UNIVERSITY OF CALIFORNIA SAN DIEGO

Toward Understanding the Dynamics of Over-parameterized Neural Networks

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Libin Zhu

Committee in charge:

Professor Mikhail Belkin, Chair
Professor Alexander Cloninger
Professor Sanjoy Dasgupta
Professor Rose Yu

2024

The Dissertation of Libin Zhu is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2024

DEDICATION

To my family, for their constant support.

EPIGRAPH

The mystery of human existence lies not in just staying alive, but in finding something to live for.

Fyodor Dostoevsky, *The Brothers Karamazov*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I am most grateful to my advisor, Dr. Mikhail (Misha) Belkin for his unwavering guidance throughout every phase of my academic journey. Misha's involvement went beyond mere supervision; it was his insightful feedback and constant encouragement that significantly shaped my growth as a researcher. I consider myself incredibly lucky to have been influenced by his enthusiasm, broad vision for research, and unique research taste. Misha's dedication to pushing the boundaries of knowledge has inspired me to pursue my research with vigor and an open mind.

I would like to express my deep gratitude to my collaborators. A special thanks to Dr. Chaoyue Liu, whose invaluable assistance was crucial when I was starting my PhD, and whose insights have been a significant part of my research achievements. I am also grateful to my collaborators Dr. Adityanarayanan (Adit) Radhakrishnan, Dr. Parthe Pandit, Dr. Pedro Cisneros-Velarde, and Dr. Arindam Banerjee. Working with Adit on catapult dynamics projects, I greatly benefited from his resourceful and innovative ideas as well as his impressive presentation skills. My collaboration with Parthe on both teaching and the bottleneck network project was exceptionally rewarding, and I was impressed by his prolific insights and dedication to research. My work with Pedro and Arindam on the restricted strong convexity project allowed me to benefit significantly from their rigorous thinking and strong mathematical skills. It has been a privilege to work alongside such insightful individuals. Their commitment to research and willingness to share their expertise has profoundly enriched my PhD experience. I am also thankful for the numerous conversations I had with them about research and life.

I also want to thank my labmates and friends for their help during my PhD. I would like to thank Dr. Siyuan Ma, Dr. Like Hui, Neil Mallinar, Amirhesam Abedsoltan, Daniel Beaglehole, Parsa Mirtaheri, Dr. Zhichao Wang, Dr. Chen Cai, Dr. Yu Hao, Dr. Preetum Nakkiran, and Dr. Jonathan Shi for their valuable discussions about research and life with me. I am grateful to my internship mentor Dr. Yukun Chen at Meta, whose guidance made my internship experience both productive and enjoyable.

I sincerely thank Dr. Alexander Cloninger, Dr. Sanjoy Dasgupta and Dr. Rose Yu for serving on my dissertation committee. Their valuable feedback helped improve my dissertation significantly.

I would like to thank my family—my parents Huamin Shan, Yueping Zhu and my sister Lifen Zhu—for their constant support and love, which have been my cornerstone throughout my PhD journey. Their belief in me has been a constant source of strength and motivation. I would also like to thank my girlfriend, Hanlu Li, for her companionship and love during my PhD.

Lastly, a light-hearted thanks to Real Madrid and Manchester United for the joy and excitement their games bring into my life. Watching them play has been a wonderful break from my studies.

Chapter 2, in full, is a reprint of Libin Zhu, Chaoyue Liu, and Mikhail Belkin. "Transition to linearity of general neural networks with directed acyclic graph architecture." NeurIPS 2022. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in full, is a reprint of Libin Zhu, Chaoyue Liu, Adityanarayanan Radhakrishnan, and Mikhail Belkin. "Quadratic models for understanding catapult dynamics of neural networks. " ICLR 2024. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in full, is a reprint of Libin Zhu, Chaoyue Liu, Adityanarayanan Radhakrishnan, and Mikhail Belkin. "Catapults in sgd: spikes in the training loss and their impact on generalization through feature learning." ICML 2024. The dissertation author was the primary investigator and author of this paper.

VITA

2018 B.S. in Mathematics, Zhejiang University, Hangzhou, Zhejiang, China

2024 Ph.D. in Computer Science, University of California San Diego, La Jolla, California, USA

ABSTRACT OF THE DISSERTATION

Toward Understanding the Dynamics of Over-parameterized Neural Networks

by

Libin Zhu

Doctor of Philosophy in Computer Science

University of California San Diego, 2024

Professor Mikhail Belkin, Chair

The practical applications of neural networks are vast and varied, yet a comprehensive understanding of their underlying principles remains incomplete. This dissertation advances the theoretical understanding of neural networks, with a particular focus on over-parameterized models. It investigates their optimization and generalization dynamics and sheds light on various deep-learning phenomena observed in practice. This research deepens the understanding of the complex behaviors of these models and establishes theoretical insights that closely align with their empirical behaviours across diverse computational tasks.

In the first part of the thesis, we analyze the fundamental properties of over-parameterized neural networks and we demonstrate that these properties can lead to the success of their

optimization. We show that feedforward neural networks corresponding to arbitrary directed acyclic graphs undergo transition to linearity. The transition to linearity is characterized by the networks converging to their first-order Taylor expansion of parameters as their "width" approaches infinity. The width of these general networks is characterized by the minimum indegree of their neurons, except for the input and first layers. We further demonstrate that the property of transition to linearity plays an important role in the success of the optimization of over-parameterized neural networks.

In this second part of the thesis, we investigate the modern training regime of over-parameterized neural networks, particularly focusing on the large learning rate regime. While neural networks can be approximated by linear models as their width increases, certain properties of wide neural networks cannot be captured by linear models. We show that recently proposed Neural Quadratic Models can exhibit the "catapult phase" [65] that arises when training such models with large learning rates. We then empirically show that the behaviour of neural quadratic models parallels that of neural networks in generalization, especially in the catapult phase regime. Our analysis further demonstrates that quadratic models can be an effective tool for analysis of neural networks.

Moreover, we extend the analysis of catapult dynamics to stochastic gradient descent (SGD). We first present an explanation regarding the common occurrence of spikes in the training loss when neural networks are trained with SGD. We provide evidence that the spikes in the training loss of SGD are caused by catapults. Second, we posit an explanation for how catapults lead to better generalization by demonstrating that catapults increase feature learning by increasing alignment with the Average Gradient Outer Product (AGOP) of the true predictor. Furthermore, we demonstrate that a smaller batch size in SGD induces a larger number of catapults, thereby improving AGOP alignment and test performance.

Overall, by integrating theoretical insights with empirical validations, this dissertation provides a new understanding of the complex dynamics governing neural network training and generalization.

Chapter 1

Introduction

Recent advances in neural networks have significantly transformed various sectors, including image recognition, natural language processing, and autonomous systems. Notwithstanding their remarkable capabilities, these networks frequently function as "black boxes", posing substantial challenges for transparency and interpretability. This opacity is particularly concerning with the rise of Large Language Models (LLMs) [84], which signal the potential emergence of general artificial intelligence and ignite discussions regarding their safe development and application. This "black box" challenge demands immediate attention and research efforts to develop methods that demystify the inner workings of these networks.

Modern deep learning models often have substantial numbers of parameters. For example, modern LLMs utilize billions or even trillions of parameters, significantly enhancing their performance [7, 84, 24, 104]. These models are typically trained using gradient descent-based methods, such as Stochastic Gradient Descent (SGD) or Adam [57]. Recently, there has been a growing understanding of the dynamics of these gradient-based methods, especially in the context of over-parameterized models that can perfectly fit the training data. The understanding of over-parameterization and gradient-based methods can lead to improved transparency and interpretability of neural networks. Therefore, in this thesis, we concentrate on over-parameterized neural networks and tackle two major problems of neural networks: optimization and generalization, within the context of gradient-based training.

The thesis is structured into chapters, each addressing a different aspect of training dynamics and generalization capabilities of neural networks.

Linear training dynamics

Neural networks are often considered highly nonlinear models due to their complex, multi-layered non-linear structures. However, for wide neural networks, i.e., the number of neurons in hidden layers is sufficiently large, they are indeed approximately linear in a ball of a fixed radius of parameters. This property was first discovered in [46] in terms of the constancy of the Neural Tangent Kernel (NTK) along the optimization path with gradient flow and was termed as *transition to linearity* from [66].

In Chapter 2, we show that the property of transition to linearity holds for a much broader class of neural networks – *feedforward neural networks*. The architecture of a feedforward neural network can generically be described by a DAG [117, 110, 74], which includes standard network architectures e.g., FCNs, CNNs and ResNets. This generalization shows that the transition to linearity, or the constant Neural Tangent Kernel, does not depend on the specific designs of the networks, and is a more fundamental and universal property.

Transition to linearity helps understand the training dynamics of wide neural networks and plays an important role in developing the optimization theory for them, as has been shown for certain particular wide neural networks [22, 21, 15, 64, 126, 125]. Specifically, their training dynamics under certain settings can be accurately characterized by the corresponding linear models, which are the first-order Taylor expansion of the neural networks. Such training dynamics are referred to as linear dynamics or kernel regimes.

Catapult dynamics

While linear training dynamics apply to small constant learning rates, practical optimization of neural networks often prefers larger learning rates, which exhibit non-linear dynamics and result in better generalization. Therefore, understanding the non-linear dynamics of neural networks is essential to explain their superior performance compared to other machine learning

models.

For wide neural networks trained with a large learning rate, a notable non-linear phenomenon termed "catapult dynamics" was observed in [65]. Here, the loss initially increases and then decreases after reaching a large value, along with the decrease of the spectral norm of the NTK. More interestingly, the neural networks have better test performance when the catapult phase phenomenon occurs.

In Chapter 3, we utilize a quadratic model as a tool to shed light on the catapult dynamics of neural networks. The quadratic model is defined as the second-order Taylor series expansion of the network function. We analytically show that quadratic models exhibit catapult dynamics when trained with a large learning rate. Moreover, we empirically show that the behavior of neural quadratic models parallels that of neural networks in generalization, especially in the catapult phase regime.

Feature learning in catapult dynamics

In Chapter 4, we extend the catapult dynamics that are typically observed in gradient descent to SGD. Specifically, we show that the spikes in the training loss of SGD, that are commonly observed in practice, are caused by catapult dynamics.

Additionally, we demonstrate that catapult dynamics improve the generalization performance in SGD, similar to their effects in GD. We first show catapults enhance generalization through increasing feature learning. The feature learning is quantified by the alignment between the Average Gradient Outer Product (AGOP) of the trained network and of the true predictor. This mechanism is empirically verified for a single catapult in GD and is shown to hold for multiple catapults in GD, which can be induced by increasing the learning rates during training. Second, we show that smaller batch size with SGD can lead to better test performance due to a higher AGOP alignment, which is caused by an increase in the number of catapults.

Chapter 2

Trainsition to linearity of general feedforward neural networks

In this chapter, we investigate a key property of neural networks known as *transition to linearity*, a phenomenon where networks behave more predictably as they grow wider. Initially identified in well-known network architectures like fully-connected [46, 64] and convolutional networks [8], this characteristic suggests that such predictability might be intrinsic to neural networks, not just specific architectures. However, the underlying mathematical structure and broader implications of this transition have not been fully understood. In this chapter, we focus on the general feedforward neural networks corresponding to Directed Acyclic Graphs (DAGs), showing that the transition to linearity is a general property across various neural network architectures. We further investigate how the transition to linearity leads to the success of optimization, potentially shedding light on effective strategies for training and network design.

2.1 Introduction

A remarkable property of wide neural networks, first discovered in [46] in terms of the constancy of the Neural Tangent Kernel along the optimization path, is that they transition to linearity (using the terminology from [66]), i.e., are approximately linear in a ball of a fixed radius. There has been an extensive study of this phenomenon for different types of standard neural networks architectures including fully-connected neural networks (FCNs), convolutional

neural networks (CNNs), ResNets [64, 15, 8, 34]. Yet the scope of the transition to linearity and the underlying mathematical structure has not been made completely clear.

In this paper, we show that the property of transition to linearity holds for a much broader class of neural networks – *feedforward neural networks*. The architecture of a feedforward neural network can generically be described by a DAG [117, 110, 74]: the vertices and the edges correspond to the neurons and the trainable weight parameters of a neural network, respectively. This DAG structure includes standard network architectures e.g., FCNs, CNNs, ResNets, as well as DenseNets [42], whose property of transition to linearity has not been studied in literature. This generalization shows that the transition to linearity, or the constant Neural Tangent Kernel, does not depend on the specific designs of the networks, and is a more fundamental and universal property.

We define the width of a feedforward neural network as the minimum in-degree of all neurons except for the input and first layers, which is a natural generalization of the the minimum number of neurons in hidden layers which is how the width is defined for standard architectures.

For a feedforward neural network, we show it transitions to linearity if its width goes to infinity as long as the in-degrees of individual neurons are bounded by a polynomial of the network width. Specifically, we control the deviation of the network function from its linear approximation by the spectral norm of the Hessian of the network function, which, as we show vanishes in a ball of fixed radius, in the infinite width limit. Interestingly, we observe that not only the output neurons, but any pre-activated neuron in the hidden layers of a feedforward neural network can be regarded as a function with respect to its parameters, which will also transition to linearity as the width goes to infinity.

The key technical difficulty is that all existing analyses for transition to linearity or constant NTK do not apply to this general DAG setting. Specifically, those analyses assume in-degrees of neurons are either the same or proportional to each other up to a constant ratio [21, 64, 8, 125, 66, 6]. However, the general DAG setting allows different scales of neuron in-degrees, for example, the largest in-degree can be polynomially large in the smallest in-degree.

In such scenarios, the $(2,2,1)$-norm in [66] and the norm of parameter change in [21, 64] scales with the maximum of in-degrees which causes a trivial bound on the NTK change. Instead, we introduce a different set of tools based on the tail bound for the norm of matrix Gaussian series [106]. Specifically, we show that the Hessian of the network function takes the form of matrix Gaussian series, whose matrix variance relies on the Hessian of connected neurons. Therefore, we reconcile the in-degree difference by building a recursive relation between the Hessian of neurons, which exactly cancels out the in-degree with the scaling factor.

Transition to linearity helps understand the training dynamics of wide neural networks and plays an important role in developing the optimization theory for them, as has been shown for certain particular wide neural networks [22, 21, 15, 64, 126, 125]. While transition to linearity is not a necessary condition for successful optimization, it provides a powerful tool for analyzing optimization for many different architectures. Specifically, transition to linearity in a ball of sufficient radius combined with a lower bound on the norm of the gradient at its center is sufficient to demonstrate the PL* condition [67] (a version of the Polyak-Łojasiewicz condition [89, 70]) which ensures convergence of optimization. We discuss this connection and provide one such lower bound in Section 2.4.

Summary of contributions

We show the phenomenon of transition to linearity in general feedforward neural networks corresponding to a DAG with large in-degree. Specifically, under the assumption that the maximum in-degree of its neurons is bounded by a polynomial of the width m (the minimum in-degree), we prove that the spectral norm of the Hessian of a feedforward neural network is bounded by $\tilde{O}(1/\sqrt{m})$ in an $O(1)$ ball. Our results generalize the existing literature on the linearity of wide feedforward neural networks. We discuss connections to optimization. Under additional assumptions we show that the norm of the gradient of a feedforward neural network is bounded away from zero at initialization. Together with the Hessian bound this implies convergence of gradient descent for the loss function.

Notations

We use bold lowercase letters, e.g., $\mathbf{w}$, to denote vectors, capital letters, e.g., A, to denote matrices, and bold capital letters, e.g., $\mathbf{H}$, to denote higher order tensors or matrix tuples. For a matrix A, we use $A_{[i,:]}$ to denote its i-th row and $A_{[:,i]}$ to denote its j-th column.

We use $\nabla_{\mathbf{w}} f(\mathbf{w}_0)$ to denote the gradient of f with respect to $\mathbf{w}$ at $\mathbf{w}_0$, and $H_f(\mathbf{w})$ to denote Hessian matrix (second derivative) of f with respect to $\mathbf{w}$. For vectors, we use $\|\cdot\|$ to denote Euclidean norm. For matrices, we use $\|\cdot\|$ to denote spectral norm and $\|\cdot\|_F$ to denote Frobenius norm. We use $\|\cdot\|_\infty$ to denote function L_∞ norm. For a set $\mathscr{S}$, we use $|\mathscr{S}|$ to denote the cardinality of the set. For $n > 0$, $[n]$ denotes the set $\{1, 2, ..., n\}$.

We use big-O notation to hide constant factors, and use big-$\tilde{O}$ notation to additionally hide logarithmic factors. In this paper, the argument of $O/\tilde{O}(\cdot)$ is always with respect to the network width.

Given a vector $\mathbf{w}$ and a constant $R > 0$, we define a Euclidean ball $\mathsf{B}(\mathbf{w}, R)$ as:

$$\mathsf{B}(\mathbf{w}, R) := \{\mathbf{v} : \|\mathbf{v} - \mathbf{w}\| \leq R\}. \tag{2.1}$$

2.2 Neural networks with acyclic graph architecture

In this section, we provide a definition and notation for general feedforward neural networks with an arbitrary DAG structure. This definition includes standard feedforward neural network architectures, such as FCNs and DenseNet.

Defining feedforward neural networks

Graph Structure. Consider a directed acyclic graph (DAG) $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ and $\mathscr{E}$ denote the sets of vertices and edges, respectively. See the left panel of Figure 2.1, for an illustrative example. For a directed edge $e \in \mathscr{E}$, we may also use the notation $e = (v_1, v_2)$ to explicitly write out the start vertex v_1 and end vertex v_2.

