Mathematical Studies in Deep Learning: Overparameterization Theories, Graph Generative Models and Applications in Algorithmic Pricing

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Yinglong Guo

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Gilad Lerman, Adviser

November 2024

© Yinglong Guo 2024

Acknowledgements

I would like to express my gratitude to the individuals who have supported and guided me throughout my research journey.

First and foremost, I am deeply thankful to my advisor, Dr. Gilad Lerman, whose invaluable guidance, thoughtful feedback, and constant encouragement have been a source of inspiration throughout my research. I also express my sincere appreciation to the members of my dissertation committee, Dr. Jeff Calder, Dr. William Leeb and Dr. Ju Sun, for their insightful feedback and to the faculty members in the School of Mathematics who offered their expertise and assistance. Special thanks to my fellow PhD candidates in the School of Mathematics, especially Shaohan Li and Cristian Camilo Chica Castano, for the stimulating discussions.

I want to express gratitude to my parents for their unwavering support in my decisions, including the pursuit of a PhD.

To my wife, Xiaoou Li, thank you for your presence, care, and support at every moment through this journey. To our beloved cats, Zhangfei and Xiahou, who provided soothing purrs and playful companionship.

The work reported in this dissertation was partially supported by NSF DMS award 2124913.

Abstract

This dissertation explores three key areas in deep learning. First, it extends the existing theoretical results on overparameterized deep neural networks to the ones incorporating leaky ReLU activation functions, offering insights into effective hyperparameter selection for such networks. Second, it proposes a novel graph data generation method, introducing an unpooling layer that leverages graph structures and resolves the non-differentiability in training using the policy gradient approach. Finally, the dissertation examines algorithmic pricing in two-sided markets, investigating how Q-learning strategies may lead to unintended collusion and proposing policy to mitigate this risk.

Contents

List of Tables **vii**

List of Figures **ix**

1 Introduction **1**

2 The Effect of Leaky ReLUs on the Training and Generalization of Overparameterized Networks **3**

2.1 Introduction 4

2.2 Problem Setup 7

2.3 Main Results 10

2.4 Ideas of Proofs 15

2.4.1 Sketch of Proofs 16

2.4.2 Discussion of Innovation 20

2.5 Numerical Experiments 23

2.5.1 Setup 23

2.5.2 Results 24

2.6 Discussion 27

3 An Unpooling Layer for Graph Generation **29**

3.1 Introduction 30

3.1.1 Related Work 31

3.1.2 This Work 33
3.2 Methodology 35
3.2.1 Unpooling Layer 35
3.2.2 Graph Generation and Training 40
3.3 Theoretical Guarantees 42
3.3.1 Guarantee of Connectivity of the Output Graph 42
3.3.2 Guarantee of Expressivity for the Unpooling Layer 42
3.4 Experiments 43
3.4.1 Datasets 43
3.4.2 Evaluation Metrics 45
3.4.3 Implementation Details 45
3.4.4 Results 47
3.5 Conclusion and Limitations 54

4 Competition and Collusion in Two-Sided Markets with an Outside Option 56
4.1 Introduction 57
4.2 Review of Our Economic Framework 60
4.2.1 The Baseline Platform Competition Game 60
4.2.2 The Infinite Repeated Game 62
4.3 Simulation Framework 64
4.3.1 Preliminaries: Multi-agent Reinforcement Learning 65
4.3.2 The Simulation Setup 66
4.4 Experimental Results 69
4.4.1 Dependence of the Collusive Level on the Network Externalities 70
4.4.2 A Study of the Collusive Level under Special Network Externalities 75

4.4.3 A Study of the Collusive Level under Special Market Parameters 79
4.4.4 Discussion of Exceptional Cases 83
4.5 Economic and Policy Discussion 87
4.5.1 Economic Discussion 88
4.5.2 Policy Discussion 91
4.6 Concluding Remarks 94

References **96**

A Supplementary Material for Chapter 2 **110**
A.1 Discussion of the Generalization Error Bound 110
A.2 Proofs 113
A.2.1 Notation 114
A.2.2 Initialization 117
A.2.3 Perturbation 131
A.2.4 Gradient Bounds and Proof of Lemma 2.2 144
A.2.5 Proof of Lemma 2.1 163
A.2.6 Conclusion of the Proof of Theorem 2.1 170
A.2.7 Proof of Theorem 2.2 172
A.2.8 Proof of Lemma 2.3 178
A.2.9 Proof of Theorem 2.3 185
A.2.10 Generalization Error Bound for SGD 185
A.2.11 Special dataset 186
A.2.12 Convergence Theorem for General Convex Loss Functions . . 189
A.2.13 Proofs for a special class of datasets 194
A.3 Supplemental numerical experiments and details for the previous experiments 202
A.3.1 Details of Implementation 203

A.3.2 Additional numerical results 208

B Supplementary Material for Chapter 3 214

B.1 Additional Details of the Unpooling Layer 214

B.2 Comments on the Codes 221

B.3 Implementation Details 222

B.3.1 Details of Other Neural Layers 222

B.3.2 Details of the Unpooling Layer 224

B.3.3 Architectures of Discriminators and Encoders 228

B.3.4 Architecture of Generators 229

B.3.5 Reconstruction Loss for UL VAE 230

B.4 Proof of Theorems 236

B.4.1 Proof of Proposition 1 236

B.4.2 Proof of Theorem 2 237

B.4.3 Proof of Corollary 3 247

B.5 Optimizing Specific Chemical Properties 247

B.6 Additional Numerical Results 249

B.6.1 Standard Deviations for Molecule Generation 249

B.6.2 Evaluation with only 1,000 generated samples. 250

B.6.3 Synthetic Samples for Molecule Generation 251

C Supplementary Material for Chapter 4 257

C.1 Sensitivity Analysis and Examples of Convergence Paths 257

C.2 Fitting a Sequence of Non-linear Functions 268

C.3 Additional Simulation Results 270

List of Tables

2.1 Training and testing errors for the synthetic dataset, F-MNIST and CIFAR-10 24

3.1 Summary metrics of data generation methods using the Waxman random graph dataset 48
3.2 Summary metrics of data generation methods using the protein dataset 49
3.3 Validity, uniqueness, novelty and their geometric mean for molecule generation using QM9 51
3.4 Validity, uniqueness, novelty and their geometric mean for molecule generation using ZINC 51

A.1 Architecture of the NNs with Leaky ReLU parameter α used for the synthetic dataset. 204
A.2 Architecture of the NNs with Leaky ReLU parameter α used for California housing. 205
A.3 Architecture of the neural networks with α Leaky ReLU parameter used for MNIST. 205
A.4 Architecture of the Transformer neural networks with α Leaky ReLU parameter used for IMDB movie reviews. 206
A.5 Architecture of the neural networks with α Leaky ReLU parameter with width of m and depth of L used for Fashion MNIST. 206

A.6 Architecture of the neural networks with α Leaky ReLU parameter used for CIFAR-10. 207
A.7 Architecture of the neural networks with α Leaky ReLU parameter used for IMDB movie reviews dataset 208

B.1 The six evaluation metrics for generated samples that aim to minimize the indicated chemical properties 249
B.2 Validity, uniqueness and novelty with standard deviation for Adj GAN, UL GAN, and UL VAE using QM9. 250
B.3 Validity, Uniqueness and Novelty with standard deviation for Adj GAN and UL GAN using ZINC. 250
B.4 Model performance with 1,000 generated samples on Waxman and protein datasets. 251
B.5 Model performance with 1,000 generated samples on QM9 and ZINC datasets. 251

C.1 Sensitivity analysis of the numerical simulation with $\Phi = \Phi_1 := [0, 0; 0, 0]$.263
C.2 Sensitivity analysis of the numerical simulation with $\Phi = \Phi_2 := [1, 0; 0, 1]$.264
C.3 Sensitivity analysis of the numerical simulation with $\Phi = \Phi_3 := [0, 1; 1, 0]$.265

List of Figures

2.1 Log-scale training and testing errors using synthetic dataset, F-MNIST and CIFAR-10 and different α's 22

2.2 Comparison of the "shape" of the theoretical upper bound of the training convergence rate with the calculated convergence rate 26

3.1 Demonstration of possible outputs of the proposed unpooling layer . . 33

3.2 Demonstration of the steps of the unpooling layer 34

3.3 Demonstration of a generative neural network with unpooling and MPNN layers 37

3.4 Distributions of node connectivity, average clustering coefficient, edge density and node features for the generated and source data using the Waxman random graph dataset 52

3.5 Demonstration of gradual change between two different molecular properties when applying UL GAN for QM9 53

4.1 Demonstration of the infinite repeated game framework 64

4.2 Demonstration of the dependence of the four fitted univariate functions on the externalities 71

4.3 Heatmaps of the fitted functions $f_{bb,ss}(\phi_{bb}, \phi_{ss})$ and $f_{sb,bs}(\phi_{bs}, \phi_{sb})$ . . . 72

4.4 Heatmaps of the fitted functions $f_{bb,sb}(\phi_{bb}, \phi_{sb})$ and $f_{ss,bs}(\phi_{ss}, \phi_{bs})$ . . . 73

4.5 Heatmaps of the fitted functions $f_{bb,bs}(\phi_{bb}, \phi_{bs})$ and $f_{ss,sb}(\phi_{ss}, \phi_{sb})$ . . . 75

4.6 Collusive level with varying ϕ_{bb} in the externality matrix 76
4.7 Collusive level with varying ϕ_{bs} in the externality matrix 77
4.8 Collusive level with varying ϕ_{bb} in the externality matrix with non-zero off-diagonal elements 78
4.9 Collusive level with varying the idiosyncratic preference parameter. . 80
4.10 Collusive level with varying the outside option utility $u_k^{(0)}$ 82
4.11 Collusive level with varying discount rate δ 82
4.12 Maximal average collusive level with varying ϕ_{bb} and ϕ_{bs} in the externality matrix 85
4.13 Collusive level with varying ϕ_{sb}, where $\Phi = [0, \phi_{sb}; \phi_{sb}, 0]$. 86
4.14 Demonstration of an uncommon scenario where the collusion prices can be lower than competition prices 87
4.15 Collusive level with varying penalty coefficient. 93

A.1 Log-scale training and testing errors using MNIST and California housing datasets and different α's 210
A.2 Log-scale training and testing errors using IMDB and MNIST datasets and different α's 211
A.3 Log-scale errors on F-MNIST with different α's 212
A.4 Log-scaled mean square errors for the synthetic dataset and California housing dataset with different α's 213

B.1 Demonstration of the architecture of the generator in UL GAN and the decoder in UL VAE for Waxman random graph dataset 232
B.2 Demonstration of the architecture of the generator in UL GAN and the decoder in UL VAE for protein dataset 233
B.3 Demonstration of the architecture of the generator in UL GAN and the decoder in UL VAE for QM9 234

B.4 Demonstration of the architecture of the generator in UL GAN for ZINC 235
B.5 Samples of molecules generated by UL GAN based on QM9 dataset. . 252
B.6 Samples of molecules generated from UL VAE based on QM9 dataset. 253
B.7 Samples of molecules generated from UL GAN based on ZINC dataset. 254
B.8 Five examples of intermediate graphs from generating graphs using UL GAN trained with ZINC dataset 255
B.9 Five examples of intermediate graphs from generating graphs using UL GAN trained with ZINC dataset 256

C.1 Examples of the converging state in numerical simulation for the platforms 266
C.2 Examples of the converging paths from Nash equilibrium state in numerical simulation for the platforms 268
C.3 The dependence of the collusive levels on the idiosyncratic preference parameter β_k in additional settings. 271
C.4 The dependence of the collusive levels on the outside utility $u_k^{(0)}$ in additional settings. 272

Chapter 1

Introduction

Deep learning is an emerging technique that has dominated many fields in machine learning in recent decades, demonstrating superior performance in a variety of tasks, such as regression, classification, data generation, and reward optimization.

This dissertation contributes to three areas in deep learning: (1) establishing a theoretical foundation to better understand the performance of deep neural networks and to guide the hyperparameter selection in regression and classification tasks, (2) developing a novel method for graph data generation, and (3) investigating the collusive behavior in two-sided markets driven by algorithmic pricing through numerical simulations with reinforcement learning.

Previous work has established theoretical frameworks for understanding the training convergence and generalization of overparamterized deep neural network with rectified Linear Unit (ReLU) activation function. This dissertation extends previous work by improving the theoretical results and applying them to leaky ReLU activation functions, a class of activation functions that has demonstrated superior performance in regression, classification and data generation tasks. Based on the theoretical results, we further provide insights into hyperparamter selection for such neural networks equipped with leaky ReLU. Recent advancements in generative models, such as Generative Adversarial Networks (GAN) and Variational Autoencoders (VAE), have significantly improved the performance of generating image and text. However, the generation of graph data, which is essential for applications such as drug discovery, has received less attention. This dissertation introduces a novel unpooling layer for graph generation, which effectively leverages graph structures and can be seamlessly integrated into

current generative models. In the realm of economics, deep learning has also become more important as companies start using algorithmic pricing strategies, including using Q-learning to set prices dynamically in the market. This trend raises concerns about the potential for unintended collusion, where companies reach prices higher than the equilibrium, due to algorithmic pricing. Focusing on two-sided markets, this dissertation investigates the factors that lead to the collusive behavior in algorithmic pricing and proposes potential mitigation strategies.

The rest of the dissertation is organized as follows. Chapter 2 establishes theorems on the training and generalization error of overparamterized deep neural networks with leaky ReLU activation functions and presents extensive numerical experiments. Chapter 3 proposes a novel method to generate graph data and includes several theoretic properties of this new method and numerical results comparing this new method and existing methods. Chapter 4 executes numerical simulation and studies the collusive behavior in this market caused by algorithmic pricing following Q-learning.

Chapter 2

The Effect of Leaky ReLUs on the Training and Generalization of Overparameterized Networks

This chapter investigates the training and generalization errors of overparameterized neural networks (NNs) with a wide class of leaky rectified linear unit (ReLU) functions. More specifically, we carefully upper bound both the convergence rate of the training error and the generalization error of such NNs and investigate the dependence of these bounds on the Leaky ReLU parameter, α. We show that $\alpha = -1$, which corresponds to the absolute value activation function, is optimal for the training error bound. Furthermore, in special settings, it is also optimal for the generalization error bound. Numerical experiments empirically support the practical choices guided by the theory.

This chapter is organized as follows: §2.1 provides background on the problem of interest for this chapter, reviews related work, and highlights our main contributions; §2.2 details the assumed setup of the NNs and the training algorithms; §2.3 presents the main theorems; §2.4 describes our technical contributions and sketches the proof of the main theorems; §2.5 provides extensive numerical tests supporting our predictions from the theory on synthetic and real datasets; §2.6 concludes this chapter and discusses its limitations; and §A discusses the generalization error bound and includes details of the proof, details of the simulation and additional numerical results.

2.1 Introduction

Deep neural networks (DNNs) have demonstrated remarkable success in diverse fields, including image classification and text recognition. Despite their achievements, a comprehensive understanding of these networks remains elusive. Theoretical justifications for their performance have primarily centered around the overparameterized setting and mainly considered a rectified linear unit (ReLU). This chapter aims to extend and generalize insights gained from recent theoretical works to any Leaky ReLU and provide practical guidance on selecting the most suitable Leaky ReLU for overparameterized networks. By doing so, we offer valuable insights for practitioners seeking optimal performance in real-world scenarios.

To address our aim, we begin by reviewing two recent theoretical trends. The first centers around a fundamental convergence theory for the training error of overparameterized neural networks (NNs). Its pioneering work by Jacot et al. (2018) studied the training dynamics using the neural tangent kernel and showed that the training error goes to zero in the asymptotic regime where the width of the layers goes to infinity. A more reasonable regime assumes a sufficiently large lower bound on the width. In such overparameterized regime, (Goodfellow et al., 2015) empirically noticed that the corresponding NNs can avoid local minima and converge to their global optimal solutions. (Du et al., 2019) proved the convergence of gradient descent (GD) for NNs with smooth and Lipschitz continuous activation functions whose width exponentially depends on the depth of the networks and polynomially depends on the number of samples. For 2-layer NNs with a ReLU, Li and Liang (2018) proved the convergence of the training error, Oymak and Soltanolkotabi (2020) reduced the width requirement for training convergence, and Song et al. (2021) established convergence whenever the width sub-quadratically depends on the number of samples and the activation functions are sufficiently smooth.

For DNNs, it has become common to consider the polynomial regime of overparameterization, where the NN widths polynomially depend both on the numbers of samples and the depths. Allen-Zhu et al. (2019b) established the first convergence result for the training error in this polynomial regime, while assuming ReLU activation functions. They separately analyzed training by gradient descent and stochastic gradient descent (SGD). Zou and Gu (2019) improved the estimates of Allen-Zhu et al. (2019b) by enhancing the lower bound of the gradient. Chen et al. (2019) further improved the polynomial dependence of the width on the number of samples that was established in Zou and Gu (2019), but on the other hand, their polynomial dependence on the depth is worse. Banerjee et al. (2023) showed that for smooth activation functions a linear dependence of the width on the number of samples is sufficient to guarantee convergence.

Another recent progress involves bounding the generalization error of overparameterized NNs. Chizat and Bach (2020) established a generalization bound of infinitely wide two-layer NNs with homogeneous activation functions for classification and showed that the probability of the misclassification bound goes to 0 as the size of the training samples increases. Arora et al. (2019) bounded the generalization error of 2-layer overparameterized NNs for classification. They also analyzed the class of functions that are learnable by two-layer NNs. Allen-Zhu et al. (2019a) studied the generalization error of two-layer and three-layer NN with a non-negative, convex, and 1-Lipschitz smooth loss function using stochastic gradient descent. They showed that overparameterization improves generalization. Cao and Gu (2020) further established the generalization error of deep NNs for classification using gradient descent. Zhu et al. (2022) extended the latter work for classification by using some other activation functions, including leaky ReLU with $\alpha \in (0, 1)$.

However, these foundational and important works have not yet provided much practical guidance for designing NNs. Practitioners often use variants of ReLU for

activation and this work aims to provide guidance on their choices. Leaky ReLU is widely used in DNNs for supervised learning tasks (Redmon et al., 2016; Ridnik et al., 2021) and for generative tasks (Radford et al., 2015; Chen et al., 2016; Karras et al., 2019; Wang et al., 2021). It is represented by the function $\sigma_\alpha(x)$, where $\sigma_\alpha(x) = x$ for $x > 0$ and $\sigma_\alpha(x) = \alpha x$ for $x \leq 0$, with α being a parameter. ReLU is a special case of Leaky ReLU when $\alpha = 0$. The Leaky ReLU function aims to prevent zero gradients for negative inputs, thus avoiding neurons from not activating. Empirical studies have demonstrated the advantage of using Leaky ReLU with small $\alpha > 0$ over ReLU (Xu et al., 2015). However, theoretical studies have primarily focused on ReLU and have not directly established the convergence theory and generalization for regression when using Leaky ReLU with any $\alpha < 1$. Moreover, the optimal choice of the Leaky ReLU parameter α to expedite the training process and enhance generalization remains unclear. Therefore, a theoretical study is needed to analyze the efficacy of leaky ReLU during training and to provide guidance on selecting the parameter α in practice.

This work studies overparameterized DNNs with a wide class of leaky ReLU activation functions and develops theories for the convergence of the training error and the upper bound of the generalization error. It builds on the proof framework and techniques introduced in previous studies, in particular, the ones of Allen-Zhu et al. (2019b), Zou and Gu (2019) and Cao and Gu (2020), but establishes the dependence of the convergence rate and the generalization error on the leaky ReLU parameter α. It reveals that the optimal convergence rate bound is achieved at $\alpha = -1$ and the optimal bound of the generalization error is achieved at $\alpha = -1$ using small training epochs as long as the NN is sufficiently deep and the dataset is sufficiently large. This means that activation by the absolute value function may outperform activation by ReLU and the commonly used leaky ReLU (with small $\alpha > 0$) in terms of faster training convergence and smaller generalization error. We are not aware of any prior use of the absolute value function for activating DNNs. We are only aware of using

it for activating the scattering network (Mallat, 2012) due to its help with "energy preservation" (Bruna and Mallat, 2013).

The main contributions of the current work are as follows:

1. We establish the convergence of the training errors in overparameterized NNs with any leaky ReLU using both GD and SGD. Our estimates clarify the effect of the Leaky ReLU parameter α on the convergence rate bound. In particular, $\alpha = -1$, yields the optimal bound.

2. We upper bound the generalization error for overparameterized NNs for regression with leaky ReLUs. For sufficiently large datasets, deep NNs and small training epochs, the bound is optimal at $\alpha = -1$.

3. We improve previous results for ReLU (see §2.4.2). In particular, we show that deep NNs achieve a similar convergence rate as a shallow NN.

4. Our predictions receive substantial support from a comprehensive set of numerical experiments

2.2 Problem Setup

We follow the model of Allen-Zhu et al. (2019b), while allowing a wide class of Leaky ReLU activation functions. We consider a dataset $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^p$, $\|\boldsymbol{x}_i\| = 1$, $\boldsymbol{y}_i \in \mathbb{R}^d$, $\|\boldsymbol{y}_i\| \leq O(1)$ and $d < O(1)$. We focus on a NN $\mathcal{N} : \mathbb{R}^p \to \mathbb{R}^d$ with L hidden layers having m neurons each and linear input and output layers. Its input layer produces $\boldsymbol{h}_0 = \boldsymbol{A}\boldsymbol{x}$, where , $\boldsymbol{A} \in \mathbb{R}^{m \times p}$. For $l \in [L] := \{1, 2, \ldots L\}$, the output of the lth hidden layer, $\boldsymbol{h}_l$, is inductively defined by

$$\boldsymbol{h}_l = \mathcal{H}_l(\boldsymbol{h}_{l-1}) = \sigma_\alpha(\boldsymbol{W}_l \boldsymbol{h}_{l-1}), \tag{2.1}$$

where $\boldsymbol{W}_l \in \mathbb{R}^{m\times m}$ and σ_α is the leaky ReLU activation function with $\alpha < 1$:

$$\sigma_\alpha(x) = \begin{cases} x, & \text{if } x \geq 0; \\ \alpha x, & \text{if } x < 0. \end{cases}$$

The output layer produces $\hat{\boldsymbol{y}} = \boldsymbol{B}\boldsymbol{h}_L$, where $\boldsymbol{B} \in \mathbb{R}^{d\times m}$. Let $\boldsymbol{W} := (\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots \boldsymbol{W}_L)$ store all the trainable parameters and we thus compactly denote $\hat{\boldsymbol{y}} = \mathcal{N}(\boldsymbol{x}; \boldsymbol{W})$. For simplicity, we initialize $\boldsymbol{A}$ and $\boldsymbol{B}$ (see below), so they are fixed, and only train $\boldsymbol{W}_l,\ l \in [L]$.

We train the NN using the mean squared error (MSE): $\mathcal{L}(\boldsymbol{W}) = \sum_{i=1}^n \|\boldsymbol{y}_i - \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{W})\|^2$. We denote its gradient by $\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W}) := (\nabla_{\boldsymbol{W}_1}\mathcal{L}(\boldsymbol{W}), \ldots \nabla_{\boldsymbol{W}_L}\mathcal{L}(\boldsymbol{W}))$. Appendix A.2.12 extends our theory to many other useful loss functions. We assume a specified upper bound $\epsilon > 0$ on the training error and express our estimates in terms of ϵ.

When discussing generalization, we assume that the set $\{\boldsymbol{x}_i\}_{i=1}^n$ is i.i.d. drawn from an arbitrary distribution $\mathcal{D}_{\boldsymbol{X}}$ and that for $1 \leq i \leq n$, $\boldsymbol{y}_i = F(\boldsymbol{x}_i)$ for an arbitrary measurable function F. The generalization error is thus $R(\boldsymbol{W}) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{\boldsymbol{X}}}\|F(\boldsymbol{x}) - \mathcal{N}(\boldsymbol{x}; \boldsymbol{W})\|^2$.

We assume the following data separation property:

Assumption 2.1

There exists $0 < \delta < c_0$, where $c_0 < 1$, so that $\min_{i,j\in[n]} \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \geq \delta > 0$. □

This assumption, suggested by Allen-Zhu et al. (2019b), is reasonable. Indeed, if, on the other hand, there exists $i \neq j \in [n]$ such that $\boldsymbol{x}_i = \boldsymbol{x}_j$, then we can assume $\boldsymbol{y}_i \neq \boldsymbol{y}_i$ (otherwise we can combine these multiple instances into one single data point). It is then impossible to obtain a zero training error, which is needed for our convergence study.

Following He et al. (2015), we initialize the network parameters as follows: $\boldsymbol{A} \sim$

Algorithm 1 Rescaled initialization

Input: Input dimension p, width of hidden layer m, output dimension d, and leaky ReLU parameter α.

Initialize:

$$\boldsymbol{A} \sim N\left(0, \frac{1}{m}\right), \ \boldsymbol{B} \sim N\left(0, \frac{1}{d}\right), \ \boldsymbol{W}_l^{(0)} \sim N\left(0, \frac{2}{m}\right), \ l \in [L]$$

Activation function:

$$\tilde{\sigma}_\alpha(x) = \begin{cases} \frac{1}{\sqrt{1+\alpha^2}} x, & \text{if } x \geq 0 \\ \frac{\alpha}{\sqrt{1+\alpha^2}} x, & \text{if } x < 0 \end{cases} \tag{2.2}$$

$N(0, 1/m)$, $\boldsymbol{B} \sim N(0, 1/d)$ and $\boldsymbol{W}_l^{(0)} \sim N(0, 2/(m(1+\alpha^2)))$ for $l \in [L]$. Note that the factor $1/(1+\alpha^2)$ ensures a constant variance for any choice of α. We can move the factor $1/(1+\alpha^2)$ from the weight initialization to the activation function, and equivalently initialize with Algorithm 1. The theoretical study of the latter formulation with its rescaled Leaky ReLU function, $\tilde{\sigma}_\alpha(x)$ (see (2.2)), turns out to be more tractable.

Algorithms 2 and 3 formulate the training procedures with simple GD and SGD, respectively.

Algorithm 2 Training (gradient descent)

Input: Learning rate η.

Initialize: Apply Algorithm 1 to obtain $\boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{W}^{(0)}$

for $t = 0$ **to** T **do**

$$\boldsymbol{W}^{(t+1)} = \boldsymbol{W}^{(t)} - \eta \nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}^{(t)}).$$

end for

Algorithm 3 Training (stochastic gradient descent)

Input Learning rate η.
Initialize: Apply Algorithm 1 to obtain $\boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{W}^{(0)}$
for $t = 0$ **to** T **do**
Randomly select batch $B \subset [n]$ with $|B| = b$.

$$\boldsymbol{W}^{(t+1)} = \boldsymbol{W}^{(t)} - \eta \nabla_{\boldsymbol{W}} \mathcal{L}_B(\boldsymbol{W}^{(t)}),$$

where $\mathcal{L}_B(\boldsymbol{W}^{(t)}) := \sum_{i \in B} \|\boldsymbol{y}_i - \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{W}^{(t)})\|^2$.
end for

2.3 Main Results

The two theorems below establish the convergence of the training error for overparameterized NNs using a Leaky ReLU function with $\alpha < 1$. The first theorem pertains to training with gradient descent (GD) (Algorithm 2), while the second applies to training with stochastic gradient descent (SGD) (Algorithm 3). Both theorems are formulated within the context outlined in §2.2. This setup includes Assumption 2.1 with a parameter δ, Algorithm 1 for the initialization of the parameters of the NN, n training points, $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^n$, where $\|\boldsymbol{x}_i\| = 1$, and $\|\boldsymbol{y}_i\| \leq O(1)$, output dimension d ($\boldsymbol{y}_i \in \mathbb{R}^d$), NN depth L, NN width m, Leaky ReLU parameter α, learning rate η, batch size b (for Algorithm 3) and a desired upper bound $\epsilon > 0$ on the training error.

Theorem 2.1

Assume the setup of §2.2, where both $m/\ln^4 m > \frac{1+\alpha^2}{(1-\alpha)^2}\Omega(\frac{n^5 L^{15} d}{\delta^4})$ and $m > \Omega\left(\ln \ln \epsilon^{-1}\right)$, and the training is according to Algorithm 2 with learning rate $\eta \leq O(\frac{d}{nL^2 m})$. Then, with probability at least $1 - e^{-\Omega(\ln m)}$,

$$\mathcal{L}(\boldsymbol{W}^{(T)}) < \epsilon \text{ and } \mathcal{L}(\boldsymbol{W}^{(t)}) \leq \gamma^t \mathcal{L}(\boldsymbol{W}^{(0)}), \forall t \leq T, \tag{2.3}$$

where

$$\gamma = 1 - \Omega\left(\frac{(1-\alpha)^2}{1+\alpha^2}\frac{\eta\delta m}{nd}\right),\ T = \frac{\ln\left(\epsilon/\mathcal{L}(\boldsymbol{W}^{(0)})\right)}{\ln\gamma}. \tag{2.4}$$

□

Theorem 2.2

Assume the setup of §2.2, where both $\frac{m}{\ln^4 m} > \frac{(1+\alpha^2)^4}{(1-\alpha)^8}\Omega(\frac{n^8L^{15}d}{b\delta^5})$ and $m\ln m > \Omega\left(\ln\ln\epsilon^{-1}\right)$ and the NN is trained according to Algorithm 3 with $\eta \leq O(\frac{d\delta}{mn^3L^3\ln^2 m})$ and $t > \frac{(1+\alpha^2)^2}{(1-\alpha)^4}\Omega(\frac{n^5L^2}{b\delta^2}\ln^2 m)$. There exists a constant $C_0 > 1$ such that

$$\begin{aligned}&\mathcal{L}(\boldsymbol{W}^{(T)}) < \epsilon \text{ and } \mathcal{L}(\boldsymbol{W}^{(t)}) \leq C_0\gamma^t\mathcal{L}(\boldsymbol{W}^{(0)})\\ &\quad\text{for all } t \leq T \text{ with probability } 1 - e^{-\Omega(\ln m)},\end{aligned} \tag{2.5}$$

where

$$\gamma = 1 - \Omega\left(\frac{(1-\alpha)^2}{1+\alpha^2}\frac{\eta b\delta m}{n^2 d}\right),\ T = \frac{\ln\left(\epsilon/C_0\mathcal{L}(\boldsymbol{W}^{(0)})\right)}{\ln\gamma}. \tag{2.6}$$

□

These theorems show that for any $\alpha < 1$ the training error linearly converges to zero when the NN width is sufficiently large and the learning rate η is sufficiently small.

Moreover, these theorems reveal the dependence of the convergence rate bound on α and this information can guide one in selecting α for optimal training speed. We note that the typical choice of the leaky ReLU parameter α (e.g., 0.01 or 0.05) does not yield a better bound for the convergence speed than ReLU (i.e., $\alpha = 0$); furthermore, the negative values of α yield better results than ReLU and the optimal choice of α is -1. In §2.4.1 we interpret this optimality in terms of obtaining the largest derivative gap of the rescaled leaky ReLU at 0. We mathematically formulate the above observation as follows:

Corollary 2.1

Assume the setup of §2.2 with either Algorithms 2 or 3 and that all parameters are chosen so that when $\alpha = 0$, $\gamma < 1$. Then $\alpha = -1$ minimizes the above convergence rate γ among all $\alpha < 1$. Moreover, γ is decreasing in α on $(-\infty, -1)$ and increasing on $(-1, 1)$. □

For $\alpha = 0$, our result improves the previous analysis of both Allen-Zhu et al. (2019b) and Zou and Gu (2019). We compare our bounds with the ones of Zou and Gu (2019), since they improved the bounds of Allen-Zhu et al. (2019b). For this purpose, we examine the difference in the setups. First, Zou and Gu (2019) divides the loss function $\mathcal{L}(\boldsymbol{W})$ by n and thus we need to convert their estimate by a factor of a power of n accordingly. Second, our proof assumes that the hidden signals are separated by $\delta < O(1)$, whereas Zou and Gu (2019) assumes that $\delta < O(1/L)$. We establish this upper bound independently of L with careful mathematical estimates; therefore, our setup eliminates implicit dependence on L in the other formulas. At last, Zou and Gu (2019) enforces the initial scaled loss to be bounded by $O(1)$ (this amounts to a bound $O(n)$ on our loss) and their conclusion holds with probability at least $1 - \Omega(1/n)$. On the other hand, we relax the initial unscaled loss to be bounded by $O(\sqrt{\ln m})$ and our conclusion holds with probability at least $1 - e^{-\Omega(lnm)}$, which we find more natural for the overparameterized regime.

After converting to our setup, the convergence rate in Zou and Gu (2019) is $1 - \Omega(\eta\delta m/(dnL))$ when using gradient descent, and our convergence rate improves to $1 - \Omega(\eta\delta m/(dn))$; also, when using SGD the convergence rate in Zou and Gu (2019) is $1 - \Omega(\eta\delta mb/(dn^2L))$ and we improve it to $1 - \Omega(\eta\delta mb/(dn^2))$. The important finding is that in the overparameterized regime, a deeper NN does not lead to slower convergence, but rather achieves a similar convergence rate as a shallow NN. One can further note that we improved the bound of Zou and Gu (2019) on m by the factor $n^{-3}L^{-1}$ for GD and $n^{-8}L^{-2}(n/b)^{-3}\delta^3$ for SGD. Furthermore, our lower bound on the

number of epochs t in Theorem 2.2 improves the one of Allen-Zhu et al. (2019b) by a factor of order $n^{-2}L^{-2}$, where there is no explicit bound in Zou and Gu (2019).

Appendix A.2.12 extends the above bounds to convex loss functions, which include the cross-entropy for classification and a special loss function proposed in Kumar et al. (2023). The convergence rate for these functions is different, but $\alpha = -1$ is still optimal for their bounds.

Next, we establish an upper bound of the generalization error of a NN trained using GD, where an analogous bound when using SGD is specified in Theorem A.1 in Appendix A.2.10. We first follow the previous analysis of generalization in overparameterized NNs by Cao and Gu (2020) and establish the corresponding bound for our setting with Leaky ReLU activation function.

Theorem 2.3

Assume the setup of §2.2 with GD, where $m = \Theta(\frac{n^{10+2\tau}L^{15+2\tau}d^{1+2\tau}}{\delta^{4-2\tau}})$ for $\tau > 0$ and $\eta = \Theta(\frac{d}{nL^2m})$. Assume further that m is larger than its lower bound and η is smaller than its upper bound in Theorem 2.1 (by an appropriate choice of the hidden constants in Θ and compared to the constants hidden in the lower bound of m and in the upper bound of η in Theorem 2.1). Then at a given training epoch $t \leq T$ (see (2.4) for T),

with probability at least $1 - e^{-\Omega(\ln m)}$, the generalization error is bounded as follows

$$R(\boldsymbol{W}^{(t)}) \leq \gamma^t \mathcal{L}(\boldsymbol{W}^{(0)})$$

$$+ \min\left\{ O\left(\frac{d^{3/2+\tau}\delta^\tau n^{1/2+\tau}}{L^{1/2-\tau}\ln m}\right), O\left(\frac{1-\alpha}{\sqrt{1+\alpha^2}}\frac{d^{1/3}t^{4/3}}{m^{1/6}n^{2/3}L^{2/3}}\right)\right\}$$

$$+ \min\left\{ O\left(\frac{\sqrt{d\ln m}\; t}{nL}\right), O\left(\frac{n^{1/2+\tau}L^{2+\tau}d^{1/2+\tau}}{\delta^{1/2-\tau}\ln m}\right)\right\} + O\left(d\sqrt{\frac{\ln m}{n}}\right). \quad (2.7)$$

□

In Appendix A.1, we clarify the above estimates for different regimes for the number of training epochs, t. In particular, we indicate a tradeoff between the first training term and the other NN-complexity terms (excluding the last term of data complexity) and show that we cannot make both of these kinds of terms sufficiently small. Stopping at a sufficiently small number of epochs results in a bound of the generalization error of order $O(\ln(m))$, which is also of order $O(\ln(n))$. This bound is composed of several terms. The term which contributes $O(\ln(m))$ is due to the training error and one cannot expect a better bound for it when having a small number of epochs. The rest of the terms do converge when n and L are sufficiently large and in this latter regime the overall bound is minimized when $\alpha = -1$. On the other hand, for larger numbers of epochs overfitting is observed, which results in divergent generalization error. Exploring the dependence of the generalization bound on t is advantageous to an epoch-independent bound, like the one pursued by Cao and Gu (2020) for classification instead of regression. Indeed, the bound of Cao and Gu (2020) is $\Theta(\text{poly}(n) \cdot n^{-1/2})$, which is significantly larger than $O(\log(n))$.

For very special datasets (e.g., single-layer ReLU NN separability) Cao and Gu

(2020) reduced the term poly(n) so their overall bound is sufficiently small. A natural, but more complicated, extension of this to regression is to consider datasets well-approximated by L-layer leaky ReLU NNs. In Appendix A.2.11, we improve the convergence rate, the lower bound of m (so its dependence on n is linear) and the generalization error bound for such datasets. However, for a large number of epochs we still notice overfitting with divergent generalization error (with a smaller rate of increase to infinity than for general datasets).

At last, Kumar et al. (2023) claimed that when using the loss function discussed in (A.122) of Appendix A.2.12, minimizing a particular generalization error bound is equivalent to minimizing the latter loss function for training. Therefore, if $\alpha = -1$ is optimal for the training error, then it is also optimal for the generalization error bound. Since we verified the optimality of $\alpha = -1$ for our upper bound of the convergence rate in Appendix A.2.12 and experimentally demonstrated instances where this bound is comparable to the actual convergence rate in Figure 2.2, we get some numerical evidence that for the latter instances $\alpha = -1$ is optimal for bounding the generalization error.

2.4 Ideas of Proofs

Our proofs follow ideas of Allen-Zhu et al. (2019b), Zou and Gu (2019) and Cao and Gu (2020) and adapt them to the general case of Leaky ReLU with $\alpha < 1$. It also adapts Cao and Gu (2020) to regression. We first sketch in §2.4.1 the basic ideas of our proofs, while we supplement all details in the appendix. We then highlight some of the innovative ideas in §2.4.2.

2.4.1 Sketch of Proofs

We describe here a quick roadmap to verifying the theory. The proofs of Theorems 2.1 and 2.2 follow the initial framework of Allen-Zhu et al. (2019b), which was later followed by Zou and Gu (2019), but consider the effect of using any leaky RELU with $\alpha < 1$.

These proofs use the following two lemmas, which are proved in §A.2.5 and §A.2.4. Let us first clarify their notation. We denote by $\|\boldsymbol{X}\|_2$ and $\|\boldsymbol{X}\|$ the spectral and Frobenius norms of a matrix $\boldsymbol{X}$. For $\boldsymbol{W} = (\boldsymbol{W}_1 \dots \boldsymbol{W}_L)$ and $\boldsymbol{V} = (\boldsymbol{V}_1 \dots \boldsymbol{V}_L)$, we define $\|(\boldsymbol{W}_1 \dots \boldsymbol{W}_L)\|_F^2 := \sum_{l\in[L]} \|\boldsymbol{W}_l\|_F^2$, $\|(\boldsymbol{W}_1 \dots \boldsymbol{W}_L)\|_2 := \max_{l\in[L]} \|\boldsymbol{W}_l\|_2$ and $\langle \boldsymbol{W}, \boldsymbol{V}\rangle := \sum_{l\in[L]} \langle \boldsymbol{W}_l, \boldsymbol{V}_l\rangle$. We denote by $\boldsymbol{W}'$ a perturbation of $\boldsymbol{W}$.

Lemma 2.1 (Semi-smoothness)
Assume the setup of §2.2. If $\|\boldsymbol{W} - \boldsymbol{W}^{(0)}\|_2 < \omega < O\left(\frac{1}{L^{9/2}\ln^{3/2} m}\right)$ and $\|\boldsymbol{W}'\|_2 < \omega$, then with a probability at least $1 - e^{-\Omega(m)}$

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{W} + \boldsymbol{W}') \le \mathcal{L}(\boldsymbol{W}) + \langle \nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}), \boldsymbol{W}'\rangle \\
&+ \frac{nL^2 m}{d} O(\|\boldsymbol{W}'\|_2^2) \\
&+ \frac{(1-\alpha)\omega^{1/3} L^2 \sqrt{mn\mathcal{L}(\boldsymbol{W}) \ln m}}{\sqrt{d(1+\alpha^2)}} O(\|\boldsymbol{W}'\|_2).
\end{aligned} \tag{2.8}$$

□

Lemma 2.2 (Gradient bounds)
Assume the setup of §2.2. If $\|\boldsymbol{W} - \boldsymbol{W}^{(0)}\|_2 < \omega < O\left(\frac{\delta^{3/2}}{n^{3/2} L^{15/2} \ln^{3/2} m}\right)$, then with a

probability at least $1 - e^{-\Omega(m\delta^2/L^3)}$

$$\|\nabla_{\boldsymbol{W}_l}\mathcal{L}(\boldsymbol{W})\|_F^2 \leq \mathcal{L}(\boldsymbol{W})O\left(\frac{mn}{d}\right), \quad \text{for } \ l \in [L] \tag{2.9}$$

$$\|\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W})\|_F^2 \geq \mathcal{L}(\boldsymbol{W})\Omega\left(\frac{(1-\alpha)^2}{(1+\alpha^2)}\frac{\delta m}{nd}\right). \tag{2.10}$$

□

We note that the factor $(1-\alpha)/\sqrt{1+\alpha^2}$ appears in both (2.8) and (2.10), where it is squared in (2.10). This factor is the derivative gap in Leaky ReLU, i.e., $\tilde{\sigma}'_\alpha(0+) - \tilde{\sigma}'_\alpha(0-)$, which can be viewed as a measure of nonlinearity. Its value is larger for Leaky ReLU with $\alpha < 0$ than for ReLU (with $\alpha = 0$) and maximized at $\alpha = -1$. Our analysis below, which combines the bounds in (2.8) and (2.10), shows that Leaky ReLU with $\alpha < 0$ leads to better control of the decay of the loss function than ReLU and that the best control is at $\alpha = -1$.

Theorem 2.1 can be proved as follows. Let $\boldsymbol{W} := \boldsymbol{W}^{(t)}$ and $\boldsymbol{W}' := -\eta\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W}^{(t)})$ and note that by gradient descent, $\boldsymbol{W} + \boldsymbol{W}' = \boldsymbol{W}^{(t+1)}$. Denoting $\mathcal{L}^{(t)} := \mathcal{L}(\boldsymbol{W}^{(t)})$ and applying (2.8) of Lemma 2.1, we can conclude that with a probability of at least $1 - e^{-\Omega(m)}$, the following inequality holds

$$\mathcal{L}^{(t+1)} \leq \mathcal{L}^{(t)} - \eta\langle\nabla_{\boldsymbol{W}}\mathcal{L}^{(t)}, \nabla_{\boldsymbol{W}}\mathcal{L}^{(t)}\rangle \tag{2.11}$$

$$+ \frac{\eta(1-\alpha)\omega^{\frac{1}{3}}L^2\sqrt{mn\mathcal{L}^{(t)}\ln m}}{\sqrt{d(1+\alpha^2)}}O\left(\|\nabla_{\boldsymbol{W}}\mathcal{L}^{(t)}\|_2\right) \tag{2.12}$$

$$+ \frac{\eta^2 nL^2 m}{d}O\left(\|\nabla_{\boldsymbol{W}}\mathcal{L}^{(t)}\|_2^2\right). \tag{2.13}$$

Using (2.10) we bound $\sqrt{\mathcal{L}^{(t)}}$ as follows with probability at least $1 - e^{-\Omega(m\delta^2/L^3)}$:

$$\sqrt{\mathcal{L}^{(t)}} \leq \frac{\sqrt{1+\alpha^2}}{1-\alpha}O\left(\sqrt{\frac{nd}{\delta m}}\right)\|\nabla_{\boldsymbol{W}}\mathcal{L}^{(t)}\|_F. \tag{2.14}$$

Applying (2.14), we control the term in (2.12), with probability at least $1 - e^{-\Omega(m\delta^2/L^3)}$,

by

$$\eta\omega^{1/3}nL^2\left(\sqrt{\ln m}/\sqrt{\delta}\right)O\left(\|\nabla_{\boldsymbol{W}}\mathcal{L}^{(t)}\|_F^2\right). \tag{2.15}$$

Using $\omega < O(\frac{\delta^{3/2}}{n^{3/2}L^{15/2}\ln^{3/2}m})$, which is required by Lemma 2.2, we reduce (2.15) to $\eta\|\nabla_{\boldsymbol{W}}\mathcal{L}^{(t)}\|_F^2/3$. Using $\eta < O(d/(nL^2m))$, which is required in Theorem 2.1, we reduce the bound in (2.13) to $\eta\|\nabla_{\boldsymbol{W}}\mathcal{L}^{(t)}\|_F^2/3$.

Next, we apply these bounds to the respective terms in (2.11) and use the identity $\langle \boldsymbol{X}, \boldsymbol{X}\rangle = \|\boldsymbol{X}\|_F^2$ for a vector of matrices $\boldsymbol{X} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_L)$ to reduce (2.11) to

$$\mathcal{L}^{(t+1)} \leq \mathcal{L}^{(t)} - 1/3\eta\|\nabla_{\boldsymbol{W}}\mathcal{L}^{(t)}\|_F^2. \tag{2.16}$$

Further application of the lower bound in (2.10) to the above equation results in $\mathcal{L}^{(t+1)} \leq \gamma\mathcal{L}^{(t)}$ with γ specified in (2.4) and we consequently conclude (2.3) of Theorem 2.1.

The above argument holds for one training step with probability at least $1 - e^{-\Omega(m)}$. This argument extends to T steps with probability at least $1 - Te^{-\Omega(m)}$. We note that the number of epochs T can be bounded using the bound ϵ on the training error, the convergence rate in (2.4) and the estimate $\mathcal{L}(\boldsymbol{W}^{(0)}) \leq O(n\sqrt{\ln m})$, which is shown in Appendix A.2.6, as follows:

$$\begin{aligned} T &= \ln(\epsilon/\mathcal{L}(\boldsymbol{W}^{(0)}))/\ln\gamma \leq \Theta(\ln(\epsilon/n\sqrt{\ln m})/\ln\gamma) \\ &\leq O\left(\frac{nd}{\eta\delta m}(\ln\epsilon^{-1} + \ln(n\sqrt{\ln m}))\right). \end{aligned}$$

Thus the total probability to ensure T-steps training with training error lower than ϵ is at least $1 - O(\frac{nd}{\eta\delta m}(\ln\epsilon^{-1} + \ln(n\sqrt{\ln m})))e^{-\Omega(m)}$. Given that $m > \Omega(\text{poly}(n, L, d, \delta^{-1}))$ and $m > \Omega(\ln\ln\epsilon^{-1})$, this probability is of order $1 - e^{-\Omega(m)}$.

In Appendix A.2.6, we demonstrate that the inequality $\|\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(0)}\|_2 < \omega <$

$O(\delta^{3/2}/(n^{3/2}L^{15/2}\ln^{3/2} m))$ holds with probability at least $1 - e^{-\Omega(\ln m)}$. Note that the latter bound implies the conditions for both Lemmas 2.1 and 2.2 and thus concludes the proof of Theorem 2.1

The proof of Theorem 2.2 is detailed in §A.2.7. We briefly describe the proof idea as follows. First, we use a similar argument as in the proof of Theorem 2.1 to bound the expectations of the loss functions at each step. Second, we use (2.9) to find an absolute upper bound of the loss functions. By combining these two bounds and using Azuma's inequality, we derive the decay of the loss function in (2.5) with the convergence rate in (2.6) in Theorem 2.2. Finally, we verify that the conditions for Lemma 2.1 and Lemma 2.2 are satisfied when the NN width satisfies $m/\ln^4 m > (1+\alpha^2)^4/(1-\alpha)^8\Omega(n^8L^{15}d/(b\delta^5))$ and thus conclude the theorem.

The proof of Theorem 2.3, which appears in §A.2.9, relies on the following lemma that bounds the generalization error for a class of NNs whose parameters are close to $\boldsymbol{W}^{(0)}$.

Lemma 2.3 (Generalization error with perturbation)
Assume the setup of §2.2, where α is the leaky ReLU parameter. If $\|\boldsymbol{W} - \boldsymbol{W}^{(0)}\| < \omega < O\left(\frac{\delta^{3/2}}{n^{3/2}L^{15/2}\ln^{3/2} m}\right)$, then with probability at least $1 - e^{-\Omega(\ln m)}$

$$\begin{aligned} R(\boldsymbol{W}) \leq \frac{1}{n}\mathcal{L}(\boldsymbol{W}) + \frac{1-\alpha}{\sqrt{1+\alpha^2}}O(d(\ln m)\sqrt{m}L^2\omega^{4/3}) + \\ O(d\sqrt{m(\ln m)/n}L\omega) + O\left(d\sqrt{\ln m/n}\right). \end{aligned} \qquad \square$$

The proof of Lemma 2.3, which appears in Appendix A.2.8, follows similar ideas as those of Cao and Gu (2020) but adapted to the different task of regression. Theorem 2.3 is a consequence of this lemma and two different estimates of the size of ω during training. The first estimate controls ω during the entire training with GD, regardless of how large the training epoch is, and is expressed in Lemma A.9. The second estimate uses direct bounds of the learning steps and provides a better upper bound of ω when

the training epoch is small.

2.4.2 Discussion of Innovation

While we followed, extended and improved an existing proof framework, we would like to emphasize some innovation in our proof techniques. To begin with, it is difficult to directly extend the previous methods to any leaky ReLU with $\alpha < 1$. Our idea of rescaling the leaky ReLU activation function, along with the observation that, with rescaled initialization, it is equivalent to using the unscaled leaky ReLU, helped tremendously simplify our initial technical and complex effort. This allowed us to elegantly use the previous ideas and further improve them. Additional technical steps that are required to address the case $\alpha \neq 0$ can be noticed in the proofs of Lemmas A.1, A.2, A.4 and A.7.

We have also made notable improvements to previous estimates. In particular, we improved the lower bound for the gradient established by Zou and Gu (2019) by a factor of L. We also eliminated the previous dependence of the convergence rate on a negative power of L, which was undesirable as it implied that deeper networks might experience slower convergence. This demonstrates that the convergence rate of deep neural networks is at least comparable to that of shallow neural networks. Specifically, the later estimates can be found in the proof of Lemma 2.2 in Appendix A.2.4. They are motivated by a suggestion from Allen-Zhu et al. (2019b) to incorporate gradients from all layers' parameters, departing from previous estimates that solely relied on the gradients of parameters from the last layer. More specifically, improved lower bounds for the gradients from all layers' parameters can be found in Lemma A.7 in Appendix A.2.4. We also obtained a tighter bound for the spectral norm of $\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(0)}$ when using SGD. This improved the lower bound on the width m for training convergence by a factor of order $n^{-8}L^{-2}(n/b)^{-3}\delta^3$.

Additionally, a more careful and fresh look helped improve the interpretation

of the results. In particular, noting the effect on the number of epochs t on the generalization error, while developing tighter bounds when t was sufficiently small, helped with a meaningful bound on the generalization error. Another example includes making all the probabilities dependent on m, a choice we deemed more suitable for the overparameterized regime. Furthermore, to avoid the hidden dependence of δ on L in the previous works, we had to develop some careful mathematical estimates (see (A.12) in the appendix), so we could explicitly identify the dependence on L and relax the previous assumption $\delta < O(1/L)$ to $\delta < O(1)$.

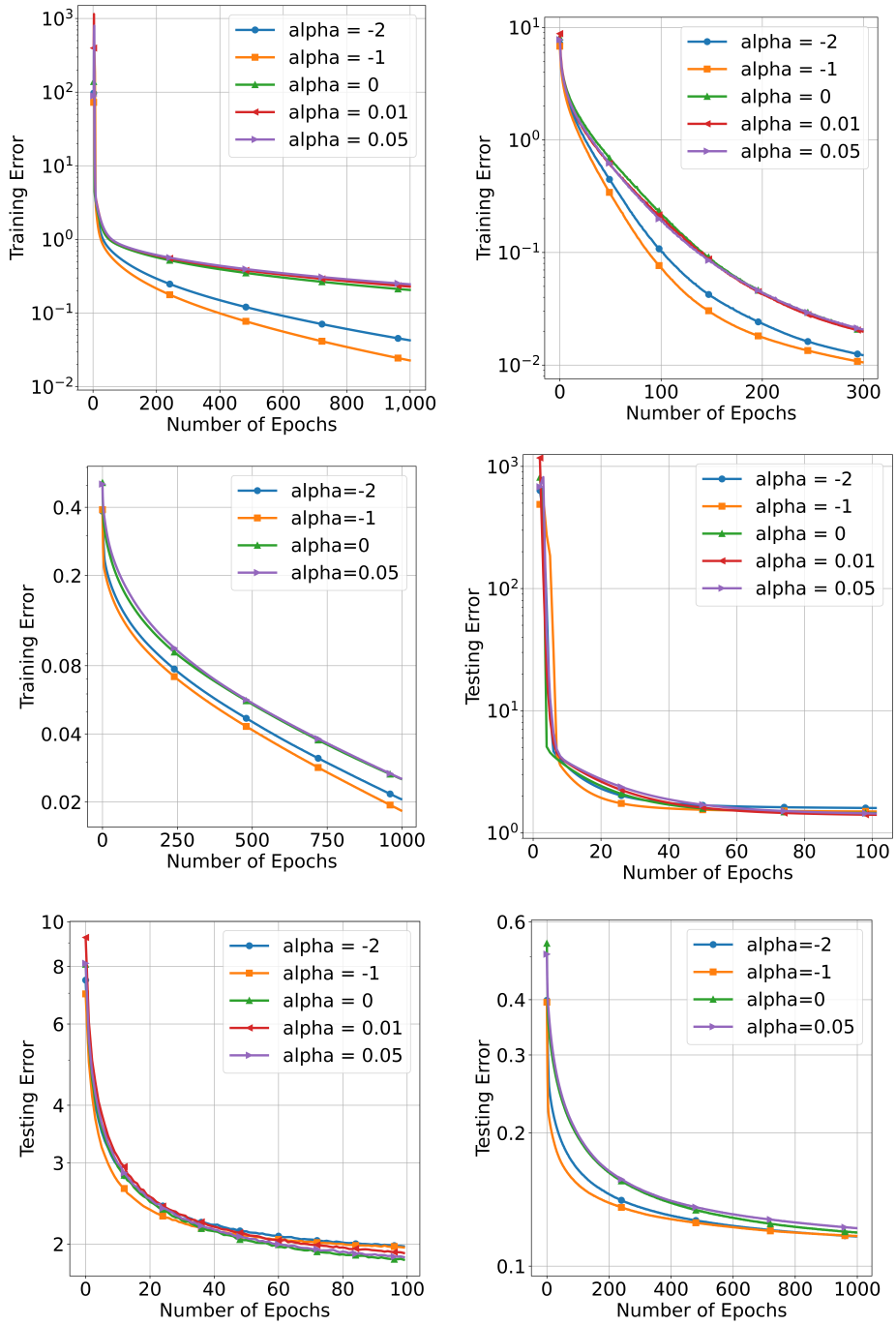

Figure 2.1: Log-scale training and testing errors using different datasets and different α's. From left to right: synthetic dataset, F-MNIST and CIFAR-10. Top row: training errors. Bottom row: testing errors.

2.5 Numerical Experiments

As our theory deals only with upper bounds, we conduct numerical experiments to examine the dependence of the actual training convergence rate and generalization error, particularly at an early epoch, on the parameter α. Our main goal is to determine whether $\alpha = -1$ is the optimal choice for convergence and generalization in overparameterized NNs with LeakyReLU activation functions. Appendix A.3 provides additional experiments.

2.5.1 Setup

We summarize our implementation for the following datasets. We provide additional details in §A.3.1.

Synthetic dataset: We simulate a dataset which contains 1,000 data points in $\mathbb{R}^5$ i.i.d. sampled from a normalized Gaussian distribution, $N(0, \boldsymbol{I}_5)$. We verified that Assumption 2.1 holds for the generated dataset with $\delta = 0.21$. We generate real-valued labels, y, by the following noisy nonlinear function of $\boldsymbol{x}$:

$$\begin{aligned} y &= \sin(10x_1 + 20x_2^3) + \cos(3x_3 + 5x_4^2) \\ &\quad + 2/(1 + \mathrm{ReLU}(0.05 + x_5))^{1/2} + 2x_1x_5 + \varepsilon, \end{aligned}$$

where $\varepsilon \sim N(0, 0.01)$. We construct NNs with five hidden layers, $m = 5,000$ and leaky ReLUs with $\alpha \in \{-2, -1, 0, 0.01, 0.05\}$. We initialize the NNs by Algorithm 1 and train them with GD using the MSE loss.

F-MNIST: This standard grayscale image classification benchmark consists of ten classes (Xiao et al., 2017). We build NNs with two hidden layers and width $m = 2,000$. We use leaky ReLUs with $\alpha \in \{-2, -1, 0, 0.01, 0.05\}$. We initialize the NNs by Algorithm 1 and train them using SGD with batch size 64 and the cross

entropy loss.

Table 2.1: Training and testing errors for the three main datasets. The first three rows report the training error at the last epoch. The next ones report the testing error at an early epoch ($t = 30$ for synthetic, $t = 20$ for F-MNIST and $t = 200$ for CIFAR-10).

Metric	Dataset	$\alpha = -2$	$\alpha = -1$	$\alpha = 0$	$\alpha = 0.05$
Final training error	Synthetic	0.039±0.002	**0.022**±0.002	0.197±0.013	0.245±0.022
	F-MNIST	0.096±0.009	**0.076**±0.008	0.211±0.018	0.229±0.032
	CIFAR-10	0.019±0.001	**0.018**±0.001	0.024±0.001	0.027±0.001
Early Epoch testing error	Synthetic	1.914±0.067	**1.775**±0.065	2.086±0.173	2.313±0.218
	F-MNIST	2.371±0.103	**2.362**±0.053	2.442±0.067	2.470±0.092
	CIFAR-10	0.146±0.004	**0.143**±0.005	0.169±0.012	0.173±0.007

CIFAR-10: This is another standard dataset for image classification (Krizhevsky et al., 2009). It consists of ten classes of RGB natural images. We modify the architecture of VGG19 (Simonyan and Zisserman, 2014) with four convolutional layers (width 512) and two linear layers (width 512) using Leaky ReLUs with $\alpha \in \{-2, -1, 0, 0.05\}$. We use Algorithm 1 to initialize the NNs and train them using SGD with batch size 64 and cross entropy loss.

To ensure that m is sufficiently large with respect to n, we randomly sample subsets of F-MNIST and CIFAR-10 (see more details in Appendix A.3.1).

2.5.2 Results

Figure 2.1 demonstrates both training errors (top) and testing errors (bottom) for the synthetic dataset, F-MNIST and CIFAR-10 (from left to right) for different αs. We remark that we use the testing error as an approximation of the generalization error. Observing the training errors in the top row we note that the convergence is fastest for the NN with $\alpha = -1$ and the ranking of α from fastest to slowest convergence corresponds to the one predicted by our theory; that is, if α obtains

a lower estimate for γ in (2.4) than α', then it results in faster convergence in our experiments. Observing the testing errors, we note that around a small training epoch (e.g., 30 for the synthetic dataset, 20 for F-MNIST, and 200 for CIFAR-10), the testing error is smallest when $\alpha = -1$. However, at larger training epochs the gaps of the testing errors are small for most of the αs.

To get a better quantitative idea, Table 2.1 summarizes for the different data sets the training error at the last epoch and the testing error at an early epoch. We ran the experiments 10 times and reported the mean and standard deviations (std's). We note that the std's are small and for better visualization we did not include them in Figure 2.1. We observe that choosing $\alpha = -1$ gives the least final training error in all datasets. Compared to ordinary ReLU, our choice of $\alpha = -1$ reduces the final training error by at least 22% (CIFAR-10) and at most 91% (synthetic). At early training epoch, compared to ordinary ReLU, the choice of $\alpha = -1$ reduces the testing error by at least 4% (F-MNIST) and at most 15% (CIFAR-10). This correlates with the predictions we made by our theory that the optimal bounds of the convergence rate and generalization error (at a sufficiently small epoch) are achieved with $\alpha = -1$.

Lastly, we compare the theoretically predicted upper bounds of the convergence rate and the empirical convergence rates with different αs. For this purpose, we ran experiments using the synthetic dataset and California housing (see its detailed description in Appendix A.3.1) with choices of α from $[-10, 0.5]$. We approximate the convergence rate for each α using the training errors from the experiments at time steps 100 (i.e., $\mathcal{L}^{(100)}$) and $1,000$ (i.e., $\mathcal{L}^{(1000)}$). The empirical convergence rate is calculated as

$$\hat{\gamma}(\alpha) := (\mathcal{L}^{(1000)}/\mathcal{L}^{(100)})^{1/900}.$$

To simplify our upper bound, we denote the constant $\Omega\left(\frac{\eta\delta m}{nd}\right)$ in (2.4) by C_γ and

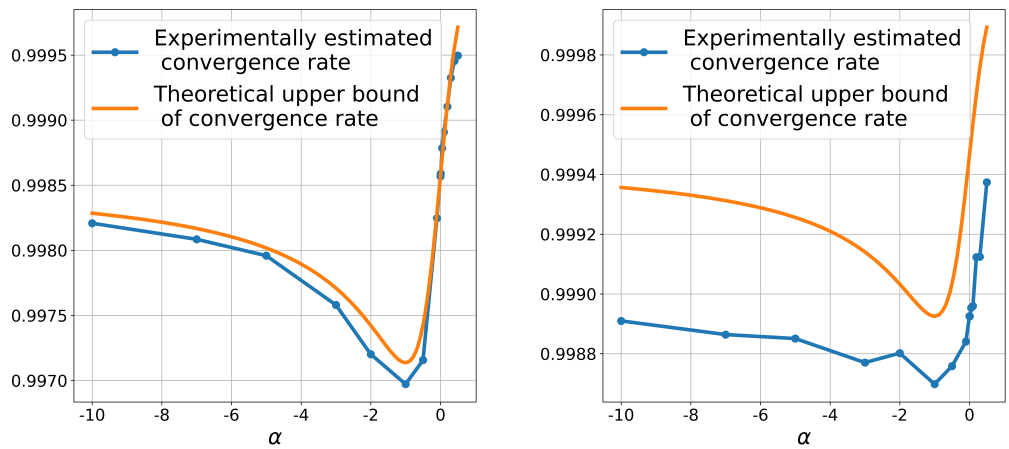

Figure 2.2: Comparison of the "shape" of the theoretical upper bound of the training convergence rate (orange line) with the calculated convergence rate (blue dots). We used the synthetic dataset (left) and California housing dataset (right) with different values of α's.

estimate its value based on the calculated convergence rate at $\alpha = 0$ as

$$C_\gamma := C_0(1 - \hat{\gamma}(0)), \tag{2.17}$$

where we choose $C_0 = 1$ for the synthetic dataset and $C_0 = 0.5$ for California housing. Consequently, we obtain our theoretical upper bounds of the convergence rates

$$\gamma(\alpha) = 1 - 0.00143\frac{(1-\alpha)^2}{1+\alpha^2} \quad \text{for the synthetic dataset,}$$
$$\gamma(\alpha) = 1 - 0.000537\frac{(1-\alpha)^2}{1+\alpha^2} \quad \text{for California housing.}$$

Figure 2.2 compares the theoretical upper bound of the convergence rate, $\gamma(\alpha)$, with the experimental convergence rate $\hat{\gamma}(\alpha)$ for the tested values of αs. It is interesting to note that the predicted upper bound dependence on α correlates very well with both numerical experiments.

Appendix A.3.2 includes additional details and numerical results. In particular, it performs experiments similar to the ones reported in Figure 2.1, while incorporating the datasets MNIST, California housing and IMDb movie reviews; the architectures of recurrent NNs and transformer NNs; and another loss function for regression. It also demonstrates how the training and testing errors depend on the NN hyperparameters (e.g., depth and width).

All codes are available at `https://github.com/sli743/leakyReLU`.

2.6 Discussion

This chapter established a mathematical theory that clarifies the impact of the Leaky ReLU parameter on bounds of both the training error convergence rate and the generalization error for overparameterized NNs. We showed that the absolute value function yields the optimal convergence rate bound for the training error and also the optimal generalization error bound when the training epoch is sufficiently small, with a sufficiently large dataset and a deep NN. Our extensive empirical tests support using the absolute value function for effective training of overparameterized NNs and for effective generalization with sufficiently small epochs and sufficiently large datasets and deep overparameterized NNs.

There are different possible extensions of our theory. For example, it is useful to extend it to other structured NNs, such as convolutional NNs (CNNs), while allowing any Leaky ReLU. Allen-Zhu et al. (2019b) established convergence for overparameterized CNNs with ReLU and one can directly extend their analysis to any Leaky ReLU. Nevertheless, it still remains open to extend the generalization theory to other structured NNs. Furthermore, it is useful to study the training convergence and generalization for larger classes of activation functions, such as the Gaussian error linear unit (Hendrycks and Gimpel, 2016).

Our work has three major limitations. First, our generalization error bound is not sufficiently small. Nevertheless, we believe it still indicates some interesting and relevant phenomena, in particular, the behavior when stopping at an early epoch. We further improved our estimates for a special class of datasets, although we observed that it was not sufficiently small in general. This is likely due to the fact that the regression setting poses greater challenges than classification. We also highlighted the possible implications of Kumar et al. (2023) to a generalization estimate given tight training error bounds.

Second, the lower bound that we require on the width, m, is generally unrealistically large and we thus find it important to extend our theory to lower values of m. Developing such a theory seems to require a careful analysis of nonlinear dynamical systems, given that current methods aim to linearize the underlying dynamical system. Nevertheless, for the special class of datasets discussed in Appendix A.2.11, we were able to provide a satisfying linear dependence of the lower bound of m on n.

Lastly, to theoretically guarantee the use of $\alpha = -1$, we need to develop respective lower bounds. We are not aware of useful and generic lower bounds and we find it rather difficult to develop them. Nevertheless, we still believe that making predictions based on the carefully developed upper bound and empirically testing them is valuable for practitioners. Indeed, our numerical results indicate the optimality of $\alpha = -1$ in many scenarios of overparameterized networks. On the other hand, we are unaware of much practical guidance that stems from the many other important and fundamental estimates in the study of overparameterized NNs. Additionally, Figure 2.2 shows cases where our upper bound for the convergence rate aligns with the observed convergence rate.

Chapter 3

An Unpooling Layer for Graph Generation

This chapter proposes a novel and trainable graph unpooling layer for effective graph generation. The unpooling layer receives an input graph with features and outputs an enlarged graph with desired structure and features. We prove that the output graph of the unpooling layer remains connected and for any connected graph there exists a series of unpooling layers that can produce it from a 3-node graph. We apply the unpooling layer within the generator of a generative adversarial network as well as the decoder of a variational autoencoder. We give extensive experimental evidence demonstrating the competitive performance of our proposed method on synthetic and real data.

This chapter is organized follows: §3.1 provides background on the problem of interest for this chapter, reviews related work, and highlights our main contributions; §3.2 details our proposed methodology; §3.3 provides theoretical guarantees of connectivity and expressivity; §3.4 reports numerical results on synthetic and real data of protein and molecule generation; §3.5 concludes this work and discusses its limitations; and §B provides details of the method and implementation, the proof and additional numerical results.

3.1 Introduction

Graph data appear in many application areas, such as chemistry (Duvenaud et al., 2015), biology (Maere et al., 2005) and social recommendation (Fan et al., 2019). Common tasks that arise with graph data include regression and classification of either graph nodes or whole graphs and graph generation, which is useful for molecule generation and drug discovery. Graph neural networks (GNNs) have successfully generalized standard methods and architectures of neural networks to graph data and have achieved great success in many common tasks.

The task of graph generation is challenging due to its vast search space and the complexity of the graph structure. Furthermore, as we clarify next, it is hard to generalize basic procedures of deep generative networks in image generation to graph generation. For image data, a generative neural network, which may take the form of the decoder of a variational auto-encoder (VAE) or the generator of a generative adversarial network (GAN), usually first converts the input to a small intermediate image and then applies convolutional-transpose layers (Zeiler et al., 2010; Radford et al., 2015) or unpooling layers (Pu et al., 2016) to upsample and refine the image. In the graph domain, it is hard to form a similar convolution-transpose or unpooling layer in order to upsample graphs. Indeed, convolution and message passing do not change the structure of the underlying graph and there is no natural way of building structure for an unpooled graph. We are unaware of any graph generation work that follows the same idea of image generation and produces intermediate graphs and upsamples them to obtain the desired graph.

Inspired by image generation, we propose a novel unpooling layer for graph data that is similar to the unpooling operator for images. By incorporating this layer, one can build a deep graph generative network that utilizes intermediate graph structures.

3.1.1 Related Work

Graph neural networks. There is already a vast amount of recent work on GNNs. Many of those works focus on regression or classification tasks of nodes or whole graphs. A common building block of a GNN is the message-passing neural network (MPNN) layer, which was first proposed for predicting molecular properties (Scarselli et al., 2008; Duvenaud et al., 2015) and was immediately extended to other applications. More recently, different variants and extensions of the MPNN layer have been proposed within GNNs. For example, graph convolutional networks (GCNs) (Kipf and Welling, 2017) use a first-order approximation of the spectral convolution to derive a simple propagation rule for node classification, graph attention networks (Veličković et al., 2018) use self-attention to assign different weights to different nodes in a neighborhood, and graph isomorphism networks (Xu et al., 2019) adopt multilayer perceptrons (MLPs) after message passing to enhance expressivity. These networks are simple to implement and also follow a message-passing scheme.

Graph pooling and unpooling. The common idea of pooling layers in 2D convolutional neural networks has been generalized to graph-based data in order to produce smaller graphs. A graph pooling layer was first proposed by Bruna et al. (2014) and later extended in various works (Defferrard et al., 2016; Ying et al., 2018; Ma et al., 2019; Lee et al., 2019). Pooling layers are widely used in classification and regression on graphs since they downsample the graph while summarizing the aggregated presence of encoded features.

Unlike the convolution-transpose or unpooling operation for images, there is no obvious way to define a trainable unpooling procedure for graphs. Some works (Jin et al., 2018, 2019; Bongini et al., 2021) sequentially expand graphs by adding one node at a time. While their operations can be considered as unpooling, they cannot be regarded as the inverse operation of common pooling, since graph pooling is not generally done by removing one node at a time. Among all works related to graph

pooling, Gao and Ji (2019) is the only one that seeks to define graph unpooling. They proposed Graph-UNet to combine pooling and unpooling processes in an encoder-decoder architecture. A Graph-UNet first pools an input graph into a smaller graph, encodes its global features and then applies the exact inverse process to perform the unpooling procedure. Since this operation deterministically depends on the pooling layers in the encoder, it is not suitable to be used for graph generation.

Graph generation. Faez et al. (2021) nicely survey graph generation models and categorized them as follows: auto-regressive (AR) (You et al., 2018b; Bongini et al., 2021; Ahn et al., 2021), VAE (Simonovsky and Komodakis, 2018; Jin et al., 2018; Samanta et al., 2020; Guo et al., 2021), GAN (De Cao and Kipf, 2018), reinforcement learning (RL) (You et al., 2018a), and normalized flow (Madhawa et al., 2019; Shi et al., 2020; Zang and Wang, 2020; Luo et al., 2021). Another recent category is diffusion models (Jo et al., 2022). There are two types of graph generation strategies: one-shot and sequential, where the former generates the output graph at once and the latter generates it in a node-by-node and edge-by-edge fashion. Most existing methods that leverage the one-shot strategy, such as De Cao and Kipf (2018); Simonovsky and Komodakis (2018); Zang and Wang (2020), produce a vectorized adjacency matrix, which does not utilize any graph structure during generation. On the other hand, methods that sequentially generate graphs typically use the graph structure. For example, You et al. (2018b); Shi et al. (2020); Luo et al. (2021) predict the next node or edge based on the features extracted from the existing graph.

Molecule generation is the most common application in this area as molecules can be naturally represented as graphs with features. However, molecules need to be chemically valid and this validity issue does not arise in general graph generation. Many methods (De Cao and Kipf, 2018; Simonovsky and Komodakis, 2018; Samanta et al., 2020; Zang and Wang, 2020) generate molecules at one-shot by producing an adjacency matrix that captures the molecular graph structure. Mahmood et al.

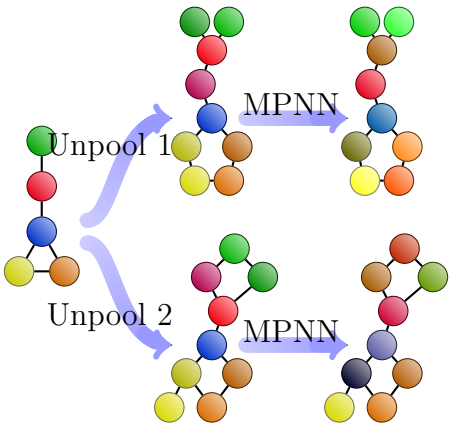

Figure 3.1: Demonstration of possible outputs of the proposed unpooling layer. Left: input graph, middle: two potential outputs of the unpooling layer, right: further application of an MPNN layer. The colors of nodes represent their features.

(2021) propose the masked graph model (MGM) that generates graphs at one-shot by sampling masked sub-graphs of the respective complete graph. This method suggests a new graph generation category. On the other hand, it is also possible to sequentially generate molecules. For example, Jin et al. (2018, 2019) first generate a junction-tree as the scaffolding and then complete the graph. The junction-tree is generated recursively from the root, one node at a time. Some other recent models (Shi et al., 2020; Ahn et al., 2021; Luo et al., 2021) apply node-by-node and edge-by-edge sequential generation. Bongini et al. (2021) break the graph generation into three subproblems: node classification (which leads to node expansion), edge classification, and edge addition, where three separate GNNs are trained for each subproblem.

3.1.2 This Work

We propose a novel unpooling layer that effectively leverages the features and structure of a given graph to form an enlarged graph with learned features and structure. One may apply additional layers, such as MPNN layers, to further refine the features of nodes and edges in the graph. Figure 3.1 demonstrates two possible outputs of the

unpooling layer with a followup MPNN layer for a given input graph. By incorporating unpooling layers in deep graph generative networks, we can generate a graph at one shot. In the experiments, we incorporate unpooling layers within both the generator of GAN and the decoder of VAE. We demonstrate in a synthetic setting how the unpooling layers reveal useful intermediate graph structures. We believe that the incorporation of such structures results in the competitive performance which is evident in all numerical experiments.

Our proposed unpooling layer (UL) leads to one-shot generation that utilizes the graph structure during generation. Among all existing methods, only the masked graph model (MGM) of Mahmood et al. (2021) applies one-shot generation that utilizes the graph structure during generation. Nevertheless, the implementation of our method is very different from Mahmood et al. (2021). In particular, the proposed unpooling layer enlarges the graph at each intermediate step, whereas Mahmood et al. (2021) sample the graph in masked subgraphs. As mentioned earlier, the graph generation category of MGM is rather different from the common categories. Furthermore, MGM was implemented and applied for molecule generation and not general graph generation. Therefore, in terms of methodology our proposed strategy is distinguished from the many previous graph generation methods.

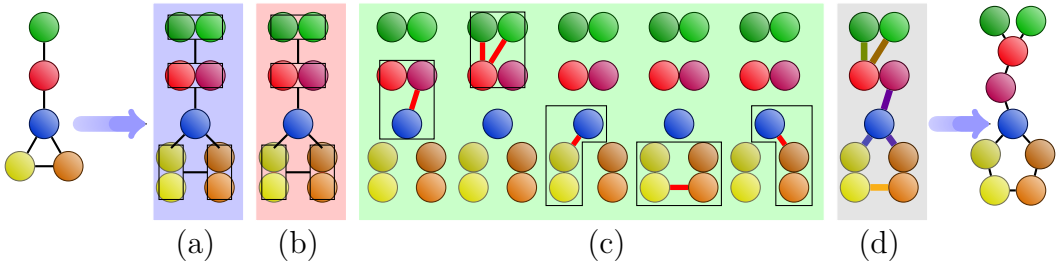

Figure 3.2: Demonstration of the steps of the unpooling layer: (a) unpool and generate children nodes; (b) build intra-links within pairs of children nodes; (c) build inter-links involving children nodes; (d) build features for the edges. Right: The output graph.

We emphasize the following contributions of our work:

- We propose a novel unpooling layer that produces an enlarged graph with learnable structure. It can be inserted into GANs and VAEs. The resulting generation framework is distinguished in its ability to both generate graphs at one shot and utilize the graph structures for generation.
- We show that the unpooling layer is valid and expressive. That is, the unpooled graph remains connected and any connected graph can be generated by a series of unpooling layers from a 3-nodes graph.
- We test the unpooling layer within both GANs and VAEs on a random graph dataset, a protein dataset and two molecule datasets and demonstrate competitive performance.

3.2 Methodology

Section 3.2.1 clarifies the construction of the unpooling layer and §3.2.2 explains how to use such layers for graph generation and how we update the parameters for the unpooling layers.

3.2.1 Unpooling Layer

Given an input graph (with features), the unpooling layer first unpools some of its nodes by replacing them with two "children" nodes and then learns a graph structure for the new set of nodes (in the output graph) and further learns new features.

Notation. We use the following notation for the input graph and its features. Its node set is $V = [N] := \{1, \cdots, N\}$; its edge set is E, where its members are of the form $\{i, j\}$ for some i, $j \in V$; its number of edges is $M = |E|$; its node feature matrix is $\boldsymbol{X} \in \mathbb{R}^{N \times d}$, where its i-th row, $\boldsymbol{x}_i$, is the d-dimensional feature of node i; and its edge feature matrix is $\boldsymbol{W} \in \mathbb{R}^{M \times e}$, where for edge $\{i, j\} \in E$, its corresponding row

of $\boldsymbol{W}$ is the e-dimensional feature of that edge, which we denote by $\boldsymbol{w}_{i,j}$. Similarly, we use the following notation for the output graph and its features: V^o is its node set, E^o is its edge set and $\boldsymbol{Y}$ and $\boldsymbol{U}$ are the feature matrices for the output nodes and edges, respectively. We remark that the size of V^o lies in $[|V|+1, 2|V|]$ and depends on the hyperparameters that determine which nodes should be unpooled.

The input and output graphs of the unpooling layers with their features are respectively denoted by

$$\mathcal{G} = (V, E, \boldsymbol{X}, \boldsymbol{W}) \text{ and } \mathcal{G}^o = (V^o, E^o, \boldsymbol{Y}, \boldsymbol{U}).$$

We will refer to them as featured graphs and to (V, E) and (V^o, E^o) as graphs.

An overview of the unpooling layer. The unpooling layer determines the output graph in a stochastic manner. Ideally, it should produce probabilities of every possible output graph based on the input graph by using trainable parameters, $\boldsymbol{\theta}_P$. That is, it would output a probability mass function, $\boldsymbol{p}$, on the sample space $\mathbb{G}^o = \{(V^o, E^o) : \text{unpooled from } \mathcal{G}\}$, where $\boldsymbol{p}(\mathcal{G}; \boldsymbol{\theta}_P) \in [0,1]^{|\mathbb{G}^o|}$. In this ideal case, the unpooling layer then randomly draws an output graph, (V^o, E^o), according to this probability mass function. The drawn probability of this sample point, which we denote by

$$\mathbb{P}(V^o, E^o | \mathcal{G}; \boldsymbol{\theta}_P), \tag{3.1}$$

can then be used to update the parameters $\boldsymbol{\theta}_P$ during training. Therefore, the unpooling layer can ideally refine the probability distribution $\boldsymbol{p}(\mathcal{G}; \boldsymbol{\theta}_P)$ in order to obtain graphs that minimize the training loss function.

Since the sample space $\mathbb{G}^o$ is huge, we cannot explicitly produce the probabilities of all possible output graphs. In practice, we use several multi-layered perceptrons (MLPs) to produce probabilities to determine if nodes should be unpooled and if

edges between unpooling children nodes should be included in the output graph. The product of all these probabilities for nodes and edges gives the probability of the entire output graph in (3.1) (assuming these events are independent), which is used during training.

After forming the output graph, the unpooling layer also produces the node and edge features using two MLPs that exploit the input featured graph and possibly the output graph, that is,

$$\boldsymbol{Y} = \text{MLP-}Y(\mathcal{G};\theta_Y), \boldsymbol{U} = \text{MLP-}W(V^o, E^o, \mathcal{G};\theta_W).$$

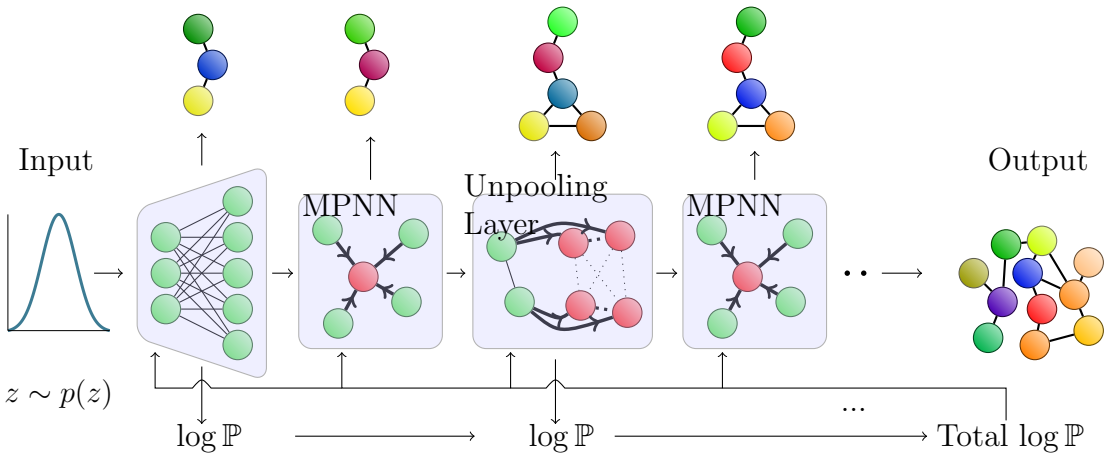

Figure 3.3: Demonstration of a generative neural network with unpooling and MPNN layers. The central row shows the input latent vector, an initial layer that creates a 3-node graph, a series of MPNN and unpooling layers, and the output featured graph (the features are represented by colors). The top row shows the intermediate hidden featured graphs outputted at every layer. The initial layer and the unpooling layers generate log probabilities. The generative GNN accumulates them to obtain the total log probability (shown in the bottom row); it uses it in the REINFORCE algorithm to train the unpooling layers (see (3.3)).

