is known how it works theoretically. To justify the empirical performance from the viewpoint of statistical learning theory, we need to analyze two factors: *generalization gap* and *optimization*. There are several studies to guarantee the generalization gaps [Du et al., 2019a, Garg et al., 2020, Jacot et al., 2018, Scarselli et al., 2018, Verma and Zhang, 2019]. However, to the best of our knowledge, few studies have provided optimization guarantees. The difficulty partly originates owing to the inter-dependency of predictions. That is, the prediction for a node depends on the neighboring nodes, as well as its feature vector. It prevents us from extending the optimization theory for inductive learning settings to transductive ones.

In this chapter, we propose the analysis of multi-scale GNNs through the lens of the boosting theory [Huang et al., 2018, Nitanda and Suzuki, 2018]. Our idea is to separate a model into two types of functions – aggregation functions $\mathcal{G}$ that mix the representations of nodes and transformation functions $\mathcal{B}$, typically common to all nodes, that convert the representations to predictions. Accordingly, we can interpret a multi-scale GNN as an ensemble of supervised models and incorporate analysis tools of inductive settings. We first consider our model in full generality and prove that as long as the model satisfies the *weak learning condition* (w.l.c.), which is a standard type of assumption in the boosting theory, it converges to the global optimum. By combining it with the evaluation of the transductive version of Rademacher complexity [El-Yaniv and Pechyony, 2009], we give a sufficient condition under which a particular type of multi-scale GNNs has the upper bound of test errors that decreases with respect to depth (the number of node aggregation operations) under the w.l.c. This is in contrast to usual GNNs suffering from the over-smoothing problem. Finally, we apply multi-scale GNNs trained with boosting algorithms, termed *Gradient Boosting Graph Neural Network* (GB-GNN), to node prediction tasks on standard benchmark datasets. We confirm that our algorithm can perform favorably compared with state-of-the-art GNNs, and our theoretical observations are consistent with the practical behaviors.

Our contributions in this chapter can be summarized as follows:

- We propose the analysis of transductive learning models via the boosting theory and derive the optimization and generalization guarantees under the w.l.c. (Theorem 5.1, Proposition 5.2).

- As a special case, we give the test error bound of a particular type of multi-scale GNNs that monotonically decreases with respect to the number of node aggregations (Theorem 5.2).

- We apply GB-GNNs, GNNs trained with boosting algorithms, to node prediction tasks on real-world datasets. We confirm that GB-GNNs perform favorably compared with state-of-the-art GNNs, and theoretical observations are consistent with the empirical behaviors.

5.2 Related Work

Graph-based Transductive Learning Algorithms Graph-based transductive learning algorithms operate on a graph given *a priori* or constructed from the representations of samples. For example, spectral graph transducer [Joachims, 2003] and the algorithm proposed in Belkin et al. [2004] considered the regularization defined by the graph Laplacian. Another example is the label propagation algorithm [Zhou et al., 2004], which propagates label information through a graph. The extension of label propagation to deep models achieved state-of-the-art prediction accuracy in semi-supervised tasks appeared in computer vision [Iscen et al., 2019]. Recently, GNNs [Gori et al., 2005, Scarselli et al., 2009] have been used to solve node prediction problems as a transductive learning task, where each sample point is represented as a node on a graph, and the goal is to predict the properties of the nodes. GNNs, especially MPNN-type (message passing neural networks) GNNs [Gilmer et al., 2017], differ from the aforementioned classical transductive learning algorithms because it mixes representations of sample points directly thorough the underlying graph.

Over-smoothing and Multi-scale GNNs Multi-scale GNNs [Abu-El-Haija et al., 2019a,b, Busch et al., 2020, Liao et al., 2019b, Luan et al., 2019, Nguyen et al., 2017, Xu et al., 2018] are a promising approach for mitigating the over-smoothing problem using the information of subgraphs at various scales. For example, the Jumping Knowledge Network [Xu et al., 2018] were intentionally designed to solve the over-smoothing problem by aggregating the outputs of the intermediate layers to the final layer. However, to the best of our knowledge, there is no theoretical explanation of why multi-scale GNNs can perform well against the over-smoothing problem. We proved that a specific instantiation of our model has a test error bound that monotonically decreases with respect to depth, thereby providing the evidence for the architectural superiority of multi-scale GNNs for the over-smoothing problem.

Boosting Interpretation of Deep Models Boosting [Freund, 1995, Schapire, 1990] is a type of ensemble method for combining several learners to create a more accurate one. For example, gradient boosting [Friedman et al., 2000, Mason et al., 2000] is a de-facto boosting algorithm owing to its superior practical performance and easy-to-use libraries [Chen and Guestrin, 2016, Ke et al., 2017, Prokhorenkova et al., 2018]. Veit et al. [2016] interpreted Residual Network (ResNet) [He et al., 2016] as a collection of relatively shallow networks. Huang et al. [2018], Nitanda and Suzuki [2018] gave another interpretation as an ensemble model and evaluated its theoretical optimization and generalization performance. In particular, Nitanda and Suzuki [2018] employed the notion of (functional) gradient boosting. Similar to these studies, we interpret a GNN as an ensemble model to derive the optimization and generalization guarantees.

AdaGCN (AdaBoosting graph convolutional network), which has been recently proposed by Sun et al. [2019], is the closest to our study. They interpreted a multi-scale GCN (graph convolutional network) [Kipf and Welling, 2017] as an ensemble model and trained it using AdaBoost [Freund and Schapire, 1995]. Although their research demonstrated the practical superiority of the boosting approach, we would argue that there is room for exploration in their theory. For example, they used the Vapnik—Chervonenkis (VC) dimension to evaluate the generalization gap. However, it is known that the VC dimension cannot explain the empirical behaviors of AdaBoost [Schapire et al., 1998] (see also [Mohri et al., 2018, Section 7.3]). Besides, they did not give optimization guarantees of AdaGCNs. In contrast, our primary goal is to devise methodologies for multi-scale GNNs with a solid theoretical backbone. To realize it, we tackle the non-i.i.d. nature of node prediction tasks and derive the optimization and refined generalization guarantees.

Generalization Analysis of GNNs in Transductive Settings It is not trivial to define the appropriate notion of generalization in a transductive learning setting because we do not need to consider the prediction accuracy of sample points that are not in a given dataset. We define the generalization gap as of discrepancy between the training and test errors in terms of the random partition of a full dataset into training and test datasets [El-Yaniv and Pechyony, 2009, Vapnik, 1982] (see Section 5.4.2 for the precise definition). This definition can admit the dependency between sample points. Furthermore, Pechyony and El-Yaniv [2009] showed that any generalization gap bound in this setting is automatically translated to the bound of the corresponding i.i.d. setting. We employed the transductive version of Rademacher complexity, introduced by El-Yaniv and Pechyony [2009] to bound generalization gaps. Similarly to supervised settings, we have the transductive version of model complexities such as the VC dimension and variants of Rademacher complexity [Tolstikhin et al., 2014, 2015]. We also have transductive PAC-Bayes bounds [Bégin et al., 2014] and stability-based bounds [Cortes et al., 2008, El-Yaniv and Pechyony, 2006] for generalization analysis.

Although several research have studied generalization of GNNs [Du et al., 2019a, Garg et al., 2020, Jacot et al., 2018, Scarselli et al., 2018, Verma and Zhang, 2019], to the best of our knowledge, none of them satisfies for our purpose. Scarselli et al. [2018] derived the upper bound of the VC dimension of GNNs. However, the derivation is specific to their model and does not apply to other GNNs. Du et al. [2019a] incorporated the idea of Neural Tangent Kernels [Jacot et al., 2018] and derived a generalization gap by reducing it to a kernel regression problem. However, they considered graph prediction problems, where each sample point itself is represented as a graph drawn from some distribution, while our problem is a node prediction problem. Verma and Zhang [2019] derived the generalization gap bounds for node prediction tasks using the stability argument. However, they only considered a GNN with a single hidden layer. It is not trivial to extend their result to multi-layered and multi-scale GNNs. Similarly to our study, [Garg et al., 2020] employed the (inductive) Rademacher complexity. However, because they did not discuss the optimization guarantee, we cannot directly derive the test error bounds from their analysis.

5.3 Problem Settings

5.3.1 Transductive Learning

We review the problem setting of transductive learning problems explained in Section 2.7.6. Let $\mathcal{X}$ and $\mathcal{Y}$ be spaces of feature vectors and labels, respectively. Let $N \in \mathbb{N}_+$ be the sample size and $V := [N]$ be the set of indices of the sample. For each sample point $i \in V$, we associate a feature-label pair $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. Let V_{train} and V_{test} be the set of training and test samples, respectively, satisfying $V_{\mathrm{train}} \cap V_{\mathrm{test}} = \emptyset$ and $V_{\mathrm{train}} \cup V_{\mathrm{test}} = V$. We denote the training and test sample sizes

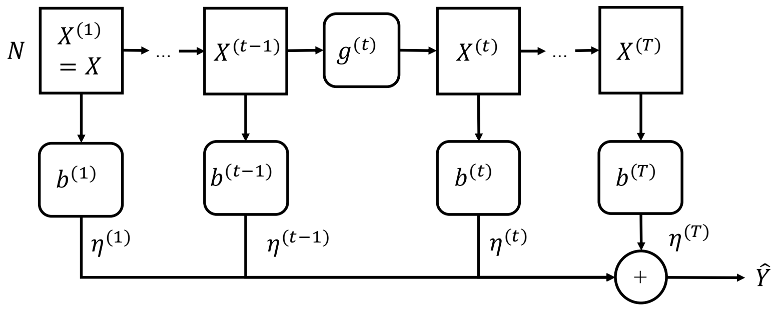


Figure 5.1: Schematic view of the model. $g^{(t)} : \mathcal{X}^N \to \mathcal{X}^N \in \mathcal{G}^{(t)}$ and $b^{(t)} : \mathcal{X}^N \to \widehat{\mathcal{Y}}^N \in \mathcal{B}^{(t)}$ are aggregation and transformation functions, respectively, and $\eta^{(t)}$ is the learning rate at the t-th iteration. We assume $\widehat{\mathcal{Y}} = \mathbb{R}$ in Sections 5.4 and 5.5 and $\mathcal{X} = \mathbb{R}^C$ in Section 5.5.

by $M := |V_{\mathrm{train}}|$ and $U := |V_{\mathrm{test}}|$, respectively. Given the collection of features $X = (x_i)_{i \in V}$ and labels $(y_i)_{i \in V_{\mathrm{train}}}$ for the training data, the task is to construct a predictor $h : \mathcal{X} \to \widehat{\mathcal{Y}}$ such that $h(x_i)$ is *close* to y_i for all $i \in V_{\mathrm{test}}$ (we define it precisely in Section 5.4.1). Here, $\widehat{\mathcal{Y}}$ denotes the range of the predictor. For later use, we define $Q := \frac{1}{M} + \frac{1}{U}$.

5.3.2 Gradient Boosting

We briefly present an overview of the gradient boosting method [Friedman, 2001, Mason et al., 2000], which is also called the restricted gradient descent [Grubb and Bagnell, 2011]. Let $\mathcal{H}$ be a subset of Hilbert space (e.g., a collection of predictors). Given a functional $\mathcal{L} : \mathcal{H} \to \mathbb{R}$ (e.g., training error), we want to find the minima of $\mathcal{L}$. Gradient boosting solves this problem by iteratively updating the predictor $h^{(t)} \in \mathcal{H}$ at each iteration t by adding a weak learner $f^{(t)}$ near the steepest direction of $\mathcal{L}$. Although a general theory can admit that $\mathcal{H}$ is infinite-dimensional (known as *functional* gradient boosting), it is sufficient for our purpose to assume that $\mathcal{H}$ is finite-dimensional. Let $\mathcal{F}^{(t)} \subset \mathcal{H}$ a hypothesis space of weak learners at iteration $t \in \mathbb{N}_+$. Gradient boosting attempts to find $f^{(t)} \in \mathcal{F}^{(t)}$ such that $f^{(t)} \in \arg\min_{f \in \mathcal{F}^{(t)}} d(-\nabla \mathcal{L}(h^{(t)}), f)$ holds true, and the step size $\eta^{(t)} > 0$. Here, d is some distance on $\mathcal{H}$ and $\nabla \mathcal{L}(h)$ is the (Fréchet) derivative of $\mathcal{L}$ at h. We update the predictor by $h^{(t+1)} = h^{(t)} + \eta^{(t)} f^{(t)}$. Because we cannot solve the minimization problem above exactly in most cases, we resort to an approximated algorithm that can find the solution near the optimal one (corresponding to Definition 5.1 below in our setting). Several boosting algorithms such as AdaBoost, Arc-x4 [Breiman, 1998], Confidence Boost [Schapire and Singer, 1999], and Logit Boost [Friedman et al., 2000] fall into this formulation by appropriately selecting $\mathcal{L}$, d and $\eta^{(t)}$'s [Mason et al., 2000].

5.3.3 Models

Figure 5.1 shows a schematic view of the model considered in this chapter. It consists of two types of components: *aggregation* functions $g^{(t)} : \mathcal{X}^N \to \mathcal{X}^N$ that mix the representations of sample points and *transformation* functions $b^{(t)} : \mathcal{X}^N \to \widehat{\mathcal{Y}}^N$ that make predictions from representations. We specify a model by defining the set of aggregation and transformation functions at each iteration t, denoted by $\mathcal{G}^{(t)}$ and $\mathcal{B}^{(t)}$, respectively. If we use the same function classes $\mathcal{G}^{(t)}$ and $\mathcal{B}^{(t)}$ for all t, we shall omit the superscript (t). Typically, a transformation function $b \in \mathcal{B}^{(t)}$ is a broadcast of the same function, i.e., b is of the form $b = (b_0, \ldots, b_0)$ for some $b_0 : \mathcal{X} \to \widehat{\mathcal{Y}}$. However, we do not assume this until necessary. We define the hypothesis space $\mathcal{F}^{(t)}$ at the t-th iteration by $\mathcal{F}^{(t)} := \{b^{(t)} \circ g^{(t)} \circ \cdots \circ g^{(1)} \mid b^{(t)} \in \mathcal{B}^{(t)}, g^{(s)} \in \mathcal{G}^{(s)} (s \in [t])\}$. Given $g^{(s)} \in \mathcal{G}^{(s)}$ selected at the $s = 1, \ldots, t-1$ iterations, we choose $g^{(t)} \in \mathcal{G}^{(t)}$ and $b^{(t)} \in \mathcal{B}^{(t)}$ to construct a weak learner $f^{(t)}(X) := b^{(t)}(g^{(t)}(X^{(t-1)}))$ and update the representation $X^{(t)} := g^{(t)}(X^{(t-1)})$. When $t = 1$, we define $f^{(1)}(X) := b^{(1)}(X)$ and $X^{(1)} := X$. We do not select $g^{(1)}$, nor do we update the representation. Algorithm 1 shows the overall training algorithm.

5.4 Main Theorems

5.4.1 Optimization

In this section, we focus on a binary classification problem. Accordingly, we set $\mathcal{Y} = \{0, 1\}$ and $\widehat{\mathcal{Y}} = \mathbb{R}$. For $\delta \geq 0$, we define $\ell_\delta(\hat{y}, y) := \mathbf{1}[(2p-1)y^\sharp < \delta]$, where $p = \mathrm{sigmoid}(\hat{y}) = (1 + \exp(-\hat{y}))^{-1}$. Note that $\ell_{\delta=0}$ is the 0–1 loss. Because it is difficult to optimize ℓ_δ, we define the sigmoid cross entropy loss $\ell_\sigma(\hat{y}, y) := -y \log p - (1-y)\log(1-p)$ as a surrogate function. For a predictor $h : \mathcal{X} \to \widehat{\mathcal{Y}}$, we define the test error by $\mathcal{R}(h) := \frac{1}{U}\sum_{n \in V_{\mathrm{test}}} \ell_\delta(h(x_n), y_n)$, and training errors by $\widehat{\mathcal{R}}(h) := \frac{1}{M}\sum_{n \in V_{\mathrm{train}}} \ell_\delta(h(x_n), y_n)$ and $\widehat{\mathcal{L}}(h) := \frac{1}{M}\sum_{n \in V_{\mathrm{train}}} \ell_\sigma(h(x_n), y_n)$. Because it is sufficient to make predictions of given samples, the values of a predictor outside of the samples do not affect the problem. Therefore, we can and do identify a predictor h with a vector $\widehat{Y} := (h(x_1), \ldots, h(x_N))^\top \in \widehat{\mathcal{Y}}^N$. Accordingly, we represent $\mathcal{R}(\widehat{Y}) := \mathcal{R}(h)$ (same is true for other errors). Similarly to previous studies [Grubb and Bagnell, 2011, Nitanda and Suzuki, 2018], we assume the following learnability condition to obtain the optimization guarantee.

Definition 5.1 (Weak Learning Condition)**.** *Let $\alpha > \beta \geq 0$, and $\boldsymbol{g} \in \mathbb{R}^N$, we say $Z \in \mathbb{R}^N$ satisfies $(\alpha, \beta, \boldsymbol{g})$-weak learning condition (w.l.c.) if it satisfies $\|Z - \alpha \boldsymbol{g}\|_2 \leq \beta \|\boldsymbol{g}\|_2$. We say a weak learner $f : \mathcal{X}^N \to \widehat{\mathcal{Y}}^N = \mathbb{R}^N$ satisfies $(\alpha, \beta, \boldsymbol{g})$-w.l.c. when $f(X)$ does.*

The following proposition provides a handy way to check the empirical satisfiability of w.l.c. It ensures that the weak learner and negative gradient face the same "direction". Using the arugment similar to Grubb and Bagnell [2011], we show that our w.l.c. is equivalent to the AdaBoost-style learnability condition [Huang et al., 2018].

Proposition 5.1. *Let $Z, \boldsymbol{g} \in \mathbb{R}^N$ such that $\boldsymbol{g} \neq 0$. There exists $\alpha > \beta \geq 0$ such that Z satisfies $(\alpha, \beta, \boldsymbol{g})$-w.l.c. if and only if $\langle Z, \boldsymbol{g} \rangle > 0$. Further, when $Z \in \{\pm 1\}^N$, this is equivalent to the condition that there exists $\delta \in (0, 1]$ such that $\sum_{n=1}^N w_n \mathbf{1}\{\mathrm{sign}(\boldsymbol{g}_n) \neq Z_n\} \leq \frac{1-\delta}{2}$ where $w_n = \frac{\boldsymbol{g}_n}{\|\boldsymbol{g}\|_1}$.*

See Section 5.A.1 for the proof. Under the condition, we have the following optimization guarantee.

Theorem 5.1. *Let $T \in \mathbb{N}_+$ and $\alpha_t > \beta_t \geq 0$ $(t \in [T])$. Define $\gamma_t := \frac{\alpha_t^2 - \beta_t^2}{\alpha_t^2}$ and $\Gamma_T := \sum_{t=1}^T \gamma_t$. If Algorithm 1 finds a weak learner $f^{(t)}$ for any $t \in [T]$, its output $\widehat{Y} \in \widehat{\mathcal{Y}}^N$ satisfies $\widehat{\mathcal{R}}(\widehat{Y}) \leq \frac{(1+e^\delta)\widehat{\mathcal{L}}(\widehat{Y}^{(1)})}{2M\Gamma_T}$. In particular, when γ_t is independent of t, the right hand side is $O(1/T)$.*

Algorithm 1 Training Algorithm

```
input  Features $X \in \mathcal{X}^N$. Labels $y_i \in \mathcal{Y}$ ($i \in V_{\text{train}}$). #iterations $T$. w.l.c. params $(\alpha_t, \beta_t)_{t\in[T]}$.
output  A collection of predictions $\widehat{Y} \in \widehat{\mathcal{Y}}^N$ for all sample points.
  Find $b^{(1)} \in \mathcal{B}^{(1)}$ and $\eta^{(1)} > 0$.
  $X^{(1)} \leftarrow X$.
  $\widehat{Y}^{(1)} \leftarrow \eta^{(1)} b^{(1)}(X)$.
  for $t = 2 \ldots T+1$ do
    Find $b^{(t)} \in \mathcal{B}^{(t)}$ and $g^{(t)} \in \mathcal{G}^{(t)}$ and set $f^{(t)}(X) \leftarrow b^{(t)}(g^{(t)}(X^{(t-1)}))$.
    Ensure $\frac{1}{M} f^{(t)}$ satisfies $(\alpha_t, \beta_t, -\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}))$-w.l.c.
    $X^{(t)} \leftarrow g^{(t)}(X^{(t-1)})$.
    $\widehat{Y}^{(t)} \leftarrow \widehat{Y}^{(t-1)} + \eta^{(t)} f^{(t)}(X)$.
  end for
  $t^* = \arg\min_{t\in[T-1]} \|\nabla\widehat{L}(\widehat{Y}^{(t)})\|_{2,1}$.
  $\widehat{Y} \leftarrow \widehat{Y}^{(t^*)}$.
```

Note that γ_t is the lower bound of the cosine value of the angle between $f^{(t)}(X)$ and $-\nabla\widehat{\mathcal{L}}(\widehat{Y})$. The proof strategy is similar to that of [Nitanda and Suzuki, 2018, Theorem 1] in that we bound the gradient of the training loss (Lemma 5.1) and apply a Kurdyka-Łojasiewicz-like inequality (Lemma 5.2). See Section 5.A.2 for the proof. We shall confirm that the w.l.c. holds empirically in the experiments in Section 5.7.1 and discuss the provable satisfiability of the w.l.c. in Section 5.8.