For a vertex $v \in \mathscr{V}$, we denote its in-degree, $\mathsf{in}(v)$, by the number of incoming edges

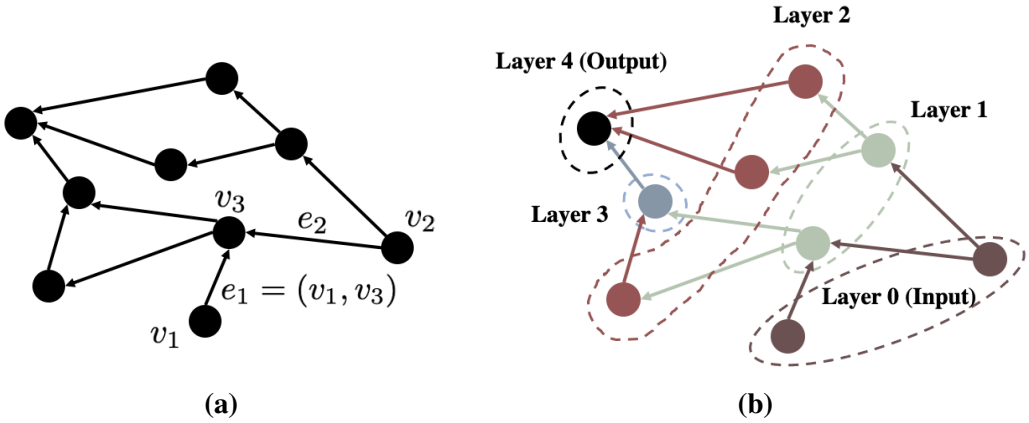


Figure 2.1. (a): An example of directed acyclic graph. v_1, v_2 and v_3 are three vertices and e_1, e_2 are two edges of the graph. v_3 has two incoming edges e_1 and e_2 which connects to v_1 and v_2 respectively. (b): Organizing the vertices into layers. The vertices with 0 in-degree are in 0-th layer (or input layer), and last layer are called output layer in which the vertices have 0 out-degree. Note that the layer index is determined by the longest path from the inputs $\mathscr{V}_{\text{input}}$, for example, the neuron in layer 3.

(edges that end with it):

$$\mathsf{in}(v) = |\mathscr{S}_{\text{in}}(v)|, \text{ with } \mathscr{S}_{\text{in}}(v) := \{u \in \mathscr{V} : (u,v) \in \mathscr{E}\}.$$

Similarly, for a vertex $v \in \mathscr{V}$, we denote its out-degree $\mathsf{out}(v)$ by the number of outgoing edges (edges that start from it):

$$\mathsf{out}(v) = |\mathscr{S}_{\text{out}}(v)|, \text{ with } \mathscr{S}_{\text{out}}(v) := \{u \in \mathscr{V} : (v,u) \in \mathscr{E}\}.$$

We call the set of vertices with zero in-degrees *input*: $\mathscr{V}_{\text{input}} = \{v \in \mathscr{V} : \mathsf{in}(v) = 0\}$, and the set of vertices with zero out-degrees *output* $\mathscr{V}_{\text{output}} = \{v \in \mathscr{V} : \mathsf{out}(v) = 0\}$.

Definition 2.2.1. For each vertex $v \in \mathscr{V} \backslash \mathscr{V}_{\text{input}}$, its distance $p(v)$, to the input $\mathscr{V}_{\text{input}}$, is defined to be the maximum length of all paths that start from a vertex within $\mathscr{V}_{\text{input}}$ and end with v.

It is easy to check that $p(v) = 0$ if $v \in \mathscr{V}_{\text{input}}$.

Feedforward neural network. Based on a given DAG architecture, we define the

feedforward neural network. Each individual vertex corresponds to a neuron additionally equipped with a scalar function (also called activation function). Each edge is associated with a real-valued weight, a trainable parameter. Each neuron is defined as a function of the weight parameters and the adjacent neurons connected by its incoming edges. The feedforward neural network is considered as the output neurons, corresponding to the output $\mathscr{V}_{\text{output}}$, of all weight parameters and input neurons which correspond to the input $\mathscr{V}_{\text{input}}$. Formally, we define the feedforward neural network as follows.

Definition 2.2.2 (Feedforward neural network)**.** Consider a DAG $\mathscr{G} = (\mathscr{V}, \mathscr{E})$. For each vertex $v \in \mathscr{V} \setminus \mathscr{V}_{\text{input}}$, we associate it with an activation function $\sigma_v(\cdot) : \mathbb{R} \to \mathbb{R}$ and each of its incoming edges $e = (u, v) \in \mathscr{E}$ with a weight variable $w_e = w_{(u,v)}$. Then we define the following functions:

$$f_v = \sigma_v(\tilde{f}_v), \quad \tilde{f}_v = \frac{1}{\sqrt{\mathsf{in}(v)}} \sum_{u \in \mathscr{S}_{\text{in}}(v)} w_{(u,v)} f_u. \tag{2.2}$$

When $v \in \mathscr{V}_{\text{input}}$, f_v is prefixed as the input data, and we denote $f_{\text{input}} := \{f_v : v \in \mathscr{V}_{\text{input}}\}$. For $v \notin \mathscr{V}_{\text{input}}$, we call f_v *neurons* and $\tilde{f}_v$ *pre-activations*. With necessary composition of functions, each f_v, and $\tilde{f}_v$, can be regarded as a function of all related weight variables and inputs f_{input}. The *feedforward neural network* is defined to be the function corresponding to the output $\mathscr{V}_{\text{output}}$:

$$f(\mathscr{W}; f_{\text{input}}) := f_{\text{output}} = \{f_v : v \in \mathscr{V}_{\text{output}}\}, \tag{2.3}$$

where $\mathscr{W} := \{w_e : e \in \mathscr{E}\}$ denotes the set of all the weight variables.

Remark 2.2.3. The validity of the definition is guaranteed by the fact that the DAG is acyclic. It makes sure that the dependence of each function f_v on other neurons can pass all the way down to the input f_{input}, through Eq. (2.2).

Remark 2.2.4. For $v \in \mathscr{V}_{\text{input}} \bigcup \mathscr{V}_{\text{output}}$, we use the identity function $\mathbb{I}(\cdot)$ as the activation functions.

Weight initialization and inputs. Each weight parameter $w_e \in \mathscr{W}$ is initialized i.i.d. following the standard normal distribution i.e., $\mathscr{N}(0,1)$. The inputs are considered given, usually

determined by datasets. Under this initialization, we introduce the scaling factor $1/\sqrt{\mathsf{in}(v)}$ in Eq. (2.5) to control the value of neurons to be of order $O(1)$. Note that this initialization is an extension of the NTK initialization [46], which was defined for FCNs therein.

Generality of DAG architecture. Including FCNs and DenseNets [42] as special examples, the class of feedforward neural networks allows much more choices of architectures, for example, neural networks with randomly dropped edges. Please see detailed discussions about these specific examples in Appendix A.1. We note that our definition of feedforward neural networks does not directly include networks with nont-trainable skip connections, e.g., ResNets, and networks with shared weights, e.g., CNNs. However, with a slight modification of the analysis, the property of transition to linearity still holds. See the detailed discussion in Appendix A.4 and A.5.

Organizing feedforward networks into layers

The architecture of the feedforward neural network is determined by the DAG $\mathscr{G}$. The complex structures of DAGs often lead to complicated neural networks, which are hard to analyze.

For the ease of analysis, we organize the neurons of the feedforward neural network into *layers*, which are sets of neurons.

Definition 2.2.5 (Layers)**.** Consider a feedforward neural network f and its corresponding graph structure $\mathscr{G}$. A layer of the network is defined to be the set of neurons which have the same distance p to the inputs. Specifically, the ℓ-th layer, denoted by $f^{(\ell)}$, is

$$f^{(\ell)} = \{f_v : p(v) = \ell, v \in \mathscr{V}, \ell \in \mathbb{N}\}. \tag{2.4}$$

It is easy to see that the layers are mutually exclusive, and the layer index ℓ is labeled from 0 to ℓ, where $L+1$ is the total number of layers in the network. As $p(v) = 0$ if and only if $v \in \mathscr{V}_{\text{input}}$, the 0-th layer $f^{(0)}$ is exactly the input layer f_{input}. The right panel of Figure 2.1

provides an illustrative example of the layer structures.

In general, the output neurons f_{output} (defined in Eq. (2.3)) do not have to be in the same layer. For the convenience of presentation and analysis, we assume that all the output neurons are in the last layer, i.e., layer ℓ, which is the case for most of commonly used neural networks, e.g., FCNs and CNNs. Indeed, our analysis applies to every output neuron (see Theorem 2.3.8), even if they are not in the same layer.

With the notion of network layers, we rewrite the neuron functions Eq. (2.2), as well as related notations, to reflect the layer information.

For ℓ-layer, $\ell = 0, 1, \cdots, L$, we denote the total number of neurons as d_ℓ, and rewrite the layer function $f^{(\ell)}$ into a form of vector-valued function

$$f^{(\ell)} = \left(f_1^{(\ell)}, f_2^{(\ell)}, ..., f_{d_\ell}^{(\ell)}\right)^T,$$

where we use $f_i^{(\ell)}$ with index $i = 1, 2, \cdots, d_\ell$ to denote each individual neuron. Correspondingly, we denote its vertex as $v_i^{(\ell)}$, and $\mathscr{S}_i^{(\ell)} := \mathscr{S}_{\text{in}}(v_i^{(\ell)})$. Hence, the in-degree $\text{in}(v_i^{(\ell)})$, denoted as $m_i^{(\ell)}$ here, is equivalent to the cardinality of the set $\mathscr{S}_i^{(\ell)}$.

Remark 2.2.6. Note that $m_i^{(\ell)}$, with the superscript ℓ, denotes an in-degree, i.e., the number of neurons that serve as direct inputs to the current neuron in ℓ-th layer. In the context of FCNs, $m_i^{(\ell)}$ is equivalent to the size of its previous layer, i.e., $(\ell-1)$-th layer, and is often denoted as $m^{(\ell-1)}$ in literature.

To write the summation in Eq. (2.2) as a matrix multiplication, we further introduce the following two vectors: (a), $f_{\mathscr{S}_i^{(\ell)}}$ represents the vector that consists of neuron components f_v with $v \in \mathscr{S}_i^{(\ell)}$; (b), $\mathbf{w}_i^{(\ell)}$ represents the vector that consists of weight parameters $w_{(u,v_i^{(\ell)})}$ with $u \in \mathscr{S}_i^{(\ell)}$. Note that both vectors $f_{\mathscr{S}_i^{(\ell)}}$ and $\mathbf{w}_i^{(\ell)}$ have the same dimension $m_i^{(\ell)}$.

With the above notation, the neuron functions Eq. (2.2) can be equivalently rewritten as:

$$f_i^{(\ell)} = \sigma_i^{(\ell)}(\tilde{f}_i^{(\ell)}),\ \ \tilde{f}_i^{(\ell)} = \frac{1}{\sqrt{m_i^{(\ell)}}} \left(\mathbf{w}_i^{(\ell)}\right)^T f_{\mathscr{S}_i^{(\ell)}}. \tag{2.5}$$

For any $\ell \in [L]$, we denote the weight parameters corresponding to all incoming edges toward neurons at layer ℓ by

$$\mathbf{w}^{(\ell)} := \left((\mathbf{w}_1^{(\ell)})^T, ..., (\mathbf{w}_{d_\ell}^{(\ell)})^T\right)^T \quad \ell \in [L]. \tag{2.6}$$

Through the way we define the feedforward neural network, the output of the neural network is a function of all the weight parameters and the input data, hence we denote it by

$$f(\mathbf{w}; \boldsymbol{x}) := f^{(L)} = \left(f_1^{(L)}, ..., f_{d_L}^{(L)}\right)^T, \tag{2.7}$$

where $\mathbf{w}$ is the collection of all the weight parameters, i.e., $\mathbf{w} := \left((\mathbf{w}^{(1)})^T, ..., (\mathbf{w}^{(L)})^T\right)^T \in \mathbb{R}^{\Sigma_\ell \Sigma_i m_i^{(\ell)}}$.

With all the notations, for a feedforward neural network, we formally define the width of it:

Definition 2.2.7 (Network width)**.** The width m of a feedforward neural network is the minimum in-degree of all the neurons except those in the input and first layers:

$$m := \inf_{\ell \in \{2,...,L\}, i \in [d_\ell]} m_i^{(\ell)}. \tag{2.8}$$

Remark 2.2.8. Note that, the network width m is determined by the in-degrees of neurons except for the input and first layers, and not necessarily relates the number of neurons in hidden layers. But for certain architectures e.g., FCNs, these two coincide that the minimum in-degree after the

first layer is the same as the minimum hidden layer size.

We say a feedforward neural network is *wide* if its width m is large enough. In this paper, we consider wide feedforward neural networks with a fixed number of layers.

2.3 Transition to linearity of feedforward neural networks

In this section, we show that the feedforward neural networks exhibit the phenomenon of transition to linearity, which was previously observed in specific types of neural networks.

Specifically, we prove that a feedforward neural network $f(\mathbf{w};\boldsymbol{x})$, when considered as a function of its weight parameters $\mathbf{w}$, is arbitrarily close to a *linear* function in the ball $\mathsf{B}(\mathbf{w}_0, R)$ given constant $R > 0$, where $\mathbf{w}_0$ is randomly initialized, as long as the width of the network is sufficiently large.

First, we make the following assumptions on the input $\boldsymbol{x}$ and the activation functions:

Assumption 2.3.1. The input is uniformly upper bounded, i.e., $\|\boldsymbol{x}\|_\infty \leq C_{\boldsymbol{x}}$ for some constant $C_{\boldsymbol{x}} > 0$.

Assumption 2.3.2. All the activation functions $\sigma(\cdot)$ are twice differentiable, and there exist constants $\gamma_0, \gamma_1, \gamma_2 > 0$ such that, for all activation functions, $|\sigma(0)| \leq \gamma_0$ and the following Lipschitz continuity and smoothness conditions are satisfied

$$\begin{aligned}
&\left|\sigma'(z_1) - \sigma'(z_2)\right| \leq \gamma_1 |z_1 - z_2|, \\
&\left|\sigma''(z_1) - \sigma''(z_2)\right| \leq \gamma_2 |z_1 - z_2|, \ \ \forall z_1, z_2 \in \mathbb{R}.
\end{aligned}$$

We note that the above two assumptions are very common in literature. Although ReLU does not satisfy Assumption 2.3.2 due to non-differentiability at point 0, we believe our main claims still hold as ReLU can be approximated arbitrarily closely by some differentiable function which satisfies our assumption.

Remark 2.3.3. By assuming all the activation functions are twice differentiable, it is not hard to see that the feedforward neural network i.e., Eq. (2.7) is also twice differentiable.

Taylor expansion. To study the linearity of a general feedforward neural network, we consider its Taylor expansion with second order Lagrange remainder term. Given a point $\mathbf{w}_0$, we can write the network function $f(\mathbf{w})$ (omitting the input argument for simplicity) as

$$f(\mathbf{w}) = \underbrace{f(\mathbf{w}_0) + (\mathbf{w}-\mathbf{w}_0)^T \nabla_{\mathbf{w}} f(\mathbf{w}_0)}_{f_{\text{lin}}(\mathbf{w})} + \underbrace{\frac{1}{2}(\mathbf{w}-\mathbf{w}_0)^T H_f(\xi)(\mathbf{w}-\mathbf{w}_0)}_{\mathscr{R}(\mathbf{w})}, \tag{2.9}$$

where ξ is a point on the line segment between $\mathbf{w}_0$ and $\mathbf{w}$. Above, $f_{\text{lin}}(\mathbf{w})$ is a linear function and $\mathscr{R}(\mathbf{w})$ is the Lagrange remainder term.

In the rest of the section, we will show that in a ball $\mathsf{B}(\mathbf{w}_0, R)$ of any constant radius $R > 0$,

$$|\mathscr{R}(\mathbf{w})| = \tilde{O}\left(1/\sqrt{m}\right) \tag{2.10}$$

where m is the network width (see Definition 2.2.7). Hence, $f(\mathbf{w})$ can be arbitrarily close to its linear approximation $f_{\text{lin}}(\mathbf{w})$ with sufficiently large m. Note that in Eq. (2.9), we consider a single output of the network function. The same analysis can be applied to multiple outputs (see Corollary A.3.1).

Remark 2.3.4. For a general function, the remainder term $\mathscr{R}(\mathbf{w})$ is not expected to vanish at a finite distance from $\mathbf{w}_0$. Hence, the transition to linearity in the ball $\mathsf{B}(\mathbf{w}_0, R)$ is a non-trivial property. On the other hand, the radius R can be set to be large enough to contain the whole optimization path of GD/SGD for various types of wide neural networks (see [67, 126], also indicated in [22, 21, 125, 64]). In Section 2.4, we will see that such a ball is also large enough to cover the whole optimization path of GD/SGD for the general feedforward neural networks. Hence, to study the optimization dynamics of wide feedforward neural networks, this ball is

large enough.

To prove Eq. (2.10), we make an assumption on the width m:

Assumption 2.3.5. The maximum in-degree of any neuron is at most polynomial in the network width m:

$$\sup_{\ell\in\{2,\dots,L\},i\in[d_\ell]} m_i^{(\ell)} = O(m^c),$$

where $c > 0$ is a constant.

This assumption puts a constraint on the neurons with large in-degrees such that the in-degrees cannot be super-polynomially large compared to m. A natural question is whether this constraint is necessary, for example, do our main results still hold in cases some in-degrees are exponentially large in m? While we believe the answer is positive, we need this assumption to apply the proof techniques. Specifically, we apply the tail bound for the norm of matrix Gaussian series [106], where there is a dimension factor equivalent to the number of weight parameters. Thus an exponentially large dimension factor would result in useless bounds. It is still an open question whether the dimension factor in the bound can be removed or moderated (see the discussion after Theorem 4.1.1 in [106]).