Detailed mechanism of the unpooling layer. The unpooling layer contains seven MLPs that serve different purposes, which we introduce below and in the

supplementary materials §B.1. For the formation of the unpooling layer We use the following three node sets that partition the input node set: (1) the set of nodes I'_s that are unchanged in the unpooling layer; (2) the set of nodes I'_u that are determined to be unpooled in the unpooling layer; and (3) the set of nodes I'_r that requires a probabilistic decision whether to unpool or not.

With the specified node sets, we describe the procedure of generating the output graph in the unpooling layer according to 3 steps, which we demonstrate in Figure 3.2.

Step 1. Generating the output nodes and node features ((a) in Figure 3.2) We first probabilistically determine which nodes in I'_r will be unpooled. For each node $i \in I'_r$, we determine the probability of unpooling it, $p_r(\boldsymbol{x}_i)$, by an MLP as follows: $p_r(\boldsymbol{x}_i) = \text{MLP-}R(\boldsymbol{x}_i)$. Then we draw uniform random variables $U_i \sim U[0,1]$ and form the following sets I_u and I_s of unpooled nodes and unchanged (or stable) nodes: $I_u := I'_u \cup \{i \in I'_r \ : \ U_i < p_r(\boldsymbol{x}_i)\}$ and $I_s := I'_s \cup \{i \in I'_r \ : \ U_i \geq p_r(\boldsymbol{x}_i)\}$.

We remark that if we want to generate the output graph with a fixed number of nodes, we could choose $I'_r = \emptyset$ for the unpooling layer.

The set of nodes of the output graph V^o is the union of the nodes in I_s (with different indices) and a set of nodes of size $2|I_u|$ representing the unpooled nodes from I_u. For each node $i \in I_s$, we denote by $f(i)$ the index of this node in the output graph. For each node $i \in I_u$, we denote by $f_1(i)$ and $f_2(i)$ the indices of the two unpooled nodes in the output graph. The output node features are generated by an MLP and two projection operators P_{S_1} and P_{S_2}, which are defined in detail in §B.3.2

$$\boldsymbol{y}_{f(i)} = \text{MLP-}y(P_{S_1}\boldsymbol{x}_i), \quad \text{for } i \in I_s.$$
$$\boldsymbol{y}_{f_j(i)} = \text{MLP-}y(P_{S_j}\boldsymbol{x}_i), \quad \text{for } i \in I_u, \ j = 1, 2.$$

Step 2. Building output edges. We sequentially generate the set E^o of edges in the output graph following the next two substeps. We initiate this set by $E^o := \emptyset$.

Step 2.1. Building intra-links ((b) in Figure 3.2). For each node $i \in I_u$, we determine whether to generate an edge that connects the children nodes $f_1(i)$ and $f_2(i)$ based on a probability, which we denote by $p_c(\boldsymbol{x}_i)$. This probability is produced using an MLP as follows: $p_c(\boldsymbol{x}_i) \equiv \text{MLP-IA}(\boldsymbol{x}_i)$. Then we draw uniform random variable $U_i \sim U[0,1]$ and if $U_i < p_c(\boldsymbol{x}_i)$, we add this edge to the output graph, that is, $E^o = E^o \cup \big\{\{f_1(i), f_2(i)\}\big\}$.

Step 2.2. Building inter-links ((c) in Figure 3.2). For each edge $\{i,j\} \in E$ in the input graph, we determine the edges for the corresponding nodes in the output graph according to the following three different cases: (1) $i,j \in I_s$: we include the edge $\{f(i), f(j)\}$ in the output graph; (2) $i \in I_s$ and $j \in I_u$: we probabilistically determine what are the edges between $f(i)$ and $\{f_1(j), f_2(j)\}$; and (3) $i,j \in I_u$: we probabilistically determine what are the edges between $\{f_1(i), f_2(i)\}$ and $\{f_1(j), f_2(j)\}$.

For each edge $\{i,j\} \in E$, we introduce node sets $N_{\{i,j\},i}$ and $N_{\{i,j\},j}$ in order to uniformly handle the above cases. For $\{i,j\} \in E$ and $i \in V$, we form $N_{\{i,j\},i}$ as follows: If $i \in I_s$, then $N_{\{i,j\},i} = \{f(i)\}$ and if $i \in I_u$, then $N_{\{i,j\},i}$ is a nonempty subset of the children nodes of i, which we probabilistically determine as follows. We use an MLP to calculate two probabilities: $(p_1, p_2) = \text{MLP-IE}(\boldsymbol{y}_{f_1(i)}, \boldsymbol{y}_{f_2(i)}, \boldsymbol{w}_{i,j}, \boldsymbol{x}_j)$. We then draw a uniform random variable $U \sim U[0,1]$ and determine $N_{\{i,j\},i}$ as follows:

$$N_{\{i,j\},i} = \begin{cases} \{f_1(i)\}, & \text{if } \ U < p_1; \\ \{f_1(i), f_2(i)\}, & \text{if } \ U \geq p_1 + p_2; \\ \{f_2(i)\}, & \text{otherwise.} \end{cases}$$

For $\{i,j\} \in E$ and $j \in V$ we similarly define $N_{\{i,j\},j}$, while swapping i and j. The edges in the output graph are updated as follows:

$$E^o = E^o \cup \{\{k,l\} : \ k \in N_{\{i,j\},i}, \ l \in N_{\{i,j\},j}\}. \tag{3.2}$$

We need to take extra care to ensure connectivity and expressivity of the output graph. We first form two additional MLPs, MLP-C and MLP-IE-A to calculate probabilities. We then probabilistically form some additional edges. These details are a bit technical and can be fully understood after getting familiar with our theory for connectivity and expressivity (see §3.3 and the proofs in the supplementary materials §B.4). Therefore we leave these details to §B.1 of the supplementary materials (see Step 2b and Step 2d in §B.1).

Step 3. Constructing the edge features ((d) in Figure 3.2). For each edge $\{k, l\} \in E^o$, we construct the edge feature $\boldsymbol{u}_{k,l}$ by an MLP as follows:

$$\boldsymbol{u}_{k,l} = \text{MLP-}u\big(\text{LeakyReLU}(\boldsymbol{y}_k + \boldsymbol{y}_l)\big).$$

Summary. We described a probabilistic construction of $\mathcal{G}^o = (V^o, E^o, \boldsymbol{Y}, \boldsymbol{U})$. It contains seven MLPs to produce various probabilities and features for the nodes and edges of $\mathcal{G}^o$. The parameters in those seven MLPs form the training parameters of the unpooling layer. The overall probability $\mathbb{P}(V^o, E^o|\mathcal{G}; \boldsymbol{\theta}_P)$ is the product of all the probabilities in the first two steps and is used to update the training parameters in the unpooling layer, while using the REINFORCE algorithm introduced below.

3.2.2 Graph Generation and Training

We use the unpooling layer within a generative GNN, which can be either a generator of a GAN or a decoder of a VAE. We describe here its basic mechanism and demonstrate it in Figure 3.3. Complete details of implementation are in both §3.4 and the supplementary materials. The generative GNN first maps a given latent vector into a featured 3-nodes graph, whose structure is probabilistically determined by an MLP and its edge features are determined by another MLP (details are in the supplementary materials). Next, it applies a GCN (such as MPNN) to update the node features for

this initial featured graph. It then sequentially applies unpooling layers, where each of them is followed by a GCN (such as MPNN), which further updates the node features. The final output is the generated graph.

The parameters used for generating features in the unpooling layer can be updated during training following the common framework of GAN or VAE. The major challenge in using the unpooling layers for graph generation is that the graph generation process is not differentiable. To overcome this, we follow REINFORCE with baseline (Weaver and Tao, 2001; Sutton and Barto, 2018) to update all the graph generation parameters in the unpooling layer.

In order to explain this procedure with more details, we need the following notation. We denote by G a generative GNN (a generator of a graph GAN or the decoder of a graph VAE) with several unpooling layers that takes a latent vector and produces a generated graph. We denote by m the number of unpooling layers of G, by U_1, U_2, $\cdots$, U_m the unpooling layers and by (V_1^o, E_1^o), $\cdots$, (V_m^o, E_m^o) the generated intermediate graphs. Let $\boldsymbol{\theta}$ denote all the parameters of G, which include the parameters of the MLPs in the unpooling layers. In view of (3.1), the total log probability of the unpooling layer U_k is $\log \mathbb{P}(V_k^o, E_k^o)$. We define the total log probability of the generator G as $\log \mathbb{P} := \sum_k \log \mathbb{P}(V_k^o, E_k^o)$ and note that $\log \mathbb{P}$ depends on $\boldsymbol{\theta}$. We denote the learning rate by α and the reward for the generated graph by r. Note that this reward depends on the specific generation task, e.g., it can be the likelihood predicted by the discriminator or the chemical property which one aims to optimize.

We update $\boldsymbol{\theta}$ as follows

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha(\nabla_{\boldsymbol{\theta}} \log \mathbb{P}|_{\boldsymbol{\theta}_k})(r - \mathbb{E}r), \tag{3.3}$$

where we approximate $\mathbb{E}r$ by the sample mean. In our experiments we incorporate the unpooling layers within a GAN and a VAE. For a GAN, we set the reward r

to be $D(G(z;\boldsymbol{\theta}))$, where D is the GAN's discriminator, in order to compete with the discriminator. For a VAE, we choose the reward r to be the negative of the reconstruction error in order to minimize the reconstruction error.

3.3 Theoretical Guarantees

We establish the connectivity and expressivity of the unpooling layer. All proofs are in the supplementary materials.

3.3.1 Guarantee of Connectivity of the Output Graph

The following proposition implies that if the input graph is connected, then the unpooling layer will produce a connected graph. This is an important property in molecule generation since otherwise the output molecule will be invalid. Adjacency matrix-based generators cannot ensure connectivity.

Proposition 3.1

Given an unpooling layer and any connected input graph $\mathcal{G}$, the output graph, $\mathcal{G}^o$, of this unpooling layer is connected. □

3.3.2 Guarantee of Expressivity for the Unpooling Layer

It is important to know whether a series of unpooling layers can produce any connected graph. For instance, in molecule generation, a good generative model should contain all valid molecules in the set of possible output. Some previous work (e.g., Jin et al. (2018)) cannot produce some valid molecular structures and is thus not fully expressive. Fortunately, we are able to produce any connected graph by applying certain unpooling layers to an input graph with three nodes (our implementation of the graph generative model starts with a 3-nodes graph). We first formulate a theorem on the expressivity

of a single unpooling layer and then formulate the desired corollary when starting with a 3-nodes graph and using a series of unpooling layers.

Theorem 3.1

Given a connected graph $\mathcal{G}^o$ with N nodes and an integer $K \in [\lceil N/2 \rceil, N-1]$, there exist an unpooling layer and an input graph $\mathcal{G}$ with K nodes so that $\mathcal{G}^o$ is the corresponding output. □

Corollary 3.1

Given a connected graph $\mathcal{G}^o$ with N nodes, there exist a 3-nodes graph $\mathcal{G}$ and $\lceil \log_2(N/3) \rceil$ unpooling layers, so that $\mathcal{G}^o$ is the output of this series of unpooling layers acting on $\mathcal{G}$. □

We remark that the proof of Theorem 3.1 naturally provides a "pooling" procedure on the graph structure which can be regarded as the inverse operation of our unpooling layer. This validates the name "unpooling".

3.4 Experiments

We demonstrate the effectiveness of the unpooling layer for molecule generation. We describe the two datasets in §3.4.1, the evaluation metrics in §3.4.2 and the details of the implemented methods in §3.4.3. We then report the results in §3.4.4, while comparing with benchmark methods. All implemented codes are provided in the supplementary materials.

3.4.1 Datasets

Waxman random graph dataset. The dataset contains randomly generated Waxman graphs (Waxman, 1988). More precisely, we first created 20,000 graphs

with 12 nodes and node features uniformly drawn from $[0, 1]^2$. For each graph, we connected nodes i and $j \in [12]$ with probability $qe^{-sd_{ij}}$, where $q = 0.65$, $s = 0.3$ and d_{ij} is the Euclidean distance between their features. We do not assign edge features. Next, for each graph we compute the largest connected subgraph as long as it has at least 5 nodes. The final set contains these subgraphs with at least 5 nodes (the ones with less nodes are discarded). Thus the number of nodes ranges from 5 to 12 and the node features are the x and y coordinates. On average, each graph contains 9.2 nodes and 10.3 edges. There are 18,910 graphs in this dataset.

Protein dataset. We use the protein dataset introduced in Guo et al. (2021), which is a benchmark for graph generation (Du et al., 2021). All the graphs contain 8 nodes and their node features are their 3D coordinate vectors. There are no edge features. On average, each graph contains 8 nodes and 19.3 edges. There are 76,000 graphs in the datasets, where we use 38,000 for training and 38,000 for testing.

Molecule datasets. We use two common datasets for molecule generation: QM9 (Ramakrishnan et al., 2014) and ZINC (Sterling and Irwin, 2015). QM9 contains 130k molecules and each molecule consists of up to 9 heavy atoms among carbon (C), oxygen (O), nitrogen (N) and fluorine (F). In §3.4.3 we explain how we choose the hyperparameters of the unpooling layers so that the generator will generate molecules with the number of atoms ranging between 6 and 9. On average, each graph contains 8.8 nodes and 9.4 edges.

ZINC contains about 250k molecules, where each molecule consists of 9 to 38 heavy atoms among carbon (C), oxygen (O), nitrogen (N), sulfur (S), fluorine (F), chlorine (Cl), bromine (Br), iodine (I) and phosphorus (P). For simplicity, we only take molecules with 11 - 36 heavy atoms (99.8% of ZINC). On average, each graph contains 23.2 nodes and 24.9 edges.

For both QM9 and ZINC, we include the following node features: atom type, chiral specification of an atom (unspecified, clockwise or counter-clockwise) and the formal

charge of an atom (0, +1 or -1). We use bond type (single, double or triple) as edge features. The node and edge features are represented as one-hot vectors.

3.4.2 Evaluation Metrics

In the numerical experiments of the Waxman random graph and protein datasets, we evaluate the similarity of the generated data and source data by comparing the distributions of some graph properties in the generated graphs and source data. We use the following graph properties for comparison: average node connectivity, average clustering coefficient, edge density and node features. For both datasets, we report the Kullback–Leibler divergence and Wasserstein distance between the distributions of the source and generated data. For the Waxman random graph dataset we further demonstrate the distributions of the four properties for the source and generated data, while considering several methods for graph generation.

In the numerical experiments of molecule generation, we compare the different generators by generating 10,000 molecules and applying the following metrics: validity (the ratio between the number of generated valid molecules and all generated graphs); uniqueness (the ratio between the number of unique valid molecules and generated valid molecules); and novelty (the ratio between the number of unique valid molecules which are different from all molecules in the dataset and the total number of generated unique valid molecules). We also report the geometric mean of the above three metrics (G-mean).

3.4.3 Implementation Details

We implemented a GAN with the unpooling layers (UL GAN), a GAN with an adjacency matrix-based generator (Adj GAN) and a VAE with the unpooling layers (UL VAE). For simplicity, we just describe here the implementation for molecule

generation using the QM9 dataset. Indeed, the implementations of Adj GAN, UL GAN and UL VAE for the other applications are similar. Additional details are in the supplementary materials.

Discriminator for Adj GAN and UL GAN. It takes an input graph and uses two MPNN layers with 128 units to generate a graph $\mathcal{G} = (V, E, \boldsymbol{X}, \boldsymbol{W})$. It then aggregates the node features (the rows $\{\boldsymbol{x}_j\}_{j \in V}$ of $\boldsymbol{X}$) to produce the following single feature vector for the graph:

$$\boldsymbol{h}(\mathcal{G}) = \sum_{j \in V} \sigma(\text{lin}_1(\boldsymbol{x}_j)) \odot \tanh(\text{lin}_2(\boldsymbol{x}_j));$$

where σ is logistic sigmoid, $\odot$ is element-wise multiplication and lin_1 and lin_2 are two layers with 128 units. Then it applies a layer lin_3 with 256 units. A final layer with a single unit then produces the output, where its activation function is tanh. A batch normalization and leaky ReLU activation function are used after the two MPNNs and $\text{lin}_1, \text{lin}_2, \text{lin}_3$ layers.

Encoder for UL VAE. It has the same architecture as the above discriminator, except that the final layer maps into a 256-dimensional vector with a linear activation function. This vector further splits to two 128-dimensional vectors: $\boldsymbol{z}_\mu$ and $\boldsymbol{z}_\sigma$. It then generates the following latent vector: $\boldsymbol{z} = \boldsymbol{z}_\mu + \exp(\frac{1}{2}\boldsymbol{z}_\sigma) \odot \boldsymbol{x}$, where $\boldsymbol{x} \sim N(0, 1)$.

Generator for Adj GAN. It contains four linear layers with $128, 256, 256, 512$ units with batch normalization and leaky ReLU activation function. The last layer generates a 9×11 tensor (matrix) for the node features and $9 \times 9 \times 4$ tensor for the edge features. We use a hard Gumbel softmax to produce the one-hot feature vectors for the nodes and edges.

Generator for UL GAN and decoder for UL VAE. It takes a 128-dimensional vector and outputs a graph. The generator contains the following layers: an initial MLP layer that takes the random noise vector and creates a 3-nodes graph with

256-dimensional node features; an MPNN layer with 128 units; an unpooling layer that maps the 3-nodes graph to a 4-or-5-nodes graph; an MPNN layer with 128 units; an unpooling layer that maps the 4-or-5-nodes graph to a 6-to-9-nodes graph; an MPNN layer with 64 units; a linear layer with 64 hidden units and two final layers that produces node and edge features in $\mathbb{R}^{10}$ and $\mathbb{R}^{3}$, respectively. A skip connection, which takes the input vector, is added to the node features after each unpooling layer. Finally, a hard Gumbel softmax generates the desired one-hot features. The dimension of the edge features in all intermediate graphs is 32. The log probabilities from this sampling process are added to obtain the total log probability when updating the parameters according to (3.3).

Training process. For UL GAN and Adj GAN, the loss function corresponds to Wasserstein GAN with gradient penalty Gulrajani et al. (2017). For UL VAE, the loss function is the sum of the reconstruction errors of node features and edge features and the Kullback-Leibler divergence between the latent vector and a standard Gaussian. We use the Adam optimizer with a learning rate 2×10^{-4} for the generator and a learning rate 10^{-4} for the discriminator with a training batch size of 64. During the training process, we evaluate the model every 500 iterations and we report the result with optimal validity before a mode collapse occurs. In each training step, we alternatively minimize the loss function and update the parameters according to the policy gradient procedure in (3.3).

3.4.4 Results

For all datasets, we generate 10,000 samples for evaluation. For the Waxman random graph and protein datasets, we compared Adj GAN, UL GAN and UL VAE with our implementations of the following baseline methods: GraphVAE (Simonovsky and Komodakis, 2018), GraphRNN You et al. (2018b), GCPN (You et al., 2018a), GraphAF (Shi et al., 2020) and SGD-VAE (Guo et al., 2021). Since GraphAF, GraphVAE and

GCPN were designed for molecule generation, we adapted them to general graph generation. We remark that GCPN is not able to produce numeric node features and we thus do not report the node feature metrics for it. In order to test the effect of skips connections, we implemented for the random graph dataset a version of UL GAN that has no skip connections from the input vector and we refer to it as UL GAN (NS).

Table 3.1: Results for the Waxman random graph dataset. We report the KL divergence and Wasserstein distance with respect to the following quantities: edge density (kl e ds and wd e ds), average node connectivity (kl cn and wd cn), average clustering coefficient (kl clst and wd clst) and node features (kl n ft and wd n ft).

Methods	kl e ds	kl clst	kl cn	kl n ft	wd e ds	wd clst	wd cn	wd n ft
GraphVAE	4.009	6.127	2.363	5.088	0.118	0.140	0.315	0.183
GraphRNN	0.592	0.034	0.373	0.717	0.052	**0.019**	0.217	0.135
GCPN	0.402	0.292	0.096	N/A	0.043	0.086	0.116	N/A
GraphAF	0.507	0.249	0.460	0.382	0.032	0.079	0.503	0.163
SGD-VAE	1.493	0.321	0.448	0.378	0.122	0.070	0.294	0.131
ADJ GAN	1.128	1.081	0.247	0.196	0.065	0.097	0.178	0.088
UL VAE	0.092	0.408	0.108	0.635	0.010	0.099	0.105	0.132
UL GAN (NS)	0.100	0.098	0.088	0.147	0.011	0.076	0.046	0.056
UL GAN	**0.001**	**0.023**	**0.034**	**0.145**	**0.004**	0.027	**0.011**	**0.042**

Figure 3.4 demonstrates the distributions of the four graph properties of both the source data and the different generating methods for the Waxman random graph dataset. For all four properties, the distribution obtained by UL GAN seems to be the closest to the source data. Table 3.1 reports the 8 evaluation metrics for the Waxman random graph dataset. We note that UL GAN outperforms the other methods in most of the metrics, except for Wasserstein distance between average clustering coefficients, where GraphRNN achieves the smallest metric. Nevertheless, ULGAN achieves the smallest KL divergence and its distribution seems to be closer to the source data according to Figure 3.4. The better performance of UL GAN over Adj GAN and of

Table 3.2: Results for the protein dataset. We report the same quantities summarized in the caption of Table 3.1.

Methods	kl e ds	kl clst	kl cn	kl n ft	wd e ds	wd clst	wd cn	wd n ft
GraphVAE	2.497	0.343	3.228	9.777	0.065	0.020	0.758	10.590
GraphRNN	0.082	0.337	0.163	2.818	0.013	0.071	0.165	4.473
GCPN	1.567	4.943	7.358	N/A	0.483	0.508	2.292	N/A
GraphAF	1.942	1.659	1.987	2.603	0.043	0.163	1.543	17.217
SGD-VAE	1.035	1.228	0.975	10.195	0.169	0.311	1.549	10.698
ADJ GAN	6.176	4.600	7.145	0.730	0.086	0.050	0.815	6.705
UL VAE	0.492	0.889	0.791	0.181	0.041	0.072	0.373	4.344
UL GAN	**0.074**	**0.224**	**0.101**	**0.095**	**0.011**	**0.009**	**0.127**	**3.142**

UL VAE over GraphVAE indicates the clear advantage of using the unpooling layer over a standard adjacency-based method. We further note that the improvement of UL GAN over UL GAN (NS) is not significant. This indicates that even without the skip connection our model performs really well and that its main advantage is due to the unpooling layer and not the skip connection.

Table 3.2 reports the evaluation metrics for the protein dataset. We note that UL GAN outperforms the other baseline methods in all the metrics. Also, the adjacency matrix-based methods (GraphVAE and Adj GAN) perform poorly in this dataset while their unpooling-layer-based counterparts (UL VAE and UL GAN) perform much better.

For QM9, we compare UL VAE, UL GAN, Adj GAN, MolGAN (De Cao and Kipf, 2018), CharacterVAE (Gómez-Bombarelli et al., 2018), GrammarVAE (Kusner et al., 2017), Graph VAE (Simonovsky and Komodakis, 2018), GraphAF (Shi et al., 2020), GraphDF (Luo et al., 2021), MoFlow (Zang and Wang, 2020), Spanning tree (Ahn et al., 2021), and MGM (Mahmood et al., 2021). For ZINC, we compare UL GAN, Adj GAN, CharacterRNN (Segler et al., 2018), LatentGAN (Prykhodko et al., 2019), junction tree VAE (JT VAE) (Jin et al., 2018), GraphAF (Shi et al., 2020), GraphDF

(Luo et al., 2021), MoFlow (Zang and Wang, 2020), and Spanning tree (Ahn et al., 2021). We do not report results of UL VAE for ZINC because its training was slow and we do not have results for its counterpart, GraphVAE. For Adj GAN, UL GAN and UL VAE, we report the means based on 100 runs of generating 10k samples. The results of the other baseline methods are copied from their original papers. Standard deviations for our implementations are included in the supplementary materials.

Table 3.3 and Table 3.4 report validity, uniqueness, novelty and their geometric mean for QM9 and ZINC, respectively. For QM9, UL GAN improves significantly from Adj GAN. Its performance is overall competitive when compared to other state-of-the-art approaches. In particular, UL GAN achieves the third-highest geometric mean. UL VAE significantly outperforms its adjacency-matrix-based counterpart, GraphVAE. For ZINC, UL GAN achieves perfect uniqueness and novelty scores. In terms of validity, it outperforms Adj GAN, whose validity and uniqueness scores are poor. We thus note that our unpooling layer is able to generate graphs of moderate sizes, while adjacency-matrix generators are only suitable for small graphs. Although some other methods achieve better validity, the overall performance of UL GAN is comparable with state-of-the-art methods.

Figure 3.5 studies the latent space structure of UL GAN for QM9. It picks two latent vectors $\boldsymbol{z}_0$ and $\boldsymbol{z}_1$ corresponding to a string-like molecule and a molecule with a ring, respectively. Then, it forms a series of latent vectors $\boldsymbol{z}_t = t\boldsymbol{z}_1 + (1-t)\boldsymbol{z}_0$ and shows their corresponding molecules. We note that as t increases the molecular structures are changing from string-like to ring-like ones.

Table 3.3: Validity, uniqueness, novelty and their geometric mean (G-mean) for molecule generation using QM9. Scores for the competing methods (listed above the middle line) were copied from their original papers.

Method	Valid	Unique	Novel	G-mean
CharacterVAE	0.103	0.675	0.900	0.397
GrammarVAE	0.602	0.093	0.809	0.356
GraphVAE	0.557	0.760	0.616	0.639
MolGAN	0.981	0.104	0.942	0.458
GraphAF	0.670	0.945	0.888	0.825
GraphDF	0.827	0.976	0.981	0.925
MoFlow	0.962	**0.992**	**0.980**	**0.978**
Spanning tree	**1.00**	0.968	0.727	0.889
MGM	0.886	0.978	0.518	0.766
UL VAE	0.735	0.940	0.949	0.869
Adj GAN	0.941	0.139	0.886	0.488
UL GAN	0.907	0.826	0.949	0.893

Table 3.4: Validity, uniqueness, novelty and their geometric mean (G-mean) for molecule generation using ZINC. Scores for the first three methods were copied from Polykovskiy et al. (2020) and scores for other competing methods (listed above the middle line) were copied from their original papers.

Method	Valid	Unique	Novel	G-mean
CharRNN	0.975	**1.00**	0.842	0.936
LatentGAN	0.897	0.997	0.950	0.947
JT VAE	**1.00**	**1.00**	0.914	0.970
GraphAF	0.680	0.991	**1.00**	0.877
GraphDF	0.890	0.992	**1.00**	0.959
MoFlow	0.818	**1.00**	**1.00**	0.935
Spanning tree	0.995	**1.00**	0.999	**0.998**
Adj GAN	0.109	0.196	**1.00**	0.277
UL GAN	0.871	**1.00**	**1.00**	0.955

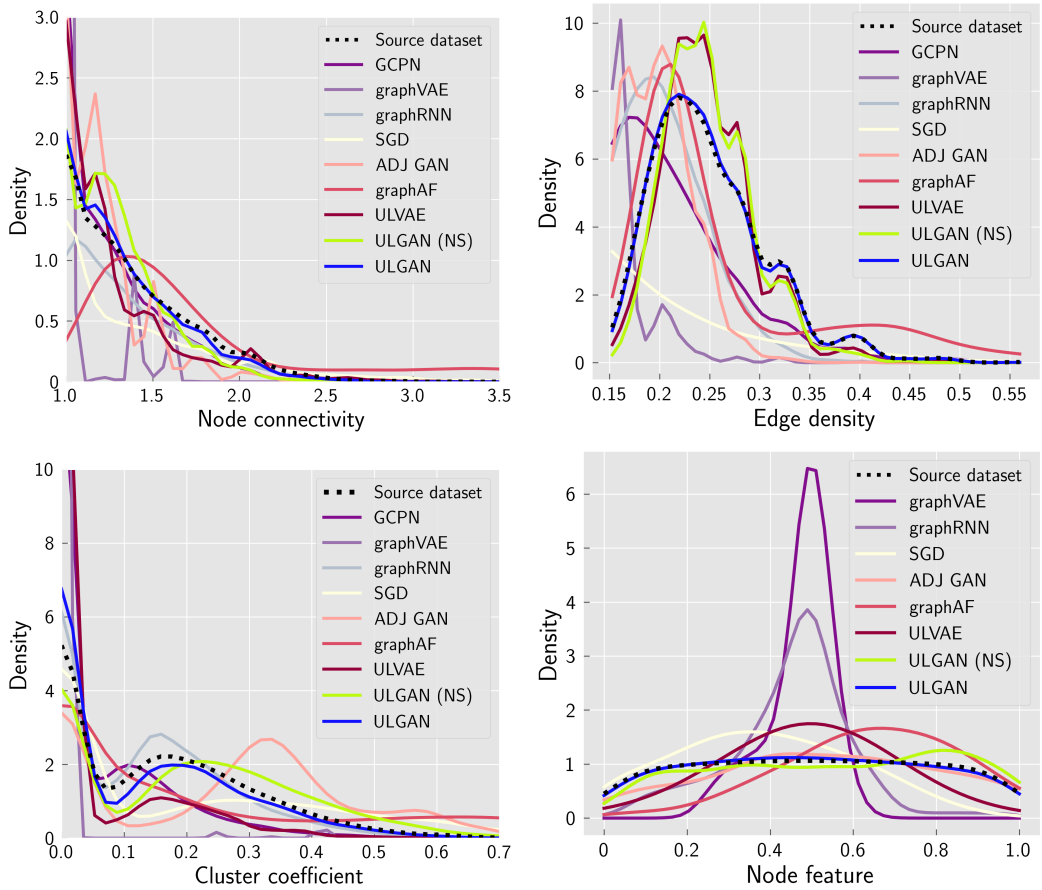

Figure 3.4: Distributions of the four graph properties for the generated and source data using the Waxman random graph dataset. The graph properties include average node connectivity (top left), average clustering coefficient (bottom left), edge density (top right), and node features (bottom right).

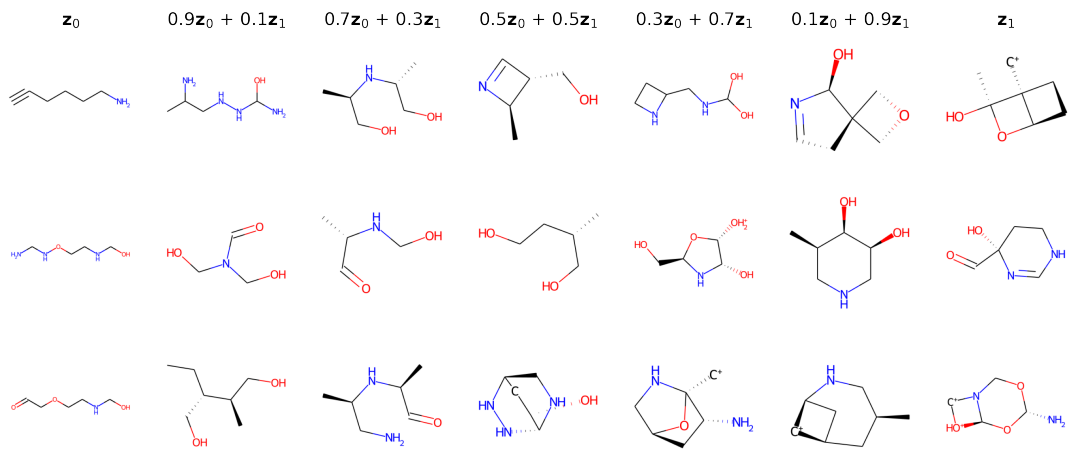

Figure 3.5: Demonstration of gradual change between two different molecular properties when applying UL GAN for QM9. Left: 3 string-like molecules (with latent vector $\boldsymbol{z}_0$), Right: 3 molecules with rings (with latent vector $\boldsymbol{z}_1$). We gradually change the latent vector (on top) and notice the gradual change of the 3 molecules from having string-like structure to having ring-like structure.

3.5 Conclusion and Limitations

This chapter introduced a novel unpooling layer that can enlarge a given graph. We have proved that this unpooling layer generates connected graphs and its range covers all possible connected graphs. We utilized such layers within GANs and VAEs and tested their performance for graph, protein and molecule generation. Our unpooling-based generation outperforms other methods for graph and protein generation and is competitive for molecule generation. In particular, a significant improvement was noticed in comparison to other methods that are based on adjacency matrices, such as Adj GAN, MolGAN and Graph VAE.

The unpooling layer can be used for other purposes. In future work, we plan to apply it to generic graph reconstruction, while considering the particular applications of recommender systems and graph anomaly detection. We also plan to apply it to conditional graph generation, that is, generating graphs with some desired properties.

The unpooling layer has some limitations. First, its training requires relatively large computational resources, since it relies on various probabilities in order to determine the graph structure. Second, it becomes difficult to optimize several unpooling layers within the generative model because the log probabilities are added and backpropagated in all the unpooling layers.

Our work has practical applications of societal impact, such as drug discovery. However, our main focus has been on general graph generation and in order to have a stronger impact on drug discovery we need to carefully specialize the method for this purpose. For example, we need to improve the druglikeness, or other desired chemical properties, of the generated molecules.

Acknowledgements

This work was partially supported by NSF award DMS 2124913 and the Kunshan Municipal Government research funding.

Chapter 4

Competition and Collusion in Two-Sided Markets with an Outside Option

Algorithmic price collusion facilitated by artificial intelligence (AI) algorithms raises significant concerns. This chapter examines how AI agents using Q-learning engage in tacit collusion in two-sided markets. Our experiments reveal that AI-driven platforms achieve higher collusion levels compared to Bertrand competition. Increased network externalities significantly enhance collusion, suggesting AI algorithms exploit them to maximize profits. Higher user heterogeneity and greater utility from outside options generally reduce collusion, while higher discount rates increase it. Tacit collusion remains feasible even at low discount rates. To mitigate collusive behavior and inform potential regulatory measures, we propose incorporating a penalty term in the Q-learning algorithm.

This chapter is organized as follows: §4.1 provides background on the problem of interest for this chapter, reviews related work, and highlights our main contributions; §4.2 establishes the economical model describing the multi-sided market; §4.3 explains the simulation framework; §4.4 presents numerical results and investigates market collusive behavior using the simulation results; §4.5 examines the economic implications; §4.6 concludes the chapter and describes potential future research directions; and §C includes details of the fitting method, further sensitivity analysis for the simulation, and additional simulation results.

4.1 Introduction

Algorithmic price collusion occurs when economic agents set prices using artificial intelligence (AI) algorithms. Through repeated interactions, these agents learn that tacit collusion is optimal, as noted by Calvano et al. (2020a).[1] Economists and antitrust authorities have expressed significant concerns about this form of collusion. The OECD (2017) raised concerns about pricing algorithms learning to collude via tacit coordination. Assad et al. (2024) suggested that algorithmic pricing in Germany's retail gasoline market increased price margins by approximately 15%. U.S. Senator Amy Klobuchar introduced the S.3686 - Preventing Algorithmic Collusion Act of 2024 to curb anticompetitive behavior through algorithmic pricing using nonpublic competitor data.

Recent experiments (Calvano et al. (2020b), Klein (2021)) have demonstrated that collusion can be achieved in Bertrand and Stackelberg competition models by simulated economic agents using Q-learning, a benchmark reinforcement learning algorithm. Building on these findings, this study experimentally investigates AI-driven platforms using Q-learning in a repeated two-sided platform competition game. We illustrate how these AI agents facilitate collusion compared to Bertrand competition. Our focus is particularly on the impact of network externalities on collusion.

In our model of repeated two-sided platform competition, multiple horizontally differentiated platforms compete to serve buyers and sellers, collectively referred to as users. These platforms repeatedly interact and independently choose prices using Q-learning, with the last period price as the state variable. This implies that platforms have bounded memory and employ one-memory strategies (Barlo et al., 2009). In each repetition, users can choose to join one of the platforms or opt for the outside

[1] Tacit collusion happens when firms coordinate behavior without explicit communication. Du and Tanriverdi (2023); Bertomeu et al. (2021) have documented evidence of this in the U.S. multihospital system and automotive industry.

option. Buyers who join a platform receive network externality benefits proportional to the number of buyers (within-side externalities) and sellers (cross-side externalities) on the same platform. Sellers who join the market receive both types of externalities as well.

Our experiments show that even with zero network externalities, AI-driven platforms achieve higher collusion levels compared to those reported by Calvano et al. (2020b) for Bertrand competition. This is likely due to the larger action space, which allows more information exchange. Furthermore, increased network externalities lead to significantly high collusion levels, suggesting AI-driven platforms can leverage these externalities to boost profits. In particular, algorithmic pricing can increase collusion in markets with significant positive within-side externalities (e.g., online/cloud gaming) and positive cross-side externalities (e.g., video streaming, social media).

Our findings indicate that higher user heterogeneity or greater utility from the outside option generally decrease collusion levels, except in certain local regions. In contrast, collusion levels typically rise with higher discount rates, especially in the presence of significant network externalities. Additionally, tacit collusion remains feasible even at very low discount factors. This contrasts with traditional literature on firms' collusion without AI agents, which suggests that collusion is feasible only at high discount factors (Tirole, 1988; Obara and Zincenko, 2017).

Finally, we propose reducing collusion through a penalty term in the Q-learning algorithm.

Related Literature. There is a growing literature on algorithmic price collusion, with a particular emphasis on using numerical simulations to show that Q-learning results in tacit collusion. Waltman and Kaymak (2008) showed that firms using Q-learning in repeated Cournot oligopoly games produce lower quantities than the competitive Nash equilibrium. Calvano et al. (2020b) showed that Q-learning firms choose high prices in repeated Bertrand games and learn strategies consistent with

tacit collusion. Similar work was done by Klein (2021) for repeated Stackelberg games. Assad et al. (2024) is the first work that uses real-life data to show that firms may increase price margins with the adoption of algorithmic pricing. Our work extends the numerical understanding of algorithmic pricing, particularly in two-sided markets with network externalities.

Studies by Johnson et al. (2023) and Brero et al. (2022) on single platforms with AI-driven sellers show how platform-designed rules can promote competition and reduce collusion. Nevertheless, this setting does not apply to ours, where multiple platforms apply AI algorithms.

Our model of repeated two-sided platform competition uses the model in Chica et al. (2023), which, in turn, builds upon previous models by White and Weyl (2016); Tan and Zhou (2021); Chica et al. (2021). These models analyze network externality effects on equilibrium outputs. Our simulations use their insights to study the impact of these externalities on the collusive levels.

Ruhmer (2010) finds that higher cross-side externalities make collusion harder to sustain, when following the model of Armstrong (2006) without AI agents. This is consistent with our numerical results, even though we consider algorithmic pricing and follow the model of Chica et al. (2023).

Theoretical work in economics on algorithmic price collusion includes Brown and MacKay (2023), which demonstrates that simple pricing algorithms can elevate price levels. Additionally, Arslantas et al. (2024) illustrates how a sophisticated agent can exploit another agent using a naive version of Q-learning, provided the former agent knows the algorithm being used.

4.2 Review of Our Economic Framework

We introduce the economics framework used in our experiments. Section 4.2.1 presents the baseline platform competition game. Section 4.2.2 extends the latter model to an infinite repeated game.

4.2.1 The Baseline Platform Competition Game

The baseline platform competition game consists of two stages. In stage I, a set of horizontally differentiated platforms strategically choose prices to maximize profits. In stage II, given the prices determined by the platforms, users on each of the two sides of the market choose whether to participate or not and if they participate they also choose which platform to join. The solution concept for the baseline game is backward induction. More specifically, N platforms provide service options for users on two sides of a market, buyers and sellers. Users in these two sides of a market are denoted with $k \in \{b, s\}$, where b and s represent buyers and sellers, respectively. These users have $N + 1$ choices, where $N \geq 2$. They can either opt out of the market by choosing the outside option, or join one of the N horizontally differentiated platforms, each one denoted with $i \in [N] := \{1, \ldots, N\}$. The users on side k opting out of the market receive a deterministic outside option utility $u_k^{(0)} \in \mathbb{R}$. The users on side k joining platform $i \in [N]$ receive a deterministic utility

$$u_k^{(i)} := \phi_k(x_b^{(i)}, x_s^{(i)}) - p_k^{(i)},$$

where $p_k^{(i)}$ is the price paid by the user to access services provided by the platform i; $x_k^{(i)}$ is the total mass of users on side k joining platform i; and

$$\phi_k(x_b^{(i)}, x_s^{(i)}) := \phi_{kb} x_b^{(i)} + \phi_{ks} x_s^{(i)}, \quad \text{with} \quad \phi_{kb}, \phi_{ks} \in \mathbb{R},$$

is the network externality function that captures the network benefits users enjoy by joining platform i. The network externalities are captured by the following linear transformation

$$(\phi_b(x_b^{(i)}, x_s^{(i)}), \phi_s(x_b^{(i)}, x_s^{(i)}))^T = \Phi \boldsymbol{x}^{(i)}, \quad \text{where} \quad \Phi = \begin{bmatrix} \phi_{bb} & \phi_{bs} \\ \phi_{sb} & \phi_{ss} \end{bmatrix}. \tag{4.1}$$

To save space, we write $\Phi = [\phi_{bb}, \phi_{bs}; \phi_{sb}, \phi_{ss}]$ when specifying choices of Φ. The endogenous mass of users on each side of the market subscribed to platform i is denoted by $\boldsymbol{x}^{(i)} := (x_b^{(i)}, x_s^{(i)}) \in [0,1]^2$ and the mass of users not participating in the market is denoted by $\boldsymbol{x}^{(0)} := (x_b^{(0)}, x_s^{(0)}) \in [0,1]^2$. Assuming all users have Gumbel-distributed idiosyncratic preferences with parameters (μ_k, β_k), $\mu_k \in \mathbb{R}$ and $\beta_k > 0$,[2] the quantities $x_k^{(i)}$ are determined through a maximization process conducted by users on side k who solve the following equations (see Chica et al. (2023)):[3]

$$x_k^{(i)} = 1 - \left(1 + \exp\left(u_k^{(i)}/\beta_k - \ln\left(\sum_{j=0,1,\cdots,N, j\neq i} e^{u_k^{(j)}/\beta_k}\right)\right)\right)^{-1} \quad i \in [N] \cup \{0\},\ k \in \{b, s\}. \tag{4.2}$$

The platforms incorporate (4.2) into their profit maximization problem as follows, where $\pi^{(i)}$ denotes the profit of platform i and Π_{tot} denotes the total profits of N colluding platforms:

(i) when competing, they solve

$$\max_{\{p_b^{(i)}, p_s^{(i)}\}} \pi^{(i)}(p_b^{(i)}, p_s^{(i)}), \text{ where } \pi^{(i)}(p_b^{(i)}, p_s^{(i)}) := x_b^{(i)} p_b^{(i)} + x_s^{(i)} p_s^{(i)}; \tag{4.3}$$

[2]The Gumbel-distribution parameter β_k measures the standard deviation of ϵ_k^i and it captures the degree of heterogeneity in users' tastes. Unlike β_k, μ_k does not affect the equilibrium output of the model (see (4.2)).

[3]We note that the model presented here is equivalent to a model in which users have Logistic-distributed preferences for the platforms, and the outside option utility is deterministic.

(ii) when colluding, they solve

$$\max_{p_b,p_s} \Pi_{\text{tot}}(p_b,p_s), \text{ where } \Pi_{\text{tot}}(p_b,p_s) := \sum_{i=1}^{N} \left(x_b^{(i)} p_b + x_s^{(i)} p_s \right). \tag{4.4}$$

The maximizer of (4.3) is called the *competitive Nash Equilibrium* (CNE) and the corresponding equilibrium quantities are denoted by $p_k^{(i),*}$ and $x_k^{(i),*}$ for $i \in [N] \cup \{0\}$ and $k \in \{b,s\}$. The maximizer of (4.4) is called the *collusive equilibrium* (CE) and the corresponding equilibrium quantities are denoted by $p_k^{(i),\mathrm{C}}$ and $x_k^{(i),\mathrm{C}}$, $i \in [N] \cup \{0\}$, $k \in \{b,s\}$. In the symmetric equilibrium, $p_k^{(i),*} = p_k^*$, $p_k^{(i),\mathrm{C}} = p_k^{\mathrm{C}}$, $x_k^{(i),*} = x_k^*$ and $x_k^{(i),\mathrm{C}} = x_k^{\mathrm{C}}$ for all $i \in [N]$. Propositions 3.2 and 3.4 in Chica et al. (2023) provide first-order conditions for solving p_k^* and p_k^{C}. Similarly, Propositions 3.3 and 3.5 in the same work provides sufficient conditions for the existence and uniqueness of symmetric CNE and CE equilibria. The symmetric CNE and CE individual platform profits are respectively defined by

$$\pi^* := \pi^{(i)}(p_b^*, p_s^*) \quad \text{and} \quad \pi^{\mathrm{C}} := \frac{\Pi_{tot}(p_b^{\mathrm{C}}, p_s^{\mathrm{C}})}{N}. \tag{4.5}$$

4.2.2 The Infinite Repeated Game

The infinite repeated game consists of a sequence of games, where at time $t \in \mathbb{N} \cup \{0\}$, platforms and users interact following the rules of the baseline platform competition game, introduced in Section 4.2.1, and additional ones. At each time step t, we use the same notations as above, but with a subscript t. We assume that users on all sides are *myopic*, i.e., they make decisions to maximize the utility at current time t by solving (4.2) which depends solely on the current prices observed in the market. We further assume that platforms compete and act strategically and determine the charged prices to maximize the total discounted future rewards at every step t based on the past market states, which we clarify next after introducing some notation and

definitions. Given a discounting rate $\delta \in (0,1)$, we define the total discounted future rewards at time t for platform i by

$$r_t^{(i)} := \sum_{\tau=0}^{\infty} \delta^\tau \mathbb{E}[\pi_{t+\tau}^{(i)}], \quad \text{where} \quad \pi_{t+\tau}^{(i)} = \sum_{k\in\{b,s\}} x_{t+\tau,k}^{(i)} p_{t+\tau,k}^{(i)} \tag{4.6}$$

and $x_{t+\tau,k}^{(i)}$ is the mass of users on side k joining platform i at time $t+\tau$, and $p_{t+\tau,k}^{(i)}$ is the price that platform i charges on side k at time $t+\tau$. Note that from (4.2), $x_{t+\tau,k}^{(i)}$ is a function of all platforms prices at time $t+\tau$. Furthermore, this observation and (4.3) imply that $\pi_{t+\tau}^{(i)}$ can be written as a function of all platforms prices at time $t+\tau$, that is,

$$\pi_{t+\tau}^{(i)} = \pi^{(i)}(\boldsymbol{p}_{t+\tau}^{(1)}, \boldsymbol{p}_{t+\tau}^{(2)}, \cdots, \boldsymbol{p}_{t+\tau}^{(N)}). \tag{4.7}$$

From the viewpoint of platform i, the policies of all other platforms are unknown, so their present and future prices are considered random variables.[4]

Each platform needs to strategically charge prices in order to maximize the expected total discounted future rewards (4.6). A common method to optimize the expectation of the total discounted future reward is Q-learning, which we introduce in § 4.3.1.

For $t \in \mathbb{N} \cup \{0\}$, denote by $\boldsymbol{p}_t := (\boldsymbol{p}_t^{(1)}, \ldots, \boldsymbol{p}_t^{(N)})$ the vector of prices chosen by the N platforms at time t, where $\boldsymbol{p}_t^{(i)} := (p_{t,b}^{(i)}, p_{t,s}^{(i)})$, $i \in [N]$. For $L \geq 1$, denoting previous time steps, and $t \geq L$, let

$$\boldsymbol{s}_{t,L} := (\boldsymbol{p}_{t-L}, \boldsymbol{p}_{t-L+1}, \ldots, \boldsymbol{p}_{t-1}) \quad \text{and} \quad H_{t,L} = \{\boldsymbol{s}_{t,L} \in \mathbb{R}^{2LN}\},$$

where one typically constrains $H_{t,L}$ to be a discrete set (see Section 4.3.2). The

[4]For $\tau = 0$, the market share for platform i, $x_{t,k}^{(i)}$, is a random variable since it depends on all the prices charged by all other platforms (see (4.2)). For $\tau > 0$, the future policies of platforms are random variables. Since the future prices depend on the future states, which depend on the platform policies, they are also random variables.

problem for each platform is to identify a policy

$$\sigma_t^{(i)} : H_{t,L} \longrightarrow \mathbb{R}^2$$

that inputs the current observed state $\boldsymbol{s}_{t,L}$ and outputs the charged price $\boldsymbol{p}_t^{(i)} \equiv (p_{t,b}^{(i)}, p_{t,s}^{(i)})$. During this infinitely repeated game, at each time step t, each platform i updates the policy $\sigma_t^{(i)}$ based on the observed data (the states and rewards) to refine this policy that aims to maximize the expected total discounted future reward. Moreover, at each time step t, each platform i uses the policy $\sigma_t^{(i)}$ to determine the charged prices $\boldsymbol{p}_t^{(i)}$. A particular framework for doing this is discussed in Section 4.3. Figure 4.1 demonstrates the different stages of the infinite repeated game.

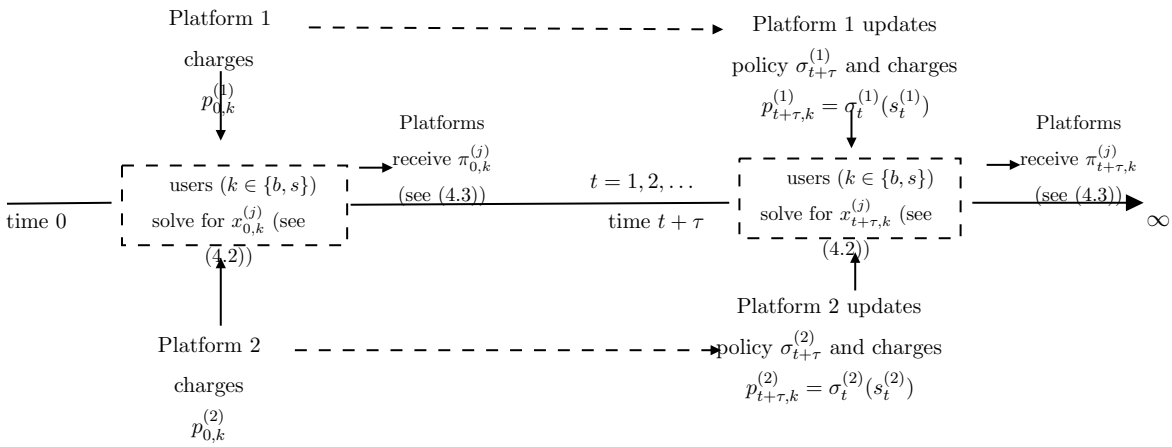

Figure 4.1: Demonstration of the infinite repeated game framework

4.3 Simulation Framework

We first review the framework of multi-agent reinforcement learning in Section 4.3.1. We then detail our simulation setting in Section 4.3.2, building upon the framework developed in Section 4.3.1.

4.3.1 Preliminaries: Multi-agent Reinforcement Learning

Multi-agent reinforcement learning considers N agents interacting in a dynamic environment. At each time $t \in \mathbb{N}$, each agent $i \in [N]$ observes a state $s_t^{(i)} \in \mathcal{S}$ and takes an action $a_t^{(i)} \in \mathcal{A}$, based on this observed state and following a policy $\sigma_t^{(i)} : \mathcal{S} \longrightarrow \mathcal{A}$, which could be either deterministic or stochastic. Here, $\mathcal{S}$ denotes the state space and $\mathcal{A}$ denotes the action space. Let $\sigma_t = (\sigma_t^{(1)}, \sigma_t^{(2)}, \cdots, \sigma_t^{(N)})$ and $A_t := (a_t^{(1)}, \cdots a_t^{(N)}) = (\sigma_t^{(1)}(s_t^{(1)}), \cdots \sigma_t^{(N)}(s_t^{(N)}))$ denote all policies and actions, respectively, at time t. We denote by $a_t^{(-i)}$, $\boldsymbol{p}_t^{(-i)}$, and $\sigma_t^{(-i)}$ the respective vectors of all actions $a_t^{(j)}$, prices $\boldsymbol{p}_t^{(j)}$, and policies $\sigma_t^{(j)}$with $j \neq i$. The agent collects a reward $\pi_t^{(i)}$, which is a random variable conditioned on the state $s_t^{(i)}$ and actions A_t. The state in the next time, $s_{t+1}^{(i)}$, is a random variable conditioned on the state $s_t^{(i)}$ and the actions A_t taken by all the agents in the current time t. Given a discounting rate $\delta \in (0, 1)$, at each time, each agent aims to find a policy in order to maximize the following expectation of the total discounted future reward given all observed states at time t:

$$\sum_{\tau=0}^{\infty} \mathbb{E}_{\pi,s,\sigma} \left[\delta^{\tau} \pi_{t+\tau}^{(i)}(s_{t+\tau}^{(i)}, A_{t+\tau}) \right]. \tag{4.8}$$

The expectation is needed due to the randomness in the rewards, the future states, and the future actions of all the agents.

Q-learning is a classic method for finding the policy that maximizes (4.8). It uses the Q-function of agent i at state s given an action a, which is defined by

$$Q^{(i)}(s, a, \sigma^{(i)}; \sigma^{(-i)}) :=$$
$$\sum_{\tau=0}^{\infty} \mathbb{E}_{\pi,s,\sigma} \left[\delta^{\tau} \pi_{t+\tau}^{(i)} \middle| s_t^{(i)} = \boldsymbol{s}, a_t^{(i)} = a, a_{t+u}^{(i)} = \sigma^{(i)}(s_{t+u}^{(i)}),\ u \geq 1,\ a_{t+v}^{(-i)} = \sigma^{(-i)}(s_{t+v}),\ v \geq 0 \right]. \tag{4.9}$$

Note that (4.9) differs from (4.8) by having agent i follow the given action a at time t instead of the policy $\sigma^{(i)}$, whereas in both formulations all other agents at times $t, t+1, \ldots,$ and agent i at times $t+1, t+2, \ldots,$ follow their policies.

We denote an optimal policy for agent i by $\sigma^{(i)*}$, which is hard to find. Q-learning overcomes this difficulty by carefully estimating the solution $Q^{(i)*}(s_t, a_t; \sigma^{(-i)})$ to the following Bellman equation

$$Q^{(i)*}(s_t, a_t; \sigma^{(-i)}) = \mathbb{E}_\pi \left[\pi(s_t, A_t)\right] + \delta \max_{a'} \mathbb{E}_{s_{t+1}} \left[Q^{(i)*}(s_{t+1}, a'; \sigma^{(-i)}) | A_t\right]. \quad (4.10)$$

It estimates $\sigma^{(i)*}$ using the following relationship between $\sigma^{(i)*}$ and $Q^{(i)*}(x, a; \sigma^{(-i)})$:

$$\sigma^{(i)*}(s) = \arg\max_a Q^{(i)*}(s, a; \sigma^{(-i)}). \quad (4.11)$$

We detail the methods for estimating the Q^*-function in (4.10) in the following section.

4.3.2 The Simulation Setup

We consider a market with two platforms, that is, we set $N = 2$.[5] At time t, each platform $i \in \{1, 2\}$ observes the following state $\boldsymbol{s}_t^{(i)} := \boldsymbol{p}_{t-1}$, which contains prices at the previous step. Platform i determines its prices $\boldsymbol{p}_t^{(i)}$ based on the observed state $\boldsymbol{s}_t^{(i)}$. At each time t, after all platforms have chosen prices $\boldsymbol{p}_t^{(i)}$, they receive the reward $\pi_t^{(i)} = \sum_{k \in \{b,s\}} x_{t,k}^{(i)} p_{t,k}^{(i)}$, where $x_{t,k}^{(i)}$ is solved using (4.2).

To simplify the computation, we allow platforms to choose from a discrete set of M prices. While it is common to expect that $p_k^* < p_k^{\mathrm{C}}$,[6] our model also allows the case $p_k^{\mathrm{C}} < p_k^*$. We further introduce the parameter $\epsilon = 0.1$ so the lowest price is slightly

[5] Since our model assumes two sides of the market, each of the N platforms must choose two prices. At each stage, our simulation estimates $2N$ different prices, and there are M^{2N} possibilities for the vector of prices, where M is the size of the set of price choices available to each platform. To make our simulations feasible, we choose $N = 2$.

[6] Note that Proposition 4.11 in Chica et al. (2023) provides sufficient conditions to guarantee that $p_k^* < p_k^{\mathrm{C}}$.

lower than $\min(p_k^*, p_k^{\mathrm{C}})$ and the highest one is slightly higher than $\max(p_k^*, p_k^{\mathrm{C}})$. For each $k \in \{b, s\}$, our set of prices is

$$\mathcal{P}_k := \Big\{ p_k^* - \epsilon(p_k^{\mathrm{C}} - p_k^*) + \frac{j}{M-1}(1+2\epsilon)(p_k^{\mathrm{C}} - p_k^*) \Big| j = 0, \cdots, M-1 \Big\}. \tag{4.12}$$

Note that the cardinality of the price space $|\mathcal{P}_k|$ is M for both $k = b$ and $k = s$. The overall state space (for both platforms) and the action space for each platform are respectively defined by

$$\mathcal{S} := (\mathcal{P}_b \times \mathcal{P}_s) \times (\mathcal{P}_b \times \mathcal{P}_s) \ \text{ and } \mathcal{A} := \mathcal{P}_b \times \mathcal{P}_s. \tag{4.13}$$

We note that the size of the state space is $|\mathcal{S}| = M^4$ and the size of the action space is $|\mathcal{A}| = M^2$.

Platform policy: We denote the estimation at time t of $Q^{(i)*}(\boldsymbol{s}, \boldsymbol{a}; \sigma^{(-i)})$ by $Q_t^{(i)}(\boldsymbol{s}, \boldsymbol{a})$, where $\boldsymbol{s} \in \mathcal{S}$, $\boldsymbol{a} \in \mathcal{A}$ and $i \in \{1, 2\}$ indexes the platform. Q-learning alternately estimates $Q_t^{(i)}$ and the stochastic policies at time t. We first assume that $Q_t^{(i)}$ is known and show how the platforms determine the stochastic policy at time t. We then explain the Q-learning estimation of $Q_t^{(i)}$. Instead of directly computing the policy as the maximum value in (4.11), Q-learning computes a softmax value using a temperature parameter $\mathcal{T}_t$. For this purpose, at time t and given a state $\boldsymbol{s}_t^{(i)} \in \mathcal{S}$ and the Q^*-function estimate, $Q_t^{(i)}$, the policy of platform i is the Boltzmann probability distribution:

$$P(\boldsymbol{a}_t^{(i)} = \boldsymbol{a} | \boldsymbol{s}_t^{(i)}) = \exp\Big(Q_t^{(i)}(\boldsymbol{s}_t^{(i)}, \boldsymbol{a})/\mathcal{T}_t\Big) / \sum_{\boldsymbol{a}' \in \mathcal{A}} \exp\Big(Q_t^{(i)}(\boldsymbol{s}_t^{(i)}, \boldsymbol{a}')/\mathcal{T}_t\Big). \tag{4.14}$$

We remark that all platforms independently determine their prices based on (4.14).[7]

[7] As $\mathcal{T}_t$ decreases, (4.14) increasingly focuses on the optimal action based on $Q_t^{(i)}$. When $\mathcal{T}_t \to 0$, the policy randomly selects between the actions that yield the maximal reward $Q_t^{(t)}$ with uniform

Q-learning estimation: At each time step, after determining the price following (4.14), platform i collects the reward $\pi_t^{(i)}$ defined by (4.7). Next, platform i updates the estimated values of the Q^*-function at the given state $\boldsymbol{s}_t^{(i)}$ and the selected action $\boldsymbol{a}_t^{(i)}$ with a learning rate α as follows:

$$Q_{t+1}^{(i)}(\boldsymbol{s}_t^{(i)}, \boldsymbol{a}_t^{(i)}) := (1-\alpha)Q_t^{(i)}(\boldsymbol{s}_t^{(i)}, \boldsymbol{a}_t^{(i)}) + \alpha\left(\pi_t^{(i)} + \delta \max_{\boldsymbol{a}} Q_t^{(i)}(\boldsymbol{s}_t^{(i)}, \boldsymbol{a})\right). \quad (4.15)$$

We remark that (4.15) is an approximation of (4.10) (see Watkins and Dayan (1992)).

We initialize the Q^*-function at $\boldsymbol{s} \in \mathcal{S}$ and $\boldsymbol{a} \in \mathcal{A}$ assuming that in all future states platform i charges $\boldsymbol{a}$ and all other platforms charge the prices in $\boldsymbol{s}$. Therefore, for platform i, state $\boldsymbol{s}$, a given action $\boldsymbol{a}$ and the price vector for platform $j \neq i$, which we denote by $\boldsymbol{p}^{(j)}$ and it is part of the state $\boldsymbol{s}$, the Q^*-function for platform i is initialized by

$$Q_0^{(i)}(\boldsymbol{s}, \boldsymbol{a}) = \sum_{\tau=0}^{\infty} \delta^\tau \pi^{(i)}(\boldsymbol{a}, \boldsymbol{p}^{(j)}) = \frac{\pi^{(i)}(\boldsymbol{a}, \boldsymbol{p}^{(j)})}{1-\delta}. \quad (4.16)$$

Parameter setup: We choose exponentially decaying temperature parameter $\mathcal{T}_t = \mathcal{T}_0 \lambda^t$ with $\mathcal{T}_0 := 1000/(1-\delta)$ and $\lambda = 1-10^{-7}$. This choice encourages exploration in the early stages and exploits optimality in the later stages. We choose both the idiosyncratic preference parameters and the outside option utilities to be the same on both sides of the market. Therefore we denote $\beta_k = \beta_b \equiv \beta_s$ and $u_k^{(0)} = u_b^{(0)} \equiv u_s^{(0)}$. We set the learning rate $\alpha = 0.15$, discount rate $\delta = 0.05$, idiosyncratic preferences $\beta_k = 1$, and outside option utility $u_k^{(0)} = -2$ for each $k \in \{b, s\}$. We choose a small value for δ, compared to the choice of the same parameter in Calvano et al. (2020b); Klein (2021), to emphasize that in our setting collusion is already present with a very small discount rate.

probabilities. In the simulation, we set $\mathcal{T}_0 = 1000$ to encourage exploration of possible actions, gradually decreasing it towards 0 to exploit optimal actions.

Reporting metric: We define the collusive level of platform i at time t as

$$\Delta_t^{(i)} := \frac{\pi_t^{(i)} - \pi^*}{\pi^{\mathrm{C}} - \pi^*}, \tag{4.17}$$

where we recall (see (4.5)) π^* and π^{C} respectively denote the CNE and CE equilibrium profits of the baseline platform competition game. When $\Delta_t^{(i)} = 0$, platform i's reward at time t equals the CNE level, π^*; whereas when $\Delta_t^{(i)} = 1$, it equals the CE level, π^{C}. Each simulation runs $T = 5 \times 10^8$ iterations and we report the overall collusive level in the last $K = 1,000$ steps as follows:

$$\tilde{\Delta} := \frac{1}{KN} \sum_{s=0}^{K-1} \sum_{i=1}^{N} \Delta_{T-s}^{(i)}. \tag{4.18}$$

4.4 Experimental Results

We report extensive numerical experiments using the setup of Section 4.3.2. Section 4.4.1 investigates the general dependence of $\tilde{\Delta}$, defined in (4.18), on the externality matrix Φ. Section 4.4.2 further explores the latter dependence for concrete and useful choices of Φ. Section 4.4.3 studies the dependence of $\tilde{\Delta}$ on the degree of heterogeneity in users' tastes, the outside option utility, and the discount rate. Lastly, Section 4.4.4 explores two interesting scenarios: 1) long-run asymmetric equilibria outperform the symmetric equilibrium, and 2) competition prices are larger than collusion prices for one side of the market. A supplementary analysis in Appendix C.1 implies that our numerical results are consistent with platforms learning tacit collusion and equilibrium strategies.

4.4.1 Dependence of the Collusive Level on the Network Externalities

We applied an additive model to infer the dependence of the collusive level, $\tilde{\Delta}$, on the externality matrix, Φ. We ran 2,500 simulations according to the setting described in Section 4.3.2. For each simulation, we randomly sampled the elements of the externality matrix Φ from independent normalized Gaussians (that is, $\phi_{kl} \sim N(0,1)$ for $k, l \in \{b, s\}$), and recorded the final collusive level, $\tilde{\Delta}$. In order to infer the dependence of $\tilde{\Delta}$ on Φ, we assume the following additive model:

$$\begin{aligned}\tilde{\Delta}(\Phi) = \Delta_0 &+ f_{bb}(\phi_{bb}) + f_{ss}(\phi_{ss}) + f_{bs}(\phi_{bs}) + f_{sb}(\phi_{sb}) + f_{bb,ss}(\phi_{bb}, \phi_{ss}) + f_{sb,bs}(\phi_{sb}, \phi_{bs}) \\ &+ f_{bb,bs}(\phi_{bb}, \phi_{bs}) + f_{bb,sb}(\phi_{bb}, \phi_{sb}) + f_{ss,bs}(\phi_{ss}, \phi_{bs}) + f_{ss,sb}(\phi_{ss}, \phi_{sb}) + \epsilon,\end{aligned} \tag{4.19}$$

where Δ_0 is the sample mean of $\tilde{\Delta}$, the next 4 functions (f_{bb}, f_{ss}, f_{bs}, f_{sb}) represent the univariate effects of the elements of Φ on $\tilde{\Delta}$, the last 6 functions ($f_{bb,ss}$, $f_{sb,bs}$, $f_{bb,bs}$, $f_{bb,sb}$, $f_{ss,bs}$, $f_{ss,sb}$) represent the bivariate effects of the elements of Φ on $\tilde{\Delta}$ and ϵ is an error term, encompassing higher-order multivariate effects. Since the equilibrium values π^* and π^C depend on Φ nonlinearly (see Section 4.2.1), the 10 functions, $f_{bb}, \cdots, f_{ss,sb}$, are nonlinear. We thus sequentially fit these functions using XGBoost (Chen and Guestrin, 2016), which is a popular non-parametric, nonlinear fitting method. To reduce the bias of the fitted functions, we alter the order of both the first four functions and the next six functions, during the sequential fitting procedure, and average the collusive level over the different orders. Appendix C.2 contains more details of implementing XGBoost.

We refer to Δ_0 as the baseline collusive level, whereas $\tilde{\Delta}$ is the collusive level. Our simulations show that Δ_0 is approximately 0.3. Next, we report our estimates for the univariate and bivariate effects of the elements of Φ on $\tilde{\Delta}$.

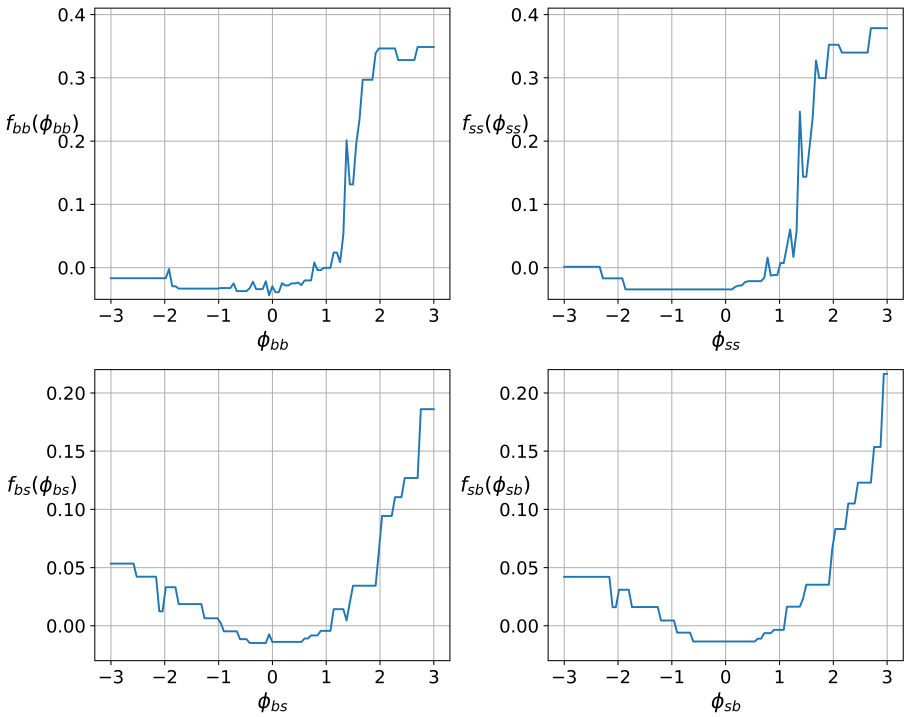

Figure 4.2: Demonstration of the dependence of the four fitted univariate functions on the externalities. Top left: $f_{bb}(\phi_{bb})$; Top right: $f_{ss}(\phi_{ss})$; Bottom left: $f_{bs}(\phi_{bs})$; Bottom right: $f_{sb}(\phi_{sb})$.

Figure 4.2 illustrates the fitted functions f_{bb}, f_{ss}, f_{bs} and f_{sb}, which capture the univariate effect of each entry in the externality matrix. The top two subfigures demonstrate the univariate effect of the within-side externalities, (ϕ_{bb} and ϕ_{ss}). In this case, the collusive level is close to zero when these externalities are less than 1, then increases sharply when these externalities increase from 1 to 2, and it is approximately flat when these externalities are above 2 with a possible increase of the collusive level when the absolute values of the negative externalities increase. We remark that we cannot confidently conclude the latter increase from the current experimental results, but latter experiments in Section 4.4.2 support such an increase, especially when considering lower values of ϕ_{bb} and ϕ_{ss}. The bottom two subfigures demonstrate the univariate effect of the cross-side externalities (ϕ_{bs} and ϕ_{sb}). In this case, the

dependence of the collusive level on the externalities is depicted by a J-shape function with a minimum when the externality is around zero. We thus note that in order to minimize the level of the algorithmic collusion, we would need to bound the values of the within-side externalities and the absolute values of the cross-side externalities. In our particular experimental setting, the desired bound is 1. In general, we expect there can be two different upper bounds for the within-side and cross-side externalities and they depend on the chosen parameters, in particular, $\{\delta, \beta_k, u_k^{(0)}\}$.

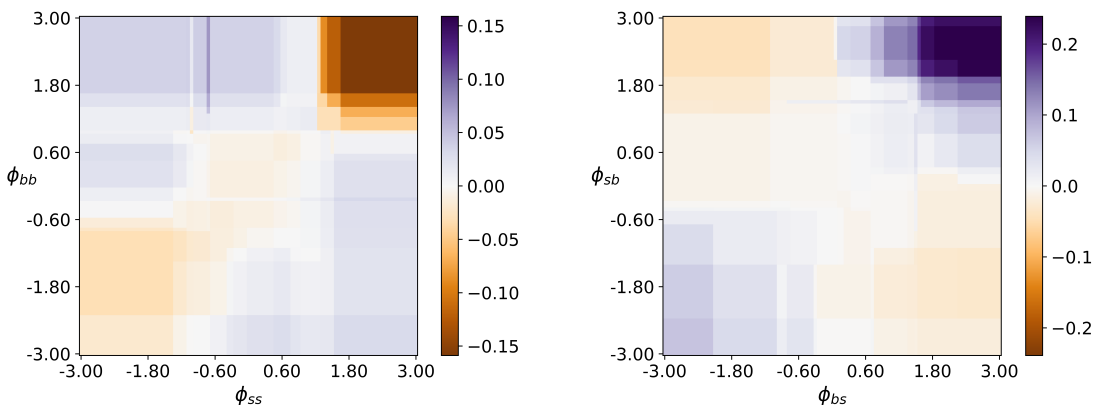

Figure 4.3: Heatmaps of the fitted functions $f_{bb,ss}(\phi_{bb}, \phi_{ss})$ (left) and $f_{sb,bs}(\phi_{bs}, \phi_{sb})$ (right), which capture the bivariate effect between ϕ_{bb} and ϕ_{ss}, and between ϕ_{bs} and ϕ_{sb}, respectively.

Figure 4.3 demonstrates the fitted functions $f_{bb,ss}$ and $f_{sb,bs}$, which capture the bivariate effects on collusion of the main diagonal and off-diagonal elements in the externality matrix. We present the images of these functions as heatmaps over their planar domains. For example, in the left-hand subfigure the domain is described by the within-side externality variables ϕ_{bb} and ϕ_{ss} and the collusive level is depicted by a heatmap, changing from purple (highly positive) to orange (highly negative). The left-hand subfigure implies that when the within-side externalities, ϕ_{bb} and ϕ_{ss}, are both large, they result in the minimal value of the bivariate effect, which is negative. In this regime, the collusive level, resulting from both the univariate and bivariate

effects, remains positive (recall that the univariate effect is demonstrated in the top subfigures of Figure 4.2). Similarly, when ϕ_{bb} and ϕ_{ss} are both sufficiently negative, their bivariate effect reduces the collusive level, albeit by a small amount. We remark that both the univariate and bivariate contributions in this case are rather small and it is hard to predict their combined effect from this experimental result, but another experiment in Section 4.4.2 indicates that they cancel each other. The right-hand side subfigure indicates that the bivariate component of the cross-side externalities reduces the collusive level when these externalities are large in absolute values and have opposite signs. On the other hand, it increases the collusive level when the cross-side externalities have the same sign and have sufficiently large absolute values. The rate of increase is larger when they are both positive.

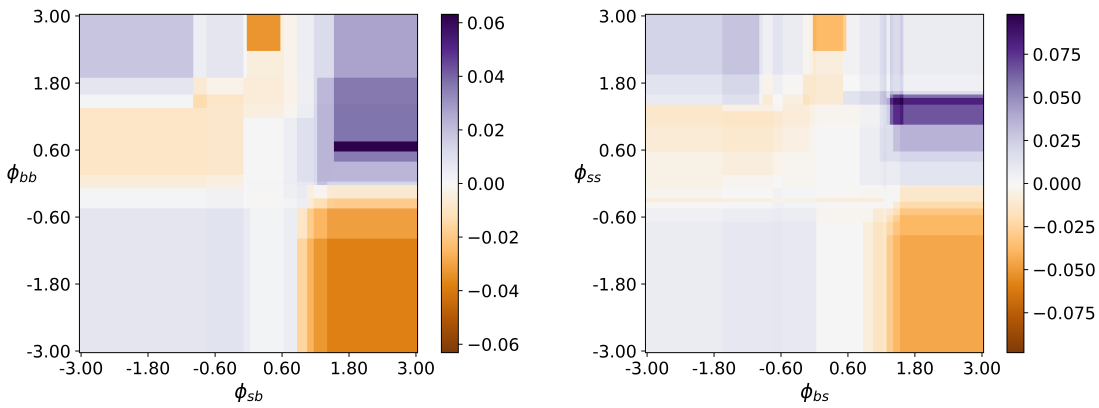

Figure 4.4: Heatmaps of the fitted functions $f_{bb,sb}(\phi_{bb}, \phi_{sb})$ (left) and $f_{ss,bs}(\phi_{ss}, \phi_{bs})$ (right), which capture the bivariate effect between ϕ_{bb} and ϕ_{sb}, and between ϕ_{ss} and ϕ_{bs}, respectively.

Figure 4.4 demonstrates the bivariate effect on collusion when both buyers and sellers benefit from population joining the market on either side b or s (but not both at the same time). That is, it demonstrates the bivariate effect for ϕ_{bb} and ϕ_{sb} when considering side b (left) and the bivariate effect for ϕ_{ss} and ϕ_{bs} when considering side s (right). The two subfigures are very similar and we thus only discuss the left one, with

the variables ϕ_{bb} and ϕ_{sb}. The bottom-right corner of this subfigure implies that if ϕ_{sb} sufficiently large and ϕ_{bb} is sufficiently negative, the bivariate effect on the collusive level is negative. In this regime, the collusive level, resulting from both the univariate and bivariate effects, remains positive (recall that the univariate effect is demonstrated in the top-left and bottom-right subfigures of Figure 4.2). On the other hand, the top-right corner in Figure 4.4 shows that when ϕ_{bb} and ϕ_{sb} are both large the bivariate effect on the collusive level is positive.

Figure 4.5 illustrates the bivariate effect on collusion when either buyers or sellers (but not both at the same time) benefit from population joining the market on sides b or s. That is, it demonstrates the bivariate effect for ϕ_{bb} and ϕ_{bs} when considering only buyers (left subfigure) and the bivariate effect for ϕ_{ss} and ϕ_{sb} when considering only sellers (right subfigure). The two subfigures are very similar and we thus only discuss the left one, with the variables ϕ_{bb} and ϕ_{bs}. We notice that when the ϕ_{bb} is sufficiently large and ϕ_{bs} is sufficiently negative, the bivariate effect on the collusion is negative. In this regime, the collusive level, resulting from both the univariate and bivariate effects, remains positive (the univariate effect is demonstrated in the left subfigures of Figure 4.2). We further notice that when ϕ_{bs} is sufficiently large and ϕ_{bb} is sufficiently negative the bivariate effect on the collusion is also negative, but smaller than the latter one. Similarly, the collusive level, resulting from both the univariate and bivariate effects, remains positive. On the other hand, when both ϕ_{bb} and ϕ_{bs} are sufficiently large, the bivariate effect on the collusive level is positive.

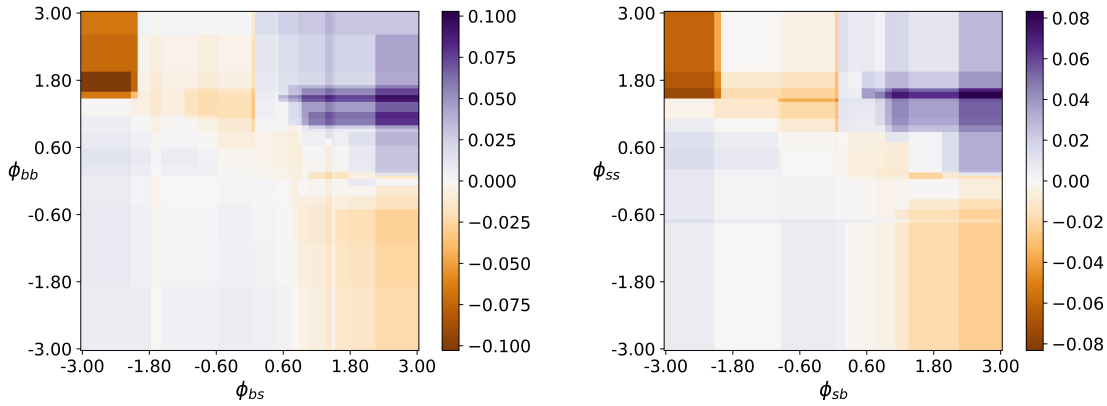

Figure 4.5: Heatmaps of the fitted functions $f_{bb,bs}(\phi_{bb}, \phi_{bs})$ (left) and $f_{ss,sb}(\phi_{ss}, \phi_{sb})$ (right), which capture the bivariate effect between ϕ_{bb} and ϕ_{bs}, and between ϕ_{ss} and ϕ_{sb}, respectively.

4.4.2 A Study of the Collusive Level under Special Network Externalities

We assume special parameterizations of the network externality matrices, Φ, and explore the dependence of $\tilde{\Delta}$ on any such Φ. This allows us to track more carefully the dependence of $\tilde{\Delta}$ on Φ in some special settings. For each specific Φ, we ran 100 simulations. Our figures present the dependence of the overall collusive level on the elements of Φ, where their main curves represent the average of the collusive levels from the 100 runs and their shaded areas represent the uncertainty level, which was computed using bootstrapping with a 99% confidence interval.

Figure 4.6 investigates the dependence of the collusive level on the within-side externalities in two controlled settings. In the first setting (left panel) $\Phi = [\phi_{bb}, 0; 0, 0]$, and in the second one (right panel) $\Phi = [\phi_{bb}, 0; 0, \phi_{bb}]$. In both cases $\phi_{bb} \in [-6, 2]$.

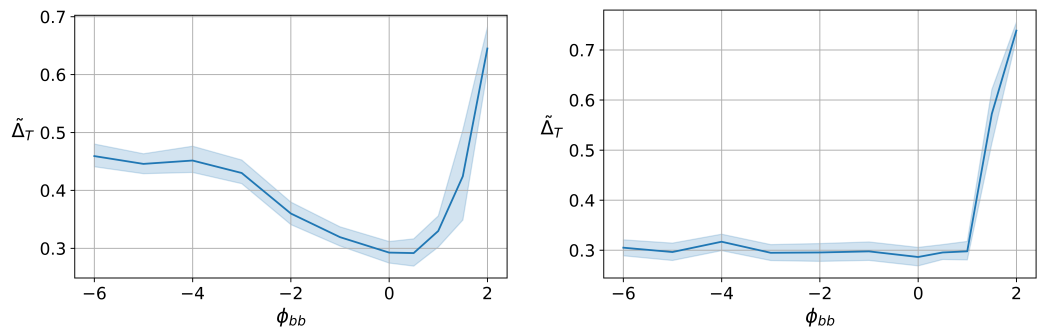

Figure 4.6: Collusive level with varying ϕ_{bb}: $\Phi = [\phi_{bb}, 0; 0, 0]$ (left) and $\Phi = [\phi_{bb}, 0; 0, \phi_{bb}]$ (right).

