Learning Graph Neural Networks with Approximate Gradient Descent

Qunwei Li,[1] Shaofeng Zou, [2] Wenliang Zhong [1]
[1] Ant Group, Hangzhou, China
[2] University at Buffalo, the State University of New York
qunwei.qw@antgroup.com, szou3@buffalo.edu, yice.zwl@antgroup.com

Abstract

The first provably efficient algorithm for learning graph neural networks (GNNs) with one hidden layer for node information convolution is provided in this paper. Two types of GNNs are investigated, depending on whether labels are attached to nodes or graphs. A comprehensive framework for designing and analyzing convergence of GNN training algorithms is developed. The algorithm proposed is applicable to a wide range of activation functions including ReLU, Leaky ReLU, Sigmod, Softplus and Swish. It is shown that the proposed algorithm guarantees a linear convergence rate to the underlying true parameters of GNNs. For both types of GNNs, sample complexity in terms of the number of nodes or the number of graphs is characterized. The impact of feature dimension and GNN structure on the convergence rate is also theoretically characterized. Numerical experiments are further provided to validate our theoretical analysis.

Introduction

Recent success of deep neural network (DNN) has dramatically boosted its application in various application domains, e.g., computer vision (Krizhevsky, Sutskever, and Hinton 2012), natural language processing (Young et al. 2018) and strategic reasoning with reinforcement learning (Sutton and Barto 2018; Silver et al. 2017). Although DNN usually has millions of parameters and a highly non-convex objective function, simple optimization techniques, e.g., stochastic gradient descent and its variants, can efficiently train such a complicated structure successfully. However, there are many applications, where data is represented in the form of graphs. For example, in social networks, a graph-based learning system can exploit the interconnection between the users to make highly accurate inferences. DNN, when applied to graph based structural data, has to face with some new challenges, and cannot be directly applied. To address the new challenges, graph neural network (GNN) has been proposed recently and has been demonstrated to have massive expressive power on graphs. GNNs have been shown to be successful in many application domains, e.g., social networks, knowledge graph, recommender systems, molecular fingerprint, and protein interface networks (Wu et al. 2019; Xu et al. 2018; Zhou et al. 2018).

Although being successful in practical applications, theoretical understanding of GNNs is still lacking in the literature. There is a strong need to theoretically understand why GNNs work well on practical graph-structured data, and to further design provably efficient learning algorithms for GNN.

However, even for commonly used DNN architectures, it is in general challenging to develop theoretical understanding of the learning dynamics and to learn simple neural networks (Goel, Klivans, and Meka 2018). Specifically, if we assume that the data is generated according to a teacher neural network, the goal is then to recover the true parameters of the teacher neural network. The major challenge lies in the highly non-convex nature of DNN's objective function. For example, in the agnostic setting with bounded distributions and square loss, learning a single ReLU with unit norm hidden weight vectors is notoriously difficult (Goel et al. 2017). It was proved by Brutzkus and Globerson (Gautier, Nguyen, and Hein 2016) that the problem of distribution-free recover of the unknown weight vector for learning one hidden layer convolutional neural network (CNN) is NP-hard, even if non-overlapping patch structure is assumed. At this point, a major open problem is still to understand mildest conditions and design of provably efficient algorithms that lead to learnability of neural networks in polynomial time.

In this paper, we focus on the parameter recovery problem for learning GNNs where the training data is generated according to a teacher GNN. We design provably efficient algorithms to recover the true parameters of the teacher GNN. Our focus is to develop a comprehensive framework for designing and analyzing convergence of GNN training algorithms.

Main Contributions

Our first contribution is the design and convergence analysis of an approximate gradient descent algorithm for training GNNs. The algorithm is based on the idea of inexact optimization and approximate training (Schmidt, Roux, and Bach 2011; Li et al. 2017; Cao and Gu 2019), and the major advantage is that it reduces computational complexity while guaranteeing convergence and learnability at the same time. We prove that the proposed algorithm recovers the underlying true parameters of the teacher network with a linear convergence rate up to statistical precision. The assumptions are mild, and can be easily satisfied in practice. Specifically, our analysis is applicable to a wide range of activation functions (see Assumptions 1 and 2), e.g., ReLU, Leaky ReLU, Sigmod, Softplus and Swish. The analysis only requires that the activation functions to be monotonic increasing, and Lipschitz, and does not depend on the specific gradient computation involved in the activation function (Brutzkus and Globerson 2017; Du, Lee, and Tian 2017; Safran and Shamir 2018).

Our second contribution is the introduction and the extension of the technique of approximate calculation in gradients for the algorithm in order to analyze GNNs. A similar idea was first proposed in (Cao and Gu 2019) to analyze the learnability of CNNs with non-overlapping convolution patches. We highlight that the non-overlapping convolution process is very different from the nature that the feature convolution at nodes in GNNs is intrinsically overlapping, as nodes may share common neighbors. The analysis framework in (Cao and Gu 2019) cannot be directly applied in GNN analysis. We extend the scope of the methodology and propose provably efficient algorithms to learn and analyze GNNs.

Our third contribution is that we investigate the empirical version of the problem where the estimation of parameters is based on n independent samples. We provide uniform convergence results of the proposed algorithm with respect to the sample complexity. We also provide the parameter training dynamics with the proposed algorithm, and show that the training is provably stable.

To the best of the authors' knowledge, these theoretical results are the first sharp analyses of statistical efficiency of GNNs. For ease of presentation, we hereby informally present the main theorem in the paper as follows. We refer readers to Theorem 2 for the precise statements.

Theorem 1 (Main Theorem (Informal)). *GNN is stably learnable with the proposed algorithms in linear time.*

Related Work

There is a recent surge of interest in theoretically understanding properties of DNNs, e.g., hardness of learning (Goel et al. 2017; Song et al. 2017), landscape of neural networks (Kawaguchi 2016; Choromanska et al. 2015; Hardt and Ma 2016; Haeffele and Vidal 2015; Freeman and Estrach 2019; Safran and Shamir 2016; Zhou and Feng 2017; Nguyen and Hein 2017a,b; Ge, Lee, and Ma 2017; Safran and Shamir 2018; Du and Lee 2018), training dynamics using gradient descent approaches (Tian 2017; Zhong et al. 2017; Li and Yuan 2017; Du et al. 2018), and design of provable learning algorithms (Goel and Klivans 2017a,b; Zhang et al. 2015). For non-convex optimization problems that satisfy strict saddle property, it was shown in (Du, Lee, and Tian 2017) and (Jin et al. 2017) that (stochastic) gradient descent converges in polynomial time. The landscape of neural networks were then extensively studied (Soltanolkotabi 2017; Kawaguchi 2016; Choromanska et al. 2015; Hardt and Ma 2016; Haeffele and Vidal 2015; Freeman and Estrach 2019; Safran and Shamir 2016; Zhou and Feng 2017; Nguyen and Hein 2017a,b; Ge, Lee, and Ma 2017; Safran and Shamir 2018; Du and Lee 2018). Specifically, algorithms were designed for specific neural network architectures, and their learning of a neural network in polynomial time and sample complexity were further characterized (Goel et al. 2017; Zhang, Lee, and Jordan 2016; Zhang et al. 2015; Sedghi and Anandkumar 2014; Janzamin, Sedghi, and Anandkumar 2015; Gautier, Nguyen, and Hein 2016; Goel and Klivans 2017b; Du, Lee, and Tian 2017).