5.4.2 Generalization

We follow the problem setting of El-Yaniv and Pechyony [2009]. For fixed $M \in \mathbb{N}_+$, we create a training set by uniformly randomly drawing M sample points *without* replacement from V and treating the remaining U sample points as a test set. We think of training and test errors as random variables with respect to the random partition of V. El-Yaniv and Pechyony [2009] introduced the Rademacher complexity for transductive learning and derived the generalization gap bounds. We obtain the following proposition by applying it to our setting. For a hypothesis space $\mathcal{F} \subset \{\mathcal{X} \to \widehat{\mathcal{Y}}\}$, we denote its transductive Rademacher complexity by $\mathfrak{R}(\mathcal{F})$. We define $S := \frac{4(M+U)(M\wedge U)}{(2(M+U)-1)(2(M\wedge U)-1)}$, which is close to 1 when M and U are sufficiently large. See Section 5.A.3 for the definition of $\mathfrak{R}(\cdot)$ and the proof of the proposition.

Proposition 5.2. *There exists a universal constant $c_0 > 0$ such that for any $\delta' > 0$, with a probability of at least $1-\delta'$ over the random partition of samples, the output $\hat{Y}$ of Algorithm 1 satisfies*

$$\mathcal{R}(\widehat{Y}) \leq \widehat{\mathcal{R}}(\widehat{Y}) + \sum_{t=1}^{T} \eta^{(t)} \mathfrak{R}(\mathcal{F}^{(t)}) + c_0 Q\sqrt{M \wedge U} + \sqrt{\frac{SQ}{2}\log\frac{1}{\delta'}}.$$

5.5 Application to Multi-scale GNNs

We shall specialize our model and derive a test error bound for multi-scale GNNs that is monotonically decreasing with respect to T. In later sections, we assume $\mathcal{X} = \mathbb{R}^C$ for some $C \in \mathbb{N}_+$. We continue to assume that $\mathcal{Y} = \{0, 1\}$ and $\widehat{\mathcal{Y}} = \mathbb{R}$. First, we specialize $\mathcal{B}^{(t)}$ as a parallel application of the same transformation function for a single sample point of the form

$$\mathcal{B}^{(t)} := \{(f_{\mathrm{base}}, \dots f_{\mathrm{base}})^\top \mid f_{\mathrm{base}} \in \mathcal{B}^{(t)}_{\mathrm{base}}\} \tag{5.1}$$

for some $\mathcal{B}^{(t)}_{\mathrm{base}} \subset \{\mathcal{X} \to \widehat{\mathcal{Y}}\}$. By assuming this structure, we can evaluate the transductive Rademacher complexity in a similar way to the inductive case. We take multi layer perceptrons (MLPs) as base functions for an example (see Proposition 5.4 for the general case). Let $L \in \mathbb{N}_+$ $\boldsymbol{C} = (C_1, \dots, C_{L+1}) \in \mathbb{N}_+^{L+1}$ and $B > 0$ such that $C_1 = C$ and $C_{L+1} = 1$. We define $\mathcal{B}^{(t)}_{\mathrm{base}} = \mathcal{B}^{(t)}_{\mathrm{base}}(\boldsymbol{C}, L, \tilde{B}^{(t)}, \sigma)$ as a collection of L-layered MLPs with width $\boldsymbol{C}$:

$$\mathcal{B}^{(t)}_{\mathrm{base}} := \left\{\boldsymbol{x} \mapsto \sigma(\cdots \sigma(\boldsymbol{x}W^{(1)}) \cdots W^{(L-1)})W^{(L)} \mid \|W^{(l)}_{\cdot c}\|_1 \le \tilde{B}^{(t)} \text{ for all } c \in [H_{l+1}]\right\}. \tag{5.2}$$

Here, $W^{(l)} \in \mathbb{R}^{C_l \times C_{l+1}}$ for $l = 1, \dots, L$[1] and $\sigma : \mathbb{R} \to \mathbb{R}$ is a 1-Lipschitz function such that $\sigma(0) = 0$ (e.g., ReLU and sigmoid). We apply σ to a vector in an element-wise manner. For $t \in \mathbb{N}_+$, $\tilde{P}^{(t)} \in \mathbb{R}^{N \times N}$, and $\tilde{C}^{(t)} > 0$, we use the aggregation functions $\mathcal{G}^{(t)} = \mathcal{G}^{(t)}(\tilde{P}^{(t)}, \tilde{C}^{(t)})$ defined by

$$\mathcal{G}^{(t)} := \{X \mapsto \tilde{P}^{(t)}XW \mid W \in \mathbb{R}^{C \times C} \mid \|W_{\cdot c}\|_1 \le \tilde{C}^{(t)} \text{ for all } c \in [C]\}. \tag{5.3}$$

When we have a graph G whose nodes are identified with sample points, typical choices of $\tilde{P}^{(t)}$ are the (normalized) adjacency matrix A of G, its augmented variant $\tilde{A}$ used in GCN[2], (normalized) graph Laplacian, or their polynomial used in e.g., LanczosNet [Liao et al., 2019b]. We can evaluate Rademacher complexity as follows. By combining it with Propositions 5.2 and Theorem 5.1, we obtain test error bounds for multi-scale GNNs. See Sections 5.A.4 and 5.A.5 for the proof.

Proposition 5.3. *Suppose we use $\mathcal{B}^{(t)}$ and $\mathcal{G}^{(t)}$ defined above. Let $D^{(t)} = 2\sqrt{2}(2\tilde{B}^{(t)})^{L-1}\prod_{s=2}^{t}\tilde{C}^{(s)}$ and $P^{(t)} := \prod_{s=2}^{t}\tilde{P}^{(s)}$. We have $\mathfrak{R}(\mathcal{F}^{(t)}) \le \frac{1}{\sqrt{MU}}D^{(t)}\|P^{(t)}X\|_{\mathrm{F}}$.*

Theorem 5.2. *Suppose we use $\mathcal{B}^{(t)}$ and $\mathcal{G}^{(t)}$ defined above. Let $T \in \mathbb{N}_+$ and $\alpha_t > \beta_t \ge 0$ ($t \in [T]$). Suppose Algorithm 1 with the learning rate $\eta^{(t)} = \frac{4}{\alpha_t}$ finds a weak learner $f^{(t)}$ for any $t \in [T]$. Then, for any $\delta' > 0$, with a probability of at least $1 - \delta'$, its output satisfies*

$$\mathcal{R}(\widehat{Y}) \le \frac{(1 + e^{\delta})\widehat{\mathcal{L}}(\widehat{Y}^{(1)})}{2M\Gamma_T} + \frac{4}{\sqrt{MU}}\sum_{t=1}^{T}\frac{D^{(t)}\|P^{(t)}X\|_{\mathrm{F}}}{\alpha_t} + c_0 Q\sqrt{M \wedge U} + \sqrt{\frac{SQ}{2}\log\frac{1}{\delta'}}. \tag{5.4}$$

In particular, if $\Gamma_T = \Omega(T^{\varepsilon})$ for some $\varepsilon > 0$, the first term is asymptotically monotonically decreasing with respect to T. If $\alpha_t^{-1}D^{(t)}\|P^{(t)}\|_{\mathrm{op}} = O(\tilde{\varepsilon}^t)$ for some $\tilde{\varepsilon} \in (0, 1)$ independent of T, the second term is bounded by a constant independent of T.

[1] As usual, we can take into account of bias terms by preprocessing the input as $\mathbb{R}^C \ni \boldsymbol{x} \mapsto (\boldsymbol{x}, 1) \in \mathbb{R}^{C+1}$

[2] Let D be the degree matrix of G and $\tilde{D} = D + I$. We define $\tilde{A} := \tilde{D}^{-\frac{1}{2}}(A + I)\tilde{D}^{-\frac{1}{2}}$ [Kipf and Welling, 2017].

Analogy to AdaBoost Bounds By interpreting T as the depth of a GNN, this theorem clarifies how the information of intermediate layers helps to mitigate the over-smoothing problem, that is, *the deeper a model is, the better it is*. Theorem 5.2 is similar to typical test error bounds for AdaBoost in that it consists of monotonically decreasing training error terms and model complexity terms independent of T (e.g., [Mohri et al., 2018, Corollay 7.5]). While hypothesis spaces are fixed for all iterations t in the AdaBoost case, they can vary in our case due to the representation mixing caused by $\mathcal{G}^{(t)}$'s. The condition on $\|P\|_{\mathrm{op}}$ ensures that the hypothesis space does not grow significantly.

Trade-off between Model Complexity and Weak Learning Condition There is a trade-off between the model complexity and the satisfiability of the w.l.c. Suppose we use the normalized adjacency matrix A as $\tilde{P}^{(t)}$ for all t (we can alternatively use the augmented version $\tilde{A}$). If the underlying graph is connected and non-bipartite, then, it is known that the eigenvalues of A satisfies $1 = \lambda_1 > \lambda_2 \geq \cdots \geq \lambda_N > -1$ (e.g., [Chung and Graham, 1997, Lemma 1.7], [Oono and Suzuki, 2020a, Proposition 1]). Let $(\xi_n)_{n\in[N]}$ be the orthonormal basis consisting of eigenvectors of A and we decompose X as $X_c = \sum_{n=1}^{N} a_{nc}\xi_n$ ($a_{nc} \in \mathbb{R}$). We denote $\tilde{X}^{(t)} := P^{(t)}X$. Assume that $|\lambda_n|$'s are small for $n \geq 2$. On the one hand, we have $\|\tilde{X}^{(t)}\|_{\mathrm{F}}^2 = \|X\|_{\mathrm{F}}^2 - \sum_{n\geq 2}\sum_{c=1}^{C}(1-\lambda_n^{2t})a_{nc}^2$. Therefore, the model complexity terms decrease rapidly with respect to t. On the other hand, we have $\tilde{X}_c^{(t)} = \xi_{1c}a_{1c} + \sum_{n=2}^{N}\lambda_n^t\xi_n a_{nc}$. Therefore, $\tilde{X}^{(t)}$ degenerates to a rank-one vector of the form $\xi_1 \otimes v$ ($v \in \mathbb{R}^C$) quickly under the condition. Since it is known that $(\xi_1)_i \propto \deg(i)^{\frac{1}{2}}$ where $\deg(\cdot)$ is the node degree (e.g., see Chung and Graham [1997]), $\tilde{X}^{(t)}$ has little information for distinguishing nodes other than node degrees (corresponding to the information-less space $\mathcal{M}$ in Oono and Suzuki [2020a]). Therefore, it is hard for weak learners to satisfy w.l.c. using the smoothened representations $\tilde{X}^{(t)}$. We shall discuss the large model complexity case in Section 5.8.

General Transformation Functions We have used MLPs as a specific choice of $\mathcal{B}^{(t)}$. More generally, by using the proposition below, we can reduce the computation of the transductive Rademacher complexity to that of the inductive counterpart without any structural assumption on $\mathcal{B}^{(t)}$ other than the parallel function application of the form Equation (5.1). See Section 5.A.6 for the proof. Note that if the order of training and test sample sizes are same, this bound does not worsen the dependency on sample sizes. This assumption corresponds to the case where the ratio r defined below satisfies $r = \Theta(1)$ as a function of N.

Proposition 5.4. *Let $r := \frac{U}{M}$. Suppose $\mathcal{B}^{(t)}$ is of the form Equation (5.1) for some $\mathcal{B}_{\mathrm{base}}^{(t)}$. Use Equation (5.3) as $\mathcal{G}^{(s)}$ for $s \in [t]$. Define $\mathcal{F}_{\mathrm{base}}^{(t)} \subset \{\mathcal{X} \to \hat{\mathcal{Y}}\}$ by*

$$\mathcal{F}_{\mathrm{base}}^{(t)} := \{\boldsymbol{x} \mapsto f(\boldsymbol{x}W^{(2)} \cdots W^{(t)}) \mid f \in \mathcal{B}_{\mathrm{base}}^{(t)}, \|W_{c\cdot}^{(s)}\|_1 \leq \tilde{C}^{(s)}, \forall c \in [C], s = 2, \ldots, t\}.$$

We denote the (non-transductive) empirical Rademacher complexity of $\mathcal{F}_{\mathrm{base}}^{(t)}$ conditioned on $P^{(t)}X$ by $\widehat{\mathfrak{R}}_{\mathrm{ind}}(\mathcal{F}_{\mathrm{base}}^{(t)}; P^{(t)}X)$ (see Definition 5.5). Then, we have $\mathfrak{R}(\mathcal{F}^{(t)}) < \frac{(1+r)^2}{r}\widehat{\mathfrak{R}}_{\mathrm{ind}}(\mathcal{F}_{\mathrm{base}}^{(t)}; P^{(t)}X)$.

Computational Complexity The memory-efficiency is an advantage of the boosting algorithm. When we train the model without fine-tuning, we do not have to retain intermediate weights and outputs. Therefore, its memory usage is constant w.r.t. T, assuming that the memory usage of transformation functions $b^{(t)}$ are the same. This is in contrast to the ordinal GNN models trained in an end-to-end manner. Fine-tuned models use memory proportional to the depth T.

5.6 Practical Improvements

Learning Kernels The one-layer transformation of a GNN $X \mapsto AXW$ can be considered as a kernelized linear model whose Gram matrix is $\mathcal{K} = AXX^\top A$. This interpretation motivates us to select aggregate functions by learning appropriate kernels from data. We employ kernel target alignment (KTA) [Cortes et al., 2012, Cristianini et al., 2002] typically used in the context of kernel methods. Specifically, for $Z \in \mathbb{R}^{N\times C}$, we denote its Gram matrix $\mathcal{K}[Z] \in \mathbb{R}^{M\times M}$ of the training data by $\mathcal{K}[Z]_{ij} := Z_i Z_j^\top$ for $i, j \in V_{\mathrm{train}}$. We define the correlation ρ by $\rho(Z, Z') := \frac{\langle \mathcal{K}[Z], \mathcal{K}[Z']\rangle}{\|\mathcal{K}[Z]\|_{\mathrm{F}}\|\mathcal{K}[Z']\|_{\mathrm{F}}}$. Given a set of aggregation functions $\mathcal{G}^{(t)}_{\mathrm{KTA}}$, we choose the aggregation function $g^{(t)}$ such that $g^{(t)} \in \arg\max_{g\in\mathcal{G}^{(t)}_{\mathrm{KTA}}} \rho(g(X^{(t-1)}), Y)$ is approximately valid[3]. If we can assume graph structures that represent the relationships of sample points, we can utilize them to define $\mathcal{G}^{(t)}_{\mathrm{KTA}}$. For example, we used the linear combinations of various powers of the adjacency matrix in our experiments.

Input Injection A matrix $P \in \mathbb{R}^{N\times N}$ defines the aggregation model $\mathcal{G}_P := \{X \mapsto PX\}$ as we do in Section 5.7.1. Typical choices of P are the (normalized) adjacency matrix, a GCN-like augmented normalized adjacency matrix, or the (normalized) graph Laplacian. If we choose $P' := \rho P + (1-\rho)I_N$ for some $\rho \in [0, 1]$, it aggregates the representations in a lazy manner using P. In the similar spirit of Nitanda and Suzuki [2020], Zhang [2019], there is another type of lazy aggregation that allows us to inject the information of unmixed features directly to the representations. Specifically, for $\rho \in [0, 1]$, we define the *input injection* model $\mathcal{G}_{\mathrm{II}}(\rho, P)$ by

$$\mathcal{G}_{\mathrm{II}}(\rho, P) := \{X \mapsto \rho PX + (1-\rho)X^{(1)}\}.$$

On one hand, $\mathcal{G}_{\mathrm{II}}(\rho, P)$ equals $\mathcal{G}_P$ when $\rho = 1$. On the other hand, when $\rho = 0$, $\mathcal{G}_{\mathrm{II}}(\rho, P)$ ignores the effect of the representation mixing and employs the original features. We can identify $\mathcal{G}_{\mathrm{II}}(\rho, P)$ with $\{(X, X') \mapsto (\rho PX + (1-\rho)X', X')\}$. Therefore, if we redefine a new input space as $\mathcal{X}' := \mathcal{X} \times \mathcal{X}$, which means that we double the input channel size, and preprocess features as $x_i \mapsto (x_i, x_i)$ for each $i \in V$, we can think the input injection model as an example of our model. For the augmented normalized adjacency matrix $\tilde{A}$, we refer to the model that uses $\mathcal{G}_{\mathrm{II}}(\rho, \tilde{A})$ as the set of aggregation functions $\mathcal{G}^{(t)}$ and the set of MLPs as $\mathcal{B}^{(t)}$ for all t as GB-GNN-II. We conducted the same experiment as the one we do in Section 5.7.1 using GB-GNN-II. The result is reported in Section 5.D.1.

[3]When the task is a classification in which $\mathcal{Y} = [K]$, we identify $Y \in \mathcal{Y}^N$ with the matrix consisting of one-hot vectors: $\tilde{Y} = (\tilde{y}_1, \ldots, \tilde{y}_N)^\top \in \mathbb{R}^{N\times K}$ with $\tilde{y}_{nk} = \mathbf{1}\{y_n = k\}$.

Fine Tuning After the training using boosting algorithms, we can optionally fine-tune the whole model. For example, if each component of the model is differentiable, we can train it in an end-to-end manner using backpropagation. Because fine-tuning does not increase the Rademacher complexity, it does not worsen the generalization gap bound. Therefore, if fine-tuning does not increase the training error, it does not worsen the test error theoretically. However, because it is not true in all cases, we should compare the model with and without fine-tuning and select the better of the two in practice.

5.7 Experiments

5.7.1 Node Prediction Tasks

To confirm that boosting algorithms can train multi-scale GNNs practically and our theoretical observations reflect practical behaviors, we applied our models, coined *Gradient Boosting Graph Neural Networks* (GB-GNN), to node classification tasks on citation network. We used Cora [McCallum et al., 2000, Sen et al., 2008], CiteSeer [Giles et al., 1998, Sen et al., 2008], and PubMed [Sen et al., 2008] datasets. We used SAMME [Hastie et al., 2009], an extension of AdaBoost for multi-class classification tasks, as a boosting algorithm. We considered two variants as aggregation functions $\mathcal{G}$: the multiplication model $\mathcal{G}_{\tilde{A}}$ by the augmented normalized adjacency matrix $\tilde{A}$ consisting of a singleton $\mathcal{G}_{\tilde{A}} := \{X \mapsto \tilde{A}X\}$ and the KTA model $\mathcal{G}_{\mathrm{KTA}}$ (we refer to them as GB-GNN-Adj and GB-GNN-KTA, respectively). We employed MLPs with single hidden layers as transformation functions $\mathcal{B}$. See Section 5.C.1 regarding the further experiment setups. In Section 5.D.1, we report the performance of three types of model variants that use (1) MLPs with different layer size, (2) Input Injection, which is another node aggregation strategy similar to GCNII [Chen et al., 2020b], and (3) SAMME.R, a different boosting algorithm.

Table 5.1 presents the prediction accuracy. It is noteworthy that boosting algorithms greedily train the models and achieve comparable performance to existing GNNs trained in an end-to-end manner by backpropagation. Fine-tuning enhanced the performance of GB-GNN-KTA in the Cora dataset. However, whether it works well depends on model–dataset combinations. One fine-tuned model failed due to the memory error. There are two reasons. First, our implementation naively processes all nodes at once. Second, memory consumption of fine-tuning models increase proportionally to the depth T. These problems are not specific to our model but common to end-to-end deep GNN models. We can solve them by node mini-batching.

Figure 5.2 shows the transition of loss values and angle between the obtained weak learners $f^{(t)}$ and negative gradients $-\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t)})$ during the training of GB-GNN-Adj using the CiteSeer dataset. Both training and test errors keep decreasing until GB-GNN has grown up to be a deep model with as many as 40 weak learners. Accordingly, the angle is acute within this period, meaning that the w.l.c. is satisfied by Proposition 5.1. In later iterations, the training and test errors saturate, and the angle fluctuates. These behaviors are consistent with Theorem 5.2, which implies that the training and test error bounds monotonically decrease under the w.l.c. We observed similar behaviors for MLPs with various layer sizes.

Table 5.1: Accuracy of node classification tasks. Numbers denote the (mean) $\pm$ (standard deviation)(%) of ten runs. $(*)$ All runs failed due to GPU memory errors. $(**)$ We have cited the result of Kipf and Welling [2017]. See Section 5.D.2 for more comprehensive comparisons with other GNNs.