In the left panel, the collusive level forms a J-shape, where it decreases on $[-6, 0]$ and increases sharply on $[0.5, 2]$. The minimum value of the collusive level is achieved when $\phi_{bb} \approx 0.5$ and it is slightly below the baseline collusive level. Note that this subfigure indicates a similar behavior of the collusive level to its univariate effect shown in the top left panel of Figure 4.2. Indeed, in this case, the collusive level depends on the single variable ϕ_{bb}, so the other univariate and bivariate functions are irrelevant. However, the minimal value of the univariate effect in the top left panel of Figure 4.2 is around zero, since it is separate from the baseline collusive level Δ_0. By adding Δ_0 to this univariate effect, we obtain a function similar to the collusive level described in the left panel of Figure 4.6. We remark that in the experiments of Section 4.4.1, our domain was restricted by the underlying Gaussian model and thus the domain in Figure 4.2 is narrower than that of Figure 4.6.

In the right panel, the collusive level sharply increases when ϕ_{bb} exceeds 1. This behavior can be explained using our previous findings. Indeed, as shown in Figures 4.2 and 4.3, when the within-side externalities, ϕ_{bb} and ϕ_{ss}, are both large, the univariate effect is more significant than the negative bivariate effect, resulting in a significant increase. We also notice that the collusive level remains flat and around Δ_0 when $\phi_{bb} = \phi_{ss}$ falls below 1. This observation also confirms our findings in the previous

section. Indeed, Figures 4.2 and 4.3 indicate that when ϕ_{bb} and ϕ_{ss} are both sufficiently negative, the (positive) univariate and (negative) bivariate effects cancel each other.

Figure 4.7 investigates the dependence of the collusive level on the cross-side externalities in two controlled settings. In the first setting (left panel) $\Phi = [0, \phi_{bs}; 0, 0]$, and in the second one (right panel) $\Phi = [0, \phi_{bs}; \phi_{bs}, 0]$. In both cases $\phi_{bs} \in [-2.5, 3]$.

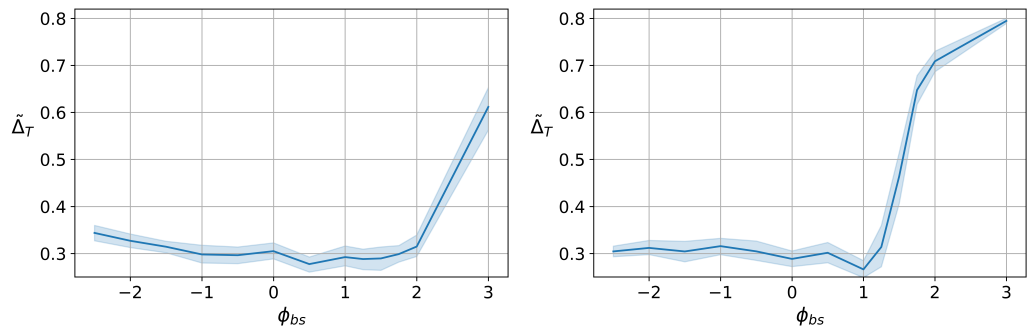

Figure 4.7: Collusive level with varying ϕ_{bs}: $\Phi = [0, \phi_{bs}; 0, 0]$ (left) and $\Phi = [0, \phi_{bs}; \phi_{bs}, 0]$ (right).

In the left panel, the collusive level increases when the cross-side externality exceeds 2, and it slightly decreases when the same externality falls below -1. This observation aligns with our findings in the previous section. Indeed, as shown by the bottom panels in Figure 4.2, the collusive level increases as ϕ_{bs} increases in absolute value with values above 1.

In the right panel, the collusive level increases when the cross-side externalities exceed 1. It has a sharper increase than the one in the left panel. These observations agree with the findings of the previous section. Indeed, Figures 4.2 and 4.3 show that the univariate and bivariate effects of ϕ_{bs} and ϕ_{sb} are both positive when $\phi_{bs} = \phi_{sb} > 1$. Furthermore, Figure 4.3 shows the positive bivariate effect between ϕ_{bs} and ϕ_{sb}, which explains the sharper increase in the right panel. The dependence of the collusive level in the right panel on smaller values of ϕ_{bs}, which are not shown in this figure, is rather unique and thus deferred to Section 4.4.4.

Figure 4.8 investigates the dependence of the collusive level on the bivariate effects between the within- and cross-side externalities in two controlled settings. For simplicity, we fix the cross-side externality and vary the within-side externality. In the first setting (left panel) $\Phi = [\phi_{bb}, 3; 0, 0]$, and in the second one (right panel) $\Phi = [\phi_{bb}, 3; 0, \phi_{bb}]$. In both cases $\phi_{bb} \in [-6, 2]$.

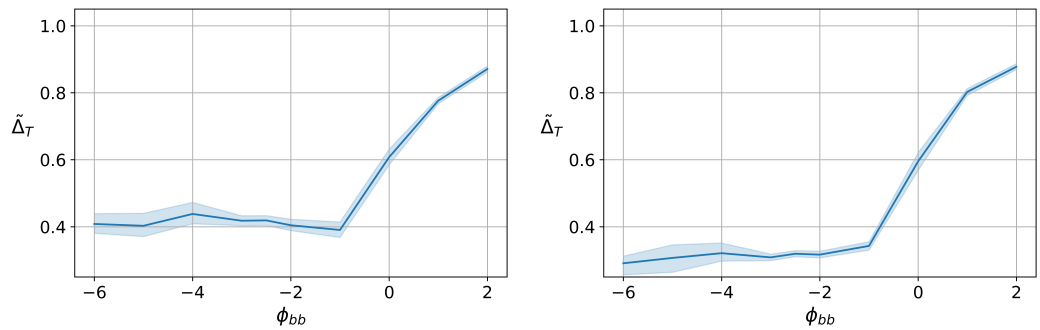

Figure 4.8: Collusive level with varying ϕ_{bb}: $\Phi = [\phi_{bb}, 3; 0, 0]$ (left) and $\Phi = [\phi_{bb}, 3; 0, \phi_{bb}]$ (right).

In the left panel, the collusive level is flat when the within-side externality falls below -1 and increases sharply as it exceeds -1. This increase can be explained when ϕ_{bb} exceeds 0 by the univariate effect for ϕ_{bb} shown in Figure 4.2. On the other hand, the increase in $[-1, 0]$ can be explained by the bivariate effect between ϕ_{bb} and ϕ_{bs}, shown in Figure 4.5. Indeed this bivariate effect increases with respect to ϕ_{bb} in $[-1, 0]$ when $\phi_{bs} = 3.0$.

In the right panel, the collusive level follows a similar pattern as in the left one. This follows from a similar explanation as above, where one should also note that when $\phi_{bs} = 3$, the bivariate effect between ϕ_{ss} and ϕ_{bs} increases with respect to ϕ_{ss} in $[-1, 0]$, as shown in Figure 4.4. Additionally, when $\phi_{bb} < 0$, the collusive level in the right panel is lower than that in the left panel (it is easiest to see this for $\phi_{bb} < -1$). We clarify this observation in view of the findings of Section 4.4.1 as follows. We note that according to the right panel of Figure 4.4 , when $\phi_{bs} = 3$, the bivariate

effect between ϕ_{ss} and ϕ_{bs} is negative when $\phi_{ss} < 0$, therefore the collusive level in the right panel is expected to be lower than the collusive level in the left panel when $\phi_{bb} = \phi_{ss} < 0$.

We make some additional remarks comparing Figures 4.6 and 4.8. The left panel in Figure 4.6 shows that the collusive level decreases with respect to ϕ_{bb}, when $\phi_{bb} < 0$ and $\phi_{bs} = 0$. On the other hand, in the left panel of Figure 4.8, the collusive level is flat or increases with respect to ϕ_{bb} when $\phi_{bb} < 0$ and $\phi_{bs} = 3$. This behavior can be explained by the contribution from the bivariate effect between ϕ_{bb} and ϕ_{bs} when ϕ_{bb} is negative. Indeed, this bivariate effect is almost flat with respect to ϕ_{bb} when $\phi_{bs} = 0$, but is increasing with respect to ϕ_{bb} when $\phi_{bs} = 3.0$ (see left panel of Figure 4.5). A similar comparison can be made for the right panels in Figures 4.6 and 4.8, and the explanation similarly follows from the right panel of Figure 4.4 and the left panel of Figure 4.5.

4.4.3 A Study of the Collusive Level under Special Market Parameters

We explore the dependence of the collusive level on the market parameters β_k, $u_k^{(0)}$ and δ. In each experiment, we fix two of the latter parameters, using the setup described in Section 4.3.2, and the matrix Φ, where its choices change with the experiments, and vary the remaining parameter. For each experiment, we ran 100 simulations, averaged the collusive levels among the 100 runs and computed the uncertainty levels using bootstrapping with a 99% confidence interval. Our figures present the averaged collusive level as a function of one parameter, where the shaded areas represent the uncertainty level.

Figure 4.9 investigates the dependence of the collusive level on the idiosyncratic preference parameter β_k, while considering two different choices of the externality

matrix Φ: A symmetric one, where $\Phi = [1, 0; 0, 1]$ (left panel) and an asymmetric one, where $\Phi = [0, 1; -1, 0]$ (right panel). In both cases, we vary the idiosyncratic preference parameters and let $\beta_k \in [0.2, 6]$. In both panels, the collusive level sharply decreases when β_k is sufficiently small. In particular, the collusive level is high only when the degree of heterogeneity in users' tastes is sufficiently small. Section 4.5.1 interprets this behavior. We note that in the left panel the sharp decrease stops when β_k exceeds 0.9, compared to the right panel where this happens ses once β_k exceeds 0.5. Furthermore, we note that in both subfigures, the collusive levels remains almost flat, at a value slightly below the collusive level $\Delta_0 = 0.3$, as the degree of heterogeneity in users' tastes exceeds 1. More experiments varying β_k with different choices of Φ are presented in Appendix C.3.

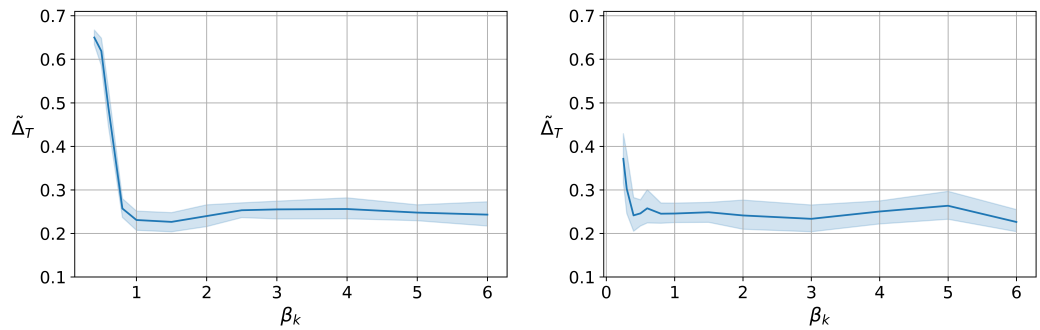

Figure 4.9: Collusive level with varying $\beta_b = \beta_s \in [0.2, 6]$ and different matrices Φ, where $\Phi = [1, 0; 0, 1]$ (left) and $\Phi = [0, 1; -1, 0]$ (right).

Figure 4.10 demonstrates the dependence of the collusive level on the outside option utility $u_k^{(0)}$ with the following choices for the externality matrix: $\Phi = [0, 0; 0, 0]$ (top left panel), $\Phi = [1, 0; 0, 1]$ (top right panel), $\Phi = [0, 1; 1, 0]$ (bottom left panel), and $\Phi = [1, 1; 1, 1]$ (bottom right panel). In all four panels, we observe a main trend of decrease of the collusive level as a function of the outside option utility over a sufficiently large domain. In all of these examples when $u_k^{(0)}$ is sufficiently small, the collusive level is at least 0.55, and when $u_k^{(0)}$ is sufficiently large, the collusive level

is around 0.3, which is near the baseline level. The most significant reduction of the collusive level happens in a narrow range and the location of this significant decrease appears to be determined by the externalities as follows. It tends to move to the left when the network externalities are small and to the right when they are large. Additional examples in Figure C.4 support this conclusion. In addition, we observe that both left subfigures exhibit another small region of increase to the baseline level after the region of sharp decrease. This is not the case for the right subfigures. We also note a similar phenomenon in Figure C.4. It seems that an increase to the baseline level after a sharp decrease occurs in cases of sufficiently small externalities, where the threshold on externalities required to guarantee such a short increase is smaller for within-side externalities than for cross-side externalities. For example, considering cases of both Figures 4.10 and C.4, this short increase is observed at $\Phi = [0, 1; 1, 0]$ but not at $\Phi = [0, 2; 2, 0]$, and is observed at $\Phi = [1, 0; 0, 0]$ but not at $\Phi = [1, 0; 0, 1]$. Lastly, it follows from the bottom right subfigure of Figure 4.10 and cases of Figure C.4 that for sufficiently large externalities, the collusive level may slightly increase before the sharp decrease.

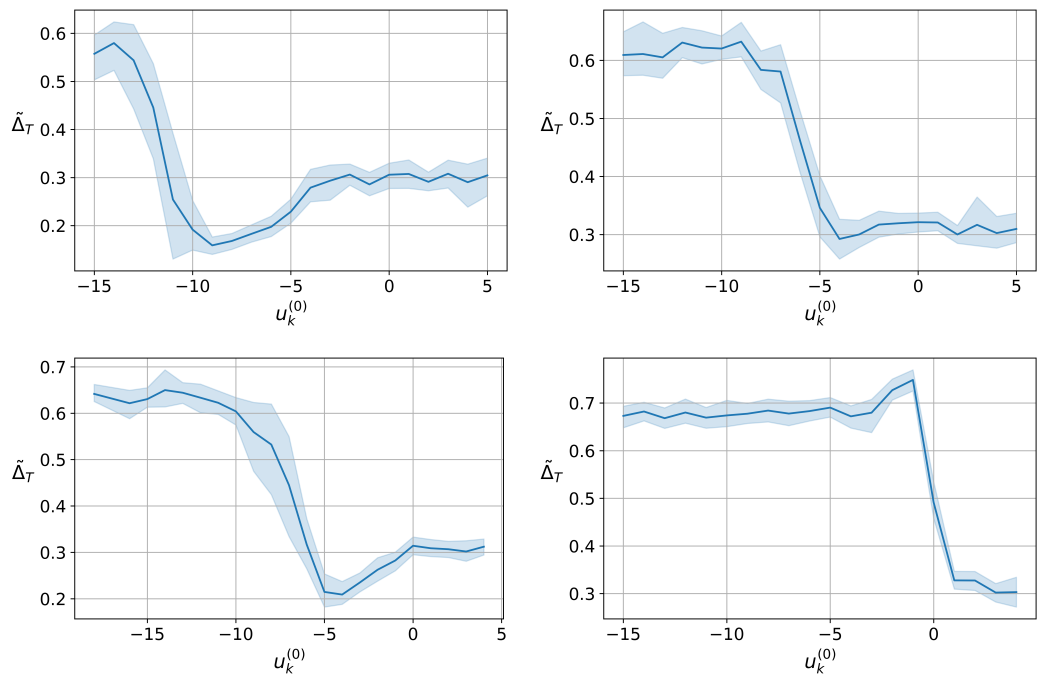

Figure 4.10: Collusive level with varying $u_k^{(0)}$, $k \in \{b, s\}$, with $\Phi = [0, 0; 0, 0]$ (top left), $\Phi = [1, 0; 0, 1]$ (top right), $\Phi = [0, 1; 1, 0]$ (bottom left) and $\Phi = [1, 1; 1, 1]$ (bottom right).

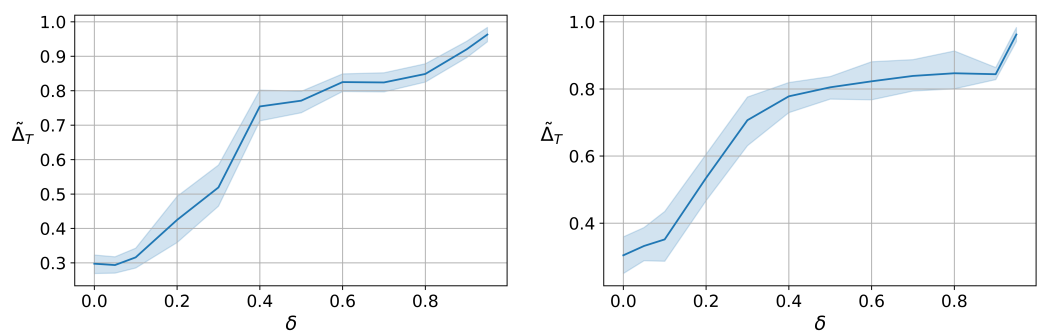

Figure 4.11: Collusive level with varying δ, where $\Phi = [0, 0; 0, 0]$ (left) and $\Phi = [1, 0; 0, 1]$ (right); in both cases $\delta \in [0.01, 0.99]$.

Figure 4.11 investigates the dependence of the collusive level on the discount rate δ, where $\delta \in [0.01, 0.99]$, with $\Phi = [0, 0; 0, 0]$ (left) and $\Phi = [1, 0; 0, 1]$ (right). In both panels, the collusive level increases with the discount rate. Thus, the more patient

platforms are about the future, the higher the collusive level becomes. We note that the collusive level in the right panel is approximately a horizontal shift of that in the left panel. Thus, with a non-zero externality matrix the collusive level increases earlier than with the zero externality matrix. The positive relationship between the collusive level and the discount rate was first observed by Calvano et al. (2020b) for Bertrand model and our experiments verify the same relationship for a multi-sided market.

4.4.4 Discussion of Exceptional Cases

We numerically demonstrate two exceptional and uncommon scenarios: asymmetric optimal prices and competition prices larger than collusion prices.

Asymmetric collusion. Our metric for the collusive level compares the platform's rewards at time t with the symmetric equilibrium quantities π^* and π^C, following previous simulations of collusion (see, e.g., Calvano et al. (2020b) and Klein (2021)). However, our specific model may give rise to asymmetric equilibria and we thus study their possibility more carefully.

To allow asymmetric equilibria, we modify the definitions of the maximum values of the total profits and the collusive level as follows. The total profit, Π_a, is

$$\Pi_a(p_b^{(1)}, p_s^{(1)}, p_b^{(2)}, p_s^{(2)}) := \sum_{i=1}^{2}(x_b^{(i)}p_b^{(i)} + x_s^{(i)}p_s^{(i)}) \equiv \sum_{i=1}^{2}\pi_t^{(i)}, \tag{4.20}$$

where unlike (4.4) it does not assume symmetric prices (that is, it does not assume that $\pi_t^{(1)} = \pi_t^{(2)}$) and its subscript a indicates asymmetry. Similarly, the collusive level Δ_t is averaged among the two firms as follows

$$\Delta_t := (\Delta_t^{(1)} + \Delta_t^{(2)})/2 \equiv \frac{\Pi_a - 2\pi^*}{2(\pi^{\mathrm{C}} - \pi^*)},$$

where $\Delta_t^{(i)}$, $i \in \{1, 2\}$, was defined in (4.17).

Figure 4.12 demonstrates the maximum value that Δ_t can achieve in two controlled settings. In the first setting (left panel) $\Phi = [\phi_{bb}, \phi_{bs}; \phi_{bs}, \phi_{bb}]$ with $\phi_{bb} \in [-1, 1]$ and $\phi_{bs} \in [-4, 4]$, and in the second one (right panel) $\Phi = [\phi_{bb}, \phi_{bs}; -\phi_{bs}, \phi_{bb}]$ with $\phi_{bb} \in [-2, 2]$ and $\phi_{bs} \in [-8, 8]$. More precisely, we maximize Δ_t over $\boldsymbol{p}_t^{(1)} \in \mathcal{A}$ and $\boldsymbol{p}_t^{(2)} \in \mathcal{A}$, where $\mathcal{A}$ was defined in (4.13), and present the maximal values using a heatmap, whose values vary from purple (Δ_t greater than 1) to orange (Δ_t less than 1). In both panels, $\beta_k = 1.0$ and $u_k^{(0)} = -2.0$, $k \in \{b, s\}$.

In the left panel, the maximal Δ_t exceeds 1 when the within-side externality ϕ_{bb} is positive and close to 1 and the cross-side externality ϕ_{bs} is sufficiently negative. We note that it is much larger than 1. In the right panel, the maximal Δ_t is larger than 1 when ϕ_{bb} is sufficiently large and ϕ_{bs} is close to zero. We note though that it never exceeds the value of 1.0175. In both panels, for all other corresponding values of ϕ_{bb} and ϕ_{bs}, the maximum value of Δ_t is either achieved at a symmetric vector price or at an asymmetric vector price with corresponding maximum value close to 1. In the latter case, of the corresponding maximum value close to 1, our measure of collusive level $\tilde{\Delta}$ using the symmetric assumption can still be used to quantify the level of collusion. That is, one may often follow up our analysis with the symmetric quantities, except for special cases where the estimated collusive level exceeds one, where one needs to use the asymmetric quantities.

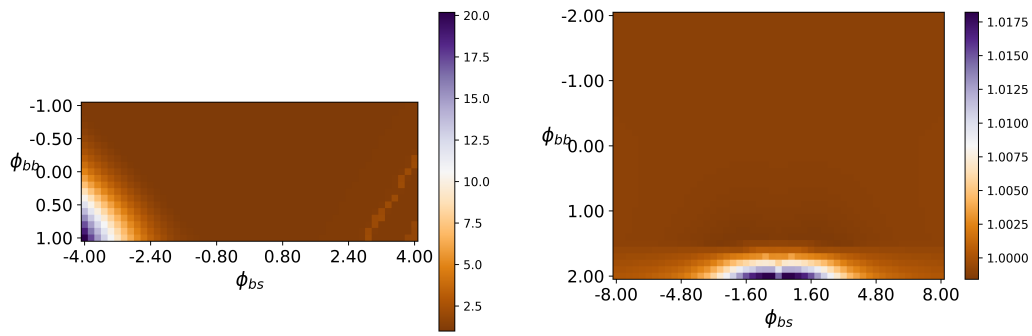

Figure 4.12: Maximal average collusive level with varying ϕ_{bb} and ϕ_{bs}, where $\Phi = [\phi_{bb}, \phi_{bs}; \phi_{bs}, \phi_{bb}]$ (left) and $\Phi = [\phi_{bb}, \phi_{bs}; -\phi_{bs}, \phi_{bb}]$ (right).

We remark that for almost all choices of β_k, $u_k^{(0)}$, δ and Φ in this paper, the asymmetric maximum value of Δ_t does not exceed the symmetric maximum value. Indeed, our simulations from Sections 4.4.1 through 4.4.3 show that $\tilde{\Delta}$ is smaller than or equal to 1. Nevertheless, when extending the right panel of Figure 4.7 to more negative values a collusive level higher than 1 is noticed, which we depict in Figure 4.13. In this figure, $\Phi = [0, \phi_{sb}; \phi_{sb}, 0]$ with $\phi_{sb} \in [-4, 2]$ and the asymmetric maximum value exceeds the symmetric maximum value when $\phi_{sb} < -2.5$. We notice that the collusive level reaches values close to 5 when the within-side externalities are zero and the cross-side externalities are equal in magnitude and sufficiently negative. Note that this scenario agrees with the one depicted in the left panel of Figure 4.12, where one can notice a similar value of 5 when $\phi_{bb} = \phi_{ss} = 0$ and $\phi_{bs} = \phi_{sb} = -4$.

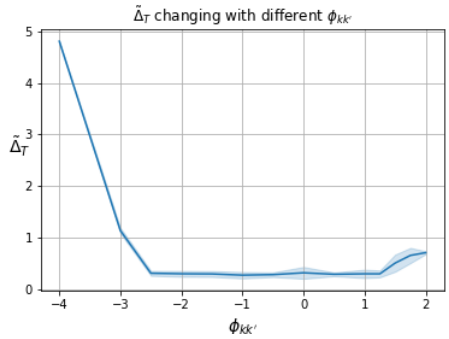

Figure 4.13: Collusive level with varying ϕ_{sb}, where $\Phi = [0, \phi_{sb}; \phi_{sb}, 0]$.

Competition prices larger than collusion prices. We recall that both the simulated action space presented in (4.12) and the formulated baseline model in Section 4.2.1 allow for the competition prices (p_k^*) to be either smaller or larger than the collusion prices (p_k^{C}). We demonstrate here an uncommon situation where the collusion price can be smaller than the competition price for one side of the market. In this example $\Phi = [1, -\phi_{sb}; \phi_{sb}, -2]$, $\beta_k = 0.5$ and $u_k^{(0)} = -1.0$, $k \in \{b, s\}$. If $\phi_{sb} \in [-5, 0.5)$, then $p_s^* < p_s^{\mathrm{C}}$. If $\phi_{sb} \in (0.5, 5]$, then $p_s^* > p_s^{\mathrm{C}}$. The left panel of Figure 4.14 demonstrates the competition and collusion prices of both sides of the market. For side b, the collusion price is always higher than the competition price. However, for side s, there are two different regimes separated by $\tilde{\phi}_{sb} \approx 0.5$. When $\phi_{sb} < \tilde{\phi}_{sb}$, the collusion price is higher than the competition price on the seller side, and when $\phi_{sb} > \tilde{\phi}_{sb}$ it is lower.

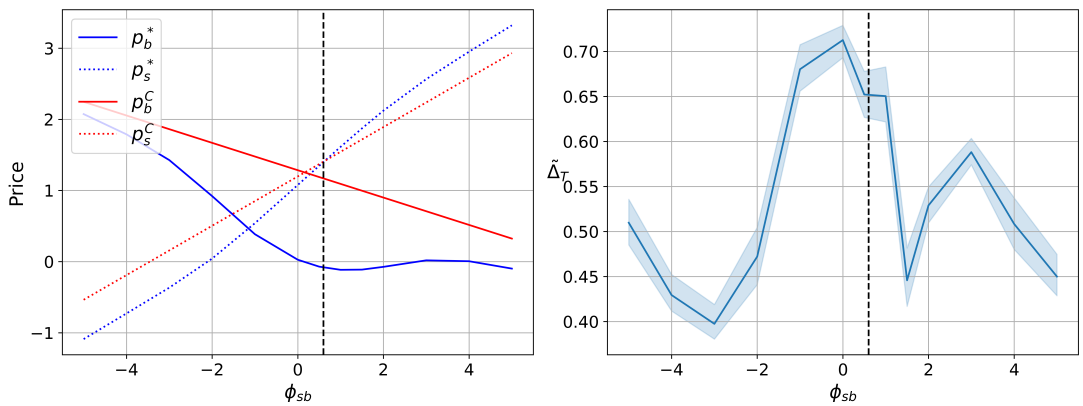

Figure 4.14: Demonstration of an uncommon scenario where the collusion prices can be lower than competition prices. Here, $\Phi = [1, -\phi_{sb}; \phi_{sb}, -2]$, where $\phi_{sb} \in [-5, 5]$. The left panel demonstrates collusion and competition prices on both sides of the market. The right panel demonstrates the collusive level. The black vertical dotted lines separate the two regimes: $p_s^* < p_s^{\mathrm{C}}$ on the left side of the black vertical dotted line and $p_s^* > p_s^{\mathrm{C}}$ on the right side of the black vertical dotted line.

The right panel of Figure 4.14 shows the collusive level for this example. The black dotted line is drawn at $\tilde{\phi}_{sb}$ to separate the two regimes. In the left regime, where $p_s^* < p_s^{\mathrm{C}}$, the collusive level decreases on $[-5, -3]$ and increases on $[-3, 0]$. This behavior is somewhat similar to the one described in the left panel of Figure 4.7. During the transition from $p_s^* < p_s^{\mathrm{C}}$ to $p_s^* > p_s^{\mathrm{C}}$, that is, in a small interval around $\tilde{\phi}_{sb}$, the collusive level decreases. Next, the collusive level increases on $[1.5, 3]$ and decreases on $[3, 5]$, which is the opposite behavior (in terms of decreasing and increasing) to the one demonstrated in the other regime.

4.5 Economic and Policy Discussion

We examine the economic implications of the numerical results presented in Section 4.4. Specifically, we analyze the impact of network externalities on the collusive level and discuss the ramifications of these findings on real-life markets. We then review

the Preventing Algorithmic Collusion Act of 2024, introduced by U.S. Senator Amy Klobuchar, and suggest an additional policy recommendation to ensure the safer use of Q-learning.

4.5.1 Economic Discussion

When $\Phi = 0$, our baseline platform competition game reduces to Bertrand competition games on each side of the market. For $\delta = 0.05$, $\beta_k = 1$ and $u_k^{(0)} = -2$, $k \in \{b, s\}$, our simulations show that Δ_0 in (4.19) is approximately 0.3. This means that when $\Phi = 0$, the profit gain relative to competition profits is about 30%. It is interesting to note that Calvano et al. (2020b) reported a different gain of approximately 20% for the same value of δ. To understand the difference, we note that in the latter paper, firms serve only one market and Bertrand competition is the baseline game. For that reason, the action space used in their simulations is smaller than the action space used in ours. The larger action space allows the platforms to algorithmically communicate more information, increasing the chance of achieving a higher collusive level.

Next, we discuss the impact of $\Phi \neq 0$ on the collusive level compared to Bertrand competition, where $\Phi = 0$. When Φ has only one non-zero entry, the dependence of the collusive level on each possible entry is depicted in Figures 4.2 and the left panel of 4.6. We note that if the nonzero externality is either positive or sufficiently negative, then the collusion is higher in platform competition than in traditional Bertrand competition. Furthermore, when Φ is a diagonal matrix, the dependence of the collusive level on these entries is depicted in Figures 4.2 and the left panel of 4.3. We note that when these entries are both positive, collusion is higher in platform competition than in single-sided Bertrand competition. These findings suggest that in markets such as online (or cloud) gaming, such as in *Xbox*, mobile and computer games, where positive within-side externalities are significant, algorithmic pricing will increase collusion levels above those observed in baseline Bertrand competition.

Finally, when Φ has only off diagonal non-zero entries, the dependence on the collusive level on these entries is depicted in Figures 4.2 and the right panel of 4.3. We note that when these entries are both positive, collusion sharply increases above the baseline Bertrand level. These findings suggest that in markets such as video streaming (e.g., *Netflix*, *Hulu*, and *Amazon*) and social media markets (e.g., *Instagram* and *TikTok*), where positive cross-side externalities are significant, high levels of collusion can be expected if platforms use algorithmic pricing.

Traditionally, platforms use positive network externalities to enhance demand and profits by subsidizing one side of the market in order to attract population on the other sides (see, e.g., Armstrong and Wright (2007), Tan and Zhou (2021), and Chica et al. (2021)). Our findings suggest that algorithmic driven platforms may also learn to use positive network externalities to significantly increase the profit.

Next we discuss the other scenario, where network externalities result in relatively small levels of collusion. First, we note that the right panels of both Figures 4.6 and 4.7 indicate examples where either the within-side externalities (Figure 4.6) or the cross-side externalities (Figure 4.7) are both negative with the same magnitude and in these cases the collusive level remains flat at the baseline competition level Δ_0. On the other hand, Figure 4.8 shows a case where the cross-side externality (ϕ_{bs}) is large and positive, and the within-side externality is sufficiently small and negative. In this scenario, the collusive level remains flat at a value slightly above Δ_0. The latter example is relevant to ride-sharing markets, where drivers compete with each other for riders, while riders benefit from faster pickup times. In this case, when using algorithmic pricing, our experiments indicate that collusive levels are close to the baseline level Δ_0.

Our findings reveal some interesting patterns in the dependence of the collusive level on three different market parameters: the degree of heterogeneity in users' tastes β_k, the constant term of the outside option utility $u_k^{(0)}$, and the discount rate δ. As

shown by Figure 4.9, the collusive level sharply decreases as the degree of heterogeneity in users' tastes increases from 0.2 to 1. Afterwards, it remains flat around values lower than the baseline collusive level Δ_0. These findings can also be observed in Appendix C.3 for multiple choices of the externality matrix Φ. We are not aware of any previous result like this. A different result states that higher degree of heterogeneity in users' tastes leads to inelastic demand and higher price, which in turn leads to higher individual profits (see, e.g., Perloff and Salop (1985) and Anderson and De Palma (1992)). However, the collusion level generally does not correlate with individual profit values.

Next, Figure 4.10 indicates a main trend of decrease of the collusive level as a function of the outside option utility. This behavior coincides with the observation of Chica et al. (2023) (see Proposition 4.6) that as the value of the outside option increases, market power held by the platforms decreases. We further noticed some very local trends, but they depend on specific choices of the externalities. This behavior has not been observed before and needs to be further explored. The main trend of the dependence of the collusive level on $u_k^{(0)}$ has relevant market implications. For instance, we note that for ride-sharing platforms using AI pricing, our analysis shows that an emphasis on increasing the outside option utility would decrease price and collusion levels. Such outside option utility can be increased by enhancing the public transportation system, and increasing mobility options such as e-bikes. Instead of investing further in such options and, in particular, in the safety of public transportation, the city of Minneapolis chose to decrease prices by passing an ordinance[8] that would force the two major ride-sharing platforms in the city, *Uber* and *Lyft*, to pay drivers the city's minimum hour wage. As a result, *Uber* and *Lyft* announced plans to leave the market. Our analysis indicates that there are other strategies to

[8]The ordinance can be found at https://lims.minneapolismn.gov/Download/FileV2/32072/Transportation-Ride-Share-Worker-Protection-Ordinance.pdf

mitigate the problem.

Finally, as shown in Figure 4.11, the collusive level increases as the discount rate increases. This result coincides with earlier results by Calvano et al. (2020b). However, the rate at which the collusive level increases w.r.t. δ seems to be larger for the case of platform competition compared to Bertrand competition. In fact, most of our experiments in section 4.4 use a value of $\delta = 0.05$, which would be considered a low discount rate. Our sensitivity analysis in Appendix indicates that this is likely tacit collusion and not just high prices. As mentioned earlier, for a more specific case, we believe that the main reason for the higher rate of increase of the collusive level in the platform competition model is due to a larger action space.

4.5.2 Policy Discussion

U.S. senator Amy Klobuchar introduced the *S.3686 - Preventing Algorithmic Collusion Act of 2024* in January 2024,[9] whose abstract is as follows:

"*A bill to prevent anticompetitive conduct through the use of pricing algorithms by prohibiting the use of pricing algorithms that can facilitate collusion through the use of nonpublic competitor data, creating an antitrust law enforcement audit tool, increasing transparency, and enforcing violations through the Sherman Act and Federal Trade Commission Act, and for other purposes.*"

This bill reflexes the increased concern by congress members and governmental institutions for the use of algorithmic price collusion. If such legislation succeeds, it will constitute a major advancement for consumer safety against the potential threats of AI. However, the above act is rather limited. Indeed, it only targets AI algorithms trained with nonpublic competitor data and characterizes them as unlawful:

[9]The act can be consulted at https://www.congress.gov/bill/118th-congress/senate-bill/3686/text,

" *SEC. 4. PREVENTING COLLUSIVE ACTIVITY IN PRICING ALGORITHMS. (a) In General.—It shall be unlawful for a person to use or distribute any pricing algorithm that uses, incorporates, or was trained with nonpublic competitor data.* "

Potentially collusive AI algorithms, such as the ones presented in this work using Q-learning, are left out of the scope of this act. We showed that these algorithms can learn to sustain high levels of collusion using only publicly available data, even in cases where agents have limited memory capacity. While this general observation has been established in previous works (Waltman and Kaymak (2008), Calvano et al. (2020b), Klein (2021) and Assad et al. (2024)), this work shows that in the presence of positive network externalities and two-sided markets, algorithmic driven platforms achieve collusive levels higher than those shown in previous works.

The question of why these algorithms can achieve high levels of collusion and whether there exists simple conditions to avoid it remains open. One way of decreasing collusion levels is obtained by increasing the value of the outside option utility, which has to be done in different ways for different markets (see e.g., the discussion in Section 4.5.1 for the ride-sharing platforms). Nevertheless, we next suggest a preliminary policy recommendation that can help avoid the risk of collusion by algorithmic driven AI agents in multiple markets. It is based on penalizing the Q-learning rewards.

Policy Recommendation (Q-learning with penalty term). We describe a very basic approach for reducing the collusive level by a potential intervention method. It can only be effective if it is enforced by regulators. The method introduces a penalty coefficient $\rho \geq 0$ that regularizes the Q-learning update formula as follows:

$$\begin{aligned} Q_{t+1}^{(i)}(\boldsymbol{s}_t^{(i)}, \boldsymbol{a}_t^{(i)}) := (1-\alpha) Q_t^{(i)}(\boldsymbol{s}_t^{(i)}, \boldsymbol{a}_t^{(i)}) \\ + \alpha \left(\pi_t^{(i)} + \delta \max_{\boldsymbol{a}} Q_t^{(i)}(\boldsymbol{s}_t^{(i)}, \boldsymbol{a}) - \rho \left((p_b^{(i)} - \bar{p}_b)_+ + (p_s^{(i)} - \bar{p}_s)_+ \right) \right), \end{aligned}$$

where $(p_k - \bar{p}_k)_+ = \max\{p_k - \bar{p}_k, 0\}$ and $\bar{p}_k = \frac{1}{N} \sum_{i=1}^{N} p_k^{(i)}$, for $k \in \{b, s\}$. Notice that

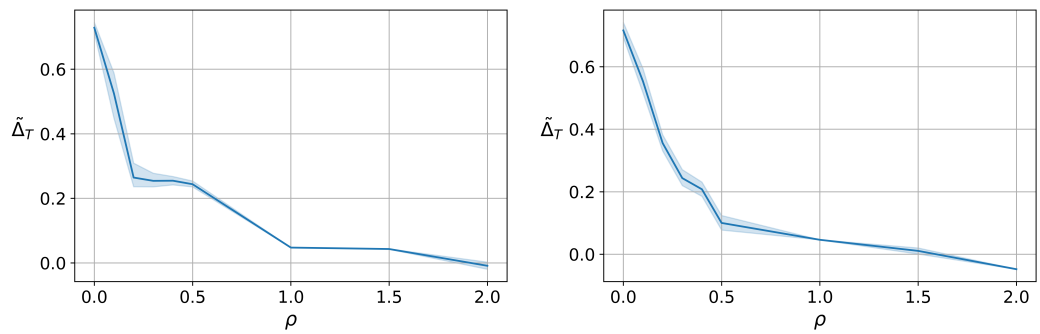

Figure 4.15: Collusive level with varying $\rho \in [0, 2]$: $\Phi = [2, 0; 0, 2]$ (left) and $\Phi = [0, 2; 2, 0]$ (right).

the penalty term, $\rho\left((p_b^{(i)} - \bar{p}_b)_+ + (p_s^{(i)} - \bar{p}_s)_+\right)$, only becomes active, if either of the current prices is larger than the average price charged by all the platforms in the market at time t.

One should fix ρ to ensure a tolerable collusive level. For example, the strongest requirement of having no collusion fixes ρ such that $\tilde{\Delta}_T = 0$. If on the other hand, one accepts the collusion level with no externalities, i.e., when $\Phi = 0$, then one may fix ρ such that $\tilde{\Delta}_T = \Delta_0$. We note that any such chosen value of ρ is a function of the parameters of the model $\{\beta_k, \delta, \Phi, u_k^{(0)}\}$. Indeed, our findings suggest that any policy recommendation aimed to reduce the risk of algorithmic price collusion needs to be dependent on market parameters.

Figure 4.15 investigates the dependence of the collusive level on the penalty coefficient ρ for a setting with $\Phi = [2, 0; 0, 2]$ (left panel) and $\Phi = [0, 2; 2, 0]$ (right panel). Without penalty, these two cases have shown significantly high collusive levels (above 0.7) as shown in the right panel of Figure 4.6 and the right panel of Figure 4.7. In both cases, the collusive level reduces sharply as ρ increases and it reaches Δ_0 when ρ is approximately 0.2 (left panel) and 0.3 (right panel). For completeness, we also show extreme cases where the collusive level can be negative, which are different from our above recommendations for choosing ρ.

4.6 Concluding Remarks

This chapter provided a framework for exploring algorithmic price collusion in a model of platform competition and numerically studied how the collusive level depends on the externality matrix.

For the case of zero network externalities, the profit gain relative to the competition profits is about 30%, higher than the value reported by Calvano et al. (2020b) for the same value of δ. We attribute this difference to the larger action space, which allows the platforms to algorithmically communicate more information, increasing the chances of achieving a high collusive level.

In common economic scenarios with positive network externalities, particularly when the positive elements are either single entries, diagonal elements, or off-diagonal elements of Φ, the collusive level in platform competition is significantly higher than in traditional Bertrand competition. This suggests that in markets such as online gaming, video streaming, and social media, AI-driven platforms may exploit positive network externalities to significantly increase profits through algorithmic pricing, indicating potential for high levels of collusion.

Our findings also reveal patterns in how the collusive level depends on user heterogeneity, discount rates, and outside option utility. Specifically, greater heterogeneity in user tastes leads to lower collusive levels. Additionally, collusion increases with higher discount rates, consistent with findings by Calvano et al. (2020b) for Bertrand competition, though the rate of increase is higher in platform competition. We provide evidence in the appendix that this collusion is tacit. Furthermore, higher outside option utility generally decreases collusion levels within a sufficiently large domain. This suggests that market regulators can reduce collusion by enhancing the value of outside options, such as improving public transportation in ride-sharing markets.

We also proposed a version of Q-learning with a penalty term to reduce the risk of

algorithmic price collusion. It can be used for policy recommendations by considering market parameters.

There are several open future research directions. First, while the platform competition model in Chica et al. (2023) allows for an unlimited number of platforms, we only conducted numerical experiments with two platforms due to the exponential growth of the state space and computational resource requirements with more platforms. Future research could explore alternative methods to estimate the Q-function to mitigate high computational demands. Second, replicating our experiments using a model that includes nonlinear network externality effects and allows for multi-homing, where users join multiple platforms, would be valuable.

References

Ahn, S., Chen, B., Wang, T., and Song, L. (2021). Spanning tree-based graph generation for molecules. In *International Conference on Learning Representations*.

Allen-Zhu, Z., Li, Y., and Liang, Y. (2019a). Learning and generalization in overparameterized neural networks, going beyond two layers. *Advances in neural information processing systems*, 32.

Allen-Zhu, Z., Li, Y., and Song, Z. (2019b). A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR.

Anderson, S. P. and De Palma, A. (1992). The logit as a model of product differentiation. *Oxford Economic Papers*, 44(1):51–67.

Armstrong, M. (2006). Competition in two-sided markets. *The RAND journal of economics*, 37(3):668–691.

Armstrong, M. and Wright, J. (2007). Two-sided markets, competitive bottlenecks and exclusive contracts. *Economic Theory*, 32:353–380.

Arora, S., Du, S., Hu, W., Li, Z., and Wang, R. (2019). Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR.

Arslantas, Y., Yuceel, E., and Sayin, M. O. (2024). Strategizing against q-learners: A control-theoretical approach. *arXiv preprint arXiv:2403.08906*.

Assad, S., Clark, R., Ershov, D., and Xu, L. (2024). Algorithmic pricing and competition: Empirical evidence from the German retail gasoline market. *Journal of Political Economy*, 132(3):000–000.

Banerjee, A., Cisneros-Velarde, P., Zhu, L., and Belkin, M. (2023). Neural tangent kernel at initialization: linear width suffices. In *Uncertainty in Artificial Intelligence*, pages 110–118. PMLR.

Barlo, M., Carmona, G., and Sabourian, H. (2009). Repeated games with one-memory. *Journal of Economic Theory*, 144(1):312–336.

Barlo, M., Carmona, G., and Sabourian, H. (2016). Bounded memory folk theorem. *Journal of economic theory*, 163:728–774.

Bertomeu, J., Evans III, J. H., Feng, M., and Tseng, A. (2021). Tacit collusion and voluntary disclosure: Theory and evidence from the US automotive industry. *Management science*, 67(3):1851–1875.

Bongini, P., Bianchini, M., and Scarselli, F. (2021). Molecular generative graph neural networks for drug discovery. *Neurocomputing*, 450:242–252.

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., and Kasneci, G. (2022). Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21.

Brero, G., Mibuari, E., Lepore, N., and Parkes, D. C. (2022). Learning to mitigate ai collusion on economic platforms. *Advances in Neural Information Processing Systems*, 35:37892–37904.

Brock, A., Donahue, J., and Simonyan, K. (2019). Large scale GAN training for high fidelity natural image synthesis. *ArXiv*, abs/1809.11096.

Brown, Z. Y. and MacKay, A. (2023). Competition in pricing algorithms. *American Economic Journal: Microeconomics*, 15(2):109–156.

Bruna, J. and Mallat, S. (2013). Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886.

Bruna, J., Zaremba, W., Szlam, A., and Lecun, Y. (2014). Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*.

Calvano, E., Calzolari, G., Denicolò, V., Harrington Jr, J. E., and Pastorello, S. (2020a). Protecting consumers from collusive prices due to ai. *Science*, 370(6520):1040–1042.

Calvano, E., Calzolari, G., Denicolo, V., and Pastorello, S. (2020b). Artificial intelligence, algorithmic pricing, and collusion. *American Economic Review*, 110(10):3267–97.

Cao, Y. and Gu, Q. (2020). Generalization error bounds of gradient descent for learning over-parameterized deep relu networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3349–3356.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 785–794.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29.

Chen, Z., Cao, Y., Zou, D., and Gu, Q. (2019). How much over-parameterization is sufficient to learn deep relu networks? *ArXiv*, abs/1911.12360.

Chica, C., Chuk, K., and Tamayo, J. A. (2021). *Exclusive Dealing and Entry by Competing Two-Sided Platforms*. Harvard Business School.

Chica, C., Guo, Y., and Lerman, G. (2023). Competition and collusion in two-sided markets with an outside option. *Available at SSRN 4641282*.

Chica, C., Guo, Y., and Lerman, G. (2024). Stochastic games, reinforcement learning and collusion with bounded memory. In preparation.

Chizat, L. and Bach, F. (2020). Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *Conference on Learning Theory*, pages 1305–1338. PMLR.

De Cao, N. and Kipf, T. (2018). MolGAN: An implicit generative model for small molecular graphs. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Du, K. and Tanriverdi, H. (2023). Does it enable collusion or competition: Examining the effects of it on service pricing in multimarket multihospital systems. *MIS Quarterly*, 47(4).

Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. (2019). Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR.

Du, Y., Wang, S., Guo, X., Cao, H., Hu, S., Jiang, J., Varala, A., Angirekula, A., and Zhao, L. (2021). Graphgt: Machine learning datasets for graph generation and transformation. In *NeurIPS 2021*.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Ertl, P. and Schuffenhauer, A. (2009). Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1(1):1–11.

Faez, F., Ommi, Y., Baghshah, M. S., and Rabiee, H. R. (2021). Deep graph generators: A survey. *IEEE Access*, 9:106675–106702.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. (2019). Graph neural networks for social recommendation. In *The World Wide Web Conference*, pages 417–426.

Gao, H. and Ji, S. (2019). Graph U-Nets. In *International Conference on Machine Learning (ICML 2019)*, pages 2083–2092. PMLR.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. (2018). Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276.

Goodfellow, I., Vinyals, O., and Saxe, A. (2015). Qualitatively characterizing neural network optimization problems. In *International Conference on Learning Representations*.

Guimaraes, G. L., Sanchez-Lengeling, B., Outeiral, C., Farias, P. L. C., and Aspuru-Guzik, A. (2017). Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models. *arXiv preprint arXiv:1705.10843*.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pages 5769—-5779.

Guo, X., Du, Y., and Zhao, L. (2021). Deep generative models for spatial networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 505–515.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Higham, C. F. and Higham, D. J. (2019). Deep learning: An introduction for applied mathematicians. *SIAM review*, 61(4):860–891.

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31.

Jin, W., Barzilay, R., and Jaakkola, T. (2018). Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*, pages 2323–2332. PMLR.

Jin, W., Yang, K., Barzilay, R., and Jaakkola, T. (2019). Learning multimodal graph-to-graph translation for molecule optimization. In *International Conference on Learning Representations*.

Jo, J., Lee, S., and Hwang, S. J. (2022). Score-based generative modeling of graphs via the system of stochastic differential equations. In *ICML*.

Johnson, J. P., Rhodes, A., and Wildenbeest, M. (2023). Platform design when sellers use pricing algorithms. *Econometrica*, 91(5):1841–1879.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR2017)*.

Klein, T. (2021). Autonomous algorithmic collusion: Q-learning under sequential pricing. *The RAND Journal of Economics*, 52(3):538–558.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. Available at `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

Kumar, R., Majmundar, K., Nagaraj, D., and Suggala, A. S. (2023). Stochastic reweighted gradient descent via distributionally robust optimization. *arXiv preprint arXiv:2306.09222*.

Kusner, M. J., Paige, B., and Hernández-Lobato, J. M. (2017). Grammar variational autoencoder. In *International Conference on Machine Learning*, pages 1945–1954. PMLR.

Lee, J., Lee, I., and Kang, J. (2019). Self-attention graph pooling. In *International Conference on Machine Learning*, pages 3734–3743. PMLR.

Li, Y. and Liang, Y. (2018). Learning overparameterized neural networks via stochastic gradient descent on structured data. *Advances in neural information processing systems*, 31.

Luo, Y., Yan, K., and Ji, S. (2021). Graphdf: A discrete flow model for molecular graph generation. In *International Conference on Machine Learning*, pages 7192–7203. PMLR.

Ma, Y., Aggarwal, C., Wang, S., and Tang, J. (2019). Graph convolutional networks with eigenpooling. In *KDD 2019 - Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 723–731.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Madhawa, K., Ishiguro, K., Nakago, K., and Abe, M. (2019). Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*.

Maere, S., Heymans, K., and Kuiper, M. (2005). Bingo: a cytoscape plugin to assess overrepresentation of gene ontology categories in biological networks. *Bioinformatics*, 21(16):3448–3449.

Mahmood, O., Mansimov, E., Bonneau, R., and Cho, K. (2021). Masked graph modeling for molecule generation. *Nature Communications*, 12(1):1–12.

Mallat, S. (2012). Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398.

Maskin, E. and Tirole, J. (1988). A theory of dynamic oligopoly, ii: Price competition, kinked demand curves, and edgeworth cycles. *Econometrica: Journal of the Econometric Society*, pages 571–599.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018). *Foundations of machine learning*. MIT press, 2nd edition.

Obara, I. and Zincenko, F. (2017). Collusion and heterogeneity of firms. *The RAND Journal of Economics*, 48(1):230–249.

OECD (2017). Algorithms and collusion: Competition policy in the digital age.

Oymak, S. and Soltanolkotabi, M. (2020). Toward moderate overparameterization: Global convergence guarantees for training shallow neural networks. *IEEE Journal on Selected Areas in Information Theory*, 1(1):84–105.

Pace, R. K. and Barry, R. (1997). Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297.

Perloff, J. M. and Salop, S. C. (1985). Equilibrium with product differentiation. *The Review of Economic Studies*, 52(1):107–120.

Polykovskiy, D., Zhebrak, A., Sanchez-Lengeling, B., Golovanov, S., Tatanov, O., Belyaev, S., Kurbanov, R., Artamonov, A., Aladinskiy, V., Veselov, M., et al. (2020). Molecular sets (MOSES): a benchmarking platform for molecular generation models. *Frontiers in pharmacology*, 11:1931.

Prykhodko, O., Johansson, S. V., Kotsias, P.-C., Arús-Pous, J., Bjerrum, E. J., Engkvist, O., and Chen, H. (2019). A de novo molecular generation method using latent vector based generative adversarial network. *Journal of Cheminformatics*, 11(1):1–13.

Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., and Carin, L. (2016). Variational autoencoder for deep learning of images, labels and captions. *Advances in neural information processing systems*, 29:2352–2360.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. (2014). Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Ridnik, T., Lawen, H., Noy, A., Ben Baruch, E., Sharir, G., and Friedman, I. (2021). Tresnet: High performance GPU-dedicated architecture. In *proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1400–1409.

Ruhmer, I. (2010). Platform collusion in two-sided markets.

Samanta, B., De, A., Jana, G., Gómez, V., Chattaraj, P. K., Ganguly, N., and Gomez-Rodriguez, M. (2020). NeVAE: A deep generative model for molecular graphs. *Journal of machine learning research. 2020 Apr; 21 (114): 1-33*.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.

Segler, M. H., Kogej, T., Tyrchan, C., and Waller, M. P. (2018). Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1):120–131.

Shamir, O. (2011). A variant of Azuma's inequality for martingales with subgaussian tails. *arXiv preprint arXiv:1110.2392*.

Shi, C., Xu, M., Zhu, Z., Zhang, W., Zhang, M., and Tang, J. (2020). Graphaf: a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*.

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

Simonovsky, M. and Komodakis, N. (2018). GraphVAE: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*, pages 412–422. Springer.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition.

Song, C., Ramezani-Kebrya, A., Pethick, T., Eftekhari, A., and Cevher, V. (2021). Subquadratic overparameterization for shallow neural networks. *Advances in Neural Information Processing Systems*, 34:11247–11259.

Sterling, T. and Irwin, J. J. (2015). Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tan, G. and Zhou, J. (2021). The effects of competition and entry in multi-sided markets. *The Review of Economic Studies*, 88(2):1002–1030.

Tirole, J. (1988). *The theory of industrial organization*. MIT press.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations (ICLR2018)*.

Waltman, L. and Kaymak, U. (2008). Q-learning agents in a cournot oligopoly model. *Journal of Economic Dynamics and Control*, 32(10):3275–3293.

Wang, X., Li, Y., Zhang, H., and Shan, Y. (2021). Towards real-world blind face restoration with generative facial prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9168–9178.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8:279–292.

Waxman, B. (1988). Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622.

Weaver, L. and Tao, N. (2001). The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, page 538–545, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

White, A. and Weyl, E. G. (2016). Insulated platform competition. *Available at SSRN 1694317*.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747.

Xu, B., Wang, N., Chen, T., and Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR2019)*.

Ying, R., You, J., Morris, C., Ren, X., Hamilton, W. L., and Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. *arXiv preprint arXiv:1806.08804*.

You, J., Liu, B., Ying, Z., Pande, V., and Leskovec, J. (2018a). Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31.

You, J., Ying, R., Ren, X., Hamilton, W., and Leskovec, J. (2018b). Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR.

Zang, C. and Wang, F. (2020). Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 617–626.

Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. (2010). Deconvolutional networks. In *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pages 2528–2535. IEEE.

Zhu, Z., Liu, F., Chrysos, G., and Cevher, V. (2022). Generalization properties of NAS under activation and skip connection search. *Advances in Neural Information Processing Systems*, 35:23551–23565.

Zou, D., Cao, Y., Zhou, D., and Gu, Q. (2020). Gradient descent optimizes over-parameterized deep relu networks. *Machine learning*, 109(3):467–492.

Zou, D. and Gu, Q. (2019). An improved analysis of training over-parameterized deep neural networks. *Advances in neural information processing systems*, 32.

Appendix A

Supplementary Material for Chapter 2

Section A.1 discusses the generalization error bound, established in Theorem 2.3, under different regimes for the number of training epochs. Section A.2 completes the proofs of the theorems stated in the main text and establishes four additional theorems: Theorem A.1, which bounds the generalization error when applying SGD; Theorem A.4, which bounds the convergence rate when using another loss function for regression; and Theorems A.2 and A.3, which bound the convergence rate and generalization error, respectively, for a special class of datasets. Section A.3 describes additional numerical experiments and the full details of implementation for both the previous and the new experiments.

A.1 Discussion of the Generalization Error Bound

In this section, we clarify the estimates for generalization error in (2.7) for different regimes of the number of training epochs, t.

We first note that the last term in (2.7) can be sufficiently small for a sufficiently large sample size n, so we may ignore it. The first bounding term in (2.7) reflects the training error and the middle two bounding terms represent the NN complexity. There is a tradeoff between the training and NN-complexity terms, as we explain below; in particular, we cannot make both of them sufficiently small. We remark that the closest bound on the generalization error for overparameterized deep NNs was

established in the context of classification using GD in Cao and Gu (2020). Their generalization bound is independent of the training epoch. Instead, their bound is of order $\Theta(\text{poly}(n) \cdot n^{-1/2})$ and is typically not small even for arbitrarily large n. For very special cases (e.g., linear separability) they reduced the term poly(n) so their overall bound is sufficiently small. In this work, we investigate the dependence of the generalization bound on t for regression without making assumptions about the data distribution. Nevertheless, one may consider similar special assumptions as in Cao and Gu (2020) and apply them to our theory in order to better control our generalization bound.

To better understand the bound in (2.7), we apply the bound on γ from Theorem 2.1 and our choice of m. We first quickly show that T is at order of $\Theta((nL)^2)$, from §2.4, we know that

$$T = \ln(\epsilon/\mathcal{L}(\boldsymbol{W}^{(0)}))/\ln\gamma \leq \Theta(\ln(\epsilon/n\sqrt{\ln m})/\ln\gamma),$$

by using (2.3) and $\eta = \Theta(d/(nmL^2))$, this upper bound is $\Theta((nL)^2)$, and when n is large, a lower bound with the same order can be achieved. We observe two different regions of $t \leq \Theta((nL)^2)$ (in §2.4, we show that $\Theta((nL)^2)$ approximates T). When $t = \Theta\left((nL)^{1-\kappa}\right)$, where $0 < \kappa < 1$, the first 3 terms of $R(\boldsymbol{W}^{(t)})$ are bounded by

$$\exp\left(-\Omega\left(\frac{(1-\alpha)^2\delta}{(1+\alpha^2)(nL)^{1+\kappa}}\right)\right)O(\ln m)+\frac{(1-\alpha)}{\sqrt{1+\alpha^2}}\,O\left(\frac{d^{1/6}\delta^{2/3}}{nL^{11/6}(nL)^{4\kappa/3}}\right)+O\left(\frac{\sqrt{d\ln m}}{(nL)^{\kappa}}\right).$$

The last two terms above are sufficiently small for sufficiently large n or L and the first training term is of the order $O(\ln m)$ and is thus the dominant one. In practice, it can be reduced through careful initialization. We note that this dominant term is minimized at $\alpha = -1$. When n and L are not sufficiently large and the second bounding term

is comparable to the first term, then the bound is minimized at a certain α between -1 and 1. If, on the other hand, $t = \Omega((nL)^{(1+\kappa)})$, where $0 < \kappa \leq 1$, then the order of the NN-complexity terms of (2.7) is $O(n^{\min\{\kappa,1/2+\tau\}}L^{\min\{\kappa,2+\tau\}})$, which becomes extremely large when n and L grow. This illustrates the overfitting phenomenon in neural network training, where the generalization error bound increases significantly as the training error approaches zero. Overall, we note that a smaller bound is obtained when $t = \Theta((nL)^{1-\kappa})$ and moreover overfitting occurs when $t = \Theta((nL)^{1+\kappa})$. These observations support the benefit of early stopping. We remark that when $t = T$, which is roughly at $\Theta((nL)^2)$, we can express the upper bound in (2.7), excluding its last term, in terms of ϵ as follows:

$$\epsilon + \min\left\{\left(\frac{(1-\alpha)^{11/3}}{(1+\alpha^2)^{11/6}}\right) O\left(\frac{d^{1/3}\delta^{4/3}}{m^{1/6}n^{10/3}L^{10/3}}\ln^{4/3}(n\sqrt{\ln m}/\epsilon)\right), O\left(\frac{d^{3/2+\tau}\delta^{\tau}n^{1/2+\tau}}{L^{1/2-\tau}\ln m}\right)\right\}$$
$$+ \min\left\{\left(\frac{(1-\alpha)^2}{1+\alpha^2}\right) O\left(\frac{d^{1/2}\delta\sqrt{\ln m}}{n^3L^3}\right)\ln(n\sqrt{\ln m}/\epsilon), O\left(\frac{n^{1/2+\tau}L^{2+\tau}d^{1/2+\tau}}{\delta^{1/2-\tau}\ln m}\right)\right\}.$$

The examination of our above theoretical results on generalization error bounds reveals two weaknesses when compared to the convergence theorems, that is, Theorems 2.1 and 2.2. Firstly, unlike the convergence rate that guarantees the training error's convergence, the generalization error bound doesn't assure a convergence to zero. Consequently, this bound may not offer a precise guideline about the optimal choice of α, especially when the number of epochs is large. Secondly, $\alpha = -1$ is the optimal choice for the generalization error bound when training terminates early and both n and L are sufficiently large. In contrast, the convergence theorem asserts that $\alpha = -1$ consistently ensures the fastest convergence. Numerical results align with these observations.

A.2 Proofs

We detail the proofs of Lemmas 2.1, 2.2 and 2.3 and the conclusion of Theorems 2.1, 2.2 and 2.3 from these lemmas. Moreover, we formulate and prove some the following additional theorems: a theorem that bounds the generalization error when using SGD, which is the analog of Theorem 2.3 for SGD instead of GD; theorems that improve our estimates for for a special class of datasets; and a theorem for the convergence theory when using a different loss function. Section A.2.1 introduces notation needed for the proof, § A.2.2 quantifies the bounds for the initial weights, § A.2.3 extends the latter bounds to weights within a small perturbation around the initialization, § A.2.4 proves the lower and upper bounds for the gradient at initial weight and within a small perturbation (Lemma 2.2), § A.2.5 shows the proof of semi-smoothness (Lemma 2.1), § A.2.6 and § A.2.7 conclude the main theorem for gradient descent and stochastic gradient descent (Theorem 2.1 and 2.2), §A.2.8 proves the upper bound of the generalization error for a class of NN functions (Lemma 2.3), §A.2.9 concludes the generalization error bound for GD (Theorem 2.3), §A.2.10 formulates and clarifies an upper bound of the generalization error for SGD, §A.2.11 introduces a special dataset and establishes theorems on the convergence rate bound and generalization error bound using this dataset, and §A.2.12 extends Theorem 2.1 and provides bounds of the convergence rate for a special loss function.

For the study of training convergence, we follow the notation and proof framework of Allen-Zhu et al. (2019b), while incorporating the improvements suggested by Zou and Gu (2019) and some additional ones. For the study of the generalization error, we follow the proof framework of Cao and Gu (2020) while extending the latter work to the task of regression. Whenever previous ideas require adaptation to Leaky ReLUs or to some of our technical contributions (summarized in §2.4.2), we prefer to repeat and even add more details so the reader can fully follow the current text and will not need

to switch between references. However, when we feel that the ideas of previous works directly extend to our setting we formulate the analogous lemmas without proving them.

A.2.1 Notation

Throughout this appendix, we denote the entries of a vector $\boldsymbol{x} \in \mathbb{R}^m$ by x_j or $(\boldsymbol{x})_j$, $j \in [m]$. We denote the entries of a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times m}$ by A_{ij} or $(\boldsymbol{A})_{ij}$, $i, j \in [m]$. For $i \in [m]$, the ith row vector of a matrix $\boldsymbol{A}$ is denoted by $\boldsymbol{A}_{i,\cdot}$ and its ith column vector is denoted by $\boldsymbol{A}_{\cdot,i}$. The default norm $\|\cdot\|$ is the ℓ_2 norm. We denote by 1_E the indicator function of the event E, which equals 1 when E occurs and 0 otherwise. We denote by $\mathcal{B}_1^m$ the unit ball in $\mathbb{R}^m$.

We use the rescaled leaky ReLU introduced in (2.2) as the activation function of the neural networks under consideration. When acting on each coordinate of a vector $\boldsymbol{x} \in \mathbb{R}^p$ we express its action using the following diagonal matrix $\boldsymbol{D}_{\boldsymbol{x}}$:

$$\tilde{\sigma}_\alpha(\boldsymbol{x}) = \boldsymbol{D}_{\boldsymbol{x}}\boldsymbol{x}, \quad \text{where } (\boldsymbol{D}_{\boldsymbol{x}})_{jj} = \frac{1_{x_j\geq 0}}{\sqrt{1+\alpha^2}} + \frac{\alpha 1_{x_j<0}}{\sqrt{1+\alpha^2}} \quad \text{and for } k \neq j \ (D_{\boldsymbol{x}})_{kj} = 0. \tag{A.1}$$

For $i \in [n]$ and a data point $\boldsymbol{x}_i \in \mathbb{R}^p$, We inductively define

$$\boldsymbol{g}_{i,l} := \boldsymbol{W}_l\boldsymbol{h}_{i,l-1}, \ \boldsymbol{h}_{i,l} := \tilde{\sigma}_\alpha(\boldsymbol{W}_l\boldsymbol{h}_{i,l-1}) \equiv \tilde{\sigma}_\alpha(\boldsymbol{g}_{i,l}), \ \boldsymbol{h}_{i,0} = \boldsymbol{A}\boldsymbol{x}_i \tag{A.2}$$

and use the notation $h_{i,l,k} := (\boldsymbol{h}_{i,l})_k$ and $g_{i,l,k} := (\boldsymbol{g}_{i,l})_k$. We denote

$$\boldsymbol{D}_{i,l} := \boldsymbol{D}_{\boldsymbol{g}_{i,l}} \ \text{ and } D_{i,l,jj} := (\boldsymbol{D}_{i,l})_{jj} \equiv \frac{1_{g_{i,l,j}\geq 0}}{\sqrt{1+\alpha^2}} + \frac{\alpha 1_{g_{i,l,j}<0}}{\sqrt{1+\alpha^2}}.$$

We further denote $\boldsymbol{D}_0 := \boldsymbol{I}$ and use the new notation to express the outputs of all

hidden layers via matrix products (where according to the notation of §2.2 $\boldsymbol{W}_0 \equiv \boldsymbol{A}$ and $\boldsymbol{W}_{L+1} \equiv \boldsymbol{B}$:

$$\begin{aligned}
\boldsymbol{g}_{i,0} &= \boldsymbol{h}_{i,0} = \boldsymbol{A}\boldsymbol{x}_i, \\
\boldsymbol{g}_{i,l} &= \boldsymbol{W}_l \boldsymbol{D}_{i,l-1} \boldsymbol{W}_{l-1} \dots \boldsymbol{W}_2 \boldsymbol{D}_{i,1} \boldsymbol{W}_1 \boldsymbol{A}\boldsymbol{x}_i, \\
\boldsymbol{h}_{i,l} &= \boldsymbol{D}_{i,l} \boldsymbol{W}_l \boldsymbol{D}_{i,l-1} \boldsymbol{W}_{l-1} \dots \boldsymbol{W}_2 \boldsymbol{D}_{i,1} \boldsymbol{W}_1 \boldsymbol{A}\boldsymbol{x}_i, \\
\boldsymbol{g}_{i,L+1} &:= \boldsymbol{B}\boldsymbol{h}_{i,L} \equiv \boldsymbol{B}\boldsymbol{D}_{i,L} \boldsymbol{W}_L \boldsymbol{D}_{i,L-1} \boldsymbol{W}_{L-1} \dots \boldsymbol{W}_2 \boldsymbol{D}_{i,1} \boldsymbol{W}_1 \boldsymbol{A}\boldsymbol{x}_i.
\end{aligned}$$

We denote the residual and its elements by

$$\boldsymbol{e}_i := \boldsymbol{g}_{i,L+1} - \boldsymbol{y}_i, \;\; e_{i,j} = (\boldsymbol{e}_i)_j$$

and the loss function by

$$\mathcal{L}(\boldsymbol{W}) := \sum_{i=1}^{n} \text{loss}(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{W}) := \sum_{i=1}^{n} \frac{1}{2} \|\boldsymbol{y}_i - \boldsymbol{g}_{i,L+1}(\boldsymbol{x}_i; \boldsymbol{W})\|^2 \equiv \frac{1}{2} \sum_{i=1}^{n} \|\boldsymbol{e}_i\|^2.$$

Section 5 in Higham and Higham (2019) presents a comprehensive derivation for the gradient of the loss function in a neural network. In our case, the activation function derivative can be written as

$$\frac{\partial h_{i,l,j}}{\partial g_{i,l,k}} = \delta_{jk} \cdot \left(\frac{1_{g_{i,l,k} \geq 0}}{\sqrt{1+\alpha^2}} + \frac{\alpha 1_{g_{i,l,k} < 0}}{\sqrt{1+\alpha^2}} \right) \equiv D_{i,l,jk}, \text{ for } l \in [L].$$

Denoting $\mathbf{Back}_{i,L+1} := \boldsymbol{B}$ and $\mathbf{Back}_{i,l} := \boldsymbol{B}\boldsymbol{D}_{i,L}\boldsymbol{W}_L \dots \boldsymbol{W}_l$ (this is the backpropagation operator) we can express the derivative of the loss with respect to the rt entry of $\boldsymbol{W}_l$, where $r, t \in [m]$, as

$$\nabla_{(\boldsymbol{W}_l)_{rt}} \text{loss}(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{W}) = (\mathbf{Back}_{i,l+1}^T \boldsymbol{e}_i)_r D_{i,l,rr} \boldsymbol{h}_{i,l-1,t}.$$

Similarly, the gradient of the loss according to the matrix $\boldsymbol{W}_l$ and according to its kth row vector, $(\boldsymbol{W}_l)_{k,\cdot}$, can be expressed as

$$\nabla_{\boldsymbol{W}_l}\text{loss}(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{W}) = \boldsymbol{D}_{i,l}\mathbf{Back}_{i,l+1}^T \boldsymbol{e}_i \boldsymbol{h}_{l-1}^T(\boldsymbol{x}_i),$$
$$\nabla_{(\boldsymbol{W}_l)_{k,\cdot}}\text{loss}(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{W}) = D_{i,l,kk}\langle(\mathbf{Back}_{i,l+1})_{\cdot,k}, \boldsymbol{e}_i\rangle \boldsymbol{h}_{l-1}(\boldsymbol{x}_i).$$

For a vector $\boldsymbol{v} \in \mathbb{R}^p$, we denote its ℓ_2 norm by $\|\boldsymbol{v}\|_2$ (where $\|\boldsymbol{v}\|_2^2 = \sum_{j\in[p]} v_j^2$), ℓ_∞ norm by $\|\boldsymbol{v}\|_\infty = \max_{j\in[p]} |v_j|$, and ℓ_0 "size" by $\|\boldsymbol{v}\|_0 = |\{j \in [p] : v_j \neq 0\}|$. For a matrix $\boldsymbol{X} \in \mathbb{R}^{m\times m}$, we denote its spectral norm by $\|\boldsymbol{X}\|_2 = \max_{j\in[m]} |\lambda_j(\boldsymbol{X})|$, Frobenius norm by $\|\boldsymbol{W}\|_F = \sqrt{\sum_{i,j\in[m]} W_{ij}^2}$, and ℓ_0 "size" by $\|\boldsymbol{D}\|_0 = |\{(i,j) \in [m]^2 : D_{ij} \neq 0\}|$. For a vector of matrices $\boldsymbol{W} = (\boldsymbol{W}_1, \ldots, \boldsymbol{W}_l)$, where $\boldsymbol{W}_1$, ..., $\boldsymbol{W}_l \in \mathbb{R}^{m\times m}$, we define its ℓ_2 norm by $\|\boldsymbol{W}\|_2 := \max_{l\in[L]} \|\boldsymbol{W}_l\|_2$ and Frobenius norm by $\|\boldsymbol{W}\|_F := \sqrt{\sum_{l=1}^L \|\boldsymbol{W}_l\|_F^2}$. For simplicity of notation we use $\|\cdot\|$ instead of $\|\cdot\|_2$ for vectors, matrices and vectors of matrices.

Throughout this appendix, we apply Algorithm 1 to initialize the weights $\boldsymbol{W}$, $\boldsymbol{A}$, $\boldsymbol{B}$ for the neural network.

We use the big O, Ω and Θ notation. That is, $f = O(N)$ or $f = \Omega(N)$ if there exists $C > 0$ and $N_0 \in \mathbb{N}$ such that $f \leq CN$ or $f \geq CN$, respectively, for all $N > N_0$. Also, $f = \Theta(N)$ if and only if $f = O(N)$ and $f = \Omega(N)$.

Throughout this appendix, we may neglect the subscript i or superscripts (t) or (0) when there is no confusion.

A.2.2 Initialization

In this section, we focus on properties of the weights initialized by Algorithm 1 without training. We thus denote $\boldsymbol{W} := \boldsymbol{W}^{(0)}$ and for any input vector $\boldsymbol{x} \in \mathbb{R}^p$ and $l \in [L]$

$$\begin{aligned}
&\boldsymbol{g}_0 = \boldsymbol{h}_0 := \boldsymbol{A}\boldsymbol{x}, \\
&\boldsymbol{g}_l := \boldsymbol{W}_l^{(0)} \boldsymbol{D}_{\boldsymbol{g}_{l-1}} \dots \boldsymbol{W}_2^{(0)} \boldsymbol{D}_{\boldsymbol{g}_1} \boldsymbol{W}_1^{(0)} \boldsymbol{A}\boldsymbol{x}, \\
&\boldsymbol{h}_l := \boldsymbol{D}_{\boldsymbol{g}_l} \boldsymbol{g}_l.
\end{aligned}$$

For simplicity, we denote $\boldsymbol{D}_l := \boldsymbol{D}_{\boldsymbol{g}_l}$.

We first establish Lemma A.1 which controls the norms of the outputs of the hidden layers with high probability. We then establish Lemma A.2 that upper bounds $\max_{i \neq j \in [n]} \langle \boldsymbol{h}_{i,l}/\|\boldsymbol{h}_{i,l}\|, \boldsymbol{h}_{j,l}/\|\boldsymbol{h}_{j,l}\| \rangle$ for all $l \in [L]$. Lastly, Lemma A.3 summarizes useful bounds of the norms of some relevant matrices.

We remark that the proof of Lemma A.1 adapts ideas of Allen-Zhu et al. (2019b) to the setting of Leaky ReLUs. The proof of Lemma A.2 follows ideas of Zou and Gu (2019), while assuming that $\delta < O(1)$ instead of $\delta < O(1/L)$ and applying minor adaptation to Leaky ReLUs. At last, Lemma A.3 directly follows the same proof argument in Allen-Zhu et al. (2019b) (while using the conclusion of Lemma A.1) and we thus omit its proof.

Lemma A.1

Assume the setup of §2.2 and the above notation. If $\boldsymbol{x} \in \mathbb{R}^p$, $\|\boldsymbol{x}\| = 1$ and ϵ is a fixed number in $(\Omega(\frac{L}{m}), 1)$, then

$$\|\boldsymbol{h}_l\| \in [1-\epsilon, 1+\epsilon] \text{ for all } \; l \in \{0\} \cup [L]$$
$$\text{with probability at least } 1 - e^{-\Omega(m\epsilon^2/L)}. \qquad \square$$

Proof: We first prove the lemma for $l = 0$. Due to the initialization of the input layer by Algorithm 1, $\boldsymbol{h}_0 = \boldsymbol{A}\boldsymbol{x} \sim N(0, \|\boldsymbol{x}\|^2/m) = N(0, 1/m)$. Therefore, $m\|\boldsymbol{h}_0\|^2 \sim \chi^2(m)$, where $\chi^2(m)$ denotes the chi-square distribution with m degrees of freedom. Using the tail bound for this sub-Gaussian distribution

$$\mathbb{P}\left(\left|\|\boldsymbol{h}_0\|^2 - 1\right| > \frac{\epsilon}{2}\right) \leq 2e^{-m\epsilon^2/32} \leq e^{-\Omega(m\epsilon^2)}. \tag{A.3}$$

We next prove the lemma for $l \geq 1$. For each layer l, we analyze the distribution of each entry of $\boldsymbol{h}_l$, and denote by $h_{l,j} := (\boldsymbol{h}_l)_j$, $j \in [m]$, conditioned on the output from the former layer $\boldsymbol{h}_{l-1}$. We note that the randomness of $\boldsymbol{h}_l$ comes from $\boldsymbol{W}_l$ given the fixed $\boldsymbol{h}_{l-1}$.

We note the following expression for $h_{l,j}$, which follows from (A.1) and (A.2):

$$h_{l,j} = \tilde{\sigma}_\alpha(g_{l,j}) = 1_{g_{l,j}>0}\frac{g_{l,j}}{\sqrt{1+\alpha^2}} + 1_{g_{l,j}\leq 0}\frac{\alpha g_{l,j}}{\sqrt{1+\alpha^2}}. \tag{A.4}$$

We remark that unlike previous analyses (Allen-Zhu et al., 2019b; Zou and Gu, 2019), we need to deal with two different terms in the sum in order to address Leaky ReLU and note just ReLU. We observe that due to the initialization of $\boldsymbol{W}_l$ and (A.2), $g_{l,j} \sim N(0, 2\sum h_{l-1,k}^2/m) = N(0, 2\|\boldsymbol{h}_{l-1}\|^2/m)$. By the symmetry of the normal distribution, $g_{l,j}$ is positive with probability 0.5. Thefore, the random variable

$$B_j := 1_{g_{l,j}>0}$$

is Bernoulli with probability 0.5, that is, $B_j \sim \mathrm{B}(0.5)$. We further note that $B_j g_{l,j} = B_j g_{l,j} | g_{l,j} > 0$. We thus rewrite (A.4) as

$$h_{l,j} = \frac{1}{\sqrt{1+\alpha^2}} B_j g_{l,j} | \{g_{l,j} > 0\} - \frac{\alpha}{\sqrt{1+\alpha^2}}(1 - B_j)(-g_{l,j}) | \{g_{l,j} \leq 0\}. \tag{A.5}$$

Conditioning on the event $g_{l,j} > 0$, $g_{l,j} \overset{d}{=} |X|$, where $X \sim N(0, 2\|\boldsymbol{h}_{l-1}\|^2/m)$. Therefore,

$$g_{l,j}\Big|\,(g_{l,j} > 0) \;\sim |N(0, 2\|\boldsymbol{h}_{l-1}\|^2/m)|.$$

Similarly,

$$-g_{l,j}\Big|\,(g_{l,j} \leq 0) \;\sim |N(0, 2\|\boldsymbol{h}_{l-1}\|^2/m)|.$$

Therefore, (A.5) and the above two equations imply the following distribution law for h_{ij}:

$$h_{l,j} \overset{d}{=} \frac{1}{\sqrt{1+\alpha^2}} B_j V_{j,1} - \frac{\alpha}{\sqrt{1+\alpha^2}}(1 - B_j) V_{j,2},$$

where $V_{j,1}$, $V_{j,2} \sim |N(0, 2\|\boldsymbol{h}_{l-1}\|^2/m)|$, $B_j \sim B(0, \frac{1}{2})$ and $V_{j,1}$, $V_{j,2}$ and B_j are independent. We further claim that if the former layer $\boldsymbol{h}_{l-1}$ is given, then $V_{j,1}$ and $V_{j,2}$ are independent for $j \in [m]$. Indeed, We first observe that conditioned on $\boldsymbol{h}_{l-1}$ the entries $h_{l,j}$, $j \in [m]$, are independent. Indeed, they depend on different rows in $\boldsymbol{W}_l$ and due to Algorithm 1 for the initialization of the lth layer these rows are independent. We also note that $V_{j,1}$ and $V_{j,2}$ only rely on $h_{l,j}$, and thus conditioned on $\boldsymbol{h}_{l-1}$ they are independent for $j \in [m]$.

We next derive an expression that clarifies the distribution of $\|\boldsymbol{h}_l\|^2$ conditioned on $\boldsymbol{h}_{l-1}$. We denote

$$P_l := \{j \in [m] : \; g_{l,j} > 0\}, \qquad K_l := |P_l|,$$
$$H_{l,1} := \frac{m}{2\|\boldsymbol{h}_{l-1}\|^2} \sum_{j \in P_l} V_{j,1}^2 | \boldsymbol{h}_{l-1}, \qquad H_{l,2} := \frac{m}{2\|\boldsymbol{h}_{l-1}\|^2} \sum_{j \in [m], j \notin P_j} V_{j,2}^2 | \boldsymbol{h}_{l-1}.$$

We note that K_l is Bernoulli with m trials and probability 0.5, i.e.,

$$K_l \sim B(m, 0.5).$$

The above observations imply that conditioning on $\boldsymbol{h}_{l-1}$ and P_l, $H_{l,1} \sim \chi^2(K_l)$ and

$H_{l,2} \sim \chi^2(m - K_l)$. Therefore, $\|\boldsymbol{h}_l\|^2$ conditioned on $\boldsymbol{h}_{l-1}$ is given by

$$\|\boldsymbol{h}_l\|^2|\boldsymbol{h}_{l-1} \overset{d}{=} \frac{2\|\boldsymbol{h}_{l-1}\|^2}{(1+\alpha^2)m} H_{l,1} + \frac{2\alpha^2\|\boldsymbol{h}_{l-1}\|^2}{(1+\alpha^2)m} H_{l,2}. \tag{A.6}$$

Note that the indices used by $H_{l,1}$ and indices used by $H_{l,2}$ do not overlap and thus form a partition of $[m]$. This partition is determined by P_l and $H_{l,1}$ and $H_{l,2}$ are conditionally independent given P_l.

We denote $\Delta_l := \frac{\|\boldsymbol{h}_l\|^2}{\|\boldsymbol{h}_{l-1}\|^2}$ and rewrite $\|\boldsymbol{h}_b\|^2$ (fixing $l = b$) as follows

$$\ln \|\boldsymbol{h}_b\|^2 = \ln \|\boldsymbol{h}_0\|^2 + \sum_{l=1}^{b} \ln \Delta_l. \tag{A.7}$$

Using the distribution of $\|\boldsymbol{h}_l\|^2$ conditioning on $\boldsymbol{h}_{l-1}$, where $1 \leq l \leq b$, we first derive upper and lower bounds of the expectation $\mathbb{E}(\ln \Delta_l|\boldsymbol{h}_{l-1})$. We then show that given $\boldsymbol{h}_{l-1}$ and other information, $\ln \Delta_l$ is an $O(m^{-1})$ sub-Gaussian random variable. With these two properties we conclude the lemma by applying a variant of Azuma's inequality for sub-Gaussian random variables on $\sum_{l=1}^{b} \ln \Delta_l$.