With these assumptions, we are ready to present our main result:

Theorem 2.3.6 (Scaling of the Hessian norm)**.** *Suppose Assumption 2.3.1, 2.3.2 and 2.3.5 hold. Given a fixed $R > 0$, with probability at least $1-\exp\big(-\Omega(\log^2 m)\big)$ over the random initialization $\mathbf{w}_0$, for all $\mathbf{w} \in \mathsf{B}(\mathbf{w}_0, R)$, each output neuron f_k of a feedforward neural network satisfies*

$$\big\|H_{f_k}(\mathbf{w})\big\| = O\left((\log m + R)^{L^2}/\sqrt{m}\right) = \tilde{O}\left(R^{L^2}/\sqrt{m}\right), \quad k \in [d_\ell]. \tag{2.11}$$

This theorem states that the Hessian matrix, as the second derivative with respect to weight parameters $\mathbf{w}$, of any output neuron can be arbitrarily small, if the network width is sufficient large.

Note that Eq. (2.11) holds for all $\mathbf{w} \in \mathsf{B}(\mathbf{w}_0, R)$ with high probability over the random initialization $\mathbf{w}_0$. The basic idea is that, the spectral norm of Hessian can be bounded at the center of the ball, i.e., $\mathbf{w}_0$, though probability bounds due to the randomness of $\mathbf{w}_0$. For all other points $\mathbf{w} \in \mathsf{B}(\mathbf{w}_0, R)$, the distance $\|\mathbf{w} - \mathbf{w}_0\|$, being no greater than R, controls $\|H(\mathbf{w}) - H(\mathbf{w}_0)\|$ such that it is no larger than the order of $\|H(\mathbf{w}_0)\|$, hence $\|H(\mathbf{w})\|$ keeps the same order. See the proof in Appendix A.2.

Using the Taylor expansion Eq. (2.9), we can bound the Lagrange remainder and have transition to linearity of the network:

Corollary 2.3.7 (Transition to linearity)**.** *Suppose Assumption 2.3.1, 2.3.2 and 2.3.5 hold. Given a fixed $R > 0$, with probability at least $1 - \exp\big(-\Omega(\log^2 m)\big)$ over the random initialization $\mathbf{w}_0$, for all $\mathbf{w} \in \mathsf{B}(\mathbf{w}_0, R)$, each f_k will be closely approximated by a linear model:*

$$|f_k(\mathbf{w}) - (f_k)_{\text{lin}}(\mathbf{w})| \leq \frac{1}{2} \sup_{\mathbf{w} \in \mathsf{B}(\mathbf{w}_0, R)} \|H_{f_k}(\mathbf{w})\| R^2 = \tilde{O}\left(R^{L^2+2}/\sqrt{m}\right).$$

For feedforward neural networks with multiple output neurons, the property of transition to linearity holds with high probability, if the number of output neurons is bounded, i.e., $d_\ell = O(1)$. See the result in Appendix A.3.

Furthermore, as defined in Definition 2.2.2, each pre-activation, as a function of all related weight parameters and inputs, can be viewed as a feedforward neural network. Therefore, we can apply the same techniques used for Theorem 2.3.6 to show that each pre-activation can transition to linearity:

Theorem 2.3.8. *Suppose Assumption 2.3.1, 2.3.2 and 2.3.5 hold. Given a fixed radius $R > 0$, with probability at least $1 - \exp\big(-\Omega(\log^2 m)\big)$ over the random initialization of $\mathbf{w}_0$, for all $\mathbf{w} \in \mathsf{B}(\mathbf{w}_0, R)$, any pre-activation in a feedforward neural network i.e., $\tilde{f}_k^{(\ell)}(\mathbf{w})$ satisfies*

$$\left\| H_{\tilde{f}_k^{(\ell)}}(\mathbf{w}) \right\| = O\left((\log m + R)^{\ell^2}/\sqrt{m}\right) = \tilde{O}\left(R^{\ell^2}/\sqrt{m}\right), \quad \ell \in [L],\ k \in [d_\ell]. \tag{2.12}$$

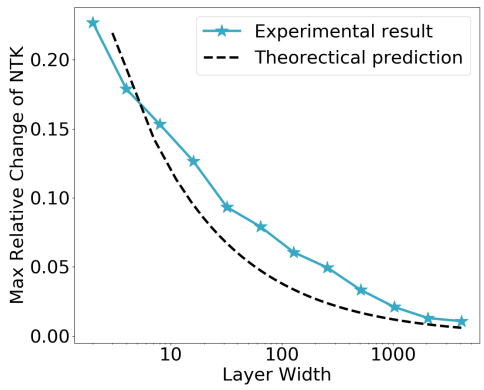


Figure 2.2. Transition to linearity of DAG network. The experimental result approximates well the theoretical prediction of relative change of tangent kernel from initialization to convergence, as a function of the network width. Each point on the solid curve is the average of independent 5 runs.

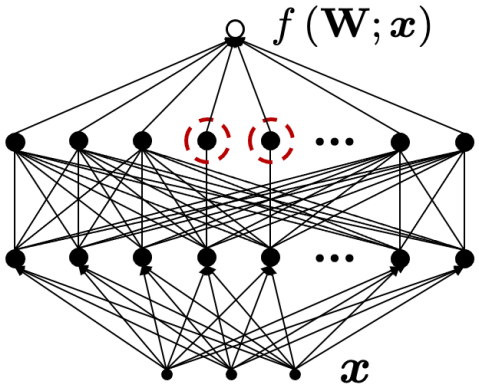


Figure 2.3. An example of DAG network with bottleneck neurons. The DAG network $f(\mathbf{W};\boldsymbol{x})$ has two bottleneck neurons (in red dashed circles) with in-degree 1, while the rest of neurons except for the input and the first layer have large in-degree. In this case, $f(\mathbf{W};\boldsymbol{x})$ will still transition to linearity with respect to $\mathbf{W}$ as the number of neurons goes to infinity.

Remark 2.3.9. Note that pre-activations in the input layer i.e., the input data and in the first layer are constant and linear functions respectively, hence the spectral norm of their Hessian is zero.

Experimental verification. To verify our theoretical result on the scaling of the Hessian norm, i.e., Theorem 2.3.6, we train a DAG network built from a 3-layer DenseNet with each weight removed i.i.d. with probability $1/2$, on 10 data points of CIFAR-2 (2-class subset of CIFAR-10 [59]) using GD. We compute the maximum relative change of the tangent kernel (definition is deferred to Eq. (2.17)) during training, i.e., $\max_t \|K_t - K_0\|/\|K_0\|$ to simulate the scaling of the spectral norm of the Hessian. We observe the convergence of loss for all widths $\{2, 2^2, ..., 2^{12}\}$, and the scaling of the Hessian follows close to the theoretical prediction of $\Theta(1/\sqrt{m})$. See Figure 2.2.

Non-linear activation on output neurons breaks transition to linearity. In the above discussions, the transition to linearity of networks are under the assumption of identity activation

function on every output neuron. In fact, the activation function on output neurons is critical to the linearity of neural networks. Simply, composing a non-linear function with a linear function will break the linearity. Consistently, as shown in [67] for FCNs, with non-linear activation function on the output, transition to linearity does not hold any more.

"Bottleneck neurons" do not necessarily break transition to linearity. We have seen that if *all* neurons have sufficiently large in-degree, the network will transition to linearity. Does transition to linearity still hold, if neurons with small in-degree exist? We call neurons with small in-degree *bottleneck neurons*, as their in-degrees are smaller than rest of neurons hence forming "bottlenecks". As we show in Appendix E, existence of these bottleneck neurons does not break the transition to linearity, as long as the number of such neurons is significantly smaller than that of non-bottleneck neurons. Figure 2.3 shows an example with two bottleneck neurons, whose in-degree is 1. This network still transitions to linearity as the number of neurons goes to infinity.

Proof sketch of Theorem 2.3.6

By Lemma A.11.1, the spectral norm of H_{f_k} can be bounded by the summation of the spectral norm of all the Hessian blocks, i.e., $\|H_{f_k}\| \leq \sum_{\ell_1,\ell_2} \|H_{f_k}^{(\ell_1,\ell_2)}\|$, where $H_{f_k}^{(\ell_1,\ell_2)} := \frac{\partial^2 f_k}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}}$. Therefore, it suffices to bound the spectral norm of each block. Without lose of generality, we consider the block with $1 \leq \ell_1 \leq \ell_2 \leq L$.

By the chain rule of derivatives, we can write the Hessian block into:

$$\frac{\partial^2 f_k}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} = \sum_{\ell'=\ell_2}^{L} \sum_{i=1}^{d_{\ell'}} \frac{\partial^2 f_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \frac{\partial f_k}{\partial f_i^{(\ell')}} := \sum_{\ell'=\ell_2}^{L} G_k^{L,\ell'}. \tag{2.13}$$

For each $G_k^{L,\ell'}$, since $f_i^{(\ell')} = \sigma\left(\tilde{f}_i^{(\ell')}\right)$, again by the chain rule of derivatives, we have

$$
\begin{aligned}
G_k^{L,\ell'} &= \sum_{i=1}^{d_{\ell'}} \frac{\partial^2 \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \frac{\partial f_k}{\partial \tilde{f}_i^{(\ell')}} + \frac{1}{\sqrt{m_k^{(L)}}} \sum_{i: f_i^{(\ell')} \in \mathscr{F}_{\mathscr{S}_k^{(L)}}} \left(\mathbf{w}_k^{(L)}\right)_{\mathsf{id}_{\ell',i}^{L,k}} \sigma''\left(\tilde{f}_i^{(\ell')}\right) \frac{\partial \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)}} \left(\frac{\partial \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_2)}}\right)^T \\
&= \frac{1}{\sqrt{m_k^{(L)}}} \sum_{r=\ell'}^{L-1} \sum_{i: f_i^{(r)} \in \mathscr{F}_{\mathscr{S}_k^{(L)}}} \left(\mathbf{w}_k^{(L)}\right)_{\mathsf{id}_{r,i}^{L,k}} \sigma'\left(\tilde{f}_s^{(r)}\right) G_i^{r,\ell'} \\
&\quad + \frac{1}{\sqrt{m_k^{(L)}}} \sum_{i: f_i^{(\ell')} \in \mathscr{F}_{\mathscr{S}_k^{(L)}}} \left(\mathbf{w}_k^{(L)}\right)_{\mathsf{id}_{\ell',i}^{L,k}} \sigma''\left(\tilde{f}_i^{(\ell')}\right) \frac{\partial \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)}} \left(\frac{\partial \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_2)}}\right)^T, \qquad (2.14)
\end{aligned}
$$

where $\mathscr{F}_{\mathscr{S}_k^{(L)}} := \{f : f \in f_{\mathscr{S}_k^{(L)}}\}$ and $\mathsf{id}_{\ell',i}^{L,k} := \{p : \left(f_{\mathscr{S}_k^{(L)}}\right)_p = f_i^{(\ell')}\}$.

The first quantity on the RHS of the above equation, $\sum \left(\mathbf{w}_k^{(L)}\right)_{\mathsf{id}_{\ell',i}^{L,k}} \sigma'\left(\tilde{f}_i^{(r)}\right) G_i^{r,\ell'}$, is a matrix Gaussian series with respect to random variables $\mathbf{w}_k^{(L)}$, conditioned on fixed $\sigma'\left(\tilde{f}_i^{(r)}\right) G_i^{r,\ell'}$ for all i such that $f_i^{(r)} \in \mathscr{F}_{\mathscr{S}_k^{(\ell)}}$. We apply the tail bound for matrix Gaussian series, Theorem 4.1.1 from [106], to bound this quantity. To that end, we need to bound its matrix variance, which suffices to bound the spectral norm of $\sum_i G_i^{r,\ell'}$ since $\sigma'(\cdot)$ is assumed to be uniformly bounded by Assumption 2.3.2. There is a recursive relation that the norm bound of $G_k^{L,\ell'}$ depends on the norm bound of $G_i^{r,\ell'}$ which appears in the matrix variance. Therefore, we can recursively apply the argument to bound each G.

Similarly, the second quantity on the RHS of the above equation is also a matrix Gaussian series with respect to $\mathbf{w}_k^{(L)}$, conditioned on fixed $\sigma''\left(\tilde{f}_i^{(\ell')}\right) \frac{\partial \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)}} \left(\frac{\partial \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_2)}}\right)^T$ for all i such that $f_i^{(\ell')} \in \mathscr{F}_{\mathscr{S}_k^{(L)}}$. As $\sigma''(\cdot)$ is assumed to be uniformly bounded by Assumption 2.3.2, we use Lemma A.2.1 to bound its matrix variance, hence the matrix Gaussian series can be bounded.

Note that such tail bound does not scale with the largest in-degree of the networks, since the in-degree of f_k, i.e., $m_k^{(L)}$, will be cancelled out with the scaling factor $\frac{1}{\sqrt{m_k^{(L)}}}$ in the bound of matrix variance. See the complete proof in Appendix A.2.

2.4 Relation to optimization

While transition to linearity is a significant and surprising property of wide networks in its own right, it also plays an important role in building the optimization theory of wide feedforward neural networks. Specifically, transition to linearity provides a path toward showing that the corresponding loss function satisfies the PL* condition in a ball of a certain radius, which is sufficient for exponential convergence of optimization to a global minimum by gradient descent or SGD [67].

Consider a supervised learning task. Given training inputs and labels $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, we use GD/SGD to minimize the square loss:

$$\mathscr{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(f(\mathbf{w};\boldsymbol{x}_i) - y_i)^2, \tag{2.15}$$

where $f(\mathbf{w};\cdot)$ is a feedforward neural network.

The loss $\mathscr{L}(\mathbf{w})$ is said to satisfy μ-PL* condition, a variant of the well-known Polyak-Łojasiewicz condition [89, 70], at point $\mathbf{w}$, if

$$\|\nabla_{\mathbf{w}}\mathscr{L}(\mathbf{w})\|^2 \geq 2\mu\mathscr{L}(\mathbf{w}), \quad \mu > 0.$$

Satisfaction of this μ-PL* condition in a ball $\mathsf{B}(\mathbf{w}_0, R)$ with $R = O(1/\mu)$ around the starting point $\mathbf{w}_0$ of GD/SGD guarantees a fast converge of the algorithm to a global minimum in this ball [67].

In the following, we use the transition to linearity of wide feedforward networks to establish the PL* condition for $\mathscr{L}(\mathbf{w})$. Taking derivative on Eq. (2.15), we get

$$\|\nabla_{\mathbf{w}}\mathscr{L}(\mathbf{w})\|^2 \geq 2\lambda_{\min}(K(\mathbf{w}))\mathscr{L}(\mathbf{w}), \tag{2.16}$$

where matrix $K(\mathbf{w})$, with elements

$$K_{i,j}(\mathbf{w}) = \nabla_{\mathbf{w}} f(\mathbf{w};\boldsymbol{x}_i)^T \nabla_{\mathbf{w}} f(\mathbf{w};\boldsymbol{x}_j) \text{ for } i,j \in [n], \tag{2.17}$$

is called Neural Tangent Kernel (NTK) [46], and $\lambda_{\min}(\cdot)$ denotes the smallest eigenvalue of a matrix. Note that, by definition, the NTK matrix is always positive semi-definite, i.e., $\lambda_{\min}(K(\mathbf{w})) \geq 0$.

Directly by the definition of PL* condition, at a given point $\mathbf{w}$, if $\lambda_{\min}(K(\mathbf{w}))$ is strictly positive, then the loss function $\mathscr{L}(\mathbf{w})$ satisfies PL* condition.

To establish convergence of GD/SGD, it is sufficient to verify that PL* condition is satisfied in a ball $\mathsf{B}(\mathbf{w}_0, R)$ with $R = O(1/\mu)$. Assuming that $\lambda_{\min}(K(\mathbf{w}_0))$ is bounded away from zero, transition to linearity extends the satisfaction of the PL* condition from one point $\mathbf{w}_0$ to all points in $\mathsf{B}(\mathbf{w}_0, R)$.

PL* condition at $\mathbf{w}_0$

For certain neural networks, e.g., FCNs, CNNs and ResNets, strict positiveness of $\lambda_{\min}(K(\mathbf{w}_0))$ can be shown, see for example, [22, 21]. We expect same holds more generally, in the case of general feedforward neural networks. Here we show that $\lambda_{\min}(K(\mathbf{w}_0))$ can be bounded from 0 for one data point under certain additional assumptions. Since there is only one data point, $\lambda_{\min}(K(\mathbf{w}_0)) = K(\mathbf{w}_0) = \|\nabla_{\mathbf{w}} f(\mathbf{w}_0)\|^2$. We also assume the following on activation functions and the input.

Assumption 2.4.1. The input $\mathbf{x}$ satisfies $\mathbf{x} \sim \mathscr{N}(0, I_{d_0})$.

Assumption 2.4.2. The activation function is homogeneous, i.e. $\sigma_i^{(\ell)}(az) = a^r \sigma_i^{(\ell)}(z), r > 0$ for any constant a. And $\inf_{\ell \in [L-1], i \in [d_\ell]} \mathbb{E}_{z \sim \mathscr{N}(0,1)} \left[\sigma_i^{(\ell)}(z)^2 \right] = C_\sigma > 0$.

Remark 2.4.3. Here for simplicity we assume the activation functions are homogeneous with the same r. It is not hard to extend the result to the case that each activation function has different r.

Proposition 2.4.4. *With Assumption 2.4.1 and 2.4.2, we have for any $k \in [d_\ell]$,*

$$\mathbb{E}_{\mathbf{x},\mathbf{w}_0}[\|\nabla_{\mathbf{w}} f_k(\mathbf{w}_0)\|] \geq \sqrt{\min\left(1, \min_{1\leq j\leq L} C_\sigma^{\sum_{l'=0}^{j-1} r^{\ell'}}\right)} = \Omega(1). \tag{2.18}$$

The proof can be found in Appendix A.7.