Model		Cora	CiteSeer	PubMed
GB-GNN	Adj.	79.9 ± 0.8	70.5 ± 0.8	79.4 ± 0.2
	Adj. + Fine Tuning	80.4 ± 0.8	70.8 ± 0.8	79.0 ± 0.5
	KTA	80.9 ± 0.9	73.1 ± 1.1	79.1 ± 0.4
	KTA + Fine Tuning	82.3 ± 1.1	70.8 ± 1.0	N.A.$^{(*)}$
GCN$^{(**)}$	–	81.5	70.3	79.0

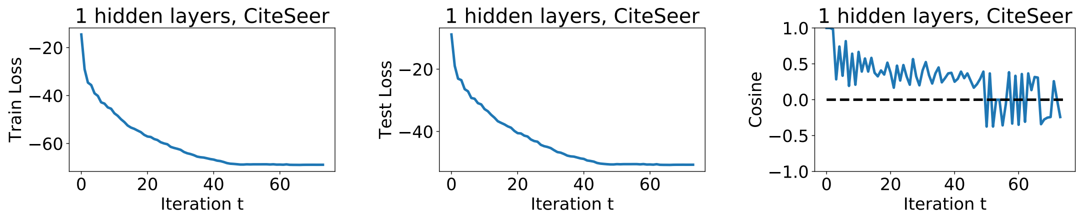


Figure 5.2: Behaviors of GB-GNN-Adj during training using the CiteSeer dataset. (Left) The transition of the training loss, (Middle) test loss, (Right) angle $\cos\theta^{(t)}$ between weak learners and negative gradients.

Table 5.2: ROC-AUC of GB-GNN for link prediction tasks. Numbers denote the (mean) $\pm$ (standard deviation)$(\%)$ of ten runs. For VGAE, we have cited the result of Kipf and Welling [2016].

Model	Cora	CiteSeer	PubMed
GB-GNN	95.8 ± 3.38	98.7 ± 0.89	92.9 ± 1.84
VGAE	91.4 ± 0.01	90.8 ± 0.02	94.4 ± 0.02

5.7.2 Link Prediction Tasks

Although our main interest is node prediction tasks, we can apply our model to other graph ML tasks. To show this, we applied GB-GNN to link prediction tasks on citation networks. In a link prediction task, we are given a graph and a collection of node features. The edge set is partially observable and contains edges for training only. The goal is to predict accurately predict whether given any node pair has an edge between them. See Section 5.C.2 for details of the experiment settings.

Table 5.2 shows the results. We compare GB-GNN with Variational Graph Auto-Encoder (VGAE) [Kipf and Welling, 2016]. We see that GB-GNN performs better than the baseline model in two out of three datasets and comparably with the baseline model in the other dataset.

5.8 Discussion

Satisfiability of Weak Learning Condition The key assumption of our theory is the w.l.c. (Definition 5.1). Although we have observed in Section 5.7.1 that the w.l.c. empirically holds, it is a natural question whether we can *provably* ensure it. To obtain the guaranteed w.l.c., the model must be sufficiently expressive so that it can approximate all possible values of the negative gradient. We can show that gradient descent can find a weak learner made of an overparameterized MLP that the w.l.c. holds with high probability, by leveraging the optimization analysis in the NTK regime [Arora et al., 2019, Du et al., 2019b] (see Section 5.B for details). However, the guaranteed w.l.c. comes at the cost of large model complexity, as evident the following proposition (see Section 5.A.7 for the proof).

Proposition 5.5. *Let $\mathcal{V}, \mathcal{V}_g \subset \mathbb{R}^N$, and $\alpha > \beta \geq 0$ such that $\{-1, 0, 1\}^N \subset \mathcal{V}_g$. If for any $\boldsymbol{g} \in \mathcal{V}_g$, there exists $Z \in \mathcal{V}$ such that Z satisfies $(\alpha, \beta, \boldsymbol{g})$-w.l.c., then, we have $\mathfrak{R}(\mathcal{V}) \geq \frac{\alpha^2 - \beta^2}{\alpha}$.*

Let $\mathcal{V}_g$ be the set of possible values that can be taken by the negative gradient and $\mathcal{V}$ be the space of outputs of weak learners at a specific iteration. Then, if we want weak learners to satisfy the w.l.c., its Rademacher complexity is inevitably as large as $\Omega(1)$ (assuming that α and β are independent of M). We leave the problem for future work whether there exists a setting that simultaneously satisfies the following conditions: (1) the w.l.c. (or similar conditions) provably holds, (2) training of weak learners is tractable, and (3) the model has a small complexity (such as the Rademacher complexity).

Choice of Transformation Functions In Section 5.5, we used an L-layered MLP with L_1 norm constraints as a transformation function $\mathcal{B}^{(t)}$. The test error bound in Theorem 5.2 can exponentially depend on L via the constant $D^{(t)}$. Since this problem occurs in inductive MLPs, too, many studies derived generalization bounds that avoid this exponential dependency [Arora et al., 2018, Nagarajan and Kolter, 2019, Wei and Ma, 2019]. We can incorporate them to obtain tighter bounds using Proposition 5.4.

Choice of Node Aggregation Functions We considered a *linear* aggregation model as $\mathcal{G}^{(t)}$ in Section 5.7.1 because GNNs that consist of linear node aggregations and non-linear MLPs is practically popular in the GNN research, such as SGC [Wu et al., 2019a], gfNN [NT and Maehara, 2019], and APPNP [Klicpera et al., 2019]. Theoretically, NT and Maehara [2019], Oono and Suzuki [2020a] claimed that non-linearity between aggregations is not essential for predictive performance.

We can alternatively use *non-linear* aggregation models. For example, consider the model $X \mapsto \sigma(\tilde{P}^{(t)}XW)$ with the same L_1 constraints as Equation (5.3) as $\mathcal{G}^{(t)}$, where $\sigma : \mathbb{R} \to \mathbb{R}$ is the ReLU function. Adding the non-linearity σ changes two things. First, $\|P^{(t)}X\|_{\mathrm{F}}$ in the bound of Theorem 5.2 is replaced with $\prod_{s=2}^{t} \|\tilde{P}^{(s)}\|_{\mathrm{op}}\|X\|_{\mathrm{F}}$. It makes the interpretation of the trade-off discussed in Section 5.5 impossible, and the bound looser, essentially because the bound loses the information of eigenvectors. Second, the bound for Rademacher complexity of $\mathcal{F}^{(t)}$ is multiplied by 2^t. It changes the condition for the monotonically decreasing test error bound with respect to T from $\alpha_t^{-1}D^{(t)}\|P^{(t)}\|_{\mathrm{op}} = O(\tilde{\varepsilon}^t)$ to a stricter one $\alpha_t^{-1}2^tD^{(t)}\prod_{s=2}^{t}\|\tilde{P}^{(s)}\|_{\mathrm{op}} = O(\tilde{\varepsilon}^t)$.

With that being said, we do not have a definitive answer whether linear aggregation models are truly superior to non-linear ones — we may be able to use techniques similar to Oono and Suzuki [2020a] for the first problem and refined analyses could eliminate the 2^t term for the second problem.

5.9 Chapter Conclusion

In this chapter, to investigate the role of skip connections in GNNs, we analyzed a certain type of transductive learning models, including multi-scale GNNs. We derived their optimization and generalization guarantees under the weak learnability condition (w.l.c.). Our idea was to interpret multi-scale GNNs as an ensemble of weak learners and apply boosting theory. As a special case, we showed that a particular type of multi-scale GNNs has a generalization bound that is decreasing with respect to the number of node aggregations under the condition. To the best of our knowledge, this is the first result that multi-scale GNNs provably avoid the over-smoothing from the viewpoint of learning theory. We confirmed that our models, coined GB-GNNs, worked comparably to existing GNNs, and that their empirical behaviors were consistent with theoretical observations. We believe that exploring deeper relationships between the w.l.c. and the underlying graph structures such as graph spectra is a promising direction for future research.

5.A Proof of Theorems and Propositions

We give proofs for the theorems and propositions in the order they appeared in the main paper.

5.A.1 Proof of Proposition 5.1

We prove the more detailed claim. Proposition 5.1 is a part of the following proposition. The third condition below means that the prediction Z is better than a random guess as a solution to the binary classification problem on the training dataset weighed by w_n's. The proof for the equivalence of the second and third conditions are similar to that of [Grubb and Bagnell, 2011, Theorem 1]

Proposition 5.6. *Let $Z, \boldsymbol{g} \in \mathbb{R}^N$ such that $\boldsymbol{g} \neq 0$. The followings are equivalent*

1. *There exist α, β such that $\alpha > \beta \geq 0$ and Z satisfies $(\alpha, \beta, \boldsymbol{g})$-w.l.c.*
2. *$\langle Z, \boldsymbol{g} \rangle > 0$.*

Under the condition, for any $r \in [\sin^2\theta, 1)$, we can take $\alpha := C$, $\beta := rC$, and $\|Z - \alpha\boldsymbol{g}\|_2 = \beta\|\boldsymbol{g}\|_2$, where

$$\cos\theta = \frac{\langle Z, \boldsymbol{g}\rangle}{\|Z\|_2\|\boldsymbol{g}\|_2}, \quad C := \frac{\langle Z, \boldsymbol{g}\rangle \pm \sqrt{\langle Z, \boldsymbol{g}\rangle^2 - (1-r^2)\|Z\|^2\|\boldsymbol{g}\|^2}}{(1-r^2)\|\boldsymbol{g}\|_2^2}.$$

Suppose further $Z \in \{\pm 1\}^N$, then, the conditions 1 and 2 are equivalent to

3. *There exists $\delta \in (0, 1]$ such that $\sum_{n=1}^N w_n \mathbf{1}\{\mathrm{sign}(\boldsymbol{g}_n) \neq Z_n\} \leq \frac{1-\delta}{2}$ where $w_n = \frac{\boldsymbol{g}_n}{\|\boldsymbol{g}\|_1}$.*

Proof. (1. $\Longrightarrow$ 2.) We have

$$\begin{aligned}\|Z - \alpha\boldsymbol{g}\|_2 \leq \beta\|\boldsymbol{g}\|_2 &\iff \|Z\|_2^2 - 2\alpha\langle Z, \boldsymbol{g}\rangle + \alpha^2\|\boldsymbol{g}\|_2^2 \leq \beta^2\|\boldsymbol{g}\|_2^2 \\ &\iff \langle Z, \boldsymbol{g}\rangle \geq \frac{\|Z\|_2^2}{2\alpha} + \frac{\alpha^2 - \beta^2}{2\alpha}\|\boldsymbol{g}\|_2^2 > 0. \end{aligned} \tag{5.5}$$

(2. $\Longrightarrow$ 1.) For $k > 0$, we define

$$\tilde{r}(k) := \frac{\|Z - k\boldsymbol{g}\|_2}{k\|\boldsymbol{g}\|_2}.$$

Then, by direct computation, we have

$$\tilde{r}(k)^2 = \frac{\|Z\|_2^2}{\|\boldsymbol{g}\|_2^2}\left(\gamma - \frac{\langle Z, \boldsymbol{g}\rangle}{\|Z\|_2^2}\right)^2 + 1 - \frac{\langle Z, \boldsymbol{g}\rangle^2}{\|Z\|_2^2\|\boldsymbol{g}\|_2^2}.$$

where $\gamma := k^{-1}$. Therefore, $\tilde{r}(k)$ is a quadratic function of γ that takes the minimum value $\sin^2\theta$ at $\gamma = \frac{\langle Z, \boldsymbol{g}\rangle}{\|Z\|_2^2} > 0$. Therefore, for any $r \in [\sin^2\theta, 1)$ there exists $k_0 > 0$ such that $\tilde{r}(k_0) = r$. Then, by setting $\alpha := k_0$ and $\beta := rk_0$, we have $\alpha > \beta \geq 0$ and

$$\|Z - \alpha\boldsymbol{g}\|_2 = \tilde{r}(k_0)\alpha\|\boldsymbol{g}\|_2 = r\alpha\|\boldsymbol{g}\|_2 = \beta\|\boldsymbol{g}\|_2.$$

By solving $\tilde{r}(k_0) = r$, we obtain $\alpha = C$ and $\beta = rC$.

(1. $\implies$ 3.) Define $w^+ := \sum_{n=1}^N w_n \mathbf{1}\{Z_n = \operatorname{sign}(\boldsymbol{g}_n)\}$ and $w^- := \sum_{n=1}^N w_n \mathbf{1}\{Z_n \neq \operatorname{sign}(\boldsymbol{g}_n)\}$. By definition, we have $w^+ + w^- = \|w\|_1 = 1$.

$$\begin{aligned}\langle Z, \boldsymbol{g} \rangle &= \sum_{n=1}^N Z_n \boldsymbol{g}_n \\ &= \sum_{n=1}^N Z_n w_n \|\boldsymbol{g}\|_1 \operatorname{sign}(\boldsymbol{g}_n) \\ &= \|\boldsymbol{g}\|_1 (w^+ - w^-). \end{aligned} \tag{5.6}$$

Therefore, using the reformulation Equation (5.5) of the assumption, we have

$$\begin{aligned} w^+ - w^- &= \frac{\langle Z, \boldsymbol{g}\rangle}{\|\boldsymbol{g}\|_1} \\ &\geq \frac{\langle Z, \boldsymbol{g}\rangle}{\|\boldsymbol{g}\|_2 \sqrt{N}} \quad (\because \text{Cauchy–Schwraz inequality})) \\ &\geq \frac{\sqrt{N}}{2\alpha \|\boldsymbol{g}\|_2} + \frac{\alpha^2 - \beta^2}{2\alpha} \frac{\|\boldsymbol{g}\|_2}{\sqrt{N}} \quad (\because \text{Equation (5.5) and } \|Z\|_2^2 = N) \\ &\geq 2\sqrt{\frac{1}{2\alpha}\frac{\alpha^2 - \beta^2}{2\alpha}} \quad (\because \text{AM–GM inequality}) \\ &= \sqrt{1 - \frac{\beta^2}{\alpha^2}}. \end{aligned}$$

Set $\delta := \sqrt{1 - \frac{\beta^2}{\alpha^2}}$. By the assumption $\alpha > \beta \geq 0$, we have $\delta \in (0, 1]$. Therefore, we have

$$\begin{aligned} w^+ &= \frac{1}{2}(w^+ + w^-) + \frac{1}{2}(w^+ - w^-) \\ &\geq \frac{1}{2}(w^+ + w^-) + \frac{\delta}{2} \\ &= \frac{1+\delta}{2}, \end{aligned}$$

which is equivalent to $w^- \leq \frac{1}{2}(1 - \delta)$.

(3. $\implies$ 2.) Using the same argument as Equation (5.6), we have

$$\langle Z, \boldsymbol{g} \rangle = \|\boldsymbol{g}\|_1 (w^+ - w^-).$$

By the assumption, we have

$$\begin{aligned} & w^- \leq \frac{1-\delta}{2}(w^+ + w^-) \\ \iff & w^+ \geq \frac{1+\delta}{2}(w^+ + w^-) \end{aligned}$$

$$\begin{aligned}
&\iff w^+ - \frac{w^+ + w^-}{2} \geq \frac{1+\delta}{2}(w^+ + w^-) - \frac{w^+ + w^-}{2} \\
&\iff \frac{1}{2}(w^+ - w^-) \geq \frac{\delta}{2}(w^+ + w^-)
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\langle Z, \boldsymbol{g} \rangle &= \|\boldsymbol{g}\|_1 (w^+ - w^-) \\
&= \|\boldsymbol{g}\|_1 \delta (w^+ + w^-) \\
&= \|\boldsymbol{g}\|_1 \delta > 0.
\end{aligned}$$

□

5.A.2 Proof of Theorem 5.1

Assumption 5.1. *$\ell : \widehat{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}$ is a non-negative C^2 convex funxtion with respect to the first variable and satisfies $|\nabla^2_{\hat{y}} \ell(\hat{y}, y)| \leq A$ for all $\hat{y} \in \widehat{\mathcal{Y}}$ and $y \in \mathcal{Y}$.*

Proposition 5.7. *The sigmoid cross entropy loss ℓ_σ satisfies Assumption 5.1 with $A = \frac{1}{4}$.*

Lemma 5.1. *Suppose the loss function ℓ satisfies Assumption 5.1 with $A > 0$. Define the training error $\widehat{\mathcal{L}}$ by $\widehat{\mathcal{L}}(\widehat{Y}) := \frac{1}{M}\sum_{n=1}^{M} \ell(\hat{y}_n, y_n)$ for $\widehat{Y}^\top = (\hat{y}_1, \ldots, \hat{y}_N)^\top$. Suppose Algorithm 1 with the learning rate $\eta^{(t)} = \frac{1}{A\alpha_t}$ finds a weak learner $f^{(t)}$ for any $t \in [T]$. Then, we have*

$$\sum_{t=1}^{T} \gamma_t \|\nabla \widehat{\mathcal{L}}(\widehat{Y}^{(t)})\|_{\mathrm{F}}^2 \leq \frac{2A\widehat{\mathcal{L}}(\widehat{Y}^{(1)})}{M}.$$

Proof. First, we define $C_f^{(t)} := (2\alpha_t)^{-1}$ and $C_{\mathcal{L}}^{(t)} := \frac{\alpha_t^2 - \beta_t^2}{2\alpha_t}$. Note that we have by definition

$$\gamma_t = 4C_f^{(t)} C_{\mathcal{L}}^{(t)}. \tag{5.7}$$

We denote $Z^{(t)\top} = (z_1^{(t)}, \ldots, z_N^{(t)})^\top := \frac{1}{M} f^{(t)}(X)^\top$ and $(\hat{y}_1^{(t)}, \ldots, \hat{y}_N^{(t)})^\top := \widehat{Y}^{(t)\top}$. Since $Z^{(t)}$ satisfies $(\alpha_t, \beta_t, -\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}))$-w.l.c., we have

$$\begin{aligned}
&\|Z^{(t)} + \alpha_t \nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})\|_{\mathrm{F}} \leq \beta_t \|\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})\|_{\mathrm{F}} \\
\iff &\|Z^{(t)}\|_{\mathrm{F}}^2 + 2\alpha_t \langle Z^{(t)}, \nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})\rangle + \alpha_t^2 \|\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})\|_{\mathrm{F}}^2 \leq \beta_t^2 \|\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})\|_{\mathrm{F}}^2 \\
\iff &\langle Z^{(t)}, \nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})\rangle + C_f^{(t)} \|Z^{(t)}\|_{\mathrm{F}}^2 \leq -C_{\mathcal{L}}^{(t)} \|\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})\|_{\mathrm{F}}^2.
\end{aligned} \tag{5.8}$$

Since $\eta^{(t)} = \frac{2C_f^{(t)}}{A}$, we have

$$\langle \nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}), Z^{(t)}\rangle + \frac{A\eta^{(t)}}{2}\|Z^{(t)}\|_{\mathrm{F}}^2 \leq -C_{\mathcal{L}}^{(t)} \|\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})\|_{\mathrm{F}}^2.$$

By Taylors' theorem, and Assumption 5.1, we have

$$\ell(\hat{y}_n^{(t)}, y_n) \leq \ell(\hat{y}_n^{(t-1)}, y_n) + \langle \nabla_{\hat{y}} \ell(\hat{y}_n^{(t-1)}, y_n), \hat{y}_n^{(t)} - \hat{y}_n^{(t-1)} \rangle + \frac{A}{2} \| \hat{y}_n^{(t)} - \hat{y}_n^{(t-1)} \|_2^2.$$

By taking the average in terms of n, we have

$$\widehat{\mathcal{L}}(\widehat{Y}^{(t)}) \leq \widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}) + \frac{1}{M} \sum_{n=1}^{M} \langle \nabla_{\hat{y}} \ell(\hat{y}_n^{(t-1)}, y_n), y_n^{(t)} - y_n^{(t-1)} \rangle + \frac{A}{2M} \sum_{n=1}^{M} \| y_n^{(t)} - y_n^{(t-1)} \|_2^2.$$

By the definition of $\widehat{Y}^{(t)}$'s, we have

$$\hat{y}_n^{(t)} - \hat{y}_n^{(t-1)} = \eta^{(t)} M z_n^{(t)}.$$

Therefore, we have

$$\begin{aligned}
\widehat{\mathcal{L}}(\widehat{Y}^{(t)}) &\leq \widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}) + \eta^{(t)} \sum_{n=1}^{M} \langle \nabla_{\hat{y}} \ell(\hat{y}_n^{(t-1)}, y_n), z_n^{(t)} \rangle + \frac{1}{2} \eta^{(t)2} A M \sum_{n=1}^{M} \| z_n^{(t)} \|_2^2 \\
&\leq \widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}) + \eta^{(t)} \sum_{n=1}^{N} \langle \nabla_{\hat{y}} \ell(\hat{y}_n^{(t-1)}, y_n), z_n^{(t)} \rangle + \frac{1}{2} \eta^{(t)2} A M \sum_{n=1}^{N} \| z_n^{(t)} \|_2^2 \\
&\leq \widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}) + \eta^{(t)} M \langle \nabla_{\hat{y}} \widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}), Z^{(t)} \rangle + \frac{1}{2} \eta^{(t)2} A M \| Z^{(t)} \|_{\mathrm{F}}^2 \\
&\leq \widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}) - \eta^{(t)} M C_{\mathcal{L}}^{(t)} \| \nabla \widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}) \|_{\mathrm{F}}^2 \quad (\because \text{Equation (5.8)}) \\
&= \widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}) - \frac{2 M C_f^{(t)} C_{\mathcal{L}}^{(t)}}{A} \| \nabla \widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}) \|_{\mathrm{F}}^2 \quad (\because \text{Definition of } \eta^{(t)}\ ) \\
&= \widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}) - \frac{M \gamma_t}{2A} \| \nabla \widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}) \|_{\mathrm{F}}^2 \quad (\because \text{Equation (5.7)})
\end{aligned}$$