Bounds on the expectation of $\ln\boldsymbol{\Delta}_l|\boldsymbol{h}_{l-1}$: We note that $\mathbb{E}(H_{l,1}|P_l) = K_l$, $\mathbb{E}(H_{l,1}|P_l) = m - K_l$ and thus $\mathbb{E}(H_{l,1}) = \mathbb{E}(\mathbb{E}(H_{l,1}|P_l)) = \mathbb{E}(K_l) = 0.5m$. Similarly, $\mathbb{E}(H_{l,2}) = 0.5m$ and therefore $\mathbb{E}(H_{l,1}) = \mathbb{E}(H_{l,2})$. Using the latter observation and (A.6) we obtain

$$\mathbb{E}(\Delta_l|\boldsymbol{h}_{l-1}) = \frac{2}{m(1+\alpha^2)} \left(\mathbb{E}(H_{l,1}) + \alpha^2\mathbb{E}(H_{l,1})\right) = 1. \tag{A.8}$$

Applying the concavity of the log function, Jensen's inequality and then (A.6) and (A.8) yields

$$\mathbb{E}\left(\frac{1}{1+\alpha^2} \ln \frac{2}{m} H_{l,1} + \frac{\alpha^2}{1+\alpha^2} \ln \frac{2}{m} H_{l,2}\right) \leq \mathbb{E} \ln(\Delta_l|\boldsymbol{h}_{l-1}) \leq \ln \mathbb{E}(\Delta_l|\boldsymbol{h}_{l-1}) = 0. \tag{A.9}$$

Using the Chernoff bound for the binomial distribution, we note that

$$K_l \in [0.4m, 0.6m], \text{ or equivalently } m - K_l \in [0.4m, 0.6m], \text{ with probability } 1 - e^{-\Omega(m)}. \tag{A.10}$$

We next use the property that if $H \sim \chi^2(K)$ and $K \in [0.4m, 0.6m]$, then $\mathbb{E} \ln \frac{2}{m} H \geq -\frac{4}{m}$ (see page 13 in the proof of Lemma 7.1 in Allen-Zhu et al. (2019b)). This property and (A.9) imply

$$\mathbb{E}\left(\ln(\Delta_l) | \boldsymbol{h}_{l-1}\right) \in \left[-\frac{4}{m}, 0\right]. \tag{A.11}$$

Conditional sub-Gaussianity of $\ln \boldsymbol{\Delta}_l$: We derive a tail bound for $\ln \Delta_l | \boldsymbol{h}_{l-1}$ and consequently conclude its sub-Gaussianity. We denote

$$E_l := \{|P_l| \in [0.4m, 0.6m]\}.$$

The combination of (A.6), basic probabilistic manipulations and the conditional

independence of $H_{l,1}$ and $H_{l,2}$ yields

$$\begin{aligned}
&\mathbb{P}\left(\left|\frac{m}{2}\Delta_l - \frac{m}{2}\right| < t \middle| \boldsymbol{h}_{l-1}, E_l, P_l\right) \\
&= \mathbb{P}\left(\left|\frac{1}{1+\alpha^2}H_{l,1} + \frac{\alpha^2}{1+\alpha^2}H_{l,2} - \frac{m}{2}\right| < t \middle| E_l, P_l\right) \\
&\geq \mathbb{P}\left(\left|\frac{1}{1+\alpha^2}H_{l,1} - \frac{1}{1+\alpha^2}\frac{m}{2}\right| < t/2 \right. \\
&\qquad \text{and} \quad \left.\left|\frac{\alpha^2}{1+\alpha^2}H_{l,2} - \frac{\alpha^2}{1+\alpha^2}\frac{m}{2}\right| < t/2 \middle| E_l, P_l\right) \\
&\geq \mathbb{P}\left(\left|\frac{1}{1+\alpha^2}H_{l,1} - \frac{1}{1+\alpha^2}\frac{m}{2}\right| < t/2 \middle| E_l, P_l\right) \\
&\quad \cdot \mathbb{P}\left(\left|\frac{\alpha^2}{1+\alpha^2}H_{l,2} - \frac{\alpha^2}{1+\alpha^2}\frac{m}{2}\right| < t/2 \middle| E_l, P_l\right) \\
&\geq \mathbb{P}\left(\left|H_{l,1} - \frac{m}{2}\right| < t/2 \middle| E_l, P_l\right) \mathbb{P}\left(\left|H_{l,2} - \frac{m}{2}\right| < t/2 \middle| E_l, P_l\right).
\end{aligned}$$

Recall that given P_l, $H_{l,1}$ and $H_{l,2}$ are $\chi^2(|P_l|)$ and $\chi^2(m - |P_l|)$, respectively. We thus apply the corresponding tail bounds of $H_{l,1}$ and $H_{l,2}$ and (A.10) to the bound above and obtain that

$$\mathbb{P}\left(\left|\frac{m}{2}\Delta_l - \frac{m}{2}\right| < t \middle| \boldsymbol{h}_{l-1}\right) \geq \left(1 - e^{-\Omega(t^2/m)}\right)^2 \geq 1 - \Omega\left(e^{-\Omega(t^2/m)}\right).$$

Consequently,

$$\mathbb{P}\left(\ln|\Delta_l| < \frac{t}{m} \middle| \boldsymbol{h}_{l-1}\right) \geq 1 - e^{-\Omega((\frac{t}{m})^2 m)} \quad \text{for} \quad t \in (0, m/4].$$

Therefore, $\ln \Delta_l$ conditioned on $\boldsymbol{h}_{l-1}$ and $K_l \in [0.4m, 0.6m]$ is $O(m^{-1})$-sub-Gaussian.

Conclusion of the proof of the lemma: We define a new variable $\tilde{\Delta}_l$, where $\tilde{\Delta}_l = \Delta_l$ if $K_l \in [0.4m, 0.6m]$ and $\tilde{\Delta}_l = 1$, otherwise. From the tail probability of $\ln \Delta_l$ and the definition, it is clear that $\ln \tilde{\Delta}_l | \boldsymbol{h}_{l-1}$ is $O(m^{-1})$-sub-Gaussian. It follows from (A.10) that with overwhelming probability $\Delta = \tilde{\Delta}$. We consider the sequence of

the following random variables $\{(\ln \tilde{\Delta}_l - \mathbb{E} \ln \tilde{\Delta}_l)|\boldsymbol{h}_{l-1}\}_{l=1}^b$. By Azuma's inequality for sub-Gaussian variables (see Theorem 2 with $c = m$ in Shamir (2011))

$$\mathbb{P}\left(\left|\sum_{l=1}^{b} \ln \Delta_l - \mathbb{E}(\ln \Delta_l|\boldsymbol{h}_{l-1})\right| > b\epsilon\right) < e^{-\Omega(b\epsilon^2 m)}.$$

Applying (A.11) to the above inequality yields

$$\mathbb{P}\left(\left|\sum_{l=1}^{b} \ln \Delta_l\right| > \epsilon + O\left(\frac{b}{m}\right)\right) < e^{-\Omega(\epsilon^2 m/b)}.$$

We can choose $\epsilon > \Omega(\frac{L}{m})$ such that

$$\mathbb{P}\left(\left|\sum_{l=1}^{b} \ln \Delta_l\right| > \epsilon/2\right) < e^{-\Omega(\epsilon^2 m/b)}$$

Combining (A.3), (A.7) and the above equation we obtain that

$$\mathbb{P}\left(\left|\|\boldsymbol{h}_b\|^2 - 1\right| > \epsilon_0\right) < e^{-\Omega(m\epsilon^2/L)}, \text{ for } \ b \in [L].$$

□

Lemma A.2

Assume the setup of §2.2 and the notation introduced in this section. If $\delta < O(1)$ and $m > \Omega(\ln nL^4)$, then

$$\max_{i \neq j \in [n]} \left\langle \frac{\boldsymbol{h}_{i,l}}{\|\boldsymbol{h}_{i,l}\|}, \frac{\boldsymbol{h}_{j,l}}{\|\boldsymbol{h}_{j,l}\|} \right\rangle^2 \leq 1 - \Omega\left(\frac{\delta^2}{L^2}\right) \quad \text{with probability at least } 1 - e^{-\Omega(\delta^4 m/L^4)}. \tag{A.12}$$

□

Proof: We separate the proof of this lemma into three parts. The first one establishes a useful upper bound of the expectation of the multiplication of two leaky ReLUs of certain inner products (see (A.13) below). Given this upper bound, the second part shows that with high probability,

$$\min_{i\neq j\in[n]} \|\boldsymbol{h}_{i,l} - \boldsymbol{h}_{j,l}\| \geq \Omega(\delta/L), \quad \text{for any } l \in [L].$$

The third part uses the result to conclude this Lemma.

Part 1. We verify the following probabilistic estimate:

$$\mathbb{E}\tilde{\sigma}_\alpha(\boldsymbol{u}^T\boldsymbol{h}_i)\tilde{\sigma}_\alpha(\boldsymbol{u}^T\boldsymbol{h}_j) \leq \frac{1}{2}\left(1 - \frac{1}{2}\theta^2\right) + \frac{(1-\alpha)^2}{(1+\alpha^2)}O(\theta^3), \tag{A.13}$$

$$\text{where } \boldsymbol{h}_i, \boldsymbol{h}_j \in \mathbb{R}^p, \quad \text{for } \theta > 0, \langle \boldsymbol{h}_i, \boldsymbol{h}_j\rangle \leq 1 - \frac{1}{2}\theta^2, \quad \text{and } \boldsymbol{u} \sim N(0, \boldsymbol{I}) \in \mathbb{R}^p.$$

Since $\boldsymbol{u} \sim N(0, \boldsymbol{I})$, $\mathbb{E}u_k u_{k'} = 0$ whenever $k \neq k'$. We denote $\boldsymbol{u} := (u_1, u_2 \cdots u_p)^T$, $\boldsymbol{h}_i := (h_{i,1}, h_{i,2}, \cdots h_{i,p})^T$ and $\boldsymbol{h}_j := (h_{j,1}, h_{j,2}, \cdots h_{j,p})^T$. We first note that

$$\begin{aligned}\mathbb{E}\left(\boldsymbol{u}^T\boldsymbol{h}_i\right)\left(\boldsymbol{u}^T\boldsymbol{h}_j\right) &= \mathbb{E}\left(\sum_{k=1}^p u_k h_{i,k}\right)\left(\sum_{k'=1}^p u_{k'} h_{j,k'}\right) \\ &= \mathbb{E}\sum_{k=1}^p u_k^2 h_{i,k} h_{j,k} = \boldsymbol{h}_i^T\boldsymbol{h}_j \mathbb{E}\boldsymbol{u}^T\boldsymbol{u} \leq 1 - \frac{1}{2}\theta^2.\end{aligned}$$

For simplicity, we denote $Z_i := \boldsymbol{u}^T\boldsymbol{h}_i$ and $Z_j := \boldsymbol{u}^T\boldsymbol{h}_j$ and thus express the above equation as

$$\mathbb{E}(Z_i Z_j) \leq 1 - \frac{1}{2}\theta^2. \tag{A.14}$$

Using the symmetry of normal distribution, we obtain that

$$\mathbb{E}(Z_i Z_j | Z_i, Z_j \geq 0) = \mathbb{E}(Z_i Z_j | Z_i, Z_j < 0)$$

and

$$\mathbb{E}(Z_iZ_j|Z_i<0,Z_j\geq 0)=\mathbb{E}(Z_iZ_j|Z_i\geq 0,Z_j<0).$$

Consequently, the expectation of $\tilde{\sigma}_\alpha(Z_i)\tilde{\sigma}_\alpha(Z_j)$ can be rewritten as

$$\begin{aligned}\mathbb{E}\tilde{\sigma}_\alpha(Z_i)\tilde{\sigma}_\alpha(Z_j)&=\frac{1}{1+\alpha^2}\Big(\mathbb{E}(Z_iZ_j|Z_i,Z_j\geq 0)+\alpha\mathbb{E}(Z_iZ_j|Z_i\geq 0,Z_j<0)\\&\quad+\alpha\mathbb{E}(Z_iZ_j|Z_i<0,Z_j\geq 0)+\alpha^2\mathbb{E}(Z_iZ_j|Z_i,Z_j<0)\Big)\\&=\mathbb{E}(Z_iZ_j|Z_i,Z_j\geq 0)+\frac{2\alpha}{1+\alpha^2}\mathbb{E}(Z_iZ_j|Z_i\geq 0,Z_j<0).\end{aligned}\tag{A.15}$$

Similarly, we express $\mathbb{E}Z_iZ_j$ as follows: (A.15)

$$\begin{aligned}\mathbb{E}Z_iZ_j&=2\mathbb{E}(Z_iZ_j|Z_i,Z_j\geq 0)+2\mathbb{E}(Z_iZ_j|Z_i\geq 0,Z_j\leq 0)\\&=2\mathbb{E}\tilde{\sigma}_\alpha(Z_1)\tilde{\sigma}_\alpha(Z_2)+\left(2-\frac{4\alpha}{1+\alpha^2}\right)\mathbb{E}(Z_1Z_2|Z_1\geq 0,Z_2\leq 0).\end{aligned}$$

Rearranging the above equation yields

$$\mathbb{E}\tilde{\sigma}_\alpha(Z_1)\tilde{\sigma}_\alpha(Z_2)=\frac{1}{2}\mathbb{E}Z_1Z_2-\frac{(1-\alpha)^2}{1+\alpha^2}\mathbb{E}(Z_1Z_2|Z_1\geq 0,Z_2<0).\tag{A.16}$$

Noting that $\mathbb{E}Z_1Z_2\leq 1-\frac{1}{2}\theta^2$ and using the proof of Lemma A.3 of Zou et al. (2020) result in

$$|\mathbb{E}(Z_1Z_2|Z_1\geq 0,Z_2<0)|\leq O(\theta^3).$$

The application of both (A.14) and the above estimate to (A.16) results in(A.13) and thus concludes this part.

Part 2. For $l=0,\ldots,L$ and $\delta_l:=\frac{\delta}{2(l+1)}$ we prove by induction:

$$\min_{i\neq j\in[n]}\|\boldsymbol{h}_{i,l}-\boldsymbol{h}_{j,l}\|\geq\delta_l \text{ with probability at least } 1-e^{-\Omega(\delta^4m/L^4)}.\tag{A.17}$$

We first prove (A.17) when $l = 0$. Recall that $\boldsymbol{h}_{i,0} = \boldsymbol{A}\boldsymbol{x}_i$ and note that for any $i, j \in [n]$,

$$\begin{aligned}
\mathbb{E}\left(\|\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_j\|^2\right) &= \mathbb{E}\left\langle \boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_j, \boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_j \right\rangle \qquad \text{(A.18)}\\
&= \mathbb{E}\|\boldsymbol{A}\boldsymbol{x}_i\|^2 + \mathbb{E}\|\boldsymbol{A}\boldsymbol{x}_j\|^2 - 2\mathbb{E}\langle \boldsymbol{A}\boldsymbol{x}_i, \boldsymbol{A}\boldsymbol{x}_j\rangle \\
&= 2 - 2\mathbb{E}\sum_{k=1}^{m}\sum_{s,t} A_{ks}x_{i,s}A_{kt}x_{j,t} = 2 - 2\mathbb{E}\sum_{k,s=1}^{m} x_{i,s}x_{j,s}A_{ks}^2 \\
&= 2 - 2\sum_{s,k} x_{i,s}x_{j,s}\mathbb{E}A_{ks}^2 = 2 - 2\sum_{s,k} x_{i,s}x_{j,s}\frac{1}{m} \\
&= 2 - 2\boldsymbol{x}_i^T\boldsymbol{x}_j. \qquad \text{(A.19)}
\end{aligned}$$

Recall that Assumption 2.1 implies that $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| \geq \delta$ and thus clearly

$$\boldsymbol{x}_i^T\boldsymbol{x}_j \leq 1 - \delta^2/2.$$

Applying this estimate in (A.19) yields the that

$$\mathbb{E}(\|\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_j\|^2) \geq \delta^2. \qquad \text{(A.20)}$$

Due to the random initialization, $m\|\boldsymbol{A}\boldsymbol{x}\|^2 \sim \chi^2(m)$ and therefore $\|\boldsymbol{A}\boldsymbol{x}\|^2$ is $(O(1/m, 4)$ sub-exponential. Since

$$\mathbb{P}(\|\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_j\|^2 > s) \leq \mathbb{P}(\|\boldsymbol{A}\boldsymbol{x}_i\|^2 > s/2) + \mathbb{P}(\|\boldsymbol{A}\boldsymbol{x}_j\|^2 > s/2),$$

the tail probability of $\|\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_j\|^2$ is of the same order as the tail probabilities of $\|\boldsymbol{A}\boldsymbol{x}_i\|^2$ and $\|\boldsymbol{A}\boldsymbol{x}_j\|^2$. Therefore, we conclude that $\|\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_j\|^2$ is also $(O(1/m), 4)$ sub-exponential. Using the assumption $\delta < c_0$, where c_0 can be appropriately chosen (here we assume that $c_0\delta < 3/4$), (A.20) and the fact that $\|\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_j\|^2$ is $(O(1/m), 4)$

sub-exponential) we conclude that

$$\begin{aligned}\mathbb{P}\left(\|\boldsymbol{A}\boldsymbol{x}_i-\boldsymbol{A}\boldsymbol{x}_j\|^2<\frac{\delta^2}{4}\right)&\le\mathbb{P}\left(\|\boldsymbol{A}\boldsymbol{x}_i-\boldsymbol{A}\boldsymbol{x}_j\|^2<\delta^2(1-\delta)\right)\\&\le\mathbb{P}\left(\|\boldsymbol{A}\boldsymbol{x}_i-\boldsymbol{A}\boldsymbol{x}_j\|^2<(1-\delta)\mathbb{E}\|\boldsymbol{A}\boldsymbol{x}_i-\boldsymbol{A}\boldsymbol{x}_j\|^2\right)\\&\le O(e^{-\delta^4 m}).\end{aligned}$$

Applying a union bound over all distinct i, $j\in[n]$, we conclude that with probability at least $1-n^2e^{-\Omega(\delta^4 m)}$,

$$\min_{i\neq j\in[n]}\|\boldsymbol{h}_{i,0}-\boldsymbol{h}_{j,0}\|\ge\frac{\delta}{2}\equiv\delta_0.$$

Next, we fix $l\in[L]$, assume that (A.17) holds for all $k\in[0,l-1]$ and verify (A.17) for l. Using the fact that $\mathbb{E}(\|\boldsymbol{h}_{i,l}\|^2|\boldsymbol{h}_{i,l-1})=\|\boldsymbol{h}_{i,l-1}\|^2$ and the definition of $\boldsymbol{h}_{i,l}$ we obtain

$$\begin{aligned}&\mathbb{E}(\|\boldsymbol{h}_{i,l}-\boldsymbol{h}_{j,l}\|^2|\boldsymbol{h}_{l-1})\\&=\mathbb{E}(\|\boldsymbol{h}_{i,l}\|^2|\boldsymbol{h}_{i,l-1})+\mathbb{E}(\|\boldsymbol{h}_{j,l}\|^2|\boldsymbol{h}_{j,l-1})-2\mathbb{E}(\langle\boldsymbol{h}_{i,l},\boldsymbol{h}_{j,l}\rangle|\boldsymbol{h}_{l-1})\\&=\|\boldsymbol{h}_{i,l-1}\|^2+\|\boldsymbol{h}_{j,l-1}\|^2-2\mathbb{E}(\langle\tilde{\sigma}_\alpha(\boldsymbol{W}_l\boldsymbol{h}_{i,l-1}),\tilde{\sigma}_\alpha(\boldsymbol{W}_l\boldsymbol{h}_{j,l-1})\rangle|\boldsymbol{h}_{l-1}).\end{aligned}\tag{A.21}$$

Applying the induction assumption (i.e., (A.13) with $\theta=\delta_{l-1}$) and the fact that $(\boldsymbol{W}_l)_{k,\cdot}\sim N\left(0,\frac{2}{m}\boldsymbol{I}\right)$ and denoting by $\boldsymbol{u}$ a random variable such that $\boldsymbol{u}\sim N(0,\boldsymbol{I})$ so $(\boldsymbol{W}_l)_{k,\cdot}^T\overset{d}{=}2\boldsymbol{u}/m$ result in

$$\begin{aligned}&\mathbb{E}(\langle\tilde{\sigma}_\alpha(\boldsymbol{W}_l\boldsymbol{h}_{i,l-1}),\tilde{\sigma}_\alpha(\boldsymbol{W}_l\boldsymbol{h}_{j,l-1})\rangle|\boldsymbol{h}_{l-1})\\&=\sum_k\mathbb{E}(\tilde{\sigma}_\alpha((\boldsymbol{W}_l)_{k,\cdot}^T\boldsymbol{h}_{i,l-1})\tilde{\sigma}_\alpha((\boldsymbol{W}_l)_{k,\cdot}^T\boldsymbol{h}_{j,l-1})|\boldsymbol{h}_{l-1})\\&=\frac{2m}{m}\mathbb{E}(\tilde{\sigma}_\alpha(\boldsymbol{u}^T\boldsymbol{h}_{i,l-1})\tilde{\sigma}_\alpha(\boldsymbol{u}^T\boldsymbol{h}_{j,l-1})|\boldsymbol{h}_{l-1})\\&\le1-\frac{1}{2}\delta_{l-1}^2+\frac{(1-\alpha)^2}{1+\alpha^2}O(\delta_{l-1}^3).\end{aligned}$$

Using Lemma A.1, we note for any $i \in [n]$, $\|\boldsymbol{h}_{i,l}\|^2 \in (1 - O(\delta_{l-1}^3), 1 + O(\delta_{l-1}^3))$ with probability at least $1 - ne^{-\Omega(\delta_{l-1}^3 m)}$. Combining this observation with (A.21) yields for a constant $C > 0$

$$\mathbb{E}(\|\boldsymbol{h}_{i,l} - \boldsymbol{h}_{j,l}\|^2 | \boldsymbol{h}_{l-1}) \geq \delta_{l-1}^2 \left(1 - C\frac{(1-\alpha)^2}{1+\alpha^2}\delta_{l-1}\right) + O(\delta_{l-1}^3).$$

It follows from (A.6) and the fact that $H_{l,1} \sim \chi^2(K_l)$ and $H_{l,2} \sim \chi^2(m - K_l)$ that $\|\boldsymbol{h}_{i,l}\|^2 | \boldsymbol{h}_{l-1}$ is $(O(1/m), 4)$ sub-exponential and thus $\|\boldsymbol{h}_{i,l} - \boldsymbol{h}_{j,l}\|^2 | \boldsymbol{h}_{l-1}$ is also $(O(1/m), 4)$ sub-exponential. Thus for $i \neq j \in [n]$,

$$\mathbb{P}\left(\|\boldsymbol{h}_{i,l} - \boldsymbol{h}_{j,l}\|^2 \leq \delta_{l-1}^2 \left(1 - \left(C\frac{(1-\alpha)^2}{1+\alpha^2}\right)\delta_{l-1}\right)(1-\delta_{l-1}) \middle| \boldsymbol{h}_{l-1}\right) \leq O(\exp(-\delta_{l-1}^4 m)).$$

Applying a union bound for all $n(n-1)/2$ pairs yields

$$\begin{aligned}\mathbb{P}\left(\min_{i \neq j \in [n]} \|\boldsymbol{h}_{i,l} - \boldsymbol{h}_{j,l}\|^2 \leq \delta_{l-1}^2 \left(1 - \left(C\frac{(1-\alpha)^2}{1+\alpha^2}\right)\delta_{l-1}\right)(1-\delta_{l-1}) \middle| \boldsymbol{h}_{l-1}\right) \\ \leq n(n-1)/2 O(\exp(-\delta_{l-1}^4 m)). \quad \text{(A.22)}\end{aligned}$$

Consequently,

$$\begin{aligned}
&1 - n^2\Omega(\exp(-\Omega(\delta_{l-1}^4 m))) \\
&\leq \mathbb{P}\left(\min_{i \neq j \in [n]} \|\boldsymbol{h}_{i,l} - \boldsymbol{h}_{j,l}\|^2 \geq \delta_{l-1}^2 \left(1 - \left(C\frac{(1-\alpha)^2}{1+\alpha^2}\right)\delta_{l-1}\right)(1-\delta_{l-1}) \middle| \boldsymbol{h}_{l-1}\right) \\
&= \mathbb{P}\left(\min_{i \neq j \in [n]} \|\boldsymbol{h}_{i,l} - \boldsymbol{h}_{j,l}\|^2 \geq \delta_{l-1}^2 \left(1 - \left(C\frac{(1-\alpha)^2}{1+\alpha^2} + 1\right)\delta_{l-1}\right) \right. \quad \text{(A.23)} \\
&\qquad \left. + C\frac{(1-\alpha)^2}{1+\alpha^2}\delta_{l-1}^4 \middle| \boldsymbol{h}_{l-1}\right) \\
&\leq \mathbb{P}\left(\min_{i \neq j \in [n]} \|\boldsymbol{h}_{i,l} - \boldsymbol{h}_{j,l}\|^2 \geq \delta_{l-1}^2 \left(1 - \left(C\frac{(1-\alpha)^2}{1+\alpha^2} + 1\right)\delta_{l-1}\right) \middle| \boldsymbol{h}_{l-1}\right). \quad \text{(A.24)}
\end{aligned}$$

Next, we verify that $1 - (C(1-\alpha)^2/(1+\alpha^2) + 1)\delta_{l-1} \geq l^2/(l+1)^2$ for a sufficiently

small c_0 (recall that $\delta < c_0$). We first note that for $l \geq 1$,

$$\frac{l^2}{(l+1)^2} = 1 - \frac{2l+1}{(l+1)^2} \leq 1 - \frac{l+1}{(l+1)^2} = 1 - \frac{1}{l+1} \leq 1 - \frac{1}{2l}.$$

Therefore, if $\delta < 1/(C(1-\alpha)^2/(1+\alpha^2)+1) < c_0$, then for any $l \in [L]$

$$1 - \left(C\frac{(1-\alpha)^2}{1+\alpha^2} + 1\right)\delta_{l-1} = 1 - \left(C\frac{(1-\alpha)^2}{1+\alpha^2} + 1\right)\frac{\delta}{2l} \geq 1 - \frac{1}{2l} \geq \frac{l^2}{(l+1)^2}.$$

Thus (A.24) implies $\min_{i \neq j \in [n]} \|\boldsymbol{h}_{i,l} - \boldsymbol{h}_{j,l}\|^2 \geq \delta_{l-1}^2 \frac{l^2}{(l+1)^2} \equiv \delta_l^2$ with probability $1 - n^2 e^{-\Omega(\delta^4 m/L^4)}$. When $m > \Omega(\ln nL^4)$, the latter probability can be written as $1 - e^{-\Omega(\delta^4 m/L^4)}$, which concludes (A.17).

Part 3. We conclude the lemma as follows. We recall that Lemma A.1 implies that with probability at least $1 - e^{-\Omega(m\delta_l^3/L)}$: $\|\boldsymbol{h}_{i,l}\|^2 \in [1 - O(\delta_l^3), 1 + O(\delta_l^3)]$. Applying this conclusion and (A.17) we conclude that for any $i \neq j \in [n]$

$$\begin{aligned}
\left\|\frac{1}{\|\boldsymbol{h}_{j,l}\|}\boldsymbol{h}_{j,l} - \frac{1}{\|\boldsymbol{h}_{i,l}\|}\boldsymbol{h}_{i,l}\right\| &= \left\|\frac{1}{\|\boldsymbol{h}_{j,l}\|}\boldsymbol{h}_{j,l} - \frac{1}{\|\boldsymbol{h}_{j,l}\|}\boldsymbol{h}_{i,l} + \frac{1}{\|\boldsymbol{h}_{j,l}\|}\boldsymbol{h}_{i,l} - \frac{1}{\|\boldsymbol{h}_{i,l}\|}\boldsymbol{h}_{i,l}\right\| \\
&\geq \frac{1}{\|\boldsymbol{h}_{j,l}\|}\|\boldsymbol{h}_{j,l} - \boldsymbol{h}_{i,l}\| - \left|\frac{1}{\|\boldsymbol{h}_{j,l}\|} - \frac{1}{\|\boldsymbol{h}_{i,l}\|}\right| \|\boldsymbol{h}_{i,l}\| \\
&\geq \delta_l(1 - \delta_l^{1/2}) \\
&\qquad \text{with probability at least } 1 - 2e^{-\Omega(m\delta^4/L^4)}.
\end{aligned}$$

We note that for $\delta < c_0 < 1/2$, $\delta_l < \frac{1}{4}$ and thus $\delta_l(1-\delta_l^{1/2}) \geq \frac{1}{2}\delta_l$. Consequently,

$$\begin{aligned}
&\left\langle \frac{1}{\|\boldsymbol{h}_{j,l}\|}\boldsymbol{h}_{j,l}, \frac{1}{\|\boldsymbol{h}_{i,l}\|}\boldsymbol{h}_{i,l} \right\rangle \\
&= \frac{1}{2}\left(\frac{\|\boldsymbol{h}_{j,l}\|^2}{\|\boldsymbol{h}_{j,l}\|^2} + \frac{\|\boldsymbol{h}_{i,l}\|^2}{\|\boldsymbol{h}_{i,l}\|^2} - \left\| \frac{1}{\|\boldsymbol{h}_{j,l}\|}\boldsymbol{h}_{j,l} - \frac{1}{\|\boldsymbol{h}_{i,l}\|}\boldsymbol{h}_{i,l} \right\|^2 \right) \\
&= 1 - \frac{1}{2}\left\| \frac{1}{\|\boldsymbol{h}_{j,l}\|}\boldsymbol{h}_{j,l} - \frac{1}{\|\boldsymbol{h}_{i,l}\|}\boldsymbol{h}_{i,l} \right\|^2 \\
&\leq 1 - \frac{1}{8}\delta_l^2 \quad \text{with probability at least } 1 - 2e^{-\Omega(m\delta^4/L^4)}.
\end{aligned}$$

Therefore, if $\delta_l^2 < 8$, then

$$\begin{aligned}
&\left\langle \frac{1}{\|\boldsymbol{h}_{j,l}\|}\boldsymbol{h}_{j,l}, \frac{1}{\|\boldsymbol{h}_{i,l}\|}\boldsymbol{h}_{i,l} \right\rangle^2 \leq \left(1 - \frac{1}{8}\delta_l^2\right)^2 \leq 1 - \frac{1}{8}\delta_l^2 \\
&\quad \text{with probability at least } 1 - 2e^{-\Omega(m\delta^4/L^4)}.
\end{aligned}$$

Finally, we apply a union bound on all the distinct i, j pairs to obtain

$$\max_{i,j\in[n]} \left\langle \frac{1}{\|\boldsymbol{h}_{j,l}\|}\boldsymbol{h}_{j,l}, \frac{1}{\|\boldsymbol{h}_{i,l}\|}\boldsymbol{h}_{i,l} \right\rangle^2 \leq 1 - \frac{1}{8}\delta_l^2 \quad \text{with probability at least } 1 - n^2 e^{-\Omega(m\delta^4/L^4)}.$$

The proof of the lemma is concluded by the above bound and the following two immediate observations: $\delta_l \equiv \delta/2(l+1) \geq \Omega(\delta/L)$ and when $m > \Omega(\ln n)L^4$ the above probability can be expressed as $1 - e^{-\Omega(\delta^4 m/L^4)}$. □

Lemma A.3

Assume the setup of §2.2 and the notation introduced in this section. If $0 \leq a < b \leq L$, then with probability at least $1 - e^{-\Omega(m/L)}$ the following statements hold:

1. $\|\boldsymbol{W}_{b+1}\boldsymbol{D}_b\boldsymbol{W}_b \ldots \boldsymbol{D}_a\| \leq O(\sqrt{L})$.

2. If $d < O(\frac{m}{L\ln m})$, then $\|\mathbf{Back}_a\| \equiv \|\boldsymbol{B}\boldsymbol{D}_L\boldsymbol{W}_L \ldots \boldsymbol{D}_a\boldsymbol{W}_a\| \leq O(\sqrt{\frac{m}{d}})$.

3. If $\boldsymbol{v} \in \mathbb{R}^m$ and $\|\boldsymbol{v}\|_0 \leq O\left(\frac{m}{L \ln m}\right)$, then $\|\boldsymbol{W}_b \boldsymbol{D}_{b-1} \ldots \boldsymbol{D}_a \boldsymbol{W}_a \boldsymbol{v}\| \leq 2\|\boldsymbol{v}\|$.

For $s < O(m/L \ln m)$ and $d < O(\frac{m}{L \ln m})$, with probability at least $1 - \exp(-\Omega(s \log m))$, the following statement holds:

4. For any vector $\boldsymbol{u} \in \boldsymbol{R}^d,\ \boldsymbol{v} \in \boldsymbol{R}^m$ such that $\|\boldsymbol{v}\|_0 \leq s$, then $|\boldsymbol{u}^T \boldsymbol{B} \boldsymbol{D}_L \boldsymbol{W}_L \cdots \boldsymbol{D}_a \boldsymbol{W}_a \boldsymbol{v}| \leq O(\sqrt{s \ln m/d} \|\boldsymbol{v}\| \|\boldsymbol{u}\|)$. □

The proof of the lemma follows the same argument of the proof of Lemma 7.3 (a), (b) and Lemma 7.4 (a), (b) in Allen-Zhu et al. (2019b) and is not directly affected by our use of Leaky ReLU. We remark though that it requires applying Lemma A.1, which was formulated for any Leaky ReLU function instead of Lemma 7.1 of Allen-Zhu et al. (2019b).

A.2.3 Perturbation

We establish Lemma A.4 which quantifies the effect of a small perturbation of the randomly initialized parameters $\boldsymbol{W}^{(0)}$ on the output of the hidden layers. Lemma A.5 uses the former lemma to bound the norms of the perturbed matrices and the perturbations themselves. The proof of Lemma A.4 directly follows ideas of Lemma 8.2 of Allen-Zhu et al. (2019b), but adapts them to the setting of Leaky ReLUs. The final conclusion of this lemma is independent of α since the leading terms turn out to be independent of α. For completeness, we find it useful to include all these details. Lemma A.5 directly follows arguments of Allen-Zhu et al. (2019b) and we thus omit its proof.

We denote the perturbation matrix by $\boldsymbol{W'}$ and the perturbed matrix of parameters by $\boldsymbol{W} := \boldsymbol{W}^{(0)} + \boldsymbol{W'}$. Given an input vector $\boldsymbol{x}$ such that $\|\boldsymbol{x}\| = 1$, we denote as follows the variables at the initialization (in first column), the variables after perturbation (in

middle column) and the perturbation themselves (in last column):

$$\begin{aligned}
&\boldsymbol{h}_0^{(0)} = \boldsymbol{A}\boldsymbol{x} && \boldsymbol{h}_0 = \boldsymbol{A}\boldsymbol{x} && \boldsymbol{h}_0' = \boldsymbol{0} \\
&\boldsymbol{g}_l^{(0)} = \boldsymbol{W}_l^{(0)}\boldsymbol{h}_{l-1}^{(0)}, && \boldsymbol{g}_l = \boldsymbol{W}_l\boldsymbol{h}_{l-1} && \boldsymbol{g}_l' = \boldsymbol{g}_l - \boldsymbol{g}_l^{(0)} \\
&(\boldsymbol{D}_l)_{jj}^{(0)} = \frac{1_{(\boldsymbol{g}_l^{(0)})_j \geq 0} + \alpha 1_{(\boldsymbol{g}_l^{(0)})_j < 0}}{\sqrt{1+\alpha^2}}, && (\boldsymbol{D}_l)_{jj} = \frac{1_{(\boldsymbol{g}_l)_j \geq 0} + \alpha 1_{(\boldsymbol{g}_l)_j < 0}}{\sqrt{1+\alpha^2}}, && \boldsymbol{D}_l' = \boldsymbol{D}_l - \boldsymbol{D}_l^{(0)} \\
&\boldsymbol{h}_l^{(0)} = \tilde{\sigma}_\alpha(\boldsymbol{W}_l^{(0)}\boldsymbol{h}_{l-1}^{(0)}) \equiv \tilde{\sigma}_\alpha(\boldsymbol{g}_l^{(0)}), && \boldsymbol{h}_l = \tilde{\sigma}_\alpha(\boldsymbol{W}_l\boldsymbol{h}_{l-1}) \equiv \tilde{\sigma}_\alpha(\boldsymbol{g}_l), && \boldsymbol{h}_l' = \boldsymbol{h}_l - \boldsymbol{h}_l^{(0)}.
\end{aligned}$$

Since we fix $\boldsymbol{A}$ and $\boldsymbol{W}_{L+1} \equiv \boldsymbol{B}$ in the training, $\boldsymbol{B}^{(0)} := \boldsymbol{B}$ and $\boldsymbol{A}^{(0)} := \boldsymbol{A}$.

Lemma A.4

If $\|\boldsymbol{W}'\|_2 = \omega < O(\frac{1}{L^{9/2}\ln^{3/2} m})$ and $m \geq \Omega(L^2)$, then the following events hold with probability at least $1 - e^{-\Omega\left(\frac{m^{1/2}}{\ln m}\right)}$

1. $\|\boldsymbol{D}_l'\|_0 < O(m\omega^{2/3}L)$ and $\|\boldsymbol{D}_l'\boldsymbol{g}_l\| < \frac{1-\alpha}{\sqrt{1+\alpha^2}}O(\omega L^{3/2})$
2. there exist vectors $\boldsymbol{g}_{l,1}'$ and $\boldsymbol{g}_{l,2}'$ such that $\boldsymbol{g}_l' = \boldsymbol{g}_{l,1}' + \boldsymbol{g}_{l,2}'$, and $\|\boldsymbol{g}_{l,1}'\| = O(\omega L^{3/2})$ and $\|\boldsymbol{g}_{l,2}'\|_\infty = O\left(\frac{\omega L^{5/2}\sqrt{\ln m}}{\sqrt{m}}\right)$,
3. $\|\boldsymbol{g}_l'\|, \|\boldsymbol{h}_l'\| < O\left(\omega L^{5/2}\sqrt{\ln m}\right)$. □

Proof: We divide the proof into two steps. First, we show that statements 2 and 3 of the lemma imply statement 1 . We then prove statements 2 and 3 of the lemma using an induction argument for $l \in \{0, 1, \ldots L\}$.

Statements 2 and 3 imply statement 1. We fix $l \in \{0, 1, \ldots L\}$. In view of Lemma A.1 and the focus on the lth layer, we assume that $\boldsymbol{h}_{l-1}^{(0)}$ is a fixed vector such that $\|\boldsymbol{h}_{l-1}^{(0)}\| \in [0.5, 1.5]$. More precisely, we can condition on $\boldsymbol{h}_{l-1}^{(0)}$ and we know that with overwhelming probability $\|\boldsymbol{h}_{l-1}^{(0)}\| \in [0.5, 1.5]$. We denote $g_{l,j}^{(0)} := (\boldsymbol{g}_l^{(0)})_j$ (note the

difference between the vector notation $\boldsymbol{g}_{i,l}$ and the scalar notation $g_{l,j}^{(0)}$). We recall that

$$\boldsymbol{g}_l^{(0)} = \boldsymbol{W}_l^{(0)}\boldsymbol{h}_{l-1}^{(0)} \sim N\left(0, \frac{2\|\boldsymbol{h}_{l-1}^{(0)}\|^2}{m}\boldsymbol{I}\right)$$
$$\text{and thus } \; g_{l,j}^{(0)} \sim N\left(0, \frac{2\|\boldsymbol{h}_{l-1}^{(0)}\|^2}{m}\right) \text{ for } j \in [m].$$

We define the following vector $\boldsymbol{d}$ and express it using the decomposition $\boldsymbol{g}_l' = \boldsymbol{g}_{l,1}' + \boldsymbol{g}_{l,2}'$ in statement 2 of this lemma:

$$\boldsymbol{d} := \boldsymbol{D}_l'(\boldsymbol{W}_l^{(0)}\boldsymbol{h}_l^{(0)} + \boldsymbol{g}_l') = \boldsymbol{D}_l'(\boldsymbol{W}_l^{(0)}\boldsymbol{h}_l^{(0)} + \boldsymbol{g}_{l,1}' + \boldsymbol{g}_{l,2}').$$

We denote $D_{l,jj}' := (\boldsymbol{D}_l')_{jj}$, $g_{l,1,j}' := (\boldsymbol{g}_{l,1}')_j$ and $g_{l,2,j}' := (\boldsymbol{g}_{l,2}')_j$.

To estimate $\|\boldsymbol{d}\|$ and $\|\boldsymbol{d}\|_0$ we define the following auxiliary sets that partition $\{j \in [m] : d_j \neq 0\}$, S_1 and S_2. To do this we arbitrarily choose a positive number $\xi > 2\|\boldsymbol{g}_{l,2}'\|_\infty$ and define

$$S_1 := \{j \in [m] : |g_{l,j}^{(0)}| < \xi, \; d_j \neq 0\}$$

and

$$S_2 := \{j : j \in [m]/S_1, d_j \neq 0\}.$$

In the rest of the proof we bound $|S_1|$, $\sum_{j\in S_1} d_j^2$, $|S_2|$ and $\sum_{j\in S_2} d_j^2$. We then use these estimates to bound $\|\boldsymbol{d}\|$ and $\|\boldsymbol{d}\|_0$.

In order to bound $|S_1|$, we first note that

$$\mathbb{P}(|g_{l,j}^{(0)}| < \xi, d_j \neq 0) \leq \mathbb{P}(|g_{l,j}^{(0)}| < \xi) \leq \Theta\left(\xi\sqrt{\frac{m}{\|\boldsymbol{h}_{l-1}^{(0)}\|^2}}\right) = \Theta(\xi\sqrt{m}).$$

Combining a Chernoff bound for the binomial distribution with the above estimate

yields

$$|S_1| < O(\xi m^{3/2}) \text{ with probability at least } 1 - e^{-\Omega(m^{3/2}\xi)}. \tag{A.25}$$

For $j \in S_1$, we upper bound the coordinate d_j of $\boldsymbol{d}$:

$$|d_j| \leq \left|\frac{1-\alpha}{\sqrt{1+\alpha^2}}\right| |g_{l,j}^{(0)} + g'_{l,1,j} + g'_{l,2,j}| \leq \left|\frac{1-\alpha}{\sqrt{1+\alpha^2}}\right| (\xi + \|\boldsymbol{g}'_2\|_\infty + |g'_{l,1,j}|).$$

For each index $j \in [m]$ such as $D'_{l,jj} \neq 0$ we note from the definition of $\boldsymbol{D}'$ that $|D'_{l,jj}| = (1-\alpha)/\sqrt{1+\alpha^2}$. By squaring both sides of the above inequality, summing over the indices in S_1 and applying (A.25), we conclude that with probability at least $1 - e^{-\Omega(m^{3/2}\xi)}$

$$\begin{aligned}
\sum_{j\in S_1} |d_j|^2 &\leq 3 \sum_{j\in S_1} \frac{(1-\alpha)^2}{1+\alpha^2} (\xi^2 + \|\boldsymbol{g}'_{l,2}\|_\infty^2 + |g'_{l,1,j}|^2) \\
&\leq \frac{3(1-\alpha)^2}{1+\alpha^2} |S_1| (\xi^2 + \|\boldsymbol{g}'_{l,2}\|_\infty^2) + \frac{3(1-\alpha)^2}{1+\alpha^2} \|\boldsymbol{g}'_{l,1}\|^2 \\
&\leq \frac{3(1-\alpha)^2}{1+\alpha^2} O\left(\xi m^{3/2}\right) (\xi^2 + \|\boldsymbol{g}'_{l,2}\|_\infty^2) + \frac{3(1-\alpha)^2}{1+\alpha^2} \|\boldsymbol{g}'_{l,1}\|^2.
\end{aligned} \tag{A.26}$$

We next estimate $|S_2|$. The definitions of the diagonal matrices $\boldsymbol{D}_l$, $\boldsymbol{D}_l^{(0)}$ and $\boldsymbol{D}'_l$ imply that if $D'_{jj} \neq 0$, then $g_{l,j}^{(0)}$ and $g_{l,j}$ have opposite signs, or equivalently, $g_{l,j}^{(0)} + g'_{l,j}$ and $g_{i,l}^{(0)}$ have opposite signs, which further implies that $|g'_{l,j}| \geq |g_{l,j}^{(0)}|$. We further note that by the triangle inequality $|g'_{l,j}| \leq |g'_{l,1,j}| + |g'_{l,2,j}|$. Combining these two observation and then applying additional basic estimates, we obtain

$$|g'_{l,1,j}| \geq |g_{l,j}^{(0)}| - |g'_{l,2,j}| \geq \xi - \|\boldsymbol{g}'_{l,2}\|_\infty \quad \text{for} \quad j \in S_2.$$

This bound clearly implies

$$\|\boldsymbol{g}'_{l,1}\|^2 \geq \sum_{j\in S_2} |g'_{l,1,j}|^2 \geq |S_2|(\xi - \|\boldsymbol{g}'_{l,2}\|_\infty)^2$$

and consequently

$$|S_2| \leq \frac{\|\boldsymbol{g}'_{l,1}\|^2}{(\xi - \|\boldsymbol{g}'_{l,2}\|_\infty)^2}. \tag{A.27}$$

For $j \in S_2$, we note as above that $g^{(0)}_{l,j}$ and $g'_{l,j}$ have opposite signs and $|g'_{l,j}| > |g^{(0)}_{l,j}|$. The combination of both of these observations imply $|g^{(0)}_{l,j} + g'_{l,j}| \leq |g'_{l,j}|$. The later observation and the partition of $\boldsymbol{g}_l$ according to the second statement of the lemma yield the following bound for $j \in S_2$:

$$|d_j| = \frac{|1-\alpha|}{\sqrt{1+\alpha^2}}|g^{(0)}_{l,j} + g'_{l,j}| \leq \frac{|1-\alpha|}{\sqrt{1+\alpha^2}}|g'_{l,j}| \tag{A.28}$$

$$\leq \frac{|1-\alpha|}{\sqrt{1+\alpha^2}}(|g'_{l,1,j}| + \|\boldsymbol{g}'_{l,2}\|_\infty). \tag{A.29}$$

Squaring both sides of (A.29), summing over $j \in S_2$ and applying (A.27) yield

$$\begin{aligned}\sum_{j\in S_2} |d_j|^2 &\leq 2\frac{(1-\alpha)^2}{1+\alpha^2}\sum_{j\in S_2}(|g'_{l,1,j}|^2 + \|\boldsymbol{g}'_{l,2}\|_\infty^2) \\ &\leq 2\frac{(1-\alpha)^2}{1+\alpha^2}(\|\boldsymbol{g}'_{l,1}\|^2 + |S_2|\|\boldsymbol{g}'_{l,2}\|_\infty^2) \\ &\leq 2\frac{(1-\alpha)^2}{1+\alpha^2}\left(\|\boldsymbol{g}'_{l,1}\|^2 + \frac{\|\boldsymbol{g}'_{l,2}\|_\infty^2\|\boldsymbol{g}'_{l,1}\|^2}{(\xi - \|\boldsymbol{g}'_{l,2}\|_\infty)^2}\right).\end{aligned} \tag{A.30}$$

Obtaining these four different estimates we conclude with bounds on $\|\boldsymbol{d}\|_0$ and

$\|\boldsymbol{d}\|$. We first note that (A.25) and (A.27) yield

$$\|\boldsymbol{d}\|_0 \le |S_1| + |S_2| \le \Theta(\xi m^{3/2}) + \frac{\|\boldsymbol{g}'_{l,1}\|^2}{(\xi - \|\boldsymbol{g}'_{l,2}\|_\infty)^2}$$
$$\text{with probability at least } \ 1 - e^{-\Omega(m^{3/2}\xi)}.$$

Since $\xi > 2\|\boldsymbol{g}'_{l,2}\|_\infty$, we can obtain the following bound:

$$\|\boldsymbol{d}\|_0 \le \Theta(\xi m^{3/2}) + \frac{4\|\boldsymbol{g}'_{l,1}\|^2}{\xi^2}.$$

In order to tighten the above bound, we minimize the right hand side term with respect to ξ and note that its minimal value is $m\|\boldsymbol{g}'_{l,1}\|^{2/3}$ and is obtained at $\xi_{\min} = \Theta\left(\|\boldsymbol{g}'_{l,1}\|^{2/3}/m^{1/2}\right)$. We note that the assumed conditions: $\omega < O\left(L^{-9/2}(\ln m)^{-3/2}\right)$, $\|\boldsymbol{g}'_{l,1}\| = O(\omega L^{3/2})$ and $\|\boldsymbol{g}'_{l,2}\|_\infty < O(\omega L^{5/2}\sqrt{\ln m}/\sqrt{m})$ imply that $\xi_{\min} > 2\|\boldsymbol{g}'_{l,2}\|_\infty$ so that the minimum is achieved. Thus, an upper bound of $\|\boldsymbol{d}\|_0$ is obtained as

$$\|\boldsymbol{d}\|_0 \le O(m\|\boldsymbol{g}'_{l,1}\|^{2/3}) \le O(m\omega^{2/3}L).$$

Combining (A.26) and (A.30) yields

$$\begin{aligned}
\|\boldsymbol{d}\|^2 &= \sum_{j=1}^{m} d_j^2 = \sum_{j\in S_1} d_j^2 + \sum_{j\in S_2} d_j^2 \\
&\le \frac{3(1-\alpha)^2}{1+\alpha^2} O\left(\xi m^{3/2}\right)(\xi^2 + \|\boldsymbol{g}_{l,2}\|_\infty^2) + \\
&\quad \frac{5(1-\alpha)^2}{1+\alpha^2}\|\boldsymbol{g}'_{l,1}\|^2 + 2\frac{(1-\alpha)^2}{1+\alpha^2}\frac{\|\boldsymbol{g}'_{l,1}\|^2\|\boldsymbol{g}'_{l,2}\|_\infty^2}{(\xi - \|\boldsymbol{g}'_{l,2}\|_\infty)^2} \\
&\le C\frac{(1-\alpha)^2}{1+\alpha^2}(\xi^3 m^{3/2} + \|\boldsymbol{g}'_{l,1}\|_2^2).
\end{aligned}$$

Plugging in $\xi = \xi_{\min}$ to the above equation and applying the second statement of this

lemma result in

$$\|\boldsymbol{d}\|^2 \leq O\left(\frac{(1-\alpha)^2}{1+\alpha^2}\|\boldsymbol{g}'_{l,1}\|^2\right) \leq \frac{1-\alpha}{\sqrt{1+\alpha^2}}O(\omega^2 L^3). \tag{A.31}$$

Consequently, our bounds for $\|\boldsymbol{D}_l\|_0$ and $\|\boldsymbol{d}\| = \|\boldsymbol{D}'_l\boldsymbol{g}_l\|$ are

$$\|\boldsymbol{D}\|_0 \leq \|\boldsymbol{d}\|_0 \leq O(m(\omega L^{3/2})^{2/3}) = O(m\omega^{2/3}L), \tag{A.32}$$

$$\|\boldsymbol{D}'_l\boldsymbol{g}_l\| = \|\boldsymbol{d}\| \leq O(\omega L^{3/2}). \tag{A.33}$$

Proof of Statements 2 and 3. We prove statements 2 and 3 of Lemma A.4 by induction on $l \in \{0, 1, \cdots L\}$. These statements clearly hold at $l = 0$ because there is no perturbation at $l = 0$ and $\boldsymbol{g}'_0 = \boldsymbol{h}'_0 = \boldsymbol{0}$. In view of the previous part of the proof, we assume the lemma holds for layers $0 \leq j \leq l-1$ and prove that the second and third statements of the lemma hold at layer l.

Following the given definitions, we expand $\boldsymbol{g}'_l$ as follows

$$\begin{aligned}
\boldsymbol{g}'_l &= \boldsymbol{W}_l\boldsymbol{D}_{l-1}\boldsymbol{g}_{l-1} - \boldsymbol{W}_l^{(0)}\boldsymbol{D}_{l-1}^{(0)}\boldsymbol{g}_{l-1}^{(0)} \\
&= (\boldsymbol{W}_l^{(0)} + \boldsymbol{W}'_l)(\boldsymbol{D}_{l-1}^{(0)} + \boldsymbol{D}'_{l-1})(\boldsymbol{g}_{l-1}^{(0)} + \boldsymbol{g}'_{l-1}) - \boldsymbol{W}_l^{(0)}\boldsymbol{D}_{l-1}^{(0)}\boldsymbol{g}_{l-1}^{(0)} \\
&= \boldsymbol{W}'_l(\boldsymbol{D}_{l-1}^{(0)} + \boldsymbol{D}'_{l-1})(\boldsymbol{g}_{l-1}^{(0)} + \boldsymbol{g}'_{l-1}) + \boldsymbol{W}_l^{(0)}\boldsymbol{D}'_{l-1}(\boldsymbol{g}_{l-1}^{(0)} + \boldsymbol{g}'_{l-1}) + \boldsymbol{W}_l^{(0)}\boldsymbol{D}_{l-1}^{(0)}\boldsymbol{g}'_{l-1}.
\end{aligned} \tag{A.34}$$

We first expand $\boldsymbol{g}'_{l-1}$ in the last term of the above equation. Similarly, we then

iteratively expand $\boldsymbol{g}'_{l-2}, \ldots, \boldsymbol{g}'_1$ and obtain the following expression:

$$\begin{aligned}
\boldsymbol{g}'_l &= \boldsymbol{W}'_l(\boldsymbol{D}^{(0)}_{l-1} + \boldsymbol{D}'_{l-1})(\boldsymbol{g}^{(0)}_{l-1} + \boldsymbol{g}'_{l-1}) + \boldsymbol{W}^{(0)}_l \boldsymbol{D}'_{l-1}(\boldsymbol{g}^{(0)}_{l-1} + \boldsymbol{g}'_{l-1}) \\
&\quad + \boldsymbol{W}^{(0)}_l \boldsymbol{D}^{(0)}_{l-1}\big(\boldsymbol{W}'_{l-1}(\boldsymbol{D}^{(0)}_{l-2} + \boldsymbol{D}'_{l-2})(\boldsymbol{g}^{(0)}_{l-2} + \boldsymbol{g}'_{l-2}) + \boldsymbol{W}^{(0)}_{l-1}\boldsymbol{D}'_{l-2}(\boldsymbol{g}^{(0)}_{l-2} + \boldsymbol{g}'_{l-2})\big) \\
&\quad + \boldsymbol{W}^{(0)}_l \boldsymbol{D}^{(0)}_{l-1}\boldsymbol{W}^{(0)}_{l-1}\boldsymbol{D}^{(0)}_{l-2}\boldsymbol{g}'_{l-2} \\
&= \ldots \\
&= \sum_{k=0}^{l-1}\left(\prod_{j=1}^{k}\boldsymbol{W}^{(0)}_{l-j+1}\boldsymbol{D}^{(0)}_{l-j}\right)\Big(\boldsymbol{W}'_{l-k}(\boldsymbol{D}^{(0)}_{l-k-1} + \boldsymbol{D}'_{l-k-1})(\boldsymbol{g}^{(0)}_{l-k-1} + \boldsymbol{g}'_{l-k-1}) \\
&\quad + \boldsymbol{W}^{(0)}_{l-k}\boldsymbol{D}'_{l-k-1}(\boldsymbol{g}^{(0)}_{l-k-1} + \boldsymbol{g}'_{l-k-1})\Big) + \left(\prod_{j=1}^{l-1}\boldsymbol{W}^{(0)}_{l-j+1}\boldsymbol{D}^{(0)}_{l-j}\right)\boldsymbol{g}'_0.
\end{aligned}$$

Since $\boldsymbol{g}'_0 = \mathbf{0}$, the last term is $\mathbf{0}$. We consequently express $\boldsymbol{g}'_l$ as a sum of the following two terms:

$$\boldsymbol{g}'_l = \sum_{k=0}^{l-1}\left(\prod_{j=1}^{k}\boldsymbol{W}^{(0)}_{l-j+1}\boldsymbol{D}^{(0)}_{l-j}\right)\Big(\boldsymbol{W}'_{l-k}(\boldsymbol{D}^{(0)}_{l-k-1} + \boldsymbol{D}'_{l-k-1})(\boldsymbol{g}^{(0)}_{l-k-1} + \boldsymbol{g}'_{l-k-1})\Big) \tag{A.35}$$

$$+ \sum_{k=0}^{l-1}\left(\prod_{j=1}^{k}\boldsymbol{W}^{(0)}_{l-j+1}\boldsymbol{D}^{(0)}_{l-j}\right)\Big(\boldsymbol{W}^{(0)}_{l-k}\boldsymbol{D}'_{l-k-1}(\boldsymbol{g}^{(0)}_{l-k-1} + \boldsymbol{g}'_{l-k-1})\Big). \tag{A.36}$$

We estimate with high probability the above first term (right hand side in (A.35)) by using the assumption $\|\boldsymbol{W}'\| < \omega$ and the first statement in Lemma A.3 (to bound $\|\prod_{j=1}^{k}\boldsymbol{W}^{(0)}_{l-j+1}\boldsymbol{D}^{(0)}_{l-j}\|$, $k = 0, 1, \ldots l-1$). We thus obtain with probability at least

$1 - Le^{\Omega(m/L)}$

$$\begin{aligned}
&\left\| \sum_{k=0}^{l-1} \left(\prod_{j=1}^{k} \boldsymbol{W}_{l-j+1}^{(0)} \boldsymbol{D}_{l-j}^{(0)} \right) \left(\boldsymbol{W}_{l-k}' (\boldsymbol{D}_{l-k-1}^{(0)} + \boldsymbol{D}_{l-k-1}')(\boldsymbol{g}_{l-k-1}^{(0)} + \boldsymbol{g}_{l-k-1}') \right) \right\| \\
&\leq L \max_k \left\| \prod_{j=1}^{k} \boldsymbol{W}_{l-j+1}^{(0)} \boldsymbol{D}_{l-j}^{(0)} \right\| \left\| \boldsymbol{W}_{l-k}' (\boldsymbol{D}_{l-k-1}^{(0)} + \boldsymbol{D}_{l-k-1}')(\boldsymbol{g}_{l-k-1}^{(0)} + \boldsymbol{g}_{l-k-1}') \right\| \\
&\leq L \cdot O(\sqrt{L}) \cdot \max_k \|\boldsymbol{W}_{l-k}'\| \cdot \|\boldsymbol{D}_{l-k-1}\| \cdot \|\boldsymbol{g}_{l-k-1}^{(0)} + \boldsymbol{g}_{l-k-1}'\| \\
&\leq L \cdot O(\sqrt{L}) \cdot \omega \cdot \frac{\max(|\alpha|, 1)}{\sqrt{1+\alpha^2}} \cdot \max_k \|\boldsymbol{g}_{l-k-1}^{(0)} + \boldsymbol{g}_{l-k-1}'\| \\
&\leq O(\omega L^{3/2}) \max_k \|\boldsymbol{g}_{l-k-1}^{(0)} + \boldsymbol{g}_{l-k-1}'\|.
\end{aligned}$$

We further use Lemma A.1 to bound $\|\boldsymbol{g}_{l-k-1}^{(0)}\|$, $k \in \{0, 1, \ldots l-1\}$, by a constant and use the induction assumption to bound $\|\boldsymbol{g}_{l-k-1}'\|$, $k \in \{0, 1, \ldots l-1\}$, by $O(\omega L^{5/2}\sqrt{\ln m})$. With probability at least $1 - O(L)e^{-\Omega(m/L)}$, the first term (right hand side in (A.35)) is thus bounded by

$$O(\omega L^{3/2})(O(1) + O(\omega L^{5/2}\sqrt{\ln m})) = O(\omega L^{3/2}). \tag{A.37}$$

In order to bound the second term, which appears in (A.36), we denote

$$\boldsymbol{d}_k := \boldsymbol{D}_{l-k-1}'(\boldsymbol{g}_{l-k-1}^{(0)} + \boldsymbol{g}_{l-k-1}'), \quad k = 0, 1, \ldots l-1$$

and

$$\boldsymbol{y}_k := \left(\prod_{j=1}^{k} \boldsymbol{W}_{l-j+1}^{(0)} \boldsymbol{D}_{l-j}^{(0)} \right) \boldsymbol{W}_{l-k}^{(0)} \boldsymbol{d}_k.$$

We show it can be decomposed into $\boldsymbol{y}_k = \boldsymbol{y}_{k,1} + \boldsymbol{y}_{k,2}$, where with probability at least

$1 - Le^{-\Omega(m/L)}$,

$$\|\boldsymbol{y}_{k,1}\| \leq O\left(\frac{(1-\alpha)\omega L^{3/2}}{(1+\alpha^2)^{1/2}\sqrt{m}}\right), \quad \|\boldsymbol{y}_{k,2}\|_\infty \leq O\left(\frac{(1-\alpha)\omega L^{3/2}\sqrt{\ln m}}{(1+\alpha^2)^{1/2}\sqrt{m}}\right).$$

Denoting $\boldsymbol{u}_k := \boldsymbol{D}_{l-1}^{(0)}\boldsymbol{W}_{l-1}^{(0)}, ...\boldsymbol{D}_{l-k}^{(0)}\boldsymbol{W}_{l-k}^{(0)}\boldsymbol{d}_k$ and applying the induction assumption we note that $\|\boldsymbol{d}_k\|_0 < O(m\omega^{2/3}L)$. Next, we apply the third statement of the Lemma A.3 for $\boldsymbol{u}_k$ (instead of $\boldsymbol{v}$) and obtain that with probability at least $1 - e^{-\Omega(m/L)}$

$$\|\boldsymbol{u}_k\| \leq 4\|\boldsymbol{d}_k\|. \tag{A.38}$$

We note that $\boldsymbol{y}_k = \boldsymbol{W}_l^{(0)}\boldsymbol{u}_k$ and thus $\boldsymbol{y}_k|\boldsymbol{u}_k \sim N\left(0, \frac{2\|\boldsymbol{u}_k\|^2}{m}\boldsymbol{I}\right)$.

We denote $y_{k,j} := (\boldsymbol{y}_k)_j$ and $\sigma^2 := 2\|\boldsymbol{u}_k\|^2/m$ and we let $b = O(\|\boldsymbol{u}_k\|\sqrt{\ln m/m})$. We investigate the tail probability of the Gaussian random variable $y_{k,j}$ conditioned on $\boldsymbol{u}_k$. It is clear that

$$\mathbb{P}(|y_{k,j}| \geq bt|\boldsymbol{u}_k) \leq \frac{1}{\sqrt{2\pi}bt/\sigma}e^{-b^2t^2/2\sigma^2} \quad \forall t \in \mathbb{N}. \tag{A.39}$$

We denote $R_t := \{j : y_{k,j} \geq bt\} \subset [m]$ and $r_t := \sqrt{m}/((\ln m)^2t^2)$. Using the independence of $\{y_{k,j}\}_{j\in[m]}$ given $\boldsymbol{u}_k$ and applying a union bound for (A.39) yield

$$\begin{aligned}
\mathbb{P}(|R_t| \geq r_t|\boldsymbol{u}_k) &\leq \binom{m}{r_t} \times \left(\frac{1}{\sqrt{2\pi}bt/\sigma}e^{-b^2t^2/2\sigma^2}\right)^{r_t} \\
&\leq \left(\frac{\|\boldsymbol{u}_k\|}{\sqrt{\pi}bt\sqrt{m(1+\alpha^2)}}\right)^{r_t}\left(\frac{me}{r_t}\right)^{r_t} e^{-\Omega(b^2t^2mr_t)} \\
&\leq O(1)\exp\left(-\Omega(b^2t^2mr_t) + \left(\frac{1}{2}\ln m - \ln b - \Omega(1)\right)r_t\right).
\end{aligned}$$

Denoting $q := \sqrt{m}/\ln^2 m$, we simplify the above bound as follows

$$\mathbb{P}(|R_t| \geq q/t^2) \leq e^{-\Omega(b^2 qm)}.$$

We further denote $Q := \{0, 1, 2, 3, ..\lfloor \frac{1}{2}\log_2 q \rfloor\}$, $N_Q := \lfloor \frac{1}{2}\log_2 q \rfloor$ and $T := \{2^p : p \in Q\}$. We designate the elements in T by $t_p := 2^p$ for $p \in Q$. Let $t_{N_Q+1} := 2^{\lfloor \frac{1}{2}\log_2 q \rfloor + 1} \equiv 2^{N_Q+1}$ and notice that $t_{N_Q+1}^2 > q$. Thus, applying the above estimate and a union bound over $t \in T$ and t_{N_Q+1}

$$|R_t| < q/t^2,\ \forall t \in T, \text{ and } |R_{t_{N_Q+1}}| < 1 \text{ with probability at least } 1-(|T|+1)e^{-\Omega(b^2 qm)}.$$

By definition, we note that when $|R_{t_{N_Q+1}}| = 0$ and $|y_{k,j}| < t_{N_Q+1}$ for $j \in R_{t_{N_Q}}$. We also note that for $j \in R_{t_p} \setminus R_{t_{p+1}}$, $|y_{k,j}| < t_{p+1}$. Thus, for $R := R_1 \equiv \{j : |y_{k,j}| \geq b\}$, we bound $\sum_{j\in R} y_{k,j}^2$ with high probability as follows

$$\begin{aligned}
\sum_{j\in R} y_{k,j}^2 &= \sum_{j\in R/R_{t_{N_Q}}} y_{k,j}^2 + \sum_{j\in R_{t_{N_Q}}} y_{k,j}^2 \leq \sum_{j\in R/R_{t_{N_Q}}} y_{k,j}^2 + |R_{t_{N_Q}}|(bt_{N_Q+1})^2 \\
&\leq \sum_{j\in R/R_{t_{N_Q}}/R_{t_{N_Q-1}}} y_{k,j}^2 + \sum_{j\in R_{t_{N_Q-1}}/R_{t_{N_Q}}} y_{k,j}^2 + |R_{t_{N_Q}}|(bt_{N_Q+1})^2 \\
&\leq \sum_{j\in R/R_{t_{N_Q}}/R_{t_{N_Q-1}}} y_{k,j}^2 + |R_{t_{N_Q-1}}|(bt_{N_Q})^2 + |R_{t_{N_Q}}|(bt_{N_Q+1})^2 \\
&\ldots\ldots \\
&\leq \sum_{p\in Q} |R_{t_p}|(b2^{p+1})^2 \leq \sum_{p\in Q} q/t_p^2 (b2^{p+1})^2 \\
&= \sum_{p\in Q} qb^2 2^2 = O(qb^2 \ln q) \text{ with probability at least } 1-\Omega(|T|)e^{-\Omega(b^2 qm)}.
\end{aligned}$$

Since $b = O\left(\|\boldsymbol{u}_k\|\sqrt{\ln m/m}\right)$ and $q = \sqrt{m}/\ln^2 m$, we express the above bound as

$$\sum_{j\in R} y_{k,j}^2 \leq O(\|\boldsymbol{u}_k\|^2/m) \text{ with probability at least } 1 - e^{-\Omega(\frac{m^{1/2}}{\ln m})}. \tag{A.40}$$

We split vector $\boldsymbol{y}_k$ into $\boldsymbol{y}_k = \boldsymbol{y}_{k,1} + \boldsymbol{y}_{k,2}$ using the indices set R as

$$\boldsymbol{y}_{k,1} = (y_{k,1}1_{1\in R}, y_{k,2}1_{2\in R}, \ldots, y_{k,m}1_{m\in R})^T, \tag{A.41}$$

$$\boldsymbol{y}_{k,2} = (y_{k,1}1_{1\notin R}, y_{k,2}1_{2\notin R}, \ldots, y_{k,m}1_{m\notin R})^T. \tag{A.42}$$

Using (A.40) and the definition of R, and then the induction assumption on the bound of $\|\boldsymbol{d}_k\|$ and (A.38) yield the following estimates with probability at least $1 - e^{-\Omega\left(\frac{m^{1/2}}{\ln m}\right)}$:

$$\|\boldsymbol{y}_{k,1}\| \leq O\left((\frac{\|\boldsymbol{u}\|}{m^{1/2}}\right) \leq O\left((\frac{(1-\alpha)\omega L^{3/2}}{(1+\alpha^2)^{1/2}m^{1/2}}\right), \tag{A.43}$$

$$\|\boldsymbol{y}_{k,2}\|_\infty \leq b = O\left(\frac{\|\boldsymbol{u}\|\sqrt{\ln m}}{\sqrt{m}}\right) \leq O\left(\frac{(1-\alpha)\omega L^{3/2}\sqrt{\ln m}}{(1+\alpha^2)^{1/2}\sqrt{m}}\right). \tag{A.44}$$

Following the later decomposition of $\boldsymbol{y}_k$ (with the components in (A.41) and (A.42)), we decompose the term in (A.36) into $\sum_{k=0}^{l-1}\boldsymbol{y}_{k,1}$ and $\sum_{k=0}^{l-1}\boldsymbol{y}_{k,2}$. We denote $\boldsymbol{g}'_{l,2} := \sum_{k=0}^{l-1}\boldsymbol{y}_{k,2}$ and $\boldsymbol{g}'_{l,1} := \boldsymbol{g}'_l - \boldsymbol{g}'_{l,2}$. We note that $\boldsymbol{g}'_{l,1}$ is the sum of the term in (A.35) and $\sum_{k=0}^{l-1}\boldsymbol{y}_{k,1}$. By using the bound of (A.35) given in (A.37) and (A.43), we bound $\boldsymbol{g}'_{l,1}$

as follows

$$\begin{aligned}
\|\boldsymbol{g}'_{l,1}\| &\leq \left\| \sum_{k=0}^{l-1} \left(\prod_{j=1}^{k} \boldsymbol{W}^{(0)}_{l-j+1} \boldsymbol{D}^{(0)}_{l-j} \right) \left(\boldsymbol{W}'_{l-k} (\boldsymbol{D}^{(0)}_{l-k-1} + \boldsymbol{D}'_{l-k-1})(\boldsymbol{g}^{(0)}_{l-k-1} + \boldsymbol{g}'_{l-k-1}) \right) \right\| \\
&\quad + \sum_{k=0}^{l-1} \|\boldsymbol{y}_{k,1}\| \\
&\leq O(\omega L^{3/2}) + \sum_{k=0}^{l-1} \|\boldsymbol{y}_{k,1}\| \\
&\leq O(\omega L^{3/2}) + L \max_{k \in \{0,1,\ldots,l-1\}} \|\boldsymbol{y}_{k,1}\| \\
&\leq O(\omega L^{3/2}) + LO\left(\frac{(1-\alpha)\omega L^{3/2}}{(1+\alpha^2)^{1/2} m^{1/2}} \right).
\end{aligned}$$

Using the fact that $m \geq \Omega(L^2)$, we show the ℓ_2 norm for $\boldsymbol{g}'_{l,1}$ in the second statement of this lemma holds:

$$\|\boldsymbol{g}'_{l,1}\| \leq O(\omega L^{3/2}) + LO\left(\frac{(1-\alpha)\omega L^{3/2}}{(1+\alpha^2)^{1/2} m^{1/2}} \right) \leq O(\omega L^{3/2}).$$

Applying the induction assumption, i.e., $\|\boldsymbol{g}'_{l-k,1}\| \leq O(\omega L^{3/2})$ for $k \in \{0, 1, \ldots, l-1\}$, and (A.44), we conclude the second statement of the lemma for layer l as follows

$$\|\boldsymbol{g}'_{l,2}\|_\infty \leq \sum_{k=0}^{l-1} \|\boldsymbol{y}_{k,2}\|_\infty \leq LO\left(\frac{1-\alpha}{(1+\alpha^2)^{1/2}} \frac{\sqrt{\ln m}\omega L^{3/2}}{\sqrt{m}} \right) = O\left(\frac{\sqrt{\ln m}\omega L^{5/2}}{\sqrt{m}} \right). \tag{A.45}$$

Finally, we note that $\|\boldsymbol{g}'_l\| \leq \|\boldsymbol{g}'_{l,1}\| + \|\boldsymbol{g}'_{l,2}\|$, and thus the first part $\|\boldsymbol{g}'_{l,1}\|$ is bounded by $O(\omega L^{3/2})$. Furthermore, applying (A.45), we bound the second part, $\|\boldsymbol{g}'_{l,2}\|$, as follows

$$\|\boldsymbol{g}'_{l,2}\| = \sqrt{\sum_{j \in S_2} g^2_{l,2,j}} \leq \sqrt{m \frac{\ln m \omega^2 L^5}{m}} = \sqrt{\ln m}\omega L^{5/2}.$$

By definition, $\boldsymbol{h}_l' = \boldsymbol{D}\boldsymbol{g}_l' + \boldsymbol{D}'\boldsymbol{g}_l^{(0)} + \boldsymbol{D}'\boldsymbol{g}_l' = \boldsymbol{D}\boldsymbol{g}_l' + \boldsymbol{D}'\boldsymbol{g}_l$. Applying $\|\boldsymbol{D}\| \leq 1$, $\|\boldsymbol{g}_l'\| \leq O(\omega L^{5/2}\sqrt{\ln m})$ and $\|\boldsymbol{D}'\boldsymbol{g}_l\| \leq O(\omega L^{3/2})$, we bound the norm of $\boldsymbol{h}_l'$ in the following way

$$\|\boldsymbol{h}_l'\| \leq O(1)O(\omega L^{5/2}\sqrt{\ln m}) + O(\omega L^{3/2}) = O(\omega L^{5/2}\sqrt{\ln m}).$$

Thus the third statement of this lemma is concluded for layer l. □

Lemma A.5

For given integer a,b as $1 \leq a < b \leq L$, and if $d < O\left(\frac{m}{L\ln m}\right)$, $\|\boldsymbol{W}'\| \leq \omega < O\left(\frac{1}{L^{9/2}\ln^{3/2} m}\right)$. Then we obtain that with probability at least $1 - e^{-\Omega(m/L)}$

1. $\|\boldsymbol{W}_b^{(0)}(\boldsymbol{D}_{i,b-1}^{(0)} + \boldsymbol{D}_{i,b-1}')\boldsymbol{W}_{b-1}^{(0)}...(\boldsymbol{D}_{i,a}^{(0)} + \boldsymbol{D}_{i,a}')\boldsymbol{W}_a^{(0)}\| \leq O(\sqrt{L})$.
2. $\|(\boldsymbol{W}_b^{(0)} + \boldsymbol{W}_b')(\boldsymbol{D}_{i,b-1}^{(0)} + \boldsymbol{D}_{i,b-1}')(\boldsymbol{W}_{b-1}^{(0)} + \boldsymbol{W}_{b-1}')...(\boldsymbol{D}_{i,a}^{(0)} + \boldsymbol{D}_{i,a}')(\boldsymbol{W}_a^{(0)} + \boldsymbol{W}_a')\| \leq O(\sqrt{L})$.
3. $\|\boldsymbol{W}_{b+1}^{(0)}(\boldsymbol{D}_{i,b}^{(0)} + \boldsymbol{D}_{i,b}')(\boldsymbol{W}_b^{(0)} + \boldsymbol{W}_b')...(\boldsymbol{D}_{i,a}^{(0)} + \boldsymbol{D}_{i,a}') - \boldsymbol{W}_{b+1}^{(0)}\boldsymbol{D}_{i,b}^{(0)}\boldsymbol{W}_b^{(0)}...\boldsymbol{W}_{a+1}^{(0)}\boldsymbol{D}_{i,a}^{(0)}\| \leq O\left(\frac{1-\alpha}{\sqrt{1+\alpha^2}}L^{3/2}\right)$.
4. $\|\boldsymbol{B}(\boldsymbol{D}_L^{(0)} + \boldsymbol{D}_L')(\boldsymbol{W}_L^{(0)} + \boldsymbol{W}_L')...(\boldsymbol{D}_{i,a}^{(0)} + \boldsymbol{D}_{i,a}') - \boldsymbol{B}\boldsymbol{D}_L^{(0)}\boldsymbol{W}_L^{(0)}...\boldsymbol{W}_{a+1}^{(0)}\boldsymbol{D}_{i,a}^{(0)}\|$ $\leq O\left(\frac{1-\alpha}{\sqrt{1+\alpha^2}}\frac{\omega^{1/3}L^2\sqrt{m\ln m}}{\sqrt{d}}\right)$.

The proof of this lemma follows the same arguments of the proofs of Lemmas 8.6 and 8.7 in Allen-Zhu et al. (2019b), but uses instead Lemma A.4 and the fact that $\|\boldsymbol{D}'\| = (1-\alpha)/\sqrt{1+\alpha^2}$.

A.2.4 Gradient Bounds and Proof of Lemma 2.2

We first introduce two lemmas (Lemmas A.6 and A.7) that provide upper and lower bounds for the Frobenius norm of a certain matrix-valued function $\boldsymbol{G}_{i,l}(\boldsymbol{v}; \boldsymbol{W}^{(0)})$ with

randomly initialized parameters $\boldsymbol{W}^{(0)}$. This function, which is defined below in (A.46) equals the gradient of the loss function when $\boldsymbol{v} = \boldsymbol{e}_i^{(0)} \equiv \boldsymbol{B}\boldsymbol{h}_{L,i}^{(0)} - \boldsymbol{y}_i$. At last, we conclude Lemma 2.2 by applying the perturbation bounds of Lemmas A.4 and A.5 in order to show that the order of the bounds in Lemmas A.6 and A.7 are not affected by a small perturbation $\boldsymbol{W}'$ as long as $\|\boldsymbol{W}'\| \leq \omega < O\left(\frac{\delta^{3/2}}{n^{3/2}L^{15/2}\ln^{3/2} m}\right)$.

We remark that the proof of Lemma A.6 is straightforward and follows Allen-Zhu et al. (2019b). The proof of Lemma A.7 follows ideas of Zou and Gu (2019), while adapting it to Leaky ReLUs and improving the lower bound of $\|\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W}^{(0)})\|_F^2$ by quantifying lower bounds for layers before L instead of only using $\|\nabla_{\boldsymbol{W}_L}\mathcal{L}(\boldsymbol{W}^{(0)})\|_F^2$ as done in Zou and Gu (2019). This improvement reduces a factor L in the lower bound, which will eventually make the learning rate of the desired theory independent of L. The idea of concluding Lemma 2.2 by examining the effect of a small perturbation on the parameter follows Allen-Zhu et al. (2019b).

We define the matrix-valued function, $\boldsymbol{G}_{i,l}(\boldsymbol{v};\boldsymbol{W})$, for $l \in [L]$ and $i \in [n]$ and $\boldsymbol{v} \in \mathbb{R}^d$ as follows

$$\boldsymbol{G}_{i,l}(\boldsymbol{v};\boldsymbol{W}) := \boldsymbol{D}_{i,l}\mathbf{Back}_{i,l}^T\boldsymbol{v}\boldsymbol{h}_{i,l-1}^T = (\mathbf{Back}_{i,l}\boldsymbol{D}_{i,l})^T\boldsymbol{v}\boldsymbol{h}_{i,l-1}^T. \tag{A.46}$$

We note that $\boldsymbol{G}_{i,l}(\boldsymbol{v};\boldsymbol{W})$ is related to the gradient of the loss function as follows:

$$\boldsymbol{G}_{i,l}(\boldsymbol{e}_i;\boldsymbol{W}) \equiv \nabla_{\boldsymbol{W}_l}\mathrm{loss}(\boldsymbol{x}_i,\boldsymbol{y}_i;\boldsymbol{W}).$$

Lemma A.6

Assume the setup of §2.2 with randomly initialized $\boldsymbol{W}^{(0)}$. If $d \leq O(\frac{m}{L\ln m})$, then with probability at least $1 - e^{-\Omega(m/L)}$

$$\|\boldsymbol{G}_{i,l}(\boldsymbol{v};\boldsymbol{W}^{(0)})\|_F^2 \leq O\left(\frac{m}{d}\right)\|\boldsymbol{v}\|^2. \tag{A.47}$$

□

Proof: The second statement in Lemma A.3 implies that $\|\mathbf{Back}_{i,l}\| < O(\sqrt{\frac{m}{d}})$ with probability at least $1 - e^{-\Omega(m/L)}$ and therefore

$$\begin{aligned}\|\boldsymbol{G}_{i,l}(\boldsymbol{v}; \boldsymbol{W}^{(0)})\|_F^2 &\leq \|\boldsymbol{D}_{i,l}\mathbf{Back}_{i,l}^{(0)T}\boldsymbol{v}\boldsymbol{h}_{i,l-1}^{(0)T}\|_F^2 \\ &\leq \|\boldsymbol{D}_{i,l}\mathbf{Back}_{i,l}^{(0)T}\boldsymbol{v}\|^2\|\boldsymbol{h}_{i,l-1}^{(0)T}\|^2 \\ &\leq O\left(\frac{m}{d}\|\boldsymbol{v}_i\|_2^2\right).\end{aligned}$$

□

Lemma A.7

Assume the setup of §2.2 and with randomly initialized $\boldsymbol{W}^{(0)}$. For any set of vector $\{\boldsymbol{v}_i\}_{i=1}^n \subset \mathbb{R}^d$,

$$\|\sum_{i=1}^n \boldsymbol{G}_{i,l}(\boldsymbol{v}_i; \boldsymbol{W}^{(0)})\|_F^2 \geq \Omega\left(\frac{(1-\alpha)^2}{(1+\alpha^2)}\frac{\delta m}{ndL}\right)\sum_{i=1}^n \|\boldsymbol{v}_i\|^2$$

$$\text{with probability } \geq 1 - e^{-\Omega(m\delta^2)}.$$

□

Proof: We separate the proof of this lemma into four parts. In the first part, we define a set in $\mathbb{R}^m$ (see (A.48) below) and show two important properties of this set (see (A.49) and (A.51) below). In the second part, we establish a lower bound for a useful function (as defined in (A.55) below) with a probability at least 0.5. In the third part, we use this lower bound to establish a lower bound of the loss function with a positive probability. In the fourth part, we conclude the lemma by using all the results proved in the former three parts.

Since we assume randomly initialized parameters without training, we simply denote $\boldsymbol{h}_{i,l} := \boldsymbol{h}_{i,l}^{(0)}$ and $\boldsymbol{W} := \boldsymbol{W}^{(0)}$ across this proof.

Part 1. We arbitrarily fix $l \in [L]$ and recall that $\boldsymbol{h}_{i,l}$ is the output of lth layer.