However, to the best of our knowledge, there has been no related attempt to understand the training dynamics and to design provably efficient learning algorithms for GNNs. In recent advances (Du, Lee, and Tian 2017; Du et al. 2017; Goel, Klivans, and Meka 2018; Brutzkus and Globerson 2017; Zhong, Song, and Dhillon 2017), the condition for (stochastic) gradient descent or its variants to recover the underlying true parameter under the teacher-student model in polynomial time were analyzed for CNNs. Specifically, for spherical Gaussian distribution and non-overlapping patch structures, gradient descent approach can recover the underlying true parameter in polynomial time for CNNs (Brutzkus and Globerson 2017; Du et al. 2017). In (Cao and Gu 2019), the information theoretic limits and the computational complexity of learning a CNN were developed.

Note that the GNN shares a similar convolutional structure as the CNN (LeCun, Bengio et al. 1995), and is therefore closely related to the CNN. Specifically, an image can be viewed as a graph grid where adjacent pixels are connected by edges. Similar to a 2D convolution for an image, graph convolution can also be performed on an image by taking an weighted average of information from adjacent pixels. Here, each pixel in an image can be viewed as a node in a graph, and its neighbors are ordered. The size of neighbors are then determined by the convolution filer size. However, different from image data, the neighbors of a node in a graph for GNNs are unordered, and vary in sizes. Moreover, information convolution in graph is by nature overlapping. Therefore, the analysis for CNNs is intrinsically different from and cannot be directly carried over to the analysis for GNNs. We will elaborate this in this paper.

Notations

Let $\|\cdot\|_2$ denote the Euclidean norm of a finite-dimensional vector or a matrix. For a positive semidefinite matrix X, we use $\sigma_m(X)$ as its nonzero smallest eigenvalue and $\sigma_M(X)$ as its nonzero largest eigenvalue in sequel. Throughout this paper, we use capital letters to denote matrices, lower case letters to denote vectors, and lower case Greek letters to denote scalars. We denote $x \wedge y \triangleq \min\{x, y\}$.

The Graph Neural Network

In this section, we formalize the GNN model with one layer of node information convolution.

Node-level GNN (NGNN) with One Graph

We first investigate one graph with n nodes, and node i has a feature $\mathbf{H}_i \in \mathbb{R}^d$ with dimension d. We define $\mathbf{H} \in \mathbb{R}^{n \times d}$ as the node feature matrix of the graph. To learn a GNN, a node first collects all the features from its neighboring nodes, and updates its own feature with a shared local transition function among these nodes. This process is termed as graph convolution and can be expressed as

$$\hat{\mathbf{H}}_i = \sigma\left(\frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\mathbf{H}_j\mathbf{W}\right), \tag{1}$$

where σ is the activation function, $\mathcal{N}_i$ is the set containing node i and all its neighboring nodes, and $\mathbf{W} \in \mathbb{R}^{d\times d_{out}}$ represents the local transition function which usually is a fully connected network taking in the averaged node feature and outputs the updated feature. After this process, the node feature gets updated with a change in dimension from d to d_{out}. The above process of graph convolution can be written in a global point of view as

$$\hat{\mathbf{H}} = \sigma(\mathbf{D}^{-1}\mathbf{A}\mathbf{H}\mathbf{W}), \tag{2}$$

where $\mathbf{D} \in \mathbb{R}^{n\times n}$ is the degree matrix of the graph and $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the corresponding adjacency matrix. Here, $\mathbf{D}^{-1}\mathbf{A}\mathbf{H}$ is the operation that one node updates its feature by taking the average of its own features and features of its neighbors.

After the graph convolution, each node has its own feature updated, by incorporating local structural information of the graph and features from its neighboring nodes. Then, the updated node feature is passed into a fully connected layer $\mathbf{v} \in \mathbb{R}^{d_{out}}$. Thus, the output of the entire graph neural network for node-level tasks is

$$\hat{\mathbf{y}} = \sigma(\mathbf{D}^{-1}\mathbf{A}\mathbf{H}\mathbf{W})\mathbf{v} \in \mathbb{R}^n. \tag{3}$$

As we have *one* graph, suppose the graph has a node-level label vector $\mathbf{y} \in \mathbb{R}^n$, and we have n ground truth data pairs $\{\mathbf{H}_i, \mathbf{y}_i\}_{i=1}^n$. We assume that the node feature matrix $\mathbf{H} \in \mathbb{R}^{n\times d}$ is generated independently from the standard Gaussian distribution, and the corresponding output $\mathbf{y} \in \mathbb{R}^n$ is generated from the teacher network with true parameters $\mathbf{W}^*$ and $\mathbf{v}^*$ as follows

$$\mathbf{y} = \sigma(\mathbf{D}^{-1}\mathbf{A}\mathbf{H}\mathbf{W}_*)\mathbf{v}_* + \epsilon. \tag{4}$$

Here, $\{\epsilon_i\}_{i=1}^n$ are independent sub-Gaussian white noises with ψ_2 norm ν, which is a broader class including Gaussian noise but far less restrictive. Without loss of generality, we assume that $\|\mathbf{W}_*\|_2 = 1$ in this paper.

Graph-level GNN (GGNN) with Multiple Graphs

We then investigate with n graphs, and the j-th graph has n_j nodes with the node feature matrix $\mathbf{H}_j \in \mathbb{R}^{n_j\times d}$. Similar to the convolution for the case of one graph for NGNN, the j-th graph updates its node features by the following convolution process

$$\hat{\mathbf{H}}_j = \sigma(\mathbf{D}_j^{-1}\mathbf{A}_j\mathbf{H}_j\mathbf{W}), \tag{5}$$

where σ is the activation function, $\mathbf{D}_j \in \mathbb{R}^{n_j\times n_j}$ is the degree matrix of j-th graph, and $\mathbf{A}_j \in \mathbb{R}^{n_j\times n_j}$ is the corresponding adjacency matrix. Here, $\mathbf{D}_j^{-1}\mathbf{A}_j\mathbf{H}_j$ is the operation that one node in the j-th graph updates its feature by taking the average of its own features and the features of its neighbors. $\mathbf{W} \in \mathbb{R}^{d\times d_{out}}$ represents the local transition function which is shared among different nodes across different graphs.