Rearranging the term, we get

$$\gamma_t \| \nabla \widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}) \|_{\mathrm{F}}^2 \leq \frac{2A}{M} \left(\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}) - \widehat{\mathcal{L}}(\widehat{Y}^{(t)}) \right).$$

By taking the summation in terms of t, we have

$$\sum_{t=1}^{T} \gamma_t \| \nabla \widehat{\mathcal{L}}(\widehat{Y}^{(t)}) \|_{\mathrm{F}}^2 \leq \frac{2A}{M} \left(\widehat{\mathcal{L}}(\widehat{Y}^{(1)}) - \widehat{\mathcal{L}}(\widehat{Y}^{(T+1)}) \right) \leq \frac{2A \widehat{\mathcal{L}}(\widehat{Y}^{(1)})}{M}.$$

We used the non-negativity of the loss function $\widehat{\mathcal{L}}$ in the final inequality. □

Lemma 5.2. *Assume the loss function is the cross entropy loss. Let* $\delta \geq 0$. *For any* $\widehat{Y} \in \widehat{\mathcal{Y}}^N$, *we have*

$$\widehat{\mathcal{R}}(\widehat{Y}) \leq (1 + e^{\delta}) \| \nabla \widehat{\mathcal{L}}(\widehat{Y}) \|_{2,1}$$

Proof. Same as [Nitanda and Suzuki, 2018, Proposition C]. □

Proof of Theorem 5.1. Since we use the cross entropy loss, by Lemma 5.2, we have

$$\widehat{\mathcal{R}}(\widehat{Y}) \leq (1+e^{\delta})\|\nabla\widehat{\mathcal{L}}(\widehat{Y})\|_{2,1} \tag{5.9}$$

By the definition of $\widehat{Y}$, t^*, and Γ, we have

$$\begin{aligned}\Gamma_T\|\nabla\widehat{\mathcal{L}}(\widehat{Y})\|_{2,1} &= \left(\sum_{t=1}^{T}\gamma_t\right)\|\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t^*)})\|_{2,1}\\ &\leq \sum_{t=1}^{T}\gamma_t\|\nabla\widehat{\mathcal{L}}(y^{(t)})\|_{2,1}^2\\ &\leq \sum_{t=1}^{T}\gamma_t\|\nabla\widehat{\mathcal{L}}(y^{(t)})\|_{\mathrm{F}}^2\end{aligned} \tag{5.10}$$

From Lemma 5.1 with $A=\frac{1}{4}$ (Proposition 5.7), we have

$$\sum_{t=1}^{T}\gamma_t\|\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t)})\|_{\mathrm{F}}^2 \leq \frac{\widehat{\mathcal{L}}[\widehat{Y}^{(1)}]}{2M} \tag{5.11}$$

Combining Equation (5.9), Equation (5.10), and Equation (5.11), we have

$$\widehat{\mathcal{R}}(\widehat{Y}) \leq \frac{(1+e^{\delta})\widehat{\mathcal{L}}[\widehat{Y}^{(1)}]}{2M\Gamma_T}.$$

□

5.A.3 Proof of Proposition 5.2

The proof is basically the application of [El-Yaniv and Pechyony, 2009, Corollary 1] to our setting. We recall the definition of the transductive Rademacher complexity introduced by El-Yaniv and Pechyony [2009].

Definition 5.2 (Transductive Rademacher Complexity). *For $p \in [0,\frac{1}{2}]$ and $\mathcal{V}\subset\mathbb{R}^N$, we define*

$$\mathfrak{R}(\mathcal{V},p) := Q\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{v\in V}\boldsymbol{\sigma}\cdot v\right],$$

Here, $Q=\frac{1}{M}+\frac{1}{U}$ and $\boldsymbol{\sigma}=(\sigma_1,\ldots,\sigma_N)$ is an sequence of i.i.d. random variables whose distribution is $\mathbb{P}(\sigma_i=1)=\mathbb{P}(\sigma_i=-1)=p$ and $\mathbb{P}(\sigma_i=0)=1-2p$. In particular, we denote $\mathfrak{R}(\mathcal{V}) := \mathfrak{R}(\mathcal{V},p_0)$ where $p_0=\frac{MU}{(M+U)^2}$.

We introduce the notion of the (weighed) sum of sets.

Definition 5.3. *For $\mathcal{V}_1, \ldots, \mathcal{V}_T \subset \mathbb{R}^N$ and $\alpha_1, \ldots, \alpha_T \in \mathbb{R}$, we define their (weighted) sum $\sum_{t=1}^T \alpha_t \mathcal{V}_t$ by*

$$\sum_{t=1}^T \alpha_t \mathcal{V}_t := \left\{ \sum_{t=1}^T \alpha_t v_t \,\middle|\, v_t \in \mathcal{V}_t \right\}.$$

We define the hypothesis space $\mathcal{H}$ by $\mathcal{H} := \sum_{t=1}^T \eta^{(t)} \mathcal{F}^{(t)}$. Note that any output $\widehat{Y}$ of Algorithm 1 satisfies $\widehat{Y} \in \mathcal{H}$. We can compute the Rademacher complexity of the sum similarly to the inductive case.

Proposition 5.8. *For $\mathcal{V}_1, \mathcal{V}_2 \subset \mathbb{R}^N$, $a_1, a_2 \in \mathbb{R}$ and $p \in [0, \frac{1}{2}]$, we have $\mathfrak{R}(a_1\mathcal{V}_1 + a_2\mathcal{V}_2) \leq |a_1|\mathfrak{R}(\mathcal{V}_1) + |a_2|\mathfrak{R}(\mathcal{V}_2)$.*

Proof. Take any realization $\boldsymbol{\sigma}$ of the N i.i.d. transductive Rademacher variable of parameter p. For any $v = a_1 v_1 + a_2 v_2 \in a_1\mathcal{V}_1 + a_2\mathcal{V}_2$ ($v_1 \in \mathcal{V}_1$, $v_2 \in \mathcal{V}_2$), we have

$$\langle \boldsymbol{\sigma}, v \rangle = \langle \boldsymbol{\sigma}, a_1 v_1 \rangle + \langle \boldsymbol{\sigma}, a_2 v_2 \rangle \leq |a_1| \sup_{v_1 \in \mathcal{V}_1} \langle \boldsymbol{\sigma}, v_1 \rangle + |a_2| \sup_{v_1 \in \mathcal{V}_2} \langle \boldsymbol{\sigma}, v_2 \rangle.$$

By taking the supremum of v, we have

$$\sup_{v \in \mathcal{V}_1 + \mathcal{V}_2} \langle \boldsymbol{\sigma}, v \rangle \leq |a_1| \sup_{v_1 \in \mathcal{V}_1} \langle \boldsymbol{\sigma}, v_1 \rangle + |a_2| \sup_{v_2 \in \mathcal{V}_2} \langle \boldsymbol{\sigma}, v_2 \rangle.$$

The proposition follows by taking the expectation with respect to $\boldsymbol{\sigma}$. □

Proof of Proposition 5.2. Let $\mathcal{H} = \sum_{t=1}^T \eta^{(t)} \mathcal{F}^{(t)}$. By [El-Yaniv and Pechyony, 2009, Corollary 1], with probability of at least $1 - \delta'$ for all $\widehat{Y}' \in \mathcal{H}$, we have

$$\mathcal{R}(\widehat{Y}') \leq \widehat{\mathcal{R}}(\widehat{Y}') + \mathfrak{R}(\mathcal{H}) + c_0 Q \sqrt{M \wedge U} + \sqrt{\frac{SQ}{2} \log \frac{1}{\delta'}}.$$

Since the output $\widehat{Y}$ of Algorithm 1 satisfies $\widehat{Y} \in \mathcal{H}$, we have

$$\mathcal{R}(\widehat{Y}) \leq \widehat{\mathcal{R}}(\widehat{Y}) + \mathfrak{R}(\mathcal{H}) + c_0 Q \sqrt{M \wedge U} + \sqrt{\frac{SQ}{2} \log \frac{1}{\delta'}}. \tag{5.12}$$

By Proposition 5.8, we have

$$\mathfrak{R}(\mathcal{H}) \leq \sum_{t=1}^T \eta^{(t)} \mathfrak{R}(\mathcal{F}^{(t)}). \tag{5.13}$$

Combining Equation (5.12) and Equation (5.13) concludes the proof. □

5.A.4 Proof of Proposition 5.3

We shall prove Proposition 5.9, which is more general than Proposition 5.3. To formulate it, we first introduce the variant of the transductive Rademacher complexity.

Definition 5.4 ((Symmetrized) Transductive Rademacher Complexity)**.** *For $\mathcal{V} \subset \mathbb{R}^N$ and $p \in [0, 1/2]$, we define the symmetrized transductive Rademacher complexity $\overline{\mathfrak{R}}(\mathcal{V}, p)$ by*

$$\overline{\mathfrak{R}}(\mathcal{V}, p) := Q\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{v \in \mathcal{V}} |\langle \boldsymbol{\sigma}, v\rangle|\right].$$

We denote $\overline{\mathfrak{R}}(\mathcal{V}) := \overline{\mathfrak{R}}(\mathcal{V}, p_0)$ for $p_0 = \frac{MU}{(M+U)^2}$. For $\mathcal{F} \subset \{\mathcal{X} \to \widehat{\mathcal{Y}}\}$, we denote $\overline{\mathfrak{R}}(\mathcal{F}, p) := \overline{\mathfrak{R}}(\mathcal{V}, p)$ where $\mathcal{V} = \{(f(X_1), \ldots, f(X_N)))^\top \mid f \in \mathcal{F}\}$, where $X_1, \ldots X_N$ are feature vectors of the given training dataset defined in Section 5.4.1.

We refer to the transductive Rademacher complexity defined in Definition 5.2 as the *unsymmetrized* transductive Rademacher complexity if necessary[4]. Note that we have by definition

$$\mathfrak{R}(\mathcal{V}, p) \leq \overline{\mathfrak{R}}(\mathcal{V}, p). \tag{5.14}$$

Using the concept of the symmetrized transductive Rademacher complexity, we state the main proposition of this section.

Proposition 5.9. *Let $p \in [0, 1/2]$. Suppose we use $\mathcal{G}^{(t)}$ and $\mathcal{B}^{(t)}$ defined by Equation (5.1) and Equation (5.2) as a model. Define $D^{(t)} = 2\sqrt{2}(2\tilde{B}^{(t)})^{L-1}\prod_{s=2}^t \tilde{C}^{(s)}$ and $P^{(t)} := \prod_{s=2}^t \tilde{P}^{(s)}$. Then, we have*

$$\overline{\mathfrak{R}}(\mathcal{F}^{(t)}, p) \leq \sqrt{2p}QB^{(t)}\|P^{(t)}X\|_{\mathrm{F}}.$$

We shall prove this proposition in the end of this section. El-Yaniv and Pechyony [2009] proved the contraction property of the unnsymmetrized Rademacher complexity. We prove the contraction property for the symmetrized variant.

Proposition 5.10. *Let $\mathcal{V} \subset \mathbb{R}^N$, $p \in [0, 1/2]$. Suppose $\rho : \mathbb{R} \to \mathbb{R}$ is L_ρ-Lipschitz and $\rho(0) = 0$. Then, we have*

$$\overline{\mathfrak{R}}(\rho \circ \mathcal{V}, p) \leq 2L_\rho \overline{\mathfrak{R}}(\mathcal{V}, p),$$

where $\rho \circ \mathcal{V} := \{(\rho(v_1), \ldots, \rho(v_N))^\top \mid \boldsymbol{v} = (v_1, \ldots, v_N)^\top \in \mathcal{V}\}$.

Proof. First, by the definition $\overline{\mathfrak{R}}$ and $\rho(0) = 0$, we have

$$\begin{aligned}
&\overline{\mathfrak{R}}(\mathcal{V} \cup \{0\}, p) = \overline{\mathfrak{R}}(\mathcal{V}, p),\\
&\overline{\mathfrak{R}}(\rho \circ (\mathcal{V} \cup \{0\}), p) = \overline{\mathfrak{R}}((\rho \circ \mathcal{V}) \cup \{0\}, p) = \overline{\mathfrak{R}}(\rho \circ \mathcal{V}, p).
\end{aligned}$$

Therefore, we can assume without loss of generality that $0 \in \mathcal{V}$. Then, we have

$$\overline{\mathfrak{R}}(\rho \circ \mathcal{V}, p) = Q\mathbb{E}_{\boldsymbol{\sigma}} \sup_{v \in \mathcal{V}} |\langle \boldsymbol{\sigma}, \rho(v)\rangle|$$

[4]We are not aware the standard notion used to tell apart the complexities defined in Definitions 5.2 and 5.4. The notion of *(un)symmetrized* is specific to this paper.

$$
\begin{aligned}
&\leq Q\mathbb{E}_{\boldsymbol{\sigma}} \sup_{v\in\mathcal{V}} \langle \boldsymbol{\sigma}, \rho(v)\rangle + Q\mathbb{E}_{\boldsymbol{\sigma}} \sup_{v\in\mathcal{V}} \langle \boldsymbol{\sigma}, -\rho(v)\rangle \\
&= \mathfrak{R}(\rho\circ\mathcal{V}, p) + \mathfrak{R}(-\rho\circ\mathcal{V}, p), \qquad (5.15)
\end{aligned}
$$

where $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_N)$ are the i.i.d. transducive Rademacher variables of parameter p. We used $0 \in \mathcal{V}$ and $\rho(0) = 0$ in the inequality above. By the contraction property of the unsymmetrized transductive Rademacher complexity ([El-Yaniv and Pechyony, 2009, Lemma 1]), we have

$$
\begin{aligned}
\mathfrak{R}(\rho\circ\mathcal{V}, p) &\leq L_\rho \mathfrak{R}(\mathcal{V}, p) \\
\mathfrak{R}(-\rho\circ\mathcal{V}, p) &\leq L_\rho \mathfrak{R}(\mathcal{V}, p)
\end{aligned}
$$

By combining them with Equation (5.14) and Equation (5.15), we have

$$
\overline{\mathfrak{R}}(\rho\circ\mathcal{V}, p) \leq 2L_\rho \mathfrak{R}(\mathcal{V}, p) \leq 2L_\rho \overline{\mathfrak{R}}(\mathcal{V}, p).
$$

□

Proof of Proposition 5.9. The proof is the extension of [Mohri et al., 2018, Exercises 3.11] to the transductive and multi-layer setting. See also the proof of [Nitanda and Suzuki, 2018, Theorem 3]. First, we note that the multiplication $X \mapsto \tilde{P}^{(s)}X$ in $\mathcal{G}^{(s)}$ and the multiplication $X \mapsto XW^{(s-1)} \in \mathbb{R}^{C\times C}$ in $\mathcal{G}^{(s-1)}$ are commutative operations. Therefore, we have

$$
\mathcal{F}^{(t)}(X) := \{(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N) \mid f \in \mathcal{B}^{(t)}_{\text{base}}, \|W^{(s)}_{\cdot c}\|_1 \leq C^{(s)} \text{ for all } c \in [C] \text{ and } s = 2, \ldots, t\},
$$

where $\boldsymbol{z}_n := f(\boldsymbol{x}_n W^{(2)} \cdots W^{(t)})$ and $\boldsymbol{x_n} := (P^{(t)}X)_n \in \mathbb{R}^C$. Therefore, it is sufficient that we first prove the proposition by assuming $\tilde{P}^{(s)} = I_N$ for all $s = 2, \ldots, t$ and then replace X with $P^{(t)}X$.

We define $\mathcal{J}^{(s)} \subset \mathbb{R}^N$ be the set of possible values of any channel of the s-th representations and $\mathcal{H}^{(l)} \subset \mathbb{R}^N$ be the set of possible values of any output channel of the l-th layer of an MLP. More concretely, we define

$$
\begin{aligned}
\mathcal{J}^{(1)} &:= \{X_{\cdot c} \mid c \in [C]\}, \\
\mathcal{J}^{(s+1)} &:= \left\{ \sum_{c=1}^{C} \boldsymbol{z}_c w_c \;\middle|\; \boldsymbol{z}_c \in \mathcal{J}^{(s)}, \|w\|_1 \leq \tilde{C}^{(s+1)} \right\},
\end{aligned}
$$

for $s = 1, \ldots t-1$. Similarly, we define

$$
\begin{aligned}
\mathcal{H}^{(1)} &:= \mathcal{J}^{(t)}, \\
\tilde{\mathcal{H}}^{(l+1)} &:= \left\{ \sum_{c=1}^{C_{l+1}} \boldsymbol{z}_c w_c \;\middle|\; \boldsymbol{z}_c \in \mathcal{H}^{(l)}, \|w\|_1 \leq \tilde{B}^{(t)} \right\}, \\
\mathcal{H}^{(l+1)} &:= \sigma \circ \tilde{\mathcal{H}}^{(l+1)} = \{\sigma(\boldsymbol{z}) \mid \boldsymbol{z} \in \tilde{\mathcal{H}}^{(l+1)}\}.
\end{aligned}
$$

for $l = 1, \ldots, L$. By the definition of $\mathcal{F}^{(t)}$, we have $\{f(X) \mid f \in \mathcal{F}^{(t)}\} = \tilde{\mathcal{H}}^{(L+1)}$. On one hand, we can bound the Rademacher complexity of $\tilde{\mathcal{H}}^{(l)}$ as

$$
Q^{-1}\overline{\mathfrak{R}}(\tilde{\mathcal{H}}^{(l+1)}, p) = \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{\|w\|_1 \leq \tilde{B}^{(l)}, Z_{\cdot c} \in \mathcal{H}^{(l)}} \left| \sum_{n=1}^{N} \sigma_n \sum_{c=1}^{C_{l+1}} Z_{nc} w_c \right| \right]
$$

$$\begin{aligned}
&= \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{\|w\|_1 \le \tilde{B}^{(l)}, Z_{\cdot c} \in \mathcal{H}^{(l)}} \left|\sum_{c=1}^{C_{l+1}} w_c \sum_{n=1}^{N} \sigma_n Z_{nc}\right|\right] \\
&= \tilde{B}^{(t)} \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{Z \in \mathcal{H}^{(l)}} \left|\sum_{n=1}^{N} \sigma_n Z_n\right|\right] \\
&= \tilde{B}^{(t)} Q^{-1} \overline{\mathfrak{R}}(\mathcal{H}^{(l)}, p). \qquad (5.16)
\end{aligned}$$

On the other hand, since σ is 1-Lipschitz, by the contraction property (Proposition 5.10), we bound the Rademacher complexity of $\mathcal{H}^{(l+1)}$ as

$$\overline{\mathfrak{R}}(\mathcal{H}^{(l+1)}, p) \le 2\overline{\mathfrak{R}}(\tilde{\mathcal{H}}^{(l+1)}, p). \qquad (5.17)$$

By combining Equation (5.16) and Equation (5.17), we have the inductive relationship.

$$\overline{\mathfrak{R}}(\mathcal{H}^{(l+1)}, p) \le 2\tilde{B}^{(t)}\overline{\mathfrak{R}}(\mathcal{H}^{(l)}, p). \qquad (5.18)$$

Using the similar argument to $\mathcal{J}^{(s)}$'s, for $s \in 2, \ldots, t-1$, we have

$$\overline{\mathfrak{R}}(\mathcal{J}^{(s+1)}, p) \le \tilde{C}^{(s)}\overline{\mathfrak{R}}(\mathcal{J}^{(s)}, p). \qquad (5.19)$$