We denote

$$\hat{\boldsymbol{h}}_{i,l} := \boldsymbol{h}_{i,l}/\|\boldsymbol{h}_{i,l}\|.$$

We form an orthogonal matrix $\boldsymbol{Q}_{i,l} \in \mathbb{R}^{m\times m}$ whose first column is $\hat{\boldsymbol{h}}_{i,l}$. We denote the matrix in $\mathbb{R}^{m\times(m-1)}$ which completes this vector by $\tilde{\boldsymbol{Q}}_{i,l}$, that is, $\boldsymbol{Q}_{i,l} := [\hat{\boldsymbol{h}}_{i,l}, \tilde{\boldsymbol{Q}}_{i,l}]$.

For a small constant $c_1 > 0$ (the choice of c_1 will be determined during the proof), we let $\gamma = c_1\delta/(nL\sqrt{m})$. For $i \in [n]$ and the fixed $l \in [L]$, we define

$$\mathcal{W}_{i,l} := \{\boldsymbol{w} \in \mathbb{R}^m : |\hat{\boldsymbol{h}}_{i,l}^T\boldsymbol{w}| < \gamma, |\langle\tilde{\boldsymbol{Q}}_{i,l}\tilde{\boldsymbol{Q}}_{i,l}^T\boldsymbol{w}, \hat{\boldsymbol{h}}_{j,l}\rangle| > 2\gamma, \forall j \in [n], j \neq i\} \subset \mathbb{R}^m. \quad \text{(A.48)}$$

We prove that for any choice of γ the sets $\mathcal{W}_{i,l}$, $i \in [n]$, have no intersection, that is,

$$\mathcal{W}_{i,l} \cap \mathcal{W}_{j,l} = \emptyset, \quad \forall i \neq j \in [n]. \quad \text{(A.49)}$$

For any $\boldsymbol{w} \in \mathcal{W}_{i,l}$, we need to prove that $\boldsymbol{w} \notin \mathcal{W}_{j,l}$, where $j \neq i \in [n]$. We prove this by contradiction. Given $\boldsymbol{w} \in \mathcal{W}_{i,l}$, we assume that there exists $j \neq i \in [n]$ such that $\boldsymbol{w} \in \mathcal{W}_{j,l}$. Since $\tilde{\boldsymbol{Q}}_{j,l}\tilde{\boldsymbol{Q}}_{j,l}^T = \boldsymbol{I} - \hat{\boldsymbol{h}}_{j,l}\hat{\boldsymbol{h}}_{j,l}^T$, we rewrite $\tilde{\boldsymbol{Q}}_{j,l}\tilde{\boldsymbol{Q}}_{j,l}^T\boldsymbol{w}$ as

$$\tilde{\boldsymbol{Q}}_{j,l}\tilde{\boldsymbol{Q}}_{j,l}^T\boldsymbol{w} = (\boldsymbol{I} - \hat{\boldsymbol{h}}_{j,l}\hat{\boldsymbol{h}}_{j,l}^T)\boldsymbol{w} = \boldsymbol{w} - \langle\boldsymbol{w}, \hat{\boldsymbol{h}}_{j,l}\rangle\hat{\boldsymbol{h}}_{j,l}. \quad \text{(A.50)}$$

Applying (A.50) and the fact that $\langle\boldsymbol{w}, \hat{\boldsymbol{h}}_{i,l}\rangle < \gamma$ and $\langle\boldsymbol{w}, \hat{\boldsymbol{h}}_{j,l}\rangle < \gamma$ for $\boldsymbol{w} \in \mathcal{W}_{i,l} \cap \mathcal{W}_{j,l}$ results in

$$\begin{aligned}
|\langle\tilde{\boldsymbol{Q}}_{j,l}\tilde{\boldsymbol{Q}}_{j,l}^T\boldsymbol{w}, \hat{\boldsymbol{h}}_{i,l}\rangle| &= |\langle\boldsymbol{w} - \langle\boldsymbol{w}, \hat{\boldsymbol{h}}_{j,l}\rangle\hat{\boldsymbol{h}}_{j,l}, \hat{\boldsymbol{h}}_{i,l}\rangle| \\
&\leq |\langle\boldsymbol{w}, \hat{\boldsymbol{h}}_{i,l}\rangle| + |\langle\boldsymbol{w}, \hat{\boldsymbol{h}}_{j,l}\rangle\langle\hat{\boldsymbol{h}}_{j,l}, \hat{\boldsymbol{h}}_{i,l}\rangle| \\
&< \gamma + \gamma|\langle\hat{\boldsymbol{h}}_{j,L}, \hat{\boldsymbol{h}}_{i,L}\rangle| \\
&\leq 2\gamma.
\end{aligned}$$

On the other hand, since $\boldsymbol{w} \in \mathcal{W}_{j,l}$, $|\langle \tilde{\boldsymbol{Q}}_{j,l}\tilde{\boldsymbol{Q}}_{j,l}^T\boldsymbol{w}, \hat{\boldsymbol{h}}_{i,l}\rangle| > 2\gamma$ for $i \neq j$, which contradicts the above equation. Therefore, we conclude (A.49).

Next, we assume $\boldsymbol{w} \sim N(0, \frac{2}{m}\boldsymbol{I})$ and prove that

$$\mathbb{P}(\boldsymbol{w} \in \mathcal{W}_{i,l}) \geq \Omega\left(\frac{\delta}{nL}\right). \tag{A.51}$$

The orthogonality of $\boldsymbol{Q}_{i,l}$ implies that $\hat{\boldsymbol{h}}_{i,l}^T\boldsymbol{w}$ and $\tilde{\boldsymbol{Q}}_{i,l}^T\boldsymbol{w}$ are independent. We thus express the probability (A.51) as follows

$$\mathbb{P}(\boldsymbol{w} \in \mathcal{W}_{i,l}) = \mathbb{P}(|\hat{\boldsymbol{h}}_{i,l}^T\boldsymbol{w}| < \gamma)\mathbb{P}(|\langle \tilde{\boldsymbol{Q}}_{i,l}\tilde{\boldsymbol{Q}}_{i,l}^T\boldsymbol{w}, \hat{\boldsymbol{h}}_{j,l}\rangle| > 2\gamma,\ \forall j \in [n], j \neq i). \tag{A.52}$$

We note that $\hat{\boldsymbol{h}}_{i,l}^T\boldsymbol{w} \sim N(0, \frac{2}{m})$ and thus express the first multiplicative term in (A.52) as

$$\mathbb{P}(|\hat{\boldsymbol{h}}_{i,l}^T\boldsymbol{w}| < \gamma) = \frac{\sqrt{m}}{\sqrt{4\pi}}\int_{-\gamma}^{\gamma} e^{-\frac{mx^2}{4}}dx \geq \Omega\left(\gamma\sqrt{m}\right),\ \text{when } \gamma\sqrt{m} < 1. \tag{A.53}$$

To express the second multiplicative term of (A.52), we first derive the distribution of $\hat{\boldsymbol{h}}_{j,l}^T\tilde{\boldsymbol{Q}}_{i,l}\tilde{\boldsymbol{Q}}_{i,l}^T\boldsymbol{w}$. Since $\tilde{\boldsymbol{Q}}_{i,l}\tilde{\boldsymbol{Q}}_{i,l}^T = \boldsymbol{I}_m - \hat{\boldsymbol{h}}_{i,l}\hat{\boldsymbol{h}}_{i,l}^T$ and $\tilde{\boldsymbol{Q}}_{i,l}^T\tilde{\boldsymbol{Q}}_{i,l} = \boldsymbol{I}_{m-1}$,

$$\begin{aligned}
\hat{\boldsymbol{h}}_{j,l}^T\tilde{\boldsymbol{Q}}_{i,l}\tilde{\boldsymbol{Q}}_{i,l}^T\boldsymbol{w} &\sim N\left(0, \frac{2}{m}\hat{\boldsymbol{h}}_{j,l}^T\tilde{\boldsymbol{Q}}_{i,l}\tilde{\boldsymbol{Q}}_{i,l}^T\tilde{\boldsymbol{Q}}_{i,l}\tilde{\boldsymbol{Q}}_{i,l}^T\hat{\boldsymbol{h}}_{j,l}\right) \\
&= N\left(0, \frac{2}{m}\hat{\boldsymbol{h}}_{j,l}^T(\boldsymbol{I} - \hat{\boldsymbol{h}}_{i,l}\hat{\boldsymbol{h}}_{i,l}^T)\hat{\boldsymbol{h}}_{j,l}\right) \\
&= N\left(0, (1 - \langle\hat{\boldsymbol{h}}_{j,l}, \hat{\boldsymbol{h}}_{i,l}\rangle^2)\frac{2}{m}\right).
\end{aligned}$$

By Lemma A.2, we recall that with probability at least $1 - e^{-\Omega(\delta^4 m/L^4)}$,

$$\langle\hat{\boldsymbol{h}}_{i,L}, \hat{\boldsymbol{h}}_{j,L}\rangle^2 \leq 1 - \Omega(\delta^2/L^2), \quad \text{for all } i \neq j \in [n].$$

We thus note that $\hat{\boldsymbol{h}}_{j,l}^T \tilde{\boldsymbol{Q}}_{i,l} \tilde{\boldsymbol{Q}}_{i,l}^T \boldsymbol{w} \sim N(0, \tau^2)$, where τ^2 is greater than $\Omega(\delta^2/mL^2)$. Consequently,

$$\mathbb{P}(|\hat{\boldsymbol{h}}_{j,l}^T \tilde{\boldsymbol{Q}}_{i,l} \tilde{\boldsymbol{Q}}_{i,l}^T \boldsymbol{w}| < 2\gamma) = \frac{1}{\sqrt{2\pi\tau^2}} \int_{-2\gamma}^{2\gamma} \exp\left(-\frac{x^2}{2\tau^2}\right) dx \leq O\left(\frac{\gamma}{\tau}\right) \leq O\left(\frac{\gamma L\sqrt{m}}{\delta}\right).$$

Applying a union bound over all $j \in [n]$, $j \neq i$, yields

$$\mathbb{P}\left(\exists j \in [n], j \neq i \text{ such that } |\hat{\boldsymbol{h}}_{j,l}^T \tilde{\boldsymbol{Q}}_{i,l} \tilde{\boldsymbol{Q}}_{i,l}^T \boldsymbol{w}| \leq 2\gamma\right) \leq nO\left(\frac{\gamma L\sqrt{m}}{\delta}\right).$$

Consequently,

$$\mathbb{P}\left(|\hat{\boldsymbol{h}}_{j,l}^T \tilde{\boldsymbol{Q}}_{i,l} \tilde{\boldsymbol{Q}}_{i,l}^T \boldsymbol{w}| > 2\gamma \ \forall j \in [n], j \neq i\right) \geq 1 - O\left(\frac{\gamma nL\sqrt{m}}{\delta}\right). \tag{A.54}$$

Plugging (A.54) and (A.53) into (A.52) yields

$$\begin{aligned}\mathbb{P}(\boldsymbol{w} \in \mathcal{W}_{i,l}) &= \mathbb{P}(|u_{i,1}| < \gamma)\mathbb{P}(|\boldsymbol{v}_{i,j}| > 2\gamma, \ \forall j \in [n], j \neq i) \\ &\geq \Omega\left(\gamma\sqrt{m}\right)\left(1 - O\left(\frac{\gamma nL\sqrt{m}}{\delta}\right)\right).\end{aligned}$$

Recall that $\gamma = c_1\delta/(nL\sqrt{m})$, we select small c_1 such that both $O(\frac{\gamma nL\sqrt{m}}{\delta}) = O(1)\cdot c_1 < 1$ and $\gamma\sqrt{m} = c_1\delta/(nL) < 1$. We thus conclude this part as follows

$$\mathbb{P}(\boldsymbol{w} \in \mathcal{W}_{i,l}) \geq \Omega\left(\frac{\delta}{nL}\right).$$

Part 2. Given integer $k \in [m]$ and $l \in [L]$, we define the following vector-valued function for $\boldsymbol{a} = (a_1, ...a_n)^T \in \mathbb{R}^n$ and $\boldsymbol{w} \in \mathbb{R}^m$:

$$\boldsymbol{b}_{k,l}(\boldsymbol{w}, \boldsymbol{a}) := \sum_{i=1}^{n} a_i \tilde{\sigma}'_\alpha(\langle \boldsymbol{w}, \boldsymbol{h}_{i,l}\rangle)\boldsymbol{h}_{i,l}. \tag{A.55}$$

We prove that conditioning on the event $\boldsymbol{w} \in \mathcal{W}_{i,l}$, a certain lower bound of $\|\boldsymbol{b}_{k,l}(\boldsymbol{w}, \boldsymbol{a})\|$ is achieved with a probability at least 0.5, that is,

$$P\left(\|\boldsymbol{b}_{k,l}(\boldsymbol{w}, \boldsymbol{a})\| \geq \frac{a_i}{2}\frac{(1-\alpha)}{\sqrt{1+\alpha^2}}\|\boldsymbol{h}_{i,l}\| \mid \boldsymbol{w} \in \mathcal{W}_{i,l}\right) > \frac{1}{2}.$$

We rewrite $\boldsymbol{w}$ as $\boldsymbol{w} = \boldsymbol{Q}_{i,l}\boldsymbol{Q}_{i,l}^T\boldsymbol{w} = (\hat{\boldsymbol{h}}_{i,l}^T\boldsymbol{w})\hat{\boldsymbol{h}}_{i,l} + \tilde{\boldsymbol{Q}}_{i,k}\tilde{\boldsymbol{Q}}_{i,k}^T\boldsymbol{w}$,

$$\langle \boldsymbol{w}, \hat{\boldsymbol{h}}_{j,l}\rangle = (\hat{\boldsymbol{h}}_{i,l}^T\boldsymbol{w})\langle \hat{\boldsymbol{h}}_{i,l}, \hat{\boldsymbol{h}}_{j,l}\rangle + \langle \tilde{\boldsymbol{Q}}_{i,k}\tilde{\boldsymbol{Q}}_{i,k}^T\boldsymbol{w}, \hat{\boldsymbol{h}}_{j,l}\rangle \quad \text{for} \quad j \neq i$$

Using the following two facts: $\boldsymbol{w} \in \mathcal{W}_{i,l}$ and both $\hat{\boldsymbol{h}}_{i,l}$ and $\hat{\boldsymbol{h}}_{j,l}$ are unit vectors, we bound the absolute value of the first term of the above expression as folows

$$|(\hat{\boldsymbol{h}}_{i,l}^T\boldsymbol{w})||\langle \hat{\boldsymbol{h}}_{i,l}, \hat{\boldsymbol{h}}_{j,l}\rangle| < \gamma.$$

Since $\boldsymbol{w} \in \mathcal{W}_{i,l}$, the magnitude of the second term is greater than 2γ. We note that the sign of $\langle \boldsymbol{w}, \hat{\boldsymbol{h}}_{j,l}\rangle$ is the same as that of $\langle \tilde{\boldsymbol{Q}}_{i,k}\tilde{\boldsymbol{Q}}_{i,k}^T\boldsymbol{w}, \hat{\boldsymbol{h}}_{j,l}\rangle$. This and the piecewise linearity of the Leaky ReLU function imply that for $\boldsymbol{w} \in \mathcal{W}_{i,l}$

$$\tilde{\sigma}'_\alpha(\langle \boldsymbol{w}, \hat{\boldsymbol{h}}_{j,l}\rangle) = \tilde{\sigma}'_\alpha(\langle \tilde{\boldsymbol{Q}}_{i,k}\tilde{\boldsymbol{Q}}_{i,k}^T\boldsymbol{w}, \hat{\boldsymbol{h}}_{j,l}\rangle\rangle), \quad \text{for} \quad j \neq i. \tag{A.56}$$

We note (A.56) implies the following expression for $\boldsymbol{b}_{k,l}(\boldsymbol{w}, \boldsymbol{a})$ for $\boldsymbol{w} \in \mathcal{W}_{i,l}$: by ,

$$\begin{aligned}
\boldsymbol{b}_{k,l}(\boldsymbol{w}, \boldsymbol{a}) &= a_i\tilde{\sigma}'_\alpha(\hat{\boldsymbol{h}}_{i,l}^T\boldsymbol{w})\boldsymbol{h}_{i,l} + \sum_{j\neq i} a_j\tilde{\sigma}'_\alpha(\hat{\boldsymbol{h}}_{j,l}^T\boldsymbol{w})\boldsymbol{h}_{j,l} \\
&= a_i\tilde{\sigma}'_\alpha(\hat{\boldsymbol{h}}_{i,l}^T\boldsymbol{w})\boldsymbol{h}_{i,l} + \sum_{j\neq i} a_j\tilde{\sigma}'_\alpha(\langle \tilde{\boldsymbol{Q}}_{i,k}\tilde{\boldsymbol{Q}}_{i,k}^T\boldsymbol{w}, \hat{\boldsymbol{h}}_{j,l}\rangle\rangle)\boldsymbol{h}_{j,l} \\
&= a_i\frac{(1-\alpha)}{\sqrt{1+\alpha^2}}1_{\hat{\boldsymbol{h}}_{i,l}^T\boldsymbol{w}>0}\boldsymbol{h}_{i,l} + a_i\frac{\alpha}{\sqrt{1+\alpha^2}}\boldsymbol{h}_{i,l} \\
&\quad + \sum_{j\neq i} a_j\tilde{\sigma}'_\alpha(\langle \tilde{\boldsymbol{Q}}_{i,k}\tilde{\boldsymbol{Q}}_{i,k}^T\boldsymbol{w}, \hat{\boldsymbol{h}}_{j,l}\rangle\rangle)\boldsymbol{h}_{j,l}.
\end{aligned}$$

We denote

$$\boldsymbol{b}_1 := a_i \frac{(1-\alpha)}{\sqrt{1+\alpha^2}} \boldsymbol{h}_{i,L-1}$$
$$\boldsymbol{r} := a_i \frac{\alpha}{\sqrt{1+\alpha^2}} \boldsymbol{h}_{i,L-1} + \sum_{j \neq i} a_j \phi'_\alpha(\langle \tilde{\boldsymbol{Q}}_i \tilde{\boldsymbol{u}}_i, \hat{\boldsymbol{h}}_{j,L-1} \rangle) \boldsymbol{h}_{j,L-1},$$

and thus express $\boldsymbol{b}_{k,l}(\boldsymbol{w}, \boldsymbol{a})$ as follows

$$\boldsymbol{b}_{k,l}(\boldsymbol{w}, \boldsymbol{a}) = \boldsymbol{b}_1 1_{\hat{\boldsymbol{h}}_{i,l}^T \boldsymbol{w} > 0} + \boldsymbol{r}. \tag{A.57}$$

By symmetry of normal distribution, we know that $\hat{\boldsymbol{h}}_{i,l}^T \boldsymbol{w} > 0$ with probability 0.5. We also note that $\hat{\boldsymbol{h}}_{i,l}^T \boldsymbol{w}$ and $\tilde{\boldsymbol{Q}}_{i,l}^T \boldsymbol{w}$ are independent and thus $1_{\hat{\boldsymbol{h}}_{i,l}^T \boldsymbol{w} > 0}$ is independent with $\boldsymbol{r}$.

We consider two possibility for $\boldsymbol{r}$:

- When $\|\boldsymbol{r}\| \geq \frac{1}{2}\|\boldsymbol{b}_1\|$, we know that with probability 0.5, $\hat{\boldsymbol{h}}_{i,l}^T \boldsymbol{w} \leq 0$, which implies $\boldsymbol{b}_{k,l}(\boldsymbol{w}, \boldsymbol{a}) = \boldsymbol{r}$, and thus $\|\boldsymbol{b}_{k,l}(\boldsymbol{w}, \boldsymbol{a})\| \geq \frac{1}{2}\|\boldsymbol{b}_1\|$. We thus note that at least with probability 0.5 that $\|\boldsymbol{b}_{k,l}(\boldsymbol{w}, \boldsymbol{a})\| \geq \frac{1}{2}\|\boldsymbol{b}_1\|$.

- When $\|\boldsymbol{r}\| < \frac{1}{2}\|\boldsymbol{b}_1\|$, we note that $\hat{\boldsymbol{h}}_{i,l}^T \boldsymbol{w} > 0$ with probability 0.5, then by triangle inequality, we imply $\|\boldsymbol{b}_{k,l}(\boldsymbol{w}, \boldsymbol{a})\| \geq \|\boldsymbol{b}_1\| - \|\boldsymbol{r}\| \geq \frac{1}{2}\|\boldsymbol{b}_1\|$.

We conclude that

$$P\left(\|\boldsymbol{b}_{k,l}(\boldsymbol{w}, \boldsymbol{a})\| \geq \frac{a_i}{2} \frac{(1-\alpha)}{\sqrt{1+\alpha^2}} \|\boldsymbol{h}_{i,l}\| \,\middle|\, \boldsymbol{w} \in \mathcal{W}_{i,l} \right) \geq \frac{1}{2}. \tag{A.58}$$

Part 3. The proof of this part does not depend on a particular choice of $i \in [n]$. For simplicity, we thus drop the subscript i in this part.

For $\boldsymbol{v} \in \mathbb{R}^d$, $k \in [m]$ and $l \in [L]$, we define $a_{k,l} := \langle (\mathbf{Back}_l)_{\cdot,k}, \boldsymbol{v} \rangle$. We want to

show that for any integers $k \in [m]$ and $l \in [L]$,

$$\mathbb{P}\left((a_{k,l})^2 \geq O\left(\frac{\|\boldsymbol{v}\|^2}{d}\right)\right) > 1 - \exp(-O(1)). \tag{A.59}$$

To prove the above statement, we also need an auxiliary statement for $l \in \{2, 3, \ldots L+1\}$,

$$\|\boldsymbol{D}_{l-1}\mathbf{Back}_l^T \boldsymbol{v}\| \geq (1-\epsilon)\sqrt{\frac{m}{2d}}\|\boldsymbol{v}\| \text{ with probability at least } 1-e^{-\Omega(m\epsilon^2/L^2)}. \tag{A.60}$$

In order to prove the above two statements (A.59) and (A.60), we first prove that $\boldsymbol{W}_l\Big|\boldsymbol{D}_l$ has the same distribution as $\boldsymbol{W}_l$, i.e., $N(0, \frac{2}{m})$. Then we use a similar argument to that in the proof of Lemma A.1 in order to show (A.60). Finally, by using the distribution of $\boldsymbol{W}_l$ given $\boldsymbol{D}_l$, together with (A.60), we prove (A.59) and conclude this part.

We prove a more general statement for conditional distributions: given a normal random vector in $\mathbb{R}^p$ as $\boldsymbol{w} \sim N(0, \sigma^2 \boldsymbol{I}_p)$, and a random vector $\boldsymbol{h} \in \mathbb{R}^p$ that satisfies following three properties:

1. $\boldsymbol{h}$ is independent with $\boldsymbol{w}$
2. The norm $\|\boldsymbol{h}\|$ is independent with the direction $\boldsymbol{h}/\|\boldsymbol{h}\|$
3. The direction $\boldsymbol{h}/\|\boldsymbol{h}\|$ is uniform distribution in the unit sphere $\mathcal{S}^{p-1}$

We further define $B := 1_{\boldsymbol{h}^T\boldsymbol{w}>0}$ as a random variable. Then the conditional distribution of $\boldsymbol{w}|B$ is the same as the unconditional distribution of $\boldsymbol{w}$, that is

$$\boldsymbol{w}|B \overset{d}{=} \boldsymbol{w} \sim N(0, \sigma^2 \boldsymbol{I}_p). \tag{A.61}$$

Remark: a normal random vector $N(\mathbf{0}, \sigma^2 \boldsymbol{I})$ satisfies the above three properties and thus $\boldsymbol{w}$ also satisfies above three properties.

We denote the unit vectors $\hat{\boldsymbol{h}} := \boldsymbol{h}/\|\boldsymbol{h}\|$ and $\hat{\boldsymbol{w}} := \boldsymbol{w}/\|\boldsymbol{w}\|$. We first note that $B \equiv 1_{\hat{\boldsymbol{h}}^T\hat{\boldsymbol{w}}>0}$ only depends on the directions of $\boldsymbol{h}$ and $\boldsymbol{w}$. By the former observation and the fact that $\|\boldsymbol{w}\|$ is independent with $\hat{\boldsymbol{w}}$, we thus note $\|\boldsymbol{w}\||B = \|\boldsymbol{w}\|$. We denote the probability density function for a random variable Y by f_Y. We next consider the probability density function $f_{\boldsymbol{w}|B}(\boldsymbol{w})$, by independence of the norm and the direction for $\boldsymbol{w}$, we obtain

$$f_{\boldsymbol{w}|B}(\boldsymbol{w}) = f_{\boldsymbol{w}|B}(\|\boldsymbol{w}\|, \hat{\boldsymbol{w}}) = f_{\|\boldsymbol{w}\||B}(\|\boldsymbol{w}\|) f_{\hat{\boldsymbol{w}}|B}(\hat{\boldsymbol{w}}) = f_{\|\boldsymbol{w}\|}(\|\boldsymbol{w}\|) f_{\hat{\boldsymbol{w}}|B}(\hat{\boldsymbol{w}}). \quad \text{(A.62)}$$

Thus, in order to show (A.61), it is sufficient suffices to show that $\hat{\boldsymbol{w}}|B \overset{d}{=} \hat{\boldsymbol{w}}$. We prove this by showing that for any set $\mathcal{A} \subset \mathcal{S}^{p-1}$ in unit sphere, $\mathbb{P}(\hat{\boldsymbol{w}} \in \mathcal{A}|B = b) = \mathbb{P}(\hat{\boldsymbol{w}} \in \mathcal{A})$ for any $b = 0$ or 1. Given $\hat{\boldsymbol{h}}$ is uniform in unit sphere, we know that for any fixed direction $\hat{\boldsymbol{w}}$, $\mathbb{P}(\hat{\boldsymbol{h}}^T\hat{\boldsymbol{w}} > 0) = 0.5$. By Bayes formula, former observation, and $\hat{\boldsymbol{h}}$ is uniform in $\mathcal{S}^{p-1}$

$$\begin{aligned}
\mathbb{P}(\hat{\boldsymbol{w}} \in \mathcal{A}|B = 1) &= \frac{\mathbb{P}(\hat{\boldsymbol{w}} \in \mathcal{A},\ B = 1)}{\mathbb{P}(B = 1)} \\
&= \frac{\mathbb{P}(B = 1|\hat{\boldsymbol{w}} \in \mathcal{A})\mathbb{P}(\hat{\boldsymbol{w}} \in \mathcal{A})}{\int_{\hat{\boldsymbol{w}}} \mathbb{P}(\hat{\boldsymbol{h}}^T\hat{\boldsymbol{w}} > 0|\hat{\boldsymbol{w}}) f_{\hat{\boldsymbol{w}}}(\hat{\boldsymbol{w}})} \\
&= \frac{\int_{\mathcal{A}} \mathbb{P}(B = 1|\hat{\boldsymbol{w}} = \hat{\boldsymbol{w}}) f_{\hat{\boldsymbol{w}}}(\hat{\boldsymbol{w}}) d\hat{\boldsymbol{w}}}{\int_{\mathcal{S}^{p-1}} \mathbb{P}(\hat{\boldsymbol{h}}^T\hat{\boldsymbol{w}} > 0|\hat{\boldsymbol{w}}) f_{\hat{\boldsymbol{w}}}(\hat{\boldsymbol{w}}) d\hat{\boldsymbol{w}}} \\
&= \frac{0.5 \int_{\mathcal{A}} f_{\hat{\boldsymbol{w}}}(\hat{\boldsymbol{w}}) d\hat{\boldsymbol{w}}}{0.5 \int_{\mathcal{S}^{p-1}} f_{\hat{\boldsymbol{w}}}(\hat{\boldsymbol{w}}) d\hat{\boldsymbol{w}}} \\
&= \int_{\mathcal{A}} f_{\hat{\boldsymbol{w}}}(\hat{\boldsymbol{w}}) d\hat{\boldsymbol{w}} = \mathbb{P}(\hat{\boldsymbol{w}} \in \mathcal{A}).
\end{aligned}$$

A similar argument leads to $\mathbb{P}(\hat{\boldsymbol{w}} \in \mathcal{A}|B = 0) = \mathbb{P}(\hat{\boldsymbol{w}} \in \mathcal{A})$. By (A.62) and above argument, we conclude (A.61).

Given the symmetry of normal distribution, we conclude that $\boldsymbol{g}_l$ satisfies the three properties we required for $\boldsymbol{h}$ above. Together with the fact that $(\boldsymbol{W}_l)_{k,\cdot}$ is normal

$N(0, 2/m\boldsymbol{I}_m)$, we thus conclude that $(\boldsymbol{W}_l)_{k,\cdot}|(\boldsymbol{D}_l)_{kk}$ is still normal $N(0, 2/m\boldsymbol{I}_m)$.

Next, we estimate the norm of $\boldsymbol{D}_{l-1}^T\mathbf{Back}_l^T\boldsymbol{v}$. We define vector $\boldsymbol{z}_l := \boldsymbol{D}_l\mathbf{Back}_{l+1}^T\boldsymbol{v}$ for $l \in [L]$ and $\boldsymbol{z}_{L+1} := \boldsymbol{v}$. We first note $\boldsymbol{z}_L = \boldsymbol{D}_L\boldsymbol{B}^T\boldsymbol{v}$ and $(\boldsymbol{B}^T\boldsymbol{v})_j \sim N(0, \|\boldsymbol{v}\|^2/d)$ for $j \in [m]$. By denoting Bournulli random variables $B_{L,j} := 1_{(\boldsymbol{g}_L)_j>0}$, each index of $\boldsymbol{z}_L$ can be expressed as

$$(\boldsymbol{z}_L)_j^2 = \frac{B_{L,j} + \alpha^2(1 - B_{L,j})}{1 + \alpha^2}(\boldsymbol{B}^T\boldsymbol{v})_j^2 \text{ for } j \in [m].$$

We denote $Q_L := \{j : B_{L,j} = 1\}$. Conditioning on Q_L, denote two independent random variables $H_{L,1} \sim \chi^2(|Q_L|)$ and $H_{L,2} \sim \chi^2(m - |Q_L|)$, we note

$$\|\boldsymbol{z}_L\|^2\Big|Q_L \overset{d}{=} \frac{\|\boldsymbol{v}\|^2}{d(1+\alpha^2)}H_{L,1} + \frac{\alpha^2\|\boldsymbol{v}\|^2}{d(1+\alpha^2)}H_{L,2}.$$

By symmetry of random variables before L layer, we know $B_{L,j} \sim$ Bournulli(0.5) and then by Chernoff bound on binomial distribution, we note that with probability at least $1 - e^{-\Omega(m\epsilon^2)}$, $|Q_L| \in [(0.5 - \epsilon/2)m, (0.5 + \epsilon/2)m]$. Given this even happen, by using tail probability for chi-squared distribution, we note that

$$\mathbb{P}(H_{L,1} < 0.5m(1 - \epsilon)) < e^{-\Omega(m\epsilon^2)}.$$

Similarly,

$$\mathbb{P}(H_{L,2} < 0.5m(1 - \epsilon)) < e^{-\Omega(m\epsilon^2)}.$$

By taking event $|Q_L| \in [(0.5 - \epsilon/2)m, (0.5 + \epsilon/2)m]$ and using above probabilities, we conclude the lower bound for $\|\boldsymbol{z}_L\|$

$$\|\boldsymbol{z}_L\|^2 \geq \frac{m\|\boldsymbol{v}\|^2}{2d}(1 - \epsilon) \text{ with probability at least } 1 - \Omega(e^{-\Omega(m\epsilon^2)}). \tag{A.63}$$

We note that $\boldsymbol{z}_{l-1} = \boldsymbol{D}_{l-1}^T\boldsymbol{W}_l^T\boldsymbol{z}_l$. Conditioning on $\boldsymbol{z}_l$, we note that $\boldsymbol{W}_l|\boldsymbol{z}_l \equiv \boldsymbol{W}_l|\boldsymbol{D}_l$

is a random matrix whose entries are i.i.d $N(0, 2/m)$. We denote a random variable $B_{l,j} := 1_{(\boldsymbol{g}_l)_j > 0}$, then

$$\|\boldsymbol{z}_{l-1}\|^2 | \boldsymbol{z}_l = \sum_{j=1}^{m} \left(\frac{B_{l,j} + \alpha^2(1 - B_{l,j})}{1 + \alpha^2} \left(\sum_i (\boldsymbol{W}_l)_{i,j} (\boldsymbol{z}_l)_i \right)^2 \right) \Big| \boldsymbol{z}_l.$$

We note that $(\sum_i \boldsymbol{W}_l)_{i,j} (\boldsymbol{z}_l)_i | \boldsymbol{z}_l \sim N(0, 2\|\boldsymbol{z}_l\|^2/m)$. We denote the indices set where $B_{l,j} = 1$ by $Q_l := \{j : B_{l,j} = 1\}$ and conditioning on Q_l, we further denote two independent random variables $H_{l,1} \sim \chi^2(|Q_l|)$ and $H_{l,2} \sim \chi^2(m - |Q_l|)$. We note that conditioning on Q_l, by similar argument we used above in proof of Lemma A.1, we know that

$$\|\boldsymbol{z}_{l-1}\|^2 \Big| \boldsymbol{z}_l, Q_l \overset{d}{=} \frac{2\|\boldsymbol{z}_l\|^2}{m(1+\alpha^2)} H_{l,1} + \frac{2\alpha^2 \|\boldsymbol{z}_l\|^2}{m(1+\alpha^2)} H_{l,2}. \tag{A.64}$$

By the same argument to derive (A.63), we know that by Chernoff bound for binomial distribution, with probability at least $1 - e^{-\Omega(m\epsilon^2)}$, $|Q_l| \in [(0.5 - \epsilon/2)m, (0.5 + \epsilon/2)m]$, thus we note that

$$\mathbb{P}(H_{l,1} < 0.5m(1-\epsilon)) < e^{-\Omega(m\epsilon^2)}, \;\; \mathbb{P}(H_{l,2} < 0.5m(1-\epsilon)) < e^{-\Omega(m\epsilon^2)}.$$

Consequently,

$$\|\boldsymbol{z}_{l-1}\|^2 \geq \|\boldsymbol{z}_l\|^2 (1-\epsilon) \;\; \text{with probability at least } 1 - \Omega(e^{-\Omega(m\epsilon^2)}). \tag{A.65}$$

For any positive number ϵ_0, when we choose $\epsilon = \epsilon_0/L$ in (A.63) and (A.65), and then by $(1 - \epsilon_0/L)^L > 1 - \epsilon_0$, we conclude that

$$\|\boldsymbol{z}_l\|^2 \geq \frac{m}{2d} \|\boldsymbol{v}\|^2 (1 - \epsilon_0) \text{ for all } l \in [L], \;\; \text{with probability at least } 1 - \Omega(L) e^{-\Omega(m\epsilon_0^2/L^2)}.$$

(A.66)

Finally, recall that $a_{k,l} = \langle (\mathbf{Back}_l)_{\cdot,k}, \boldsymbol{v} \rangle$ and by definition of $\boldsymbol{z}_l$ in above proof, we note that $a_{k,l} \equiv \langle (\boldsymbol{W}_l)_{\cdot,k}, \boldsymbol{z}_l \rangle$. We note that $(\boldsymbol{W}_l)_{\cdot,k} | \boldsymbol{z}_l = (\boldsymbol{W}_l)_{\cdot,k} | \boldsymbol{D}_l$, by first statement we proved in this part, we further can derive that $(\boldsymbol{W}_l)_{\cdot,k} | \boldsymbol{z}_l \sim N(0, 2/m\boldsymbol{I})$. Thus, we know that conditioning on $\boldsymbol{z}_l$,

$$a_{k,l} | \boldsymbol{z}_l \sim N(0, \frac{2\|\boldsymbol{z}_l\|^2}{m}),$$

By the tail probability of normal, we note that the with a constant probability that $a_{k,l}$ is lower bounded as

$$\mathbb{P}\left((a_{k,l})^2 \geq O\left(\frac{2\|\boldsymbol{z}_l\|^2}{m} \right) \right) > 1 - \exp(-\Omega(1)).$$

Combining with (A.66), which holds with an overwhelming probability, with a small constant choice of ϵ_0, we conclude

$$\mathbb{P}\left((a_{k,l})^2 \geq O\left(\frac{\|\boldsymbol{v}\|^2}{d} \right) \right) > 1 - \exp(-\Omega(1)) \text{ for} l \in [L].$$

Lastly, we also show this is also true for $l = L+1$. Recall that $\mathbf{Back}_{L+1} \equiv \boldsymbol{B}$ and that $a_{k,L+1} \equiv \langle \boldsymbol{B}_{\cdot,k}, \boldsymbol{v} \rangle \sim N\left(0, \frac{\|\boldsymbol{v}\|^2}{d}\right)$. By using normal distribution property,

$$\mathbb{P}\left((a_{k,L+1})^2 \geq O\left(\frac{\|\boldsymbol{v}\|^2}{d} \right) \right) > 1 - \exp(-\Omega(1)).$$

We conclude this part by the final statement that

$$\mathbb{P}\left((a_{k,l})^2 \geq O\left(\frac{\|\boldsymbol{v}\|^2}{d} \right) \right) > 1 - \exp(-\Omega(1)) \quad \text{for } l \in [L+1]. \tag{A.67}$$

Part 4. We denote a vector $\boldsymbol{a}_{k,l} \in \mathbb{R}^n$ by denoting its entries as $(\boldsymbol{a}_{k,l})_i := \langle (\mathbf{Back}_{i,l})_{\cdot,k}, \boldsymbol{v}_i \rangle$

for $i \in [n]$. By definition (A.55), we note that $\boldsymbol{b}_{k,l-1}((\boldsymbol{W}_l)_{k,\cdot}, \boldsymbol{a}_{k,l+1}) \equiv (\sum_{i=1}^n \boldsymbol{G}_{i,l}(\boldsymbol{v}_i; \boldsymbol{W}))_{k,\cdot}$, by the definition of Frobenius norm of a vector of matrices,

$$\left\| \sum_{i=1}^n \boldsymbol{G}_{i,l}(\boldsymbol{v}_i; \boldsymbol{W}) \right\|_F^2 = \sum_{k=1}^m \|\boldsymbol{b}_{k,l-1}((\boldsymbol{W}_l)_{k,\cdot}, \boldsymbol{a}_{k,l+1})\|^2. \tag{A.68}$$

Due to (A.49), for any vector $\boldsymbol{w} \in \mathbb{R}^m$ and any integer $l \in [L]$, we note

$$1 \geq \sum_{i=1}^n 1_{\boldsymbol{w} \in \mathcal{W}_{i,l-1}}. \tag{A.69}$$

It follows from (A.68) and (A.69),

$$\begin{aligned} \left\| \sum_{i=1}^n \boldsymbol{G}_{i,l}(\boldsymbol{v}_i; \boldsymbol{W}) \right\|_F^2 &\geq \sum_{k=1}^m \|\boldsymbol{b}_{k,l-1}((\boldsymbol{W}_l)_{k,\cdot}, \boldsymbol{a}_{k,l+1})\|^2 \sum_{i=1}^n 1_{(\boldsymbol{W}_l)_{k,\cdot} \in \mathcal{W}_{i,l-1}} \\ &= \sum_{k=1}^m \sum_{i=1}^n \|\boldsymbol{b}_{k,l-1}((\boldsymbol{W}_l)_{k,\cdot}, \boldsymbol{a}_{k,l+1})\|^2 1_{(\boldsymbol{W}_l)_{k,\cdot} \in \mathcal{W}_{i,l-1}} \end{aligned}$$

By (A.58), we know that with probability at least 0.5, conditioning on $(\boldsymbol{W}_l)_{k,\cdot} \in \mathcal{W}_{i,l-1}$,

$$\|\boldsymbol{b}_{k,l-1}((\boldsymbol{W}_l)_{k,\cdot}, \boldsymbol{a}_{k,l+1})\|^2 \geq \frac{(\boldsymbol{a}_{k,l+1})_i^2}{4} \frac{(1-\alpha)^2}{1+\alpha^2} \|\boldsymbol{h}_{i,l-1}\|^2$$

We introduce the following new event $\mathcal{V}_{i,l}$ as follows

$$\mathcal{V}_{i,l} := \Big\{ (\boldsymbol{W}_l)_{k,\cdot} \in \mathcal{W}_{i,l-1},\ (\boldsymbol{a}_{k,l+1})_i^2 \geq \frac{\|\boldsymbol{v}_i\|^2}{2d},\ \|\boldsymbol{h}_{i,l-1}\| \geq \frac{1}{2} \Big\}.$$

Using this event, the observation $\mathcal{V}_{i,l} \subset \{(\boldsymbol{W}_l)_{k,\cdot} \in \mathcal{W}_{i,l-1}\}$, the definition of $\mathcal{V}_{i,l}$ and

(A.58), we obtain the following lower bound on the squared norm in (A.68):

$$\begin{aligned}\left\|\sum_{i=1}^{n}\boldsymbol{G}_{i,l}(\boldsymbol{v}_i;\boldsymbol{W})\right\|_F^2 &\geq \sum_{k=1}^{m}\sum_{i=1}^{n}\|\boldsymbol{b}_{k,l-1}((\boldsymbol{W}_l)_{k,\cdot},\boldsymbol{a}_{k,l+1})\|^2 1_{(\boldsymbol{W}_l)_{k,\cdot}\in\mathcal{W}_{i,l-1}}\\ &\geq \sum_{k=1}^{m}\sum_{i=1}^{n}\|\boldsymbol{b}_{k,l-1}((\boldsymbol{W}_l)_{k,\cdot},\boldsymbol{a}_{k,l+1})\|^2 1_{\mathcal{V}_{i,l}}\\ &\geq \sum_{k=1}^{m}\sum_{i=1}^{n}\frac{\|\boldsymbol{v}_i\|^2}{32d}\frac{(1-\alpha)^2}{1+\alpha^2}\mathbb{P}(\mathcal{V}_{i,l}).\end{aligned}$$

For simplicity, we denote

$$Z_k := \sum_{i=1}^{n}\frac{\|\boldsymbol{v}_i\|^2}{32d}\frac{(1-\alpha)^2}{1+\alpha^2}1_{\mathcal{V}_{i,l}}.$$

To lower bound the probability $\mathbb{P}(\mathcal{V}_{i,l})$, we note that $\boldsymbol{W}_l$, $\boldsymbol{a}_{k,l+1}$ and $\boldsymbol{h}_{i,l-1}$ are independent because they depend on $\boldsymbol{W}_{l+k}$ for $k\in[L-l+1]$, $\boldsymbol{W}_l$ and $\boldsymbol{W}_{l-k}$ for $k\in[l]$. We note that $(\boldsymbol{a}_{k,l+1})_i$ is corresponding to $a_{k,l+1}$ with selecting $\boldsymbol{v}=\boldsymbol{v}_i$ in the statement proven in the previous part (A.59). Then by using (A.52), (A.59) and applying Lemma A.1

$$\begin{aligned}\mathbb{P}(\mathcal{V}_{i,l}) &= \mathbb{P}\left((\boldsymbol{W}_l)_{k,\cdot}\in\mathcal{W}_{i,l-1}\right)\mathbb{P}\left((\boldsymbol{a}_{k,l+1})_i^2\geq\frac{\|\boldsymbol{v}_i\|^2}{2d}\right)\mathbb{P}\left(\|\boldsymbol{h}_{i,l-1}\|\geq\frac{1}{2}\right)\\ &\geq \Omega\left(\frac{\delta}{nL}\right)\times(1-\exp(-\Omega(1)))\times\left(1-e^{-\Omega(m/L)}\right)=\Omega\left(\frac{\delta}{nL}\right).\end{aligned}$$

By property of indicator function, we note that

$$\mathbb{E}Z_k = \sum_{i=1}^{n}\frac{\|\boldsymbol{e}_i\|^2}{32d}\frac{(1-\alpha)^2}{1+\alpha^2}\mathbb{P}(\mathcal{V}_{i,l})$$

and

$$\operatorname{Var}Z_k = \frac{\|\boldsymbol{e}_i\|^2}{32d}\frac{(1-\alpha)^2}{1+\alpha^2}1_{\mathcal{V}_{i,l}}\mathbb{P}(\mathcal{V}_{i,l})(1-\mathbb{P}(\mathcal{V}_{i,l}).$$

Then, by using Hoeffding inequality, with probability at least $1 - e^{-\Omega(m\delta^2/L^2)}$ that

$$\begin{aligned}\sum_{k=1}^{m} Z_k &\geq \frac{m}{2}\sum_{i=1}^{n}\frac{\|\boldsymbol{e}_i\|^2}{32d}\frac{(1-\alpha)^2}{1+\alpha^2}\mathbb{P}(\mathcal{V}_i)\\ &\geq \frac{Cm}{2}\sum_{i=1}^{n}\frac{\|\boldsymbol{v}_i\|^2}{32d}\frac{(1-\alpha)^2}{1+\alpha^2}\Omega\left(\frac{\delta}{nL}\right).\end{aligned}$$

Thus we conclude the Lemma, for all $l \in [L]$, as follows:

$$\left\|\sum_{i=1}^{n}\boldsymbol{G}_{i,l}(\boldsymbol{v}_i;\boldsymbol{W}^{(0)})\right\|_F^2 \geq \sum_{k=1}^{m} Z_k \geq \Omega\left(\frac{(1-\alpha)^2}{(1+\alpha^2)}\frac{\delta m}{ndL}\right)\sum_{i=1}^{n}\|\boldsymbol{v}_i\|^2.$$

□

At last, we conclude the proof of Lemma 2.2.

Proof: [Proof of Lemma 2.2] In order to prove the lower and upper bounds for the gradient for parameters $\boldsymbol{W}$ close to $\boldsymbol{W}^{(0)}$, we need leverage Lemma A.5 to show that after perturbation from $\boldsymbol{W}^{(0)}$, the change in gradient has a smaller order than the upper bound in Lemma A.6 and the lower bound in Lemma A.7. Then the same upper and lower bounds hold for $\boldsymbol{W}$ such that $\|\boldsymbol{W}^{(0)} - \boldsymbol{W}\| < \omega$ and thus conclude Lemma 2.2.

We denote a perturbation of the function $\boldsymbol{G}_{i,l}(\boldsymbol{v};\boldsymbol{W}^{(0)})$ with respect to $\boldsymbol{W}^{(0)}$,

$$
\begin{aligned}
&\boldsymbol{G}_{i,l}(\boldsymbol{v};\boldsymbol{W}) - \boldsymbol{G}_{i,l}(\boldsymbol{v};\boldsymbol{W}^{(0)}) \\
&= (\boldsymbol{v}^T\boldsymbol{B}\boldsymbol{D}_{i,L}\boldsymbol{W}_L...\boldsymbol{D}_{i,l+1}\boldsymbol{W}_{l+1}\boldsymbol{D}_{i,l})^T\boldsymbol{h}_{i,l-1}^T \\
&\quad - (\boldsymbol{v}^T\boldsymbol{B}\boldsymbol{D}_{i,L}^{(0)}\boldsymbol{W}_L^{(0)}...\boldsymbol{D}_{i,l+1}^{(0)}\boldsymbol{W}_{l+1}^{(0)}\boldsymbol{D}_{i,l}^{(0)})^T\boldsymbol{h}_{i,l-1}^{(0)T} \\
&= (\boldsymbol{v}^T\boldsymbol{B}\boldsymbol{D}_{i,L}\boldsymbol{W}_L...\boldsymbol{D}_{i,l+1}\boldsymbol{W}_{l+1}\boldsymbol{D}_{i,l})^T\boldsymbol{h}_{li,-1}^T \\
&\quad - (\boldsymbol{v}^T\boldsymbol{B}\boldsymbol{D}_{i,L}^{(0)}\boldsymbol{W}_L^{(0)}...\boldsymbol{D}_{i,l+1}^{(0)}\boldsymbol{W}_{l+1}^{(0)}\boldsymbol{D}_{i,l}^{(0)})^T\boldsymbol{h}_{i,l-1}^T \\
&\quad + (\boldsymbol{v}^T\boldsymbol{B}\boldsymbol{D}_{i,L}^{(0)}\boldsymbol{W}_L^{(0)}...\boldsymbol{D}_{i,l+1}^{(0)}\boldsymbol{W}_{l+1}^{(0)}\boldsymbol{D}_{i,l}^{(0)})^T\boldsymbol{h}_{i,l-1}^T \\
&\quad - (\boldsymbol{v}^T\boldsymbol{B}\boldsymbol{D}_{i,L}^{(0)}\boldsymbol{W}_L^{(0)}...\boldsymbol{D}_{i,l+1}^{(0)}\boldsymbol{W}_{l+1}^{(0)}\boldsymbol{D}_{i,l}^{(0)})^T\boldsymbol{h}_{i,l-1}^{(0)T}
\end{aligned}
$$

Using $\|\boldsymbol{u}\boldsymbol{v}^T\|_F \leq \|\boldsymbol{u}\|\|\boldsymbol{v}\|$, and denoting vectors $\boldsymbol{v}_i \in \mathbb{R}^d$, we derive the bound for the change of the gradient by

$$
\begin{aligned}
&\|\sum_{i=1}^n \boldsymbol{G}_{i,l}(\boldsymbol{v}_i;\boldsymbol{W}) - \boldsymbol{G}_{i,l}(\boldsymbol{v}_i;\boldsymbol{W}^{(0)})\|_F \\
&\leq \sum_{i=1}^n \|\boldsymbol{v}_i^T(\boldsymbol{B}\boldsymbol{D}_L\boldsymbol{W}_L...\boldsymbol{D}_{l+1}\boldsymbol{W}_{l+1}\boldsymbol{D}_l - \boldsymbol{B}\boldsymbol{D}_L^{(0)}\boldsymbol{W}_L^{(0)}...\boldsymbol{D}_{l+1}^{(0)}\boldsymbol{W}_{l+1}^{(0)})\boldsymbol{D}_l^{(0)}\|\|\boldsymbol{h}_{l-1}\| \\
&\quad + \|\boldsymbol{v}_i^T\boldsymbol{B}\boldsymbol{D}_L^{(0)}\boldsymbol{W}_L^{(0)}...\boldsymbol{D}_{l+1}^{(0)}\boldsymbol{W}_{l+1}^{(0)}\boldsymbol{D}_l^{(0)}\|\|\boldsymbol{h}_{l-1} - \boldsymbol{h}_{l-1}^{(0)}\|. \qquad \text{(A.70)}
\end{aligned}
$$

By Lemma A.5,

$$
\begin{aligned}
&\|(\boldsymbol{v}^T\boldsymbol{B}\boldsymbol{D}_L\boldsymbol{W}_L\cdots\boldsymbol{D}_{l+1}\boldsymbol{W}_{l+1}) - (\boldsymbol{v}^T\boldsymbol{B}\boldsymbol{D}_L^{(0)}\boldsymbol{W}_L^{(0)}\cdots\boldsymbol{D}_{l+1}^{(0)}\boldsymbol{W}_{l+1}^{(0)})\| \\
&\leq O\left(\omega^{1/3}L^2\frac{\sqrt{m\ln m}}{\sqrt{d}}\right)\|\boldsymbol{v}\| \text{ with probability at least } 1-e^{-\Omega(m/L)}. \qquad \text{(A.71)}
\end{aligned}
$$

By Lemma A.1,

$$
\|\boldsymbol{h}^{(0)}\| \leq 1.1 \text{ with probability at least } 1-e^{-\Omega(m/L)}. \qquad \text{(A.72)}
$$

By Lemma A.4,

$$\|\boldsymbol{h}_{l-1} - \boldsymbol{h}_{l-1}^{(0)}\| \leq O(\omega L^{5/2}\sqrt{\ln m}) \text{ with probability at least } 1 - e^{-\Omega(m/L)}. \tag{A.73}$$

We note that the combination of (A.72), (A.73) and the bound $\omega < O\left(\frac{1}{L^{5/2}\sqrt{\ln m}}\right)$ (which is a weaker bound than the one stated in the lemma) imply

$$\|\boldsymbol{h}\| \leq O(1) \text{ with probability at least } 1 - e^{-\Omega(m/L)}. \tag{A.74}$$

By applying (A.71), (A.72), (A.73) and (A.74) to (A.70), we conclude that with probability at least $1 - e^{-\Omega(m/L)}$

$$\left\|\sum_{i=1}^{n} \boldsymbol{G}_{i,l}(\boldsymbol{v}_i; \boldsymbol{W}) - \boldsymbol{G}_{i,l}(\boldsymbol{v}_i; \boldsymbol{W}^{(0)})\right\|_F^2 \leq O\left(\omega^{2/3} L^4 \frac{m \ln m}{d}\right) \sum_{i=1}^{n} \|\boldsymbol{v}_i\|^2. \tag{A.75}$$

We note that for $l \in [L]$, $i \in [n]$,

$$\nabla_{\boldsymbol{W}_l} \text{loss}(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{W}) = \boldsymbol{G}_{i,l}(\boldsymbol{e}_i; \boldsymbol{W}).$$

and thus

$$\nabla_{\boldsymbol{W}_l} \mathcal{L}(\boldsymbol{W}) = \sum_{i=1}^{n} \boldsymbol{G}_{i,l}(\boldsymbol{e}_i; \boldsymbol{W}). \tag{A.76}$$

Therefore, substituting $\boldsymbol{v}_i = \boldsymbol{e}_i$ in (A.75), the left-hand side of (A.75) becomes the perturbation of the gradient of the loss function. Since $\omega < O\left(\frac{\delta^{3/2}}{n^{3/2} L^{15/2} \ln^{3/2} m}\right)$ and $\delta < c_0$,

$$\omega^{2/3} L^4 \frac{m \ln m}{d} < O\left(\frac{\delta m}{ndL}\right) < O\left(\frac{mn}{d}\right). \tag{A.77}$$

For the upper bound, by Lemma A.6, (A.76), (A.75) and then by (A.77), with probability at least $1 - e^{-\Omega(m/L)}$,

$$\begin{aligned}
\|\nabla_{\boldsymbol{W}_l}\mathcal{L}(\boldsymbol{W})\|_F^2 &= \left\|\sum_{i=1}^n \boldsymbol{G}_{i,l}(\boldsymbol{e}_i;\boldsymbol{W})\right\|_F^2 \\
&\le 2\left\|\sum_{i=1}^n \boldsymbol{G}_{i,l}(\boldsymbol{e}_i;\boldsymbol{W}^{(0)})\right\|_F^2 + 2\left\|\sum_{i=1}^n \boldsymbol{G}_{i,l}(\boldsymbol{e}_i;\boldsymbol{W}^{(0)}) - \boldsymbol{G}_{i,l}(\boldsymbol{e}_i;\boldsymbol{W})\right\|_F^2 \\
&\le \left(O\left(\frac{mn}{d}\right) + O\left(\omega^{2/3}L^4\frac{m\ln m}{d}\right)\right)\sum_{i=1}^n \|\boldsymbol{e}_i\|^2 \\
&\le O\left(\frac{mn}{d}\right)\sum_{i=1}^n \|\boldsymbol{e}_i\|^2 \\
&= O\left(\frac{mn}{d}\right)\mathcal{L}(\boldsymbol{W}).
\end{aligned}$$

By definition, we further conclude that

$$\|\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W})\|^2 \le \max_{l\in[L]}\|\nabla_{\boldsymbol{W}_l}\mathcal{L}(\boldsymbol{W})\|_F^2 \le O\left(\frac{mn}{d}\right)\mathcal{L}(\boldsymbol{W}).$$

For the lower bound, by Lemma A.7, (A.76), (A.75) and then by (A.77), with probability at least $1 - e^{-\Omega(m\delta^2)}$,

$$\begin{aligned}
\|\nabla_{\boldsymbol{W}_l}\mathcal{L}(\boldsymbol{W})\|_F^2 &= \left\|\sum_{i=1}^n \boldsymbol{G}_{i,l}(\boldsymbol{e}_i;\boldsymbol{W})\right\|_F^2 \\
&\ge \frac{1}{2}\left\|\sum_{i=1}^n \boldsymbol{G}_{i,l}(\boldsymbol{e}_i;\boldsymbol{W}^{(0)})\right\|_F^2 - \left\|\sum_{i=1}^n \boldsymbol{G}_{i,l}(\boldsymbol{e}_i;\boldsymbol{W}^{(0)}) - \boldsymbol{G}_{i,l}(\boldsymbol{e}_i;\boldsymbol{W})\right\|_F^2 \\
&\ge \Omega\left(\frac{(1-\alpha)^2}{1+\alpha^2}\frac{\delta m}{ndL} - O\left(\omega^{2/3}L^4\frac{m\ln m}{d}\right)\right)\sum_{i=1}^n \|\boldsymbol{e}_i\|^2 \\
&\ge \Omega\left(\frac{(1-\alpha)^2}{1+\alpha^2}\frac{\delta m}{ndL}\right)\mathcal{L}(\boldsymbol{W}).
\end{aligned}$$

By definition, we conclude that

$$\|\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W})\|_F^2 = \sum_{l\in[L]} \|\nabla_{\boldsymbol{W}_l}\mathcal{L}(\boldsymbol{W})\|_F^2 \geq \Omega\left(\frac{(1-\alpha)^2}{1+\alpha^2}\frac{\delta m}{nd}\right)\mathcal{L}(\boldsymbol{W}). \tag{A.78}$$

□

A.2.5 Proof of Lemma 2.1

We prove Lemma 2.1 by adapting the arguments of the proof of Theorem 4 in Allen-Zhu et al. (2019b) to Leaky ReLUs.

Let us first introduce some notation. We let $\boldsymbol{W}^*$ be a vector of matrices satisfying $\|\boldsymbol{W}^* - \boldsymbol{W}^{(0)}\| < \omega$, where we think of $\boldsymbol{W}^*$ as a vector of matrices at an arbitrary training step (we will apply the lemma in this way). We denote a perturbation of $\boldsymbol{W}^*$ by $\boldsymbol{W}'$ and the perturbed matrix by $\boldsymbol{W} := \boldsymbol{W}^* + \boldsymbol{W}'$. Additional notation corresponding to the original, perturbation and perturbed settings (of $\boldsymbol{W}^*$, $\boldsymbol{W}$ and $\boldsymbol{W}'$, respectively) is summarized as follows:

$$\begin{aligned}
\boldsymbol{g}^*_{i,l} &= \boldsymbol{W}^*_l \boldsymbol{h}^*_{i,l-1} & \boldsymbol{g}_{i,l} &= \boldsymbol{W}_l \boldsymbol{h}_{i,l-1} \\
(\boldsymbol{D}_{i,l})^*_{jj} &= \frac{1_{(\boldsymbol{g}^*_{i,l})_j \geq 0} + \alpha 1_{(\boldsymbol{g}^*_{i,l})_j < 0}}{\sqrt{1+\alpha^2}} & (\boldsymbol{D}_{i,l})_{jj} &= \frac{1_{(\boldsymbol{g}_{i,l})_j \geq 0} + \alpha 1_{(\boldsymbol{g}_{i,l})_j < 0}}{\sqrt{1+\alpha^2}} \\
\boldsymbol{h}^*_{i,l} &= \tilde{\sigma}_\alpha(\boldsymbol{W}^*_l \boldsymbol{h}^*_{l-1}) \equiv \tilde{\sigma}_\alpha(\boldsymbol{g}^*_{i,l}) & \boldsymbol{h}_{i,l} &= \tilde{\sigma}_\alpha(\boldsymbol{W}_l \boldsymbol{h}_{l-1}) \equiv \tilde{\sigma}_\alpha(\boldsymbol{g}_{i,l}) \\
\boldsymbol{e}^*_{i,l} &= \boldsymbol{y}_i - \boldsymbol{B}\boldsymbol{h}^*_{i,L} & \boldsymbol{e}_{i,l} &= \boldsymbol{y}_i - \boldsymbol{B}\boldsymbol{h}^*_{i,L}.
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{g}'_{i,l} &= \boldsymbol{g}_{i,l} - \boldsymbol{g}^*_{i,l} & \boldsymbol{D}'_{i,l} &= \boldsymbol{D}_{i,l} - \boldsymbol{D}^*_{i,l} \\
\boldsymbol{h}'_{i,l} &= \boldsymbol{h}_{i,l} - \boldsymbol{h}^*_{i,l} & \boldsymbol{e}'_{i,l} &= \boldsymbol{e}_{i,l} - \boldsymbol{e}^*_{i,l}
\end{aligned}$$

The loss functions at $\boldsymbol{W}^*$ and $\boldsymbol{W}$ are expressed as

$$\mathcal{L}(\boldsymbol{W}^*) = \frac{1}{2}\sum_{i=1}^{n} \|\boldsymbol{e}_i^*\|^2,\ \mathcal{L}(\boldsymbol{W}) = \frac{1}{2}\sum_{i=1}^{n} \|\boldsymbol{e}_i\|^2. \tag{A.79}$$

We introduce an auxiliary lemma before proving Lemma 2.1.

Lemma A.8

There exists a set of diagonal matrices $\boldsymbol{D}''_{i,l} \in [-\sqrt{2}, \sqrt{2}]^{m\times m}$ so that

$$\boldsymbol{h}'_{i,l} = \boldsymbol{h}_{i,l} - \boldsymbol{h}^*_{i,l} = \sum_{a=1}^{l} (\boldsymbol{D}^*_{i,l} + \boldsymbol{D}''_{i,l})\boldsymbol{W}^*_l(\boldsymbol{D}^*_{i,l-1} + \boldsymbol{D}''_{i,l-1})...\boldsymbol{W}^*_{a+1}(\boldsymbol{D}^*_{i,a} + \boldsymbol{D}''_{i,a})\boldsymbol{W}'_a\boldsymbol{h}_{i,a-1}.$$

Furthermore, the following bounds hold

$$\|\boldsymbol{h}'_{i,l}\| \le O(L^{3/2})\|\boldsymbol{W}'\|, \quad \|\boldsymbol{B}\boldsymbol{h}'_{i,L}\| \le O(L\sqrt{m/d})\|\boldsymbol{W}'\| \text{ and } \|\boldsymbol{D}''_{i,l}\|_0 \le O(m\omega^{2/3}L).$$

The proof of this lemma is identical to the proof of Claim 11.2 in Allen-Zhu et al. (2019b). It is obtained by replacing $|D''_{k,k}| \le 1$ in the second statement of Proposition 11.3 in Allen-Zhu et al. (2019b) with $|D''_{k,k}| \le \sqrt{2}$ in order to fit the setting of Leaky ReLUs.

The rest of this section provides a detailed proof of Lemma 2.1.

Proof: [Proof of Lemma 2.1] We first express the loss function at $\boldsymbol{W}$ as follows

$$
\begin{aligned}
&\text{loss}(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{W}) \\
&= \frac{1}{2}\|\boldsymbol{B}\boldsymbol{h}_{i,L} - \boldsymbol{y}_i\|^2 = \frac{1}{2}\|\boldsymbol{B}(\boldsymbol{h}_{i,L} - \boldsymbol{h}^*_{i,L}) + \boldsymbol{B}\boldsymbol{h}^*_{i,L} - \boldsymbol{y}_i\|^2 \\
&= \frac{1}{2}\|\boldsymbol{e}^*_i + \boldsymbol{B}(\boldsymbol{h}_{i,L} - \boldsymbol{h}^*_{i,L})\|^2 \\
&= \frac{1}{2}\|\boldsymbol{e}^*_i\|^2 + \frac{1}{2}\|\boldsymbol{B}(\boldsymbol{h}_{i,L} - \boldsymbol{h}^*_{i,L})\|^2 + \langle \boldsymbol{e}^*_i, \boldsymbol{B}(\boldsymbol{h}_{i,L} - \boldsymbol{h}^*_{i,L})\rangle \\
&= \text{loss}^*_i + \frac{1}{2}\|\boldsymbol{B}(\boldsymbol{h}_{i,L} - \boldsymbol{h}^*_{i,L})\|^2 + \boldsymbol{e}^{*T}_i \boldsymbol{B}(\boldsymbol{h}_{i,L} - \boldsymbol{h}^*_{i,L}) \\
&= \text{loss}^*_i + \frac{1}{2}\|\boldsymbol{B}(\boldsymbol{h}_{i,L} - \boldsymbol{h}^*_{i,L})\|^2 + \boldsymbol{e}^{*T}_i \boldsymbol{B}\boldsymbol{h}'_{i,L}.
\end{aligned} \tag{A.80}
$$

Then we expand $\langle \nabla \mathcal{L}(\boldsymbol{W}^*), \boldsymbol{W}' \rangle$ as

$$
\begin{aligned}
&\langle \nabla \mathcal{L}(\boldsymbol{W}^*), \boldsymbol{W}' \rangle \\
&= \sum_{l=1}^{L} \langle \nabla_{\boldsymbol{W}_l} \mathcal{L}(\boldsymbol{W}^*), \boldsymbol{W}'_l \rangle = \sum_{l=1}^{L}\sum_{i=1}^{n} \langle \boldsymbol{D}^*_{i,l}\boldsymbol{W}^{*T}_{l+1}\boldsymbol{D}^*_{i,l+1}...\boldsymbol{D}^*_{i,L}\boldsymbol{B}^T\boldsymbol{e}^*_i\boldsymbol{h}^{*T}_{l-1}(\boldsymbol{x}_i), \boldsymbol{W}'_l \rangle \\
&= \sum_{l=1}^{L}\sum_{i=1}^{n} \langle \boldsymbol{D}^*_{i,l}\boldsymbol{W}^{*T}_{l+1}\boldsymbol{D}^*_{i,l+1}...\boldsymbol{D}^*_{i,L}\boldsymbol{B}^T\boldsymbol{e}^*_i\boldsymbol{h}^{*T}_{l-1}(\boldsymbol{x}_i), \boldsymbol{W}'_l \rangle \\
&= \sum_{l=1}^{L}\sum_{i=1}^{n} \boldsymbol{e}^{*T}_i\boldsymbol{B}\boldsymbol{D}^*_{i,L}\boldsymbol{W}^*_L...\boldsymbol{D}^*_{i,l+1}\boldsymbol{W}^*_{l+1}\boldsymbol{D}^*_{i,l}\boldsymbol{W}'_l\boldsymbol{h}^*_{l-1}(\boldsymbol{x}_i).
\end{aligned}
$$

The above two equations imply the following estimate

$$
\begin{aligned}
&\mathcal{L}(\boldsymbol{W}^* + \boldsymbol{W}') - \mathcal{L}(\boldsymbol{W}^*) - \langle \nabla \mathcal{L}(\boldsymbol{W}^*), \boldsymbol{W}' \rangle \\
&= -\langle \nabla \mathcal{L}(\boldsymbol{W}^*), \boldsymbol{W}' \rangle + \sum_{i=1}^{n} (\text{loss}_i - \text{loss}_i^*) \\
&= -\sum_{l=1}^{L} \sum_{i=1}^{n} \boldsymbol{e}_i^{*T} \boldsymbol{B} \boldsymbol{D}_{i,L}^* \boldsymbol{W}_L^* ... \boldsymbol{D}_{i,l+1}^* \boldsymbol{W}_{l+1}^* \boldsymbol{D}_{i,l}^* \boldsymbol{W}_l' \boldsymbol{h}_{l-1}^*(\boldsymbol{x}_i) \\
&\quad + \sum_{i=1}^{n} \big(\frac{1}{2} \|\boldsymbol{B}(\boldsymbol{h}_{i,L} - \boldsymbol{h}_{i,L}^*)\|^2 + \boldsymbol{e}_i^{*T} \boldsymbol{B}(\boldsymbol{h}_{i,L} - \boldsymbol{h}_{i,L}^*) \big) \\
&= \sum_{i=1}^{n} e_i^{*T} \boldsymbol{B} \Big((\boldsymbol{h}_{i,L} - \boldsymbol{h}_{i,L}^*) - \sum_{l=1}^{L} \boldsymbol{D}_{i,L}^* \boldsymbol{W}_L^* ... \boldsymbol{D}_{i,l+1}^* \boldsymbol{W}_{l+1}^* \boldsymbol{D}_{i,l}^* \boldsymbol{W}_l' \boldsymbol{h}_{l-1}^*(\boldsymbol{x}_i) \Big) \qquad \text{(A.81)} \\
&\quad + \frac{1}{2} \sum_{i=1}^{n} \|\boldsymbol{B}(\boldsymbol{h}_{i,L} - \boldsymbol{h}_{i,L}^*)\|^2. \qquad \text{(A.82)}
\end{aligned}
$$

Lemma A.8 provides the following upper bound for (A.82)

$$
\frac{1}{2} \sum_{i=1}^{n} \|\boldsymbol{B}(\boldsymbol{h}_{i,L} - \boldsymbol{h}_{i,L}^*)\|^2 \leq O(nL^2 m/d) \|\boldsymbol{W}'\|^2. \qquad \text{(A.83)}
$$

We note that (A.81) can be differently expressed by using Lemma A.8 to replace $\boldsymbol{h} - \boldsymbol{h}^*$ with some diagonal matrices, $\boldsymbol{D}_{i,l}''$, and by adding and subtracting the term

$\sum_{l=1}^{L} \boldsymbol{D}_{i,L}^{*}\boldsymbol{W}_{L}^{*}\dots\boldsymbol{D}_{i,l+1}^{*}\boldsymbol{W}_{l+1}^{*}\boldsymbol{D}_{i,l}^{*}\boldsymbol{W}_{l}'\boldsymbol{h}_{i,l}$ as follows

$$
\begin{aligned}
&\boldsymbol{e}_i^{*T}\boldsymbol{B}\Big((\boldsymbol{h}_{i,L}-\boldsymbol{h}_{i,L}^{*})-\sum_{l=1}^{L}\boldsymbol{D}_{i,L}^{*}\boldsymbol{W}_{L}^{*}...\boldsymbol{D}_{i,l+1}^{*}\boldsymbol{W}_{l+1}^{*}\boldsymbol{D}_{i,l}^{*}\boldsymbol{W}_{l}'\boldsymbol{h}_{l-1}^{*}(\boldsymbol{x}_i)\Big)\\
&=\boldsymbol{e}_i^{*T}\boldsymbol{B}\Big(\sum_{l=1}^{L}(\boldsymbol{D}_{i,L}^{*}+\boldsymbol{D}_{i,L}'')\boldsymbol{W}_{L}^{*}...\boldsymbol{W}_{l+1}^{*}(\boldsymbol{D}_{i,l}^{*}+\boldsymbol{D}_{i,l}'')\boldsymbol{W}_{l}'\boldsymbol{h}_{i,l-1}\\
&\quad-\sum_{l=1}^{L}\boldsymbol{D}_{i,L}^{*}\boldsymbol{W}_{L}^{*}...\boldsymbol{D}_{i,l+1}^{*}\boldsymbol{W}_{l+1}^{*}\boldsymbol{D}_{i,l}^{*}\boldsymbol{W}_{l}'\boldsymbol{h}_{l-1}^{*}(\boldsymbol{x}_i)\Big)\\
&=\boldsymbol{e}^{*T}\boldsymbol{B}\Big(\sum_{l=1}^{L}\big((\boldsymbol{D}_{L}^{*}+\boldsymbol{D}_{L}'')\boldsymbol{W}_{L}^{*}...\boldsymbol{W}_{l+1}^{*}(\boldsymbol{D}_{l}^{*}+\boldsymbol{D}_{l}'')\boldsymbol{W}_{l}'\\
&\qquad-\boldsymbol{D}_{L}^{*}\boldsymbol{W}_{L}^{*}...\boldsymbol{W}_{l+1}^{*}\boldsymbol{D}_{l}^{*}\boldsymbol{W}_{l}'\big)\boldsymbol{h}_{l-1} &\text{(A.84)}\\
&\quad-\sum_{l=1}^{L}\boldsymbol{D}_{L}^{*}\boldsymbol{W}_{L}^{*}...\boldsymbol{D}_{l+1}^{*}\boldsymbol{W}_{l+1}^{*}\boldsymbol{D}_{l}^{*}\boldsymbol{W}_{l}'\big(\boldsymbol{h}_{l-1}-\boldsymbol{h}_{l-1}^{*}(\boldsymbol{x})\big)\Big), &\text{(A.85)}
\end{aligned}
$$

where we neglected all the subscripts i in the last two rows of the equation.

Next, we upper bound (A.84) and (A.85). In order to bound (A.84), we first use Lemma A.5 to obtain the following bound

$$
\begin{aligned}
&\|\boldsymbol{B}(\boldsymbol{D}_{i,L}^{*}+\boldsymbol{D}_{i,L}'')\boldsymbol{W}_{L}^{*}...\boldsymbol{W}_{l+1}^{*}(\boldsymbol{D}_{i,l}^{*}+\boldsymbol{D}_{i,l}'')\boldsymbol{W}_{l}'-\boldsymbol{B}\boldsymbol{D}_{i,L}^{*}\boldsymbol{W}_{L}^{*}...\boldsymbol{W}_{l+1}^{*}\boldsymbol{D}_{i,l}^{*}\boldsymbol{W}_{l}'\|\\
&\le O\left(\frac{1-\alpha}{\sqrt{1+\alpha^2}}\frac{\omega^{1/3}L^2\sqrt{m\ln m}}{\sqrt{d}}\right)\|\boldsymbol{W}_{l}'\|. &\text{(A.86)}
\end{aligned}
$$

Using (A.79), we note that $(\sum_{i=1}^{n}\|\boldsymbol{e}_i^{*}\|)^2\le n\sum_{i=1}^{n}\|\boldsymbol{e}_i^{*}\|^2=n\mathcal{L}(\boldsymbol{W}^{*})$. Combining this

fact and (A.86) yields the following bound for (A.84):

$$\sum_{i=1}^{n} \boldsymbol{e}_i^{*T} \boldsymbol{B}\Big(\sum_{l=1}^{L} \big((\boldsymbol{D}_{i,L}^* + \boldsymbol{D}_{i,L}'')\boldsymbol{W}_L^* ... \boldsymbol{W}_{l+1}^* (\boldsymbol{D}_{i,l}^* + \boldsymbol{D}_{i,l}'')\boldsymbol{W}_l' \\ - \boldsymbol{D}_{i,L}^* \boldsymbol{W}_L^* ... \boldsymbol{W}_{l+1}^* \boldsymbol{D}_{i,l}^* \boldsymbol{W}_l' \big) \boldsymbol{h}_{i,l-1} \Big) \\ \leq \sqrt{n\mathcal{L}(\boldsymbol{W}^*)} O\left(\frac{1-\alpha}{\sqrt{1+\alpha^2}} \frac{\omega^{1/3} L^2 \sqrt{m \ln m}}{\sqrt{d}} \right) \|\boldsymbol{W}'\|. \tag{A.87}$$

In order to bound (A.85), we apply Lemma A.3 and Lemma A.5 to obtain

$$\|\boldsymbol{B}\boldsymbol{D}^*{}_{i,L}\boldsymbol{W}^*{}_L \dots \boldsymbol{D}^*{}_{i,l+1}\boldsymbol{W}^*{}_{l+1}\boldsymbol{D}^*{}_{i,l}\| \\ \leq \|\boldsymbol{B}\boldsymbol{D}^0{}_{i,L}\boldsymbol{W}^0{}_L \dots \boldsymbol{D}^0{}_{i,l+1}\boldsymbol{W}^0{}_{l+1}\boldsymbol{D}^0{}_{i,l}\| \\ + \|\boldsymbol{B}\boldsymbol{D}^0{}_{i,L}\boldsymbol{W}^0{}_L \dots \boldsymbol{D}^0{}_{i,l+1}\boldsymbol{W}^0{}_{l+1}\boldsymbol{D}^0{}_{i,l} - \boldsymbol{B}\boldsymbol{D}^*{}_{i,L}\boldsymbol{W}^*{}_L \dots \boldsymbol{D}^*{}_{i,l+1}\boldsymbol{W}^*{}_{l+1}\boldsymbol{D}^*{}_{i,l}\| \\ \leq O(\sqrt{Lm/d}) + O\left(\frac{1-\alpha}{\sqrt{1+\alpha^2}} \frac{\omega^{1/3} L^2 \sqrt{m \ln m}}{\sqrt{d}} \right). \tag{A.88}$$

Lemma A.8 implies that $\|\boldsymbol{h} - \boldsymbol{h}^*\| = \|\boldsymbol{h}'\| \leq O(L^{3/2}\|\boldsymbol{W}'\|)$. Combining this observation and (A.88) results in

$$\left\| \sum_{i=1}^{n} \boldsymbol{e}_i^{*T} \sum_{l=1}^{L} \boldsymbol{B}\boldsymbol{D}_{i,L}^* \boldsymbol{W}_L^* ... \boldsymbol{D}_{i,l+1}^* \boldsymbol{W}_{l+1}^* \boldsymbol{D}_{i,l}^* \boldsymbol{W}_l' \boldsymbol{h}_{i,l-1}' \right\| \\ \leq \sum_{i=1}^{n} \|\boldsymbol{e}_i^*\| O(L^2 \sqrt{m/d}) \|\boldsymbol{W}'\|^2. \tag{A.89}$$

In order to bound $\|\boldsymbol{e}_i^*\|$, we first note that at initialization

$$\|\boldsymbol{e}_i^{(0)}\| = \|\boldsymbol{B}\boldsymbol{h}_{L,i}^{(0)} - \boldsymbol{y}_i\| \leq \|\boldsymbol{y}_i\| + \|\boldsymbol{B}\boldsymbol{h}_{L,i}^{(0)}\|,$$

where

$$\boldsymbol{B}\boldsymbol{h}_{L,i}^{(0)} \sim N\left(0, \frac{\|\boldsymbol{h}\|^2}{d} \boldsymbol{I}_d\right).$$

For this Gaussian distribution and $d < O(1)$,

$$\mathbb{P}\left(\|(\boldsymbol{B}\boldsymbol{h}_{L,i}^{(0)})\|^2 > \frac{\sqrt{m}}{\sqrt{d}}\right) \leq e^{-\Omega(\frac{m}{d})} = e^{-\Omega(m)}.$$

Therefore, with probability at least $1 - e^{-\Omega(m)}$,

$$\|\boldsymbol{e}_i^{(0)}\| \leq O\left(\frac{\sqrt{m}}{\sqrt{d}}\right).$$

For general $\boldsymbol{e}_i^*$, $\|\boldsymbol{e}_i^*\| = \left\|\boldsymbol{B}\big(\boldsymbol{h}_{i,L}^{(0)} + (\boldsymbol{h}_{i,L}^* - \boldsymbol{h}_{i,L}^{(0)})\big) - \boldsymbol{y}_i\right\|$. Lemma A.8 implies that if $\omega \leq O(1/L)$

$$\|\boldsymbol{e}_i^*\| \leq \|\boldsymbol{e}_i^{(0)}\| + \|\boldsymbol{B}(\boldsymbol{h}_{i,L}^* - \boldsymbol{h}_{i,L}^{(0)})\| \leq O\left(\frac{\sqrt{m}}{\sqrt{d}}\right). \tag{A.90}$$

The combination of (A.90) and (A.89) results in the following bound on the term specified in (A.85)

$$\left\|\sum_{i=1}^{n} \boldsymbol{e}_i^{*T} \sum_{l=1}^{L} \boldsymbol{B}\boldsymbol{D}_{i,L}^* \boldsymbol{W}_L^* \ldots \boldsymbol{D}_{i,l+1}^* \boldsymbol{W}_{l+1}^* \boldsymbol{D}_{i,l}^* \boldsymbol{W}_l' \boldsymbol{h}_{i,l-1}'\right\| \leq O\left(\frac{nL^2 m}{d}\right) \|\boldsymbol{W}'\|^2 \tag{A.91}$$

Combining the bounds in (A.83), (A.87) and (A.91) we bound the terms in (A.81) and (A.82) with the above specified probability. We thus conclude the desired result, that is, if $\boldsymbol{W}^*$ is such that $\|\boldsymbol{W}^* - \boldsymbol{W}^{(0)}\| < \omega$, then with probability at least $1 - e^{-\Omega(m)}$

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{W}^* + \boldsymbol{W}') - \mathcal{L}(\boldsymbol{W}^*) - \langle \nabla \mathcal{L}(\boldsymbol{W}^*), \boldsymbol{W}' \rangle \\
&\leq \sqrt{n\mathcal{L}(\boldsymbol{W}^*)} O\left(\frac{1-\alpha}{\sqrt{1+\alpha^2}} \frac{\omega^{1/3} L^2 \sqrt{m \ln m}}{\sqrt{d}}\right) \|\boldsymbol{W}'\| + O(nL^2 m/d) \|\boldsymbol{W}'\|^2.
\end{aligned}$$

□

A.2.6 Conclusion of the Proof of Theorem 2.1

Most of the proof of Theorem 2.1 was given in §2.4. The only part that remains unverified is to show that during training

$$\|\boldsymbol{W}^t - \boldsymbol{W}^{(0)}\| < \omega < O\left(\frac{\delta^{3/2}}{n^{3/2}L^{15/2}\ln^{3/2} m}\right).$$

For this purpose, we establish Lemma A.9 below.