For GGNN, graph j with n_j node features has to aggregate the node features such that the graph has a unique graph-level representation. In the following, we discuss several aggregation methods which are most widely used in the literature.

Particular node feature as graph embedding. This method picks a specific node feature to represent the global graph embedding. Define $\mathbf{g}_j \in R^d$ as the embedding for the j-th graph, which in this case is expressed as

$$\mathbf{g}_j^T = \mathbf{a}_j\sigma(\mathbf{D}_j^{-1}\mathbf{A}_j\mathbf{H}_j\mathbf{W}), \tag{6}$$

where $\mathbf{a}_j \in R^{1\times n_j}$ is a row vector with "1" as its i-th element and otherwise "0". Consequently, the i-th node feature is picked to represent the j-th graph for follow-up graph-level tasks.

Attention-based Aggregation. In this approach, the graph embedding is a weighted sum of the node features and the weight is usually termed as attention that is learned from the corresponding node feature. The attention-based weighted feature aggregation for the j-th graph can be expressed as

$$\mathbf{g}_j^T = \mathbf{a}_j\sigma(\mathbf{D}_j^{-1}\mathbf{A}_j\mathbf{H}_j\mathbf{W}), \tag{7}$$

where $\mathbf{a}_j \in R_+^{1\times n_j}$ is an attention row vector for all the nodes, with non-negative elements summing up to 1.

Averaging. Averaging is a special case of the attention based aggregation where the attention is identical for all the node features. Thus, we can write

$$\mathbf{g}_j^T = \frac{1}{n_j}\mathbf{1}_{n_j}\sigma(\mathbf{D}_j^{-1}\mathbf{A}_j\mathbf{H}_j\mathbf{W}), \tag{8}$$

where $\mathbf{1}_{n_j}$ is a n_j dimensional row vector of all ones.

After the graph convolution and the node feature aggregation, each graph has its own embedding, incorporating its local structural information and all its node features. Then, the graph embedding is passed into a fully connected layer $\mathbf{v} \in \mathbb{R}^d$. Thus, the output of the j-th graph for graph-level tasks is

$$\hat{y}_j = \mathbf{a}_j\sigma(\mathbf{D}_j^{-1}\mathbf{A}_j\mathbf{H}_j\mathbf{W})\mathbf{v} \in \mathbb{R}. \tag{9}$$

As we have *multiple* graphs, suppose each graph has a graph-level label $y_j \in \mathbb{R}$, and we have n ground truth data pairs $\{\mathbf{H}_j, y_j\}_{j=1}^n$. We assume that each node feature matrix $\mathbf{H}_j \in \mathbb{R}^{n_j\times d}$ is generated independently from the standard Gaussian distribution, and the corresponding output $y_j \in \mathbb{R}$ is generated from the teacher network with true parameters $\mathbf{W}^*$ and $\mathbf{v}^*$ as follows,

$$y_j = \mathbf{a}_j\sigma(\mathbf{D}_j^{-1}\mathbf{A}_j\mathbf{H}_j\mathbf{W}_*)\mathbf{v}_* + \epsilon_j. \tag{10}$$

Here, $\{\epsilon_j\}_{j=1}^n$ are independent sub-Gaussian white noises with ψ_2 norm ν. Through out this paper, we assume that $\|\mathbf{W}_*\|_2 = 1$. For simplicity, we assume that the learning process for $\mathbf{W}$ and $\mathbf{v}$ is disjoint from that for $\mathbf{a}_j$, or $\mathbf{a}_j$ is fixed and does not need training.

Assumptions

In this paper, we make the following assumptions, which can be easily satisfied with commonly used activation functions in practice, e.g., ReLU, Leaky ReLU, Sigmod and Softplus.

Assumption 1. *σ is a non-trivial increasing function, and is 1-Lipschitz continuous, i.e.,* $|\sigma(x_1) - \sigma(x_2)| \leq |x_1 - x_2|, \forall x_1, x_2 \in \mathbb{R}$.

Assumption 2. $\|\sigma(x)\|_2 \leq L_\sigma \|x\|_2, \forall x \in \mathbb{R}^n$.

Algorithm and Main Results

We study the sample complexity and training dynamics of learning GNNs with one convolution layer. We construct novel algorithms for training GNNs, and further prove that, with high probability, the proposed algorithms with proper initialization and training step size grant a linear convergence to the ground-truth parameters up to statistical precision. Our results apply to general non-trivial, monotonic and Lipschitz continuous activation functions including ReLU, Leaky ReLU, Sigmod, Softplus and Swish.

Algorithm Design

We present the algorithm in Algorithm 1 for NGNN and GGNN with two auxiliary matrices defined as

$$\Xi_N = \mathbb{E}\left[\mathbf{H}^\top \mathbf{A}^\top (\mathbf{D}^{-1})^\top \sigma(\mathbf{D}^{-1}\mathbf{A}\mathbf{H})\right], \tag{11}$$

and

$$\Xi_G = \frac{1}{n}\sum_{j=1}^{n} \mathbb{E}\left[\mathbf{H}_j^\top \mathbf{A}_j^\top (\mathbf{D}_j^{-1})^\top \mathbf{a}_j^\top \mathbf{a}_j \sigma(\mathbf{D}_j^{-1}\mathbf{A}_j\mathbf{H}_j)\right]. \tag{12}$$

It is easy to see that the above two matrices are invertible under Assumption 1. For learning NGNN (similar for GGNN), if we replace Ξ_N^{-1} by the gradient $\sigma'\left(\mathbf{D}^{-1}\mathbf{A}\mathbf{H}\right)$, we have exact gradients for $\mathbf{W}$ and $\mathbf{v}$ with a square loss $\frac{1}{n}\left\|\mathbf{y} - \sigma(\mathbf{D}^{-1}\mathbf{A}\mathbf{H}\mathbf{W})\mathbf{v}\right\|_2^2$. However, calculating $\sigma'\left(\mathbf{D}^{-1}\mathbf{A}\mathbf{H}\right)$ for some activation functions could be computationally expensive. Note that to compute these matrices of (11) and (12), one only needs to make an epoch pass on the dataset. Compared with a regular training by DNNs with tons of epoch passes on the entire dataset to acquire exact gradients, obtaining the two matrices adds very limited burden to the training computation. Hence, the gradient of $\mathbf{W}$ is inexact in the algorithm and a similar idea is also adopted in analysis of CNNs (Cao and Gu 2019).