Let $P_c \in \mathbb{R}^C$ be the projection matrix onto the c-th coordinate. Then, for the base step, we can evaluate the Rademacher complexity of $\mathcal{J}^{(1)}$ as

$$\begin{aligned}
Q^{-1}\overline{\mathfrak{R}}(\mathcal{J}^{(1)}, p) &= \mathbb{E}_{\boldsymbol{\sigma}}\left[\max_{c \in [C]} \left|\sum_{n=1}^{N} \sigma_n X_{nc}\right|\right] \\
&= \mathbb{E}_{\boldsymbol{\sigma}}\left[\max_{c \in [C]} \left|\left(\sum_{n=1}^{N} \sigma_n X_n\right) P_c\right|\right] \\
&\le \mathbb{E}_{\boldsymbol{\sigma}}\left[\max_{c \in [C]} \|P_c\|_{\mathrm{op}} \left\|\sum_{n=1}^{N} \sigma_n X_n\right\|_2\right] \\
&\le \mathbb{E}_{\boldsymbol{\sigma}}\left\|\sum_{n=1}^{N} \sigma_n X_n\right\|_2 \\
&\le \sqrt{\mathbb{E}_{\boldsymbol{\sigma}} \sum_{c=1}^{C}\left(\sum_{n=1}^{N} \sigma_n X_{nc}\right)^2} \quad (\because \text{Jensen's inequality}) \\
&= \sqrt{\mathbb{E}_{\boldsymbol{\sigma}} \sum_{c=1}^{C} \sum_{n,m=1}^{N} \sigma_n \sigma_m X_{nc} X_{mc}} \\
&= \sqrt{\sum_{c=1}^{C} \sum_{n=1}^{N} 2p(X_{nc})^2} \qquad (5.20) \\
&= \sqrt{2p}\|X\|_{\mathrm{F}}. \qquad (5.21)
\end{aligned}$$

Here, we used in Equation (5.20) the equality

$$\mathbb{E}_{\boldsymbol{\sigma}} \sigma_m \sigma_n = 2p\delta_{mn}, \tag{5.22}$$

which is shown by the independence of transductive Rademacher variables. By using the inequalities we have proved so far, we obtain

$$\begin{aligned}
\overline{\mathfrak{R}}(\mathcal{F}^{(t)}, p) &= \overline{\mathfrak{R}}(\tilde{\mathcal{H}}^{(L+1)}, p) \\
&= 2\overline{\mathfrak{R}}(\mathcal{H}^{(L)}, p) \quad (\because \text{Equation (5.17) }) \\
&\leq 2(2\tilde{B}^{(t)})^{L-1}\overline{\mathfrak{R}}(\mathcal{H}^{(1)}, p) \quad (\because \text{Equation (5.18)}) \\
&\leq 2(2\tilde{B}^{(t)})^{L-1}\overline{\mathfrak{R}}(\tilde{\mathcal{J}}^{(t)}, p) \quad (\because \text{Definition of } \mathcal{H}^{(1)}) \\
&\leq 2(2\tilde{B}^{(t)})^{L-1}\left(\prod_{s=2}^{t} \tilde{C}^{(s)}\right)\overline{\mathfrak{R}}(\tilde{\mathcal{J}}^{(1)}, p) \quad (\because \text{Equation (5.19)}) \\
&\leq \sqrt{p}QD^{(t)}\|X\|_{\mathrm{F}}, \quad (\because \text{Equation (5.21)})
\end{aligned}$$

where we used $D^{(t)} = 2\sqrt{2}(2\tilde{B}^{(t)})^{L-1}\prod_{s=2}^{t}\tilde{C}^{(s)}$. Therefore, the proposition is true for $\tilde{P}^{(s)} = I_N$ for all $s = 2, \ldots, t$. As stated in the beginning of the proof, we should replace X with $P^{(t)}X$ in the general case. □

Proof of Proposition 5.3. By applying Proposition 5.9 with $p = p_0 = \frac{MU}{(M+U)^2}$ and using Equation (5.14), we have

$$\mathfrak{R}(\mathcal{F}^{(t)}) \leq \overline{\mathfrak{R}}(\mathcal{F}^{(t)}, p_0) \leq \sqrt{\frac{MU}{(M+U)^2}}QD^{(t)}\|P^{(t)}X\|_{\mathrm{F}} = \frac{D^{(t)}\|P^{(t)}X\|_{\mathrm{F}}}{\sqrt{MU}}.$$

□

5.A.5 Proof of Theorem 5.2

Proof. By Proposition 5.2, with probability $1 - \delta'$, we have

$$\mathcal{R}(\widehat{Y}) \leq \widehat{\mathcal{R}}(\widehat{Y}) + \sum_{t=1}^{T}\eta^{(t)}\mathfrak{R}(\mathcal{F}^{(t)}) + c_0 Q\sqrt{M \wedge U} + \sqrt{\frac{SQ}{2}\log\frac{1}{\delta'}}. \tag{5.23}$$

By Theorem 5.1, we have

$$\widehat{\mathcal{R}}(\widehat{Y}) \leq \frac{(1 + e^{\delta})\widehat{\mathcal{L}}(\widehat{Y}^{(1)})}{2M\Gamma_T}. \tag{5.24}$$

By Proposition5.3, we have

$$\mathfrak{R}(\mathcal{F}^{(t)}) \leq \frac{D^{(t)}\|P^{(t)}X\|_{\mathrm{F}}}{\sqrt{MU}} \tag{5.25}$$

By applying Equation (5.24) and Equation (5.25) to Equation (5.23) and substituting the learning rate $\eta^{(t)} = \frac{4}{\alpha_t}$, we obtained Equation (5.4). In particular, when $\Gamma_T = O(T^{\varepsilon})$, the first term of Equation (5.4) is $O(T^{-\varepsilon})$, which is asymptotically monotonically decreasing (assuming δ, $\widehat{Y}^{(1)}$, and M is independent of T). When $\alpha_t^{-1} B^{(t)} \|P^{(t)}\|_{\mathrm{op}}^t = O(\tilde{\varepsilon}^t)$, the second term of Equation (5.4) is bounded by

$$\begin{aligned}\frac{4}{\sqrt{MU}} \sum_{t=1}^{T} \frac{D^{(t)} \|P^{(t)} X\|_{\mathrm{F}}}{\alpha_t} &\leq \frac{4\sqrt{2}\|X\|_{\mathrm{F}}}{\sqrt{MU}} \sum_{t=1}^{T} \frac{B^{(t)} \|P^{(t)}\|_{\mathrm{op}}}{\alpha_t} \\ &\lesssim \frac{\|X\|_{\mathrm{F}}}{\sqrt{MU}} \frac{1}{1-\tilde{\varepsilon}}.\end{aligned}$$

The upper bound is independent of T (assuming that $\|X\|_{\mathrm{F}}$, M, and U are independent of T). □

5.A.6 Proof of Proposition 5.4

First, we recall the usual (i.e., inductive) version of the Rademacher complexity. We employ the following definition.

Definition 5.5 ((Inductive) Empirical Rademacher Complexity). *For $\mathcal{F}_{\mathrm{base}} \subset \{\mathcal{X} \to \widehat{\mathcal{Y}}\}$ and $Z = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N) \in \mathcal{X}^N$, we define the (inductive) empirical Rademacher complexity $\widehat{\mathfrak{R}}_{\mathrm{ind}}(\mathcal{F}_{\mathrm{base}})$ conditioned on Z by*

$$\widehat{\mathfrak{R}}_{\mathrm{ind}}(\mathcal{F}_{\mathrm{base}}; Z) := \frac{1}{N} \mathbb{E}_{\boldsymbol{\varepsilon}} \left[\sup_{f \in \mathcal{F}_{\mathrm{base}}} \sum_{n=1}^{N} \varepsilon_n f(\boldsymbol{z}_n) \right],$$

where $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_N)$ is the i.i.d. Rademacher variables defined by $\mathbb{P}(\varepsilon_i = 1) = \mathbb{P}(\varepsilon_i = -1) = 1/2$.

Proof of Proposition 5.4. Similarly to Proposition 5.9 it is sufficient that we first prove the proposition by assuming $\tilde{P}^{(s)} = I_N$ for all $s = 2, \ldots, t$ and then replace X with $P^{(t)} X$. By definition, the transductive Rademacher variable of parameter $p = 1/2$ equals to the (inductive) Rademacher variable. Therefore, we have

$$\begin{aligned}Q^{-1} \mathfrak{R}(\mathcal{F}^{(t)}, 1/2) &= \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{f \in \mathcal{F}^{(t)}} \sum_{n=1}^{N} \sigma_n f(X)_n \right] \\ &= \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{f_{\mathrm{base}} \in \mathcal{B}_{\mathrm{base}}^{(t)}, \|W^{(s)}\|_1 \leq \tilde{C}^{(s)}} \sum_{n=1}^{N} \sigma_n f_{\mathrm{base}}(X W^{(2)} \cdots W^{(t)}) \right] \\ &= N \widehat{\mathfrak{R}}_{\mathrm{ind}}(\mathcal{F}_{\mathrm{base}}^{(t)}; X). \end{aligned} \tag{5.26}$$

Since $p_0 := \frac{MU}{(M+U)^2} < 1/2$, by the monotonicity of the transductive Rademacher complexity (see El-Yaniv and Pechyony [2009, Lemma 1]), we have

$$\mathfrak{R}(\mathcal{F}^{(t)}) = \mathfrak{R}(\mathcal{F}^{(t)}, p_0) < \mathfrak{R}(\mathcal{F}^{(t)}, 1/2). \tag{5.27}$$

The proposition follows from Equation (5.26) and Equation (5.27) as follows

$$\mathfrak{R}(\mathcal{F}^{(t)}) < QN\widehat{\mathfrak{R}}_{\mathrm{ind}}(\mathcal{F}^{(t)}_{\mathrm{base}}; X) = \frac{(1+r)^2}{r}\widehat{\mathfrak{R}}_{\mathrm{ind}}(\mathcal{F}^{(t)}_{\mathrm{base}}; X)$$

□

5.A.7 Proof of Proposition 5.5

Proof. We denote $p_0 = \frac{MU}{(M+U)^2}$. Let $\sigma_1, \ldots, \sigma_N$ be the i.i.d. transductive Rademacher variable of parameter p_0. Since $\{-1, 0, 1\}^N \subset \mathcal{V}_g$, for any realization of $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_N)$, we have $\boldsymbol{\sigma} \in \mathcal{V}_g$. By the assumption, there exists $Z_{\boldsymbol{\sigma}} \in \mathcal{V}$ such that

$$\|Z_{\boldsymbol{\sigma}} - \alpha\boldsymbol{\sigma}\|_2 \leq \beta\|\boldsymbol{\sigma}\|_2.$$

Set $C_f := (2\alpha)^{-1}$ and $C_{\mathcal{L}} := \frac{\alpha^2-\beta^2}{2\alpha}$. Similarly to the proof of Theorem 5.1, we have

$$\begin{aligned}
&\|Z_{\boldsymbol{\sigma}} - \alpha\boldsymbol{\sigma}\|_2 \leq \beta\|\boldsymbol{\sigma}\|_2 \\
\iff &\|Z_{\boldsymbol{\sigma}}\|_2^2 - 2\alpha\langle Z_{\boldsymbol{\sigma}}, \boldsymbol{\sigma}\rangle + \alpha^2\|\boldsymbol{\sigma}\|_2^2 \leq \beta^2\|\boldsymbol{\sigma}\|_2^2 \\
\iff &C_f\|Z_{\boldsymbol{\sigma}}\|_2^2 + C_{\mathcal{L}}\|\boldsymbol{\sigma}\|_2^2 \leq \langle Z_{\boldsymbol{\sigma}}, \boldsymbol{\sigma}\rangle.
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
Q^{-1}\mathfrak{R}(\mathcal{V}) &= \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{Z\in\mathcal{F}}\langle\boldsymbol{\sigma}, Z\rangle\right] \\
&\geq \mathbb{E}_{\boldsymbol{\sigma}}\left[\langle\boldsymbol{\sigma}, Z_{\boldsymbol{\sigma}}\rangle\right] \\
&\geq \mathbb{E}_{\boldsymbol{\sigma}}\left[C_{\mathcal{L}}\|\boldsymbol{\sigma}\|_2^2 + C_f\|f_{\boldsymbol{\sigma}}\|_2^2\right] \\
&\geq \mathbb{E}_{\boldsymbol{\sigma}}\left[C_{\mathcal{L}}\|\boldsymbol{\sigma}\|_2^2\right] \\
&= C_{\mathcal{L}} \cdot 2Np_0.
\end{aligned}$$

In the last equality, we used Equation (5.22). Therefore, we have $\mathfrak{R}(\mathcal{F}) \geq 2C_{\mathcal{L}}Np_0Q = \frac{\alpha^2-\beta^2}{\alpha}$.

□

5.B Provable Satisfiability of Weak Learning Condition using Over-parameterized Models

In this section, we show that there exists a model that for any w.l.c. parameters α and β, we can find a weak learner which *probably* satisfies the w.l.c. using the gradient descent algorithm. To ensure the w.l.c., the set of transformation functions $\mathcal{B}$ must be sufficiently large so that it can approximate all possible values of the negative gradient $-\nabla\widehat{\mathcal{L}}$ can take. We can accomplish it by leveraging the universal approximation property of MLPs, similarly to graph isomorphism networks (GIN) [Xu et al., 2019], but for a different purpose. We adopt the recent studies that proved the global convergence of over-parameterized MLPs trained by a tractable algorithm (e.g., Arora et al. [2019], Du et al. [2019b]).

Let $R \in \mathbb{N}_+$. For the parameter $\Theta = (\theta_{ri}) \in \mathbb{R}^{R\times N}$, we consider an MLP with a single hidden layer $f_\Theta : \mathbb{R}^C \to \mathbb{R}$ defined by

$$f_\Theta(\boldsymbol{x}) := \frac{1}{\sqrt{R}} \sum_{r=1}^{R} a_r \sigma(\theta_{r\cdot}^\top \boldsymbol{x}),$$

where $a_r \in \{-1, 1\}$ and σ is the ReLU activation function: $\sigma(x) := x \vee 0$ (we apply ReLU in an element-wise manner for a vector input). We define the set of transformation functions $\mathcal{B} := \{(f_\Theta, \ldots, f_\Theta) \mid \Theta \in \mathbb{R}^{R\times N}\}$. At the t-th iteration, given a gradient $\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})$, we initialize the model with $a_r \overset{\text{i.i.d.}}{\sim} \mathrm{Unif}(\{-1, 1\})$ and $\theta_{ri} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I)$ independently and train it with the gradient descent to optimize Θ by minimizing the mean squared error between the output of the model and the properly normalized negative gradient. Using the result of [Du et al., 2019b], we obtain the following guarantee.

Proposition 5.11. *Suppose Algorithm 1 finds $g^{(s)} \in \mathcal{G}^{(s)}$ ($s \in [t]$) such that $X^{(t)} = g^{(t)} \circ \cdots \circ g^{(1)}(X) = \begin{bmatrix}\boldsymbol{x}_1 & \cdots & \boldsymbol{x}_N\end{bmatrix}^\top \in \mathbb{R}^{N\times C}$ ($\boldsymbol{x}_i \in \mathbb{R}^C$) satisfies the conditions that $\boldsymbol{x}_i \neq 0$ for all $i \in [N]$ and $\boldsymbol{x}_i \nparallel \boldsymbol{x}_j$ for all $i \neq j \in [N]$. Let $\delta, \alpha, \beta > 0$. Then, there exists $R = O(N^6\delta^{-3})$ such that for all $\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})$, with a probability of at least $1-\delta$, the gradient descent algorithm finds $b^{(t)} \in \mathcal{B}^{(t)}$ such that the t-th weak learner $f^{(t)}$ satisfies the $(\alpha, \beta, -\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}))$-w.l.c.*

Proof. We define $H^\infty \in \mathbb{R}^{N\times N}$ by $H^\infty_{ij} := \mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(0,I)}[\boldsymbol{x}_i^\top \boldsymbol{x}_j \mathbf{1}\{\boldsymbol{w}^\top \boldsymbol{x}_i \geq 0\}\mathbf{1}\{\boldsymbol{w}^\top \boldsymbol{x}_j \geq 0\}]$. Let λ_0 be the lowest eigenvalue of H^∞. Under the assumption, we know $\lambda_0 > 0$ by Theorem 3.1 of Du et al. [2019b]. We train the parameter $\Theta_{\cdot c\cdot}$ using the dataset $((\boldsymbol{x}_1, -\alpha[\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}]_1), \ldots, (\boldsymbol{x}_N, -\alpha[\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})]_N)$. We denote the parameter of the MLP at the k-th iteration of the gradient descent by $\Theta^{(k)}$. We denote the output of the model $f_{\Theta^{(k)}}$ by $\boldsymbol{u}^{(k)} := (f_{\Theta^{(k)}}(\boldsymbol{x}_1), \ldots, f_{\Theta^{(k)}}(\boldsymbol{x}_N))^\top$. By [Du et al., 2019b, Theorem 4.1], with probability $1-\delta$, we have

$$\|\boldsymbol{u}^{(k)} + \alpha\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})\|_2 \leq \left(1 - \frac{\eta\lambda_0}{2}\right)^k \|\boldsymbol{u}^{(k)} + \alpha\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})\|_2,$$

where $\eta = O\left(\frac{\lambda_0}{N^2}\right)$. Set

$$k := \log\left(\frac{\beta\|\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})\|_2}{\|\boldsymbol{u}^{(0)} + \alpha\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})\|_2 \vee 1}\right)\left(\log\left(1 - \frac{\eta\lambda_0}{2}\right)\right)^{-1}.$$

Then, with probability $1-\delta$, we have

$$\|\boldsymbol{u}^{(k)} + \alpha\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})\|_2 \leq \beta\|\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})\|_2,$$

which means $\boldsymbol{u}^{(k)}$ satisfies $(\alpha, \beta, -\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)}))$-w.l.c. □

Remark 5.1. *Du et al. [2019b] assumed that any feature vector $\boldsymbol{x}$ of the training data satisfies $\|\boldsymbol{x}\| = 1$. However, as commented in Du et al. [2019b], we can loosen this condition as follows: there exists $c_{low}, c_{high} > 0$ such that any feature vector $\boldsymbol{x}$ satisfies $c_{low} \leq \|\boldsymbol{x}\| \leq c_{high}$.*

Although this instantiation provably satisfies the w.l.c. with high probability, its model complexity is extremely large because it has as many as $O(N^6)$ parameters. As we saw in Section 5.8, such a large model complexity is inevitable as long as the gradient can take arbitrary values.

Table 5.3: Dataset specifications.

	#Node	#Edge	#Class (K)	Chance Rate
Cora	2708	5429	6	30.2%
CiteSeer	3312	4732	7	21.1%
PubMed	19717	44338	3	39.9%

5.C Experiment Settings

5.C.1 Experiments in Section 5.7.1 (Node Prediction Tasks)

Dataset

We used the Cora [McCallum et al., 2000, Sen et al., 2008], CiteSeer [Giles et al., 1998, Sen et al., 2008], and PubMed [Sen et al., 2008] datasets. Each dataset represents scientific papers as the nodes and citation relationships as the edges of a graph. For each paper, the genre of this paper is associated as a label. The task is to predict the genre of papers from word occurrences and the citation relationships. Table 5.3 shows the statistics of datasets. We obtained the preprocessed dataset from the code repository of Kipf and Welling [2017][5] and split each dataset into train, validation, and test datasets in the same way as experiments in Kipf and Welling [2017].

Model

As shown in Table 5.6 in Section 5.D.1, we have tested four base models: GB-GNN-Adj, GB-GNN-KTA, GB-GNN-II, and GB-GNN-SAMME.R. For each model, we consider three types of variants: (1) the base model, (2) the model with fine tuning, and (3) models with different layer sizes ($L = 0, 2, 3, 4$), We have shown the result of variants (1) and (2) of GB-GNN-Adj and GB-GNN-KTA in the main paper. See Section 5.D.1 for the results of other models.

Node Aggregation Functions For the aggregation functions $\mathcal{G}$, we used the matrix multiplication model with the augmented normalized adjacency matrix $\tilde{A}$ of the underlying graph $\mathcal{G}_{\tilde{A}}$ (for GB-GNN-Adj) and the KTA model $\mathcal{G}_{\mathrm{KTA}}$ (for GB-GNN-KTA). We also employed the input injection model with the augmented normalized adjacency matrix $\mathcal{G}_{\mathrm{II}}(\rho, \tilde{A})$ (for GB-GNN-II). For the KTA models, we used $\mathcal{G}_{\mathrm{KTA}} := \{g : X \mapsto wX + \sum_{k=0}^{N_{\mathrm{deg}}} w_k \tilde{A}^{2^k} X \mid w, w_k \in \mathbb{R}\}$. We treat weights w and w_k's in $\mathcal{G}_{\mathrm{KTA}}$ as learnable parameters and the mixing parameter ρ of $\mathcal{G}_{\mathrm{II}}(\rho, \tilde{A})$ as a hyperparameter.

Boosting Algorithms We used two boosting algorithms SAMME and SAMME.R. SAMME is the default boosting algorithm and is applied to GB-GNN-Adj, GB-GNN-KTA, and GB-GNN-II. We used SAMME.R in combination with the matrix multiplication model $\mathcal{G}_{\tilde{A}}$ only (for GB-GNN-SAMME.R).

[5] `https://github.com/tkipf/gcn` (Retrieved on December 2, 2020)

Transformation Functions For the transformation functions $\mathcal{B}$, we used MLPs with ReLU activation functions that have $L = 0, \ldots, 4$ hidden layers followed by the argmax operation. We showed results for the $L = 1$ model in the main paper. See Section 5.D.1 for other models.