The above proposition establishes a positive lower bound on $\|\nabla_{\mathbf{w}} f(\mathbf{w}_0)\|$, hence also on $\lambda_{\min}(K(\mathbf{w}_0))$. Using Eq. (2.16), we get that the loss function $\mathscr{L}(\mathbf{w})$ satisfies PL* at $\mathbf{w}_0$.

Extending PL* condition to $\mathsf{B}(\mathbf{w}_0, R)$

Now we use transition to linearity to extend the satisfaction of PL* condition to the ball $\mathsf{B}(\mathbf{w}_0, R)$. In Theorem 2.3.6, we see that, a feedforward neural network $f(\mathbf{w})$ transitions to linearity, i.e., $\|H_f(\mathbf{w})\| = \tilde{O}(1/\sqrt{m})$ in this ball. An immediate consequence is that, for any $\mathbf{w} \in \mathsf{B}(\mathbf{w}_0, R)$,

$$|\lambda_{\min}(K(\mathbf{w})) - \lambda_{\min}(K(\mathbf{w}_0))| \leq O\left(\sup_{\mathbf{w}\in\mathsf{B}(\mathbf{w}_0,R)} \|H_f(\mathbf{w})\|\right).$$

Since $\lambda_{\min}(K(\mathbf{w}_0))$ is bound from 0 and $\|H_f(\mathbf{w})\|$ can be arbitrarily small as long as m is large enough, we have $\lambda_{\min}(K(\mathbf{w}))$ is lower bounded from 0 in the whole ball. Specifically, there is a $\mu > 0$ such that

$$\inf_{\mathbf{w}\in\mathsf{B}(\mathbf{w}_0,R)} \lambda_{\min}(K(\mathbf{w})) \geq \mu.$$

Moreover, the radius R can be set to be $O(1/\mu)$, while keeping the above inequality hold. Then, applying the theory in [67], existence of global minima of $\mathscr{L}(\mathbf{w})$ and convergence of GD/SGD can be established.

For the case of multiple data points, extra techniques are needed to lower bound the minimum eigenvalue of the tangent kernel. Since we focus more on the transition to linearity of feedforward neural networks in this paper, we leave it as a future work.

Non-linear activation function on outputs and transition to linearity. In this paper, we mainly discussed feedforward neural networks with linear activation function on output neurons. In most of the literature also considers this setting [46, 76, 81, 22, 21, 126, 125]. In fact, as pointed out in [67] for FCNs, while this linearity of activation function on the outputs is necessary for transition to linearity, it is not required for successful optimization. Specifically, simply adding a nonlinear activation function on the output layer causes the Hessian norm to be $O(1)$, independently of the network width. Thus transition to linearity does not occur. However, the corresponding square loss can still satisfy the PL* condition and the existence of global minimum and efficient optimization can still be established.

2.5 Discussion and future directions

In this work, we showed that transition to linearity arises in general feedforward neural networks with arbitrary DAG architectures, extending previous results for standard architectures [46, 64, 66]. We identified the minimum in-degree of all neurons except for the input and first layers as the key quantity to control the transition to linearity of general feedforward neural networks.

We showed that the property of transition to linearity is flexible to the choice of the neuron function Eq. (2.2). For example, skip connections Eq. (A.12) and shared weights Eq. (A.14) do not break the property. Therefore, we believe our framework can be extended to more complicated neuron functions, e.g., attention layers [41]. For non-feedforward networks, such as RNN, recent work [5] showed that they also have a constant NTK. For this reason, we expect transition to linearity also to occur for non-feedforward networks.

Another direction of future work is better understanding of optimization for DAG networks, which requires a more delicate analysis of the NTK at initialization. Specifically, with multiple training examples, a lower bound on the minimum eigenvalue of the NTK of the DAG networks is sufficient for the PL* condition to hold, thus guaranteeing the convergence of

GD/SGD.

2.6 Acknowledgements

Chapter 2, in full, is a reprint of Libin Zhu, Chaoyue Liu, and Mikhail Belkin. "Transition to linearity of general neural networks with directed acyclic graph architecture." NeurIPS 2022. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Catapult dynamics of neural networks

The recent observation of the transition to linearity in neural networks as their width approaches infinity marks a significant milestone in the theoretical understanding of these complex models. However, the optimization theory based on the transition to linearity requires a small, constant learning rate to guarantee the success of the optimization. In contrast, modern deep learning often uses large learning rates to enhance test performance. A notable finding concerning the dynamics of neural networks trained with large learning rates is the catapult dynamics [65], which has been shown to consistently improve the test performance of neural networks. Understanding the catapult phase phenomenon seems to be important to understanding the success of neural networks. Given that the optimization theory based on the transition to linearity fails to explain this phenomenon, there is a clear need for an alternative mathematical model.

In this chapter, we utilize quadratic models to bridge the gap between theoretical predictions and empirical observations in network training. We demonstrate that quadratic models exhibit similar behavior in both optimization and generalization, especially when the catapult phase occurs. Therefore, quadratic models can be an effective tool to understand the dynamics of neural networks.

3.1 Introduction

A recent remarkable finding on neural networks, originating from [46] and termed as the "transition to linearity" [66], is that, as network width goes to infinity, such models become linear functions in the parameter space. Thus, a linear (in parameters) model can be built to accurately approximate wide neural networks under certain conditions. While this finding has helped improve our understanding of trained neural networks [21, 83, 126, 76, 51, 15], not all properties of finite width neural networks can be understood in terms of linear models, as is shown in several recent works [116, 85, 71, 26]. In this work, we show that properties of finitely wide neural networks in optimization and generalization that cannot be captured by linear models are, in fact, manifested in quadratic models.

The training dynamics of linear models with respect to the choice of the learning rates[1] are well-understood [88]. Indeed, such models exhibit *linear* training dynamics, i.e., there exists a critical learning rate, η_{crit}, such that the loss converges monotonically if and only if the learning rate is smaller than η_{crit} (see Figure 3.1a).

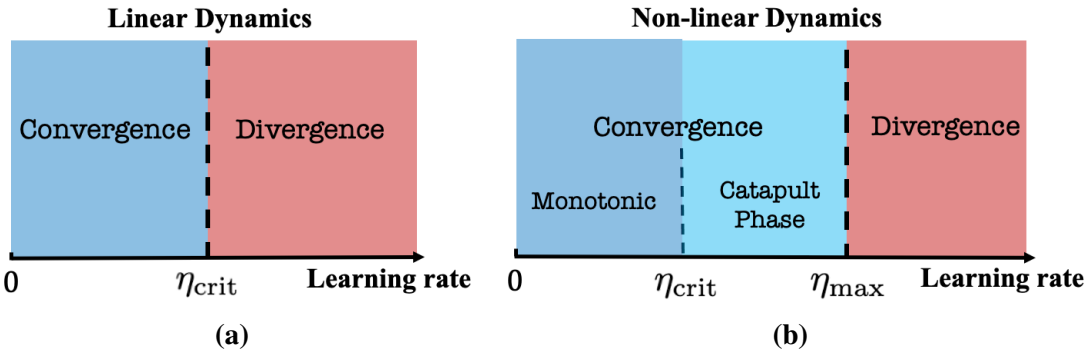
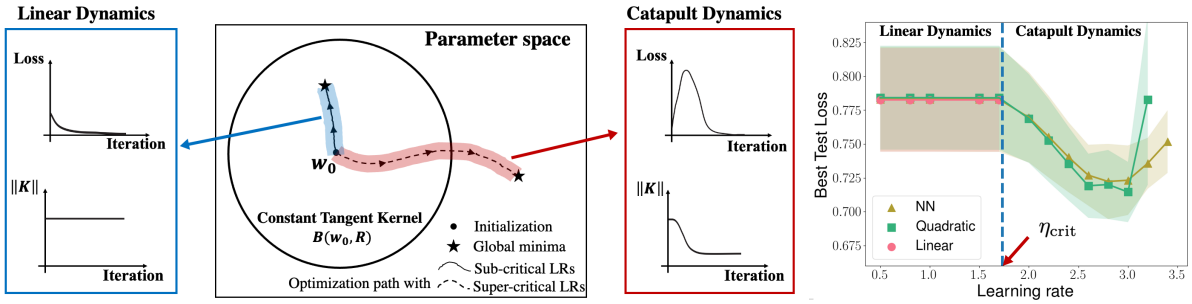


Figure 3.1. Optimization dynamics for linear and non-linear models based on choice of learning rate. (a): Linear models either converge monotonically if learning rate is less than η_{crit} and diverge otherwise. (b): Unlike linear models, *finitely wide neural networks* and *NQMs Eq. (3.2) (or general quadratic models Eq. (3.3))* can additionally observe a catapult phase when $\eta_{\text{crit}} < \eta < \eta_{\text{max}}$.

[1] Unless stated otherwise, we always consider the setting where models are trained with squared loss using gradient descent.



(a) Optimization dynamics for f (wide neural networks): linear dynamics and catapult dynamics.

(b) Generalization performance for f, f_{lin} and f_{quad}.

Figure 3.2. (a): Optimization dynamics of wide neural networks with sub-critical and super-critical learning rates. With sub-critical learning rates ($0 < \eta < \eta_{\text{crit}}$), the tangent kernel of wide neural networks is nearly constant during training, and the loss decreases monotonically. The whole optimization path is contained in the ball $B(\mathbf{w}_0, R) := \{\mathbf{w} : \|\mathbf{w} - \mathbf{w}_0\| \leq R\}$ with a finite radius R. With super-critical learning rates ($\eta_{\text{crit}} < \eta < \eta_{\text{max}}$), the catapult phase happens: the loss first increases and then decreases, along with a decrease of the norm of the tangent kernel . The optimization path goes beyond the finite radius ball. (b): Test loss of f_{quad}, f and f_{lin} plotted against different learning rates. With sub-critical learning rates, all three models have nearly identical test loss for any sub-critical learning rate. With super-critical learning rates, f and f_{quad} have smaller best test loss than the one with sub-critical learning rates. Experimental details are in Appendix B.14.

Recent work [64] showed that the training dynamics of a wide neural network $f(\mathbf{w};\boldsymbol{x})$ can be accurately approximated by that of a linear model $f_{\text{lin}}(\mathbf{w};\boldsymbol{x})$:

$$f_{\text{lin}}(\mathbf{w};\boldsymbol{x}) = f(\mathbf{w}_0;\boldsymbol{x}) + (\mathbf{w} - \mathbf{w}_0)^T \nabla f(\mathbf{w}_0;\boldsymbol{x}), \tag{3.1}$$

where $\nabla f(\mathbf{w}_0;\boldsymbol{x})$ denotes the gradient[2] of f with respect to trainable parameters $\mathbf{w}$ at an initial point $\mathbf{w}_0$ and input sample $\boldsymbol{x}$. This approximation holds for learning rates less than $\eta_{\text{crit}} \approx 2/\|\nabla f(\mathbf{w}_0;\boldsymbol{x})\|^2$, when the width is sufficiently large.

However, the training dynamics of finite width neural networks, f, can sharply differ from those of linear models when using large learning rates. A striking non-linear property of wide neural networks discovered in [65] is that when the learning rate is larger than η_{crit} but smaller than a certain maximum learning rate, η_{max}, gradient descent still converges but

[2]For non-differentiable functions, e.g. neural networks with ReLU activation functions, we define the gradient based on the update rule used in practice. Similarly, we use H_f to denote the second derivative of f in Eq. (3.2).

experiences a "catapult phase." Specifically, the loss initially grows exponentially and then decreases after reaching a large value, along with the decrease of the norm of tangent kernel (see Figure 3.2a), and therefore, such training dynamics are *non-linear* (see Figure 3.1b).

As linear models cannot exhibit such a catapult phase, under what models and conditions does this phenomenon arise? The work of [65] first observed the catapult phase phenomenon in finite width neural networks and analyzed this phenomenon for a two-layer linear neural network. However, a theoretical understanding of this phenomenon for general non-linear neural networks remains open. In this work, we utilize a quadratic model as a tool to shed light on the optimization and generalization discrepancies between finite and infinite width neural networks. We define *Neural Quadratic Model (NQM)* by the second order Taylor series expansion of $f(\mathbf{w};\boldsymbol{x})$ around the point $\mathbf{w}_0$:

$$\textbf{NQM}: \quad f_{\text{quad}}(\mathbf{w}) = f(\mathbf{w}_0) + (\mathbf{w}-\mathbf{w}_0)^T \nabla f(\mathbf{w}_0) + \frac{1}{2}(\mathbf{w}-\mathbf{w}_0)^T H_f(\mathbf{w}_0)(\mathbf{w}-\mathbf{w}_0). \tag{3.2}$$

Here in the notation we suppress the dependence on the input data $\boldsymbol{x}$, and $H_f(\mathbf{w}_0)$ is the Hessian of f with respect to $\mathbf{w}$ evaluated at $\mathbf{w}_0$. Note that $f_{\text{quad}}(\mathbf{w}) = f_{\text{lin}}(\mathbf{w}) + \frac{1}{2}(\mathbf{w}-\mathbf{w}_0)^T H_f(\mathbf{w}_0)(\mathbf{w}-\mathbf{w}_0)$.

Indeed, we note that NQMs are contained in a more general class of quadratic models:

$$\textbf{General Quadratic Model}: \qquad g(\mathbf{w};\boldsymbol{x}) = \mathbf{w}^T \phi(\boldsymbol{x}) + \frac{1}{2}\gamma \mathbf{w}^T \Sigma(\boldsymbol{x}) \mathbf{w}, \tag{3.3}$$

where $\mathbf{w}$ are trainable parameters and $\boldsymbol{x}$ is input data. We discuss the optimization dynamics of such general quadratic models in Section 3.3 and show empirically that they exhibit the catapult phase phenomenon in Appendix B.14. Note that the two-layer linear network analyzed in [65] is a special case of Eq. (3.3), when $\phi(\boldsymbol{x}) = 0$ (See Appendix B.13).

Main Contributions. We prove that NQMs, f_{quad}, which approximate shallow fully-connected ReLU activated neural networks, exhibit catapult phase dynamics. Specifically, we analyze the optimization dynamics of f_{quad} by deriving the evolution of f_{quad} and the tangent

kernel during gradient descent with squared loss, for a single training example and multiple uni-dimensional training examples. We identify three learning rate regimes yielding different optimization dynamics for f_{quad}, which are (1) converging monotonically (linear dynamics); (2) converging via a catapult phase (catapult dynamics); and (3) diverging. We provide a number of experimental results corroborating our theoretical analysis (See Section 3.3).

We then empirically show that NQMs, for the architectures of shallow (see Figure 3.2b as an example) and deep networks, have better test performances when catapult dynamics happens. While this was observed for some synthetic examples of neural networks in [65], we systematically demonstrate the improved generalization of NQMs across a range of experimental settings. Namely, we consider fully-connected and convolutional neural networks with ReLU and other activation functions trained with GD/SGD on multiple vision, speech and text datatsets (See Section 3.4).

To the best of our knowledge, our work is the first to analyze the non-linear wide neural networks in the catapult regime through the perspective of the quadratic approximation. While NQMs (or quadratic models) were proposed and analyzed in [95], our work focuses on the properties of NQMs in the large learning rate regime, which has not been discussed in [95]. Similarly, the following related works did not study catapult dynamics. [43] analyzed higher order approximations to neural networks under gradient flow (infinitesimal learning rates). [11] studied different quadratic models with randomized second order terms and [121] considered the loss in the quadratic form, where no catapult phase happens. A recent work showed the existence of the catapult phase in two-layer, homogenous networks [75].

Discontinuity in dynamics transition. In the ball $B(\mathbf{w}_0, R) := \{\mathbf{w} : \|\mathbf{w} - \mathbf{w}_0\| \leq R\}$ with constant radius $R > 0$, the transition to linearity of a wide neural network (with linear output layer) is continuous in the network width m. That is, the deviation from the network function to its linear approximation within the ball can be continuously controlled by the Hessian of the

network function, i.e. H_f, which scales with m [66]:

$$\|f(\mathbf{w}) - f_{\text{lin}}(\mathbf{w})\| \leq \sup_{\mathbf{w} \in B(\mathbf{w}_0, R)} \|H_f(\mathbf{w})\| R^2 = \tilde{O}(1/\sqrt{m}). \tag{3.4}$$

Using the inequality from Eq. (3.4), we obtain $\|f_{\text{quad}} - f_{\text{lin}}\| = \tilde{O}(1/\sqrt{m})$, hence f_{quad} transitions to linearity continuously as well in $B(\mathbf{w}_0, R)$[3]. Given the continuous nature of the transition to linearity, one may expect that the transition from non-linear dynamics to linear dynamics for f and f_{quad} is continuous in m as well. Namely, one would expect that the domain of catapult dynamics, $[\eta_{\text{crit}}, \eta_{\text{max}}]$, shrinks and ultimately converges to a single point, i.e., $\eta_{\text{crit}} = \eta_{\text{max}}$, as m goes to infinity, with non-linear dynamics turning into linear dynamics. However, as shown both analytically and empirically, the transition is *not* continuous, for both network functions f and NQMs f_{quad}, since the domain of the catapult dynamics can be independent of the width m (or γ). Additionally, the length of the optimization path of f in catapult dynamics grows with m since otherwise, the optimization path could be contained in a ball with a constant radius independent of m, in which f can be approximated by f_{lin}. Since the optimization of f_{lin} diverges in catapult dynamics, by the approximation, the optimization of f diverges as well, which contradicts the fact that the optimization of f can converge in catapult dynamics (See Figure 3.2a).