Convergence Analysis

For the ease of presentation, we first introduce some quantities that we use in this paper.

$$\gamma_1 = \begin{cases} \sqrt{\frac{\alpha L_\sigma^2 d_{out}}{2} - 3\alpha^2(d+4d\bar{d})^2 L_\sigma^4}, & \text{NGNN}, \\ \sqrt{\frac{\alpha L_\sigma^2 d_{out}}{2} - 12\alpha^2 d^2 L_\sigma^4 n_{\max}^4}, & \text{GGNN}; \end{cases} \tag{13}$$

$$\gamma_2 = \begin{cases} \sqrt{\frac{2\alpha(d+4d\bar{d})^2}{d_{out}} + 6\alpha^2(d+4d\bar{d})^2 L_\sigma^2}, & \text{NGNN}, \\ \sqrt{\frac{8\alpha d^2 n_{\max}^4}{d_{out}} + 24\alpha^2 d^2 L_\sigma^4 n_{\max}^2}, & \text{GGNN}; \end{cases} \tag{14}$$

$$\gamma_3 = \sqrt{\frac{2\alpha + 3\alpha^2 L_\sigma^2 d_{out}}{\gamma_1^2 L_\sigma^2 d_{out}}}, \tag{15}$$

Algorithm 1: Approximate Gradient Descent for Learning GNNs

```
  Input: Training data $\{\mathbf{H}, \mathbf{y}\}$, number of iterations $T$,
         step size $\alpha$, initialization $\mathbf{W}_0, \mathbf{v}_0, t = 1$.
1 while $t < T$ do
2     $\bar{c} = \begin{cases} \mathbf{D}^{-1}\mathbf{A}\mathbf{H}, & \text{NGNN}, \\ \mathbf{D}_j^{-1}\mathbf{A}_j\mathbf{H}_j, & \text{GGNN}. \end{cases}$
3     $\bar{y} = \begin{cases} \sigma(c'\mathbf{W}_t)\mathbf{v}_t, & \text{NGNN}, \\ \mathbf{a}_j\sigma(c'\mathbf{W}_t)\mathbf{v}_t, & \text{GGNN}. \end{cases}$
4     $\mathbf{G}_t^W =$
        $\begin{cases} -\frac{1}{n}\Xi_N^{-1}\bar{c}^\top(\mathbf{y} - \bar{y})\mathbf{v}_t^\top, & \text{NGNN}, \\ -\frac{1}{n}\Xi_G^{-1}\sum_{j=1}^{n}\bar{c}^\top\mathbf{a}_j^\top(y_j - \bar{y})\mathbf{v}_t^\top, & \text{GGNN}. \end{cases}$
5     $\mathbf{g}_t^v =$
        $\begin{cases} -\frac{1}{n}\sigma(\mathbf{W}_t^\top\bar{c}^\top)(\mathbf{y} - \bar{y}), & \text{NGNN}, \\ -\frac{1}{n}\sum_{j=1}^{n}\sigma(\mathbf{W}_t^\top\bar{c}^\top)\mathbf{a}_j^\top(y_j - \bar{y}), & \text{GGNN}. \end{cases}$
6     $\mathbf{U}_{t+1} = \mathbf{W}_t - \alpha\mathbf{G}_t^W$
7     $\mathbf{W}_{t+1} = \mathbf{U}_{t+1}/\|\mathbf{U}_{t+1}\|_2$
8     $\mathbf{v}_{t+1} = \mathbf{v}_t - \alpha\mathbf{g}_t^v$
9 end
  Result: $\mathbf{W}_T, \mathbf{v}_T$
```

where $\bar{d} = \frac{1}{n}\sum_i d_i$ is the average node degree in the graph for NGNN, and $n_{\max} = \max_j n_j$ is the maximum number of nodes in a single graph for GGNN. Note that the above quantities only depend on the sample and feature properties, model architecture, and learning rate, and are independent of the number of samples n.

Next, we define

$$D = \max\left\{\|\mathbf{v}_0 - \mathbf{v}_*\|_2, \sqrt{\frac{2\gamma_2\|\mathbf{v}_*\|_2^2}{\gamma_1} + \gamma_3}\right\}, \tag{16}$$

and

$$\rho = \begin{cases} \min\left\{\mathbf{v}_0^\top\mathbf{v}_*, \frac{L_\sigma^2 d_{out}\|\mathbf{v}_*\|_2^2}{(d+4d\bar{d})L_\sigma^2+1}\right\}, & \text{NGNN}, \\ \min\left\{\mathbf{v}_0^\top\mathbf{v}_*, \frac{L_\sigma^2 d_{out}\|\mathbf{v}_*\|_2^2}{2dL_\sigma^2 n_{\max}^2+1}\right\}, & \text{GGNN}. \end{cases} \tag{17}$$

and let $D_0 = D + \|\mathbf{v}_*\|_2$. Similarly, the above quantities do not depend on the number of samples n.

Useful Lemmas

We provide in this section some useful lemmas to drive our main result. This could also serve as sketch of the proof of the main results and to help improve the presentation. First, two lemmas of the general results for both NGNN and GGNN are presented.

Lemma 1. *As the assumptions in Theorem 2 hold, there exists η_W such that we have*

$$\|\mathbf{W}_{t+1} - \mathbf{W}_*\|_2 - \frac{4\eta_W\left(1+\alpha\rho+\sqrt{1+\alpha\rho}\right)}{\rho} \leq \frac{1}{\sqrt{1+\alpha\rho}}\left[\|\mathbf{W}_t - \mathbf{W}_*\|_2 - \frac{4\eta_W\left(1+\alpha\rho+\sqrt{1+\alpha\rho}\right)}{\rho}\right]. \tag{18}$$

Lemma 2. *As the assumptions in Theorem 2 hold, there exist σ_m, σ_M, L, and η_v, such that we have*

$$\|\mathbf{v}_{t+1} - \mathbf{v}_*\|_2^2 \leq \left(1 - \frac{\alpha\sigma_m}{2} + 3\sigma_M^2\alpha^2\right) \|\mathbf{v}_t - \mathbf{v}_*\|_2^2 \tag{19}$$

$$+\left(\frac{\alpha L^2}{\sigma_m} + 3L^2\alpha^2\right) \|\mathbf{W}_t - \mathbf{W}_*\|_2^2 \|\mathbf{v}_*\|_2^2 + \left(\frac{2\alpha}{\sigma_m} + 3\alpha^2\right)\eta_v^2. \tag{20}$$

Next, we provide results specifically for NGNN and GGNN, respectively. We start by defining the following

$$\bar{\mathbf{G}}^W(\mathbf{W}, \mathbf{v}) = \mathbb{E}_{\mathbf{H}}\left[\mathbf{G}^W(\mathbf{W}, \mathbf{v})\right] \tag{21}$$

$$\bar{\mathbf{g}}^v(\mathbf{W}, \mathbf{v}) = \mathbb{E}_{\mathbf{H}}\left[\mathbf{g}^v(\mathbf{W}, \mathbf{v})\right] \tag{22}$$