Lemma A.9

Assume the setup of §2.2, where the learning rate satisfies $\eta < \frac{\delta^{3/2}d^{1/2}}{n^3L^{15/2}m^{1/2}\ln^2 m}$ and the width m of the neural network satisfies $\frac{m}{\ln^4 m} > \Omega\left(\frac{1+\alpha^2}{(1-\alpha)^2}\frac{n^5L^{15}d}{\delta^4}\right)$. Then in the training stage described by Algorithm 2

$$\|\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(0)}\| < O\left(\frac{\delta^{3/2}}{n^{3/2}L^{15/2}\ln^{3/2} m}\right) \quad \text{with probability at least} \quad 1 - e^{-\Omega(\ln m)}. \tag{A.92}$$

□

Proof: We first establish the bound

$$\mathcal{L}(\boldsymbol{W}^{(0)}) < O(n\ln^{1/2} m) \quad \text{with probability at least } 1 - e^{-\Omega(\ln m)}. \tag{A.93}$$

We note that $\boldsymbol{B}\boldsymbol{h}_{i,L} \sim N\left(0, \frac{\|\boldsymbol{h}_{i,L}\|^2}{d}\right)$ and thus $\frac{d}{\|\boldsymbol{h}_{i,L}\|^2}\|\boldsymbol{B}\boldsymbol{h}_{i,L}\|^2 \,|\, \boldsymbol{h}_{i,L} \sim \chi^2(d)$. Applying this observation and Lemma A.1 (i.e., $\|\boldsymbol{h}_i\| \in [0.5, 1.5]$, with probability at least $1 - e^{-\Omega(m/L)}$) yields

$$\mathbb{P}\left(\frac{d}{\|\boldsymbol{h}_{i,L}\|^2}\|\boldsymbol{B}\boldsymbol{h}_{i,L}\|^2 > (1+\epsilon)d\right) < e^{-\Omega(d\epsilon^2)}.$$

Choosing $\epsilon = \sqrt{\ln m}$ and applying a union bound over $i \in [n]$ (but noting that since $m > \Omega(n)$ the probability $1 - ne^{-\Omega(d \ln m)}$ is of the same order as $1 - e^{-\Omega(d \ln m)}$), we obtain the bound

$$\|\boldsymbol{B}\boldsymbol{h}_{i,L}\|^2 \leq O(\sqrt{\ln m}) \text{ with probability at least } 1 - e^{-\Omega(d \ln m)}. \tag{A.94}$$

Therefore, we conclude (A.93) as follows:

$$\mathcal{L}(\boldsymbol{W}^{(0)}) = \sum_{i=1}^{n} \|\boldsymbol{y}_i - \boldsymbol{B}\boldsymbol{h}_{i,L}\|^2 \leq n(O(1) + O(\sqrt{\ln m})) = O(n\sqrt{\ln m}).$$

Next, we prove (A.92) by induction on $t = 1, \cdots$. It is trivial that the statement holds for $t = 0$.

To prove the induction step we follow ideas that were introduced in the proof of Lemma 4.1 in Zou and Gu (2019). Using the induction assumption, we can apply (2.16) and then (2.14) (indeed, the conditions for these bounds are guaranteed by the induction assumption) and consequently obtain

$$\begin{aligned}\sqrt{\mathcal{L}(\boldsymbol{W}^{(s)})} - \sqrt{\mathcal{L}(\boldsymbol{W}^{(s+1)})} &= \frac{\mathcal{L}(\boldsymbol{W}^{(s)}) - \mathcal{L}(\boldsymbol{W}^{(s+1)})}{\sqrt{\mathcal{L}(\boldsymbol{W}^{(s)})} + \sqrt{\mathcal{L}(\boldsymbol{W}^{(s+1)})}} \geq \Omega(1)\frac{\eta\|\nabla\mathcal{L}(\boldsymbol{W}^{(s)})\|_F^2}{\sqrt{\mathcal{L}(\boldsymbol{W}^{(s)})}} \\ &\geq \frac{(1-\alpha)}{\sqrt{1+\alpha^2}}\Omega\left(\sqrt{\frac{\delta m}{nd}}\right)\eta\|\nabla\mathcal{L}(\boldsymbol{W}^{(s)})\|_F,\end{aligned}$$

or equivalently,

$$\eta\|\nabla\mathcal{L}(\boldsymbol{W}^{(s)})\|_F \leq \frac{\sqrt{1+\alpha^2}}{(1-\alpha)}\Omega\left(\sqrt{\frac{nd}{\delta m}}\right)\left(\sqrt{\mathcal{L}(\boldsymbol{W}^{(s)})} - \sqrt{\mathcal{L}(\boldsymbol{W}^{(s+1)})}\right). \tag{A.95}$$

Combining the training procedure with (A.95) yields

$$\begin{aligned}\|\boldsymbol{W}^{(t)}-\boldsymbol{W}^{(0)}\| &\le \eta\sum_{s=0}^{t-1}\|\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W}^{(s)})\| \\ &\le \frac{\sqrt{1+\alpha^2}}{(1-\alpha)}\Omega\left(\sqrt{\frac{nd}{\delta m}}\right)\left(\sqrt{\mathcal{L}(\boldsymbol{W}^{(0)})}-\sqrt{\mathcal{L}(\boldsymbol{W}^{(t)})}\right) \\ &\le \frac{\sqrt{1+\alpha^2}}{(1-\alpha)}\Omega\left(\sqrt{\frac{nd}{\delta m}}\right)\sqrt{\mathcal{L}(\boldsymbol{W}^{(0)})}. \end{aligned} \tag{A.96}$$

Applying (A.93) to the bound above we conclude that when $\frac{m}{\ln^4 m} > \frac{1+\alpha^2}{(1-\alpha)^2}\Omega(n^5L^{15}d/\delta^4)$,

$$\|\boldsymbol{W}^{(t)}-\boldsymbol{W}^{(0)}\| \le O\left(\frac{\delta^{3/2}}{n^{3/2}L^{15/2}\ln^{3/2}m}\right) \text{ with probability at least } 1-e^{-\Omega(\ln m)}.$$

□

A.2.7 Proof of Theorem 2.2

Throughout this proof we assume that $\|\boldsymbol{W}^{(t)}-\boldsymbol{W}^{(0)}\| \le O(\frac{\delta^{3/2}}{n^{3/2}L^{15/2}\ln^{3/2}m})$ during training, which is a sufficient condition for some of the propositions used, such as for Lemma 2.1. After finalizing the proof under this assumption, we establish Lemma A.10 that guarantees this assumption.

Applying Lemma 2.1 and taking expectations yield

$$\begin{aligned}\mathbb{E}\mathcal{L}(\boldsymbol{W}^{(t+1)}) &= \mathbb{E}\mathcal{L}(\boldsymbol{W}^{(t)}-\eta\nabla_{\boldsymbol{W}}\mathcal{L}_B(\boldsymbol{W}^{(t)})) \\ &\le \mathbb{E}\mathcal{L}(\boldsymbol{W}^{(t)}) - \mathbb{E}\eta\langle\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W}^{(t)}),\nabla_{\boldsymbol{W}}\mathcal{L}_B(\boldsymbol{W}^{(t)})\rangle \\ &\quad+\frac{\eta(1-\alpha)\omega^{\frac{1}{3}}L^2\sqrt{mn\mathcal{L}(\boldsymbol{W}^{(t)})\ln m}}{\sqrt{d(1+\alpha^2)}}\mathbb{E}O\left(\|\nabla_{\boldsymbol{W}}\mathcal{L}_B(\boldsymbol{W}^{(t)})\|\right) \\ &\quad+\frac{\eta^2nL^2m}{d}\mathbb{E}O\left(\|\nabla_{\boldsymbol{W}}\mathcal{L}_B(\boldsymbol{W}^{(t)})\|^2\right). \end{aligned} \tag{A.97}$$

Applying the following basic observations:

$$\mathbb{E}\langle \nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}^{(t)}), \nabla_{\boldsymbol{W}} \mathcal{L}_B(\boldsymbol{W}^{(t)})\rangle = \frac{b}{n}\|\nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}^{(t)})\|_F^2,$$

$$\|\nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}^{(t)})\| = \max_{l\in[L]} \|\nabla_{\boldsymbol{W}_l} \mathcal{L}(\boldsymbol{W}^{(t)})\| \leq \max_{l\in[L]} \|\nabla_{\boldsymbol{W}_l} \mathcal{L}(\boldsymbol{W}^{(t)})\|_F \leq \|\nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}^{(t)})\|_F,$$

while selecting $\omega < \frac{\delta^{3/2}}{n^3 L^6 \ln^{3/2} m}$ and $\eta < \frac{d}{bL^2 m}$, to (A.97) results in

$$\mathbb{E}\mathcal{L}(\boldsymbol{W}^{(t+1)}) \leq \mathcal{L}(\boldsymbol{W}^{(t)}) - \frac{\eta b}{n}\|\nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}^{(t)})\|_F^2 \leq \left(1 - \Omega\left(\frac{(1-\alpha)^2}{1+\alpha^2}\frac{\eta\delta m b}{n^2 d}\right)\right)\mathcal{L}(\boldsymbol{W}^{(t)}). \tag{A.98}$$

For simplicity, we define

$$\gamma := \left(1 - \Omega\left(\frac{(1-\alpha)^2}{1+\alpha^2}\frac{\eta\delta m b}{n^2 d}\right)\right),$$

and (A.98) becomes

$$\mathbb{E}\mathcal{L}(\boldsymbol{W}^{(t+1)}) \leq \gamma \mathcal{L}(\boldsymbol{W}^{(t)}). \tag{A.99}$$

Next, we establish a bound for $\mathcal{L}(\boldsymbol{W}^{(t+1)})$ without expectation. We note that (2.9) implies

$$\|\nabla_{\boldsymbol{W}_l} \mathcal{L}_B(\boldsymbol{W}^{(t)})\|_F^2 \leq (bm/d)\mathcal{L}(\boldsymbol{W}^{(t)}),$$

and consequently

$$\|\nabla_{\boldsymbol{W}} \mathcal{L}_B(\boldsymbol{W}^{(t)})\|_F^2 \leq \frac{bmL}{d}\mathcal{L}(\boldsymbol{W}^{(t)}) \text{ and } \|\nabla_{\boldsymbol{W}} \mathcal{L}_B(\boldsymbol{W}^{(t)})\|^2 \leq \frac{bm}{d}\mathcal{L}(\boldsymbol{W}^{(t)}). \tag{A.100}$$

The application of Lemma 2.1, (A.100) and our choice of η results in

$$\begin{aligned}\mathcal{L}(\boldsymbol{W}^{(t+1)}) &\leq \mathcal{L}(\boldsymbol{W}^{(t)}) + \eta\|\nabla_{\boldsymbol{W}}\mathcal{L}_B(\boldsymbol{W}^{(t)})\|_F\|\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W}^{(t)})\|_F + \eta\frac{b^2mn}{d}\mathcal{L}(\boldsymbol{W}^{(t)}) \\ &\leq \left(1 + O\left(\frac{\eta mL\sqrt{nb}}{d}\right)\right)\mathcal{L}(\boldsymbol{W}^{(t)}). \end{aligned} \tag{A.101}$$

For simplicity, we define $\beta := 1 + O(\eta mL\sqrt{nb}/d)$, and (A.101) becomes

$$\mathcal{L}(\boldsymbol{W}^{(t+1)}) \leq \beta\mathcal{L}(\boldsymbol{W}^{(t)}). \tag{A.102}$$

We denote

$$\mathcal{L}^t := \mathcal{L}(\boldsymbol{W}^{(t)})$$

and define the filtration

$$\mathcal{F}_t := \sigma(\boldsymbol{W}^{(0)}, ..\boldsymbol{W}^{(t)}).$$

We further define

$$Y_t := \ln\mathcal{L}^t - \ln\mathcal{L}^{t-1} - \mathbb{E}(\ln\mathcal{L}^t - \ln\mathcal{L}^{t-1}|\mathcal{F}_{t-1})$$

and

$$X_t := \sum_{s=1}^{t} Y_s.$$

We note that $\{X_t\}$ is a martingale.

We will use Azuma's inequality to bound X_t. We thus need to show that $\{X_t\}$ is $\boldsymbol{c}-$Lipschitz (i.e., $|Y_t| \leq c_t$,). We verify the $\boldsymbol{c}-$Lipschitz property by applying the

definition of Y_t, (A.102) and (A.99) as follows:

$$\begin{aligned}|Y_{t+1}| &= |\ln \mathcal{L}^{(t+1)} - \ln \mathcal{L}^{(t)} - \mathbb{E} \ln \mathcal{L}^{(t+1)} - \ln \mathcal{L}^{(t)} | \mathcal{F}_t| \\ &\leq \ln \beta - \ln \gamma = \ln \frac{\beta}{\gamma}.\end{aligned}$$

Then by Azuma's inequality,

$$\mathbb{P}\left(|X_t - EX_t| \geq \lambda\right) \leq 2 \exp\left(-\frac{\lambda^2}{2t \ln^2 \beta/\gamma}\right). \tag{A.103}$$

Choosing $\lambda = \sqrt{t} \ln(\beta/\gamma) \ln m$ in (A.103) yields

$$|X_t| \leq \sqrt{t} \ln(\beta/\gamma) \ln m \ \text{ with probability at least } 1 - e^{-\Omega(\ln^2 m)}.. \tag{A.104}$$

Applying the definition of Y_t and (A.99) results in

$$\ln \mathcal{L}^t = X_t + \ln \mathcal{L}^{(0)} + \sum_{s=1}^{t} \mathbb{E}(Y_s - Y_{s-1} | \mathcal{F}_{s-1}) \leq X_t + \ln \mathcal{L}^{(0)} + t \ln \gamma.$$

We further apply the above observation and (A.104) to conclude that with probability at least $1 - e^{-\Omega(\ln^2 m)}$

$$\begin{aligned}\ln \mathcal{L}^{(t)} &\leq \ln \mathcal{L}^{(0)} + t \ln \gamma + \sqrt{t} \ln\left(\frac{\beta}{\gamma}\right) \ln m \\ &\leq \ln \mathcal{L}^{(0)} + \frac{\ln^2\left(\frac{\beta}{\gamma}\right) \ln^2 m}{4|\ln \gamma|} - \left(\sqrt{|\ln \gamma|}\sqrt{t} - \frac{\ln \frac{\beta}{\gamma} \ln m}{2\sqrt{|\ln \gamma|}}\right)^2.\end{aligned}$$

We note that for $f(x) = (ax+b)^2$ and $x > 4b/a$, $f(x) \geq \frac{1}{2} a^2 x^2$. Using this fact, we

conclude that when $\sqrt{t} > \frac{2\ln\frac{\beta}{\gamma}\ln m}{|\ln\gamma|}$, or equivalently, when $t > \frac{4\ln^2\frac{\beta}{\gamma}\ln^2 m}{\ln^2\gamma}$,

$$\ln\mathcal{L}^{(t)} \leq \ln\mathcal{L}^{(0)} + \frac{\ln^2\frac{\beta}{\gamma}\ln^2 m}{4|\ln\gamma|} + t \times 1_{\left\{t > \frac{4\ln^2\frac{\beta}{\gamma}\ln^2 m}{\ln^2\gamma}\right\}} \ln\gamma$$
$$\text{with probability } \geq 1 - e^{-\Omega(\ln^2 m)}. \tag{A.105}$$

This implies that when $t > \frac{4\ln^2\frac{\beta}{\gamma}\ln^2 m}{\ln^2\gamma}$ we achieve linear convergence with a convergence rate of γ. By our choice of η, the additional term in (A.105) is bounded as follows

$$: \frac{\ln^2(\beta/\gamma)\ln^2 m}{|\ln\gamma|} \leq O\left(\frac{(\beta-1)^2}{1-\gamma}\ln^2 m\right) = O\left(\eta\frac{mn^3L^2\ln^2 m}{d\delta}\right) < O(1).$$

The above lower bound of t that guarantees linear convergence can be further simplified. Since $\frac{x}{1+x} \leq \ln(1+x) \leq x$, we note $t \geq \frac{((\beta-1)^2+\frac{(\gamma-1)^2}{\gamma^2})\gamma^2\ln^2 m}{(\gamma-1)^2}$ and thus

$$t \geq \ln^2 m \times \left(1 + \frac{(\beta-1)^2}{(\gamma-1)^2}\right).$$

Recalling the expressions for β and γ, we conclude that linear convergence is achieved when

$$t > \Omega\left(\frac{(1+\alpha^2)^2}{(1-\alpha)^4}\frac{n^5L^2}{\delta^2 b}\ln^2 m\right). \tag{A.106}$$

The above argument holds for one training step with probability at least $1 - e^{-\Omega(m)}$. It extends to T steps with probability at least $1 - Te^{-\Omega(m)}$. We note that the number of epochs T can be bounded using the bound ϵ on the training error, the convergence

rate in (2.6), and (A.93), as follows:

$$\begin{aligned} T &= \ln(\epsilon/C_0\mathcal{L}(\boldsymbol{W}^{(0)}))/\ln\gamma < \Theta(\ln(\epsilon/C_0 n\sqrt{\ln m})/\ln\gamma) \\ &\leq O\left(\frac{\eta b\delta m}{n^2 d}(\ln\epsilon^{-1} + \ln(C_0 n\sqrt{\ln m}))\right). \end{aligned}$$

Therefore, the probability that ensures T-steps training with training error lower than ϵ is at least

$$1 - O\left(\frac{\eta b\delta m}{n^2 d}(\ln\epsilon^{-1} + \ln(C_0 n\sqrt{\ln m}))\right) e^{-\Omega(m)}.$$

Because $m > \Omega(\text{poly}(n, L, d, \delta^{-1}, b))$ and also $m > \Omega(\ln\ln\epsilon^{-1})$, this probability is of order $1 - e^{-\Omega(m)}$.

Lemma A.10

Assume the setup of §2.2 with learning rate $\eta < \frac{\delta^{3/2}d^{1/2}}{b^{1/2}n^3L^{15/2}m^{1/2}\ln^2 m}$ and neural network width m satisfying $\frac{m}{\ln^4 m} > \frac{(1+\alpha^2)^4}{(1-\alpha)^8}\Omega\left(\frac{n^8L^{15}d}{\delta^5 b}\right)$. Then during training according to Algorithm 3,

$$\|\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(0)}\| < O\left(\frac{\delta^{3/2}}{n^{3/2}L^{15/2}\ln^{3/2} m}\right) \quad \text{with probability at least } 1 - e^{-\Omega(\ln m)}.$$

Proof: The proof is similar to Lemma A.9, but with different bounds in this SGD setting. We first show bound the perturbation at initialization as follows:

$$\begin{aligned} \|\boldsymbol{W}^{(1)} - \boldsymbol{W}^{(0)}\| &= \eta\,\|\nabla_{\boldsymbol{W}}\mathcal{L}_B(\boldsymbol{W}^{(0)})\| \\ &\leq \eta\, O\left(\sqrt{\frac{mnb}{d}}\right)\mathcal{L}(\boldsymbol{W}^{(0)}) \\ &< O\left(\frac{\delta^{3/2}}{n^{3/2}L^{15/2}\ln^{3/2} m}\right). \end{aligned}$$

We denote $T_0 := \Omega\left(\frac{(1+\alpha^2)^2}{(1-\alpha)^4}\frac{n^5L^2}{\delta^2 b}\ln^2 m\right)$. Combining the SGD update step, (A.100),

(A.105), (A.106) and our choice of η yields

$$
\begin{aligned}
&\|\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(0)}\| \\
&\leq \eta \sum_{s=0}^{t-1} \|\nabla_{\boldsymbol{W}} \mathcal{L}_B(\boldsymbol{W}^{(s)})\| \\
&\leq \eta \sum_{s=0}^{t-1} \sqrt{\frac{mb}{d}} \sqrt{\mathcal{L}(\boldsymbol{W}^{(s)})} \\
&\leq \sqrt{\frac{mb}{d}} \eta \left(\frac{1}{1-\sqrt{\gamma}} + T_0 \right) \sqrt{\mathcal{L}(\boldsymbol{W}^{(0)})} \\
&\leq O(1) \sqrt{\frac{mb}{d}} \eta \left(\frac{1+\alpha^2}{(1-\alpha)^2} \frac{n^2 L \eta}{\delta m b} + \Omega \left(\frac{(1+\alpha^2)^2}{(1-\alpha)^4} \frac{n^5 L^2}{\delta^2 b} \ln^2 m \right) \right) \sqrt{\mathcal{L}(\boldsymbol{W}^{(0)})} \\
&\leq O(1) \sqrt{\frac{mb}{d}} \eta \Omega \left(\frac{(1+\alpha^2)^2}{(1-\alpha)^4} \frac{n^5 L^2}{\delta^2 b} \ln^2 m \right) \sqrt{\mathcal{L}(\boldsymbol{W}^{(0)})} \\
&\leq O(1) \sqrt{\frac{mb}{d}} \frac{d\delta}{m n^3 L^2 \ln^2 m} \Omega \left(\frac{(1+\alpha^2)^2}{(1-\alpha)^4} \frac{n^5 L^2}{\delta^2 b} \ln^2 m \right) \sqrt{\mathcal{L}(\boldsymbol{W}^{(0)})} \\
&= O(1) \frac{(1+\alpha^2)^2}{(1-\alpha)^4} \frac{\sqrt{d} n^2}{\sqrt{mb}\delta} \sqrt{\mathcal{L}(\boldsymbol{W}^{(0)})}.
\end{aligned}
$$

It is thus clear that when $\frac{m}{\ln^4 m} > \left(\frac{1+\alpha^2}{(1-\alpha)^2} \right)^4 \Omega \left(\frac{n^8 L^{15} d}{\delta^5 b} \right)$,

$$
\|\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(0)}\| \leq O\left(\frac{\delta^{3/2}}{n^{3/2} L^{15/2} \ln^{3/2} m} \right).
$$

□

A.2.8 Proof of Lemma 2.3

To simplify the proof, we study the generalization error of each output coordinate separately. We denote the $k-$th row of the matrix $\boldsymbol{B}$ by $\boldsymbol{B}_{k,\cdot}$ and treat it as a column

vector. For $k \in [d]$, we define function

$$f_k(\boldsymbol{x}; \boldsymbol{W}) := \boldsymbol{B}_{k,\cdot}^T \boldsymbol{h}_L(\boldsymbol{x}),$$

that is, $f_k(\boldsymbol{x}; \boldsymbol{W})$ is the $k-$th coordinate of the NN output vector. The loss function can be written as $\text{loss}_k(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{W}) := (f_k(\boldsymbol{x}; \boldsymbol{W}) - \boldsymbol{y}_k)^2$. Recall that the underlying measurable function $F(\boldsymbol{x})$ (i.e., $\boldsymbol{y}_i = F(\boldsymbol{x}_i)$) is a $d-$dimensional vector-valued function and we denote by $F_k(\boldsymbol{x})$ the $k-$th coordinate of $F(\boldsymbol{x})$. The generalization error is similarly defined as $R_k(\boldsymbol{W}) := \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_{\boldsymbol{X}}}(f_k(\boldsymbol{x}; \boldsymbol{W}) - F_k(\boldsymbol{x}))^2$. We also denote

$$B_\omega(\boldsymbol{W}^{(0)}) := \{\boldsymbol{W} : \|\boldsymbol{W} - \boldsymbol{W}^{(0)}\| < \omega\}.$$

From Lemma A.9 and Lemma A.10, with high probability, $\boldsymbol{W}^{(t)}$ is close to $\boldsymbol{W}^{(0)}$ during the training. Therefore, we just need to only consider NN functions whose parameters $\boldsymbol{W}$ fall in a small ball around $\boldsymbol{W}^{(0)}$, i.e. $\|\boldsymbol{W} - \boldsymbol{W}^{(0)}\| < \omega$, where $\omega < O\left(\frac{\delta^{3/2}}{n^{3/2} L^{15/2} \ln^{3/2} m}\right)$. For a given $k \in [d]$, we denote the corresponding function class as

$$\mathcal{G}_{k,\omega} := \{g : (x, y) \mapsto f_k(\boldsymbol{x}; \boldsymbol{W}) : \|\boldsymbol{W} - \boldsymbol{W}^{(0)}\| < \omega\}$$

We introduce the empirical Rademacher complexity on the dataset $\{x_i, y_i\}_{i=1}^n$ as follows:

$$\hat{\mathcal{R}}(\mathcal{G}_{k,\omega}) := \mathbb{E}_\sigma \sup_{g \in \mathcal{G}_{k,\omega}} \sum_{i=1}^n \sigma_i g(x_i, y_i)$$

For $k \in [d]$, we first bound the generalization error on the $k-$th coordinate of the output vector.

We first note that by (A.94) and Lemma A.4, with high probability that $f_k(\boldsymbol{x}_i; \boldsymbol{W}) < O(\ln^{1/4} m)$ for all $i \in [n]$ and $\|\boldsymbol{W} - \boldsymbol{W}^{(0)}\| < \omega$. We apply Theorem 11.3 in Mohri et al. (2018) with function class $\mathcal{G}_{k,\omega}$, and thus bound $R_k(\boldsymbol{W})$ with probability at least

$1 - \Omega(1/m)$ by

$$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{\boldsymbol{X}}}\text{loss}_k(\boldsymbol{x}, F(\boldsymbol{x}); \boldsymbol{W}) \leq \frac{1}{n}\sum_{i=1}^{n}\text{loss}_k(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{W}) + 2O\left(\ln^{1/4} m\right)\hat{\mathcal{R}}(\mathcal{G}_{k,\omega}) + O\left(\ln^{1/4} m\right) O\left(\sqrt{\frac{\ln 2m}{2n}}\right). \tag{A.107}$$

We note that the first term in (A.107) is bounded by the previously discussed training error, and the third term in (A.107) is very small when we collect a sufficiently large dataset since m is polynomially dependent on n. Next, we estimate the bound for the second term, the empirical Rademacher complexity.

$$\begin{aligned}
\hat{\mathcal{R}}(\mathcal{G}_{k,\omega}) = \mathbb{E}_\sigma \sup_{\boldsymbol{W}\in B_\omega(\boldsymbol{W}^{(0)})} \frac{1}{n}\sum_{i=1}^{n}\sigma_i\Big(& f_k(x_i, y_i; \boldsymbol{W}) - f_k(x_i, y_i; \boldsymbol{W}^{(0)}) \\
& - \langle \nabla_{\boldsymbol{W}} f_k(x_i, y_i; \boldsymbol{W}^{(0)}), \boldsymbol{W} - \boldsymbol{W}^{(0)}\rangle \\
& + f_k(x_i, y_i; \boldsymbol{W}^{(0)}) + \langle \nabla_{\boldsymbol{W}} f_k(x_i, y_i; \boldsymbol{W}^{(0)}), \boldsymbol{W} - \boldsymbol{W}^{(0)}\rangle\Big) \\
\leq \sup_{\boldsymbol{W}\in B_\omega(\boldsymbol{W}^{(0)})} \sup_i \big| & f_k(x_i, y_i; \boldsymbol{W}) - f_k(x_i, y_i; \boldsymbol{W}^{(0)}) \\
& - \langle \nabla_{\boldsymbol{W}} f_k(x_i, y_i; \boldsymbol{W}^{(0)}), \boldsymbol{W} - \boldsymbol{W}^{(0)}\rangle\big| && \text{(A.108)} \\
& + \mathbb{E}_\sigma \sup_{\boldsymbol{W}\in B_\omega(\boldsymbol{W}^{(0)})} \frac{1}{n}\sum_{i=1}^{n}\sigma_i f_k(x_i, y_i; \boldsymbol{W}^{(0)}) && \text{(A.109)} \\
& + \mathbb{E}_\sigma \sup_{\boldsymbol{W}\in B_\omega(\boldsymbol{W}^{(0)})} \frac{1}{n}\sum_{i=1}^{n}\sigma_i \langle \nabla_{\boldsymbol{W}} f_k(x_i, y_i; \boldsymbol{W}^{(0)}), \boldsymbol{W} - \boldsymbol{W}^{(0)}\rangle. && \text{(A.110)}
\end{aligned}$$

We first consider (A.109), since there is no dependence on $\boldsymbol{W}$, it is clear that

$$\mathbb{E}_\sigma \sup_{\boldsymbol{W}\in B_\omega(\boldsymbol{W}^{(0)})} \frac{1}{n}\sum_{i=1}^{n}\sigma_i f_k(x_i, y_i; \boldsymbol{W}^{(0)}) = \frac{1}{n}\sum_{i=1}^{n} f_k(x_i, y_i; \boldsymbol{W}^{(0)})\mathbb{E}_\sigma\sigma_i = 0. \tag{A.111}$$

In order to help bound (A.108), which is the first term in our bound of the empirical Rademacher complexity for the NN function class, we introduce the following lemma.

Lemma A.11

If $\omega < O\left(\frac{\delta^{3/2}}{n^{3/2}L^{15/2}\ln^{3/2} m}\right)$ and $\|\boldsymbol{W}-\boldsymbol{W}^{(0)}\| < \omega$, then for any $\boldsymbol{x} \in \boldsymbol{R}^p$, with probability at least $1 - \exp(-\Omega(\sqrt{m}/\ln m)$,

$$\begin{aligned}&\left|f_k(\boldsymbol{x};\boldsymbol{W}) - f_k(\boldsymbol{x};\boldsymbol{W}^{(0)}) - \langle \nabla_{\boldsymbol{W}} f_k(\boldsymbol{x};\boldsymbol{W}^{(0)}), \boldsymbol{W}-\boldsymbol{W}^{(0)}\rangle\right| \\ &< \frac{1-\alpha}{\sqrt{1+\alpha^2}} O(\omega^{4/3}L^2\sqrt{m\ln m}).\end{aligned} \qquad \square$$

We prove Lemma A.11 at the end of this section after we finalize the proof of Lemma 2.3 (while applying Lemma A.11).

Using Lemma A.11, the term (A.108) can be bounded as

$$\begin{aligned}&\sup_i \left|f_k(x_i,y_i;\boldsymbol{W}) - f_k(x_i,y_i;\boldsymbol{W}^{(0)}) - \langle \nabla_{\boldsymbol{W}} f_k(x_i,y_i;\boldsymbol{W}^{(0)}), \boldsymbol{W}-\boldsymbol{W}^{(0)}\rangle\right| \\ &\leq \frac{1-\alpha}{\sqrt{1+\alpha^2}}\omega^{4/3}L^2\sqrt{m\ln m}.\end{aligned}$$

Applying Cauchy-Schwarz inequality and Jessen's inequality to (A.110) and using Lemma 2.2, we conclude

$$\begin{aligned}&\left|\mathbb{E}_\sigma \sup_{\boldsymbol{W}\in B_\omega(\boldsymbol{W}^{(0)})} \frac{1}{n}\sum_{i=1}^n \sigma_i \langle \nabla_{\boldsymbol{W}} f_k(\boldsymbol{x}_i,\boldsymbol{y}_i;\boldsymbol{W}^{(0)}), \boldsymbol{W}-\boldsymbol{W}^{(0)}\rangle\right| \\ &\leq \omega \mathbb{E}_\sigma \sup_{\boldsymbol{W}\in B_\omega(\boldsymbol{W}^{(0)})} \frac{1}{n}\sum_{i=1}^n\sum_{l=1}^L \|\nabla_{\boldsymbol{W}_l} f_k(\boldsymbol{x}_i,\boldsymbol{y}_i;\boldsymbol{W}^{(0)})\|_F \\ &\leq \frac{\omega}{n}\sum_{l=1}^L \sqrt{\sum_{i=1}^n \|\nabla_{\boldsymbol{W}_l} f_k(\boldsymbol{x}_i,\boldsymbol{y}_i;\boldsymbol{W}^{(0)})\|_F^2} \\ &\leq \frac{\omega}{n} L\sqrt{mn} \leq \frac{\omega L\sqrt{m}}{\sqrt{n}}\end{aligned}$$

Using the bound for (A.108), (A.109) and (A.110) in (A.107), it follows that with

probability at least $1 - e^{-\Omega(\ln m)}$

$$R_k(\boldsymbol{W}) \leq \frac{1}{n}\text{loss}_k(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{W}) + \frac{1-\alpha}{\sqrt{1+\alpha^2}}\, O\left(\ln m \sqrt{m} L^2 \omega^{4/3}\right) + \omega\, O\left(L\sqrt{m \ln m/n}\right) + O\left(\sqrt{\frac{\ln m}{n}}\right).$$

Summing over $k \in [d]$, we conclude Lemma 2.3 as follows

$$R(\boldsymbol{W}) \leq \frac{1}{n}\text{loss}(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{W}) + \frac{1-\alpha}{\sqrt{1+\alpha^2}}\, O\left(d \ln m \sqrt{m} L^2 \omega^{4/3}\right) + O\left(d\sqrt{m \ln m/n} L\omega\right) + O\left(d\sqrt{\frac{\ln m}{n}}\right).$$

Finally, we complete this section by presenting the proof of Lemma A.11.

Proof: [Proof of Lemma A.11] Using the notation of §A.2.3 (in particular, $\boldsymbol{h}_l$ and $\boldsymbol{h}_l^{(0)}$) and the definitions of $f_k(\boldsymbol{x}; \boldsymbol{W}^{(0)})$ and $f_k(\boldsymbol{x}; \boldsymbol{W})$ and recalling that $\boldsymbol{h}_0 = \boldsymbol{h}_0^{(0)}$ and $\boldsymbol{h}_l = \boldsymbol{D}_l \boldsymbol{W}_l \boldsymbol{h}_{l-1}$ we derive the following expression:

$$\begin{aligned}
&f_k(\boldsymbol{x}; \boldsymbol{W}) - f_k(\boldsymbol{x}; \boldsymbol{W}^{(0)}) && \text{(A.112)}\\
&= \boldsymbol{B}_{k,\cdot}^T \left(\boldsymbol{D}_L \boldsymbol{W}_L \cdots \boldsymbol{D}_1 \boldsymbol{W}_1 - \boldsymbol{D}_L^{(0)} \boldsymbol{W}_L^{(0)} \cdots \boldsymbol{D}_1^{(0)} \boldsymbol{W}_1^{(0)}\right) \boldsymbol{A}\boldsymbol{x}\\
&= \boldsymbol{B}_{k,\cdot}^T \Big(\boldsymbol{D}_L \boldsymbol{W}_L \boldsymbol{h}_{L-1} - \boldsymbol{D}_L^{(0)} \boldsymbol{W}_L^{(0)} \boldsymbol{h}_{L-1}\\
&\quad + \boldsymbol{D}_L^{(0)} \boldsymbol{W}_L^{(0)} \boldsymbol{h}_{L-1} - \boldsymbol{D}_L^{(0)} \boldsymbol{W}_L^{(0)} \boldsymbol{D}_{L-1}^{(0)} \boldsymbol{W}_{L-1}^{(0)} \boldsymbol{h}_{L-2}\\
&\quad + \boldsymbol{D}_L^{(0)} \boldsymbol{W}_L^{(0)} \boldsymbol{D}_{L-1}^{(0)} \boldsymbol{W}_{L-1}^{(0)} \boldsymbol{h}_{L-2} - \boldsymbol{D}_L^{(0)} \boldsymbol{W}_L^{(0)} \boldsymbol{D}_{L-1}^{(0)} \boldsymbol{W}_{L-1}^{(0)} \boldsymbol{D}_{L-2}^{(0)} \boldsymbol{W}_{L-2}^{(0)} \boldsymbol{h}_{L-3}\\
&\cdots \cdots && \text{(A.113)}\\
&+ \boldsymbol{D}_L^{(0)} \boldsymbol{W}_L^{(0)} \cdots \boldsymbol{D}_2^{(0)} \boldsymbol{W}_2^{(0)} \boldsymbol{h}_1 - \boldsymbol{D}_L^{(0)} \boldsymbol{W}_L^{(0)} \cdots \boldsymbol{D}_2^{(0)} \boldsymbol{W}_2^{(0)} \boldsymbol{D}_1^{(0)} \boldsymbol{W}_1^{(0)} \boldsymbol{h}_0\Big).
\end{aligned}$$

Let a and b be two integers in $[1, L]$. If $b \leq a$, we denote

$$(\boldsymbol{D}^{(0)}\boldsymbol{W}^{(0)})_{a\mapsto b} := \boldsymbol{D}_a^{(0)}\boldsymbol{W}_a^{(0)}\boldsymbol{D}_{a-1}^{(0)}\boldsymbol{W}_{a-1}^{(0)}\cdots\boldsymbol{D}_b^{(0)}\boldsymbol{W}_b^{(0)}.$$

If $a < b$, we denote

$$(\boldsymbol{D}^{(0)}\boldsymbol{W}^{(0)})_{a\mapsto b} := \boldsymbol{I}.$$

Applying $\boldsymbol{h}_l = \boldsymbol{D}_l\boldsymbol{W}_l\boldsymbol{h}_{l-1}$ for the first term in each line of (A.113), (A.113) can be written as

$$\begin{aligned}
&f_k(\boldsymbol{x};\boldsymbol{W}) - f_k(\boldsymbol{x};\boldsymbol{W}^{(0)})\\
&= \boldsymbol{B}_{k,\cdot}^T\Big((\boldsymbol{D}_L\boldsymbol{W}_L - \boldsymbol{D}_L^{(0)}\boldsymbol{W}_L^{(0)})\boldsymbol{h}_{L-1}\\
&+ (\boldsymbol{D}^{(0)}\boldsymbol{W}^{(0)})_{L\mapsto L}(\boldsymbol{D}_{L-1}\boldsymbol{W}_{L-1} - \boldsymbol{D}_{L-1}^{(0)}\boldsymbol{W}_{L-1}^{(0)})\boldsymbol{h}_{L-2}\\
&+ (\boldsymbol{D}^{(0)}\boldsymbol{W}^{(0)})_{L\mapsto L-1}(\boldsymbol{D}_{L-2}\boldsymbol{W}_{L-2} - \boldsymbol{D}_{L-2}^{(0)}\boldsymbol{W}_{L-2}^{(0)})\boldsymbol{h}_{L-3}\\
&\cdots\ \cdots\\
&+ (\boldsymbol{D}^{(0)}\boldsymbol{W}^{(0)})_{L\mapsto 2}(\boldsymbol{D}_1\boldsymbol{W}_1 - \boldsymbol{D}_1^{(0)}\boldsymbol{W}_1^{(0)})\boldsymbol{h}_0\Big)\\
&= \boldsymbol{B}_{k,\cdot}^T\sum_{l=1}^{L}(\boldsymbol{D}^{(0)}\boldsymbol{W}^{(0)})_{L\mapsto l+1}(\boldsymbol{D}_l\boldsymbol{W}_l - \boldsymbol{D}_l^{(0)}\boldsymbol{W}_l^{(0)})\boldsymbol{h}_{l-1}\\
&= \boldsymbol{B}_{k,\cdot}^T\sum_{l=1}^{L}(\boldsymbol{D}^{(0)}\boldsymbol{W}^{(0)})_{L\mapsto l+1}(\boldsymbol{D}_l - \boldsymbol{D}_l^{(0)})\boldsymbol{W}_l\boldsymbol{h}_{l-1} && \text{(A.114)}\\
&\quad + \boldsymbol{B}_{k,\cdot}^T\sum_{l=1}^{L}(\boldsymbol{D}^{(0)}\boldsymbol{W}^{(0)})_{L\mapsto l+1}\boldsymbol{D}_l^{(0)}(\boldsymbol{W}_l - \boldsymbol{W}_l^{(0)})(\boldsymbol{h}_{l-1} - \boldsymbol{h}_{l-1}^{(0)}) && \text{(A.115)}\\
&\quad + \boldsymbol{B}_{k,\cdot}^T\sum_{l=1}^{L}(\boldsymbol{D}^{(0)}\boldsymbol{W}^{(0)})_{L\mapsto l+1}\boldsymbol{D}_l^{(0)}(\boldsymbol{W}_l - \boldsymbol{W}_l^{(0)})\boldsymbol{h}_{l-1}^{(0)}. && \text{(A.116)}
\end{aligned}$$

According statement 1 in Lemma A.4, with probability at least $1 - e^{-\Omega(\sqrt{m}/\ln m)}$,$\|\boldsymbol{D}_l'\boldsymbol{g}_l\| < (1-\alpha)/\sqrt{1+\alpha^2}O(L^{3/2}\omega)$ and $\|\boldsymbol{D}_l'\|_0 \leq O(m\omega^{2/3}L)$. Combining this with statement 4 in Lemma A.3 with $\boldsymbol{v} = (1, 1, \cdots, 1)^T \in \mathbb{R}^d$, we

bound the norm $\|(\boldsymbol{B}_k^T \boldsymbol{D}^{(0)} \boldsymbol{W}^{(0)})_{L \mapsto l+1}\|$ by $O(\omega^{1/3}\sqrt{mL \ln m})$ with probability at least $1 - \exp(-\Omega(m\omega^{3/2} L \ln m)$. Then (A.114) can be bounded (with the same probability) by

$$\frac{1-\alpha}{\sqrt{1+\alpha^2}} O(\omega^{4/3} L^2 \sqrt{m \ln m}). \tag{A.117}$$

By using statement 3 in Lemma A.4, i.e., $\|\boldsymbol{h}_l - \boldsymbol{h}_l^{(0)}\| < O(\omega L^{5/2} \ln m)$ with probability at least $1 - e^{-\Omega(\sqrt{m}/\ln m)}$, we note the norm of the summation in (A.115) is bounded by $O(\omega^2 L^{5/2} \ln m)$ (with the latter probability), which is much smaller than (A.117) when ω is small as given.

By noting that the gradient of $f_k(\boldsymbol{x}; \boldsymbol{W})$ with respect to $\boldsymbol{W}_l$ can be written as

$$\nabla_{\boldsymbol{W}_l} f_k(\boldsymbol{x}; \boldsymbol{W}^{(0)}) = (\boldsymbol{B}_{k,\cdot}^T \boldsymbol{D}_L^{(0)} \boldsymbol{W}_L^{(0)} \cdots \boldsymbol{W}_{l+1}^{(0)} \boldsymbol{D}_l^{(0)})^T \boldsymbol{h}_{l-1}^{(0)T},$$

we express the summands in (A.116) as follows

$$\langle \nabla_{\boldsymbol{W}_l} f_k(\boldsymbol{x}; \boldsymbol{W}^{(0)}), \boldsymbol{W}_l - \boldsymbol{W}_l^{(0)} \rangle \equiv \boldsymbol{B}_{k,\cdot}^T \boldsymbol{D}_L^{(0)} \boldsymbol{W}_L^{(0)} \cdots \boldsymbol{W}_{l+1}^{(0)} \boldsymbol{D}_l^{(0)} (\boldsymbol{W}_l - \boldsymbol{W}_l^{(0)}) \boldsymbol{h}_{l-1}^{(0)}. \tag{A.118}$$

Using (A.118) and bounding (A.114) and (A.115) by (A.117), we conclude that with probability at least $1 - e^{-\Omega(\sqrt{m} \ln m)}$

$$\begin{aligned} &\left| f_k(\boldsymbol{x}; \boldsymbol{W}) - f_k(\boldsymbol{x}; \boldsymbol{W}^{(0)}) - \langle \nabla_{\boldsymbol{W}} f_k(\boldsymbol{x}; \boldsymbol{W}^{(0)}), \boldsymbol{W} - \boldsymbol{W}^{(0)} \rangle \right| \\ &< \frac{1-\alpha}{\sqrt{1+\alpha^2}} O(\omega^{4/3} L^2 \sqrt{m \ln m}). \end{aligned}$$

□

A.2.9 Proof of Theorem 2.3

The key idea of the proof of this theorem is to establish a bound for ω, such that $\|\boldsymbol{W}^{(t)} - \boldsymbol{W}\| < \omega$ during training. Considering the learning rate η and training steps t, we first establish a simple bound for ω as

$$\|\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(0)}\| \leq \sum_{t=0}^{t-1} \eta \|\nabla_{\boldsymbol{W}} \mathcal{L}^{(t)}\| \leq \eta \sqrt{\frac{mn}{d}} \sum_{t=0}^{t} \sqrt{\mathcal{L}^{(t)}} \leq \eta t \sqrt{\frac{nm \ln m}{d}}.$$

Furthermore, in the proof of Lemma A.9, the following universal bound of ω was introduced:

$$\|\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(0)}\| \leq O\left(\sqrt{\frac{nd}{\delta m}}\right) \sqrt{\mathcal{L}^{(0)}}.$$

By combining these two bounds with the universal bound $\omega < O(\frac{\delta^{3/2}}{n^{3/2} L^{15/2} \ln^{3/2} m})$ and using Lemma 2.3, we conclude the theorem.

A.2.10 Generalization Error Bound for SGD

We present a theorem similar to Theorem 2.3 that establishes the upper bound of the generalization error for SGD.

Theorem A.1

Assume the setup of §2.2 with SGD, where $m = \Theta\left(\frac{n^{13+2\epsilon} L^{15+2\epsilon} d^{1+2\epsilon}}{b\delta^{5-2\epsilon}}\right)$ for $\epsilon > 0$ and $\eta = \Theta(\frac{d\delta}{n^3 L^3 m \ln^2 m})$. Assume further that m is larger than its lower bound and η is smaller than its upper bound in Theorem 2.2 (by an appropriate choice of the hidden constants in Θ and in comparison to the constants hidden in the lower bound of m and the upper bound of η in Theorem 2.2). Then at a given training epoch t, with

probability at least $1 - e^{-\Omega(\ln m)}$, the generalization error is bounded as follows

$$R(\boldsymbol{W}^{(t)}) \leq \gamma^t O(\ln m) + \min\left\{\left(\frac{1-\alpha}{\sqrt{1+\alpha^2}}\right) O\left(\frac{d^{1/3}t^{4/3}}{m^{1/6}n^{10/3}L^2\ln^{8/3} m}\right), \right.$$
$$\left. O\left(\frac{d^{3/2+\epsilon}n^{2+\epsilon}}{b^{1/2}\delta^{1/2-\epsilon}L^{1/2-\epsilon}\ln m}\right)\right\}$$
$$+ \min\left\{O\left(\frac{\sqrt{d}\, t}{n^3L^2\ln^{3/2} m}\right), O\left(\frac{n^{2+\epsilon}L^{2+\epsilon}d^{1/2+\epsilon}}{b^{1/2}\delta^{1-\epsilon}\ln m}\right)\right\} + O\left(d\sqrt{\frac{\ln m}{n}}\right). \tag{A.119}$$

□

The proof is similar to the proof of Theorem 2.3. We estimate the bound of ω when t is small as

$$\|\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(0)}\| < O\left(\frac{mn\eta t}{d}\right)\sqrt{\mathcal{L}^{(0)}}.$$

Also, the bound of ω in the entire training for SGD can be obtained in the proof of Lemma A.10 as

$$\|\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(0)}\| < O\left(\frac{d\sqrt{n}}{\delta\sqrt{mb}}\sqrt{\mathcal{L}^{(0)}}\right).$$

Then combining these two bounds of ω and using Lemma 2.3, we could conclude the theorem.

A.2.11 Special dataset

In this section, we consider a special class of datasets and improve our theory for datasets from this class. We first introduce the special dataset and establish the assumption, then present theorems to bound the convergence rate and generalization error under this assumption. The proof will be given in Appendix A.2.13.

First, with the parameters $\boldsymbol{W}^{(0)}$ before the $l-$th layer, we denote the output at

the $l-$th layer as

$$\mathcal{N}_l(\boldsymbol{x}; \boldsymbol{A}, \boldsymbol{W}_1^{(0)}, \boldsymbol{W}_2^{(0)}, \cdots \boldsymbol{W}_{l-1}^{(0)}, \boldsymbol{u}) = \tilde{\sigma}_\alpha(\boldsymbol{u}^T \tilde{\sigma}_\alpha(\boldsymbol{W}_{l-1}^{(0)} \tilde{\sigma}_\alpha(\boldsymbol{W}_{l-2}^{(0)} \cdots \tilde{\sigma}_\alpha(\boldsymbol{W}_1^{(0)} \boldsymbol{A}\boldsymbol{x})))),$$

and define the following class of functions:

$$\begin{aligned}\mathcal{F}_l := \Big\{ & \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \cdots, f_d(\boldsymbol{x}))^T : \mathbb{R}^p \mapsto \mathbb{R}^d, \text{ where} \\ & f_j(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{u}} c_j(\boldsymbol{u}) \mathcal{N}_l(\boldsymbol{x}; \boldsymbol{A}, \boldsymbol{W}_1^{(0)} \cdots \boldsymbol{W}_{l-1}^{(0)}, \boldsymbol{u}) \text{ for } \boldsymbol{u} \in \mathbb{R}^m \sim N\left(0, \frac{2}{m}\right) \text{ and} \\ & c_j : \mathbb{R}^m \mapsto \mathbb{R} \text{ such that } |c_j(\cdot)| \leq 1 \text{ for } j \in [d] \Big\}.\end{aligned} \tag{A.120}$$

We note that this function class $\mathcal{F}_l$ includes functions defined by an $l-$layer leaky ReLU neural network, where the first $l-1$ layers use the initialized parameters $\boldsymbol{W}^{(0)}$ and only the parameters of the $l-$th layer are tuned with a certain regularization condition ($\|c_j\|_\infty < 1$). Given this function class, we restrict our discussion to datasets satisfying the assumption below. For such datasets, we can improve the upper bound for the convergence rate, the lower bound for the width m, and the upper bound for the generalization error.

Assumption A.1

For any small constant $0 < \lambda < \frac{1}{2\sqrt{nd}}$, there exists $f \in \mathcal{F}_{L-1}$ such that

$$\|\boldsymbol{f}(\boldsymbol{x}_i) - (\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i)\| < \lambda, \text{ for all } \ i \in [n].$$

Theorem A.2

Assume the setup in §2.2 with a dataset satisfying Assumption A.1, where both $m/ln^4 m > \Omega(d^5 n L^{12})$ and $m > \Omega(\ln \ln \epsilon^{-1})$, and the NN is trained according to

Algorithm 2, with learning rate $\eta \leq O(\frac{d}{nL^2m})$. Then with probability at least $1 - e^{-\Omega(\ln m)}$

$$\mathcal{L}(\boldsymbol{W}^{(T)}) < \epsilon \text{ and } \mathcal{L}(\boldsymbol{W}^{(t)}) \leq \gamma^t \mathcal{L}(\boldsymbol{W}^{(0)}), \forall t \leq T,$$

where

$$\gamma = 1 - \Omega\left(\frac{(1-\alpha)^2}{1+\alpha^2}\frac{\eta m}{d^2}\right) \text{ and } T = \frac{\ln\left(\epsilon/\mathcal{L}(\boldsymbol{W}^{(0)})\right)}{\ln \gamma}. \qquad \square$$

Theorem A.3

Assume the setup of §2.2 with GD, a dataset satisfying Assumption A.1, $m = \Theta(n^{1+2\tau}L^{12+2\tau}d^{5+2\tau})$ for $\tau > 0$ and $\eta = \Theta(\frac{d}{nL^2m})$. Assume further that m is larger than its lower bound and η is smaller than its upper bound in Theorem A.2 (by an appropriate choice of the hidden constants in Θ and compared to the constants hidden in the lower bound of m and in the upper bound of η in Theorem A.2). Then at a given training epoch $t \leq T$ (see (2.4) for T), with probability at least $1 - e^{-\Omega(\ln m)}$, the generalization error is bounded as follows

$$\begin{aligned} R(\boldsymbol{W}^{(t)}) \leq \gamma^t \mathcal{L}(\boldsymbol{W}^{(0)}) &+ \min\left\{O\left(\frac{d^{3/2+\tau}n^{1/2+\tau}L^\tau}{\ln m}\right), O\left(\frac{1-\alpha}{\sqrt{1+\alpha^2}}\frac{d^{1/3}t^{4/3}}{m^{1/6}n^{2/3}L^{2/3}}\right)\right\} \\ &+ \min\left\{O\left(\frac{\sqrt{d\ln m}\, t}{nL}\right), O\left(\frac{n^\tau L^{1+\tau}d^{2+\tau}}{\ln m}\right)\right\} + O\left(d\sqrt{\frac{\ln m}{n}}\right). \end{aligned}$$

$\square$

We notice that for datasets satisfying Assumption A.1 several significant improvements from the previous estimates are obtained. Firstly, the lower bound for m is improved to linear dependence on n, whereas in the general scenario the lower bound grows as n^5. Secondly, the bound of $1 - \gamma$ is improved in Theorem A.2 by a factor

of $\frac{n}{\delta d}$. Thirdly, several terms in the generalization error bound in Theorem A.3 are improved from Theorem 2.3, including the first term in the first minimum is improved by a factor of $\frac{1}{L}$ and the second term in the second minimum is improved by a factor of $\frac{\sqrt{d^3\delta}}{L\sqrt{n}}$. On the other hand, the optimal choice of α remains the same as the dependence on α is the same as that in Theorems 2.1 and 2.3.

A.2.12 Convergence Theorem for General Convex Loss Functions

We extend our convergence theory, in particular Theorem 2.1, to convex loss functions, i.e., loss functions of the form

$$\mathcal{L}_{\text{convex}}(\boldsymbol{W}) = \sum_{i}^{n} l(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i), \quad \text{where } l(\boldsymbol{y}_i, \cdot) \text{ is convex.} \tag{A.121}$$

These include common loss functions for classification, such as the binary cross entropy and categorical cross entropy. Furthermore, it also includes the following loss function suggested in (Kumar et al., 2023):

$$\mathcal{L}_{\exp}(\boldsymbol{W}) := \frac{1}{2}\sum_{i}^{n} e^{\lambda\|\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i\|^2}., \tag{A.122}$$

Kumar et al. (2023) obtained a special bound for the generalization error when using this loss function. We later use the following theorem and the proposition of Kumar et al. (2023) to infer that $\alpha = -1$ is also optimal for generalization when using re-weighted gradient descent and overparameterized neural networks.

We next formulate the main theorem using the following definition. Let $\boldsymbol{W}^*$ denote the matrix of parameters minimizing the loss function and define

$$\mathcal{E}^{(t)} := \mathcal{L}_{\text{convex}}(\boldsymbol{W}^{(t)}) - \mathcal{L}_{\text{convex}}(\boldsymbol{W}^*).$$

Theorem A.4

Assume the setup of §2.2 with the convex loss function defined in (A.121), where the width m satisfies both $m/\ln^4 m \geq \frac{1+\alpha^2}{(1-\alpha)^2}\Omega(\frac{n^6L^{16}d}{\delta^4})$ and $m > \Omega(\ln(\epsilon^{-1}\ln\epsilon^{-1}))$ and the training is according to Algorithm 2 with learning rate $\eta \leq O(\frac{d}{nL^2m})$. Then with probability at least $1-\exp^{-\Omega(\ln m)}$,

$$\mathcal{E}^{(T)} < \epsilon \text{ and } \mathcal{E}^{(t+1)} \leq \gamma^{(t)}\mathcal{E}^{(t)}, \quad \forall t \leq T, \tag{A.123}$$

where

$$\gamma^{(t)} = 1-\mathcal{E}^{(t)}\Omega\left(\frac{(1-\alpha)^2}{(1+\alpha^2)}\frac{\delta\eta m}{n^2dL}\right) \text{ and } T \leq O\left(\frac{1+\alpha^2}{(1-\alpha)^2}\frac{n^2dL}{\eta\delta m\epsilon}(\ln\epsilon^{-1}+\ln(n\sqrt{\ln m}))\right). \tag{A.124}$$

□

Combining (A.123) and the expression for $\gamma^{(t)}$ in (A.124), we note that the rate of convergence is slower than the one in (2.3) and that $\alpha = -1$ corresponds to the smallest upper bound for the number of epochs needed for the training error to be smaller than ϵ.

Proposition 3.1 in Kumar et al. (2023) implies that minimizing the generalization error bound is equivalent to minimizing the training error when using the modified loss function given in (A.122). Applying (A.124) of Theorem A.4, we conclude that when using the modified loss function in (A.122) for training, the choice of $\alpha = -1$ yields the smallest bound for the required number of training epochs to achieve a bound ϵ on the training error. Using this observation with arbitrarily small ϵ and the above discussed theory of Kumar et al. (2023) we can conclude that $\alpha = -1$ minimizes the generalization error bound with the smallest upper bound on the number of training epochs for which the training error is guaranteed to be less than ϵ. Nevertheless, this discussion involves an upper bound we obtained for the number of epochs and does

not apply to the actual number of epochs. Consequently, the above stated prediction may not be precise, that is, it is possible that at a smaller number of epochs than the bound, one may obtain an error less than ϵ by $\alpha \neq -1$. For a synthetic dataset, we empirically verified the predicted optimal choice of $\alpha = -1$ (see Figure A.4).

Proof: [Proof of Theorem A.4]

We consider the modified loss function defined in (A.121), and, for simplicity, we let $\gamma = 1$ in this section, while the proof can be easily extended for any $\gamma > 0$. By introducing the convex function $l(\boldsymbol{y}, \boldsymbol{z})$, the loss function for each data point $i \in [n]$ can be written as

$$\text{loss}_{\text{convex}}(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{W}) := l(\boldsymbol{y}_i, \boldsymbol{g}_{i,L+1}(\boldsymbol{x}_i; \boldsymbol{W}))$$

We denote the following notation in this subsection,

$$\boldsymbol{e}_i := \nabla_{\boldsymbol{z}} l(\boldsymbol{y}_i, \boldsymbol{g}_{i,L+1}(\boldsymbol{x}_i; \boldsymbol{W})). \tag{A.125}$$

We will establish similar gradient bounds as in Lemma 2.2 and a similar semi-smoothness inequality as in Lemma 2.1 for the loss function defined in (A.121), and then we will prove the convergence theory with this loss function using some of the above established results.

For the first part, in order to achieve the gradient bound, we follow the proof of Lemma 2.2 with the modified loss function defined in (A.121). The proof is mostly the same as the one in Appendix A.2.4. The only difference is the use of the previous definition in (A.46). We remark that it is straightforward to verify that $\boldsymbol{G}_{i,l}(\boldsymbol{e}_i; \boldsymbol{W}) \equiv \nabla_{\boldsymbol{W}_l} \text{loss}_{\text{convex}}(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{W})$ by using the definition of $\boldsymbol{e}_i$ in (A.125). By

Lemma A.6 and Lemma A.7

$$\|\nabla_{\boldsymbol{W}_l}\mathcal{L}_{\text{convex}}(\boldsymbol{W})\|_F^2 \leq \sum_{i=1}^{n}\|\boldsymbol{e}_i\|^2 O\left(\frac{mn}{d}\right), \quad \text{for } \ l \in [L] \tag{A.126}$$

$$\|\nabla_{\boldsymbol{W}}\mathcal{L}_{\text{convex}}(\boldsymbol{W})\|_F^2 \geq \sum_{i=1}^{n}\|\boldsymbol{e}_i\|^2 \Omega\left(\frac{(1-\alpha)^2}{(1+\alpha^2)}\frac{\delta m}{nd}\right). \tag{A.127}$$

Remark: In the above bounds, when we use the MSE loss function, $\sum_{i=1}^{n}\|\boldsymbol{e}_i\|^2 \equiv \mathcal{L}$, which yields the original bounds provided in Lemma 2.2.

In order to show the semi-smoothness, we follow the proof of Lemma 2.1 in Appendix A.2.5, where most parts are exactly the same. By using (A.127) and (A.126), and plugging the notation of $\boldsymbol{e}_i$ (defined in (A.125)) into (A.80), the semi-smoothness inequality becomes

$$\begin{aligned}\mathcal{L}_{\text{convex}}(\boldsymbol{W}+\boldsymbol{W}') \leq \mathcal{L}_{\text{convex}}(\boldsymbol{W}) + \langle\nabla_{\boldsymbol{W}}\mathcal{L}_{\text{convex}}(\boldsymbol{W}), \boldsymbol{W}'\rangle + \frac{nL^2m}{d}O(\|\boldsymbol{W}'\|_2^2) \\ + \frac{(1-\alpha)\omega^{1/3}L^2\sqrt{mn\sum_{i=1}^{n}\|\boldsymbol{e}_i^{(t)}\|^2\ln m}}{\sqrt{d(1+\alpha^2)}}O(\|\boldsymbol{W}'\|_2).\end{aligned} \tag{A.128}$$

Lastly, we prove the convergence for this loss function. By using (A.128) and the same argument discussed in §2.4.1, the inequality (2.16) still holds. Then using the lower bound of the gradient in (A.127), (2.16) becomes

$$\mathcal{L}_{\text{convex}}(\boldsymbol{W}^{(t+1)}) \leq \mathcal{L}_{\text{convex}}(\boldsymbol{W}^{(t)}) - \Omega\left(\frac{(1-\alpha)^2}{(1+\alpha^2)}\frac{\delta\eta m}{nd}\right)\sum_{i=1}^{n}\|\boldsymbol{e}_i^{(t)}\|^2. \tag{A.129}$$

By convexity of $l(\boldsymbol{y},\boldsymbol{z})$, we first establish that for any $i \in [n]$ and any $\boldsymbol{y},\boldsymbol{z} \in \mathbb{R}^d$,

$$l(\boldsymbol{y}_i,\boldsymbol{y}) - \boldsymbol{y}_{\boldsymbol{y}_i,\boldsymbol{z}} \leq \langle\nabla_{\boldsymbol{y}}l(\boldsymbol{y}_i,\boldsymbol{y}), \boldsymbol{y}-\boldsymbol{z}\rangle \leq \|\nabla_{\boldsymbol{y}}l(\boldsymbol{y}_i,\boldsymbol{y})\|\|\boldsymbol{y}-\boldsymbol{z}\|.$$

We denote by $\boldsymbol{W}^*$ the optimal parameter that minimizes $\mathcal{L}_{\text{exp}}(\boldsymbol{W})$. Letting $\boldsymbol{y} :=$

$\boldsymbol{g}_{i,L+1}(\boldsymbol{x}_i;\boldsymbol{W}^{(t)})$ and $\boldsymbol{z} := \boldsymbol{g}_{L+1}(\boldsymbol{x}_i;\boldsymbol{W}^*)$ and using the above inequality result in

$$\|\boldsymbol{e}_i^{(t)}\| \geq \frac{l(\boldsymbol{y}_i, \boldsymbol{g}_{i,L+1}(\boldsymbol{x}_i;\boldsymbol{W}^{(t)})) - l(\boldsymbol{y}_i, \boldsymbol{g}_{L+1}(\boldsymbol{x}_i;\boldsymbol{W}^*))}{\|\boldsymbol{g}_{i,L+1}(\boldsymbol{x}_i;\boldsymbol{W}^{(t)}) - \boldsymbol{g}_{L+1}(\boldsymbol{x}_i;\boldsymbol{W}^*)\|}. \tag{A.130}$$

Using Lemma A.5, yields that, with probability at least $1 - e^{-\Omega(\ln m)}$,

$$\begin{aligned}
&\|\boldsymbol{g}_{i,L+1}(\boldsymbol{x}_i;\boldsymbol{W}^*) - \boldsymbol{g}_{i,L+1}(\boldsymbol{x}_i;\boldsymbol{W}^{(t)})\| \leq \|\boldsymbol{g}_{i,L+1}(\boldsymbol{x}_i;\boldsymbol{W}^*) - \boldsymbol{g}_{i,L+1}(\boldsymbol{x}_i;\boldsymbol{W}^{(0)})\| \\
&\quad + \|\boldsymbol{g}_{i,L+1}(\boldsymbol{x}_i;\boldsymbol{W}^{(t)}) - \boldsymbol{g}_{i,L+1}(\boldsymbol{x}_i;\boldsymbol{W}^{(0)})\| \\
&\leq (\|\boldsymbol{W}^{(0)} - \boldsymbol{W}^*\| + \omega)\|\nabla_{\boldsymbol{W}}\boldsymbol{g}_{i,L+1}(\boldsymbol{x}_i;\boldsymbol{W}^{(0)})\| \leq O(\sqrt{L}).
\end{aligned} \tag{A.131}$$

Applying (A.131) to (A.130) results in

$$\|\boldsymbol{e}_i^{(t)}\| \geq \left(l(\boldsymbol{y}_i, \boldsymbol{g}_{i,L+1}(\boldsymbol{x}_i;\boldsymbol{W}^{(t)})) - l(\boldsymbol{y}_i, \boldsymbol{g}_{L+1}(\boldsymbol{x}_i;\boldsymbol{W}^*))\right)/O(\sqrt{L}). \tag{A.132}$$

Applying (A.132) to (A.129), we derive that

$$\begin{aligned}
&\mathcal{L}_{\text{convex}}(\boldsymbol{W}^{(t+1)}) - \mathcal{L}_{\text{convex}}(\boldsymbol{W}^*) \\
&\leq \mathcal{L}_{\text{convex}}(\boldsymbol{W}^{(t)}) - \mathcal{L}_{\text{convex}}(\boldsymbol{W}^*) \\
&\qquad - \Omega\left(\frac{(1-\alpha)^2}{(1+\alpha^2)}\frac{\delta\eta m}{n^2 dL}\right)\left(\mathcal{L}_{\text{convex}}(\boldsymbol{W}^{(t)}) - \mathcal{L}_{\text{convex}}(\boldsymbol{W}^*)\right)^2.
\end{aligned} \tag{A.133}$$

In order to derive a bound for the number of training epoch T that is required for $\mathcal{L}^{(T)} - \mathcal{L}^* < \epsilon$, for $t < T$, by assuming $\mathcal{L}^{(t)} - \mathcal{L}^* > \epsilon$, the above equation is bounded by

$$\begin{aligned}
&\mathcal{L}_{\text{convex}}(\boldsymbol{W}^{(t+1)}) - \mathcal{L}_{\text{convex}}(\boldsymbol{W}^*) \\
&\leq \left(1 - \epsilon\Omega\left(\frac{(1-\alpha)^2}{(1+\alpha^2)}\frac{\delta\eta m}{n^2 dL}\right)\right)\left(\mathcal{L}_{\text{convex}}(\boldsymbol{W}^{(t)}) - \mathcal{L}_{\text{convex}}(\boldsymbol{W}^*)\right).
\end{aligned}$$

The lower bound for m becomes $m \ln^4 m > \frac{1+\alpha^2}{(1-\alpha)^2}\Omega(n^6 L^{16} d/\delta^4)$ to ensure the same

perturbation bound in Lemma A.9. Denoting $\gamma := \left(1 - \epsilon\Omega\left(\frac{(1-\alpha)^2}{(1+\alpha^2)}\frac{\delta\eta m}{n^2 dL^2}\right)\right)$, it follows that

$$T = \frac{\ln\epsilon^{-1} + \ln(\mathcal{L}_{\text{convex}}(\boldsymbol{W}^{(0)}) - \mathcal{L}_{\text{convex}}(\boldsymbol{W}^*))}{\ln\gamma^{-1}} \leq O\left(\frac{n^2 dL}{\eta\delta m\epsilon}(\ln\epsilon^{-1} + \ln(n\sqrt{\ln m}))\right).$$

We follow the exact same steps in the proof of Lemma A.9 in Appendix A.2.6 and verify that when $m > \ln(\epsilon^{-1}\ln\epsilon^{-1})$, the probability that (A.133) holds for $T-$steps is at least $1 - e^{-\Omega(\ln m)}$. □

A.2.13 Proofs for a special class of datasets

This section includes the proof of Theorems A.2 and A.3 in Appendix A.2.11. We first present several lemmas and their proofs, then use these lemmas to prove those theorems.

Lemma A.12

Consider a dataset $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i\in[n]}$ satisfying Assumption A.1, where $m \geq \Omega(nd)$, then there exists a vector $\boldsymbol{u}_{l,j} \in \mathcal{B}_1^m \subset \mathbb{R}^m$, such that

$$|\langle \boldsymbol{u}_{l,j}, \boldsymbol{h}_{i,L-1}^{(0)}\rangle - (y_{i,j} - \hat{y}_{i,j})| \leq O(\lambda), \quad \text{for all} \quad i \in [n], j \in [d],$$
$$\text{with probability at least} \quad 1 - e^{-\Omega(\lambda^2 m)}.$$ □

Proof: We complete the proof by constructing the following unit vector $\boldsymbol{u}_j$:

$$\boldsymbol{u}_j = \frac{1}{\sqrt{2m}}\left(c_j(\sqrt{m/2}(\boldsymbol{W}_{L-1}^{(0)})_{1,\cdot}), c_j(\sqrt{m/2}(\boldsymbol{W}_{L-1}^{(0)})_{2,\cdot}), \cdots, c_j(\sqrt{m/2}(\boldsymbol{W}_{L-1}^{(0)})_{m,\cdot})\right)^T.$$

One can easily verify that $\boldsymbol{u}_j \in \mathcal{B}_1^m$ by using the fact that $|c_j| < 1$, which is guaranteed by the definition of the function class in (A.120).

The inner product of $\boldsymbol{u}_j$ and $\boldsymbol{h}_{i,L-1}^{(0)}$ is given by

$$\begin{aligned}
&\langle \boldsymbol{u}_j, \boldsymbol{h}_{i,L-1}^{(0)} \rangle \qquad (A.134)\\
&= \frac{1}{\sqrt{2m}} \sum_{k=1}^{m} c_j(\sqrt{m/2}(\boldsymbol{W}_{L-1}^{(0)})_{k,\cdot}) \mathcal{N}_{L-1}(\boldsymbol{x}; \boldsymbol{A}, \boldsymbol{W}_1^{(0)}, \cdots \boldsymbol{W}_{L-2}^{(0)}, \boldsymbol{W}_{L-1}^{(0)})\\
&= \frac{1}{\sqrt{2m}} \sum_{k=1}^{m} c_j(\sqrt{m/2}(\boldsymbol{W}_{L-1}^{(0)})_{k,\cdot}) \sqrt{\frac{2}{m}} \mathcal{N}_{L-1}(\boldsymbol{x}; \boldsymbol{A}, \boldsymbol{W}_1^{(0)}, \cdots \boldsymbol{W}_{L-2}^{(0)}, \sqrt{m/2}\boldsymbol{W}_{L-1}^{(0)})\\
&= \frac{1}{m} \sum_{k=1}^{m} c_j(\sqrt{m/2}(\boldsymbol{W}_{L-1}^{(0)})_{k,\cdot}) \mathcal{N}_{L-1}(\boldsymbol{x}; \boldsymbol{A}, \boldsymbol{W}_1^{(0)}, \cdots \boldsymbol{W}_{L-2}^{(0)}, \sqrt{m/2}\boldsymbol{W}_{L-1}^{(0)})
\end{aligned}$$

For simplicity, we denote that

$$Z_{k,i,j} := c_j(\sqrt{m/2}(\boldsymbol{W}_{L-1}^{(0)})_{k,\cdot}) \mathcal{N}_{L-1}(\boldsymbol{x}; \boldsymbol{A}, \boldsymbol{W}_1^{(0)}, \cdots \boldsymbol{W}_{L-2}^{(0)}, \sqrt{m/2}\boldsymbol{W}_{L-1}^{(0)}),$$

and above equation becomes

$$\langle \boldsymbol{u}_j, \boldsymbol{h}_{i,L-1}^{(0)} \rangle = \frac{1}{m} \sum_{k=1}^{m} Z_{k,i,j}. \qquad (A.135)$$

Noting that $\sqrt{m/2}(\boldsymbol{W}_{L-1}^{(0)})_{k,\cdot} \sim N(0,1)$, by using (A.120), it implies that $\mathbb{E}_{\boldsymbol{W}^{(0)}} Z_{k,i,j}((\boldsymbol{W}_{L-1})_{k,\cdot}^{(0)}) = f_j(\boldsymbol{x}_i)$.

Since $|c_j(\cdot)| < 1$ is bounded, $Z_{k,i,j}$ is a sub-Gaussian random variable, therefore we can apply Hoeffding's inequality and conclude that

$$\left| \frac{1}{m} \sum_{k=1}^{m} Z_{k,i,j} - f_j(\boldsymbol{x}_i) \right| \leq \lambda, \quad \text{with probability at least } \; 1 - e^{-\Omega(\lambda^2 m)}. \qquad (A.136)$$

Using (A.135), (A.136) and Assumption A.1, it follows that with probability at least

$1-(nd)e^{-\Omega(\lambda^2 m)}$

$$|\langle \boldsymbol{u}_{l,j}, \boldsymbol{h}_{i,L-1}^{(0)}\rangle - (y_{i,j}-\hat{y}_{i,j})| \leq |\langle \boldsymbol{u}_{l,j}, \boldsymbol{h}_{i,L-1}^{(0)}\rangle - f_j(\boldsymbol{x}_i)| + |f_j(\boldsymbol{x}_i) - (y_{i,j}-\hat{y}_{i,j})|$$
$$\leq 2\lambda. \text{ for all } \ i\in[n],\ j\in[d].$$

We conclude the Lemma by noting that the probability is at least $1-e^{-\Omega(\lambda^2 m)}$ when $m \geq \Omega(nd)$. □

Lemma A.13

Under Assumption A.1, when $m \geq \Omega(nd)$, the lower bound for the gradient of the loss function becomes

$$\|\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W}^{(0)})\|_F^2 \geq \Omega\left(\frac{(1-\alpha)^2}{1+\alpha^2}\frac{m}{d^2}\right)\mathcal{L}(\boldsymbol{W}^{(0)}), \quad \text{with probability at least } \ 1-e^{-\Omega(m)}.$$

Proof: We first note that by definition, $\|\nabla_{\boldsymbol{W}}\mathcal{L}\|_F \geq \|\nabla_{\boldsymbol{W}_L}\mathcal{L}\|$. Then by definition of matrix $F-$norm and (A.46), it follows that

$$\|\nabla_{\boldsymbol{W}_L}\mathcal{L}(\boldsymbol{W}^{(0)})\|_F^2 = \sum_{k=1}^{m}\left\|\sum_{i=1}^{n}(\boldsymbol{G}_{i,L}(\boldsymbol{e}_i;\boldsymbol{W}^{(0)}))_{k,\cdot}\right\|_2^2.$$

We write that the $k-$th row of the matrix $\sum_{i=1}^{n}\boldsymbol{G}_{i,L}(\boldsymbol{e}_i,\boldsymbol{W}^{(0)})$ by

$$\left\|\sum_{i=1}^{n}(G_{i,L}(\boldsymbol{e}_i;\boldsymbol{W}^{(0)}))_{k,\cdot}\right\|_2^2 = \left\|\sum_{i=1}^{n}\boldsymbol{B}_{k,\cdot}^T\boldsymbol{e}_i D_{i,L,kk}\boldsymbol{h}_{i,L-1}\right\|_2^2. \tag{A.137}$$

Using the vector $\boldsymbol{u}_j \in \mathcal{B}_1^m$ chosen in Lemma A.12, and denoting $\boldsymbol{u} := \frac{1}{d}\sum_{j=1}^d \boldsymbol{u}_j$, we note that $\|\boldsymbol{u}\| \leq 1$. Thus we conclude that

$$\begin{aligned}
&\left\|\sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T \boldsymbol{e}_i D_{i,L,kk} \boldsymbol{h}_{i,L-1}\right\|_2^2 \geq \left|\left\langle \sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T \boldsymbol{e}_i D_{i,L,kk} \boldsymbol{h}_{i,L-1}, \boldsymbol{u}\right\rangle\right|^2 \\
&= \left|\sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T \boldsymbol{e}_i D_{i,L,kk} \left\langle \boldsymbol{h}_{i,L-1}, \boldsymbol{u}\right\rangle\right|^2 \\
&= \frac{1}{1+\alpha^2}\left|\sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T \boldsymbol{e}_i (\alpha + (1-\alpha) 1_{\boldsymbol{h}_{i,L,k}>0}) \left\langle \boldsymbol{h}_{i,L-1}, \boldsymbol{u}\right\rangle\right|^2 \\
&= \frac{1}{1+\alpha^2}\left|(1-\alpha)\sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T \boldsymbol{e}_i 1_{\boldsymbol{h}_{i,L,k}>0} \left\langle \boldsymbol{h}_{i,L-1}, \boldsymbol{u}\right\rangle + \alpha \sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T \boldsymbol{e}_i \left\langle \boldsymbol{h}_{i,L-1}, \boldsymbol{u}\right\rangle\right|^2
\end{aligned} \tag{A.138}$$

By Jensen's inequality, we note that the expectation of (A.138) becomes

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{h}_{L-1}}\left|(1-\alpha)\sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T \boldsymbol{e}_i 1_{\boldsymbol{h}_{i,L,k}>0} \left\langle \boldsymbol{h}_{i,L-1}, \boldsymbol{u}\right\rangle + \alpha \sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T \boldsymbol{e}_i \left\langle \boldsymbol{h}_{i,L-1}, \boldsymbol{u}\right\rangle\right|^2 \\
&\geq \left|\mathbb{E}_{\boldsymbol{h}_{L-1}}\left((1-\alpha)\sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T \boldsymbol{e}_i 1_{\boldsymbol{h}_{i,L,k}>0} \left\langle \boldsymbol{h}_{i,L-1}, \boldsymbol{u}\right\rangle + \alpha \sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T \boldsymbol{e}_i \left\langle \boldsymbol{h}_{i,L-1}, \boldsymbol{u}\right\rangle\right)\right|^2.
\end{aligned} \tag{A.139}$$

Since $1_{\boldsymbol{h}_{i,L,k}>0}$ is independent with $\boldsymbol{h}_{i,L-1}$, for any integer N, the conditional expectation can be given as

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{h}_{L-1}}\left(\sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T \boldsymbol{e}_i 1_{\boldsymbol{h}_{i,L,k}>0} \left\langle \boldsymbol{h}_{i,L-1}, \boldsymbol{u}\right\rangle \,\middle|\, \sum 1_{\boldsymbol{h}_{i,L,k}>0} = N\right) \\
&= \frac{N}{n}\mathbb{E}_{\boldsymbol{h}_{L-1}}\left(\sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T \boldsymbol{e}_i \left\langle \boldsymbol{h}_{i,L-1}, \boldsymbol{u}\right\rangle\right)
\end{aligned} \tag{A.140}$$

Moreover, since $1_{\boldsymbol{h}_{i,L,k}>0}$ a Bernoulli random variable $B(0.5)$, when $n > 100$, using an

approximation of the probability by the central limit theorem, we know that

$$\sum_i 1_{\boldsymbol{h}_{i,L,k}>0} > n/2 + \sqrt{n}, \quad \text{with probability at least } 0.1, \tag{A.141}$$

$$\sum_i 1_{\boldsymbol{h}_{i,L,k}>0} < n/2 - \sqrt{n}, \quad \text{with probability at least } 0.1. \tag{A.142}$$

To find a lower bound for (A.139), we consider two cases for the second term, when $|\mathbb{E}_{\boldsymbol{h}_{L-1}}\alpha\sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T\boldsymbol{e}_i\langle\boldsymbol{h}_{i,L-1,\boldsymbol{u}}\rangle| > \frac{n}{2}|\mathbb{E}_{\boldsymbol{h}_{L-1}}(\boldsymbol{B}_{k,\cdot}^T\boldsymbol{e}_i\langle\boldsymbol{h}_{i,L-1},\boldsymbol{u}\rangle)|$, then by (A.142) and (A.140), we know that with probability at least 0.1 that

$$\begin{aligned}&\left|\mathbb{E}_{\boldsymbol{h}_{L-1}}\left((1-\alpha)\sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T\boldsymbol{e}_i 1_{\boldsymbol{h}_{i,L,k}>0}\langle\boldsymbol{h}_{i,L-1},\boldsymbol{u}\rangle + \alpha\sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T\boldsymbol{e}_i\langle\boldsymbol{h}_{i,L-1},\boldsymbol{u}\rangle\right)\right| \\ &\geq \frac{1}{\sqrt{n}}\left|\mathbb{E}_{\boldsymbol{h}_{L-1}}\left(\sum_{i=1}^n(1-\alpha)\boldsymbol{B}_{k,\cdot}^T\boldsymbol{e}_i\langle\boldsymbol{h}_{i,L-1},\boldsymbol{u}\rangle\right)\right|.\end{aligned} \tag{A.143}$$

Using similar argument, when $|\mathbb{E}_{\boldsymbol{h}_{L-1}}\alpha\sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T\boldsymbol{e}_i\langle\boldsymbol{h}_{i,L-1,\boldsymbol{u}}\rangle| \leq \frac{n}{2}|\mathbb{E}_{\boldsymbol{h}_{L-1}}(\boldsymbol{B}_{k,\cdot}^T\boldsymbol{e}_i\langle\boldsymbol{h}_{i,L-1},\boldsymbol{u}\rangle)|$, by using (A.141), it also follows that with probability at least 0.1 that (A.143) holds. Thus we conclude that

$$\begin{aligned}&\left|\mathbb{E}_{\boldsymbol{h}_{L-1}}\left((1-\alpha)\sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T\boldsymbol{e}_i 1_{\boldsymbol{h}_{i,L,k}>0}\langle\boldsymbol{h}_{i,L-1},\boldsymbol{u}\rangle + \alpha\sum_{i=1}^n \boldsymbol{B}_{k,\cdot}^T\boldsymbol{e}_i\langle\boldsymbol{h}_{i,L-1},\boldsymbol{u}\rangle\right)\right| \\ &\geq \frac{1}{\sqrt{n}}\left|\mathbb{E}_{\boldsymbol{h}_{L-1}}\left(\sum_{i=1}^n(1-\alpha)\boldsymbol{B}_{k,\cdot}^T\boldsymbol{e}_i\langle\boldsymbol{h}_{i,L-1},\boldsymbol{u}\rangle\right)\right|, \quad \text{with probability at least } 0.1.\end{aligned} \tag{A.144}$$

Combining (A.138), (A.139), and (A.144), it follows that

$$\mathbb{E}_{\boldsymbol{h}_{L-1}} \left\| \sum_{i=1}^{n} \boldsymbol{B}_{k,\cdot}^T \boldsymbol{e}_i D_{i,L,kk} \boldsymbol{h}_{i,L-1} \right\| \geq \frac{(1-\alpha)^2}{n(1+\alpha^2)} \left| \mathbb{E}_{\boldsymbol{h}_{L-1}} \sum_{i=1}^{n} \boldsymbol{B}_{k,\cdot}^T \boldsymbol{e}_i \langle \boldsymbol{h}_{i,L-1}, \boldsymbol{u} \rangle \right|^2, \quad \text{with probability at least } 0.1. \tag{A.145}$$

By using the lower bound of $|\boldsymbol{B}_{\cdot,k}^T \boldsymbol{a}|$ derived in (A.59), we obtain that with at least a constant probability $p_0 := 1 - \exp(-\Omega(1))$

$$\left| \mathbb{E}_{\boldsymbol{h}_{L-1}} \sum_{i=1}^{n} \boldsymbol{B}_{k,\cdot}^T \boldsymbol{e}_i \langle \boldsymbol{h}_{i,L-1}, \boldsymbol{u} \rangle \right|^2 = \left| \boldsymbol{B}_{k,\cdot}^T \left(\mathbb{E}_{\boldsymbol{h}_{L-1}} \sum_{i=1}^{n} \boldsymbol{e}_i \langle \boldsymbol{h}_{i,L-1}, \boldsymbol{u} \rangle \right) \right|^2 \geq \sum_{j=1}^{d} \left| \mathbb{E}_{\boldsymbol{h}_{L-1}} \sum_{i=1}^{n} \boldsymbol{e}_{i,j} \langle \boldsymbol{h}_{i,L-1}, \boldsymbol{u} \rangle \right|^2. \tag{A.146}$$

We thus conclude that by (A.137), Hoeffding inequality, (A.145) and (A.146), with probability at least $1 - e^{-\Omega(m)}$

$$\sum_{k=1}^{m} \left\| \sum_{i=1}^{n} (\boldsymbol{G}_{i,L}(\boldsymbol{e}_i; \boldsymbol{W}^{(0)}))_{k,\cdot} \right\| = \sum_{k=1}^{m} \mathbb{E}_{\boldsymbol{h}_{L-1}} \left\| \sum_{i=1}^{n} \boldsymbol{B}_{k,\cdot}^T \boldsymbol{e}_i D_{i,L,kk} \boldsymbol{h}_{i,L-1} \right\| \geq \frac{0.1 p_0 m}{2} \frac{(1-\alpha)^2}{n(1+\alpha^2)} \sum_{j=1}^{d} \left| \mathbb{E}_{\boldsymbol{h}_{L-1}} \sum_{i=1}^{n} \boldsymbol{e}_{i,j} \langle \boldsymbol{h}_{i,L-1}, \boldsymbol{u} \rangle \right|^2. \tag{A.147}$$

Using (A.147), Assumption A.1 and the fact that $\mathbb{E}\boldsymbol{e}_{i,j}\boldsymbol{e}_{i,j'} = 0$ if $j \neq j'$, when

$\mathcal{L} > 1$ imply

$$\sum_{j=1}^{d}\left|\mathbb{E}_{\boldsymbol{h}_{L-1}}\sum_{i=1}^{n}\boldsymbol{e}_{i,j}\langle\boldsymbol{h}_{i,L-1},\boldsymbol{u}\rangle\right| = \frac{1}{d}\left|\mathbb{E}\sum_{i=1}^{n}\boldsymbol{e}_{i,j}\left(\langle\boldsymbol{h}_{i,L-1},\boldsymbol{u}_j\rangle - y_{i,j} + \hat{y}_{i,j} + y_{i,j} - \hat{y}_{i,j}\right)\right|$$
$$\geq \frac{1}{d}\sum_{j=1}^{d}\left|\sum_{i=1}^{n}\boldsymbol{e}_{i,j}\left(y_{i,j} - \hat{y}_{i,j}\right)\right| - \frac{1}{d}\sum_{j=1}^{d}\left|\sum_{i=1}^{n}\boldsymbol{e}_{i,j}\left(\langle\boldsymbol{h}_{i,L-1},\boldsymbol{u}_j\rangle - (y_{i,j} - \hat{y}_{i,j})\right)\right|$$
$$\geq \frac{1}{d}\sum_{j=1}^{d}\left|\sum_{i=1}^{n}\boldsymbol{e}_{i,j}^2\right| - \frac{\lambda}{d}\sum_{j=1}^{d}\left|\sum_{i=1}^{n}\boldsymbol{e}_{i,j}\right| \geq \frac{1}{d}\mathcal{L} - \lambda\frac{\sqrt{n}}{\sqrt{d}}\mathcal{L} \geq \frac{1}{2d}\mathcal{L}.$$

Applying (A.147) and (A.148), we conclude that

$$\sum_{k=1}^{m}\left\|\sum_{i=1}^{n}\boldsymbol{B}_{k,\cdot}^T\boldsymbol{e}_i D_{i,L,kk}\boldsymbol{h}_{i,L-1}\right\|_2^2 \geq \frac{m}{nd^2}\frac{(1-\alpha)^2}{1+\alpha^2}\mathcal{L}^2, \text{ with probability at least } 1-e^{-\Omega(m)}. \tag{A.148}$$

Considering at initial parameter $\boldsymbol{W}^{(0)}$,

$$\mathcal{L}(\boldsymbol{W}^{(0)}) = \sum_{i=1}^{n}\|\boldsymbol{y}_i - \boldsymbol{B}\boldsymbol{h}_{i,L}\|^2 = \sum_{i=1}^{n}\|\boldsymbol{y}_i\|^2 + \|\boldsymbol{B}\boldsymbol{h}_{i,L}\|^2 - 2\langle\boldsymbol{y}_i,\boldsymbol{B}\boldsymbol{h}_{i,L}\rangle.$$

Using the fact that $\mathbb{E}\langle\boldsymbol{y}_i,\boldsymbol{B}\boldsymbol{h}_{i,L}\rangle = 0$ and Lemma A.1, we establish a lower bound for $\mathcal{L}(\boldsymbol{W}^{(0)})$,

$$\mathcal{L}(\boldsymbol{W}^{(0)}) \geq \Omega(n). \tag{A.149}$$

Applying the definition of the gradient norms, and (A.148) with the lower bound of

$\mathcal{L}(\boldsymbol{W}^{(0)})$ in (A.149), we conclude that

$$\begin{aligned}\|\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W}^{(0)})\|_F^2 &\geq \sum_{k=1}^{m}\left\|\sum_{i=1}^{n}\boldsymbol{B}_{k,\cdot}^T\boldsymbol{e}_i D_{i,L,kk}\boldsymbol{h}_{i,L-1}\right\|_2^2 \\ &\geq \Omega\left(\frac{(1-\alpha)^2}{1+\alpha^2}\frac{m}{d^2}\right)\mathcal{L}(\boldsymbol{W}^{(0)}),\end{aligned}$$

with probability at least $1-e^{-\Omega(m)}$.