3.2 Notation and preliminary

We use bold lowercase letters to denote vectors and capital letters to denote matrices. We denote the set $\{1, 2, \cdots, n\}$ by $[n]$. We use $\|\cdot\|$ to denote the Euclidean norm for vectors and the spectral norm for matrices. We use $\odot$ to denote element-wise multiplication (Hadamard product) for vectors. We use $\lambda_{\text{max}}(A)$ and $\lambda_{\text{min}}(A)$ to denote the largest and smallest eigenvalue of a matrix A, respectively.

Given a model $f(\mathbf{w}; \boldsymbol{x})$, where $\boldsymbol{x}$ is input data and $\mathbf{w}$ are model parameters, we use $\nabla_{\mathbf{w}} f$ to represent the partial first derivative $\partial f(\mathbf{w}; \boldsymbol{x}) / \partial \mathbf{w}$. When clear from context, we let $\nabla f := \nabla_{\mathbf{w}} f$

[3] For general quadratic models in Eq. (3.3), the transition to linearity is continuously controlled by γ.

for ease of notation. We use H_f and $H_{\mathscr{L}}$ to denote the Hessian (second derivative matrix) of the function $f(\mathbf{w};\boldsymbol{x})$ and the loss $\mathscr{L}(\mathbf{w})$ with respect to parameters $\mathbf{w}$, respectively.

In the paper, we consider the following supervised learning task: given training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ with data $\boldsymbol{x}_i \in \mathbb{R}^d$ and labels $y_i \in \mathbb{R}$ for $i \in [n]$, we minimize the empirical risk with the squared loss $\mathscr{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^n (f(\mathbf{w};\boldsymbol{x}_i) - y_i)^2$. Here $f(\mathbf{w};\cdot)$ is a parametric family of models, e.g., a neural network or a kernel machine, with parameters $\mathbf{w} \in \mathbb{R}^p$. We use full-batch gradient descent to minimize the loss, and we denote trainable parameters $\mathbf{w}$ at iteration t by $\mathbf{w}(t)$. With constant step size (learning rate) η, the update rule for the parameters is:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \frac{d\mathscr{L}(\mathbf{w})}{d\mathbf{w}}(t), \quad \forall t \geq 0.$$

Definition 3.2.1 (Tangent Kernel)**.** The tangent kernel $K(\mathbf{w};\cdot,\cdot)$ of $f(\mathbf{w};\cdot)$ is defined as

$$K(\mathbf{w};\boldsymbol{x},\boldsymbol{z}) = \langle \nabla f(\mathbf{w};\boldsymbol{x}), \nabla f(\mathbf{w};\boldsymbol{z}) \rangle, \quad \forall \boldsymbol{x},\boldsymbol{z} \in \mathbb{R}^d. \tag{3.5}$$

In the context of the optimization problem with n training examples, the tangent kernel matrix $K \in \mathbb{R}^{n\times n}$ satisfies $K_{i,j}(\mathbf{w}) = K(\mathbf{w};\boldsymbol{x}_i,\boldsymbol{x}_j)$, $i,j \in [n]$. The critical learning rate for optimization is given as follows.

Definition 3.2.2 (Critical learning rate)**.** With an initialization of parameters $\mathbf{w}_0$, the critical learning rate of $f(\mathbf{w};\cdot)$ is defined as

$$\eta_{\text{crit}} := 2/\lambda_{\max}(H_{\mathscr{L}}(\mathbf{w}_0)). \tag{3.6}$$

A learning rate η is said to be *sub-critical* if $0 < \eta < \eta_{\text{crit}}$ or *super-critical* if $\eta_{\text{crit}} < \eta < \eta_{\max}$. Here $\eta_{\max}$ is the maximum leaning rate such that the optimization of $\mathscr{L}(\mathbf{w})$ initialized at $\mathbf{w}_0$ can converge.

Dynamics for Linear models

When f is linear in $\mathbf{w}$, the gradient, ∇f, and tangent kernel are constant: $K(\mathbf{w}(t)) = K(\mathbf{w}_0)$. Therefore, gradient descent dynamics are:

$$F(\mathbf{w}(t+1)) - \boldsymbol{y} = (I - \eta K(\mathbf{w}_0))(F(\mathbf{w}(t)) - \boldsymbol{y}), \quad \forall t \geq 0, \tag{3.7}$$

where $F(\mathbf{w}_0) = [f_1(\mathbf{w}_0), ..., f_n(\mathbf{w}_0)]^T$ with $f_i(\mathbf{w}_0) = f(\mathbf{w}_0; \boldsymbol{x}_i)$.

Noting that $H_{\mathscr{L}}(\mathbf{w}_0) = \nabla F(\mathbf{w}_0)^T \nabla F(\mathbf{w}_0)$ and tangent kernel $K(\mathbf{w}_0) = \nabla F(\mathbf{w}_0) \nabla F(\mathbf{w}_0)^T$ share the same positive eigenvalues, we have $\lambda_{\max}(H_{\mathscr{L}}(\mathbf{w}_0)) = \lambda_{\max}(K(\mathbf{w}_0))$, and hence,

$$\eta_{\text{crit}} = 2/\lambda_{\max}(K(\mathbf{w}_0)). \tag{3.8}$$

Therefore, from Eq. equation 3.7, if $0 < \eta < \eta_{\text{crit}}$, the loss $\mathscr{L}$ decreases monotonically and if $\eta > \eta_{\text{crit}}$, the loss $\mathscr{L}$ keeps increasing. Note that the critical and maximum learning rates are equal in this setting.

3.3 Optimization dynamics in Neural Quadratic Models

In this section, we analyze the gradient descent dynamics of the NQM corresponding to a two-layer fully-connected neural network. We show that, unlike a linear model, the NQM exhibits a catapult dynamics: the loss increases at the early stage of training then decreases afterwards. We further show that the top eigenvalues of the tangent kernel typically become smaller as a consequence of the catapult.

Neural Quadratic Model (NQM). Consider the NQM that approximates the following two-layer neural network:

$$f(\mathbf{u}, \mathbf{v}; \boldsymbol{x}) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma\left(\frac{1}{\sqrt{d}} \mathbf{u}_i^T \boldsymbol{x}\right), \tag{3.9}$$

where $\mathbf{u}_i \in \mathbb{R}^d$, $v_i \in \mathbb{R}$ for $i \in [m]$ are trainable parameters, $\boldsymbol{x} \in \mathbb{R}^d$ is the input, and $\sigma(\cdot)$ is the ReLU activation function. We initialize $\mathbf{u}_i \sim \mathcal{N}(0, I_d)$ and $v_i \in \text{Unif}[\{-1,1\}]$ for each i independently. Letting $g(\mathbf{u},\mathbf{v};\boldsymbol{x}) := f_{\text{quad}}(\mathbf{u},\mathbf{v};\boldsymbol{x})$, this NQM has the following expression (See the full derivation in Appendix B.1):

$$g(\mathbf{u},\mathbf{v};\boldsymbol{x}) = f(\mathbf{u}_0,\mathbf{v}_0;\boldsymbol{x}) + \frac{1}{\sqrt{md}}\sum_{i=1}^{m} v_{0,i}(\mathbf{u}_i-\mathbf{u}_{0,i})^T\boldsymbol{x}\mathbb{1}_{\left\{\mathbf{u}_{0,i}^T\boldsymbol{x}\geq 0\right\}} + \frac{1}{\sqrt{md}}\sum_{i=1}^{m}(v_i-v_{0,i})\sigma\left(\mathbf{u}_{0,i}^T\boldsymbol{x}\right) + \frac{1}{\sqrt{md}}\sum_{i=1}^{m}(v_i-v_{0,i})(\mathbf{u}_i-\mathbf{u}_{0,i})^T\boldsymbol{x}\mathbb{1}_{\left\{\mathbf{u}_{0,i}^T\boldsymbol{x}\geq 0\right\}}. \tag{3.10}$$

Given training data $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$, we minimize the empirical risk with the squared loss $\mathscr{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^n (g(\mathbf{w};\boldsymbol{x}_i)-y_i)^2$ using GD with constant learning rate η. Throughout this section, we denote $g(\mathbf{u}(t),\mathbf{v}(t);\boldsymbol{x})$ by $g(t)$ and its tangent kernel $K(\mathbf{u}(t),\mathbf{v}(t))$ by $K(t)$, where t is the iteration of GD. We assume $\|\boldsymbol{x}_i\| = O(1)$ and $|y_i| = O(1)$ for $i \in [n]$, and we assume the width of f is much larger than the input dimension d and the data size n, i.e., $m \gg \max\{d,n\}$. Hence, d and n can be regarded as small constants. In the whole paper, we use the big-O and small-o notation with respect to the width m. Below, we start with the single training example case, which already showcases the non-linear dynamics of NQMs.

Catapult dynamics with a single training example

In this subsection, we consider training dynamics of NQM Eq. (3.10) with a single training example $(\boldsymbol{x}, y)$ where $\boldsymbol{x} \in \mathbb{R}^d$ and $y \in \mathbb{R}$. In this case, the tangent kernel matrix K reduces to a scalar, and we denote K by λ to distinguish it from a matrix.

By gradient descent with step size η, the updates for $g(t)-y$ and $\lambda(t)$, which we refer to as dynamics equations, can be derived as follows (see the derivation in Appendix B.2):

Dynamics equations.

$$g(t+1)-y=\left(1-\eta\lambda(t)+\underbrace{\frac{\|\boldsymbol{x}\|^2}{md}\eta^2(g(t)-y)g(t)}_{R_g(t)}\right)(g(t)-y):=\mu(t)(g(t)-y), \tag{3.11}$$

$$\lambda(t+1)=\lambda(t)-\underbrace{\eta\frac{\|\boldsymbol{x}\|^2}{md}(g(t)-y)^2\left(4\frac{g(t)}{g(t)-y}-\eta\lambda(t)\right)}_{R_\lambda(t)},\quad \forall t\geq 0. \tag{3.12}$$

Note that as the loss is given by $\mathscr{L}(t)=\frac{1}{2}(g(t)-y)^2$, to understand convergence, it suffices to analyze the dynamics equations above. Compared to the linear dynamics Eq. (3.7), this non-linear dynamics has extra terms $R_g(t)$ and $R_\lambda(t)$, which are induced by the non-linear term in the NQM. We will see that the convergence of gradient descent depends on the scale and sign of $R_g(t)$ and $R_\lambda(t)$. For example, for constant learning rate that is slightly larger than η_{crit} (which would result in divergence for linear models), $R_\lambda(t)$ stays positive during training, resulting in both monotonic decrease of tangent kernel λ and the loss.

As $\lambda(t)=\lambda_0-\sum_{\tau=0}^{t-1}R_\lambda(\tau)$, to track the scale of $|\mu(t)|$, we will focus on the scale and sign of $R_g(t)$ and $R_\lambda(t)$ in the following analysis. For the scale of λ_0, which is non-negative by Definition 3.2.1, we can show that with high probability over random initialization, $|\lambda_0|=\Theta(1)$ (see Appendix B.9). And $|g(0)|=O(1)$ with high probability as well [22]. Therefore the following discussion is with high probability over random initialization. We start by establishing monotonic convergence for sub-critical learning rates.

Monotonic convergence: sub-critical learning rates ($\eta<2/\lambda_0=\eta_{\text{crit}}$)

The key observation is that when $|g(t)|=O(1)$, and $\lambda(t)=\Theta(1)$, $|R_g(t)|$ and $|R_\lambda(t)|$ are of the order $o(1)$. Then, the dynamics equations approximately reduce to the ones of linear

dynamics:

$$g(t+1)-y=(1-\eta\lambda(t)+o(1))\,(g(t)-y),$$
$$\lambda(t+1)=\lambda(t)+o(1).$$

Note that at initialization, the output satisfies $|g(0)|=O(1)$, and we have shown $\lambda_0=\Theta(1)$. With the choice of η, we have for all $t\geq 0$, $|\mu(t)|=|1-\eta\lambda(t)+o(1)|<1$; hence, $|g(t)-y|$ decreases monotonically. The cumulative change on the tangent kernel will be $o(1)$, i.e., $\sum_t |R_\lambda(t)|=o(1)$, since for all t, $|R_\lambda(t)|=O(1/m)$ and the loss decreases exponentially hence $\sum|R_\lambda(t)|=O(1/m)\cdot\log O(1)=o(1)$. See Appendix B.3 for a detailed discussion.

Catapult convergence: super-critical learning rates ($\eta_{\rm crit}=2/\lambda_0<\eta<4/\lambda_0=\eta_{\max}$)

The training dynamics are given by the following theorem.

Theorem 3.3.1 (Catapult dynamics on a single training example)**.** *Consider training the NQM Eq. (3.10) with squared loss on a single training example by GD. With a super-critical learning rate* $\eta\in\left[\frac{2+\varepsilon}{\lambda_0},\frac{4-\varepsilon}{\lambda_0}\right]$ *where* $\varepsilon=\Theta\left(\frac{\log m}{\sqrt{m}}\right)$*, the catapult happens: with high probability over random initialization, the loss increases to the order of* $\Omega\left(\frac{m(\eta\lambda_0-2)^2}{\log m}\right)$ *then decreases to* $O(1)$.

Proof of Theorem 3.3.1. We use the following transformation of the variables to simplify notations.

$$u(t)=\frac{\|\boldsymbol{x}\|^2}{md}\eta^2(g(t)-y)^2,\quad w(t)=\frac{\|\boldsymbol{x}\|^2}{md}\eta^2(g(t)-y)y,\quad v(t)=\eta\lambda(t).$$

Then the Eq. (3.11) and Eq. (3.12) are reduced to

$$u(t+1)=(1-v(t)+u(t)+w(t))^2u(t):=\kappa(t)u(t), \tag{3.13}$$
$$v(t+1)=v(t)-u(t)(4-v(t))-4w(t). \tag{3.14}$$

At initialization, since $|g(0)|=O(1)$, we have $u(0)=O\left(\frac{1}{m}\right)$ and $w(0)=O\left(\frac{1}{m}\right)$. Note that by

definition, for all $t \geq 0$, $u(t) \geq 0$ and we have $v(t) \geq 0$ since $\lambda(t)$ is the tangent kernel for a single training example.

In the following, we will analyze the above dynamical equations. To make the analysis more understandable, we separate the dynamics during training into increasing phase and decreasing phase. We denote $\delta := (\eta - \eta_{\text{crit}})\lambda_0 = \eta\lambda_0 - 2$.

Increasing phase. In this phase, $|u(t)|$ increases exponentially from $O\left(\frac{1}{m}\right)$ to $\Theta\left(\frac{\delta^2}{\log m}\right)$ and $|v(t) - v(0)| = O\left(\frac{\delta}{\log m}\right)$. This can be shown by the following lemma.

Lemma 3.3.2. *For $T > 0$ such that $\sup_{t\in[0,T]} u(t) = O\left(\frac{\delta^2}{\log m}\right)$, $u(t)$ increases exponentially with $\inf_{t\in[0,T]} \kappa(t) \geq \left(1 + \delta - O\left(\frac{\delta}{\log m}\right)\right)^2 > 1$ and $\sup_{t\in[0,T]} |v(t) - v(0)| = O\left(\frac{\delta}{\log m}\right)$.*

Proof. See the proof in Section B.4. □

After the increasing phase, based on the order of $u(t)$ we can infer the order of loss is $\Theta\left(\frac{m\delta^2}{\log m}\right)$.

Decreasing phase. When $u(t)$ is sufficiently large, $v(t)$ will have non-negligible decrease which leads to the decreasing of $\kappa(t)$, hence in turn making $u(t)$ decrease as well. Consequently, we have:

Lemma 3.3.3. *There exists $T^* > 0$ such that $u(T^*) = O\left(\frac{1}{m}\right)$.*

Proof. See the proof in Section B.5. □

Then accordingly, the loss is of the order $O(1)$. □

Once the loss decreases to the order of $O(1)$, the catapult finishes and we in general have $\eta < 2/\lambda(t)$ as $|\mu(t)| = |1 - \eta\lambda(t) + R_{\mathbf{g}}(t)| < 1$ where $|R_{\mathbf{g}}(t)| = O(\mathscr{L}(t)/m) = O(1/m)$. Therefore the training dynamics fall into linear dynamics, and we can use the same analysis for sub-critical learning rates for the remaining training dynamics. The stableness of the steady-state equilibria of dynamical equations can be guaranteed by the following:

Theorem 3.3.4. *For dynamical equations Eq. (3.11) and (3.12), the stable steady-state equilibria satisfy* $g(t) = y$ *(i.e.,loss is* 0*), and* $\lambda(t) \in [\varepsilon, 2/\eta - \varepsilon]$ *with* $\varepsilon = \Theta(\log m/\sqrt{m})$.

Divergence ($\eta > \eta_{\max} = 4/\lambda_0$)

Initially, it follows the same dynamics with that in the increasing phase in catapult convergence: $|g(t) - y|$ increases exponentially as $|\mu(t)| > 1$ and the $\lambda(t)$ almost does not change as $R_\lambda(t)$ is small. However, note that $R_\lambda(t) > 0$ since 1) $g(t)/(g(t) - y) \approx 1$ when $g(t)$ becomes large and 2) $\eta > 4/\lambda(t)$. Therefore, $\lambda(t)$ keeps increasing during training, which consequently leads to the divergence of the optimization. See Appendix B.7 for a detailed discussion.

Catapult dynamics with multiple training examples

In this subsection we show the catapult phase will happen for NQMs Eq. (3.9) with multiple training examples. We assume unidimensional input data, which is common in the literature and simplifies the analysis for neural networks (see for example [109, 97]).

Assumption 3.3.5. The input dimension $d = 1$ and not all x_i is 0, i.e., $\sum |x_i| > 0$.

We similarly analyze the dynamics equations with different learning rates for multiple training examples (see the derivation of Eq. (B.7) and (B.8) in Appendix) which are update equations of $\mathbf{g}(t) - \boldsymbol{y}$ and $K(t)$. And similarly, we show there are three training dynamics: monotonic convergence, catapult convergence and divergence.