The SAMME algorithm assumes that each weak learner outputs one of label categories, while SAMME.R assumes that the probability distribution over the set of categorical labels. Therefore, we added the argmax operation to the MLPs when we used SAMME and the softmax operation to the MLPs when SAMME.R,

We only imposed soft restrictions on MLPs in the models using regularization methods such as Dropout and weight decay. This is different from the MLP model defined in Section 5.5 in the main paper, which hard-thresholded the norms of weights and bias.

We treat weights in the MLP as trainable parameters and treat architectural parameters (e.g., unit size) other than the layer size L as hyperparameters (see Table 5.4 for the complete hyperparameters).

Training

We used the SAMME or SAMME.R algorithm to train the model. At the t-th iteration, we give $X^{(t-1)}$ and Y as a set of feature vectors and labels, respectively to the model. We picked B training sample points randomly and trained the transformation functions $\mathcal{B}^{(t)}$ using them. We used a gradient-based optimization algorithm to minimize the cross entropy between the prediction of the weak learner and the ground truth labels. We initialized the model (i.e., MLP) using the default initialization method implemented in PyTorch.

For the aggregation model $\mathcal{G}^{(t)}$, if it does not have a learnable parameter, that is, if $\mathcal{G}^{(t)}$ consists of a single function, we just applied the function to convert $X^{(t-1)}$ into $X^{(t)}$. For the KTA model, which has learnable parameters w and w_k's, we trained the model g using a gradient-based optimization to maximize the correlation between gram matrices created from transformed features $g(X^{(t-1)})$ and labels Y. The correlation is defined as follows:

$$\frac{\langle \mathcal{K}[g(X^{(t-1)})], \mathcal{K}[Y] \rangle}{\|\mathcal{K}[g(X^{(t-1)})]\|_{\mathrm{F}} \|\mathcal{K}[Y]\|_{\mathrm{F}}},$$

where $\mathcal{K}$ is the operator that takes the outer product of training sample points defined in Section 5.6. We initialized weights w and w_k's with 1.

After the training using boosting algorithm, we optionally trained the whole model as fine-tuning. When we used SAMME, we replaced the argmax operation in the transformation functions $\mathcal{B}$ with the softmax function along class labels to make the model differentiable. When we used SAMME.R, we did not change the same architecture in the training and fine tuning phases. We trained the whole model in an end-to-end manner using a gradient-based optimization algorithm to minimize the cross entropy between the prediction of the model and the ground truth label.

Evaluation

We split the dataset into training, validation, and test datasets. For each hyperparameter, we trained a model using the training dataset and evaluated it using the validation dataset. We defined the performance of a set of hyperparameters as the accuracy on the validation dataset at the iteration that

Table 5.4: Hyperparameters of node prediction experiments in Section 5.7.1. $X \sim \mathrm{LogUnif}[a, b]$ means the random variable $\log_{10} X$ obeys the uniform distribution over $[a, b]$. $(*)$ For KTA + Fine Tuning setting, we reduce the number of weak learners to 40 due to GPU memory constraints. $(**)$ Learning rate corresponds to α when Optimization algorithm is Adam Kingma and Ba [2015].

Category	Name	Value
Boosting	#Weak learners	$\{1, 2, \ldots, 100\ (40^{(*)})\}$
	Minibatch size B	$\{1, 2, \ldots, \|V_{\mathrm{train}}\|\}$
	Clipping value	$\mathrm{LogUnif}[-10, -5]$
Model	Epoch	$\{10, 20, \ldots, 100\}$
	Optimization algorithm	$\{\mathrm{SGD}, \mathrm{Adam}, \mathrm{RMSProp}\}$
	Learning rate$^{(**)}$	$\mathrm{LogUnif}[-5, -1]$
	Momentum	$\mathrm{LogUnif}[-10, -1]$
	Weight decay	$\mathrm{LogUnif}[-10, -1]$
	Unit size	$\{10, 11, \ldots, 200\}$
	Dropout	$\{\mathrm{ON(ratio{=}0.5)}, \mathrm{OFF}\}$
Input Injection	Mixing ratio ρ	$\mathrm{Unif}[0, 1]$
Kernel Target Alignment	Epoch	$\{5, 6, \ldots, 30\}$
	Optimization algorithm	$\{\mathrm{SGD}, \mathrm{Adam}, \mathrm{RMSProp}\}$
	Learning rate$^{(**)}$	$\mathrm{LogUnif}[-5, -1]$
	Degree N_{deg}	3
Fine Tuning	Epoch	$\{1, 2, \ldots, 100\}$
	Optimization algorithm	$\{\mathrm{SGD}, \mathrm{Adam}, \mathrm{RMSProp}\}$
	Learning rate$^{(**)}$	$\mathrm{LogUnif}[-5, -1]$
	Momentum	$\mathrm{LogUnif}[-10, -1]$
	Weight decay	$\mathrm{LogUnif}[-10, -1]$

maximizes the validation accuracy. If a model has a fine-tuning phase, we used the accuracy after the fine-tuning as the performance. We chose the set of hyperparameters that maximizes the performance using a hyperparameter optimization algorithm. We employed Tree-structured Parzen Estimator [Bergstra et al., 2011] and for hyperparameter optimization and the median stopping rule implemented in Optuna for pruning unpromising sets of hyperparameters. Table 5.4 shows the set of hyperparameters. We define the final performance of the model as the accuracy on the test dataset attained by the optimized set of hyperparameters.

For each pair of the dataset and the model, we ran the above evaluation ten times and computed the mean and standard deviation of the performance.

Implementation and Computational Resources

Experimental code is written in Python3. We used PyTorch [Paszke et al., 2019] and Ignite for the implementation and training of models, Optuna [Akiba et al., 2019] for the hyperparameter optimization, NetworkX [Hagberg et al., 2008] for preprocessing graph objects, and SciPy [Virtanen et al., 2020] for miscellaneous machine learning operations. We ran each experiment on a docker image (OS: Ubuntu18.04) built on a cluster. The image has two CPUs and single GPGPUs (NVIDIA Tesla V100).

5.C.2 Experiments in Section 5.7.2 (Link Prediction Tasks)

Dataset

We assume that the raw dataset consists of a graph (V, E) with $N = |V|$ nodes and a collection of C_0-dimensional node features $X \in \mathbb{R}^{N \times C_0}$. We refer to the element of E as a *positive* edge. We divide the set of positive edges E into training, validation and test samples and denote them by E^+_{train}, E^+_{val}, and E^+_{test}, respectively. We define $N_{\mathrm{t}} := |E^+_{\mathrm{t}}|$ $(\mathrm{t} \in \{\mathrm{train}, \mathrm{val}, \mathrm{test}\})$. Let $\overline{E} := (V \times V) \setminus E$ be the set of node pairs that do not have an edge. In a link prediction tasks, we are given a graph $(V, E^+_{\mathrm{train}})$ with a collection of node features X. The goal is to output $Y \in \{0, 1\}^{N \times N}$ where Y_{ij} indicates the existence or non-existence of an edge between the node pair i and j.

For evaluation, we made negative edge sets for validation and test in the same way as Kipf and Welling [2016] as follows. We sampled N_{val} edges uniformly randomly without replacement from $\overline{E}$ treated them as negative validation edges. We denote the set of sampled edges by E^-_{val}. Similarly, we sampled N_{test} edges uniformly randomly without replacement from $\overline{E} \setminus E^-_{\mathrm{val}}$ treated them as negative test edges. We denote the set of sampled edges by E^-_{test}.

We used three citation networks datasets: Cora, CiteSeer, and PubMed. See Table 5.3 for the dataset specifications. We used the same train/validation/test dataset partition as the one used in Kipf and Welling [2016][6]. Note that the dataset only contains positive edges and randomly generate negative edges at the time of training Therefore, negative edges can differ for each experiment runs. Also, the sampled negative edges can be different from experiments in Kipf and Welling [2016].

Model

The model consists of GB-GNN and a link predictor module. GB-GNN makes node representations and the link predictor module outputs the probability of the edge existence using a pair of node representations.

We used GB-GNN-adj with the transformation function $\mathcal{B}$ being a ReLU MLP with a single hidden layer as a GB-GNN model. Specifically, at the t-th iteration, we computed node representations $Z^{(t)} \in \mathbb{R}^{N \times C}$ as follows:

$$\begin{aligned} X^{(t)} &= \tilde{A}^{t-1} X, \\ Z^{(t)}_i &= \mathrm{MLP}^{(t)}_{\mathcal{B}}(X^{(t)}_i) \quad (i \in V). \end{aligned}$$

Here, $X \in \mathbb{R}^{N \times C_0}$ is the collection of input feature vectors, $\tilde{A}$ is the augmented normalized adjacency matrix, and $\mathrm{MLP}^{(t)}_{\mathcal{B}}$ is an MLP that serves as a transform function $b^{(t)}$ at the t-th iteration. $\mathrm{MLP}^{(t)}_{\mathcal{B}}$ has a single hidden layer with C hidden units and C output units.

[6]`https://github.com/tkipf/gae` (Retrieved on December 2, 2020)

We fed the representations $Z^{(t)}$ to the link predictor module to obtain edge probabilities. More concretely, for a node pair $i, j \in V$, the link predictor module computes the edge probability p_{ij} between i and j as follows:

$$\begin{aligned}
Z'_i &= \mathrm{sigmoid}(Z_i^{(t)}), \\
Z'_j &= \mathrm{sigmoid}(Z_j^{(t)}), \\
Z'_{ij} &= \mathrm{MLP}^{(t)}_{\mathrm{link}}(Z'_i \| Z'_j), \\
p_{ij} &= \mathrm{sigmoid}(Z'_{ij}).
\end{aligned}$$

Here, $\mathrm{sigmoid}$ is the sigmoid function and $\|$ is the vector concatenation operator, and $\mathrm{MLP}^{(t)}_{\mathrm{link}}$ is a ReLU MLP at the t-th iteration. $\mathrm{MLP}^{(t)}_{\mathrm{link}}$ has a single hidden layer with $2C$ hidden units and a single output unit.

Training

First, we sampled N_{train} node pairs from $(V \times V) \setminus E^{+}_{\mathrm{train}}$ with replacement and treated them as negative training edges. We denote the set of sampled edges by E^{-}_{train}. At the t-th iteration, we randomly sampled B_{w} data points from E^{+}_{train} and E^{-}_{train} without replacement, respectively. We trained the t-th weak learner in a end-to-end manner using E^{+}_{train} and E^{-}_{train}. That is, we train the t-th MLPs $\mathrm{MLP}^{(t)}_{\mathcal{B}}$ and $\mathrm{MLP}^{(t)}_{\mathrm{link}}$ simultaneously using a gradient-based optimization algorithm. The training objective is the sigmoid cross-entropy between the predicted edge probability and the label (i.e., existence or non-existence of edges).

We used SAMME.R as an ensemble algorithm.

Evaluation

The evaluation protocol was the same as the one used in the node prediction tasks in Section 5.7.1 except one point. Specifically, the objective of hyperparameter optimization is the area under the ROC curve (ROC-AUC) on the validation dataset, which was the accuracy on the validation dataset in Section 5.7.1. See Section 5.C.1 for the full evaluation protocol. Table 5.5 shows hyperparameters of the model and its training.

Implementation and Computational Resources

The implementation and computational resources are the same as the ones in Section 5.C.1.

5.D Additional Experiment Results

5.D.1 More Results for Model Variants

Table 5.6 shows the result of the prediction accuracies of models that use MLPs with various layer size L as transformation functions $\mathcal{B}^{(t)}$. It also shows the results for the input injection model (GB-GNN-II) and the SAMME.R model (GB-GNN-SAMME.R) we have introduced in Section 5.C.1.

Table 5.5: Hyperparameters of link prediction experiments in Section 5.7.2

Category	Name	Value
Boosting	#Weak learners T	$\{10, 11, \ldots, 100\}$
	Weak learner sample size B_{w}	$\{1, 2, \ldots, 1024\}$
	Minibatch size B	$\{1, 2, \ldots, B_w\}$
	Clipping value	$\mathrm{LogUnif}[-5, -2]$
Model	Epoch	$\{10, 11, \ldots, 1000\}$
	Optimization algorithm	$\{\mathrm{SGD}, \mathrm{Adam}, \mathrm{RMSProp}\}$
	Learning rate	$\mathrm{LogUnif}[-4, -1]$
	Momentum	$\mathrm{LogUnif}[-10, -1] - 10^{-10}$
	Weight decay	$\mathrm{LogUnif}[-10, -1] - 10^{-10}$
	Unit size C	$\{10, 11, \ldots, 1024\}$
	Dropout	$\{\mathrm{ON(ratio{=}0.5)}, \mathrm{OFF}\}$

Figure 5.3–5.5 show the experiment results for GB-GNN-adj with $L = 0, \ldots, 4$ layers using the Cora, CiteSeer, and PubMed datasets. Figure 5.3 shows the transition of the training loss. Figure 5.4 shows the transition of the test loss. Figure 5.5 shows the transition of the cosine values between the negative gradient $-\nabla\widehat{\mathcal{L}}(\widehat{Y}^{(t-1)})$ and the weak learner $f^{(t)}$ at the t-th iteration.

5.D.2 Performance Comparison with Existing GNN Models

Table 5.7 shows the accuracies of node prediction tasks on citation networks for various GNN models. We borrowed the results of the official repository of Deep Graph Library (GDL)[7] [Wang et al., 2019a], a package for deep learning on graphs.

[7]`https://github.com/dmlc/dgl` (Retrieved on December 2, 2020)

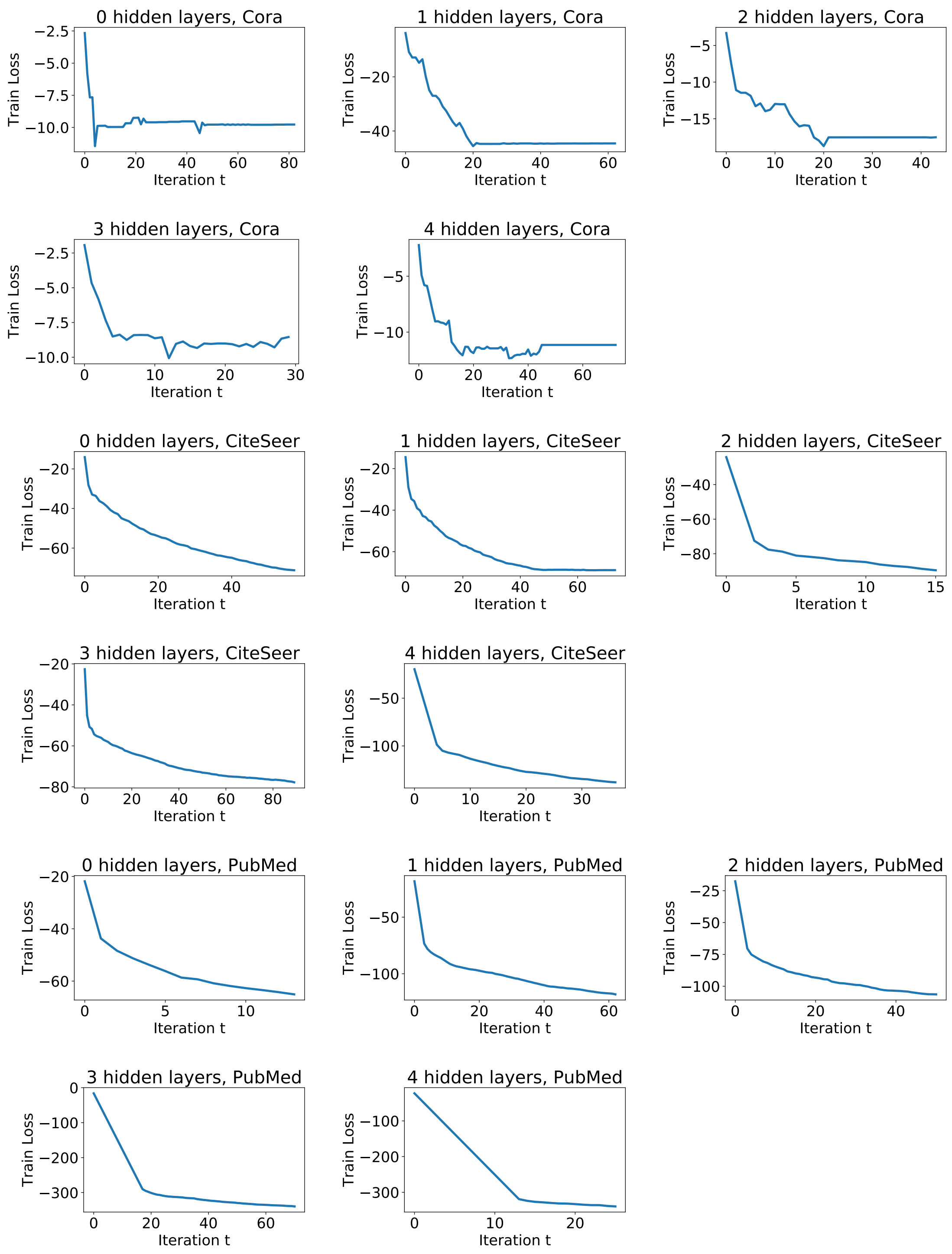


Figure 5.3: Train loss transition of GB-GNN-adj for the Cora (1st and 2nd rows), CiteSeer (3rd and 4th rows), and PubMed (5th and 6th rows) datasets.

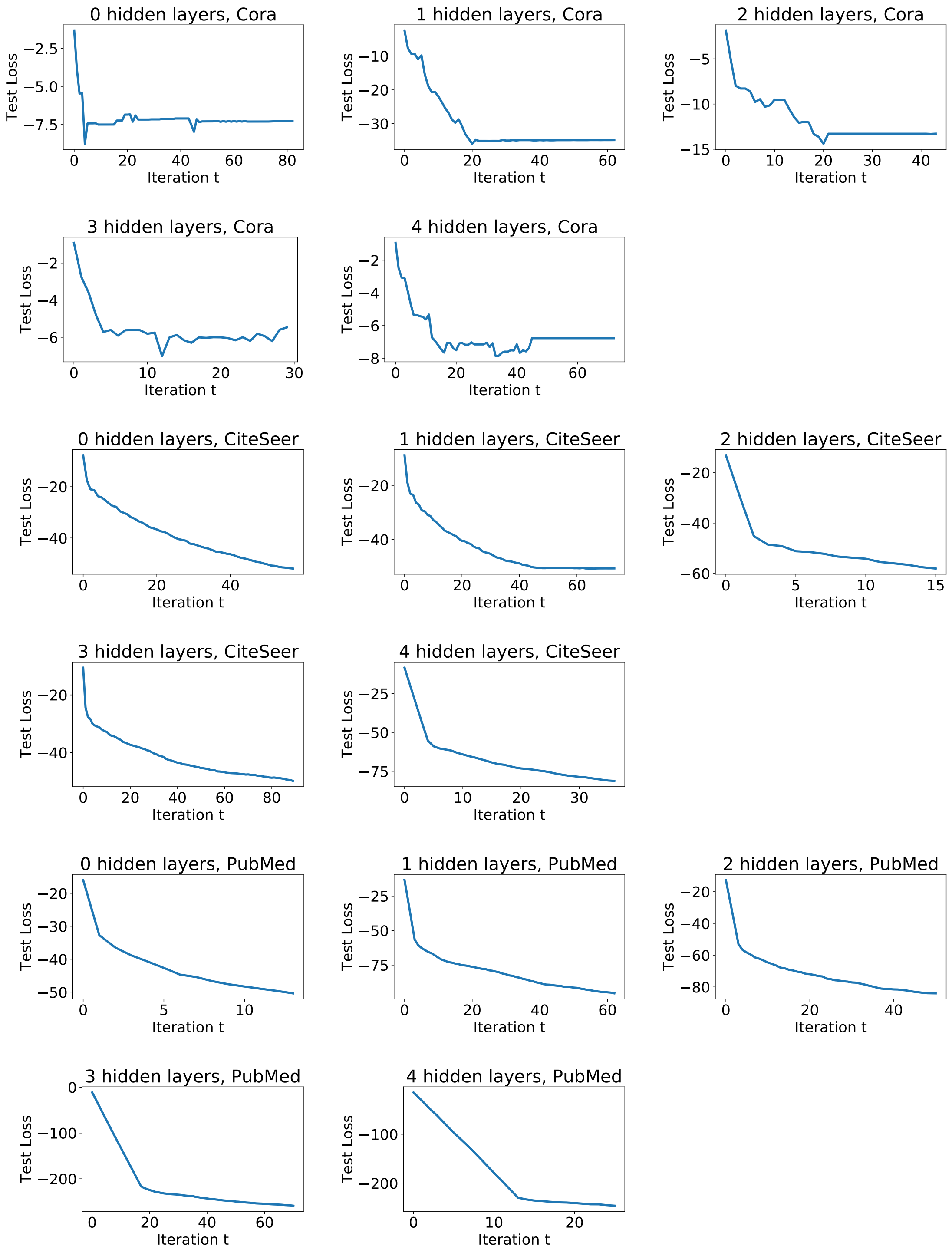


Figure 5.4: Test loss transition of GB-GNN-adj for the Cora (1st and 2nd rows), CiteSeer (3rd and 4th rows), and PubMed (5th and 6th rows) datasets.