□

Lemma A.14

Assume the setup of §2.2 and the dataset satisfy Assumption A.1, when $\|\boldsymbol{W}-\boldsymbol{W}^{(0)}\| < \omega < O(\frac{1}{d^{3/2}L^6\ln^{3/2}m})$, with probability at least $1-e^{-\Omega(m)}$,

$$\|\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W})\| \geq \Omega\left(\frac{(1-\alpha)^2}{1+\alpha^2}\frac{m}{d^2}\right)\mathcal{L}(\boldsymbol{W}).$$

Proof: For $\boldsymbol{W}$ such that $\|\boldsymbol{W}^{(0)}-\boldsymbol{W}\| < \omega$, we use the same argument in the proof of Lemma 2.2. It is straightforward to verify that $\omega^{2/3}L^4 m\ln m/d$ (this is the right hand side of (A.75)) is smaller than $O(m/d^2)$ by plugging in $\omega < O(\frac{1}{d^{3/2}L^6\ln^{3/2}m})$. We conclude that for $\boldsymbol{W}$ such that $\|\boldsymbol{W}-\boldsymbol{W}^{(0)}\| < \omega$, with probability at least $1-e^{-\Omega(m)}$

$$\|\nabla_{\boldsymbol{W}}\mathcal{L}(\boldsymbol{W})\| \geq \Omega\left(\frac{(1-\alpha)^2}{1+\alpha^2}\frac{m}{d^2}\right)\mathcal{L}(\boldsymbol{W}). \tag{A.150}$$

□

Compared to the conclusion in Lemma 2.2 with Lemma A.14, the lower bound is improved by a factor of $\frac{n}{\delta d}$ when the dataset satisfies Assumption A.1.

Proof: [Proof of Theorem A.2] We follow the exact same proof of Theorem 2.1, with the lower bound of the gradient given in Lemma A.14. It is straight-forward to derive

the same inequality (2.16), and by the lower bound in (A.150), we obtain that with probability $1 - e^{-\Omega(m)}$

$$\mathcal{L}^{(t+1)} \leq \left(1 - \Omega(\frac{(1-\alpha)^2}{1+\alpha^2}\frac{\eta m}{d^2})\right)\mathcal{L}^{(t)}.$$

We conclude the theorem by verifying that during the training process, we always have $|\boldsymbol{W}^{(0)} - \boldsymbol{W}^{(0)}| < \omega < O(\frac{1}{d^{3/2}L^6 \ln^{3/2} m})$, which satisfies the condition for both Lemmas A.14 and 2.1. Following the same argument that derives (A.96) and using the lower bound in (A.150), we achieve that

$$\|\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(0)}\| \leq \frac{\sqrt{1+\alpha^2}}{1-\alpha}\Omega\left(\frac{d}{\sqrt{m}}\right)\sqrt{\mathcal{L}^{\boldsymbol{W}^{(0)}}}. \tag{A.151}$$

By further applying (A.93), we verify that when $m/\ln^4 m > \frac{1+\alpha^2}{(1-\alpha)^2}\Omega(d^5 n L^{12})$, we can derive that

$$\|\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(0)}\| \leq O\left(\frac{1}{d^{3/2}L^6 \ln^{3/2} m}\right). \tag{A.152}$$

□

Proof: [Proof of Theorem A.3] The universal bound of $\|\boldsymbol{W}^{(t)} - \boldsymbol{W}^{(0)}\|$ is improved as shown in (A.151) and (A.152). Then, we can conclude the theorem by following exactly the same as the proof of Theorem 2.3 which is shown in Appendix A.2.9. □

A.3 Supplemental numerical experiments and details for the previous experiments

Section A.3.1 provides the full details of implementation for both the previous and the new experiments. Section A.3.2 describes new numerical experiments.

A.3.1 Details of Implementation

We provide some general implementation details and also details specific to the different datasets. Two datasets are new to this section. For completeness, we repeat some information that was provided in Section 2.5.1.

General implementation details: Throughout the numerical experiments, we applied Algorithm 1 to initialize the parameters of the neural networks. In order to implement the rescaled leaky ReLU as given in (2.2), we introduce a MULTIPLIER(c) layer, which simply does element-wise multiplication with a given constant c. By combining Leaky ReLU(α) and MULTIPLIER $\left(1/\sqrt{1+\alpha^2}\right)$, we replicate the rescaled Leaky ReLU with parameter α.

In the experiments, we train the NN on the training set and report the error on a reserved testing set (we view it as an approximation for the generalization error). For the synthetic dataset, we generated additional 500 synthetic data points for the testing set. For the real dataset, we performed a standard training-testing split for each dataset.

Synthetic dataset: The architecture of the NNs that we used for the synthetic dataset is shown in Table A.1. We generate 1,000 data points as the training dataset and 500 data points as the testing dataset (following the model and sampling procedure described in the main text). We train the NN with GD and a learning rate of 10^{-4}.

California housing: We use an updated version of the California housing dataset, which can be downloaded from Kaggle (`https://www.kaggle.com/datasets/camnugent/california-housing-prices`) and is licensed by CC0. This dataset was drawn from the 1990 U.S. Census and contains 20,640 observations with 10 different characteristics. Nine of them are numerical ones (e.g., the median income for households and the median value of the houses within a block) and are given in the original dataset Pace and Barry (1997). An additional categorical characteristic is the ocean proximity. Borisov et al. (2022) used

this dataset as a benchmark for regression, where one needs to predict the value of the house given the other numerical characteristics. The last characteristic, which is the median house value for households within a block, provides labels for the dependent variable. We follow a similar setting of regression, but we also use the categorical feature of the updated dataset. We standardize the 9 numerical characteristics using the respective means and standard deviations of the training data. We generated a one-hot coding vector for the feature "ocean proximity", including 5 categories, <1H OCEAN, INLAND, NEAR OCEAN, NEAR BAY, and ISLAND. In total, the input $\boldsymbol{x}$ is a 13−dimensional vector. The training data contains 15,480 data points and the testing data contains 5,160 data points.

We built NNs to predict the median housing value in the dataset. The architecture of the NNs is given in Table A.2. We applied Algorithm 3 with a batch size of 512 and a learning rate of 10^{-5} to train the NNs.

Table A.1: Architecture of the NNs with Leaky ReLU parameter α used for the synthetic dataset.

Layer		Parameter
	Linear	(5, 5000)
Repeat 5 times	Linear	(5000, 5000)
	Leaky Relu	α
	Multipler	$1/(1+\alpha^2)^{1/2}$
	Linear	(5000, 1)

MNIST: We used the MNIST dataset of 28×28 images of handwritten digits in order to classify handwritten digits. This dataset is licensed by CC BY-SA 3.0. We flattened each image to a vector of length 784. We randomly sample 2,100 data points from the training set of MNIST as our training set, and use the rest as our testing set. We normalized the training data with 0.5 mean and 0.5 standard deviation. We

Table A.2: Architecture of the NNs with Leaky ReLU parameter α used for California housing.

Layer		Parameter
	Linear	$(13, 5000)$
Repeat 7 times	Linear	$(5000, 5000)$
	Leaky Relu	α
	Multipler	$1/(1+\alpha^2)^{1/2}$
	Linear	$(5000, 1)$

Table A.3: Architecture of the neural networks with α Leaky ReLU parameter used for MNIST.

Layer	Parameter
Linear	(784, 2000)
Linear	(2000, 2000)
Leaky ReLU	α
Multiplier	$\frac{1}{\sqrt{1+\alpha^2}}$
Linear	(2000, 2000)
Leaky ReLU	α
Multiplier	$\frac{1}{\sqrt{1+\alpha^2}}$
Linear	(2000, 10)

applied SGD with batch size 64 and a learning rate of 10^{-3}. The architecture of the NN is presented in Table A.3 and we use leaky ReLUs with $\alpha \in \{-2, -1, 0, 0.01, 0.05\}$. MNIST was also used to test the Transformer networks. In this case, we normalized the training set with 0.1307 mean and 0.3081 standard deviation. Furthermore, we used the Vision Transformer (ViT) (Dosovitskiy et al., 2020) architecture and applied SGD with batch size 100 and learning rate 10^{-4}. The details of this architecture are shown in Table A.4.

F-MNIST: We used the F-MNIST dataset of 28×28 images of clothing or accessory items for classification. This dataset is licensed by MIT. We flattened

Table A.4: Architecture of the Transformer neural networks with α Leaky ReLU parameter used for IMDB movie reviews.

Layer	Parameter
Positional Embedding	(49, 64)
Transformer Encoder	head dim = 64, output dim = 64, number of heads = 8 number of layers = 6 mlp dim = 8192
Linear	(64,8192)
Leaky ReLU	α
Multiplier	$\frac{1}{\sqrt{1+\alpha^2}}$
Linear	(8192, 10)

Table A.5: Architecture of the neural networks with α Leaky ReLU parameter with width of m and depth of L used for Fashion MNIST.

Layer		Parameter
	Linear	$(784, m)$
Repeat L times	Linear	(m, m)
	Leaky Relu	α
	Multipler	$1/(1+\alpha^2)^{1/2}$
	Linear	$(m, 10)$

each image to vectors of length 784. We randomly sample 3000 data points from the training set of F-MNIST as the training set, and use the rest for testing. We normalized the training data with 0.5 mean and 0.5 standard deviation. We applied SGD with batch size 64 and a learning rate of 10^{-5}. The architecture of the NN is presented in Table A.5 and we use leaky ReLUs with $\alpha \in \{-2, -1, 0, 0.01, 0.05\}$.

CIFAR-10: We used the CIFAR-10 dataset of 32×32 RGB images with 10 categories for classification. This dataset is licensed by MIT. We randomly sample

Table A.6: Architecture of the neural networks with α Leaky ReLU parameter used for CIFAR-10.

Layer	Parameter
CNN	conv3-512
Leaky ReLU	α
Multiplier	$\frac{1}{\sqrt{1+\alpha^2}}$
CNN	conv3-512
Leaky ReLU	α
Multiplier	$\frac{1}{\sqrt{1+\alpha^2}}$
Max Pooling	2×2
CNN	conv3-512
Leaky ReLU	α
Multiplier	$\frac{1}{\sqrt{1+\alpha^2}}$
CNN	conv3-512
Leaky ReLU	α
Multiplier	$\frac{1}{\sqrt{1+\alpha^2}}$
Max Pooling	2×2
Flatten	
Linear	(32,768, 512)
Leaky ReLU	α
Multiplier	$\frac{1}{\sqrt{1+\alpha^2}}$
Linear	(512, 10)

2560 data points from the training set of CIFAR-10 as our training set, and randomly sample 2560 data points from the testing set of CIFAR-10 as our testing set. We normalized the training data with 0.5 mean and 0.5 standard deviation. Then we applied SGD with batch size 64 and a learning rate of 10^{-6}. The architecture of the NN is presented in Table A.6.

IMDB movie reviews dataset: This is a dataset of highly popular movie review paragraphs and it is used for positive or negative sentiment classification (Maas et al., 2011). We downloaded it from the following URL: `http://ai.stanford.edu/~amaas/data/sentiment/`. We randomly sample 5,000 data points from the IMDB

Table A.7: Architecture of the neural networks with α Leaky ReLU parameter used for IMDB movie reviews dataset

Layer	Parameter
Embedding	(1000, 64)
LSTM	input dim = 64, hidden dim = 256, number of layers = 2
Dropout	0.3
Linear	(256,4096)
Leaky ReLU	α
Multiplier	$\frac{1}{\sqrt{1+\alpha^2}}$
Linear	(4096, 1)

movie reviews as the training dataset and use the rest as the testing dataset. We processed the data as follows. We first recorded the words that appeared at least once in the training dataset. For each word, a unique integer was assigned to index it. Then we mapped each paragraph to a vector, whose i-th entry is the assigned index of the i-th word of the paragraph. Finally, we padded each vector with zeros, so that each vector was of the same length. We used the zero-padded vectors as the input of our neural network. After preprocessing, we applied SGD with batch size 50 and learning rate 10^{-5} with an LSTM network, whose architecture is presented in Table A.7.

A.3.2 Additional numerical results

We describe here two additional experiments.

Additional experiments using MNIST and California housing: We ran the same experiments done in Section 2.5.1, but with MNIST and California housing. Figure A.1 demonstrates the training errors (top) and testing errors (bottom) for the two datasets. For MNIST (left) we used the cross entropy loss and for California

housing (right) the MSE loss. For the training errors, we note that the convergence rate is the fastest at $\alpha = -1$ for both datasets and this observation aligns with our theoretical prediction and the previous experiments. The testing errors are rather similar for different choices of α. For the California housing dataset, we note that $\alpha = -1$ achieves the smallest testing error at an early epoch (about $t = 20$), but the advantage is marginal compared to other α's. For MNIST, we note that $\alpha = -1$ gets to the same level of testing error as the other αs from a much larger initial testing error. We note that $\alpha = -1$ is not optimal for the testing error. This might happen because the number of samples and the depth are not sufficiently large enough.

Experiments with Long Short-Term Memory (LSTM) and Transformer networks: We ran the same experiment done in Section 2.5 on MNIST and IMDB with Transformer and LSTM networks, respectively. These architectures are described in Section A.3.1. Figure A.2 demonstrates the training errors (top) and testing errors (bottom). For MNIST (right) we used the negative log likelihood loss and for IMDB (left) we used the binary cross entropy loss. For both algorithms, the training errors converge fastest with $\alpha = -1$. This observation agrees with our theoretical findings and previous experiments. The testing error for IMDB decreases during the first 100 epochs and then increases for the rest of the training. This is because of severe overfitting that is due to the following property of IMDB: the training dataset is small (we used randomly sampled 5,000 data points to be able to deal with sufficiently large widths for overparameterization) compared to the input data dimension (we have 1,000 unique words). The testing error on MNIST, on the other hand, is also the lowest for $\alpha = -1$, but there's no overfitting phenomenon since MNIST is a simple dataset.

Dependence on m and L: We demonstrate the dependence of the training error on m and the testing error on L. We thus ran additional experiments on F-MNIST with different choices of L and m and $\alpha \in \{-2, -1, 0, 0.01, 0.05\}$. First, we fixed $L = 2$

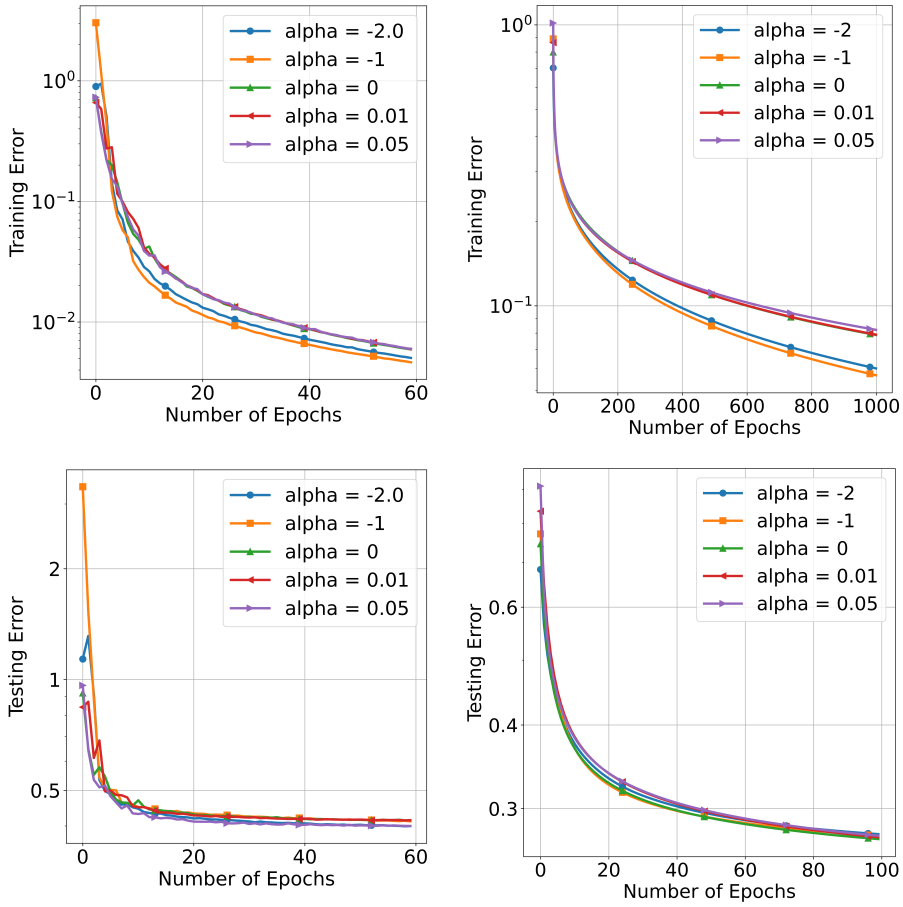

Figure A.1: Log-scale training and testing errors using different datasets and different α's. Left: cross entropy errors for MNIST; Right: MSE for California housing. Top row: training errors. Bottom row: testing errors.

and tested $m \in \{1,000, 2,000, 5,000, 10,000\}$. Next, we fixed $m = 5,000$ and tested $L \in \{2, 3, 4\}$. The architectures of thes NNs were given in Table A.5, but with the latter choices of m and L.

Figure A.3 shows the dependence of training errors on m and the dependence of testing errors on L. We note that the training error is monotonically decreasing w.r.t. the width m. We also note that the minimal training error is always achieved at $\alpha = -1$ for different choices of m. This matches our theoretical predictions. We note that the testing error is decreasing for $L \leq 4$. This aligns with equation (A.119)

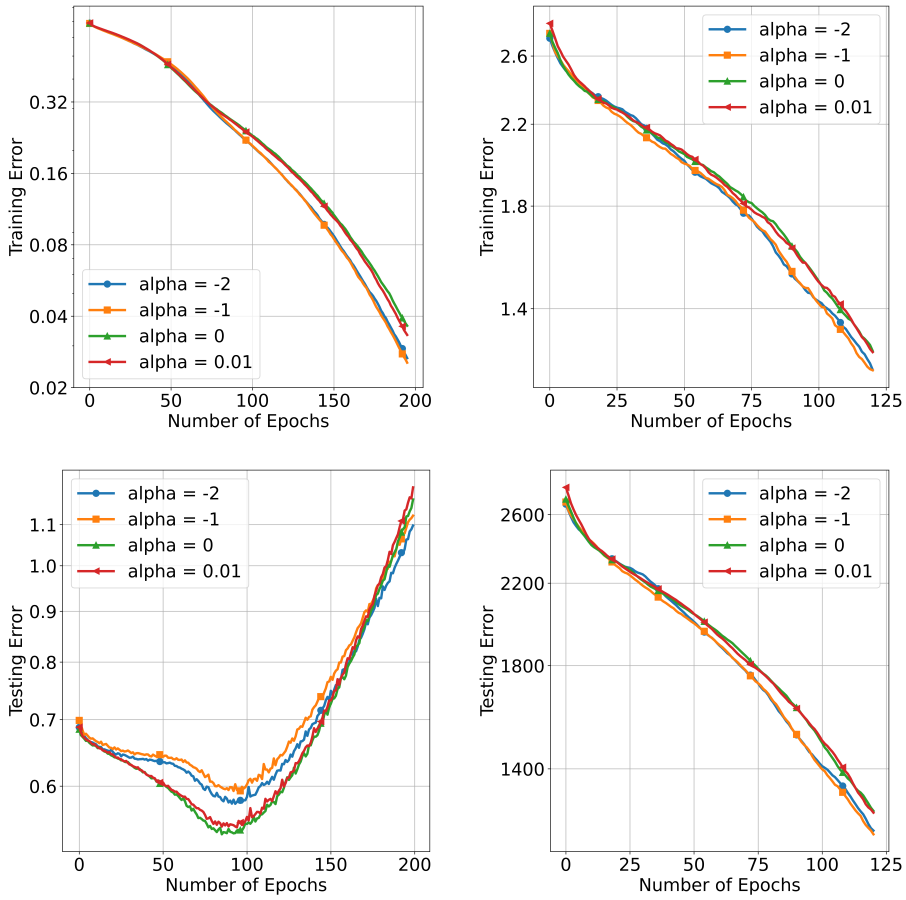

Figure A.2: Log-scale training and testing errors using different datasets and different α's. Left: binary entropy errors for IMDB; Right: negative log likelihood errors for Transformer on MNIST. Top row: training errors. Bottom row: testing errors.

when t is small. Moreover, when $L = 4$, we observe that the minimal testing error is achieved at $\alpha = -1$.

Results using the loss function given in (A.122)**:** We perform numerical experiments using the exponential loss function in the two datasets for the regression task, the synthetic dataset and the California housing dataset. Figure A.4 reports the results. We observe that $\alpha = -1$ achieves both the optimal training error and the optimal generalization error in the synthetic dataset. Furthermore, $\alpha = -1$ achieves optimal training error and second optimal generalization error for the California

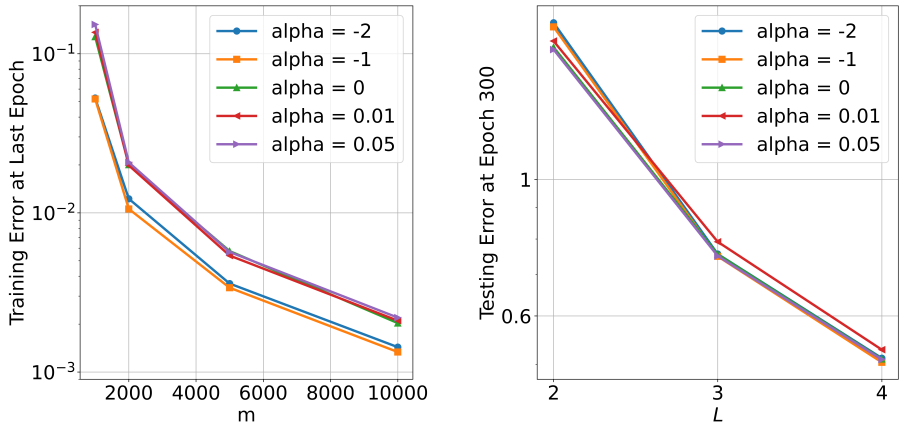

Figure A.3: Log-scale errors on F-MNIST with different α's. Left: training errors at the last epoch with $L = 2$ and different widths (ms); Right: testing errors at the epoch $t = 300$ with $m = 5000$ and different depths (Ls).

housing dataset, though the difference in the testing error is small.

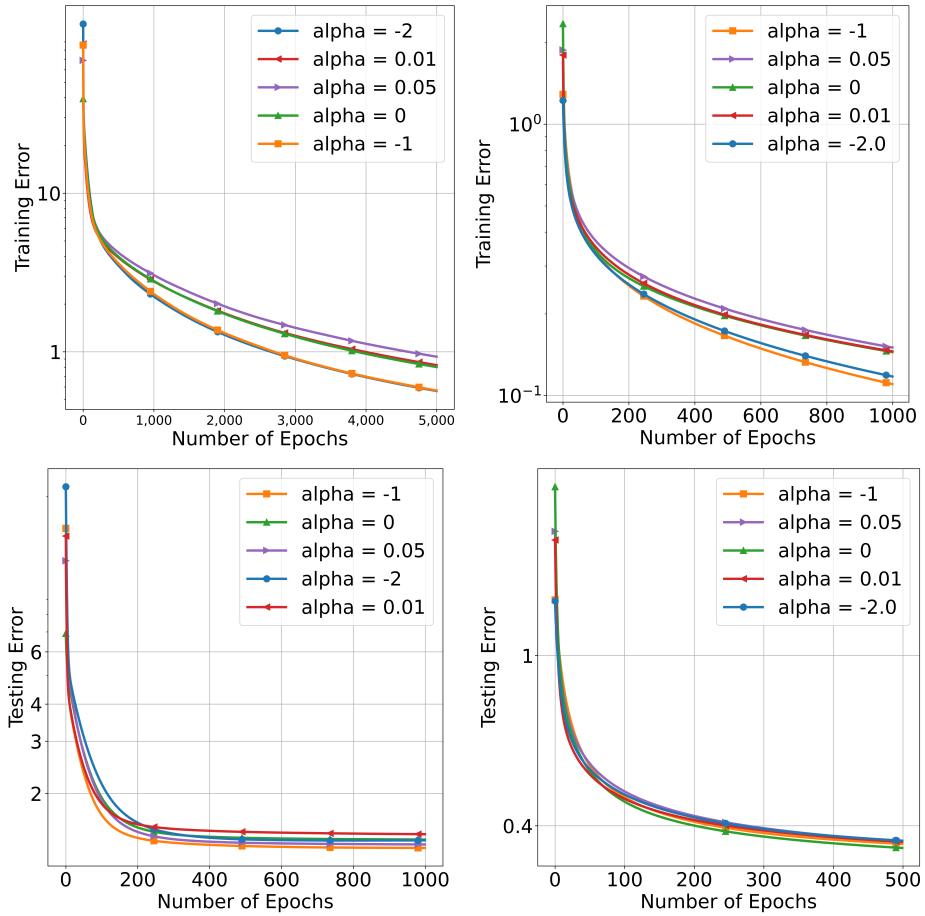

Figure A.4: Log-scaled mean square errors for the synthetic dataset with different α's using the loss function of (A.122). Left: MSE for the synthetic dataset. Right: MSE for California housing dataset. Top row: training errors. Bottom row: testing errors.

Appendix B

Supplementary Material for Chapter 3

We supplement the main text as follows: §B.1 includes more details on the unpooling layer; §B.2 discusses the tested codes; §B.3 describes implementation details of both the unpooling layer and the generative neural network; §B.4 carefully validates the theory stated in §3.3; §B.5 reports molecule generation results in a particular setting that aims to optimize specific chemical properties; and §B.6 reports some additional numerical results that supplement the ones in §3.4.

B.1 Additional Details of the Unpooling Layer

We describe in more details the unpooling layer mechanism. For completeness, we repeat details explained in §3.2.1.

Notations. For an unpooling layer U, the input featured graph is denoted by $\mathcal{G} = (V, E, \boldsymbol{X}, \boldsymbol{W})$, which is an undirected graph (V, E) with feature matrices$\boldsymbol{X}$ and $\boldsymbol{W}$ for nodes and edges respectively. More specifically, $V = [N] := \{1, 2, \cdots N\}$ are node indices, E is the edge set whose members are of the form $\{i, j\}$ where $i, j \in V$, and $M = |E|$ is the number of edges in the graph. $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ is the node features matrix whose i-th row $\boldsymbol{x}_i$ is a d-dimensional vector for node i. $\boldsymbol{W} \in \mathbb{R}^{M\times e}$ is the edge features matrix whose rows are $e-$dimensional vectors for edge features and we refer to the feature vector on edge $\{i, j\} \in E$ as $\boldsymbol{w}_{i,j}$.

Detailed Steps of the Unpooling Layer. We provide more details of the unpooling layer according to the following steps. Recall that the unpooling layer enlarges an input featured graph $\mathcal{G} = (V, E, \boldsymbol{X}, \boldsymbol{W})$ to a graph $\mathcal{G}^o = (V^o, E^o, \boldsymbol{Y}, \boldsymbol{U})$, while using the set of unchanged (or stable) nodes, I'_s, the set of unpooled nodes, I'_u, and the set of nodes that may be unpooled, I'_r.

Step 1. Generating nodes and node features. We first probabilistically determine the eventual set of nodes to be unpooled and consequently form the set of nodes in the output graph. We finally construct node features.

Step 1a. Determining nodes to be unpooled. We determine a set of nodes to be unpooled and a set of nodes to stay unchanged based on I'_s, I'_r and I'_u. For each node $i \in I'_r \subset V$, we determine the probability of being unpooled by

$$p_r(\boldsymbol{x}_i) = \text{MLP-}R(\boldsymbol{x}_i).$$

A node $i \in I'_r$ is unpooled if a randomly drawn uniform random variable $U_i \sim U[0, 1]$ is smaller than $p_r(\boldsymbol{x}_i)$. We form a set of nodes to be unpooled as

$$I_u = I'_u \cup \{i : i \in I'_r, U_i < p_r(\boldsymbol{x}_i)\}.$$

The rest nodes stay unchanged and form a set $I_s := V \setminus I_u$. The total log probability in this step is

$$\text{logP-R} = \sum_{i \in I'_r} \log \left(p_r(\boldsymbol{x}_i) \mathbb{1}_{U_i < p_r(\boldsymbol{x}_i)} + (1 - p_r(\boldsymbol{x}_i)) \mathbb{1}_{U_i \geq p_r(\boldsymbol{x}_i)} \right) \tag{B.1}$$

We remark that it is possible to have $I'_r = \emptyset$ (e.g. the unpooling layers used for the protein dataset), thus all the unpooling nodes are pre-determined. In this case, we obtain $I_u := I'_u$ and $I_s = I'_s$ without using MLP-R and we do not establish the log probability logP-R.

Step 1b. Unpooling nodes and constructing node features. From the previous step, we obtain I_s, a set of nodes to stay unchanged, and I_u, a set of nodes to be unpooled. We thus clarified the set V^o, where we note that $|V^o| = |I_s| + 2|I_u|$. Given two children nodes in V^o, we refer to the "subdivided" node in V as their parent node. We define a map f that assigns to each node in $i \in I_s$ the corresponding index, $f(i) \in V^o$, and to each node in $i \in I_u$ the set of two indices of its children nodes in V^o; in the latter case we denote $f(i) = \{f_1(i), f_2(i)\}$. The feature vectors of the nodes in V^o are obtained by an MLP for vertices, MLP-y. To produce different features for the children nodes $f_1(i)$ and $f_2(i)$, we apply MLP-y to two different pieces of the node feature vector $\boldsymbol{x}_i$. More specifically, we use the orthogonal projectors, P_{S_1} and P_{S_2}, onto two fixed subspaces, S_1 and S_2 in $\mathbb{R}^d$ (which are later specified in §B.3), as follows:

$$\begin{aligned} \boldsymbol{y}_{f(i)} &= \text{MLP-}y(P_{S_1}\boldsymbol{x}_i), && \text{for } \ i \in I_s; \\ \boldsymbol{y}_{f_j(i)} &= \text{MLP-}y(P_{S_j}\boldsymbol{x}_i), && \text{for } \ j = 1, 2, \ i \in I_u. \end{aligned}$$

One can also use three different MLP-y's for the static nodes and the two different children nodes.

Step 2. Building edges. Next, the unpooling layer takes the generated children nodes in V^o and their node features and builds edges for them. It first builds intra-links, which are edges within the pairs of children nodes; it then builds inter-links, which are edges between pairs of children nodes and images (according to the function f) of neighbors of their parent node (in I_u). At last, a single step ensures the connectivity of the output graph.

We also aggregate all log-probabilities in all three steps and use that in the policy gradient algorithm to train the layer and tune those probabilities for edge construction.

We start the edge generation process with $E^o = \emptyset$.

Step 2a. Building intra-links. We aim to probabilistically sample a set V_c of nodes in $I_u \subset V$ whose two children (in V^0) will be connected to each other by intra-links. We first initialize $V_c = \emptyset$. For each node $j \in I_u$ with feature vector $\boldsymbol{x}_j$ we add node j into V_c with a probability outputted by an MLP for intra-links, $p_c(j) = \text{MLP-IA}(\boldsymbol{x}_j)$. That is, we draw a uniform random variable, $U \sim U[0,1]$, and if $U < p_c(j)$, then $V_c = V_c \cup \{j\}$ and $\{f_1(j), f_2(j)\}$ is an intra-link, which is added to E^o; otherwise, V_c is unchanged and no intra-link is added.

We add all the intra-link to edge set in output graph

$$E^o = E^o \cup \big\{\{f_1(j), f_2(j)\}, \forall j \in V_c\big\}.$$

We track the log probabilities for later use as follows:

$$\text{logP-IA} := \sum_{j \in V_c} \ln(p_c(j)) + \sum_{j \notin V_c} \ln(1 - p_c(j)). \tag{B.2}$$

Step 2b. Finding a shared node for disconnected children. For disconnected children pairs, we designate a node that will connect to these children. In the next step these nodes and possibly additional ones will be connected to a subset of the children pairs. For each node $j \in I_u \setminus V_c$, we denote the set of all edges in E that are connected to j by $E_j = \big\{\{i_1, j\}, ...\{i_{m_j}, j\}\big\}$, and we calculate probabilities of selecting those edges as

$$\big(p_b(j, i_1), p_b(j, i_2), ...p_b(j, i_{m_j})\big) \propto \big(\text{MLP-C}(\boldsymbol{y}_{f_1(j)}, \boldsymbol{y}_{f_2(j)}, \boldsymbol{w}_{\{j,i_1\}}, \boldsymbol{x}_{i_1}),$$
$$\text{MLP-C}(\boldsymbol{y}_{f_1(j)}, \boldsymbol{y}_{f_2(j)}, \boldsymbol{w}_{\{j,i_2\}}, \boldsymbol{x}_{i_2}), ..., \text{ MLP-C}(\boldsymbol{y}_{f_1(j)}, \boldsymbol{y}_{f_2(j)}, \boldsymbol{w}_{\{j,i_{m_j}\}}, \boldsymbol{x}_{i_{m_j}})\big).$$

We draw from uniform distribution $U \sim U[0,1]$ and let

$$b_j = \begin{cases} i_1, & \text{if } U < p_b(j, i_1); \\ i_k, & \text{if } U \in \left[\sum_{l=1}^{k-1} p_b(j, i_l), \sum_{l=1}^{k} p_b(j, i_l)\right); \\ i_{m_j}, & \text{otherwise.} \end{cases}$$

Then, we define $N_{\{b_j,j\}}(j) := \{f_1(j), f_2(j)\}$.

In the next step, each edge $\{i,j\}$ in E gives rise to adding edges between $N_{\{i,j\}}(i)$ and $N_{\{i,j\}}(j)$ in $\mathcal{G}^o$. Therefore, our approach ensures that $f_1(j)$ and $f_2(j)$ both connect to $N_{\{b_j,j\}}(b_j)$ so $\mathcal{G}^o$ is connected.

We track the log-probabilities from this step that ensures connectivity as follows:

$$\text{logP-C} := \sum_{j \in I_u, j \notin V_c} \ln(p_b(j, b_j)) \tag{B.3}$$

Step 2c. Building inter-links. For each edge $\{i,j\} \in E$ and node j, we first generate a set of nodes $N_{\{i,j\}}(j) \subset V^o$ as follows. For $j \in I_u \setminus V_c$ and $i = b_j$, we have already defined $N_{\{i,j\}}(j)$ in the above step. If $j \in I_s$, we let $N_{\{i,j\}}(j) = \{f(j)\}$. If $j \in I_u$, we first calculate probabilities from an MLP of intra-links as follows: $(p_1(i,j), p_2(i,j)) = \text{MLP-IE}(\boldsymbol{y}_{f_1(j)}, \boldsymbol{y}_{f_2(j)}, \boldsymbol{w}_{\{i,j\}}, \boldsymbol{x}_i)$. Then, we draw a random variable $U \sim U(0,1)$ and let

$$N_{\{i,j\}}(j) = \begin{cases} \{f_1(j)\}, & \text{if } U < p_1(i,j); \\ \{f_1(j), f_2(j)\}, & \text{if } U \geq p_1(i,j) + p_2(i,j); \\ \{f_2(j)\}, & \text{otherwise.} \end{cases}$$

We similarly form $N_{\{i,j\}}(i)$ by swapping the j and i nodes.

The log probability for each edge-node pair $(\{i,j\},j)$ is

$$\text{logP-IE}(\{i,j\},j) := \Big(\mathbb{1}_{N_{\{i,j\}}(j)=\{f_1(j)\}} \ln(p_1(i,j)) + \mathbb{1}_{N_{\{i,j\}}(j)=\{f_2(j)\}} \ln(p_2(i,j)) \tag{B.4}$$

$$+ \mathbb{1}_{N_{\{i,j\}}(j)=\{f_1(j),f_2(j)\}} \ln(1-p_1(i,j)-p_2(i,j))\Big).$$

Lastly, we add to E^o all inter-links, i.e., all possible edges that connect nodes in $N_{\{i,j\}}(i)$ and $N_{\{i,j\}}(j)$:

$$E^o = E^o \cup \big\{\{k,l\},\ \forall k \in N_{\{i,j\}}(i),\ l \in N_{\{i,j\}}(j)\big\}. \tag{B.5}$$

Let $A_j := \{j \in V_c\}$ and $A_{i,j} := \{j \in I_u,\ j \notin V_c,\ i \neq b_j\}$. The cumulative log-probability for inter-links outputted by MLP-IE is

$$\text{logP-IE} = \sum_{\{i,j\}\in E} \mathbb{1}_{A_j \cup A_{i,j}} \text{logP-IE}(\{i,j\},j) + \mathbb{1}_{A_i \cup A_{j,i}} \text{logP-IE}(\{i,j\},i). \tag{B.6}$$

Step 2d. Building additional edges between children node pairs. We insert additional edges between children nodes in order to assure the expressivity of the output graph (this will be clarified in the proof of Theorem 2). Let $E_u = \{\{i,j\} \in E : i \in I_u, j \in I_u\} \subset E$ denote the collection of edges whose two ends are both unpooling nodes. We initialize $E_a := \emptyset$. For each edge $\{i,j\} \in E_u$, we generate a probability of adding one more additional edge using an MLP as follows: $p_a(\{i,j\}) = \text{MLP-IE-A}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{w}_{\{i,j\}})$. We draw a random variable $U_1 \sim U[0,1]$ and if $U_1 < p_a(\{i,j\})$, we let $E_a := E_a \cup \{\{i,j\}\}$ and $E^o := E^o \cup E_a^o$, where E_a^o defined in the following three cases: (1) if $|N_{\{i,j\}}(i)| = |N_{\{i,j\}}(j)| = 1$, then $E_a^o(\{i,j\}) := \{\{k,l\}, k \in f(i), l \in f(j),$ and $k \notin N_{\{i,j\}}(i), l \notin N_{\{i,j\}}(j)\}$; (2) if $|N_{\{i,j\}}(i)| + |N_{\{i,j\}}(j)| = 3$, without loss of generality, assume $|N_{\{i,j\}}(i)| = 1$ and $|N_{\{i,j\}}(j)| = 2$, then draw

$U_2 \sim U[0,1]$. If $U_2 < \frac{p_1(i,j)}{p_1(i,j)+p_2(i,j)}$ ($p_1(i,j)$ and $p_2(i,j)$ were obtained in Step 3c), we set $r_{ij}(j) := 1$, otherwise $r_{ij}(j) := 2$. We then define $E_a^o(\{i,j\}) := \{\{k, f_{r_{ij}(j)}(j)\}$, $k \in f(i)$, and $k \notin N_{\{i,j\}}(i)\}$; (3) if $|N_{\{i,j\}}(i)| + |N_{\{i,j\}}(j)| = 4$, then $E_a^o(\{i,j\}) := \emptyset$. We thus updated the edge set of the output graph as follows

$$E^o := E^o \bigcup \bigcup_{\{i,j\} \in E_a} E_a^o(\{i,j\}).$$

We also record the total log probability of this step:

$$\begin{aligned}\text{logP-A} = &\sum_{\{i,j\} \in E_a} \ln p_a(\{i,j\}) + \sum_{\{i,j\} \notin E_a, \{i,j\} \in E_c} \ln(1 - p_a(\{i,j\})) \\ &+ \sum_{\substack{\{i,j\} \in E_a, \\ |N_{\{i,j\}}(i)|=1, \\ |N_{\{i,j\}}(j)|=2}} \ln \frac{p_{r_{ij}(j)}(i,j)}{p_1(i,j) + p_2(i,j)}. \end{aligned} \tag{B.7}$$

Summary of step 2. The edges of the output graph include all intra-links and inter-links as follows:

$$\begin{aligned} E^o = &\Big\{\{f_1(j), f_2(j)\} : j \in I_u, j \in V_c\Big\} \\ &\bigcup \Big\{\{k,l\} : \{i,j\} \in E, k \in N_{\{i,j\}}(i), l \in N_{\{i,j\}}(j)\Big\} \\ &\bigcup \bigcup_{\{i,j\} \in E_a} E_a^o(\{i,j\}). \end{aligned}$$

Step 3. Constructing edge features. Using the node features and edges from obtained from the previous steps, for each edge $\{k,l\} \in E^o$, we build the corresponding edge feature $\boldsymbol{u}_{k,l} = \text{MLP-}u(\boldsymbol{y}_k, \boldsymbol{y}_l)$.

The overall probability for updating the training parameters: The final probability, P, is the product of all probabilities from step 1a and step 2a-2d. We

obtain its logarithm, logP, by combining (B.1) - (B.4) and (B.6) as follows:

$$\text{logP} = \text{logP-R} + \text{logP-IA} + \text{logP-C} + \text{logP-IE} + \text{logP-A}.$$

This probability is used to update the training parameters in the unpooling layer while using the REINFORCE algorithm.

B.2 Comments on the Codes

We included a zipped folder of codes that implement the proposed method and some baseline methods. It also contains notebooks that implement numerical experiments for the Waxman random graph, protein, QM9, and ZINC datasets. All the implemented codes can be found in `https://github.com/guo00413/graph_unpooling`.

We implemented five baseline methods in the numerical experiments for the Waxman random graph and the protein dataset: GraphVAE (Simonovsky and Komodakis, 2018), GraphRNN (You et al., 2018b), GCPN (You et al., 2018a), GraphAF (Shi et al., 2020) and SGD-VAE (Guo et al., 2021). We implement GraphVAE and GraphRNN based on codes from `https://github.com/JiaxuanYou/graph-generation` (licensed by MIT), we implement GCPN based on codes from `https://github.com/bowenliu16/rl_graph_generation` (licensed by BSD 3-Clause). We implement SGD-VAE and GraphAF from scratch, because we do not find codes with the appropriate license to use and modify. We select model hyperparameters based on their original papers.

We remark that GCPN can not produce numeric node features due to its generation strategy. As a result, we do not report the metric related to node features for GCPN. To clarify, GCPN (You et al., 2018a) enlarges a single node at each time based on the existing graph as follows: it first adds 9 new nodes, corresponding to 9 types of

heavy atoms in ZINC dataset; it constructs one edge connecting one of the new nodes to one of the nodes in the existing graph; it further sequentially determines to stop this step or to construct more edges. This approach can naturally handle graphs with categorical node features, but it fails to generate numeric node features. Therefore in the numerical experiments, we only generate a non-featured graph from GCPN and do not report metrics related to node features.

B.3 Implementation Details

Section B.3.1 introduces layers used in the experiments other than the unpooling layer; §B.3.2 provides details of the components in the unpooling layer; §B.3.3 describes the architectures of the discriminators in UL GAN and of the encoders in UL VAE; §B.3.4 details the architectures of the generative networks, including generators in UL GAN and decoders in UL VAE; and §B.3.5 explains how we calculate the reconstruction error for UL VAE.

We remark that all numerical experiments are run in a machine with Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz and NVIDIA TITAN RTX (576 tensor cores, and 24 GB GDDR6 memory).

B.3.1 Details of Other Neural Layers

Aggregation. We introduce the following aggregation function $\text{agg}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \text{LeakyReLU}(\boldsymbol{x}_1 + \boldsymbol{x}_2)$. It is used when the inputs are order invariant, for example, to produce edge feature based on features of two end nodes.

MLP. We denote a multilayer perceptron by $\text{MLP}[k_0, k_1, k_2, ..k_m]$. It contains $m-1$ hidden blocks, where the i-th block, $i = 1, \ldots, m-1$, cascades a linear layer with input dimension k_{i-1} and output dimension k_i, a batch normalization and a LeakyReLU activation with negative slope 0.05 (the same slope is used for all

LeakyReLU activations in our experiments); and an output block which is a linear layer with output dimension k_m.

Initial layer. An initial layer of a generator takes a latent vector $\boldsymbol{z}$ and produces a 3-nodes graph. To define an initial layer, we need to specify the dimension of the input vector, d_{in}, the dimension of the output node feature, d_x, the dimension of the output edge feature, d_w.

The initial layer generates a 3-nodes graph as follows: First, a multilayer perceptron MLP-INI-V = MLP$[d_{in}, 6d_x, 3d_x]$ takes a d_{in}-dimensional input $\boldsymbol{z}$ and generates the initial node features for the 3-nodes graph. It further reshapes it to a matrix in $\mathbb{R}^{3\times d_x}$, where d_x is the dimension of the output node features. Second, it calculates the probabilities from $(p_1, p_2, p_3, p_4) = \text{MLP-INI-E}(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = \text{softmax}(\text{MLP}[3d, 16, 4])(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$, draws a uniform random variable $U \sim U[0, 1]$, and determines the initial edge set to be

$$E = \begin{cases} \{\{1,2\},\{1,3\}\}, & \text{if } U < p_1, \\ \{\{1,2\},\{2,3\}\}, & \text{if } p_1 \le U < p_1 + p_2 \\ \{\{1,3\},\{2,3\}\}, & \text{if } p_1 + p_2 \le U < p_1 + p_2 + p_3 \\ \{\{1,2\},\{2,3\},\{1,3\}\}, & \text{otherwise} \end{cases}$$

The log probability from the initial layer is $\ln(p_s)$, where $s \in \{1, 2, 3, 4\}$ corresponding to the determined initial edge set. Third, it constructs edge feature vectors using a multilayer perceptron as follows:
$\text{MLP-INI-W}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \text{LeakyReLU}(\text{BN}(\text{MLP}[d, d_w, d_w](\text{agg}(\boldsymbol{x}_i, \boldsymbol{x}_j))))$, for each $\{i, j\} \in E$, where BN is a batch normalization.

Skip connection. The generator of UL GAN adopts a skip connection procedure (Brock et al., 2019) to avoid mode collapse. A skip connection can be specified by the input dimension, d_z, the node feature dimension, d_y, the multiplier for hidden dimension, N_z, and the maximal number of nodes in the output graph, N. The skip

connection generates additional node features by

$$\text{LeakyReLU}(\text{BN}(\text{MLP}[d_z, N_z d_y, N d_y](\boldsymbol{z}))).$$

MPNN. Our implemented models use an edge-conditional MPNN (Gilmer et al., 2017; Simonovsky and Komodakis, 2017), which can be characterized by input node feature dimension d_x, input edge feature dimension d_w and output node feature dimension d_y. For a featured graph $(V, E, \boldsymbol{X}, \boldsymbol{W})$ with d_x-dimensional node features and d_w-dimensional edge features, the rows of the output features matrix $\boldsymbol{Y}$ are formed by

$$\boldsymbol{y}_j = \text{LeakyReLU}\left(\text{BN}\left(\boldsymbol{x}_j \boldsymbol{\Theta}_s + \sum_{i \in N_j} \boldsymbol{x}_i H_n(\boldsymbol{w}_{i,j})\right)\right),$$

where N_j is a set of neighbors of j, H_n is a linear layer, which maps an edge feature vector to a matrix with the same dimensions as $\boldsymbol{\Theta}_s$.

Final layer for edge features. The final layer outputs feature vectors for each edge in the output graph, while leveraging the node features of the connecting nodes and the edge features from the last intermediate graph. The parameters of this layer include the dimension of input edge features, d_w, the dimension of input node features, d_x, and the dimension of the output edge features, d_u. It produces the final edge features by

$$\begin{aligned}
&\text{MLP-}uf(\boldsymbol{w}_{i,j}, \boldsymbol{x}_i, \boldsymbol{x}_j) \\
&= \text{MLP}[d_x + 2d_w, d_u](\boldsymbol{w}_{i,j}, \text{MLP}[d_x, d_w, d_w](\text{agg}(\boldsymbol{x}_i, \boldsymbol{x}_j)), \text{agg}(\boldsymbol{x}_i, \boldsymbol{x}_j)).
\end{aligned}$$

B.3.2 Details of the Unpooling Layer

Hyperparameters. Recall from §B.1 that an unpooling layer takes a featured input graph $\mathcal{G} = (V, E, \boldsymbol{X}, \boldsymbol{W})$, with d_x-dimensional node features and d_w-dimensional edge

features. This unpooling layer is thus specified using the following hyperparameters:

- I'_s: a set of nodes fixed as static (i.e., they will not be unpooled).

- I_r: a set of nodes that are determined to be static based on the probabilities $p^s_j = \text{MLP-S}(\boldsymbol{x}_j)$ for $j \in I_r$, where $\text{MLP-S} = \text{sigmoid}(\text{MLP}[d_x, \lfloor d_x/2 \rfloor, 1])$. The overall static nodes for the unpooling layer are

$$I_s = I'_s \cup \{j : j \in I_r,\, U_j < \text{MLP-S}(\boldsymbol{x}_j)\},$$

  where U_j are i.i.d. random variables drawn from a uniform distribution on $U[0, 1]$.

- k_v, d_y: hidden dimension and output dimension of node features. We let $d'_x = \lfloor d_x/2 \rfloor + \lfloor d_x/4 \rfloor$ and

$$\text{MLP-}y = \text{MLP}[d'_x, k_v, d_y],$$

  so that the child node feature vector is given by $\boldsymbol{y}_{f_j(i)} = \text{MLP-}y(P_{S_j}(\boldsymbol{x}_i))$. The latter projection, P_{S_j}, is defined for $j = 1, 2$, and each feature vector $\boldsymbol{x} = (x_1, ...x_d)$ of a node in I_u as

$$P_{S_1}(\boldsymbol{x}) = (x_1, ..., x_{d_s}, x_{d_s+1}, ...x_{d_s+D})^T,$$
$$P_{S_2}(\boldsymbol{x}) = (x_1, ..., x_{d_s}, x_{d_s+D+1}, ...x_{d_s+2D})^T,$$

  where $d_s = \lfloor d_x/2 \rfloor$ and $D = \lfloor \frac{d_x}{4} \rfloor$.

- k_{ia} and k_{ie}: hidden dimensions in MLP-IA and MLP-IE, respectively. The probabilities used for generating intra- and inter-links are given by

$$\text{MLP-IA}(\boldsymbol{x}) = \text{sigmoid}(\text{MLP}[d_y, k_{ia}, 1](\boldsymbol{x})), \tag{B.8}$$

$$\begin{aligned}\text{MLP-IE}(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{w}, \boldsymbol{x}) =& \text{softmax}\Big(\text{MLP-IE-1}(\boldsymbol{y}_1, \boldsymbol{w}, \boldsymbol{x}), \text{MLP-IE-1}(\boldsymbol{y}_2, \boldsymbol{w}, \boldsymbol{x}),\\ &\text{MLP-IE-2}(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{w}, \boldsymbol{x})\Big),\end{aligned} \tag{B.9}$$

where the first two MLPs used in MLP-IE are the same networks, defined as

$$\text{MLP-IE-1} := \text{MLP}[d_y + d_w + d_x, k_{ie}, 1],$$

and the third term is defined as

$$\text{MLP-IE-2}(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{w}, \boldsymbol{x}) := \text{MLP}[d_y + d_w + d_x, k_{ie}, 1](\text{agg}(\boldsymbol{y}_1, \boldsymbol{y}_2), \boldsymbol{w}, \boldsymbol{x}).$$

In practice MLP-C in Step 3b in §B.1 is based on MLP-IE as follows. It uses MLP-IE-2 and calculates

$$h_C(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{w}, \boldsymbol{x}) := \text{MLP-IE-2}(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{w}, \boldsymbol{x}).$$

The probabilities used in Step 3b in §B.1 are then calculated as

$$\begin{aligned}\big(p_b(j, i_1), p_b(j, i_2), ...p_b(j, i_{m_j})\big) = \text{softmax}\big(&h_C(\boldsymbol{y}_{f_1(j)}, \boldsymbol{y}_{f_2(j)}, \boldsymbol{w}_{\{j,i_1\}}, \boldsymbol{x}_{i_1}),\\ &h_C(\boldsymbol{y}_{f_1(j)}, \boldsymbol{y}_{f_2(j)}, \boldsymbol{w}_{\{j,i_2\}}, \boldsymbol{x}_{i_2}), ...,\ h_C(\boldsymbol{y}_{f_1(j)}, \boldsymbol{y}_{f_2(j)}, \boldsymbol{w}_{\{j,i_{m_j}\}}, \boldsymbol{x}_{i_{m_j}})\big).\end{aligned}$$

- k_w, d_u: hidden dimension and output dimension of the edge features. The output

edge feature vectors are built as follows:

$$\boldsymbol{u}_{i,j} = \text{MLP-}u(\boldsymbol{y}_i, \boldsymbol{y}_j) = \text{LeakyReLU}(\text{BN}(\text{MLP}[d_y, k_w, d_u](\text{agg}(\boldsymbol{y}_i, \boldsymbol{y}_j)))).$$

Preference score. We further introduce preference score and modify the direct way of generating the probabilities obtained in (B.9), in order to avoid all the edges being inherited by one single node in a pair of children nodes and thus to ensure that the unpooling layer can produce all possible output graphs with probabilities that are not too small. Denote

$$h_s(\boldsymbol{y}, \boldsymbol{w}, \boldsymbol{x}) := \text{MLP-IE-1}(\boldsymbol{y}, \boldsymbol{w}, \boldsymbol{x}) \;\text{ and }\; h_b(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{w}, \boldsymbol{x})) := \text{MLP-IE-2}(\boldsymbol{y}, \boldsymbol{w}, \boldsymbol{x}),$$

where the forms of MLP-IE-1 and MLP-IE-2 are defined below (B.9). For each child node $f_1(j)$ with feature $\boldsymbol{y}$, which is generated from an input node j, we rescale the h_s's to obtain the preference score $h_s^{(p)}$ when $N_{\{k,j\}}(j) = \{f_1(j)\}$ as follows:

$$\big(\{h_s^{(p)}(\boldsymbol{y}, k) : k \in N_j\}, h_{s,n}^{(p)}(\boldsymbol{y})\big) := \text{softmax}\big(\{h_s(\boldsymbol{y}, \boldsymbol{w}_{j,k}, \boldsymbol{x}_k) : k \in N_j\}, h_s^{(0)}(\boldsymbol{y})\big),$$

where N_j is set of neighbors of node j, and $h_s^{(0)}(\boldsymbol{y})$ is capturing zero-preference (i.e., preference of not connecting any inter-link) and is calculated by a multilayer perceptron $\text{MLP}[d_y, 2d_y, 1]$.

Similarly, we also rescale the h_b's to obtain the preference score when $N_{\{k,j\}}(j) = \{f_1(j), f_2(j)\}$. Let $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ be feature vectors of children nodes generated from j. Let $\boldsymbol{x}$ be the feature vector of node j in the input graph. With an added zero

preference $h_b^{(0)}(\boldsymbol{x}) = \mathrm{MLP}[d_x, 2d_x, 1](\boldsymbol{x})$, the preference score is calculated as

$$\begin{aligned}&\left(\{h_b^{(p)}(\boldsymbol{y}_1, \boldsymbol{y}_2, k) : k \in N_j\}, h_{b,n}^{(p)}(\boldsymbol{y}_1, \boldsymbol{y}_2)\right)\\ &:= \mathrm{softmax}\left(\{h_b(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{w}_{j,k}, \boldsymbol{x}_k) : k \in N_j\}, h_b^{(0)}(\boldsymbol{x})\right).\end{aligned}$$

We use the preference scores $h_s^{(p)}$ and $h_b^{(p)}$ instead of MLP-IE to obtain probabilities for building $N_{\{k,j\}}(j)$ for the inter-link in Step 2c in §B.1 as

$$\begin{aligned}(p_1, p_2) &= \left(\frac{(h_s^{(p)}(\boldsymbol{y}_1, k)}{Z}, \frac{h_s^{(p)}(\boldsymbol{y}_1, k)}{Z}\right),\\ &\text{where} \quad Z = \left(h_s^{(p)}(\boldsymbol{y}_1, k) + h_s^{(p)}(\boldsymbol{y}_2, k) + h_b^{(p)}(\boldsymbol{y}_1, \boldsymbol{y}_2, k)\right).\end{aligned}$$

B.3.3 Architectures of Discriminators and Encoders

Discriminators of UL GAN. The details of the discriminator in UL GAN and the encoder in UL VAE for QM9 have been described in §4.3. In this part, we describe the details for other datasets.

The discriminator in the Waxman random graph data takes an input graph and uses two MPNN layers with 64 and 128 units to generate a graph $\mathcal{G} = (V, E, \boldsymbol{X}, \boldsymbol{W})$. It then aggregates the node features (the rows $\{\boldsymbol{x}_j\}_{j \in V}$ of $\boldsymbol{X}$) to produce the following single feature vector for the graph:

$$\boldsymbol{h}(\mathcal{G}) = \sum_{j \in V} \sigma(\mathrm{lin}_1(\boldsymbol{x}_j)) \odot \tanh(\mathrm{lin}_2(\boldsymbol{x}_j));$$

where σ is logistic sigmoid, $\odot$ is element-wise multiplication and lin_1 and lin_2 are two layers with 128 units. Then it uses a global sum pooling to extract a signal for each graph. It applies two linear layers with 128 and 256 units. A final layer with a single unit then produces the output, where its activation function is sigmoid.

The LeakyReLU activation function (with 0.05 negative slope) is used after the two MPNNs and all linear layers.

The discriminator in the protein dataset is similar to the discriminator in the random graph dataset. Some parameters were changed to simplify the network that fit the protein dataset; we used one MPNN layer with 128 units and one linear layer with 256 units after the global pooling.

The discriminator in ZINC is the same as the discriminator used in QM9 as described in §4.3.

Encoder in UL VAE. It has the same architecture as the corresponding discriminator described above, except that the final layer maps into a 256-dimensional vector with a linear activation function. This vector further splits to two 128-dimensional vectors: $\boldsymbol{z}_\mu$ and $\boldsymbol{z}_\sigma$. It then generates the following latent vector: $\boldsymbol{z} = \boldsymbol{z}_\mu + \exp(\frac{1}{2}\boldsymbol{z}_\sigma) \odot \boldsymbol{x}$, where $\boldsymbol{x} \sim N(0, 1)$.

Additional adjustment when the number of nodes of the output graph is not fixed. For the ZINC dataset, since the number of nodes of the output graph is not fixed by the generator, we added one trivial predictive network to the discriminator. This additional network contains one hidden layer with 256 units and only takes the global sum of node feature vectors and edge feature vectors as input, to encourage a proper distribution of the numbers of atoms and edges. The final output of the discriminator is the sum of the outputs of this network and the original GNN discriminator.

B.3.4 Architecture of Generators

We describe details of the architecture of the generator in UL GAN and decoder in UL VAE for the different datasets.

Architecture of the generator in UL GAN. We show the generators in UL GAN (same as the decoders in UL VAE, if applicable) for the Waxman random graph,

protein, QM9 and ZINC datasets in Figure B.1, Figure B.2, Figure B.3, and Figure B.4, respectively. For simplicity, we neglect batch sizes. For example, the input should be $\boldsymbol{z} \in \mathbb{R}^{B\times 128}$, where B is the batch size.

Loss function. In practice, we alternatively train the generator with a GAN's loss function and with REINFORCE. In each training step, we first update the generator using the loss function $-D(G(z))$. We then update the parameter of the generator, θ_G, using

$$\boldsymbol{\theta}_G^{(t+1)} := \boldsymbol{\theta}_G^{(t)} + \alpha\Big(\nabla_{\boldsymbol{\theta}}\mathrm{logP}|_{\boldsymbol{\theta}_G^{(t)}}\Big)(D(G(z)) - \mathbb{E}(D(G(z)))),$$

where the learning rate $\alpha = 5 \times 10^{-2}$.

B.3.5 Reconstruction Loss for UL VAE

We calculate the reconstruction loss for UL VAE following Simonovsky and Komodakis (2018).

Given two featured graphs with the same number of nodes, we calculate their distance by summing the distances between node features, the distances between adjacency matrices and the distances between edge features of these two graphs. We used the same formula described in Section 3.3 in Simonovsky and Komodakis (2018).

When the number of nodes may vary (e.g., in the QM9 dataset, the corresponding graphs may contain 6–9 nodes) and the input/output graph has less nodes than the maximal number of nodes (e.g., less than 9 nodes in the QM9 experiment), we add artificial nodes to the graph so that it contains the maximal number of nodes (e.g., 9 nodes for QM9). We add an additional binary feature to the node features: it assigns 0 to the real nodes and 1 to the artificial nodes.

For an input graph and an output graph with the same numbers of nodes (if they have different node numbers, we perform the above procedure to make them the same), we permute the node indices of the output graph in order to achieve a minimal

distance to the input graph. This minimal distance is regarded as the reconstruction loss between the input and output graph. More details of how to permute the indices appear in Section 3.4 of Simonovsky and Komodakis (2018).

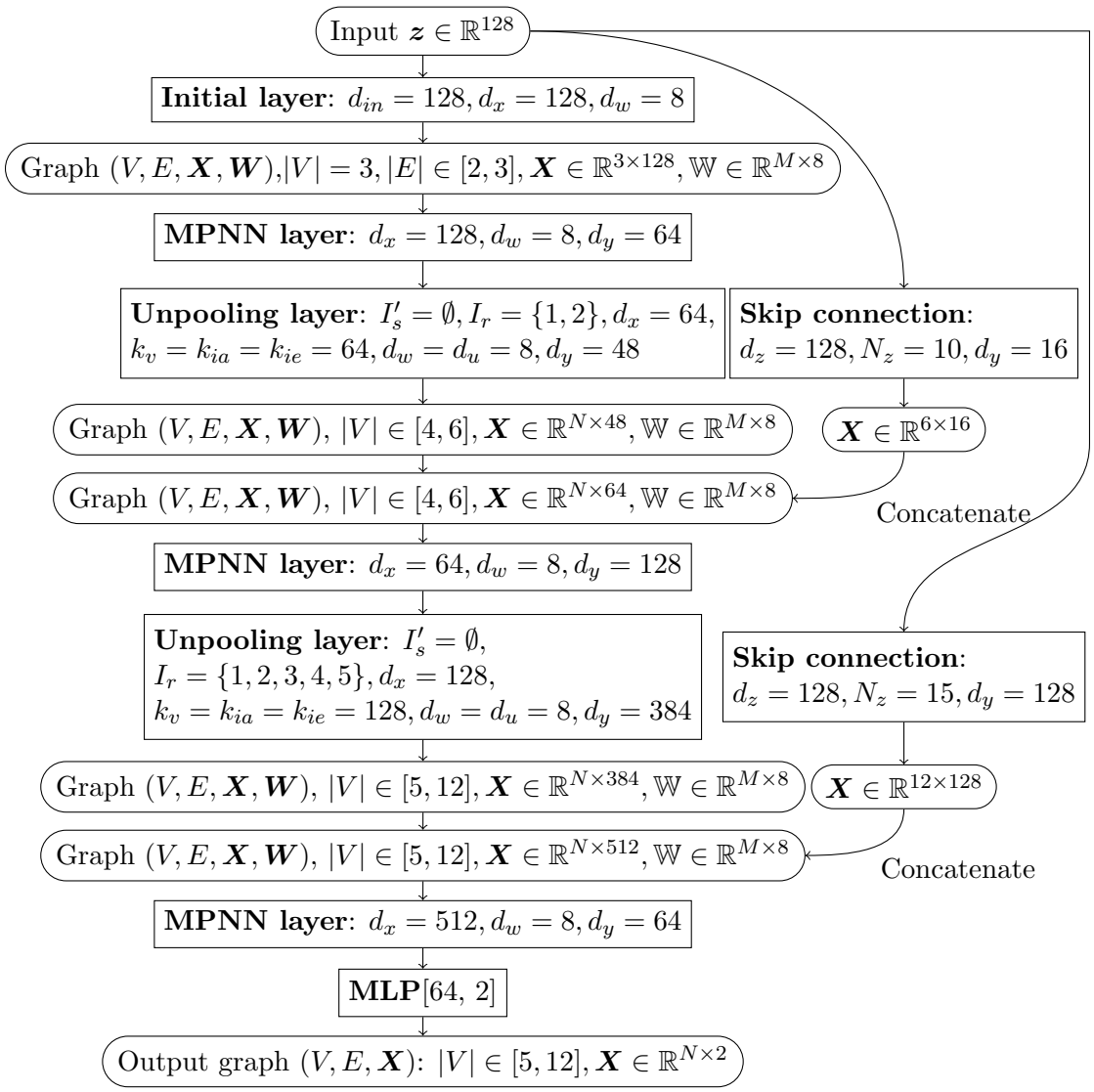

Figure B.1: Demonstration of the architecture of the generator in UL GAN and the decoder in UL VAE for Waxman random graph dataset. The rectangles present neural network layers and the rounded rectangles clarify the shape of the hidden data.

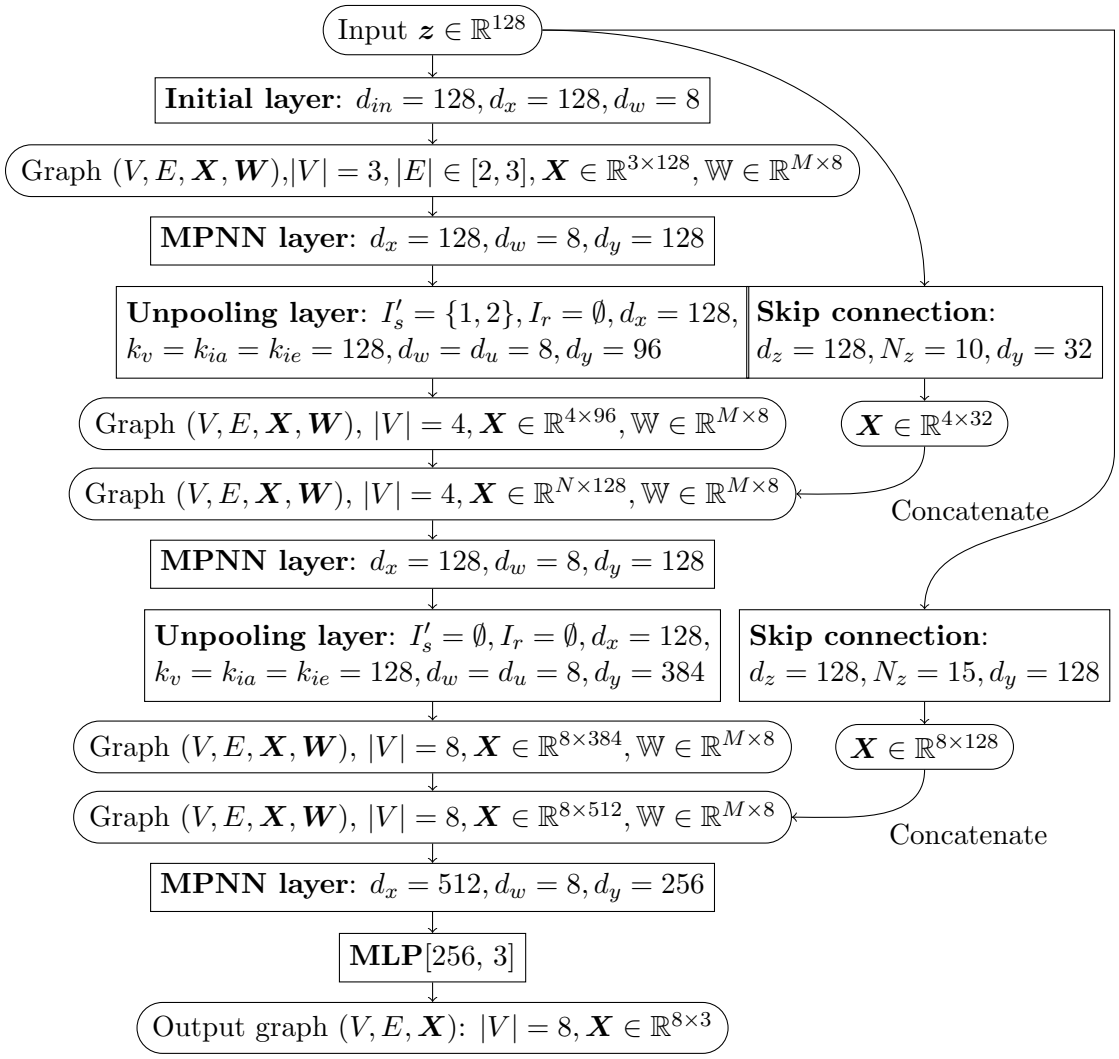

Figure B.2: Demonstration of the architecture of the generator in UL GAN and the decoder in UL VAE for protein dataset. The rectangles present neural network layers and the rounded rectangles clarify the shape of the hidden data.

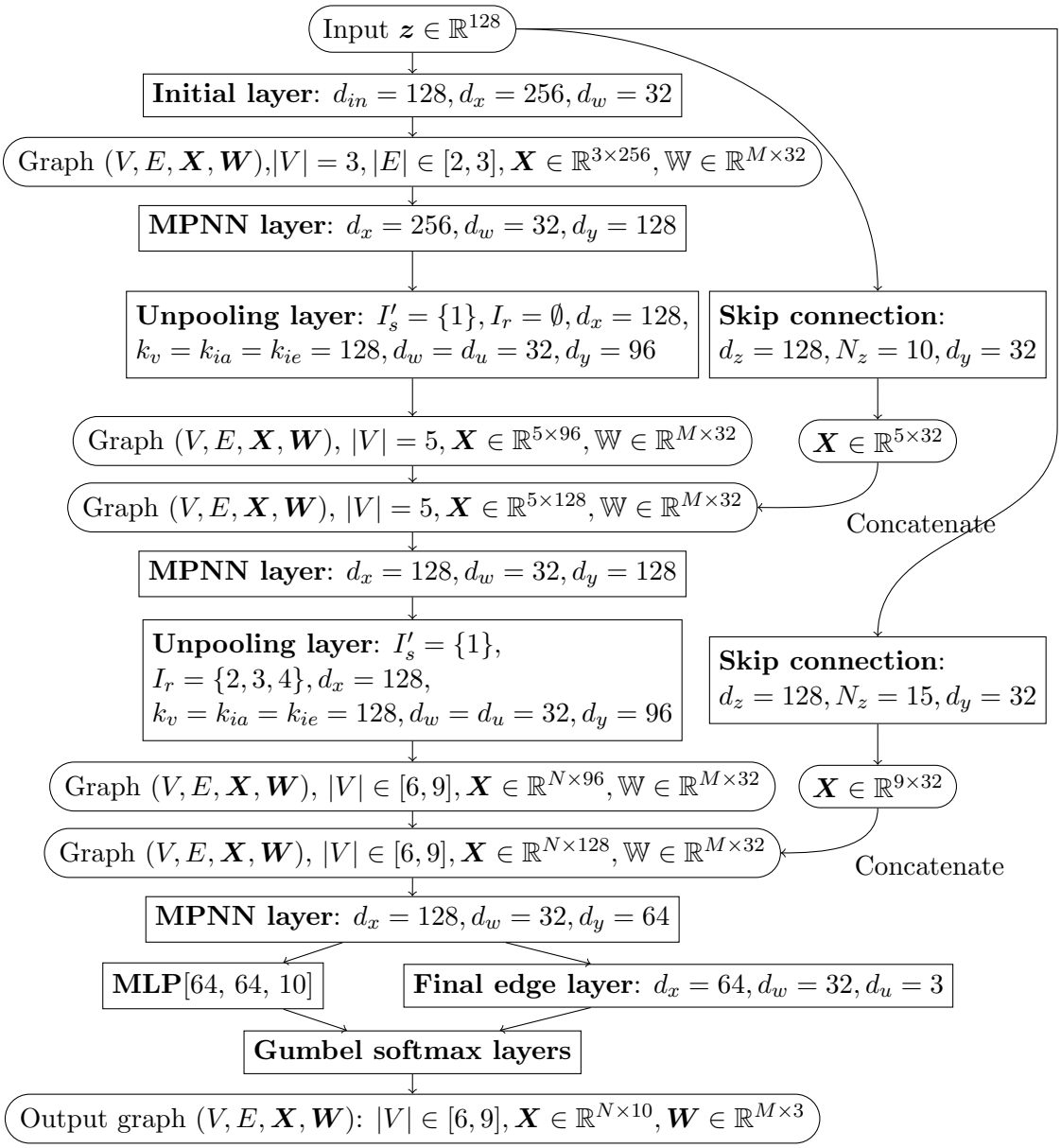

Figure B.3: Demonstration of the architecture of the generator in UL GAN and the decoder in UL VAE for QM9. The rectangles present neural network layers and the rounded rectangles clarify the shape of the hidden data.

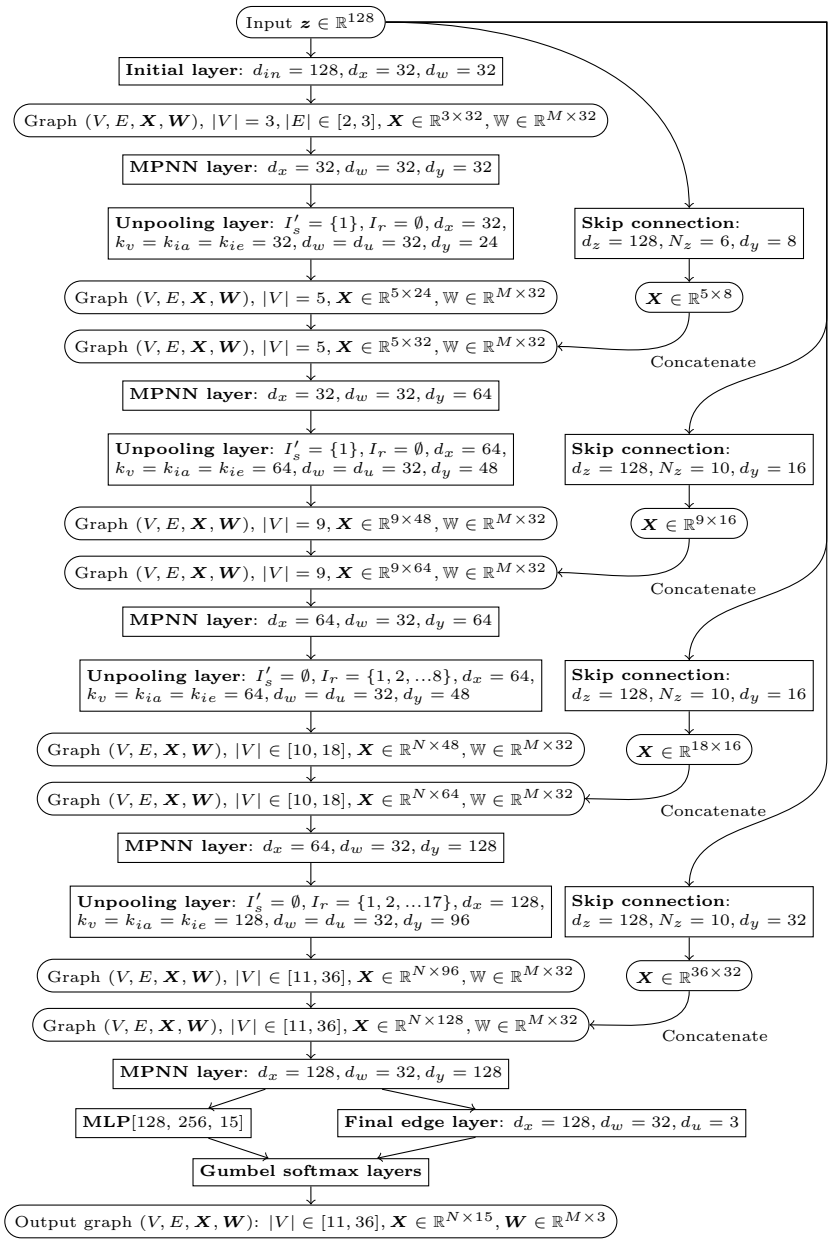

Figure B.4: Demonstration of the architecture of the generator in UL GAN for ZINC. The rectangles present neural network layers and the rounded rectangles clarify the shape of the hidden data.

B.4 Proof of Theorems

In this section, we prove claims on connectivity and expressivity for the output graph of our unpooling layer. For the ease of presenting, we denote a graph by $\mathcal{G} = (V, E)$ and omit features in nodes and edges because features are irrelevant with graph connectivity and expressivity.