In the analysis, we consider the training dynamics projected to two orthogonal eigenvectors of the tangent kernel, i.e., $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, and we show with different learning rates, the catapult phase can occur only in the direction of $\boldsymbol{p}_1$, or occur in both directions. We consider the case where $2/\lambda_2(0) < 4/\lambda_1(0)$ hence the catapult can occur in both directions. The analysis for the other case can be directly obtained from our results. We denote the loss projected to $\boldsymbol{p}_i$ by $\Pi_i \mathscr{L} := \frac{1}{2} \langle \mathbf{g} - \boldsymbol{y}, \boldsymbol{p}_i \rangle^2$ for $i = 1, 2$. We have $\Pi_i \mathscr{L}(0) = O(1)$ with high probability over random initialization of weights.

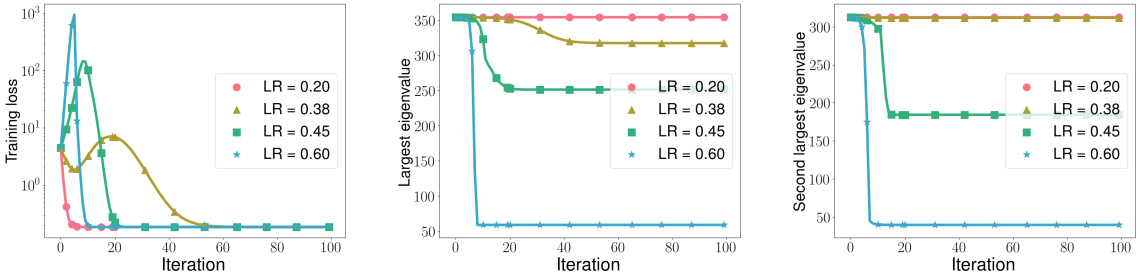
We formulate the result for the catapult dynamics, which happens when training with super-critical learning rates, into the following theorem, and defer the proof of it and the full discussion of training dynamics to Appendix B.11.

Theorem 3.3.6 (Catapult dynamics on multiple training examples)**.** *Supposing Assumption 3.3.5 holds, consider training the NQM Eq. (3.10) with squared loss on multiple training examples by GD. Then, with high probability over random initialization we have*

1. *with* $\eta \in \left[\frac{2+\varepsilon}{\lambda_1(0)}, \frac{2-\varepsilon}{\lambda_2(0)}\right]$, *the catapult only occurs in eigendirection* $\boldsymbol{p}_1$*:* $\Pi_1 \mathscr{L}$ *increases to the order of* $\Omega\left(\frac{m(\eta\lambda_1(0)-2)^2}{\log m}\right)$ *then decreases to* $O(1)$*;*
2. *with* $\eta \in \left[\frac{2+\varepsilon}{\lambda_2(0)}, \frac{4-\varepsilon}{\lambda_1(0)}\right]$, *the catapult occurs in both eigendirections* $\boldsymbol{p}_1$ *and* $\boldsymbol{p}_2$*:* $\Pi_i \mathscr{L}$ *for* $i = 1, 2$ *increases to the order of* $\Omega\left(\frac{m(\eta\lambda_i(0)-2)^2}{\log m}\right)$ *then decreases to* $O(1)$*,*

where $\varepsilon = \Theta\left(\frac{\log m}{\sqrt{m}}\right)$.

We verify the results for multiple training examples via the experiments in Figure 3.3.



(a) Training loss **(b)** Largest eigenvalue of tangent kernel **(c)** Second largest eigenvalue of tangent kernel

Figure 3.3. Training dynamics of NQMs for multiple examples case with different learning rates. By our analysis, two critical values are $2/\lambda_1(0) = 0.37$ and $2/\lambda_2(0) = 0.39$. When $\eta < 0.37$, linear dynamics dominate hence the kernel is nearly constant; when $0.37 < \eta < 0.39$, the catapult phase happens in $\boldsymbol{p}_1$ and only $\lambda_1(t)$ decreases; when $0.39 < \eta < \eta_{\max}$, the catapult phase happens in $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ hence both $\lambda_1(t)$ and $\lambda_2(t)$ decreases. The experiment details can be found in Appendix B.14.

Connection to general quadratic models and wide neural networks

General quadratic models. As mentioned in the introduction, NQMs are contained in a general class of quadratic models of the form given in Eq. (3.3). Additionally, we show that the two-layer linear neural network analyzed in [65] is a special case of Eq. (3.3), and we provide a more general condition for such models to have catapult dynamics in Appendix B.13. Furthermore, we empirically observe that a broader class of quadratic models g can have catapult dynamics simply by letting $\phi(\boldsymbol{x})$ and Σ be random and assigning a small value to γ (See Appendix B.14).

Wide neural networks. We have seen that NQMs, with fixed Hessian, exhibit the catapult phase phenomenon. Therefore, the change in the Hessian of wide neural networks during training is not required to produce the catapult phase. We will discuss the high-level idea of analyzing the catapult phase for a general NQM with large learning rates, and empirically show that this idea applies to neural networks. We train an NQM Eq. (3.2) f_{quad} on n data points $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R}$ with GD. The dynamics equations take the following form:

$$\mathbf{f}_{\text{quad}}(t+1) - \boldsymbol{y} = \left(I - \eta K(t) + \underbrace{\frac{1}{2}\eta^2 G(t) \nabla \mathbf{f}_{\text{quad}}(t)^T}_{R_{\mathbf{f}_{\text{quad}}}(t)} \right) (\mathbf{f}_{\text{quad}}(t) - \boldsymbol{y}), \tag{3.15}$$

$$K(t+1) = K(t) - \underbrace{\frac{1}{4}\eta \left(4G(t) \nabla \mathbf{f}_{\text{quad}}(t)^T - \eta G(t) G(t)^T \right)}_{R_K(t)}, \tag{3.16}$$

where $G_{i,:}(t) = (\mathbf{f}_{\text{quad}}(t) - \boldsymbol{y})^T \nabla \mathbf{f}_{\text{quad}}(t) H_f(\boldsymbol{x}_i) \in \mathbb{R}^m$ for $i \in [n]$.

In our analysis for f_{quad} which approximates two-layer networks in Section 3.3, we show that catapult dynamics occur in the top eigenspace of the tangent kernel. Specifically, we analyze the dynamics equations confined to the top eigendirection of the tangent kernel $\boldsymbol{p}_1$ (i.e, $\Pi_1 \mathscr{L}$ and $\lambda_1(t)$). We show that $\boldsymbol{p}_1^T R_{\mathbf{f}_{\text{quad}}} \boldsymbol{p}_1$ and $\boldsymbol{p}_1^T R_K \boldsymbol{p}_1$ scale with the loss and remain positive when the loss becomes large, therefore $\boldsymbol{p}_1^T K \boldsymbol{p}_1$ (i.e., $\lambda_{\max}(K)$) as well as the loss will be driven down,

and consequently we yield catapult convergence.

We empirically verify catapults indeed happen in the top eigenspace of the tangent kernel for additional NQMs and wide neural networks in Appendix B.14. Furthermore, a similar behaviour of top eigenvalues of the tangent kernel with the one for NQMs is observed for wide neural networks when training with different learning rates (See Figure B.1 in Appendix B.14).

3.4 Quadratic models parallel neural networks in generalization

In this section, we empirically compare the test performance of three different models considered in this paper upon varying learning rate. In particular, we consider (1) the NQM, f_{quad}; (2) corresponding neural networks, f; and (3) the linear model, f_{lin}.

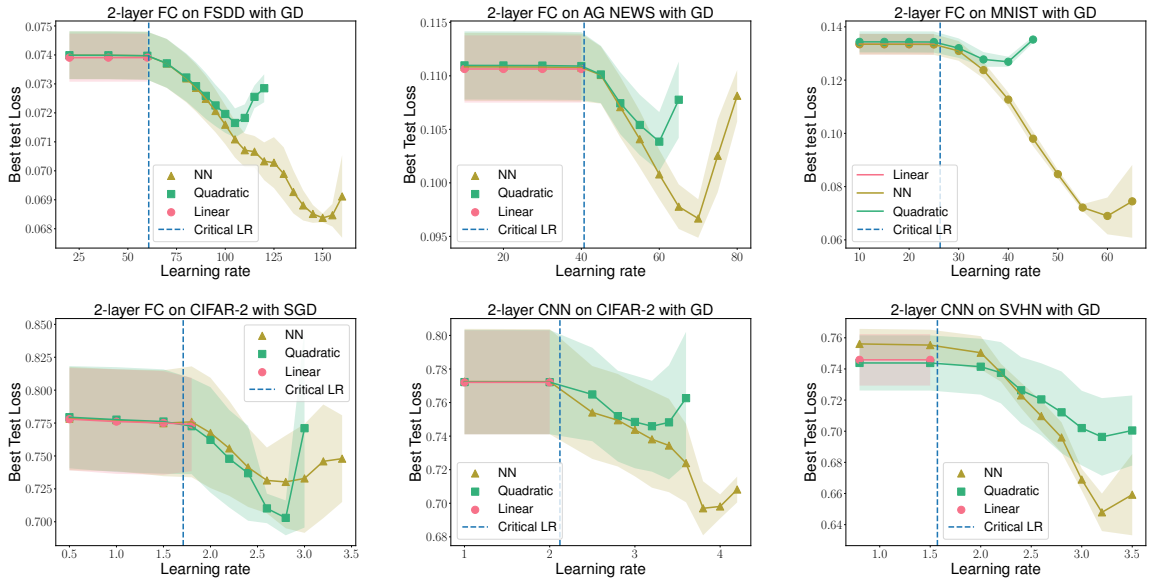


Figure 3.4. Best test loss plotted against different learning rates for $f(\mathbf{w})$, $f_{\text{lin}}(\mathbf{w})$ and $f_{\text{quad}}(\mathbf{w})$ across a variety of datasets and network architectures.

We implement our experiments on 3 vision datasets: CIFAR-2 (a 2-class subset of CIFAR-10 [59]), MNIST [61], and SVHN (The Street View House Numbers) [79], 1 speech dataset: Free Spoken Digit dataset (FSDD) [47] and 1 text dataset: AG NEWS [31].

In all experiments, we train the models by minimizing the squared loss using standard GD/SGD with constant learning rate η. We report the best test loss achieved during the training process with each learning rate. Experimental details can be found in Appendix B.14. We also report the best test accuracy in Appendix B.14. For networks with 3 layers, see Appendix B.14. From the experimental results, we observe the following:

Sub-critical learning rates. In accordance with our theoretical analyses, we observe that all three models have nearly identical test loss for any sub-critical learning rate. Specifically, note that as the width m increases, f and f_{quad} will transition to linearity in the ball $B(\mathbf{w}_0, R)$:

$$\|f - f_{\text{lin}}\| = \tilde{O}(1/\sqrt{m}), \quad \|f_{\text{quad}} - f_{\text{lin}}\| = \tilde{O}(1/\sqrt{m}),$$

where $R > 0$ is a constant which is large enough to contain the optimization path with respect to sub-critical learning rates. Thus, the generalization performance of these three models will be similar when m is large, as shown in Figure 3.4.

Super-critical learning rates. The best test loss of both $f(\mathbf{w})$ and $f_{\text{quad}}(\mathbf{w})$ is consistently smaller than the one with sub-critical learning rates, and decreases for an increasing learning rate in a range of values beyond η_{crit}, which was observed for wide neural networks in [65].

As discussed in the introduction, with super-critical learning rates, both f_{quad} and f can be observed to have catapult phase, while the loss of f_{lin} diverges. Together with the similar behaviour of f_{quad} and f in generalization with super-critical learning rates, we believe NQMs are a better model to understand f in training and testing dynamics, than the linear approximation f_{lin}.

In Figure 3.4 we report the results for networks with ReLU activation function. We also implement the experiments using networks with Tanh and Swish [93] activation functions, and observe the same phenomena in generalization for f, f_{lin} and f_{quad} (See Appendix B.14).

3.5 Summary and Discussion

Summary. In this paper, we use quadratic models as a tool to better understand optimization and generalization properties of finite width neural networks trained using large learning rates. Notably, we prove that quadratic models exhibit properties of neural networks such as the catapult phase that cannot be explained using linear models, which importantly includes linear approximations to neural networks given by the neural tangent kernel. Interestingly, we show empirically that quadratic models mimic the generalization properties of neural networks when trained with large learning rate, and that such models perform better than linearized neural networks.

Future directions. As quadratic models are more analytically tractable than finite width neural networks, these models open further avenues for understanding the good performance of finite width networks in practice. In particular, one interesting direction of future work is to understand the change in the kernel corresponding to a trained quadratic model. As we showed, training a quadratic model with large learning rate causes a decrease in the eigenvalues of the neural tangent kernel, and it would be interesting to understand the properties of this changed kernel that correspond with improved generalization. Indeed, prior work [71] has analyzed the properties of the "after kernel" corresponding to finite width neural networks, and it would be interesting to observe whether similar properties hold for the kernel corresponding to trained quadratic models.

Another interesting avenue of research is to understand whether quadratic models can be used for representation learning. Indeed, prior work [116] argues that networks in the neural tangent kernel regime do not learn useful representations of data through training. As quadratic models trained with large learning rate can already exhibit nonlinear dynamics and better capture generalization properties of finite width networks, it would be interesting to understand whether such models learn useful representations of data as well.

3.6 Acknowledgements

Chapter 3, in full, is a reprint of Libin Zhu, Chaoyue Liu, Adityanarayanan Radhakrishnan, and Mikhail Belkin. “Quadratic models for understanding catapult dynamics of neural networks. ” ICLR 2024. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Feature learning of catapult dynamics

In this chapter, we focus on understanding the dynamics of training neural networks using Stochastic Gradient Descent (SGD) with large learning rates—a practice that has shown empirical success but remains partially unexplained by current optimization theories. Notably, significant spikes in training loss are commonly observed in such settings, prompting key questions about the mechanisms at play and their implications for model generalization. Here, we connect these loss spikes, catapult dynamics, and the observed superior generalization with smaller batch sizes in SGD into a cohesive framework. This chapter aims to provide a clear explanation of these phenomena and their interrelations, enhancing our comprehension of neural network training. Through combining theoretical insights with empirical evidence, we show how these dynamics influence the learning process and impact the generalization performance of deep learning models from the perspective of feature learning.

4.1 Introduction

Training algorithms are a key ingredient to the success of deep learning. Stochastic gradient descent (SGD) [94], a stochastic variant of gradient descent (GD), has been effective in finding parameters that yield good test performance despite the complicated nonlinear nature of neural networks. Empirical evidence suggests that training networks using SGD with a larger learning rate results in better predictors [27, 99, 29]. In such settings, it is common to observe

significant spikes in the training loss [60, 96, 56, 115] (see Figure 4.1 as an example). One may not *a priori* expect the training loss to decrease back to its "pre-spike" level after a large spike. Yet, this is what is commonly observed in training. Furthermore, the resulting "post-spike" model can yield improved generalization performance [36, 119, 42].

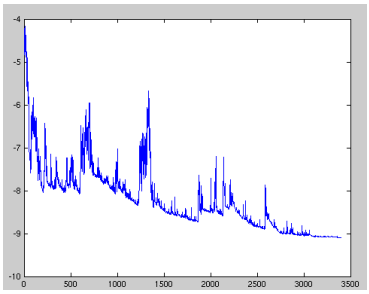


Figure 4.1. Spikes in training loss when optimized using SGD (x-axis: iteration). (Source: Wikipedia).

Why do spikes occur during training, and how do the spikes relate to generalization?

In this work, we answer these questions by connecting three common but seemingly unrelated phenomena in deep learning:

1. Spikes in the training loss of SGD,
2. Catapult dynamics in GD [65],
3. Better generalization when training networks with small batch SGD as opposed to larger batch size or GD.

In particular, we show that spikes in the training loss of SGD are caused by catapult dynamics, which were originally characterized in [65] as a single spike in the loss when training with GD and large learning rate. We then show that smaller batch size in SGD results in a greater number of catapults. We connect the optimization phenomena of catapults to generalization by showing that catapults improve generalization through increasing feature learning, which is quantified by the alignment between the Average Gradient Outer Product (AGOP) of the trained network

Table 4.1. Smaller SGD batch size leads to a higher (better) AGOP alignment and smaller (better) test loss. The results correspond to Figure 4.9a (a synthetic dataset).

Batch size	AGOP alignment	Test loss
2000 (GD)	0.81	0.74
50	0.84	0.71
10	0.89	0.59
5	0.95	0.42

and the true AGOP [35, 40, 112, 105, 91]. Since decreasing batch size in SGD leads to more catapults, our result implies that SGD with small batch size yields improved generalization (see Table 4.1 for an example). We outline our specific contributions in the context of optimization and generalization below.

Optimization. We demonstrate that spikes in the training loss, specifically measured by Mean Squared Error, occur in the top eigenspace of the Neural Tangent Kernel, a kernel resulting from the linearization of a neural network [46]. Namely, we project the residual (i.e., the difference between the predicted output and the target output) to the top eigenspace of the tangent kernel and show that spikes in the total loss function correspond to the spikes in the components of the loss in this low-dimensional subspace (see Section 4.3). In contrast, the components of the loss in the space spanned by the remaining eigendirections decrease monotonically. Thus, the catapult phenomenon occurs in the span of the top eigenvectors while the remaining eigendirections are not affected. This explains why the loss drops quickly to pre-spike levels, namely the loss value right before the spike, from the peak of the spike. We further show that multiple catapults can be generated in GD by increasing the learning rate during training (see Section 4.3). While prior work [65] observed that the spectral norm of the tangent kernel decreased for one catapult, we extend that observation by showing that the norm decreases after each catapult.