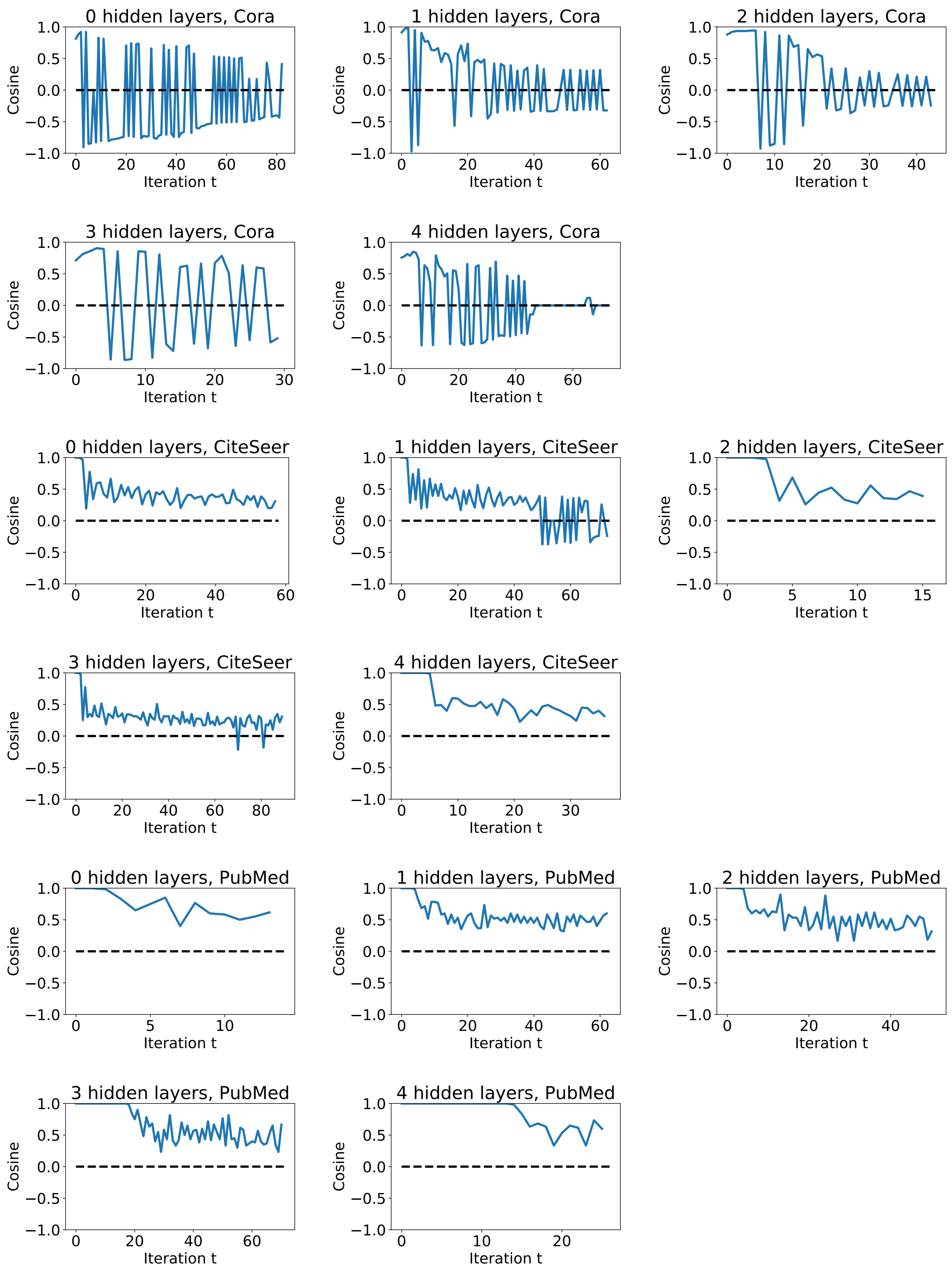


Figure 5.5: Direction of weak learners for GB-GNN-adj for the Cora (1st and 2nd rows), CiteSeer (3rd and 4th rows), and PubMed (5th and 6th rows) datasets.

Table 5.6: Accuracy of node classification tasks on citation networks. L denotes the number of hidden layers. Numbers are $(\text{mean}) \pm (\text{standard deviation})$ of ten runs. $(*)$ All runs failed due to GPU memory errors.

	L	Cora	CiteSeer	PubMed
GB-GNN-Adj	0	79.4 ± 1.9	70.3 ± 0.5	78.8 ± 0.7
	1	79.9 ± 0.8	70.5 ± 0.8	79.4 ± 0.2
	2	79.9 ± 1.3	68.5 ± 1.3	78.9 ± 0.6
	3	77.4 ± 0.8	64.4 ± 1.5	78.0 ± 0.5
	4	75.6 ± 2.6	60.7 ± 1.7	77.9 ± 0.6
GB-GNN-Adj. + Fine Tuning	1	80.4 ± 0.8	70.8 ± 0.8	79.0 ± 0.5
GB-GNN-KTA	0	80.0 ± 0.8	70.0 ± 1.8	79.4 ± 0.1
	1	80.9 ± 0.9	73.1 ± 1.1	79.1 ± 0.4
	2	79.8 ± 1.3	68.8 ± 1.1	79.1 ± 0.4
	3	78.5 ± 0.9	65.2 ± 1.5	78.4 ± 0.8
	4	76.0 ± 2.5	65.6 ± 1.7	78.0 ± 0.7
GB-GNN-KTA + Fine Tuning	1	82.3 ± 1.1	70.8 ± 1.0	N.A.$^{(*)}$
GB-GNN-II	0	79.2 ± 1.3	71.4 ± 0.3	79.3 ± 0.5
	1	79.8 ± 1.3	71.3 ± 0.5	79.4 ± 0.3
	2	79.9 ± 0.8	69.8 ± 1.1	79.3 ± 0.3
	3	78.7 ± 1.7	66.7 ± 1.9	79.2 ± 0.6
	4	75.4 ± 2.0	65.1 ± 2.5	78.6 ± 0.7
GB-GNN-II + Fine Tuning	1	80.8 ± 1.3	70.8 ± 0.9	79.2 ± 0.8
GB-GNN-SAMME.R	0	81.0 ± 0.8	70.4 ± 0.7	78.9 ± 0.3
	1	82.2 ± 1.2	71.6 ± 0.5	78.8 ± 0.3
	2	80.5 ± 0.8	67.4 ± 1.1	78.9 ± 0.3
	3	79.6 ± 1.1	64.4 ± 1.4	78.8 ± 0.5
	4	78.9 ± 1.8	64.6 ± 1.3	78.1 ± 0.6
GB-GNN-SAMME.R + Fine Tuning	1	82.1 ± 1.0	71.3 ± 0.8	79.4 ± 0.4

Table 5.7: Accuracy comparison with baseilne GNN models. Created from the official repository of DGL as of May 23rd, 2020. Adj.: Matrix multiplication model $\mathcal{G}_{\tilde{A}}$ by the normalized adjacency matrix $\tilde{A}$. KTA: Kernel target alignment model $\mathcal{G}_{\mathrm{KTA}}$. II: Input injection model $\mathcal{G}_{\mathrm{II}}$. FT: Fine Tuning. Paper: accuracies are cited from the paper in the Ref. column. DGL: accuracies are cited from the official implementation of DGL. $(*)$ Not available due to GPU memory errors. $(**)$ Not available from the DGL repository.

Model	Source	Framework	Cora	Citeseer	Pubmed
GB-GNN-Adj	–	PyTorch	79.9	70.5	79.4
-Adj + FT			80.4	70.7	79.0
-KTA			80.9	73.1	79.4
-KTA + FT			82.3	70.8	N.A.$^{(*)}$
-II			79.8	71.4	79.4
-II + FT			80.8	70.8	79.2
-SAMME.R			82.2	71.6	78.8
-SAMME.R + FT			82.1	71.3	79.4
GCN [Kipf and Welling, 2017]	Paper	–	81.5	70.3	79.0
	DGL	PyTorch	81.0	70.2	78.0
		TensorFlow	81.0	70.7	79.2
TAGCN [Du et al., 2017]	Paper	–	83.3	71.4	79.4
	DGL	PyTorch	81.2	71.5	79.4
		MXNet	82.0	70.2	79.8
GraphSAGE [Hamilton et al., 2017]	DGL	PyTorch	83.3	71.1	78.3
		MXNet	81.7	69.9	79.0
MoNet [Monti et al., 2017]	DGL	PyTorch	81.6	N.A.$^{(**)}$	76.3
		MXNet	81.4	N.A.$^{(**)}$	74.8
GAT [Veličković et al., 2018]	Paper	–	83.0	72.5	79.0
	DGL	PyTorch	84.0	70.9	78.6
		TensorFlow	84.2	70.9	78.5
SGC [Wu et al., 2019a]	Paper	–	83.0	72.5	79.0
	DGL	PyTorch	84.2	70.9	78.5
DGI [Veličković et al., 2019]	Paper	–	82.3	71.8	76.8
	DGL	PyTorch	81.6	69.4	76.1
		TensorFlow	81.6	70.2	77.2
APPNP [Klicpera et al., 2019]	Paper	–	85.0	75.7	79.7
	DGL	PyTorch	83.7	71.5	79.3
		MXNet	83.7	71.3	79.8

Conclusion

Summary and Follow-up Work

In this dissertation, we study *how structures in DL models affect their theoretical characteristics* to fill the gap between theory and practice of DL. Toward this goal, we posed these questions in the introduction.

Problem 1. *Why do skip connections promote predictive performance of CNNs?*

Problem 2. *Can non-linearity mitigate over-smoothing of GNNs caused by node aggregations?*

Problem 3. *What is the role of skip connections in GNNs? How do they affect the over-smoothing?*

First, we analyzed the approximation and estimation abilities of ResNet-type CNNs in Chapter 3 to answer Problem 1. We have shown that the approximation error of ResNet-type CNNs is at least as good as corresponding block-sparse FNNs. Using this correspondence, we derived estimation error bounds for ResNet-type CNNs. In particular, ResNet-type CNNs have the minimax optimality in non-parametric regression problems when the true function is in Hölder class without imposing L_0 constraints on CNN parameters, as opposed to FNNs in which no optimality result is known in this setting. Our analysis gave a theoretical explanation for the superiority of skip connections from the viewpoint of sparsity.

It was left as future work to explore when ResNet-type CNNs can achieve better approximation or estimation error rates than FNNs. As follow-up work, Kohler et al. [2020] proved that when the true function is a hierarchical max-pooling model, the estimation error rate of CNNs was input-dimension-free. They defined the *order* of the hierarchical max-pooling model and replaced the input dimension in the rate with the order (see the paper for the precise definition). Although Kohler et al. [2020] did not prove the lower bound for FNNs, their result suggested that CNNs can achieve better rates than FNNs for sufficiently high-dimensional data.

We have shown that skip connections remove unrealistic sparse constraints from FNNs while preserving the minimax optimality. This result posed one question: is the minimax optimality without sparse constraints specific to ResNet-type CNNs? After the study of Chapter 3, Shen et al. [2019] proved the minimax optimality of dense FNNs for the β-Hölder functions ($\beta \in (0, 1]$) and Lu et al. [2020] for β-times continuous differential functions (i.e., $\beta \in \mathbb{N}_+$). That is, general FNNs or CNNs have the same properties in common in these settings. These results do not contradict our results because they did not deny the minimax optimality of dense ResNet-type CNNs.

Next, we considered Problem 2, that is, the expressive power of GNNs, particularly loss of expressive power caused by the over-smoothing problem in Chapter 4. We interpreted the forward propagation of a GNN as a discrete-time dynamics and defined the over-smoothing as the convergence to an invariant space that is "information-less" for node prediction tasks. We derived the exponential convergence of a ReLU GCN to the invariant space whose rate is determined by an underlying graph's spectra. Since this rate was the same as the one for linear GCNs, our result implied that the ReLU non-linearity does not contribute to mitigating the over-smoothing problem. Although our study is the first theoretical explanation of over-smoothing for non-linear GNNs, our analysis is restricted to ReLU GCNs.

A limitation of our study was that the applicable GNN model was ReLU GCNs without bias terms. After the study of Chapter 4, several studies have extended our results to other GNN models. Cai and Wang [2020] proved the over-smoothing of GCNs with the Leaky-ReLU activation functions. They used the same "information-less" invariant space as ours. However, differently from our analysis, they employed the Dirichlet energy as a criterion of over-smoothing. Another work is Huang et al. [2020]. They extended our analysis to more general GNNs, including ReLU GCNs with bias terms, ReLU GCNs with skip connections, and APPNP (Section 2.6.1).

Our study related the expressive power of GNNs with the spectral distribution of underlying graphs. Several studies used our analysis tools to justify their proposed methods theoretically. For example, DropEdge [Rong et al., 2020] randomly pruned edges in underlying graphs to mitigate over-smoothing. From our theory's viewpoint, DropEdge tweaked the spectral distribution of underlying graphs and relaxed the condition on weight scales under which GNNs were destined to over-smoothing.

Finally, we study Problem 3 in Chapter 5. To understand the role of skip connection in GNNs, we derived the optimization and generalization bounds of multi-scale GNNs. Our analysis's key was to interpret a multi-scale GNN as an ensemble of single-path GNNs and apply boosting theory. The derived test error bounds are similar to ones for AdaBoost. Our result suggests that when we can sufficiently train subnetworks at each scale, the multi-scale structure induced by skip connections has guaranteed predictive power to solve the node prediction task at hand, even if the over-smoothing problem can happen. The problem of our analysis is that parameters in the weak learning assumptions are not directly computable. We need further consideration for the direct relationship between experimentally observable quantities and the optimization and generalization bounds. Another problem is that our theory is not directly applicable to end-to-end learning models. Whether the theory is specific to GNNs trained with boosting algorithms or universal to multi-scale GNNs is left for future research.

In parallel to our study, Zhu et al. [2020] recently explained the effectiveness of multi-scale structure from the viewpoint of *homophily*. Homophily is a property of graph-structured data where neighboring nodes are likely to have the same label. They showed that many standard GNN models such as GCN and GAT did not perform well under a low homophily regime. Based on this observation, they proposed H2GCN, a new multi-scale GNN model. They showed that adding multi-scale structures made GNN models work better in high and low homophily regimes.

Future Perspective

Continuous Model One approach for analyzing DL models is to lift them to their continuous counterparts. For example, we can think of a ResNet-type CNN as the Euler scheme discretization of a function obeying the corresponding continuous-time ODE, a continuous *temporal* limit of ResNet. There are two other types continuous limits: one is *spatial* continuity that takes the infinite limit of input dimensions and the other one is *integral representation* that takes the limit of an infinite number of intermediate layer units, including the analysis known as the *mean-field regime* [Nitanda and Suzuki, 2017, Chizat and Bach, 2018, Mei et al., 2018].

These approaches benefit from translating complex problems in a narrow space into easier problems in a large space. For example, in the mean-field regime analysis, we can transform a non-convex optimization problem in a finite-dimensional space into a convex optimization problem in an infinite-dimensional space. In addition, these approaches enable us to bring various mathematical tools such as functional analysis, (partial) differential equation theory, and optimal control theory.

Several studies have investigated the continuous limit of ResNet-type CNNs and GNNs, which were our primary interest. Neural ODE [Chen et al., 2018c] is a practical model that can be thought of as a ResNet-type model. CGCN [Xhonneux et al., 2020] explained in Section 2.6.1 have is a continuous GNN continuous in the temporal direction. Regarding the spatial continuity, node aggregation operations of GNNs can be intuitively thought of as a discretization of the heat equation. Another approach of spatial continuity is to use Graphon, a way of taking to the continuous limit of graphs (see, e.g., Lovász [2012]). This approach is useful for analyzing graph classification tasks and model transferability [Ruiz et al., 2020, Keriven et al., 2020], that is, the model's generalization ability to graphs with different sizes.

We expect that the idealization of a model by continuity reveals the model's essential nature and thus clarifies the impact of model structures on theoretical and empirical behaviors.

Unified Theory for Model Structures We have seen that structures in DL models significantly impact their theoretical properties, not only the expressive power but also the optimization and generalization abilities. We believe this study takes a step further in understanding the role of structures in DL models and provides a principled methodology for searching for new structures.

Nevertheless, we have analyzed model structures individually. For example, graph convolution operations were proposed to extend CNN convolution to arbitrary graphs, on the ground that both CNNs and GNNs should capture inductive biases that neighbor nodes are highly correlated. However, it is known that GNNs have phenomena that are not observed in CNNs. The over-smoothing problem we addressed in Chapter 4 is one such examples. Therefore, we adopted different approaches to analyzing skipped connections of CNNs and GNNs in Chapters 3 and 5, respectively.

Considering the wide variety of structures, not restricted to three examples we dealt with in this study, the development of unified theories that encompass various models gives further understanding of the role of structures in DL models.

Bibliography

S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. V. Steeg, and A. Galstyan. MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 21–29. PMLR, 2019a.

S. Abu-El-Haija, B. Perozzi, A. Kapoor, and J. Lee. N-GCN: Multi-scale graph convolutionfor semi-supervised node classification. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2019b.

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 2623–2631, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3330701.

B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nature biotechnology*, 33(8):831–838, 2015.

D. Angluin and L. G. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. *Journal of Computer and system Sciences*, 18(2):155–193, 1979.

S. Arora, R. Ge, B. Neyshabur, and Y. Zhang. Stronger generalization bounds for deep nets via a compression approach. volume 80 of *Proceedings of Machine Learning Research*, pages 254–263. PMLR, 2018.

S. Arora, S. Du, W. Hu, Z. Li, and R. Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 322–332. PMLR, 2019.

J. Atwood and D. Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems 29*, pages 1993–2001. Curran Associates, Inc., 2016.

A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

A. R. Barron. Approximation and estimation bounds for artificial neural networks. *Machine learning*, 14(1):115–133, 1994.

P. L. Bartlett, O. Bousquet, and S. Mendelson. Local rademacher complexities. *The Annals of Statistics*, 33(4):1497–1537, 2005.

P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

L. Bégin, P. Germain, F. Laviolette, and J.-F. Roy. Pac-bayesian theory for transductive learning. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 105–113. PMLR, 2014.

M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *International Conference on Computational Learning Theory*, pages 624–638. Springer, 2004.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32): 15849–15854, 2019.

J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Inc., 2011.

H. Bölcskei, P. Grohs, G. Kutyniok, and P. Petersen. Optimal approximation with sparsely connected deep neural networks. *SIAM Journal on Mathematics of Data Science*, 1(1):8–45, 2019.

L. Breiman. Arcing classifier (with discussion and a rejoinder by the author). *The Annals of Statistics*, 26(3):801–849, 1998.

M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.

D. Busbridge, D. Sherburn, P. Cavallo, and N. Y. Hammerla. Relational graph attention networks. *arXiv preprint arXiv:1904.05811*, 2019.

J. Busch, J. Pi, and T. Seidl. PushNet: Efficient and adaptive neural message passing. *arXiv preprint arXiv:2003.02228*, 2020.

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

J. Chen, T. Ma, and C. Xiao. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*, 2018a.

L. Chen, J. Li, J. Peng, T. Xie, Z. Cao, K. Xu, X. He, and Z. Zheng. A survey of adversarial learning on graphs. *arXiv preprint arXiv:2003.05730*, 2020a.

M. Chen, J. Pennington, and S. Schoenholz. Dynamical isometry and a mean field theory of RNNs: Gating enables signal propagation in recurrent neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 873–882. PMLR, 2018b.

M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li. Simple and deep graph convolutional networks. In *Proceedings of Machine Learning and Systems 2020*, pages 3730–3740. 2020b.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems 31*, pages 6571–6583. Curran Associates, Inc., 2018c.

T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

L. Chizat and F. Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Advances in Neural Information Processing Systems 31*, pages 3036–3046. Curran Associates, Inc., 2018.

E. Choi, Z. Xu, Y. Li, M. Dusenberry, G. Flores, E. Xue, and A. Dai. Learning the graphical structure of electronic health records with graph convolutional transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 606–613, 2020.

F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

A. Choromanska, Y. LeCun, and G. B. Arous. Open problem: The landscape of the loss surfaces of multilayer networks. In P. Grünwald, E. Hazan, and S. Kale, editors, *Proceedings of The 28th Conference on Learning Theory*, volume 40 of *Proceedings of Machine Learning Research*, pages 1756–1760. PMLR, 2015.

F. Chung and L. Lu. Connected components in random graphs with given expected degree sequences. *Annals of combinatorics*, 6(2):125–145, 2002.

F. Chung and M. Radcliffe. On the spectra of general random graphs. *the electronic journal of combinatorics*, 18(1):215, 2011.

F. Chung, L. Lu, and V. Vu. The spectra of random graphs with given expected degrees. *Internet Mathematics*, 1(3):257–275, 2004.