B.4.1 Proof of Proposition 1

Let k^o, $l^o \in V^o$ be two arbitrary nodes in the output graph $\mathcal{G}^o = (V^o, E^o)$. In order to prove the connectivity of $\mathcal{G}^o$ we need to find a path in $\mathcal{G}^o$ connecting k^o and l^o. Recall that $\mathcal{G} = (V, E)$ denotes the input graph. Let k, $l \in V$ be the parent nodes of k^o and l^o, respectively. Since the input graph $\mathcal{G}$ is connected, k and l are connected by a path. We denote its length by $n-1$, where $n \geq 2$, and its edges by $\{i_1, i_2\}$, $\{i_2, i_3\}$, ...$\{i_{n-1}, i_n\}$, where $i_1 = k$ and $i_n = l$. Recall that for $r \in [n]$ $N_{\{i_r, i_{r+1}\}}(i_r)$ is a subset of the children of node i_r connected to $N_{\{i_r, i_{r+1}\}}(i_{r+1})$, which is a subset of the children of node i_{r+1}. For $r \in [n]$ we arbitrarily choose a vertex in $N_{\{i_r, i_{r+1}\}}(i_r)$ and denote it by $v_{\{i_r, i_{r+1}\}}(i_r)$. We thus note that the following edges exist in the output graph:

$$\left\{v_{\{i_1,i_2\}}(i_1), v_{\{i_1,i_2\}}(i_2)\right\}, \left\{v_{\{i_2,i_3\}}(i_2), v_{\{i_2,i_3\}}(i_3)\right\}, \cdots \left\{v_{\{i_{n-1},i_n\}}(i_{n-1}), v_{\{i_{n-1},i_n\}}(i_n)\right\}.$$

In order to prove that k^o and l^o connect by a path we verify the following properties:

1. For all $r \in [n-2]$, either $v_{\{i_r, i_{r+1}\}}(i_{r+1}) = v_{\{i_{r+1}, i_{r+2}\}}(i_{r+1})$ or there is a path connecting $v_{\{i_r, i_{r+1}\}}(i_{r+1})$ and $v_{\{i_{r+1}, i_{r+2}\}}(i_{r+1})$

2. Either $k^o = v_{\{i_1, i_2\}}(i_1)$ or there is a path connecting k^o and $v_{\{i_1, i_2\}}(i_1)$

3. Either $l^o = v_{\{i_{n-1}, i_n\}}(i_n)$ or there is a path connecting l^o and $v_{\{i_{n-1}, i_n\}}(i_n)$

To prove the first property we note that $v_{\{i_r,i_{r+1}\}}(i_{r+1}) \in \{f_1(i_{r+1}), f_2(i_{r+1})\}$ and $v_{\{i_{r+1},i_{r+2}\}}(i_{r+1}) \in \{f_1(i_{r+1}), f_2(i_{r+1})\}$. If $v_{\{i_r,i_{r+1}\}}(i_{r+1}) \neq v_{\{i_{r+2},i_{r+1}\}}(i_{r+1})$, then $\{v_{\{i_r,i_{r+1}\}}(i_{r+1}), v_{\{i_{r+1},i_{r+2}\}}(i_{r+1})\} = \{f_1(i_{r+1}), f_2(i_{r+1})\}$. To show that there is a path connecting $v_{\{i_r,i_{r+1}\}}(i_{r+1})$ and $v_{\{i_{r+1},i_{r+2}\}}(i_{r+1})$, it suffices to show that there is a path connecting $f_1(i_{r+1})$ and $f_2(i_{r+1})$. Assume an arbitrary node $j \in V$. If $j \in V_c$ the construction in Step 3a guarantees the existence of an edge connecting $f_1(j)$ and $f_2(j)$. Otherwise, based on Step 3b, there exists a node b_j such that $N_{\{b_j,j\}}(j) = \{f_1(j), f_2(j)\}$, so that $f_1(j)$ and $f_2(j)$ are connected via $f(b_j)$. Letting $j = i_{r+1}$ we conclude this property.

The above argument also applies to the other two properties.

B.4.2 Proof of Theorem 2

We will first define a pooling process and show that there exists an unpooling layer that acts as an inverse of this pooling procedure (see Lemma B.1). We then show that any graph with N nodes can be pooled to a graph with $\lceil N/2 \rceil$ nodes (Lemma B.2 clarifies the case where N is even and Lemma B.3 clarifies the cases where N is odd). Finally we use these observations to conclude the proof of the theorem.

We define the pooling process by using an "eligible" set. For a graph $\mathcal{G}^o = (V^o, E^o)$, a set of pairs of nodes in V^o, $S = \{(i_1, j_1), ...(i_n, j_n)\}$, is called *eligible* if all nodes $i_1 \cdots i_n$ and $j_1 \cdots j_n$ are distinct and for any $m \in [n]$, i_m and j_m are connected by a path of length at most 2; that is, there are two cases: either $\{i_m, j_m\} \in E^o$ or there exists $k \in V^o$, such that $\{i_m, k\} \in E^o$ and $\{k, j_m\} \in E^o$. Using an arbitrarily chosen eligible set S, we describe a specific pooling process with respect to S that produces a graph $\mathcal{G}$ from $\mathcal{G}^o$ as follows. We initialize $\mathcal{G}$ with $V = V^o$ and $E = E^o$. For $m = 1, 2, \cdots n$, we follow the next three steps: (1) We remove from V the nodes i_m and j_m. We remove from E all edges in $E_m := \{e \in E : i_m \in e \text{ or } j_m \in e\}$; (2) We add a new node i'_m to V; (3) We add the following set of new edges to

E: $\{\{i'_m, k\} : k \in V$ and either $\{i_m, k\} \in E_m$ or $\{j_m, k\} \in E_m\}$. It is clear that the resulting $\mathcal{G}$ is connected if $\mathcal{G}^o$ is connected.

We introduce a lemma showing that for a given pooling process which maps $\mathcal{G}^o$ to $\mathcal{G}$, there exists an unpooling layer as the inverse of this pooling process, i.e., it maps $\mathcal{G}$ to $\mathcal{G}^o$.

Lemma B.1

For any pooling process that maps $\mathcal{G}^o = (V^o, E^o)$ to $\mathcal{G} = (V, E)$, there exists an unpooling layer that maps $\mathcal{G}$ to $\mathcal{G}^o$. □

Proof: Consider the pooling process with respect to the eligible pairs in $S = \{(i_1, j_1), ...(i_n, j_n)\} \subset V^o$. We use the same notation as above for $i'_1, i'_2, ...i'_n \subset V$ that were pooled by the respective eligible pairs.

We construct an unpooling layer whose input is $\mathcal{G}$ and its output is $\hat{\mathcal{G}}^o = (\hat{V}^o, \hat{E}^o)$. The unpooling layer unpools the nodes $i'_1, i'_2, ...i'_n$ and keeps the remaining nodes unchanged. It forms the following children nodes of $i'_1, i'_2, ...i'_n$ in $\hat{V}^o$: $(f_1(i'_1), f_2(i'_1)), ...(f_1(i'_n), f_2(i'_n))$, respectively. For every $r \in [n]$, we identify $(f_1(i'_r), f_2(i'_r))$ with (i_r, j_r) and re-index respectively, so $\hat{V}^o = V^o$.

It remains to show that we can find an unpooling layer such that $\hat{E}^o = E^o$. We first note that the edges in E^o that do not contain nodes from the eligible set remain unchanged in E and E^o since the pooling process is identical on those edges (for clarity, these edges are the ones in $\{\{i, j\} \in E^o : i, j \notin \{i_1, i_2, ...i_n, j_1, j_2, ...j_n\}\}$). Since we restricted above the unpooling layer to only unpool $i'_1, i'_2, ...i'_n$ those edges also remain unchanged in $\hat{E}^o$. We thus only need to show that we can construct the unpooling layer so that the set of edges in $\hat{E}^o$ that contain children nodes is the set of edges that contain nodes from the eligible set in E^o. Each edge in the latter set falls into one of the following three categories, for which we establish the required equality with the corresponding edges in $\hat{E}^o$:

1. Edges whose end nodes form a pair $\{i_r, j_r\} \in S$. For each such edge, we select the intra-link in the unpooling layer (step 3a) so that there is an edge connecting $f_1(i'_r)$ and $f_2(i'_r)$ in $\hat{E}^o$.

2. Edges between an eligible pair (i_r, j_r) and a static node k in V^o. That is, for a fixed $r \in [n]$ and a static node k, there three possibilities for the set of these edges: $\{\{i_r, k\}\}$ or $\{\{j_r, k\}\}$ or $\{\{i_r, k\}, \{j_r, k\}\}$. In view of the pooling process, the edge $\{i'_r, k\}$ is in E. In Step 3c, there are three possibilities for determining $N_{\{i'_r,k\}}(i'_r)$ and we need to select the unpooling layer to match these three possibilities. That is, $N_{\{i'_r,k\}}(i'_r) = \{f_1(i'_r)\}$ in the first case, where the set of the above edges is $\{\{i_r, k\}\}$; $N_{\{i'_r,k\}}(i'_r) = \{f_2(i'_r)\}$ in the second case, where the set of the above edges is $\{\{j_r, k\}\}$ and $N_{\{i'_r,k\}}(i'_r) = \{f_1(i'_r), f_2(i'_r)\}$ in the third case, where the set of the above edges is $\{\{i_r, k\}, \{j_r, k\}\}$. In view of the way $N_{\{i'_r,k\}}(i'_r)$ is used to build inter-links (see (B.5)), the edges in $\hat{E}^o$ between $(f_1(i'_r), f_2(i'_r))$ and k are the same as the ones in E^o between (i_r, j_r) and k.

3. Edges between two different eligible pairs, (i_r, j_r) and (i_s, j_s), that is $\{\{k, l\} \in E^o : k \in \{i_r, j_r\}, l \in \{i_s, j_s\}\}$. For fixed $s, r \in [n]$ this set of edges in E^o is a nonempty subset of the following set of four edges: $\{\{i_r, i_s\}, \{i_r, j_s\}, \{j_r, i_s\}, \{j_r, j_s\}\}$. Therefore, there are $2^4 - 1 = 15$ such edge sets. In view of the pooling process, the edge $\{i'_r, i'_s\}$ is in E. The unpooling layer unpools i'_r to $(f_1(i'_r), f_2(i'_r))$ and unpools i'_s to $(f_1(i'_s), f_2(i'_s))$. We claim that according to Steps 3c and 3d, the unpooling layer can produce all the 15 possible edge sets between the pair $(f_1(i'_r), f_2(i'_r))$ and the pair $(f_1(i'_s), f_2(i'_s))$. To clarify this claim we specify for all 15 possible edges between (i_r, j_r) and (i_s, j_s) the choice of $N_{\{i'_r,i'_s\}}(i'_r), N_{\{i'_r,i'_s\}}(i'_s)$ in Step 3c and the choice of the additional edge in Step 3d:

   - If the edge set is $\{\{i_r, i_s\}\}$, we set $N_{\{i'_r,i'_s\}}(i'_r) = \{f_1(i'_r)\}, N_{\{i'_r,i'_s\}}(i'_s) = \{f_1(i'_s)\}$ and we do not insert an edge in Step 3d

- If the edge set is $\{\{i_r, j_s\}\}$, we set $N_{\{i'_r,i'_s\}}(i'_r) = \{f_1(i'_r)\}, N_{\{i'_r,i'_s\}}(i'_s) = \{f_2(i'_s)\}$ and we do not insert an edge in Step 3d
- If the edge set is $\{\{j_r, i_s\}\}$, we set $N_{\{i'_r,i'_s\}}(i'_r) = \{f_2(i'_r)\}, N_{\{i'_r,i'_s\}}(i'_s) = \{f_1(i'_s)\}$ and we do not insert an edge in Step 3d
- If the edge set is $\{\{j_r, j_s\}\}$, we set $N_{\{i'_r,i'_s\}}(i'_r) = \{f_2(i'_r)\}, N_{\{i'_r,i'_s\}}(i'_s) = \{f_2(i'_s)\}$ and we do not insert an edge in Step 3d
- If the edge set is $\{\{i_r, i_s\}, \{i_r, j_s\}\}$, we set $N_{\{i'_r,i'_s\}}(i'_r) = \{f_1(i'_r)\}, N_{\{i'_r,i'_s\}}(i'_s) = \{f_1(i'_s), f_2(i'_s)\}$ and we do not insert an edge in Step 3d
- If the edge set is $\{\{j_r, i_s\}, \{j_r, j_s\}\}$, we set $N_{\{i'_r,i'_s\}}(i'_r) = \{f_2(i'_r)\}, N_{\{i'_r,i'_s\}}(i'_s) = \{f_1(i'_s), f_2(i'_s)\}$ and we do not insert an edge in Step 3d
- If the edge set is $\{\{i_r, i_s\}, \{j_r, i_s\}\}$, we set $N_{\{i'_r,i'_s\}}(i'_r) = \{f_1(i'_r), f_2(i'_r)\}, N_{\{i'_r,i'_s\}}(i'_s) = \{f_1(i'_s)\}$ and we do not insert an edge in Step 3d
- If the edge set is $\{\{i_r, j_s\}, \{j_r, j_s\}\}$, we set $N_{\{i'_r,i'_s\}}(i'_r) = \{f_1(i'_r), f_2(i'_r)\}, N_{\{i'_r,i'_s\}}(i'_s) = \{f_2(i'_s)\}$ and we do not insert an edge in Step 3d
- If the edge set is $\{\{i_r, i_s\}, \{j_r, j_s\}\}$, we set $N_{\{i'_r,i'_s\}}(i'_r) = \{f_1(i'_r)\}, N_{\{i'_r,i'_s\}}(i'_s) = \{f_1(i'_s)\}$ and in Step 3d we insert the edge $\{f_2(i'_r), f_2(i'_s)\}$
- If the edge set is $\{\{i_r, j_s\}, \{j_r, i_s\}\}$, we set $N_{\{i'_r,i'_s\}}(i'_r) = \{f_1(i'_r)\}, N_{\{i'_r,i'_s\}}(i'_s) = \{f_2(i'_s)\}$ and in Step 3d we insert the edge $\{f_2(i'_r), f_1(i'_s)\}$
- If the edge set is $\{\{i_r, i_s\}, \{i_r, j_s\}, \{j_r, i_s\}\}$, we set

$N_{\{i'_r,i'_s\}}(i'_r) = \{f_1(i'_r), f_2(i'_r)\}, N_{\{i'_r,i'_s\}}(i'_s) = \{f_1(i'_s)\}$ and in Step 3d we insert the edge $\{f_1(i'_r), f_2(i'_s)\}$

- If the edge set is $\{\{i_r, i_s\}, \{j_r, i_s\}, \{j_r, j_s\}\}$, we set $N_{\{i'_r,i'_s\}}(i'_r) = \{f_1(i'_r), f_2(i'_r)\}, N_{\{i'_r,i'_s\}}(i'_s) = \{f_1(i'_s)\}$ and in Step 3d we insert the edge $\{f_2(i'_r), f_2(i'_s)\}$
- If the edge set is $\{\{i_r, i_s\}, \{i_r, j_s\}, \{j_r, j_s\}\}$, we set $N_{\{i'_r,i'_s\}}(i'_r) = \{f_1(i'_r), f_2(i'_r)\}, N_{\{i'_r,i'_s\}}(i'_s) = \{f_2(i'_s)\}$ and in Step 3d we insert the edge $\{f_1(i'_r), f_1(i'_s)\}$
- If the edge set is $\{\{i_r, j_s\}, \{j_r, i_s\}, \{j_r, j_s\}\}$, we set $N_{\{i'_r,i'_s\}}(i'_r) = \{f_1(i'_r), f_2(i'_r)\}, N_{\{i'_r,i'_s\}}(i'_s) = \{f_2(i'_s)\}$ and in Step 3d we insert the edge $\{f_2(i'_r), f_1(i'_s)\}$
- If the edge set is $\{\{i_r, i_s\}, \{i_r, j_s\}, \{j_r, i_s\}, \{j_r, j_s\}\}$, we set $N_{\{i'_r,i'_s\}}(i'_r) = \{f_1(i'_r), f_2(i'_r)\}, N_{\{i'_r,i'_s\}}(i'_s) = \{f_1(i'_s), f_2(i'_s)\}$ and we do not insert an edge in Step 3d

The inter-link construction in (B.5) for the above specified choices of $N_{\{i'_r,i'_s\}}(i'_r)$ and $N_{\{i'_r,i'_s\}}(i'_s)$ together with the above specified choices of inserting an additional edge imply that the output edges between $(f_1(i'_r), f_2(i'_r))$ and $(f_1(i'_s), f_2(i'_s))$ in $\hat{E}^o$ are the same as the edges between (i_r, j_r) and (i_s, j_s).

□

To show that for a graph $\mathcal{G}^o = (V^o, E^o)$ with N node there is an unpooling layer that maps a graph $\mathcal{G}$ with $N - n$ nodes to $\mathcal{G}^o$, we need to show that there exists an eligible set $S = \{(i_1, j_1), ...(i_n, j_n)\} \subset V^o$. Indeed, the pooling process with respect to S maps $\mathcal{G}^o$ to a graph $\mathcal{G}$ with $N - n$ nodes and by Lemma B.1 there exists an unpooling layer that maps $\mathcal{G}$ to $\mathcal{G}^o$.

The next two lemmas conclude the theorem by implying that for a connected graph $\mathcal{G}^o$ there exists an eligible set of maximal size, i.e., $n = \lfloor N/2 \rfloor$. The first lemma

considers a graph with an even number of nodes and the second lemma considers a graph with an odd number of nodes.

Lemma B.2

For any connected graph $\mathcal{G}^o = (V^o, E^o)$ with $2K$ nodes, there exists an eligible set S containing K pairs of nodes in $\mathcal{G}^o$ such that the pooling process with respect to S maps $\mathcal{G}^o$ to a graph $\mathcal{G}$ with K nodes. □

Proof: We prove this lemma by induction using $M = 1, \ldots, K$. When $M = 1$, the lemma is trivial.

Assume the lemma holds for $M = 1, \ldots, K-1$. Given a connected graph $\mathcal{G}^o = (V^o, E^o)$ with $2K$ nodes, we prove the result, while considering the following two different scenarios:

Case 1: There exists a node in V^o of degree 1. We arbitrarily choose such a node and denote it by j. We consider its only neighbor, which we denote $k \in V^o$, and remove the pair of nodes (j, k) and all the edges connected to them from the graph $\mathcal{G}^o$. The remaining graph has $2K - 2$ nodes. If it is also connected, then by the induction assumption there exists an eligible set with $K - 1$ pairs of nodes. By adding the pair (j, k) to that eligible set, we obtain an eligible set with K pairs and conclude the proof.

If, on the other hand, the remaining graph is not connected, we partition it into maximally connected subgraphs $\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_m$. That is, each subgraph is connected, but any two subgraphs are not connected to each other. Since $\mathcal{G}_1, \ldots \mathcal{G}_m$ are not connected to each other and to the degree-one node j, they all connect to the node k.

We consider the following four steps that assist in finding an eligible set:

1. We identify the maximally connected subgraphs with even numbers of nodes. Clearly, each of these numbers is not greater than $2K - 2$ and thus by induction all the nodes of each such subgraph can be used to form an eligible set. The

union of all these eligible sets forms a larger eligible set that uses all the nodes in these subgraphs. If all maximally connected subgraphs have even numbers of nodes, then we terminate the procedure at this step.

2. We re-index the maximally connected subgraphs with odd numbers of nodes as $\mathcal{G}_1, ...\mathcal{G}_{2s}$, for some $s \in \mathbb{N}$. We note that the total number, $2s$, is indeed even since the total number of the remaining nodes is even and the number of nodes in each subgraph is odd. For $1 \leq i \leq 2s$, let g_i be a node in $\mathcal{G}_i$ that connects to k (we commented above on its existence). We form the following s pairs: $(g_1, g_2), (g_3, g_4), ...(g_{2s-1}, g_{2s})$. We note that they form an eligible set since they all connect to k. This eligible set only uses the nodes $\{g_i\}_{i=1}^{2s}$. We terminate the procedure at this step whenever all maximally connected subgraphs with an odd number of nodes only contain a single node (so that all nodes of these subgraphs are used by this eligible set).

3. If $\mathcal{G}_i \neq (\{g_i\}, \emptyset)$ and the number of nodes of $\mathcal{G}_i$ is odd, then we remove g_i from $\mathcal{G}_i$ and also remove all the edges connected to g_i. The remaining graph contains an even number of nodes. If the remaining graph is connected, then we find an eligible set that uses all nodes in this remaining subgraph (its existence follows from the induction assumption). We terminate the procedure at this step if each of these remaining subgraphs is connected (the union of all such eligible sets then form a larger eligible set that uses all nodes in these subgraphs).

4. If a remaining subgraph from the above step (having g_i and its connected edges removed from $\mathcal{G}_i$) is not connected, we form its maximally connected subgraphs $\mathcal{H}_1^{(i)}, \ldots \mathcal{H}_{l_i}^{(i)}$. We note that $\mathcal{H}_1^{(i)}, \ldots \mathcal{H}_{l_i}^{(i)}$ are all connected to g_i (since $\mathcal{G}_i$ is connected and $\mathcal{H}_1, \ldots \mathcal{H}_{m_i}$ are not connected to each other). We also observe that $\cup_{r=1}^{l_i} \mathcal{H}_r^{(i)}$ contains less than $2K-2$ nodes (indeed the number of nodes of $\mathcal{G}_i$ is strictly less than the number of nodes in $\cup_{r=1}^{m} \mathcal{G}_r$, which is $2K-2$; we remark

that since g_i was removed from $\mathcal{G}_i$ the bound $2K-2$ is not tight).

If this procedure terminates in its first three steps, then it finds an eligible set that uses all nodes in the maximally connected subgraphs and thus its size is $2K-2$. Otherwise, we iteratively apply the same four-steps procedure on the maximally connected subgraphs of the last step. At each iteration the total number of nodes in the maximally connected subgraphs reduces (we clarified this at the end of the fourth step). Since the graph is finite, the iteration either terminates or $\cup_{r=1}^{l_i}\mathcal{H}_r^{(i)}$ in step 4 of the procedure is of size 2, that is, there are two subgraphs with single nodes. When inputting these single nodes at the next iteration, the procedure will terminate at step 2.

The final eligible set is the union of all the eligible sets iteratively generated from steps (1)-(3) and the pair (j,k). This eligible set uses all the nodes in V^o and thus it contains K pairs.

We remark that we introduced Case 1 in order to help the reader master the idea of the proof in a simpler case. We actually demonstrated how to consecutively handle the case where a single node is connected to maximally connected subgraphs whose total number of nodes is even. Starting with a node of degree 1 allowed us to proceed with this idea in one direction. Next, we pick up two different points and proceed with this idea in two different directions and, in fact, we could have started the proof with the latter setting right away. This setting has two subcases (2A and 2B). In Case 2A, we still have an even number of nodes in the maximally connected subgraphs, which are connected to a single node, so the ideas of Case 1 immediately apply. In Case 2B, the latter number of nodes is odd, but we can somehow reduce it to Case 2A.

Case 2: There does not exist any node in V^o with degree 1. In this case we randomly select two neighboring nodes $j,k \in V^o$. We consider the remaining graph after removing these two nodes from V^o and also remove all edges connected to them from E^o. If the remaining graph is connected or if all the maximally connected

subgraphs of the remaining graph contain an even number of nodes, then the induction assumption concludes the proof. Otherwise, the remaining graph is partitioned into maximally connected subgraphs $\mathcal{G}_1, ..., \mathcal{G}_{n+m}$. We reindex these subgraphs so $\mathcal{G}_1, ...\mathcal{G}_m$ are all connected to node j and not connected to node k in $\mathcal{G}^o$. The other maximally connected subgraphs, $\mathcal{G}_{m+1}, ..\mathcal{G}_{m+n}$, are connected to k in $\mathcal{G}^o$ (they may or may not be connected to j). We prove this case by further considering two different scenarios:

Case 2A: $\cup_{i=1}^{m}\mathcal{G}_i$ contains an even number of nodes. In this case, it is clear that $\cup_{i=m+1}^{n+m}\mathcal{G}_i$ also contains an even number of nodes. We could iteratively perform the above four-steps procedure introduced in Case 1 on $\{\mathcal{G}_i\}_{i=m+1}^{n+m}$ (as they all connect to k) and on $\{\mathcal{G}_i\}_{i=1}^{m}$ (as they all connect to j). Following the same argument at the end of the proof of case 1, we obtain two eligible sets that cover $\{\mathcal{G}_i\}_{i=m+1}^{n+m}$ and $\{\mathcal{G}_i\}_{i=1}^{m}$, respectively. The union of these two sets and the pair (j, k) yields an eligible set that uses all nodes in $\mathcal{G}^o$, which concludes the proof.

Case 2B: $\cup_{i=1}^{m}\mathcal{G}_i$ contains an odd number of nodes. We note that $\cup_{i=m+1}^{n+m}\mathcal{G}_i$ contains an odd number of nodes. We further note that within $\{\mathcal{G}_i\}_{i=1}^{m}$, there is an odd number of subgraphs that contain an odd number of nodes. After reindexing, these subgraphs are $\mathcal{G}_1, \mathcal{G}_2, ...\mathcal{G}_{2r+1}$, where $2r + 1 \leq m$. We pick one node from $\mathcal{G}_{2r+1}$ that is connected to j and denote it by l. By definition, l is not connected to $\mathcal{G}_i$, $i \in [2r]$ or $i = 2r + 2, \ldots m$, and also not connected to node k.Using this special node l, we redefine the "remaining graph" that was described in the beginning of case 2. That is, we remove from $\mathcal{G}^o$ the pair (j, l) (and the associated edges), whereas before the pair (j, k) was removed from it. We can similarly identify maximally connected subgraphs of this new remaining graph. Now the collection of the maximally connected subgraphs that connect to j but not connect to l contains an even number of nodes in total. Indeed, this collection contains all nodes in the former subgraphs, $\mathcal{G}_1, \ldots \mathcal{G}_{2r}$, $\mathcal{G}_{2r+2}, \ldots, \mathcal{G}_{m+n}$ and also the node k and there is an odd number of nodes in every $\mathcal{G}_i$, $i = 1, ...2r$, an even number of nodes in every $\mathcal{G}_i$, $i = 2r + 2, \ldots m$ and an odd number

of nodes in $\cup_{i=m+1}^{n+m}\mathcal{G}_i$ and the single node k, which yield an even number of total nodes. Note that we transformed the case to the one in case 2A (with the initially selected pair (j,l)) and the proof is thus concluded. □

Lemma B.3

For a connected graph $\mathcal{G}^o$ with $2K+1$ nodes, there exists an eligible set S containing K pairs of nodes in $\mathcal{G}^o$ and the pooling process with respect to S maps this graph to a graph with $K+1$ nodes. □

Proof: We note that there exists a node in $\mathcal{G}^o$ so that the remaining graph is still connected after removing this node from V^o and removing all edges connected to this node from E^o. Indeed, it can be selected as a node with degree 1 in the spanning tree of $\mathcal{G}^o$. Considering the remaining graph with $2K$ nodes (after removing this node and the associated edges), Lemma B.2 implies the existence of an eligible set S containing K pairs of nodes so that the pooling process with respect to S will map $\mathcal{G}^o$ to a graph with $K+1$ nodes.

□

Proof of Theorem 2. To conclude the theorem, we just need to show that for $\mathcal{G}^o=(V^o,E^o)$ with $|V^o|=N$, and any number $K\in[\lceil N/2\rceil, N-1]$, there exists a pooling process with an eligible set S (containing $N-K$ pairs) that maps $\mathcal{G}^o$ to a graph $\mathcal{G}$ with K nodes. It is sufficient to prove the statement for $K=\lceil N/2\rceil$ with the eligible set S^* that contains $N-\lceil N/2\rceil$ pairs. Indeed, for any $K>\lceil N/2\rceil$, we can take a subset S_K of the eligible set S^* (containing $N-\lceil N/2\rceil$ pairs) such that $|S_K|=|S^*|-(K-\lceil N/2\rceil)$. The pooling process with respect to S_K produces a graph with K nodes.

Given a graph $\mathcal{G}^o$ with N nodes, Lemma B.2 and Lemma B.3 imply that there exists an eligible set S^* with $\lfloor N/2\rfloor$ pairs. Therefore, the pooling procedure with respect to S^* maps $\mathcal{G}^o$ to a graph $\mathcal{G}$ with $\lceil N/2\rceil$ nodes.

B.4.3 Proof of Corollary 3

For any connected graph $\mathcal{G}^o$ with N nodes, iterative application of Theorem 2 implies the existence of a series of unpooling layers U_k, $k \geq 1$, and intermediate graphs $\mathcal{G}_k$, $k \geq 1$, with $|\mathcal{G}_k| = \lceil |\mathcal{G}_{k-1}|/2 \rceil$, so that $U_k(\mathcal{G}_k) = \mathcal{G}_{k-1}$, where $\mathcal{G}_0 := \mathcal{G}^o$. We stop the process once $|\mathcal{G}_k| \in [4, 6]$ (we will reach this range because for any $N' > 6$, $\lceil N'/2 \rceil \geq 4$). Another application of Theorem 2 yields the existence of a 3-nodes graph, $\mathcal{G}$, and a last unpooling layer, U_{k+1}, so that $U_{k+1}(\mathcal{G}) = \mathcal{G}_k$.

That is, there exist a 3-nodes graph $\mathcal{G}$ and a series of unpooling layers $U_1, ..U_{k+1}$ so that $U_1 \circ U_2 \circ ... \circ U_{k+1}(\mathcal{G}) = \mathcal{G}^o$. Note that the number of unpooling layers is $k + 1 = \lceil \log_2(N/3) \rceil$.

B.5 Optimizing Specific Chemical Properties

In this task, we consider the following three chemical properties (Guimaraes et al., 2017), which we evaluate only on the valid molecules among the 10,000 generated ones: druglikeliness, solubility and synthesizability. We calculate the first property using the Quantitative Estimate of Druglikeness (QED) package in RDKit (`https://www.rdkit.org/`) (licensed by BSD 3-Clause). These scores lie in $[0, 1]$ and their values aim to express the likelihood of being a drug. We calculate solubility by the log octanol-water partition coefficient using the Crippen package in RDKit. We rescale this value to lie in $[0, 1]$, where 1 is the most soluble value. In order to calculate synthesizability, we calculate the synthetic accessibility (Ertl and Schuffenhauer, 2009) and rescale this value to lie in $[0, 1]$ where 1 is the easiest to synthesize. We use codes from `https://github.com/connorcoley/scscore` , licensed by MIT.

For the generative model, we follow with the same Wasserstein GAN architecture as described in §B.3 (with the final activation function to be a sigmoid function), but the discriminator minimizes the error between its output and the objective chemical

property; it then outputs a reward score, which we need to maximize when training the generator.

Result of optimizing chemical properties. Using QM9, we generated molecules that aimed to maximize druglikeliness, solubility and synthesizability. Table B.1 reports the six evaluation metrics (listed in its columns, whereas the properties we aimed to maximize are in its rows). In terms of generating molecules with the targeted chemical properties, UL GAN is competitive with the other approaches (this is noticed when looking at columns 4, 5, 6 of rows 1, 2, 3, respectively. When considering the other evaluation metrics, UL GAN generally outperforms Adj GAN, except for the objective of druglikeliness and the metric of synthesizability (first row and sixth column). Note that the uniqueness of UL GAN and Adj GAN is very low because our generator does not aim to compete with the discriminator, but to generate molecules with a maximal property of interest. We did not report the good performance of UL GAN when considering this task with ZINC, since the other methods we compared with were not tested on ZINC; furthermore, the superiority of UL GAN over Adj GAN for ZINC is already obvious from Table 4.

Table B.1: The six evaluation metrics (in rows) for generated samples that aim to minimize the three indicated chemical properties (in columns). We remark that QED is the acronym for quantitative estimate of druglikeliness. Scores for competing methods (above the indicated line) were copied from their original papers. NA means that the score is not available in the original papers. Syn-ability refers to Synthesizability.

Objective	Method	Valid	Unique	Novel	QED	Solubility	Syn-ability
QED	ORWGAN	0.882	0.694	NA	0.52	0.35	0.32
	Naive RL	0.971	0.540	NA	0.57	0.50	0.53
	MolGAN	1.00	0.022	NA	**0.62**	0.59	0.53
	Adj GAN	0.991	0.005	0.865	0.443	0.288	0.658
	UL GAN	0.9888	0.051	0.978	0.598	0.497	0.485
Solubility	ORWGAN	0.965	0.459	NA	0.50	0.55	0.63
	Naive RL	0.927	1.00	NA	0.49	0.78	0.70
	MolGAN	0.998	0.002	NA	0.44	**0.89**	0.22
	Adj GAN	0.940	0.003	0.958	0.378	0.367	0.007
	UL GAN	0.993	0.010	0.781	0.507	0.700	0.793
Syn-ability	ORWGAN	0.965	0.459	NA	0.51	0.45	0.83
	Naive RL	0.977	0.136	NA	0.52	0.46	0.83
	MolGAN	1.00	0.021	NA	0.53	0.68	0.95
	Adj GAN	0.999	0.003	0.833	0.360	0.331	0.835
	UL GAN	1.00	0.006	0.433	0.468	0.569	**0.953**

B.6 Additional Numerical Results

We supplement the numerical results in §3.4. Section B.6.1 reports standard deviations for the earlier experiments on QM9 and ZINC; §B.6.2 includes generation results with only 1,000 samples; and §B.6.3 demonstrates examples of the generated molecules from UL GAN and UL VAE.

B.6.1 Standard Deviations for Molecule Generation

Table B.2 supplements Table 3 in the main manuscript and reports the means and standard deviations of the evaluation metrics for our methods in QM9, including Adj

GAN, UL GAN, and UL VAE. Table B.3 supplements Table 4 in the main manuscript and reports the means and standard deviations of the evaluation metrics for our methods in ZINC, including UL GAN and Adj GAN and UL VAE. These means and standard deviations are calculated from 100 runs.

Table B.2: Validity, uniqueness and novelty with standard deviation for Adj GAN, UL GAN, and UL VAE using QM9.

Method	Valid	Unique	Novel
UL VAE	0.735 (± 0.004)	0.940 (± 0.003)	0.949 (± 0.002)
Adj GAN	0.941 (± 0.002)	0.139 (± 0.002)	0.886 (± 0.006)
UL GAN	0.907 (± 0.003)	0.826 (± 0.004)	0.949 (± 0.002)

Table B.3: Validity, Uniqueness and Novelty with standard deviation for Adj GAN and UL GAN using ZINC.

Method	Valid	Unique	Novel
Adj GAN	0.109 (± 0.003)	0.196 (± 0.011)	1.00 (± 0)
UL GAN	0.871 (± 0.004)	1.00 (± 0)	1.00 (± 0)

B.6.2 Evaluation with only 1,000 generated samples.

In the experiments of §3.4, we generated 10,000 samples and reported statistics based on these samples. To check whether the performance is preserved for a smaller sample, we report here results of UL GAN and UL VAE when generating only 1,000 samples. Table B.4 reports such results for the Waxman and protein dataset and Table B.5 reports such results for QM9 and ZINC datasets. We note that all the reported metrics, but the uniqueness in QM9, are similar to the ones in Table 3.1, Table 3.2, Table 3.3 and Table 3.4, where 10k samples were generated. The uniqueness in QM9

Table B.4: Model performance with 1,000 generated samples on Waxman and protein datasets.

	Waxman random graph							
Method	kl e ds	kl clst	kl cn	kl n ft	wd e ds	wd clst	wd cn	wd n ft
UL VAE	0.115	0.443	0.279	0.451	0.009	0.100	0.106	0.126
UL GAN	0.007	0.030	0.033	0.152	0.002	0.022	0.012	0.039
	Protein dataset							
	kl e ds	kl clst	kl cn	kl n ft	wd e ds	wd clst	wd cn	wd n ft
UL VAE	0.565	1.235	0.822	0.260	0.034	0.071	0.319	4.400
UL GAN	0.084	0.484	0.136	0.234	0.014	0.011	0.115	3.958

Table B.5: Model performance with 1,000 generated samples on QM9 and ZINC datasets.

	QM9		
Method	Valid	Unique	Novel
UL VAE	0.737	0.991	0.932
UL GAN	0.905	0.970	0.927
	ZINC		
	Valid	Unique	Novel
UL VAE	NA	NA	NA
UL GAN	0.870	1.00	1.00

is significantly higher with 1,000 samples. Polykovskiy et al. (2020) also noticed a higher uniqueness rate with 1,000 samples than with 10,000 samples.

B.6.3 Synthetic Samples for Molecule Generation

We demonstrate samples generated from the UL GAN for QM9 in Figure B.5 and samples generated from the UL VAE in Figure B.6. Also, we present samples generated from the UL GAN for ZINC in Figure B.7.

We also illustrate some examples of the evolution of intermediate graphs and illustrate how a graph is generated from a generative GNN using unpooling layers, in Figure B.8 for QM9 and in Figure B.9 for ZINC.

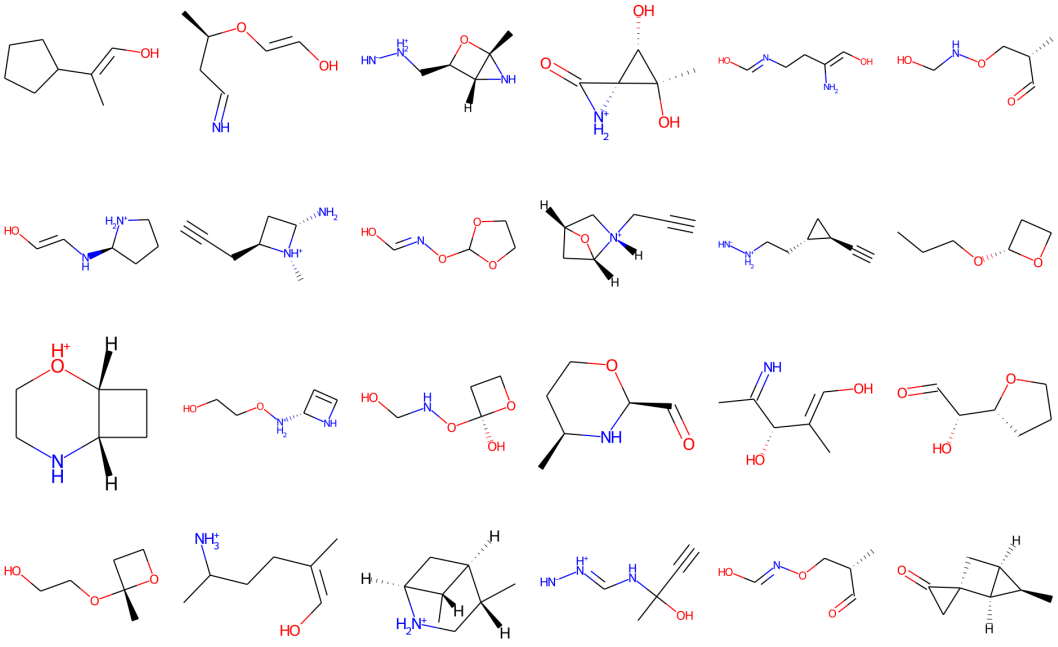

Figure B.5: Samples of molecules generated by UL GAN based on QM9 dataset.

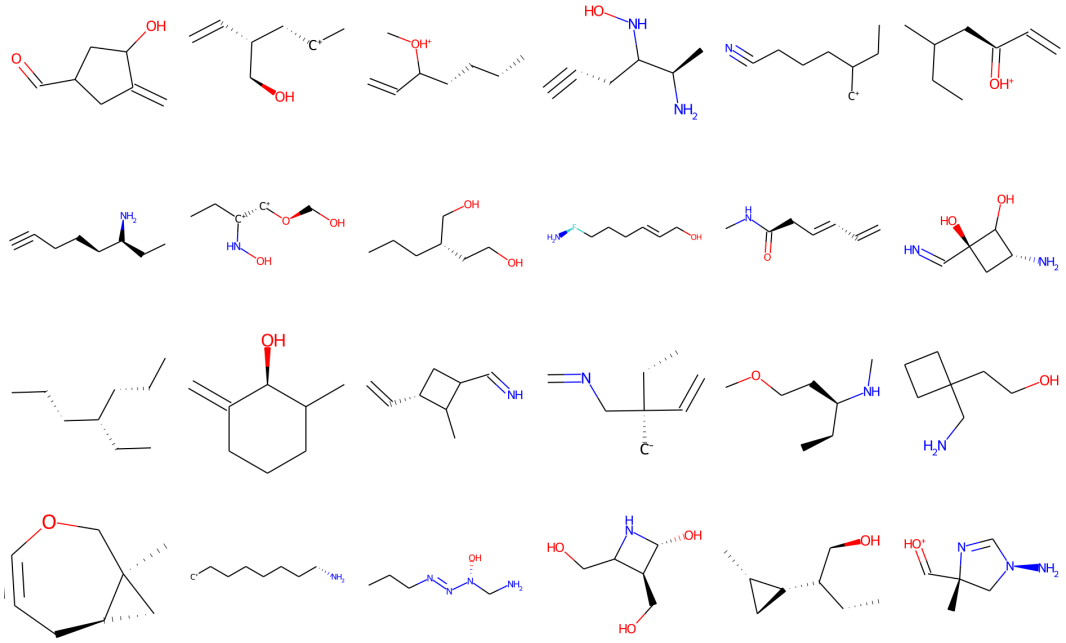

Figure B.6: Samples of molecules generated from UL VAE based on QM9 dataset.

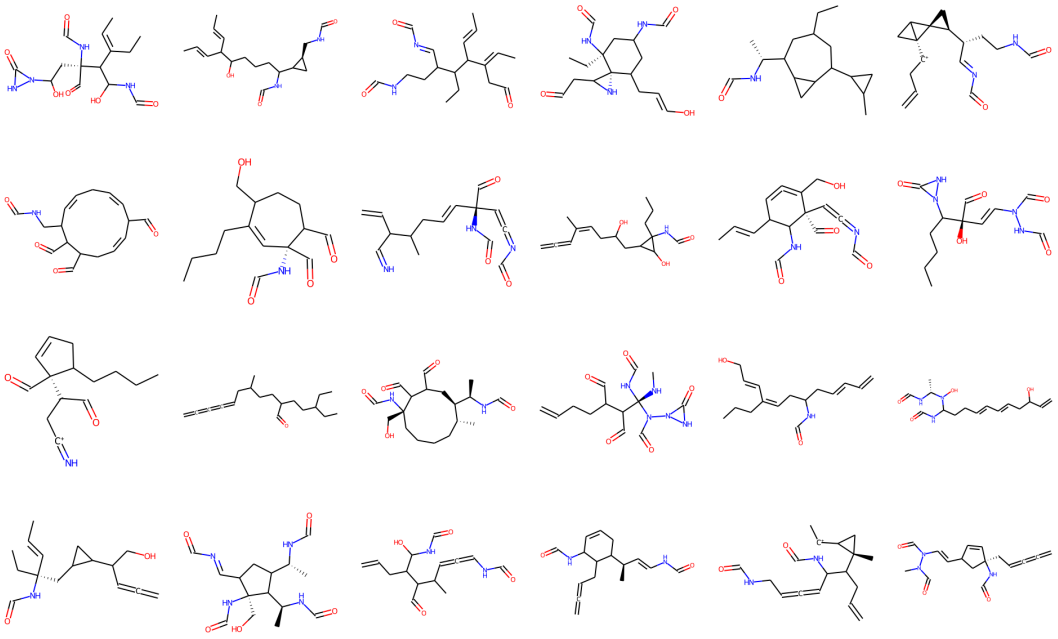

Figure B.7: Samples of molecules generated from UL GAN based on ZINC dataset.

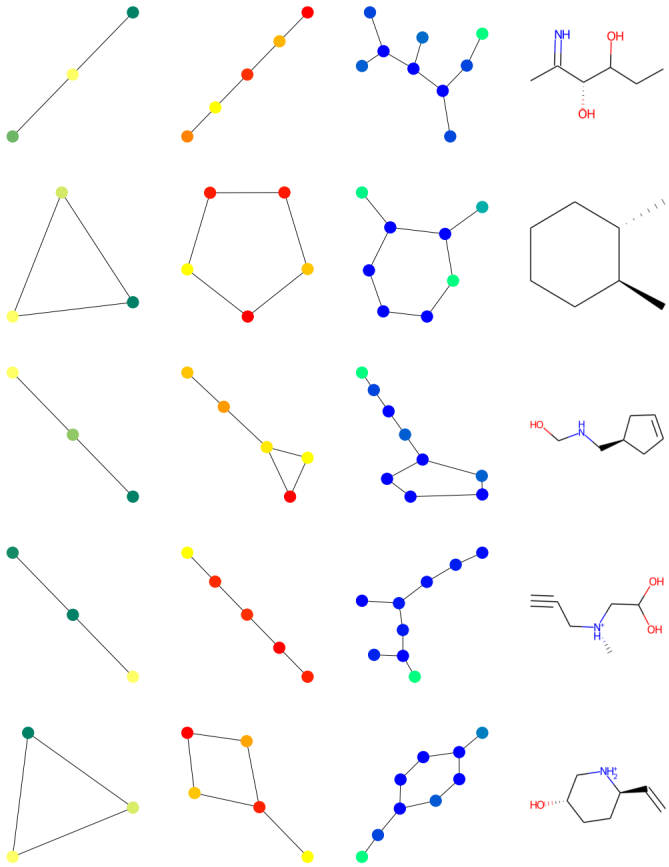

Figure B.8: Five examples of generated graphs using UL GAN trained with QM9. Each row represents one example, showing intermediate graphs in the generation process. Left column: initial 3-nodes graph; Middle 2–3 columns: intermediate graphs after unpooling layers; Right column: the final generated molecule. The color represents one dimension of the node features.

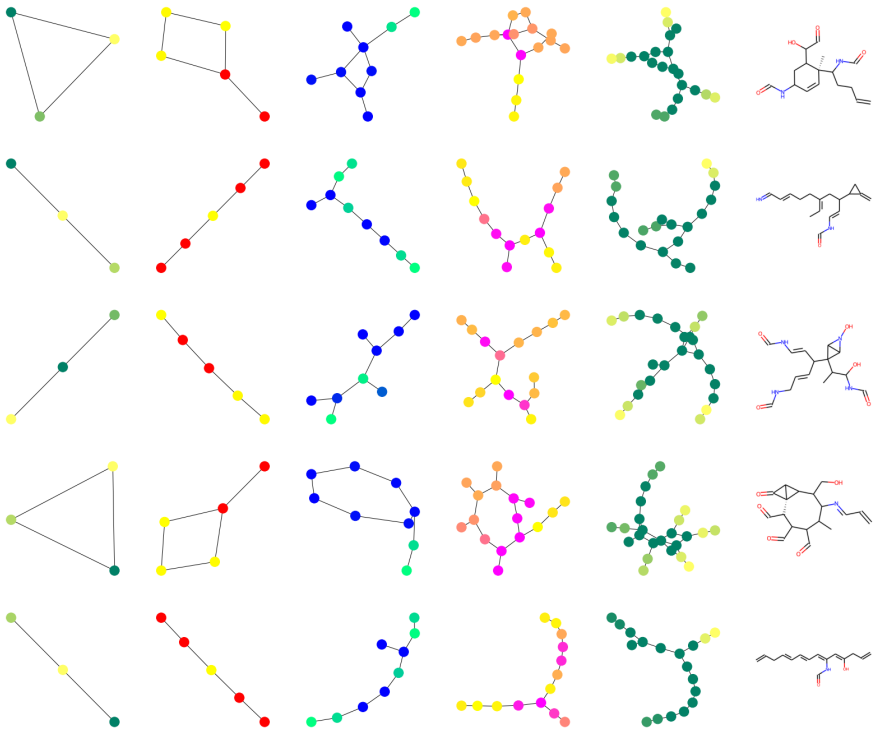

Figure B.9: Five examples of generated graphs using UL GAN trained with ZINC dataset. Each row represents one example, showing intermediate graphs in the generation process. Left column: initial 3-nodes graph; Middle 2–5 columns: intermediate graphs after unpooling layers; Right column: the final generated molecule. The color represents one dimension of the node features.

Appendix C

Supplementary Material for Chapter 4

Section C.1 presents sensitivity analysis and exemplifies convergence paths. Section C.2 provides extra details on how to fit the univariate and bivariate functions of equation (4.19). Section C.3 provides additional numerical experiments on the dependence of the collusive level on β_k and $u_k^{(0)}$.

C.1 Sensitivity Analysis and Examples of Convergence Paths

We assume special parameterizations of the network externality matrix, Φ, and perform a sensitivity analysis for our simulation framework. For each specific Φ, we ran 500 simulations and classify the behavior in the last 5000 time steps of each run into one of five categories:

1. Symmetric 1-cycle (or 1-Sym), where platforms 1 and 2 repeatedly choose actions p_1 and p_2, respectively, and $p_1 = p_2$.

2. Asymmetric 1-cycle (or 1-Asym), where platforms 1 and 2 repeatedly choose actions p_1 and p_2, respectively, and $p_1 \neq p_2$.

3. Cycle of length 2-4 (or C2-4), where platforms 1 and 2 repeatedly choose actions that follow a pattern of length two to four. For example, the actions in a cycle of length 3 are $\{(p_1, p_2), (q_1, q_2), (r_1, r_2)\}$.

4. Cycle of length 5-8 (or C5-8), where the two platforms repeatedly choose actions that follow a pattern of length five to eight.

5. Cycle of length at least 9 (or C9), where the two platforms repeatedly choose actions that follow a pattern of length at least nine.

Let $k \in \{1, \ldots, 5\}$ index the five categories described above, and C_k denote the set of simulations within category k. For each $k \in \{1, \ldots, 5\}$ and a chosen Φ, we report the following measures:

- Frequency (Freq.) of the category among the 500 simulations.

- Average (Avg.) and standard deviation (S.D.) of the collusive level $\tilde{\Delta}$ within the category.

- Frequency of equilibrium, which quantifies the proportion of simulations in which the AI driven platforms learn the action that achieves maximal total future reward given that the opponent plays with the final Q-function. We compute this frequency, denoted by $Y_k \in [0, 1]$ for $k \in \{1, \ldots, 5\}$, according to the following three steps:

  1. Let Q_i^* be the final Q-function of platform $i \in \{1, 2\}$ in a given simulation $s \in C_k$. It is the numerical approximation to Q_i^* defined in (4.10), as explained in Section 4.3.2. We use this notation in defining other frequencies below. We estimate, without loss of generality, platform's 1 best response to Q_2^* using backward induction. For $\hat{T} = 10$, initiate

$$Q_1^{(\hat{T})}(p, x) = \pi^{(1)}(p, \arg\max_{p'} Q_2^*(p', x)),$$

  where p is a vector of prices and x is the state variable.

For $t \in \{1, \ldots, \hat{T}\}$, compute via backward induction

$$Q_1^{(t-1)}(p, x) = \pi^{(1)}(p, \arg\max_{p'} Q_2^*(p', x)) + \delta \max_{p''} Q_1^{(t)}(p'', x'),$$

where $x' = (p, \arg\max_{p'} Q_2^*(p', x))$. The matrix $Q_1^{(0)}$ is used as an approximation to platform's 1 best response to Q_2^* in the given simulation $s \in C_k$.

2. Let P_s denote the set of states used in the convergence path of Q_i^* in the given simulation $s \in C_k$. We use this definition in defining the other frequencies below. For each state $(p_1, p_2) \in P_s$, we form an indicator variable $X_s(p_1, p_2)$ that checks if the action that platform 1 takes using Q_1^* approximates the action that it would take using $Q_1^{(0)}$. Thus, for each $(p_1, p_2) \in P_s$, let $p^0 := \arg\max_{p'} Q_1^{(0)}(p', (p_1, p_2))$ and $p^* := \arg\max_{p'} Q_1^*(p', (p_1, p_2))$, then $X_s(p_1, p_2) = 1$ if and only if either $p^0 = p^*$ or p^0 and p^* are neighbors with distance at most 1 in the set of indices of the Q-matrix function.
3. Let $Y_k = \frac{1}{|C_k|} \sum_{s \in C_k} \frac{1}{|P_s|} \sum_{(p_1, p_2) \in P_s} X_s(p_1, p_2)$.

- Frequency of one-step equilibrium, which quantifies the proportion of simulations in which the AI driven platforms learn the action that achieves maximal one-step reward given that the opponent plays with the final Q-function. Note that the underlying learning involves no memory. We compute this frequency, denoted by $Y_k^{(\text{one})} \in [0, 1]$ for $k \in \{1, \ldots, 5\}$, according to the following three steps:

1. Let $Q_i^{(\text{one})}(p, x) = \pi^{(1)}(p, \arg\max_{p'} Q_2^*(p', x))$.
2. Let $p^{(\text{one})} := \arg\max_{p'} Q_1^{(\text{one})}(p', (p_1, p_2))$. We define $X_s^{(\text{one})}(p_1, p_2) = 1$ if and only if either $p^{(\text{one})} = p^*$ or $p^{(\text{one})}$ and p^* are neighbors with distance at most 1 in the set of indices of the Q-matrix function.
3. Let $Y_k^{(\text{one})} = \frac{1}{|C_k|} \sum_{s \in C_k} \frac{1}{|P_s|} \sum_{(p_1, p_2) \in P_s} X_s^{(\text{one})}(p_1, p_2)$.

- Frequency of converging back to the limiting action, which quantifies the proportion of simulations in which after one platform unilaterally deviates to the one-stage Nash equilibrium price, platforms return back to the limiting action. We compute this frequency, $Y_k^{(\mathrm{b})} \in [0,1]$ for $k \in \{1,\dots,5\}$, according to the following three steps:

  1. At time $T+1$, set $p_{T+1}^{(1)} = (p_b^*, p_s^*)$, while $p_{T+1}^{(2)} \in \arg\max_p Q_2^*(p,(p_T^{(1)}, p_T^{(2)}))$. For $\tau \in \{2,\dots,101\}$ and $i \in \{1,2\}$, let $p_{T+\tau}^{(i)} \in \arg\max_p Q_i^*(p,(p_{T+\tau-1}^{(1)}, p_{T+\tau-1}^{(2)}))$.
  2. Let $p^{(\mathrm{b}),(i)} := p_{T+101}^{(i)}$. We define $X_s^{(\mathrm{b})}(p^{(\mathrm{b}),(1)}, p^{(\mathrm{b}),(2)}) = 1$ if and only if $(p^{(\mathrm{b}),(1)}, p^{(\mathrm{b}),(2)}) \in P_s$.
  3. Let $Y_k^{(\mathrm{b})} = \frac{1}{|C_k|}\sum_{s\in C_k} X_s^{(\mathrm{b})}(p^{(\mathrm{b}),(1)}, p^{(\mathrm{b}),(2)})$.

- Frequency of converging back to the limiting action for both platforms, which quantifies the proportion of simulations in which after both platforms deviate to the one-stage Nash equilibrium price, they return back to the limiting action. This frequency is computed similarly to the one above, where the only difference is that $p_{T+1}^{(1)} = p_{T+1}^{(2)} = (p_b^*, p_s^*)$.

- Average and standard deviation of the Q-loss w.r.t. P_s. The Q-loss quantifies how close are the observed rewards on path P_s, $\sum_{\tau=0}^{100}\delta^\tau \pi_{t+\tau}^{(i)}$, to the optimal rewards given by the best response $Q_i^{(0)}$. We highlight that after convergence, the observed rewards should be pretty close in value to $\max_{p\in\mathcal{A}} Q_i^*(p,x)$ for each state $x \in P$. Let $T_0 = T - 5000$, we compute the Q-loss for a given simulation s as

$$\frac{1}{|P_s|}\sum_{t=T_0}^{T_0+|P_s|}\left|\sum_{\tau=0}^{100}\delta^\tau \pi_{t+\tau}^{(i)} - \max_{p\in\mathcal{A}} Q_i^0(p,x_t)\right|.$$

  The average and standard deviation are taken w.r.t. all simulations $s \in C_K$ for any $k \in \{1,\dots,5\}$.

- Average and standard deviation of the Q-loss w.r.t. all states. The Q-loss w.r.t. all states quantifies in average how close are the maximum values of $Q_i^*(\cdot, x)$ and $Q_i^0(\cdot, x)$, where the maximum is taken over all actions and the average over all states $x \in \mathcal{S}$. We compute the Q-loss w.r.t. all states for a given simulation s as
$$\frac{1}{|\mathcal{S}|} \sum_{x \in \mathcal{S}} \left| \max_{p \in \mathcal{A}} Q_i^*(p, x) - \max_{p \in \mathcal{A}} Q_i^0(p, x) \right|.$$
The average and standard deviation are taken with respect to all simulations $s \in C_K$ for any $k \in \{1, \ldots, 5\}$. When the Q-loss w.r.t. all states equals 0, platforms' maximal rewards of the final Q-function yield the same maximal rewards of the best response Q-function. We interpret this as follows: the final Q-function exhibits behavior consistent with equilibrium off-path, also known as subgame perfection.

Tables C.1, C.2 and C.3 show the above measures for the following three respective choices of Φ: $\Phi_1 := [0, 0; 0, 0]$, $\Phi_2 := [1, 0; 0, 1]$ and $\Phi_3 := [0, 1; 1, 0]$. The other parameters are set as in the default setting, i.e., $\delta = 0.05$, $\beta_k = 1$, $u_k^{(0)} = -2$.

In all tables, the frequency of symmetric 1-cycles is relatively small, where its average is 5.8%. Even though this event could be considered rare, this category is important since within it we can find one-memory stationary equilibria, which are of great interest (see, e.g., Barlo et al. (2016) and Chica et al. (2024)). Note that the frequency of asymmetric 1-cycles is very small, where its average is 2.7%. We will thus omit interpretations of results for this category. The average of both frequencies of cycles of length 2-4 and 5-8 is 34% and the one for cycles of length at least 9 is 23.3%.

The average collusive level among all tables and categories, excluding 1-Asy, is approximately 29%. This value is only slightly below the baseline collusive level Δ_0.

For 1-Sym, the frequencies of equilibrium are 46%, 28% and 52% for Φ_1, Φ_2 and Φ_3, respectively. Thus, approximately half of the simulations within 1-Sym result

in equilibrium behavior for Φ_1 and Φ_3, with a smaller value of 28% for Φ_2. These numbers suggest that in cases of zero-externalities or non-zero cross-side externalities, equilibrium behavior is more likely than in cases of non-zero within-side externalities. Note that for the frequency of one-step equilibrium in 1-Sym, we still observe a larger proportion of one-step equilibrium for Φ_1 and Φ_3 compared to Φ_2.

For the categories C2-4, C5-8 and C9, the average frequencies of equilibrium are 28.8%, 42.9% and 49.4% for Φ_1, Φ_2 and Φ_3, respectively. Thus, on average 46% of the simulations with cycle lengths greater than one result in equilibrium behavior for Φ_2 and Φ_3, with a smaller value of 28.8% for Φ_1. These numerical results suggest that in cases of non-zero externalities and for cycles of lengths greater than one, equilibrium behavior is more likely than in the zero-externality case. The above results for 1-Sym, C2-4, C5-8 and C9 suggest that in a sufficiently large percentage of cases, AI driven platforms exhibit behavior that is consistent with Nash equilibrium.

The frequencies of converging back to the limiting action for one platform in the 1-Sym category are 84%, 81% and 76% for Φ_1, Φ_2 and Φ_3, respectively. Thus, in at least 76% of the cases, platforms converge back to the limiting action after one unilateral price change to the Nash equilibrium price. For the categories C2-4, C5-8 and C9, the frequencies of converging back to the limiting action for one platform are 90%, 92% and 95% for Φ_1, Φ_2 and Φ_3, respectively. Note that the smallest frequency of converging back to the limiting action for one platform in the 1-Sym category is achieved at Φ_3, which suggests that non-zero cross-side externalities make it harder to sustain collusion. A similar result is shown by Ruhmer (2010) for a case of platform competition without AI agents. Nonetheless, this result does not hold for cycles of lengths greater than one, where the smallest frequency is achieved at Φ_1, i.e., at the zero-externalities case. As a general observation, the above results suggest that AI driven platforms can learn behavior that is consistent with tacit collusion. Note that similar behavior is found for the frequencies of converging back to the limiting action

for both platforms.

The average Q-losses (on path) for all tables and categories, excluding 1-Asy, are all less than 0.019. This observation and the definition of the Q-loss imply that the observed rewards on path are sufficiently close to the optimal rewards given by the best response matrix. These results suggest that the likelihood of the final action achieving maximal total future reward is very high. Similarly, the average Q-losses w.r.t. all states for all categories, excluding 1-Asy, are less than 0.043. Thus, platforms' maximal rewards of the final Q-function are relatively close to the maximal rewards of the best response Q-function. Furthermore, the Q-function exhibits behavior consistent with the equilibrium off-path or subgame perfection.

Metric	1-Sym	1-Asy	C2-4	C5-8	C9	All
Freq.	0.040	0.025	0.300	0.307	0.328	1.000
Avg. Collusive Level	0.273	0.326	0.294	0.301	0.295	0.297
S.D. of Collusive Level	0.082	0.069	0.063	0.049	0.042	0.054
Freq. of Eq.	0.462	0.0	0.279	0.296	0.29	0.288
Freq. of one-step Eq.	0.385	0.125	0.314	0.324	0.345	0.325
Freq. Conv. back (one)	0.846	0.750	0.907	0.909	0.896	0.898
Freq. Conv. back (both)	0.846	0.750	0.856	0.879	0.925	0.882
Avg. Q loss (on path)	0.003	0.006	0.005	0.005	0.005	0.005
S.D. of Q loss (on path)	0.001	0.004	0.003	0.002	0.001	0.002
Avg. Q loss (all)	0.009	0.009	0.009	0.009	0.009	0.009
S.D. of Q loss (all)	0.000	0.000	0.000	0.000	0.000	0.000

Table C.1: Sensitivity analysis of the numerical simulation with $\Phi = \Phi_1 := [0, 0; 0, 0]$.

Metric	1-Sym	1-Asy	C2-4	C5-8	C9	All
Freq.	0.084	0.013	0.355	0.368	0.179	1.000
Avg. Collusive Level	0.297	0.433	0.298	0.301	0.322	0.305
S.D. of Collusive Level	0.061	0.230	0.065	0.058	0.067	0.068
Freq. of Eq.	0.281	0.0	0.44	0.436	0.411	0.414
Freq. of one-step Eq.	0.063	0.000	0.366	0.326	0.320	0.313
Freq. Conv. back (one)	0.812	0.400	0.904	0.957	0.912	0.911
Freq. Conv. back (both)	0.812	0.400	0.889	0.943	0.897	0.897
Avg. Q loss (on path)	0.011	0.042	0.015	0.017	0.019	0.016
S.D. of Q loss (on path)	0.005	0.030	0.011	0.010	0.011	0.011
Avg. Q loss (all)	0.043	0.043	0.043	0.043	0.043	0.043
S.D. of Q loss (all)	0.001	0.002	0.001	0.001	0.001	0.001

Table C.2: Sensitivity analysis of the numerical simulation with $\Phi = \Phi_2 := [1, 0; 0, 1]$.

Metric	1-Sym	1-Asy	C2-4	C5-8	C9	All
Freq.	0.049	0.042	0.364	0.353	0.193	1.000
Avg. Collusive Level	0.264	0.282	0.279	0.265	0.287	0.275
S.D. of Collusive Level	0.057	0.079	0.075	0.053	0.061	0.065
Freq. of Eq.	0.524	0.336	0.533	0.499	0.451	0.497
Freq. of one-step Eq.	0.245	0.342	0.369	0.365	0.321	0.351
Freq. Conv. back (one)	0.762	0.500	0.930	0.967	0.976	0.926
Freq. Conv. back (both)	0.667	0.333	0.904	0.954	0.964	0.898
Avg. Q loss (on path)	0.009	0.019	0.013	0.012	0.015	0.013
S.D. of Q loss (on path)	0.004	0.010	0.009	0.006	0.009	0.008
Avg. Q loss (all)	0.038	0.038	0.038	0.038	0.038	0.038
S.D. of Q loss (all)	0.001	0.001	0.001	0.001	0.001	0.001

Table C.3: Sensitivity analysis of the numerical simulation with $\Phi = \Phi_3 := [0, 1; 1, 0]$.

Figure C.1 shows four examples of convergence paths, one from C2-4 (top left panel), one from C5-8 (top right panel) and two from C9 (bottom left and right panels), where $\Phi = [0, 1; 1, 0]$. The horizontal blue dashed lines at the bottom and top portion of each panel represent the Nash and Collusion equilibrium prices, respectively, i.e., the lines at $p_b^* = p_s^*$ and $p_b^{\mathrm{C}} = p_s^{\mathrm{C}}$, respectively. Note that buyer and seller prices are equal due to the symmetric choice of Φ. The vertical blue dotted lines represent where a cycle ends, while the blue and orange curves represent the buyer and seller prices, respectively, chosen by platform 1 on the equilibrium path. The top left panel shows a convergence path of length four, in which the buyer's price starts above the Nash equilibrium price. Then, in two steps, it reaches the Nash equilibrium price, followed by a sudden increase to the starting price. This price pattern is similar to an Edgeworth cycle, where prices start well above the Nash equilibrium price, then

slowly converge to the Nash equilibrium, followed by a sudden increase to the initial high price (see, .e.g., Maskin and Tirole (1988)). Note that the seller's price oscillates between two levels above the Nash equilibrium price. The top right and bottom left and right subfigures show more intricate patterns. However, a common feature among them is that prices oscillate between the Nash equilibrium price and a level higher than the Nash price. This indicates that platforms learn behavior consistent with punishment and reward strategies.

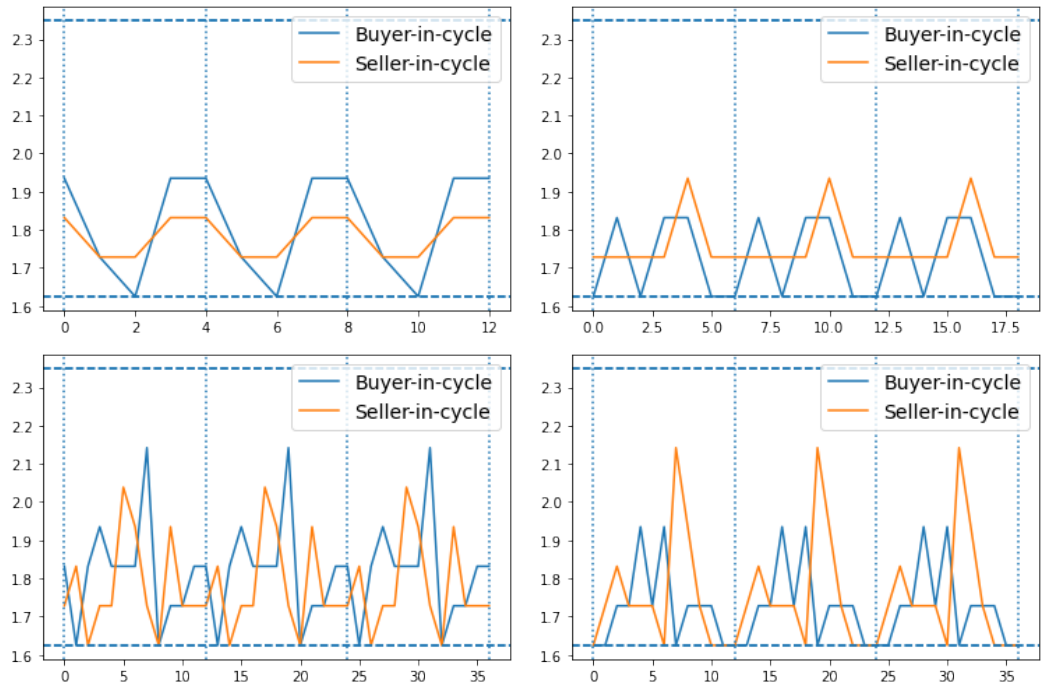

Figure C.1: Examples of convergence paths from categories C2-4 (top left panel), C5-8 (top right panel) and C9 (bottom left and right panels), where $\Phi = [0, 1; 1, 0]$. The horizontal blue dashed lines at the bottom and top of each panel represent the lines at $p_b^* = p_s^*$ and $p_b^C = p_s^C$, respectively. The vertical blue dotted lines indicate where a cycle ends. The blue and orange curves represent the buyer and seller prices, respectively, chosen by platform 1.

Finally, Figure C.2 illustrates four scenarios depicting the convergence path of platform 1 after both platforms revert to the limiting action following a deviation to

the one-stage Nash equilibrium price by both platforms. The top left and right panels present an example for Φ_1, while the bottom left and right panels provide examples for Φ_2 and Φ_3, respectively. In all four cases, the horizontal blue dashed lines at the bottom and top of each panel represent the lines at $p_b^* = p_s^*$ and $p_b^{\mathrm{C}} = p_s^{\mathrm{C}}$, respectively. The vertical blue dotted lines indicate where each cycle starts and ends. The blue and orange curves represent the buyer and seller prices, respectively, chosen by platform 1 immediately after the deviation to the Nash equilibrium price. The green and red curves represent platform 1's buyer and seller prices once they revert to the limiting action. In all four panels, we observe that in less than ten steps, platform 1's prices revert to the limiting action (the leftmost vertical dotted line appears before step 10). The top left panel shows that after one deviation to the Nash equilibrium price by both platforms, platform 1's prices slowly increase above the Nash equilibrium price, followed by a small decrease and then a convergence to a price above the Nash equilibrium price. Similar behavior is observed in the top right panel. The bottom left and right panels show more intricate patterns. However, as mentioned earlier, in both figures, prices converge back to the limiting action after 9 steps. The patterns they follow after reaching the limiting action indicate that firms oscillate between the Nash equilibrium price and a price higher than the Nash equilibrium price.

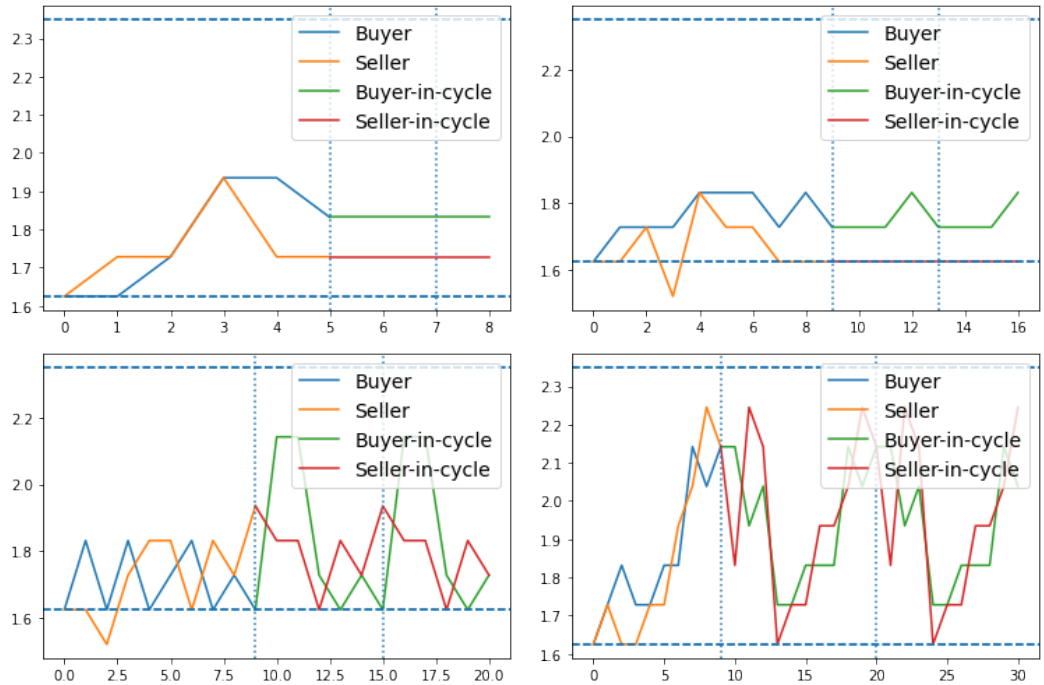

Figure C.2: Four examples of convergence paths after both platforms revert to the limiting action following a deviation to the one-stage Nash equilibrium price by both platforms. There are two examples for Φ_1 (top left and right panels), one for Φ_2 (bottom left), and one for Φ_3 (bottom right). The blue and orange curves represent the buyer and seller prices, respectively, chosen by platform 1 immediately after the deviation to the Nash equilibrium price. The green and red curves represent platform 1's buyer and seller prices once they revert to the limiting action.

C.2 Fitting a Sequence of Non-linear Functions

We provide the details of how to fit the univariate and bivariate functions in (4.19) in order to describe how the collusive level depends on the externality matrix Φ via an additive model. We first let Δ_0 equal the sample mean of $\tilde{\Delta}$. In order to find the four univariate functions $f_{bb}(\phi_{bb})$, $f_{ss}(\phi_{ss})$, $f_{bs}(\phi_{bs})$ and $f_{sb}(\phi_{sb})$, we fit them sequentially using residuals. We consider all the twenty-four permutations of 4 elements, represented by bb, ss, bs, sb. Without loss of generality, we describe the fitting procedure for a

given permutation $o := (bb, ss, bs, sb)$, where the same procedure is followed for all other permutations. We follow a regression setting with the predictor ϕ_{bb} and the response

$$y_{bb} := \tilde{\Delta} - \Delta_0,$$

and we apply XGBoost, which is a nonlinear regression method, to fit $\hat{f}_{bb}^{(o)}(\phi_{bb})$ that approximates y_{bb}. For the above o, we proceed from bb to ss as follows. We define

$$y_{ss} := y_{bb} - \hat{f}_{bb}^{(o)}(\phi_{bb})$$

and consider a regression setting with y_{ss} as response and ϕ_{ss} as predictor. We apply XGBoost to approximate y_{ss} by $\hat{f}_{ss}^{(o)}(\phi_{ss})$. Similarly, we transition from ss to bs using the residuals

$$y_{bs} := y_{ss} - \hat{f}_{ss}^{(o)}(\phi_{ss})$$

and fitting $\hat{f}_{bs}^{(o)}(\phi_{bs})$ to approximate y_{bs} by XGBoost, and we transition from bs to sb using the residual

$$y_{sb} := y_{bs} - \hat{f}_{bs}^{(o)}(\phi_{ss})$$

and fitting $\hat{f}_{sb}^{(o)}(\phi_{sb})$ to approximate y_{sb} by XGBoost. We repeat this process for the rest of the twenty-three permutations and thus obtain twenty-four approximation for the functions $f_{bb}(\phi_{bb})$, $f_{ss}(\phi_{ss})$, $f_{bs}(\phi_{bs})$ and $f_{sb}(\phi_{sb})$, specified in (4.19). We average over the twenty-four approximations to obtain the estimator $\hat{f}_{bb}(\phi_{bb}), \hat{f}_{ss}(\phi_{ss}), \hat{f}_{bs}(\phi_{bs})$ and $\hat{f}_{sb}(\phi_{sb})$.

Next, we apply the procedure described above for the six bivariate functions in (4.19), such as $f_{bb,ss}(\phi_{bb}, \phi_{ss})$ and $f_{bs,sb}(\phi_{bs}, \phi_{sb})$. For a given permutation of the six pairs of variables, e.g.,

$$o = ((bb, ss), (bs, sb), (bb, sb), (bb, bs), (ss, sb), (ss, bs)),$$

we follow the same procedure introduced above to iteratively fit the bivariate functions using XGBoost. In the first iteration, the response variable is

$$y_1 := \tilde{\Delta} - \Big(\Delta_0 + \hat{f}_{bb}(\phi_{bb}) + \hat{f}_{ss}(\phi_{ss}) + \hat{f}_{bs}(\phi_{bs}) + \hat{f}_{sb}(\phi_{sb})\Big)$$

and for the above permutation o there are two predictors ϕ_{bb} and ϕ_{ss}. XGBoost then approximates y_1 by the function $\hat{f}^{(o)}(\phi_{bb}, \phi_{ss})$. In the following iterations, the response is the corresponding residual and the predictors are the two corresponding variables. For example, assuming the permutation o, in the second iteration the response variable is

$$y_1 - \hat{f}^{(o)}(\phi_{bb}, \phi_{ss})$$

and the predictors are ϕ_{bs} and ϕ_{sb}. We average the obtained approximations over all 720 possible permutations of these six bivariate functions to approximate each bivariate function.

C.3 Additional Simulation Results

We provide Figures C.3 and C.4, which complement Figures 4.9 and 4.10, respectively, with additional choices of the externality matrix.

A common trend in both Figures 4.9 and C.3 is that the collusive level decreases as β_k increases.

Section 4.4.3 summarizes the main common observation to both Figure 4.10 and Figure C.4.

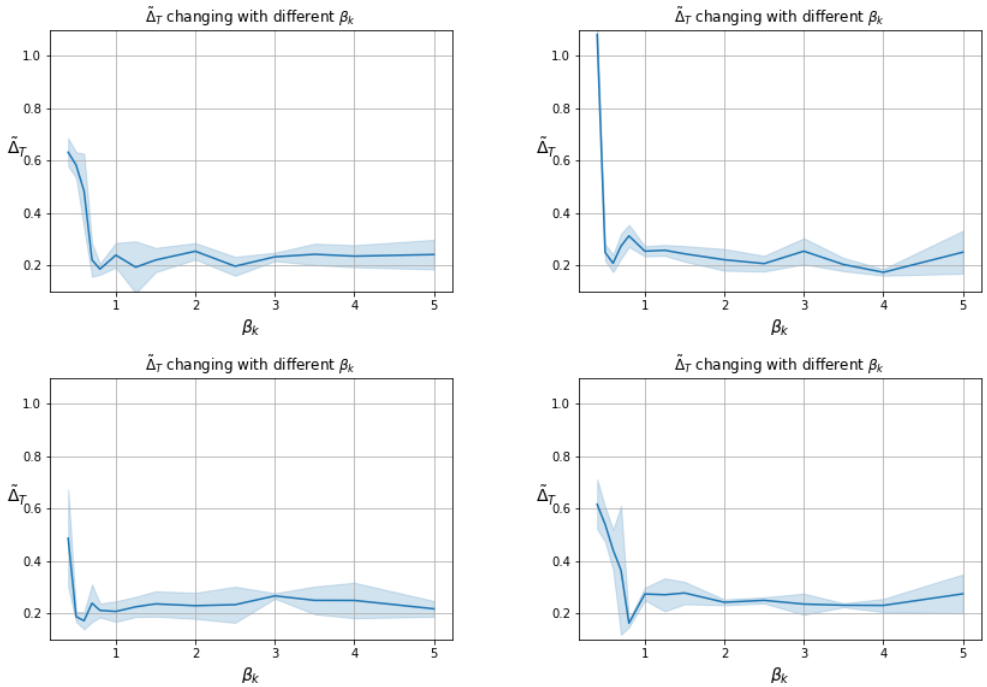

Figure C.3: The collusive level with varying $\beta_k \in [0.2, 5]$; Top left: $\Phi = [0, 1; 1, 0]$; Top right: $\Phi = [0, -1; -1, 0]$; Bottom left: $\Phi = [0, 1; 0, 0]$; Bottom right: $\Phi = [1, 0; 0, -1]$.

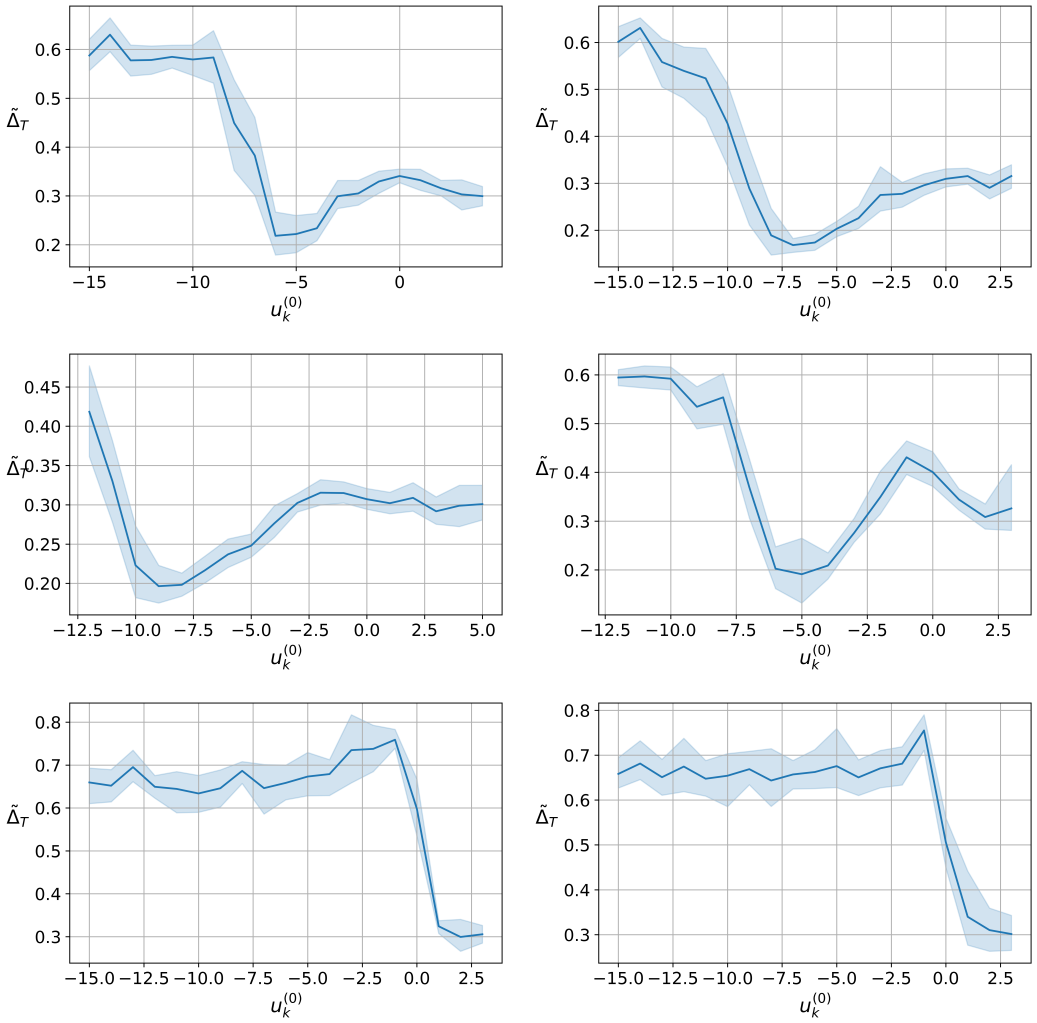

Figure C.4: The collusive level with varying $u_k^{(0)}$, for $k \in \{b, s\}$, where $\Phi = [1, 0; 0, 0]$ (top left), $\Phi = [0, 1; 0, 0]$ (top right), $\Phi = [0, 1; -1, 0]$ (middle left), $\Phi = [1, 0; 0, -1]$ (middle right), $\Phi = [2, 0; 0, 2]$ (bottom left), and $\Phi = [0, 2; 2, 0]$ (bottom right).