We further provide evidence for catapults in SGD with large learning rates (see Sec-

tion 4.3). Namely, we demonstrate that spikes in the loss when training with SGD correspond to catapults by showing that similarly to GD:

1. The spikes occur in the top eigenspace of the tangent kernel,

2. Each spike results in a decrease in the spectral norm of the tangent kernel.

We corroborate our findings across several network architectures including Wide ResNet [119] and ViT [20] and datasets including CIFAR-10 [59] and SVHN [79].

Moreover, as small batch size leads to higher variance in the eigenvalues of the tangent kernel for any given batch, small batch size results in an increased number of catapults.

Generalization. We posit that catapults improve the generalization performance by alignment between the AGOP of the trained network with that of the true model[1]. The AGOP identifies the features that lead to greatest change in predictor output when perturbed and has been recently posited as the mechanism through which neural networks learn features [91, 13]. We use AGOP alignment to provide an explanation for prior empirical results from [65, 124] showing that a single catapult can lead to better test performance in GD. Moreover, we extend these prior results to show that test performance continues to improve as the number of catapults increases in GD. Thus, we show that decreasing batch size with SGD can lead to better test performance due to an increase in the number of catapults. We further demonstrate that AGOP alignment is an effective measure of generalization by showing that test error is highly correlated with the AGOP alignment when training on the same task across different optimization algorithms including Adagrad [23], Adadelta [120] and Adam [57] etc. We corroborate our findings on CelebA [69] and SVHN [79] datasets and architectures including fully-connected and convolutional neural networks. See Section 4.4.

Related works

Linear dynamics and catapult phase phenomenon. Recent studies have shown that (stochastic) GD for wide neural networks provably converges to global minima with an appropri-

[1]When the underlying model is not available, we use a SOTA model as a substitute.

ately small learning rate [21, 126, 67]. These works leveraged the fact that neural networks with sufficiently large widths, under specific initialization conditions, can be accurately approximated by their linearization obtained by the first-order Taylor expansion [46, 66, 68, 123]. Therefore, their training dynamics are close to the dynamics of the corresponding linear models, under which the training loss decreases monotonically. Such a training regime is commonly referred to as the kernel regime. However, under the same setup of the kernel regime except using a large learning rate, GD will experience a catapult phase [65]: the training loss increases drastically in the beginning stage of training then decreases, while GD still converges. Recent studies focusing on understanding catapults in GD include [124], which considers quadratic approximations of neural networks, and [75], examining two-layer homogeneous neural networks. Our work investigates the impact of catapults in SGD on both optimization and generalization through experimental approaches.

Edge of stability. A phenomenon related to catapults is the "Edge of Stability" (EoS), which describes the dynamics of the training loss and the sharpness, i.e., eigenvalues of the Hessian of the loss, at the later stage of training networks with GD [16] and SGD [50, 49]. There is a growing body of work analyzing the mechanism of EoS in training dynamics with GD [9, 4, 17, 108, 3, 2], and SGD [53]. It was conjectured in [16] that at EoS for GD the spikes in the training loss are micro-catapults. Our work provides evidence that the spikes in the training loss using SGD are catapults and demonstrates the connection between the loss spikes and feature learning.

Generalization and sharpness. It has been observed that networks trained with SGD generalize better than GD, and smaller batch sizes often lead to better generalization performance [62, 55, 30, 48, 73, 54, 100]. Empirically, it has been observed that training with SGD results in flat minima [38, 39]. However, we noticed that it is not always the case, e.g., [28]. A number of works been argued that flatness of the minima is connected to the generalization performance [80, 111, 58, 114, 52, 19], however we know only one theoretical result in that direction [18]. Training algorithms aiming to find a flat minimum were developed and shown to

perform well on a variety of tasks [45, 25]. As an explanation for empirically observed improved generalization, prior work [65] argued that a single catapult with GD resulted in flatter minima. In this work we propose a different line of investigation to understand generalization properties of GD-based algorithms based on feature learning as measured by the alignment with AGOP.

4.2 Preliminaries

Notation. We use bold letters (e.g., $\mathbf{w}$) to denote vectors and capital letters (e.g., K) to denote matrices. For a matrix, we use $\|\cdot\|_F$ to denote its Frobenius norm and use $\|\cdot\|_2$ to denote its spectral norm. For trainable parameters, we use superscript t, as in $\mathbf{w}^t$, to denote the time stamp during training. We use the big-O notation $O(\cdot)$ to hide constants, and use $\tilde{O}(\cdot)$ to further hide logarithmic factors. For a map $f(\mathbf{w}) : \mathbb{R}^p \to \mathbb{R}^c$, we use $\nabla_{\mathbf{w}} f(\mathbf{v})$ and $\nabla^2_{\mathbf{w}} f(\mathbf{v})$ to denote the first and second order derivative of f w.r.t. $\mathbf{w}$ evaluated at $\mathbf{v}$ respectively.

Optimization task. Consider a parameterized model $f(\mathbf{w};\cdot) : \mathbb{R}^p \to \mathbb{R}$ (e.g., a neural network) with parameters $\mathbf{w}$ and a training dataset $\mathscr{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ with data $\boldsymbol{x}_i \in \mathbb{R}^d$ and labels $y_i \in \mathbb{R}$ for $i \in [n]$. Denote $X \in \mathbb{R}^{n\times d}$ as the collection of training input data, with each row of X representing an input $\boldsymbol{x}_i$, and $\boldsymbol{y} := (y_1, \cdots, y_n)^T$. We further write $\mathbf{f} \in \mathbb{R}^n$ as the predictions of f on X. The goal of the optimization task is to minimize the Mean Square Error (MSE)

$$\mathscr{L}(\mathbf{w}; (X, \boldsymbol{y})) = \frac{1}{n}\sum_{i=1}^{n} (f(\mathbf{w};\boldsymbol{x}_i) - y_i)^2 = \frac{1}{n}\|\mathbf{f} - \boldsymbol{y}\|^2. \tag{4.1}$$

Let $\mathbf{w}_0$ be the weight parameters at initialization. Mini-batch SGD is conducted as follows: at each step t, randomly sample a batch $\mathscr{B} \subset \mathscr{D}$ (of batch size b), and perform the update following

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \frac{\eta}{b}\frac{\partial}{\partial \mathbf{w}} \sum_{(\boldsymbol{x}_j, y_j)\in\mathscr{B}} (f(\mathbf{w}^t;\boldsymbol{x}_j) - y_j)^2,$$

where η is the learning rate. When $b = n$, mini-batch SGD reduces to the full-batch gradient descent (GD).

Neural Tangent Kernel (NTK). Proposed in [46], NTK is a useful tool in understanding and analyzing over-parameterized neural networks.

Definition 4.2.1 ((Neural) Tangent Kernel)**.** The (neural) tangent kernel $K(\mathbf{w};\cdot,\cdot)$ for a parameterized machine learning model $f(\mathbf{w};\cdot):\mathbb{R}^p\times\mathbb{R}^d\to\mathbb{R}$ is defined as:

$$\forall \boldsymbol{x},\boldsymbol{z}\in\mathbb{R}^d,\quad K(\mathbf{w};\boldsymbol{x},\boldsymbol{z})=\left\langle \frac{\partial f(\mathbf{w};\boldsymbol{x})}{\partial \mathbf{w}},\frac{\partial f(\mathbf{w};\boldsymbol{z})}{\partial \mathbf{w}}\right\rangle.$$

Given the training data inputs X, the NTK can be evaluated on any pair of inputs $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, which results in a $n\times n$ matrix K, called the NTK matrix. By definition, the NTK matrix K is symmetric and positive semi-definite. Therefore, it can be decomposed as $K=\sum_{j=1}^n \lambda_j \mathbf{u}_j {\mathbf{u}_j}^T$, with $\lambda_j\in\mathbb{R}$ and $\mathbf{u}_j\in\mathbb{R}^n$, $j\in\{1,\cdots,n\}$, being the eigenvalues and unit-length eigenvectors, respectively. Without loss of generality, we assume $\lambda_1\geq\lambda_2\geq\cdots\geq\lambda_n\geq 0$.

Top-eigenspace and decomposition of the loss. Given an integer s, $1\leq s<n$, we call the *top eigenspace* (or *top-s eigenspace*) of NTK as the subspace spanned by the top eigenvectors $\mathbf{u}_j$ with $1\leq j\leq s$. We also define projection operators $\mathscr{P}_{\leq s}:\mathbb{R}^n\to\mathbb{R}^n$ and $\mathscr{P}_{>s}:\mathbb{R}^n\to\mathbb{R}^n$, such that for any vector $\mathbf{v}\in\mathbb{R}^n$ the followings hold:

$$\mathscr{P}_{\leq s}\mathbf{v}=\sum_{i=1}^{s}\langle \mathbf{v},\mathbf{u}_i\rangle\,\mathbf{u}_i,\quad \mathscr{P}_{>s}\mathbf{v}=\sum_{i=s+1}^{n}\langle \mathbf{v},\mathbf{u}_i\rangle\,\mathbf{u}_i.$$

The MSE Eq. (4.1) can be decomposed as

$$\mathscr{L}=\frac{1}{n}\|\mathbf{f}-\boldsymbol{y}\|_2^2=\frac{1}{n}\|\mathscr{P}_{\leq s}(\mathbf{f}-\boldsymbol{y})\|_2^2+\frac{1}{n}\|\mathscr{P}_{>s}(\mathbf{f}-\boldsymbol{y})\|_2^2=:\mathscr{L}_{\leq s}+\mathscr{L}_{>s}. \tag{4.2}$$

Critical learning rate. When a constant learning rate of the algorithm is used throughout the training, it is important to select the learning rate η, as a large η easily leads to a divergence of loss and a small η slows down the training procedure. A conventional wisdom is to set η no larger than the *critical learning rate* $\eta_{\text{crit}}(\mathbf{w}):=\frac{2}{\lambda_{\max}(H_{\mathscr{L}}(\mathbf{w}))}$, where $H_{\mathscr{L}}:=\nabla^2_{\mathbf{w}}\mathscr{L}(\mathbf{w})$ denotes

the Hessian of the loss. This intuition follows from the well-known lemma in optimization:

Lemma 4.2.2 (Descent Lemma [78])**.** *For a smooth loss* $\mathscr{L}(\mathbf{w}) : \mathbb{R}^p \to \mathbb{R}$*, suppose that* $\lambda_{\max}(H_{\mathscr{L}}(\mathbf{w})) \leq \beta$ *for all* $\mathbf{w} \in \mathbb{R}^p$*, then GD satisfies:*

$$\mathscr{L}(\mathbf{w}^{t+1}) \leq \mathscr{L}(\mathbf{w}^t) - \eta \left(1 - \frac{\eta\beta}{2}\right) \left\|\nabla_{\mathbf{w}} \mathscr{L}(\mathbf{w}^t)\right\|^2.$$

For $\eta < 2/\beta$, the descent lemma guarantees the decrease of the loss. Note that this inequality is tight for quadratic loss, e.g., loss for linear models. For neural networks with sufficient width trained with a constant learning rate smaller than η_{crit}, due to *transition to linearity* [66], the critical learning rate η_{crit} almost does not change during training [64]. Furthermore, by decomposing the Hessian of the loss, it can be seen that η_{crit} can be well-approximated by NTK (exact, for linear models): $\eta_{\text{crit}} \approx n/\|K\|_2 = n/\lambda_1$, as detailed in Appendix C.1. For neural networks that are not wide, [86, 3, 108] showed the approximation still holds and we provide additional evidence for SGD trained with a large learning rate in Appendix C.1.

Note that unless specified, the critical learning rate is evaluated at initialization $\mathbf{w}_0$.

Catapult dynamics. It was recently observed in [65] that, for wide neural network, full batch GD with a learning rate that is larger than η_{crit} (e.g., $\eta \in (\eta_{\text{crit}}, 2\eta_{\text{crit}})$ as shown in [65]) surprisingly ends up with a convergence. Instead of the expected divergence, the loss decreases after a drastic increase at the beginning stage of training, forming a loss spike (see Figure 4.2). Moreover, $\|K\|_2$ is observed to be smaller at the end of the spike.

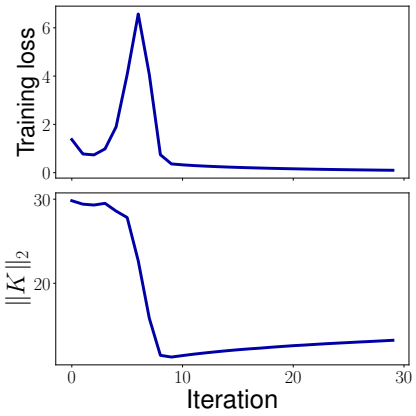


Figure 4.2. An illustration of the catapult. This experiment corresponds to Figure 4.3a.

Interestingly, the solution found by this large-learning-rate GD turns out to perform better in terms of test loss. Intuitively, the decrease in $\|K\|_2$ raises the divergence threshold $n/\|K\|_2$ which allows a final convergence.

In this paper, we refer *catapult dynamics* as the phenomenon of a drastic increase followed by a fast decrease in the training loss which is triggered by a learning rate larger than η_{crit} and accompanied by a decreasing $\|K\|_2$.

4.3 Catapults in optimization

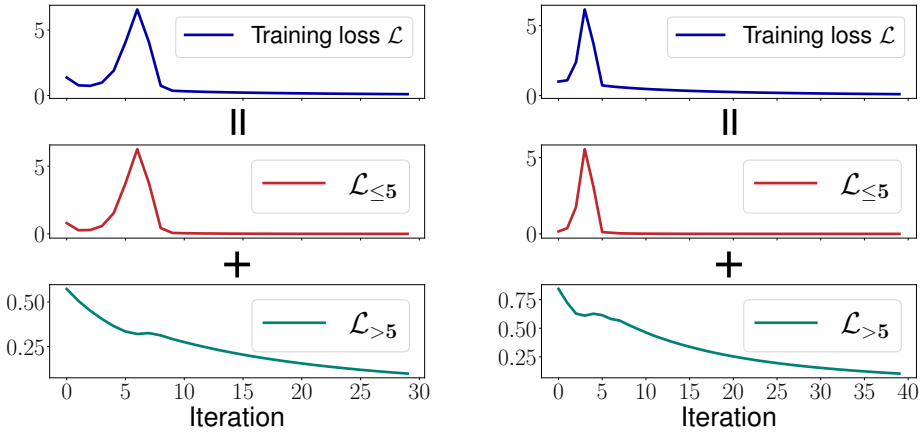
Catapults occur in the top eigenspace of the tangent kernel for GD

The training dynamics of the machine learning model, e.g., a neural network, are closely related to its NTK $K^t := K(\mathbf{w}^t; X, X) \in \mathbb{R}^{n\times n}$. Specifically, when the loss is optimized by gradient flow (continuous-time GD) with learning rate η, the output follows the dynamic equation [64]:

$$\frac{d\mathbf{f}^t}{dt} = -2\eta \frac{K^t}{n}(\mathbf{f}^t - \boldsymbol{y}).$$

By discrete time GD, this becomes

$$\mathbf{f}^{t+1} - \boldsymbol{y} = \left(I_n - 2\eta \frac{K^t}{n}\right)(\mathbf{f}^t - \boldsymbol{y}) + \Delta_{H^t_{\mathbf{f}}}, \tag{4.3}$$



(a) Loss decomposition(FCN) **(b)** Loss decomposition(CNN)

Figure 4.3. Catapult occurring in the top eigenspace of NTK in GD for 5-layer FCN (a) and CNN (b). The training loss is decomposed into the eigenspace of NTK, i.e., $\mathscr{L} = \mathscr{L}_{\leq 5} + \mathscr{L}_{>5}$. In the experiment, both networks are trained by GD on 128 data points from CIFAR-10 with learning rate 6 and 8 respectively (the critical learning rates are 3.6 for FCN and 4.5 for CNN).

with $\Delta_{H_{\mathbf{f}}^t} := \left\langle \mathbf{w}^{t+1} - \mathbf{w}^t, \nabla^2_{\mathbf{w}} \mathbf{f}(\xi)(\mathbf{w}^{t+1} - \mathbf{w}^t) \right\rangle \in \mathbb{R}^n$ and $\xi = \tau \mathbf{w}^t + (1-\tau)\mathbf{w}^{t+1}, \tau \in (0,1)$.

Note that for finitely wide neural networks, $\left\| \Delta_{H_{\mathbf{f}}^t} \right\|_2$ is small compared to the first term [86, 108] and is exactly zero for infinitely wide neural networks [64]. Therefore, the training dynamics of neural networks are mainly determined by the first term in R.H.S. of the above equation, which relies on the spectral information of the NTK K^t. This data-dependent NTK is also useful for understanding the generalization performance of neural networks [26, 10, 85, 72].

Consider decomposing Eq. (4.3) into eigendirections of the NTK K^t, i.e., $\langle \mathbf{f}^t - \mathbf{y}, \mathbf{u}_i^t \rangle$. Supposing the dynamics among eigendirections are not interacting and $\mathbf{u}_i$ is constant, we expect that the increase of training loss during catapult occurs in the top few eigendirections where $\eta > n/\lambda_i$, while the loss corresponding to the remaining eigendirections remain decreasing. Indeed, this has been theoretically shown to be true on quadratic models that approximate wide neural networks [124].

Claim 1. *The catapult occurs in the top eigenspace of the tangent kernel: the loss component corresponding to the top-s eigenspace $\mathscr{L}_{\leq s}$ has a spike during the catapult, while the loss*

component in the complementary eigenspace $\mathscr{L}_{>s}$ decreases monotonically.