Analysis for NGNN

Lemma 3. *If $n \geq \frac{8\ln\frac{2}{\delta}}{dd_{\min}}$, with probability at least $1-\delta$*

$$\frac{\left\|\mathbf{G}^W(\mathbf{W}, \mathbf{v}) - \mathbf{G}^W(\mathbf{W}', \mathbf{v})\right\|_2}{\|\mathbf{W} - \mathbf{W}'\|_2} \leq D_0^2 \left\|\Xi^{-1}\right\|_2 (d + 4d\bar{d}), \tag{23}$$

$$\frac{\left\|\mathbf{G}^W(\mathbf{W},\mathbf{v}) - \mathbf{G}^W(\mathbf{W},\mathbf{v}')\right\|_2}{\|\mathbf{v} - \mathbf{v}'\|_2} \leq 4D_0 \left\|\Xi^{-1}\right\|_2 (d+4d\bar{d})L_\sigma, \tag{24}$$

$$\frac{\|\mathbf{g}^v(\mathbf{W}, \mathbf{v}) - \mathbf{g}^v(\mathbf{W}', \mathbf{v})\|_2}{\|\mathbf{W} - \mathbf{W}'\|_2} \leq 5(d + 4d\bar{d})D_0L_\sigma$$

$$+ \frac{d + 4d\bar{d}}{2} + \nu^2, \tag{25}$$

$$\frac{\|\mathbf{g}^v(\mathbf{W}, \mathbf{v}) - \mathbf{g}^v(\mathbf{W}, \mathbf{v}')\|_2}{\|\mathbf{v} - \mathbf{v}'\|_2} \leq L_\sigma^2(d + 4d\bar{d}). \tag{26}$$

Lemma 4. *If $\sqrt{n\log n} \geq \frac{3}{2}\log\frac{2}{\delta}$, with probability at least $1-\delta$*

$$\left\|\mathbf{G}^W - \bar{\mathbf{G}}^W\right\|_2 \tag{27}$$

$$\leq \frac{4}{c}\sqrt{\frac{\log n}{n}} \left\|\Xi^{-1}\right\|_2 D_0\left(D_0\frac{d}{d_{\min}}(1+|\sigma(0)|) + \sqrt{\frac{d}{d_{\min}}}\nu\right) \tag{28}$$

$$\|\mathbf{g}^v - \bar{\mathbf{g}}^v\|_2 \tag{29}$$

$$\leq \frac{2}{c}\sqrt{\frac{\log n}{n}}\left(D_0\frac{d}{d_{\min}}(1+|\sigma(0)|)^2 + \sqrt{\frac{d}{d_{\min}}}(1+|\sigma(0)|)\nu\right) \tag{30}$$

for some absolute constant c.

Analysis for GGNN

Lemma 5. *If $\sum_{j=1}^n n_j \geq \frac{8\ln\frac{1}{\delta}}{d}$, with probability at least $1-\delta$*

$$\frac{\left\|\mathbf{G}^W(\mathbf{W}, \mathbf{v}) - \mathbf{G}^W(\mathbf{W}', \mathbf{v})\right\|_2}{\|\mathbf{W} - \mathbf{W}'\|_2} \leq 2dD_0^2 \left\|\Xi^{-1}\right\|_2 n_{\max}^2, \tag{31}$$

$$\frac{\left\|\mathbf{G}^W(\mathbf{W}, \mathbf{v}) - \mathbf{G}^W(\mathbf{W}, \mathbf{v}')\right\|_2}{\|\mathbf{v} - \mathbf{v}'\|_2} \leq 8D_0 d\Xi^{-1} n_{\max}^2 L_\sigma, \tag{32}$$

$$\frac{\|\mathbf{g}^v(\mathbf{W}, \mathbf{v}) - \mathbf{g}^v(\mathbf{W}', \mathbf{v})\|_2}{\|\mathbf{W} - \mathbf{W}'\|_2} \leq 10dD_0L_\sigma n_{\max}^2$$

$$+ d\sqrt{n_{\max}^3} + \sqrt{n_{\max}}\nu^2, \tag{33}$$

$$\frac{\|\mathbf{g}^v(\mathbf{W}, \mathbf{v}) - \mathbf{g}^v(\mathbf{W}, \mathbf{v}')\|_2}{\|\mathbf{v} - \mathbf{v}'\|_2} \leq 2dL_\sigma^2 n_{\max}^2. \tag{34}$$

Lemma 6. *If $\sqrt{n\log n} \geq \frac{3}{2}\log\frac{2}{\delta}$, with probability at least $1-\delta$*

$$\left\|\mathbf{G}^W - \bar{\mathbf{G}}^W\right\|_2 \tag{35}$$

$$\leq \frac{4}{c}\sqrt{\frac{\log n}{n}} \left\|\Xi^{-1}\right\|_2 \sqrt{n_{\max}}\left(\sqrt{n_{\max}}2(1+|\sigma(0)|)D_0 + \nu\right), \tag{36}$$

$$\|\mathbf{g}^v - \bar{\mathbf{g}}^v\|_2 \tag{37}$$

$$\leq \frac{4}{c}\sqrt{\frac{\log n}{n}}\sqrt{n_{\max}}(1+|\sigma(0)|)\left(\sqrt{n_{\max}}2(1+|\sigma(0)|)D_0 + \nu\right) \tag{38}$$

for some absolute constant c.

Next, we provide the main results for NGNN and GGNN with the proposed algorithms.

Theorem 2. *Suppose that the initialization $(\mathbf{W}_0, \mathbf{v}_0)$ satisfies*

$$\mathrm{Tr}\left(\mathbf{W}_*^\top \mathbf{W}_0\right) \geq 0, \mathbf{v}_*^\top \mathbf{v}_0 \geq 0, \tag{39}$$

and the learning rate α is chosen such that

$$\alpha \leq \begin{cases} \frac{1}{2(\|\mathbf{v}_0 - \mathbf{v}_*\|_2^2 + \|\mathbf{v}_*\|_2^2)} \wedge \frac{d_{out}}{6(d+4d\bar{d})^2L_\sigma^2}, & \textit{NGNN}, \\ \frac{1}{2(\|\mathbf{v}_0 - \mathbf{v}_*\|_2^2 + \|\mathbf{v}_*\|_2^2)} \wedge \frac{d_{out}}{24d^2L_\sigma^2 n_{\max}^4}, & \textit{GGNN}. \end{cases} \tag{40}$$