F. R. Chung and F. C. Graham. *Spectral graph theory*. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Soc., 1997.

T. S. Cohen, M. Geiger, J. Köhler, and M. Welling. Spherical CNNs. In *International Conference on Learning Representations*, 2018.

A. Coja-Oghlan. On the laplacian eigenvalues of $G_{n,p}$. *Combinatorics, Probability and Computing*, 16(6):923–946, 2007.

C. Cortes, M. Mohri, D. Pechyony, and A. Rastogi. Stability of transductive regression algorithms. In *Proceedings of the 25th international conference on Machine learning (ICML)*, pages 176–183, 2008.

C. Cortes, M. Mohri, and A. Rostamizadeh. Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13(28):795–828, 2012.

N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola. On kernel-target alignment. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 367–373. MIT Press, 2002.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song. Adversarial attack on graph structured data. volume 80 of *Proceedings of Machine Learning Research*, pages 1115–1124. PMLR, 2018a.

H. Dai, Y. Tian, B. Dai, S. Skiena, and L. Song. Syntax-directed variational autoencoder for structured data. In *International Conference on Learning Representations*, 2018b.

N. De Cao and T. Kipf. MolGAN: An implicit generative model for small molecular graphs. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29*, pages 3844–3852. Curran Associates, Inc., 2016.

J. Du, S. Zhang, G. Wu, J. M. Moura, and S. Kar. Topology adaptive graph convolutional networks. *arXiv preprint arXiv:1710.10370*, 2017.

S. S. Du, K. Hou, R. R. Salakhutdinov, B. Poczos, R. Wang, and K. Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Advances in Neural Information Processing Systems 32*, pages 5723–5733. Curran Associates, Inc., 2019a.

S. S. Du, X. Zhai, B. Poczos, and A. Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations (ICLR)*, 2019b.

R. M. Dudley. The sizes of compact subsets of hilbert space and continuity of gaussian processes. *Journal of Functional Analysis*, 1(3):290–330, 1967.

D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems 28*, pages 2224–2232. Curran Associates, Inc., 2015.

R. El-Yaniv and D. Pechyony. Stable transductive learning. In *International Conference on Computational Learning Theory*, pages 35–49. Springer, 2006.

R. El-Yaniv and D. Pechyony. Transductive rademacher complexity and its applications. *Journal of Artificial Intelligence Research*, 35:193–234, 2009.

P. Erdös and A. Rényi. On random graphs I. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

F. Fan, D. Wang, and G. Wang. Universal approximation by a slim network with sparse shortcut connections. *arXiv preprint arXiv:1811.09003*, 2018.

M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Y. Freund. Boosting a weak learning algorithm by majority. *Information and computation*, 121 (2):256–285, 1995.

Y. Freund and R. E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

J. Friedman, T. Hastie, R. Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The Annals of Statistics*, 28(2): 337–407, 2000.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

K. Fukushima and S. Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and Cooperation in Neural Nets*, pages 267–285, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg. ISBN 978-3-642-46466-9.

H. Gao and S. Ji. Graph u-nets. volume 97 of *Proceedings of Machine Learning Research*, pages 2083–2092. PMLR, 2019.

V. K. Garg, S. Jegelka, and T. Jaakkola. Generalization and representational limits of graph neural networks. *arXiv preprint arXiv:2002.06157*, 2020.

E. N. Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.

C. L. Giles, K. D. Bollacker, and S. Lawrence. Citeseer: an automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pages 89–98. ACM, 1998.

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017.

E. Giné and R. Nickl. *Mathematical foundations of infinite-dimensional statistical models*, volume 40. Cambridge University Press, 2015.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. volume 9 of *Proceedings of Machine Learning Research*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

A. Grover, A. Zweig, and S. Ermon. Graphite: Iterative generative modeling of graphs. volume 97 of *Proceedings of Machine Learning Research*, pages 2434–2444. PMLR, 2019.

A. Grubb and D. Bagnell. Generalized boosting algorithms for convex optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1209–1216, 2011.

G. L. Guimaraes, B. Sanchez-Lengeling, C. Outeiral, P. L. C. Farias, and A. Aspuru-Guzik. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017.

A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In G. Varoquaux, T. Vaught, and J. Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

T. Hagerup and C. Rüb. A guided tour of Chernoff bounds. *Information processing letters*, 33(6): 305–308, 1990.

W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30*, pages 1024–1034. Curran Associates, Inc., 2017.

W. L. Hamilton. *Graph Representation Learning*, volume 14. Morgan and Claypool, 2020.

D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

D. Han, J. Kim, and J. Kim. Deep pyramidal residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

T. Hastie, S. Rosset, J. Zhu, and H. Zou. Multi-class AdaBoost. *Statistics and its Interface*, 2(3): 349–360, 2009.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

S. Honda, H. Akita, K. Ishiguro, T. Nakanishi, and K. Oono. Graph residual flow for molecular graph generation. *arXiv preprint arXiv:1909.13521*, 2019.

K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4 (2):251–257, 1991.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu. Squeeze-and-excitation networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(8):2011–2023, 2020a. ISSN 0162-8828. doi: 10.1109/TPAMI.2019.2913372.

W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020b.

F. Huang, J. Ash, J. Langford, and R. Schapire. Learning deep ResNet blocks sequentially using boosting theory. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 2058–2067. PMLR, 2018.

G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2261–2269. IEEE, 2017.

W. Huang, Y. Rong, T. Xu, F. Sun, and J. Huang. Tackling over-smoothing for general graph convolutional networks. *arXiv preprint arXiv:2008.09864*, 2020.

M. Imaizumi and K. Fukumizu. Deep neural networks learn non-smooth functions effectively. In *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 869–878. PMLR, 2019.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456. PMLR, 2015.

A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Label propagation for deep semi-supervised learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5070–5079, 2019.

K. Ishiguro, S.-i. Maeda, and M. Koyama. Graph warp module: an auxiliary module for boosting the power of graph neural networks in molecular graph analysis. *arXiv preprint arXiv:1902.01020*, 2019.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems 31*, pages 8571–8580. Curran Associates, Inc., 2018.

W. Jin, R. Barzilay, and T. Jaakkola. Junction tree variational autoencoder for molecular graph generation. volume 80 of *Proceedings of Machine Learning Research*, pages 2323–2332. PMLR, 2018.

W. Jin, Y. Li, H. Xu, Y. Wang, and J. Tang. Adversarial attacks and defenses on graphs: A review and empirical study. *arXiv preprint arXiv:2003.00653*, 2020.

T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 290–297, 2003.

C. Johnson. Stabilization of linear dynamical systems with respect to arbitrary linear subspaces. *Journal of Mathematical Analysis and Applications*, 44(1):175–186, 1973.

J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

J. Johnson, A. Gupta, and L. Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

P. C. Kainen, V. Kůrková, and M. Sanguineti. Approximating multivariable functions by feedforward neural nets. In *Handbook on Neural Information Processing*, pages 143–181. Springer, 2013.

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc., 2017.

N. Keriven, A. Bietti, and S. Vaiter. Convergence and stability of graph convolutional networks on large random graphs. *arXiv preprint arXiv:2006.01868*, 2020.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

T. N. Kipf and M. Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

J. Klicpera, A. Bojchevski, and S. Günnemann. Combining neural networks with personalized pagerank for classification on graphs. In *International Conference on Learning Representations (ICLR)*, 2019.

J. M. Klusowski and A. R. Barron. Approximation by combinations of ReLU and squared ReLU ridge functions with ℓ_1 and ℓ_0 controls. *IEEE Transactions on Information Theory*, 64(12): 7649–7656, 2018.

M. Kohler, A. Krzyzak, and B. Walter. On the rate of convergence of image classifiers based on convolutional neural networks. *arXiv preprint arXiv:2003.01526*, 2020.

V. Koltchinskii. Local rademacher complexities and oracle inequalities in risk minimization. *The Annals of Statistics*, 34(6):2593–2656, 2006.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017. ISSN 0001-0782. doi: 10.1145/3065386. URL `https://doi.org/10.1145/3065386`.

M. J. Kusner, B. Paige, and J. M. Hernández-Lobato. Grammar variational autoencoder. volume 70 of *Proceedings of Machine Learning Research*, pages 1945–1954. PMLR, 2017.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

M. Ledoux and M. Talagrand. *Probability in Banach Spaces: isoperimetry and processes*. Springer Science & Business Media, 2013.

H. Lee, R. Ge, T. Ma, A. Risteski, and S. Arora. On the ability of neural nets to express distributions. In *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 1271–1296. PMLR, 2017.

G. Li, M. Muller, A. Thabet, and B. Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9267–9276, 2019.

G. Li, C. Xiong, A. Thabet, and B. Ghanem. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020a.

H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6389–6399. Curran Associates, Inc., 2018a.

Q. Li, Z. Han, and X.-M. Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018b.

Y. Li, R. Zemel, and M. Brockschmidt. Gated graph sequence neural networks. In *International Conference on Learning Representations*, 2016.

Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018c.

Y. Li, W. Jin, H. Xu, and J. Tang. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020b.

R. Liao, Y. Li, Y. Song, S. Wang, W. Hamilton, D. K. Duvenaud, R. Urtasun, and R. Zemel. Efficient graph generation with graph recurrent attention networks. In *Advances in Neural Information Processing Systems 32*, pages 4255–4265. Curran Associates, Inc., 2019a.

R. Liao, Z. Zhao, R. Urtasun, and R. Zemel. LanczosNet: Multi-scale deep graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2019b.

H. Lin and S. Jegelka. ResNet with one-neuron hidden layers is a universal approximator. In *Advances in Neural Information Processing Systems 31*, pages 6169–6178. Curran Associates, Inc., 2018.

Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 2077–2085. Association for Computing Machinery, 2018.

L. Lovász. *Large networks and graph limits*, volume 60. American Mathematical Soc., 2012.

J. Lu, Z. Shen, H. Yang, and S. Zhang. Deep network approximation for smooth functions. *arXiv preprint arXiv:2001.03040*, 2020.

Y. Lu, A. Zhong, Q. Li, and B. Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3276–3285. PMLR, 2018.

Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The expressive power of neural networks: a view from the width. In *Advances in Neural Information Processing Systems 30*, pages 6231–6239. Curran Associates, Inc., 2017.

S. Luan, M. Zhao, X.-W. Chang, and D. Precup. Break the ceiling: Stronger multi-scale deep graph convolutional networks. In *Advances in Neural Information Processing Systems 32*, pages 10943–10953. Curran Associates, Inc., 2019.

Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang. Graph convolutional networks with eigenpooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 723–731. ACM, 2019. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3330982.

A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

K. Madhawa, K. Ishiguro, K. Nakago, and M. Abe. Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems 12*, pages 512–518. MIT Press, 2000.

A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

S. Mei and A. Montanari. The generalization error of random features regression: Precise asymptotics and double descent curve. *arXiv preprint arXiv:1908.05355*, 2019.

S. Mei, A. Montanari, and P.-M. Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018. doi: 10.1073/pnas.1806579115.

S. Mendelson. Improving the sample complexity using global data. *IEEE transactions on Information Theory*, 48(7):1977–1991, 2002.

C. D. Meyer. *Matrix analysis and applied linear algebra*, volume 71. Siam, 2000.

H. N. Mhaskar. Approximation properties of a multilayered feedforward artificial neural network. *Advances in Computational Mathematics*, 1(1):61–80, 1993.

G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT Press, 2018.

F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5115–5124, 2017.

V. Nagarajan and Z. Kolter. Deterministic PAC-bayesian generalization bounds for deep networks via generalizing noise-resilience. In *International Conference on Learning Representations*, 2019.

H. Nguyen, S. Maeda, and K. Oono. Semi-supervised learning of hierarchical representations of molecules using neural message passing. *arXiv preprint arXiv:1711.10168*, 2017.

A. Nitanda and T. Suzuki. Stochastic particle gradient descent for infinite ensembles. *arXiv preprint arXiv:1712.05438*, 2017.

A. Nitanda and T. Suzuki. Functional gradient boosting based on residual network perception. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 3819–3828. PMLR, 2018.

A. Nitanda and T. Suzuki. Functional gradient boosting for learning residual-like networks with statistical guarantees. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2981–2991. PMLR, 2020.

J. R. Norris. *Markov chains*. Number 2 in Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge university press, 1998.

H. NT and T. Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

K. Oono and T. Suzuki. Approximation and non-parametric estimation of ResNet-type convolutional neural networks. volume 97 of *Proceedings of Machine Learning Research*, pages 4922–4931. PMLR, 2019.

K. Oono and T. Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations (ICLR)*, 2020a.

K. Oono and T. Suzuki. Optimization and generalization analysis of transduction through gradient boosting and application to multi-scale graph neural networks. *Advances in Neural Information Processing Systems*, 33, 2020b.

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, November 1999.

S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

D. Pechyony and R. El-Yaniv. *Theory and Practice of Transductive Learning*. PhD thesis, Computer Science Department, Technion, 2009.

D. Perekrestenko, P. Grohs, D. Elbrächter, and H. Bölcskei. The universal approximation power of finite-width deep ReLU networks. *arXiv preprint arXiv:1806.01528*, 2018.

B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710. Association for Computing Machinery, 2014. ISBN 9781450329569. doi: 10.1145/2623330.2623732.

P. Petersen and F. Voigtlaender. Equivalence of approximation by convolutional neural networks and fully-connected networks. *arXiv preprint arXiv:1809.00973*, 2018a.

P. Petersen and F. Voigtlaender. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Networks*, 108:296–330, 2018b.

A. Pinkus. Density in approximation theory. *Surveys in Approximation Theory*, 1:1–45, 2005.

L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. CatBoost: Unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems 31*, pages 6638–6648. Curran Associates, Inc., 2018.

Y. Rong, W. Huang, T. Xu, and J. Huang. DropEdge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations (ICLR)*, 2020.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

L. Ruiz, L. F. Chamon, and A. Ribeiro. Graphon neural networks and the transferability of graph neural networks. *arXiv preprint arXiv:2006.03548*, 2020.

R. Salakhutdinov and G. Hinton. Deep boltzmann machines. volume 5 of *Proceedings of Machine Learning Research*, pages 448–455, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 2009. PMLR.

B. Sanchez-Lengeling, C. Outeiral, G. L. Guimaraes, and A. Aspuru-Guzik. Optimizing distributions over molecular space. an objective-reinforced generative adversarial network for inverse-design chemistry (organic), Aug 2017.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

F. Scarselli, A. C. Tsoi, and M. Hagenbuchner. The Vapnik–Chervonenkis dimension of graph and recursive neural networks. *Neural Networks*, 108:248–259, 2018.

R. E. Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999.

R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5):1651–1686, 1998.

M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

J. Schmidt-Hieber. Nonparametric regression using deep neural networks with relu activation function. *Annals of Statistics*, 48(4):1875–1897, 2020.

P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

W. Shang, K. Sohn, D. Almeida, and H. Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2217–2225. PMLR, 2016.

Z. Shen, H. Yang, and S. Zhang. Deep network approximation characterized by number of neurons. *arXiv preprint arXiv:1906.05497*, 2019.

M. Simonovsky and N. Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422. Springer, 2018.

S. Sonoda and N. Murata. Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, 43(2):233–268, 2017.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15: 1929–1958, 2014.

R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

K. Sun, Z. Lin, and Z. Zhu. AdaGCN: Adaboosting graph convolutional networks into deep models. *arXiv preprint arXiv:1908.05081*, 2019.

L. Sun, Y. Dou, C. Yang, J. Wang, P. S. Yu, and B. Li. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528*, 2018.

T. Suzuki. Fast generalization error bound of deep learning from a kernel perspective. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1397–1406. PMLR, 2018.

T. Suzuki. Adaptivity of deep ReLU network for learning in Besov and mixed smooth Besov spaces: optimal rate and curse of dimensionality. In *International Conference on Learning Representations*, 2019.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 4278–4284. AAAI Press, 2017.

M. Telgarsky. Benefits of depth in neural networks. In *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 1517–1539. PMLR, 23–26 Jun 2016.

T. Teshima, I. Ishikawa, K. Tojo, K. Oono, M. Ikeda, and M. Sugiyama. Coupling-based invertible neural networks are universal diffeomorphism approximators. *Advances in Neural Information Processing Systems*, 33, 2020.

S. Tokui, K. Oono, S. Hido, and J. Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 2015.

S. Tokui, R. Okuta, T. Akiba, Y. Niitani, T. Ogawa, S. Saito, S. Suzuki, K. Uenishi, B. Vogel, and H. Yamazaki Vincent. Chainer: A deep learning framework for accelerating the research cycle. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2002–2011. ACM, 2019.

I. Tolstikhin, G. Blanchard, and M. Kloft. Localized complexities for transductive learning. In *Proceedings of The 27th Conference on Learning Theory*, volume 35 of *Proceedings of Machine Learning Research*, pages 857–884. PMLR, 2014.

I. Tolstikhin, N. Zhivotovskiy, and G. Blanchard. Permutational rademacher complexity. In *International Conference on Algorithmic Learning Theory*, pages 209–223. Springer, 2015.

A. B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer Publishing Company, Incorporated, 1st edition, 2008. ISBN 0387790519, 9780387790510.

V. Vapnik. *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics)*. Springer-Verlag, 1982.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

A. Veit, M. J. Wilber, and S. Belongie. Residual networks behave like ensembles of relatively shallow networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 550–558. Curran Associates, Inc., 2016.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm. Deep graph infomax. In *International Conference on Learning Representations (ICLR)*, 2019.

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

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17:261–272, 2020.

M. J. Wainwright. *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2019. doi: 10.1017/9781108627771.

M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, Z. Huang, Q. Guo, H. Zhang, H. Lin, J. Zhao, J. Li, A. J. Smola, and Z. Zhang. Deep Graph Library: Towards efficient and scalable deep learning on graphs. *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019a.

X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, WWW '19, pages 2022–2032. Association for Computing Machinery, 2019b. ISBN 9781450366748. doi: 10.1145/3308558.3313562.

C. Wei and T. Ma. Data-dependent sample complexity of deep neural networks via lipschitz augmentation. In *Advances in Neural Information Processing Systems 32*, pages 9725–9736. Curran Associates, Inc., 2019.

B. Weisfeiler and L. A.A. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.

D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger. Simplifying graph convolutional networks. volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR, 2019a.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019b.

L.-P. Xhonneux, M. Qu, and J. Tang. Continuous graph neural networks. In *Proceedings of Machine Learning and Systems 2020*, pages 7258–7267. 2020.

S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

H. Xu, Y. Li, W. Jin, and J. Tang. Adversarial attacks and defenses: Frontiers, advances and practice. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, pages 3541–3542. Association for Computing Machinery, 2020a. ISBN 9781450379984.

H. Xu, Y. Ma, H.-C. Liu, D. Deb, H. Liu, J.-L. Tang, and A. K. Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178, Apr 2020b.

K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5453–5462. PMLR, 2018.

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

J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh. Graph R-CNN for scene graph generation. In *Proceedings of the European Conference on Computer Vision*, pages 670–685, 2018.

M. Yao and T. Jiliang. *Deep Learning on Graphs*. Cambridge University Press, 2020.

D. Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94: 103–114, 2017.

D. Yarotsky. Universal approximations of invariant maps by neural networks. *arXiv preprint arXiv:1804.10306*, 2018.

Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems 31*, pages 4800–4810. Curran Associates, Inc., 2018.

J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec. GraphRNN: Generating realistic graphs with deep auto-regressive models. volume 80 of *Proceedings of Machine Learning Research*, pages 5708–5717. PMLR, 2018.

S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim. Graph transformer networks. In *Advances in Neural Information Processing Systems 32*, pages 11983–11993. Curran Associates, Inc., 2019.

W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977.

A. Zaeemzadeh, N. Rahnavard, and M. Shah. Norm-preservation: Why residual networks can become extremely deep? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

C. Zang and F. Wang. Moflow: An invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, pages 617–626, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3403104.

C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 793–803. Association for Computing Machinery, 2019.

J. Zhang. Gresnet: Graph residuals for reviving deep graph neural nets from suspended animation. *arXiv preprint arXiv:1909.05729*, 2019.

J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D. Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 339–349, 2018.

Z. Zhang, P. Cui, and W. Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020. doi: 10.1109/TKDE.2020.2981333.

L. Zhao and L. Akoglu. Pairnorm: Tackling oversmoothing in {gnn}s. In *International Conference on Learning Representations*, 2020.

D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, pages 321–328. MIT Press, 2004.

D.-X. Zhou. Universality of deep convolutional neural networks. *arXiv preprint arXiv:1805.10769*, 2018.

J. Zhou and O. G. Troyanskaya. Predicting effects of noncoding variants with deep learning-based sequence model. *Nature methods*, 12(10):931, 2015.

P. Zhou and J. Feng. Understanding generalization and optimization performance of deep CNNs. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5960–5969. PMLR, 2018.

J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33, 2020.

M. Zitnik, M. Agrawal, and J. Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):457–466, 2018.

D. Zügner, A. Akbarnejad, and S. Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 2847–2856. Association for Computing Machinery, 2018. ISBN 9781450355520.