If the number of samples n is large enough such that for NGNN

$$\frac{2}{c}\sqrt{\frac{\log n}{n}} \leq \frac{\frac{\rho\,\mathrm{Tr}(\mathbf{W}_*^\top \mathbf{W}_0)}{16(1+\alpha\rho)\|\Xi^{-1}\|_2 D_0} \wedge \frac{\rho}{\|\mathbf{v}_*\|_2(1+|\sigma(0)|)}}{D_0\frac{d}{d_{\min}^2}(1+|\sigma(0)|) + \frac{d}{d_{\min}}\nu}, \tag{41}$$

and for GGNN

$$\frac{2}{c}\sqrt{\frac{\log n}{n}} \leq \frac{\frac{\rho\,\mathrm{Tr}(\mathbf{W}_*^\top \mathbf{W}_0)}{16(1+\alpha\rho)\|\Xi^{-1}\|_2 D_0} \wedge \frac{\rho}{\|\mathbf{v}_*\|_2(1+|\sigma(0)|)}}{n_{\max}(1+|\sigma(0)|)D_0 + \sqrt{n_{\max}}\nu}, \tag{42}$$

for some large enough absolute constant c, then with probability at least $1-\delta$ and for large n such that $\sqrt{n\log n} \geq \frac{3}{2}\log\frac{2}{\delta}$ and $n \geq \frac{8\ln\frac{2}{\delta}}{dd_{\min}}$ for NGNN or $\sum_{j=1}^n n_j \geq \frac{8\ln\frac{1}{\delta}}{d}$ for GGNN, we have that

$$\|\mathbf{W}_t - \mathbf{W}_*\|_2 \leq \left(\frac{1}{\sqrt{1+\alpha\rho}}\right)^t \|\mathbf{W}_0 - \mathbf{W}_*\|_2 \tag{43}$$

$$+ \eta_W(6\alpha + \frac{8}{\rho}) \tag{44}$$

with

$$\bar{a}_W = \frac{4}{c}\sqrt{\frac{\log n}{n}} \left\|\Xi^{-1}\right\|_2 D_0, \tag{45}$$

and

$$\eta_W = \begin{cases} \bar{a}_W \left(D_0 \frac{d}{d_{\min}}(1+|\sigma(0)|) + \sqrt{\frac{d}{d_{\min}}\nu} \right), \textit{NGNN}, \\ \bar{a}_W \sqrt{n_{\max}} \left(\sqrt{n_{\max}}(1+|\sigma(0)|)D_0 + \nu \right), \textit{GGNN}; \end{cases} \tag{46}$$

and

$$\|\mathbf{v}_t - \mathbf{v}_*\|_2 \le \left(\sqrt{1-\gamma_1^2}\right)^t \|\mathbf{v}_0 - \mathbf{v}_*\|_2$$
$$+ \gamma_2 \|\mathbf{W}_0 - \mathbf{W}_*\|_2 \|\mathbf{v}_*\|_2 \sqrt{t} \left(\sqrt{1-\gamma_1^2} \vee \sqrt{\frac{1}{1+\alpha\rho}} \right)^{t-1}$$
$$+ \gamma_3 \eta_v + \frac{\gamma_2}{\gamma_1}(6\alpha + \frac{8}{\rho}) \|\mathbf{v}_*\|_2 \eta_W \tag{47}$$

with

$$\bar{a}_v = \frac{2}{c}\sqrt{\frac{\log n}{n}}, \tag{48}$$

and

$$\eta_v = \begin{cases} \bar{a}_v \left(D_0 \frac{d}{d_{\min}}(1+|\sigma(0)|)^2 + \sqrt{\frac{d}{d_{\min}}}(1+|\sigma(0)|)\nu \right), \textit{NGNN}, \\ \bar{a}_v \sqrt{n_{\max}}(1+|\sigma(0)|) \left(\sqrt{n_{\max}}(1+|\sigma(0)|)D_0 + \nu \right), \textit{GGNN}. \end{cases} \tag{49}$$

Remark 1. *Essentially, it is shown in Theorem 2 that the proposed algorithm can provably learn GNNs with a linear convergence to the true parameters of GNNs up to a statistical error. The error is governed by the number of training samples n, and approaches 0 when n is infinitely large. Therefore, exact convergence of the algorithm is guaranteed for large n, and this result is intuitively desirable and expected.*

Remark 2. *Theorem 2 requires that the initialization satisfies* (39)*, which may not necessarily be satisfied by a random initialization. One solution is to try out all 4 sign combinations of*

$$(\mathbf{W}_0, \mathbf{v}_0) \in \{(\mathbf{W}, \mathbf{v}), (-\mathbf{W}, \mathbf{v}), (\mathbf{W}, -\mathbf{v}), (-\mathbf{W}, -\mathbf{v})\}$$

as suggested in (Du et al. 2017). This guarantees that the initialization condition of (39) *can be satisfied, and further the convergence of the algorithm.*

Remark 3. *The conditions on the number of samples in* (41) *and* (42) *pose sufficient conditions on the number of samples n for learning NGNN and GGNN, respectively. Intuitively, an efficient learning algorithm needs sufficiently many learning samples, and we provide such analytical results in this regard by the two conditions.*

It would also be interesting to explore the impact of the structures of the graphs on the convergence, e.g., the convergence could be possibly optimized with respect to the spectral properties fo the graphs . Although we include node degrees, number of nodes in the derived convergence, which also relate to the structures of the graphs, we leave such a work to the interested community and our future work due to the length limit of this paper.

While the general differences between the GNNs and CNNs are discussed in Related Work, we emphasize hereby that the resulting difference for technical analysis is even more significant. This is due to unordered and overlapping neighbors in GNNs. There has been limited work on CNNs with overlapping structures (see (Cao and Gu 2019) and the references therein). For technical analysis, non-overlapping CNNs yields independent data variables for each convolution patch and then tractable concentration bounds like Lemma 5.2 in (Cao and Gu 2019), which leads to easier and simpler analysis compared with the analysis in this paper. On the contrary, Also, the analysis over graph for GNNs adds much more difficulties compared with that of CNNs. We develop new techniques in the theoretical analysis to tackle challenges from dependent variables (from overlapping neighbors) in graphs (it is equivalently grid for CNNs), and to design provable efficient algorithm on top of it. Please refer to Lemmas 3-10 in the supplementary material for related detail. Thus, the analyses for CNNs are special cases in our paper.

Training Dynamics

In the previous section, we show that the outputs of the proposed algorithms are provably convergent to underlying true parameters. At this point, we have not fully understood the training dynamics of the estimated parameters $\mathbf{W}_t$ and $\mathbf{v}_t$. A missing building brick to the analysis would be the proposition of a compact subspace that the training sequences of $\mathbf{W}_t$ and $\mathbf{v}_t$ lie within.

Toward this goal, we first define the space

$$\mathcal{W} = \left\{\mathbf{W} : \|\mathbf{W}\|_2 = 1, \mathrm{Tr}(\mathbf{W}_*^\top \mathbf{W}) \ge \mathrm{Tr}\left(\mathbf{W}_*^\top \mathbf{W}_0\right)/2\right\}, \tag{50}$$

and

$$\mathcal{V} = \left\{\mathbf{v} : \|\mathbf{v} - \mathbf{v}_*\|_2 \le D, \mathbf{v}_*^\top \mathbf{v} \ge \rho\right\}. \tag{51}$$

We then have the following theorem on the stability of the training procedure.

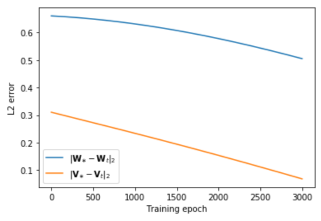
Theorem 3. *Suppose the assumptions in Theorem 2 hold, then the training dynamics are confined to*

$$\{\mathbf{W}_t, \mathbf{v}_t\} \in \mathcal{W} \times \mathcal{V}. \tag{52}$$

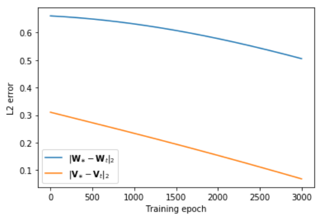
Theorem 3 states that the training process of the proposed learning algorithm is provably stable, in addition to Theorem 2 which grants convergence of the training outputs.

Experimental Results

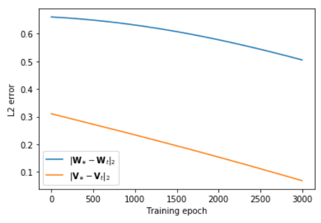
We provide numerical experiments to support and validate our theoretical analysis. We test Algorithm 1 for NGNN and GGNN tasks, respectively. Different activation functions of ReLU, Leaky ReLU, Sigmoid, Tanh and Swish are used in the networks, and the results with respect to the distance between the estimated and the true parameters, i.e., $\|\mathbf{W}_* - \mathbf{W}_t\|_2$ and $\|\mathbf{v}_* - \mathbf{v}_t\|_2$, versus the number of training epochs are



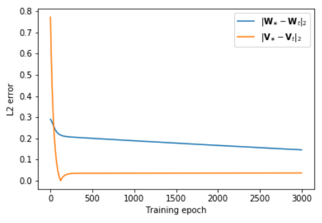
(a) ReLU



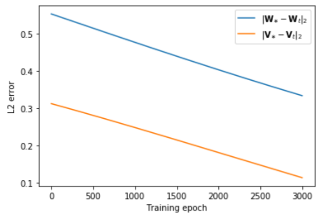
(b) Leaky ReLU (0.2)



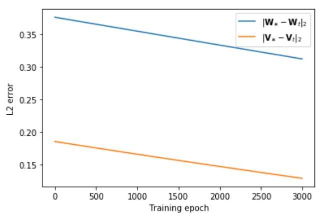
(c) Leaky ReLU (0.05)



(d) Sigmoid

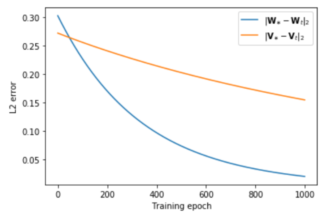


(e) Tanh

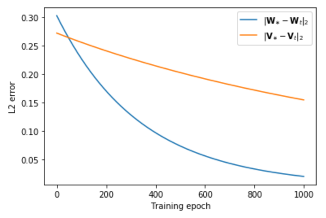


(f) Swish

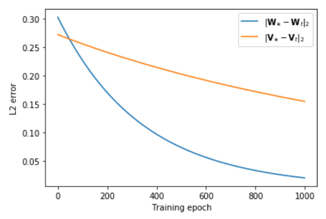
Figure 1: Training dynamics for NGNN with different activation functions.



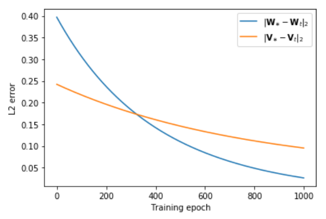
(a) ReLU



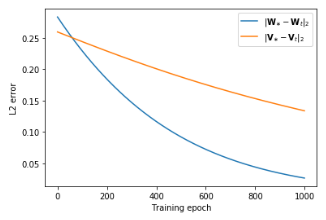
(b) Leaky ReLU (0.2)



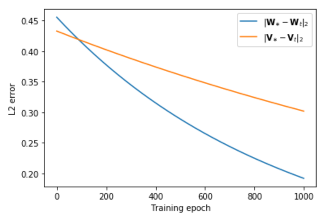
(c) Leaky ReLU (0.05)



(d) Sigmoid



(e) Tanh



(f) Swish

Figure 2: Training dynamics for GGNN with different activation functions.

reported. Specifically, for the networks with Leaky ReLU, we show results for two different slope parameter of 0.2 and 0.05. We choose $d = 2$ and $d_{out} = 1$, and set the variance ν to 0.04. We generate $\mathbf{W}_*$ from unit sphere with a normalized Gaussian matrix, and generate $\mathbf{v}_*$ as a standard Gaussian vector. The nodes in the graphs are probabilistically connected according to the distribution of Bernoulli(0.5).

NGNN Tasks

For node-level tasks, we have one graph and 1000 nodes.

The learning rate α is chosen as 0.1. As presented in Figure 1, we can see clearly the stable training of the GNN and the efficiency of the algorithm. It is show that $\mathbf{W}$ converges slower than $\mathbf{v}$. Another interesting finding is that when Sigmoid is used as the activation function, the training of $\mathbf{v}$ very quickly converges to the optimum, and then lingers at the neighborhood of $\mathbf{v}_*$. This observation validates our theoretical result that the convergence is up to a statistical error. Also due to this reason, the actual convergence might be slightly slower than linear rate, which however is inevitable because of the statistical model that we consider.

GGNN Tasks

For GGNN tasks, we have 1000 graphs and each graph has 5 nodes. The learning rate α is chosen as 0.005. As presented in Figure 2, we can see clearly the training dynamics of the GNN and the efficiency of the algorithm. The curves are smooth, indicating that the training is very stable. By looking at the slope of the curves, for NGNN, we find that $\mathbf{W}$ converges faster than $\mathbf{v}$, which is very different from the NGNN task.

Conclusion

In this paper, we developed the first provably efficient algorithm for learning GNNs with one hidden layer for node information convolution. We investigated two types of GNNs, namely, node-level and graph-level GNNs. Our results provided a comprehensive framework and a set of tools for designing and analyzing GNNs. More importantly, our results only rely on mild condition on the activation functions, which can be satisfied by a wide range of activation functions used in practice. We constructed our training algorithm using the idea of approximate gradient descent, and proved that it recovers the true parameters of the teacher network with a linear convergence rate. How fast the algorithm converges depends on various parameters of the algorithm, which was also analytically characterized.

References

Brutzkus, A.; and Globerson, A. 2017. Globally optimal gradient descent for a convnet with gaussian inputs. In *International Conference on Machine Learning*, 605–614.

Cao, Y.; and Gu, Q. 2019. Tight Sample Complexity of Learning One-hidden-layer Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, 10611–10621.

Choromanska, A.; Henaff, M.; Mathieu, M.; Arous, G. B.; and LeCun, Y. 2015. The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, 192–204.

Du, S.; and Lee, J. 2018. On the Power of Overparametrization in Neural Networks with Quadratic Activation. In *International Conference on Machine Learning*, 1329–1338.

Du, S. S.; Lee, J. D.; and Tian, Y. 2017. When is a convolutional filter easy to learn? *arXiv preprint arXiv:1709.06129* .

Du, S. S.; Lee, J. D.; Tian, Y.; Poczos, B.; and Singh, A. 2017. Gradient descent learns one-hidden-layer cnn: Don't be afraid of spurious local minima. *arXiv preprint arXiv:1712.00779* .

Du, S. S.; Wang, Y.; Zhai, X.; Balakrishnan, S.; Salakhutdinov, R. R.; and Singh, A. 2018. How many samples are needed to estimate a convolutional neural network? In *Advances in Neural Information Processing Systems*, 373–383.

Freeman, C. D.; and Estrach, J. B. 2019. Topology and geometry of half-rectified network optimization. In *International Conference on Learning Representations*.

Gautier, A.; Nguyen, Q. N.; and Hein, M. 2016. Globally optimal training of generalized polynomial neural networks with nonlinear spectral methods. In *Advances in Neural Information Processing Systems*, 1687–1695.

Ge, R.; Lee, J. D.; and Ma, T. 2017. Learning one-hidden-layer neural networks with landscape design. *arXiv preprint arXiv:1711.00501* .

Goel, S.; Kanade, V.; Klivans, A.; and Thaler, J. 2017. Reliably Learning the ReLU in Polynomial Time. In *Conference on Learning Theory*, 1004–1042.

Goel, S.; and Klivans, A. 2017a. Eigenvalue decay implies polynomial-time learnability for neural networks. In *Advances in Neural Information Processing Systems*, 2192–2202.

Goel, S.; and Klivans, A. 2017b. Learning depth-three neural networks in polynomial time. *arXiv preprint arXiv:1709.06010* .

Goel, S.; Klivans, A.; and Meka, R. 2018. Learning One Convolutional Layer with Overlapping Patches. In *International Conference on Machine Learning*, 1783–1791.

Haeffele, B. D.; and Vidal, R. 2015. Global optimality in tensor factorization, deep learning, and beyond. *arXiv preprint arXiv:1506.07540* .

Hardt, M.; and Ma, T. 2016. Identity matters in deep learning. *arXiv preprint arXiv:1611.04231* .

Janzamin, M.; Sedghi, H.; and Anandkumar, A. 2015. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *arXiv preprint arXiv:1506.08473* .

Jin, C.; Ge, R.; Netrapalli, P.; Kakade, S. M.; and Jordan, M. I. 2017. How to escape saddle points efficiently. In *International Conference on Machine Learning*, 1724–1732.

Kawaguchi, K. 2016. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, 586–594.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, 1097–1105.

LeCun, Y.; Bengio, Y.; et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361(10): 1995.

Li, Q.; Zhou, Y.; Liang, Y.; and Varshney, P. K. 2017. Convergence analysis of proximal gradient with momentum for nonconvex optimization. In *International Conference on Machine Learning*, 2111–2119.

Li, Y.; and Yuan, Y. 2017. Convergence analysis of two-layer neural networks with relu activation. In *Advances in Neural Information Processing Systems*, 597–607.

Nguyen, Q.; and Hein, M. 2017a. The loss surface and expressivity of deep convolutional neural networks. *arXiv preprint arXiv:1710.10928* .

Nguyen, Q.; and Hein, M. 2017b. The loss surface of deep and wide neural networks. In *International Conference on Machine Learning*, 2603–2612.

Safran, I.; and Shamir, O. 2016. On the quality of the initial basin in overspecified neural networks. In *International Conference on Machine Learning*, 774–782.

Safran, I.; and Shamir, O. 2018. Spurious Local Minima are Common in Two-Layer ReLU Neural Networks. In *International Conference on Machine Learning*, 4433–4441.

Schmidt, M.; Roux, N. L.; and Bach, F. R. 2011. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Advances in Neural Information Processing Systems*, 1458–1466.

Sedghi, H.; and Anandkumar, A. 2014. Provable methods for training neural networks with sparse connectivity. *arXiv preprint arXiv:1412.2693* .

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676): 354–359.

Soltanolkotabi, M. 2017. Learning relus via gradient descent. In *Advances in Neural Information Processing Systems*, 2007–2017.

Song, L.; Vempala, S.; Wilmes, J.; and Xie, B. 2017. On the complexity of learning neural networks. In *Advances in Neural Information Processing Systems*, 5514–5522.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. The MIT Press.

Tian, Y. 2017. An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis. In *International Conference on Machine Learning*, 3404–3413.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2019. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596* .

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* .

Young, T.; Hazarika, D.; Poria, S.; and Cambria, E. 2018. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine* 13(3): 55–75.

Zhang, Y.; Lee, J. D.; and Jordan, M. I. 2016. l1-regularized neural networks are improperly learnable in polynomial time. In *International Conference on Machine Learning*, 993–1001.

Zhang, Y.; Lee, J. D.; Wainwright, M. J.; and Jordan, M. I. 2015. Learning halfspaces and neural networks with random initialization. *arXiv preprint arXiv:1511.07948* .

Zhong, K.; Song, Z.; and Dhillon, I. S. 2017. Learning non-overlapping convolutional neural networks with multiple kernels. *arXiv preprint arXiv:1711.03440* .

Zhong, K.; Song, Z.; Jain, P.; Bartlett, P. L.; and Dhillon, I. S. 2017. Recovery guarantees for one-hidden-layer neural networks. In *International Conference on Machine Learning*, 4140–4149.

Zhou, J.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; and Sun, M. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434* .

Zhou, P.; and Feng, J. 2017. The landscape of deep learning algorithms. *arXiv preprint arXiv:1705.07038* .