Remark 4.3.1. We note that the catapult does not occur in all eigendirections, as the learning rate η cannot be arbitrarily large. Instead, there is a maximum learning rate $\eta_{\max}$ such that if $\eta > \eta_{\max}$ the algorithm will diverge. For instance, $\eta_{\max} = 2\eta_{\text{crit}}$ for quadratic models [124] and $\eta_{\max} \approx 6\eta_{\text{crit}}$ for ReLU networks [65]. Therefore, for any learning rate $\eta \in (\eta_{\text{crit}}, \eta_{\max})$ such that catapult occurs, only the top few eigendirections satisfy $n/\lambda_i < \eta_{\max}$. We consistently observe that s is a small constant no larger than 10 in all our experiments.

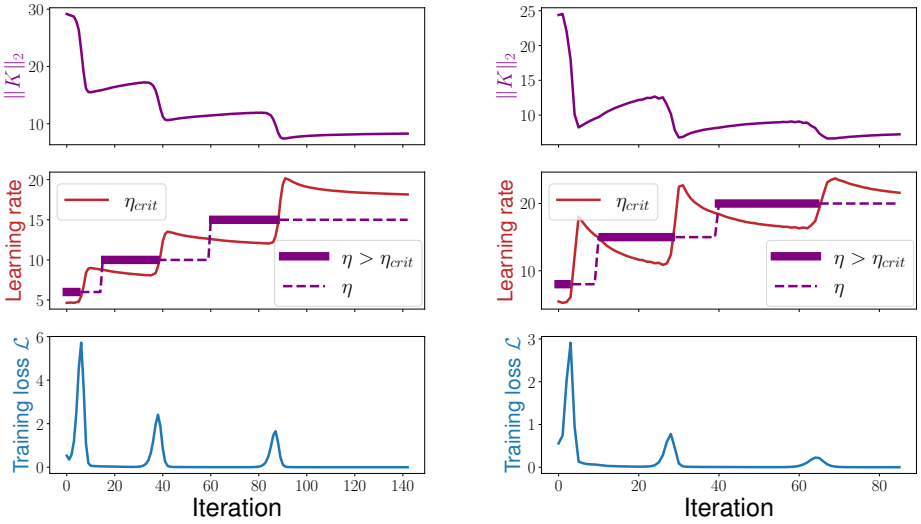
We empirically justify Claim 1 for neural networks. In particular, we consider three neural network architectures: a 5-layer Fully Connected Neural Network (FCN), a 5-layer Convolutional Neural Network (CNN), and Wide ResNets 10-10; and three datasets CIFAR-10, SVHN, and a synthetic dataset. The details of experimental setup can be found in Appendix C.6. We present a selection of the results in Figure 4.3 with the remaining results in Figure C.3 and C.4 in Appendix C.2. We can see that $\mathscr{L}_{\leq 5}$ corresponds to the spike in the training loss while $\mathscr{L}_{>5}$ decreases almost monotonically. Concurrently with this study, [122] showed that the loss spike in GD is primarily due to the low-frequency component, corroborating our findings through a frequency perspective.

We note that the same phenomenon holds for multidimensional outputs. See more details in Figure C.5 in Appendix C.2.

Inducing multiple catapults in GD

While prior work showed a single catapult during training with gradient descent [65, 124, 53], we present that catapults can be induced multiple times by repeatedly increasing the learning rate during training.

Specifically, during a catapult, the norm of NTK $\|K\|_2$ decreases, which leads to an increase in the critical learning rate $\eta_{\text{crit}} \approx n/\|K\|_2$, see Figure 4.4. When the loss starts to decrease during a catapult, η_{crit} surpasses the current learning rate η of the algorithm. Hence, after each catapult, one can reset the algorithmic learning rate η to be greater than the current



(a) 5-layer FCN **(b)** 5-layer CNN

Figure 4.4. Multiple catapults during GD with increased learning rates. We train a 5-layer FCN and CNN on a subset of CIFAR-10 using GD. The learning rate is increased two times for each experiment. The experimental details can be found in Appendix C.6.

η_{crit} to trigger another catapult. In practice, we observe that a sequence of catapults can be triggered by repeating the above procedure. See Figure 4.4 for a demonstration of various neural network architectures.

Interestingly, with multiple catapults, the gradient descent can ultimately converge with a much larger learning rate, which leads to a divergence, instead of a catapult, if set as the initial learning rate of gradient descent (see Figure C.7 in Appendix C.2). Furthermore, thanks to the relation $\eta_{\text{crit}} \approx n/\|K\|_2$, this indicates that the multiple catapults achieve a much smaller $\|K\|_2$ which can not be obtained in the scenario of a single catapult. See Figure 4.4 for an experimental demonstration. Moreover, the multiple catapults lead to better generalization performance than a single catapult. We defer this discussion of generalization performance to Section 4.4.

Catapults in SGD

In this section, we consider the stochastic setting, and argue that the spikes often observed in the training loss of SGD (e.g., Figure 4.1) are in fact *catapults*.

Mechanism of catapults in SGD. Recall that the catapults are triggered when $\eta > \eta_{\text{crit}}$. Unlike in deterministic gradient descent, the mini-batch stochastic training dynamics is determined by the NTK matrix evaluated on the given batch X_{batch}. Specifically, the update equation of mini-batch SGD becomes (c.f. Eq.(4.3) of GD):

$$\mathbf{f}_{\text{batch}}^{t+1} - \boldsymbol{y}_{\text{batch}} = \left(I_b - 2\eta \frac{K_t(X_{\text{batch}})}{b} \right) (\mathbf{f}_{\text{batch}}^{t} - \boldsymbol{y}_{\text{batch}}) + \Delta_{H_{\mathbf{f}_{\text{batch}}^{t}}}, \tag{4.4}$$

where b is the mini batch size, $\mathbf{f}_{\text{batch}} := \mathbf{f}(X_{\text{batch}})$ and $\boldsymbol{y}_{\text{batch}}$ is the label corresponds to X_{batch}. It is important to note that in mini-batch SGD the critical learning rate $\eta_{\text{crit}}(X_{\text{batch}})$ becomes batch dependent: for batches that have relatively large (small, respectively) $\|K_{\text{batch}}\|_2$, the corresponding critical learning rate $\eta_{\text{crit}}(X_{\text{batch}})$ is relatively small (large, respectively). Then, if $\eta_{\text{crit}}(X_{\text{batch}})$ of a given batch is smaller than the algorithmic learning rate η of SGD, we expect a catapult will happen: an increase in the running training loss.

Indeed, this expectation is confirmed in our experiments. Specifically, we train the network on a synthetic dataset with SGD and consider batch size one. We set the algorithmic

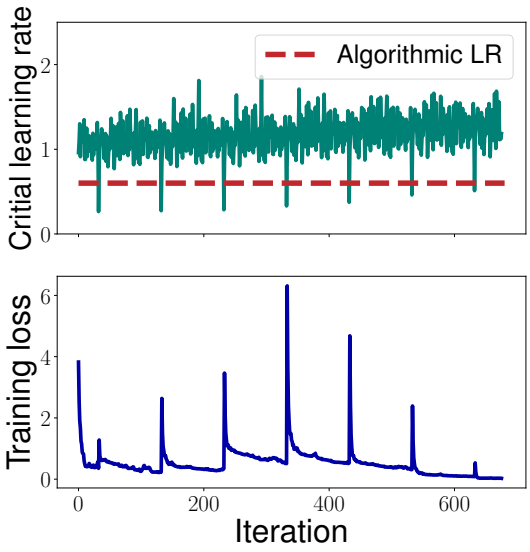
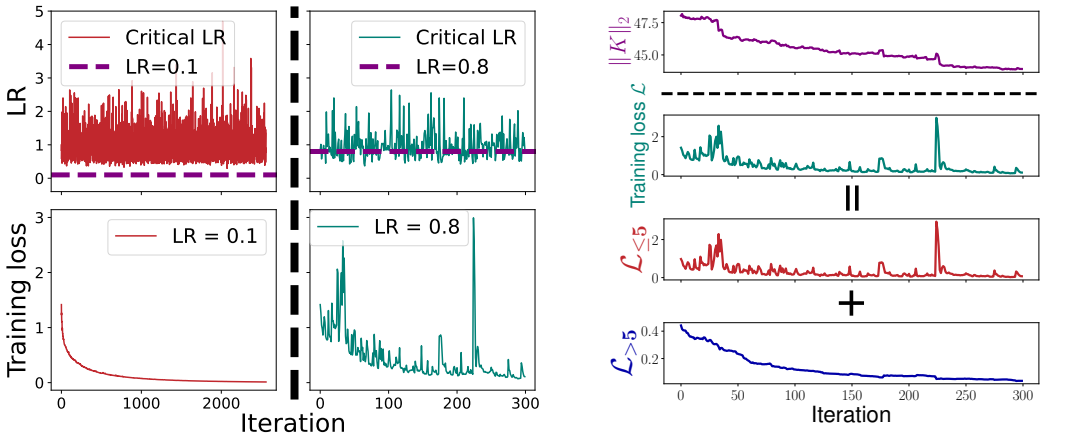


Figure 4.5. Exact match between the occasion when $\eta > \eta_{\text{crit}}(X_{\text{batch}})$ and loss spike for SGD. We train a two-layer neural network on a synthetic dataset using SGD with batch size one.

learning rate higher than the critical learning rate for only one training example. As expected, we observe that the loss spike only occurs when the gradient is computed based on that particular training example. See the results in Figure 4.5 and the detailed experimental setup in Appendix C.6.

In more practical scenarios, we train a shallow network by SGD with mini-batch size 32, on a subset of CIFAR-10 with training size 128. First, when the algorithmic learning rate η is smaller than $\eta_{\text{crit}}(X_{\text{batch}})$ of all the batches (as shown in the case of $\eta = 0.1$ in (Figure 4.6 upper left)), we observe that the training loss of mini-batch SGD monotonically decreases until convergence without any spike; when η becomes greater than $\eta_{\text{crit}}(X_{\text{batch}})$ for some of the batches (as shown in the case of $\eta = 0.8$ in Figure 4.6 upper right), many spikes appear in the training loss. Moreover, we show that these spikes in the (total) training loss are caused by large learning rates for batches. Specifically, for the case of $\eta = 0.8$, we verify that whenever the (total) training loss increases, the algorithmic learning rate η is larger than the critical learning



(a) Critial learning rates and training loss **(b)** Loss decomposition with $\eta = 0.8$

Figure 4.6. (a): Critical learning rates of batches and the training loss of SGD with learning rate 0.1 (left two subfigures) and 0.8 (right two subfigures). (b): Loss decomposition with $\eta = 0.8$. LR is an abbreviation for learning rate. We train a two-layer neural network on 128 data points of CIFAR-10 using SGD with batch size 32. We further decompose the loss into the top-5 eigendirections of the NTK $\mathscr{L}_{\leq 5}$ and the remaining eigendirections $\mathscr{L}_{>5}$ corresponding to $\eta = 0.8$.

rate $\eta_{\text{crit}}(X_{\text{batch}})$ for the current batch X_{batch}. This phenomenon is further verified for 5-layer FCN and CNN. See Table 4.2.

Decreases in the spectral norm of the tangent kernel correspond to spikes. As shown in prior work [65] and in the multiple catapults in Section 4.3 an important characterization of the catapult dynamics is the decreasing NTK norm $\|K\|_2$. Here, we experimentally show that the spectral norm of the NTK decreases whenever there is a spike in the SGD training loss.

Specifically, we consider four network architectures: (1) 5-layer FCN, (2) 5-layer CNN (the same as the ones in Figure 4.3), (3) Wide ResNets 10-10 and (4) ViT-4. We train neural networks on a subset of CIFAR-10 using SGD. Figure 4.7 shows some of the results (more results on various datasets and parameterizations are available in Appendix C.3). One can easily see that at each spike of the training loss, there is a significant drop in the spectral norm of NTK $\|K\|_2$, while $\|K\|_2$ are mostly increasing or staying unchanged at other steps. This empirical evidence corroborates that these spikes are indeed (mini-)catapults, instead of some random fluctuations in the training loss. All experimental details can be found in Appendix C.6.

Catapults occur in the top eigenspace of the tangent kernel for SGD. As discussed in Section 4.3, another characteristic of the catapults is that they occur in the top eigenspace of the tangent kernel. We show that these loss spikes in SGD also occur in the top eigenspace, as

Table 4.2. The match rate between $\Delta\mathscr{L}^t := \mathscr{L}^{t+1} - \mathscr{L}^t > 0$ and $\eta > \eta^t_{\text{crit}}(X_{\text{batch}})$. For each network architecture, we calculate the match rate as the ratio of occurrences where $\eta > \eta^t_{\text{crit}}(X_{\text{batch}})$ for all t such that $\mathscr{L}^{t+1} > \mathscr{L}^t$ until convergence of SGD (see the training loss in Figure 4.6(c) for shallow net and Figure 4.7 (a,b) for deep nets). Each result is the average of 3 independent runs.

Network Architecture	Match rate between $\Delta\mathscr{L} > 0$ and $\eta > \eta_{\text{crit}}(X_{\text{batch}})(\%)$
Shallow network	97.32 ± 0.45
5-layer FCN	96.17 ± 1.46
5-layer CNN	94.67 ± 3.27

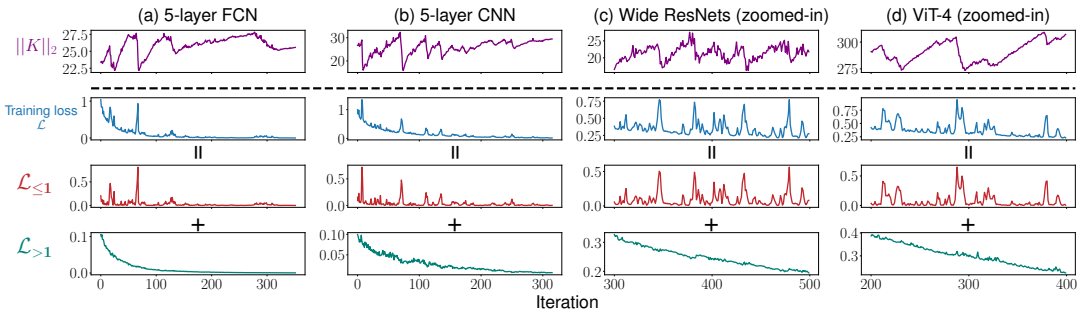


Figure 4.7. Catapult dynamics in SGD for modern deep architectures. The training loss is decomposed based on the eigendirections of the NTK: $\mathscr{L}_{\leq 1}$ and $\mathscr{L}_{>1}$. We train the networks on a subset of CIFAR-10 using SGD. The complete versions of Panel (c) and (d) can be found in Figure C.8 in Appendix C.3.

another evidence that these spikes are catapults.

In the experiments, we decompose the training loss of SGD into $\mathscr{L}_{\leq 1}$ and $\mathscr{L}_{>1}$ based on the eigendirections of the tangent kernel. We observe that $\mathscr{L}_{\leq 1}$ corresponds to the spikes in the training loss, while the decrease of $\mathscr{L}_{>1}$ is nearly monotonic, with only small oscillations present. See Figure 4.6b for the shallow network with $\eta = 0.8$ and Figure 4.7 for deep networks. Note that for deep neural networks, compared to the catapults in GD where they occur in the top-5 eigendirections of the NTK (Figure 4.3), we consistently observe that for SGD, catapults occur only in the top-1 eigendirection. Additional empirical validation can be found in Appendix C.3.

This observation, along with the results that the spectral norm of the NTK decreases corresponding to the loss spike, is consistent with our findings for GD and provides evidence that the spikes in training loss for neural networks are caused by catapults.

Remark 4.3.2 (Top eigenspace accounts for the sharp loss spikes in SGD). In SGD training loss, the sharp spikes we observe last only a few iterations before rapidly returning to their pre-spike levels. These spikes can be attributed to catapults occurring in the top-1 eigendirection of the tangent kernel. Consider the loss change in each eigendirection of the tangent kernel. We expect that the rate of loss change in each eigendirection depends on the corresponding eigenvalue's size. Therefore, with a constant learning rate, changes happen faster in the top eigendirections, which accounts for the sharp loss spikes in SGD as they occur in the top-1 eigendirection.

Remark 4.3.3 (Catapults in SGD with cyclical learning rate schedule). Training neural networks with the learning rate cyclically varying between selected boundary values was widely shown to improve the generalization performance of neural networks with less tuning [45, 98]. We empirically show that the increasing phase of the cyclical learning rate schedule induces catapults in SGD. Specifically, we observe that there is a spike in the training loss when the learning rate is increased. We demonstrate that the loss spikes are caused by catapults, by providing similar evidence to the case of SGD with a constant learning rate. See the results in Figure C.12 in Appendix C.3.

4.4 Catapults lead to better generalization through feature learning

Previous empirical results from [65, 124] show that a single catapult can lead to better test performance in GD for wide neural networks. In this section, we observe a similar trend in our experiments for both GD and SGD with multiple catapults. We posit an explanation for this phenomenon by demonstrating that catapults improve feature learning by increasing alignment between the Average Gradient Outer Products (AGOP) of the trained network and the true model, therefore improving generalization. We formalize this claim as follows. Let $\{(x_i, f^*(x_i))\}_{i=1}^n$ denote training data with $f^*(x)$ denoting the true model. Then, for any predictor f, the AGOP, $G(f, \{x_1, \ldots, x_n\})$ is given as follows:

$$G(f, \{x_1, \ldots, x_n\}) = \frac{1}{n} \sum_{i=1}^{n} \nabla_x f(x_i) \nabla_x f(x_i)^T \; ; \tag{4.5}$$

where $\nabla_x f$ denotes the gradient of f with respect to the input x.[2] We will suppress the dependence on the data $\{x_i\}_{i=1}^n$ to simplify notation. Assuming the data x_i are i.i.d. samples from an underlying data distribution, in the limit as $n \to \infty$, Eq. equation 4.5 converges to a quantity referred to as the Expected Gradient Outer Product (EGOP). Letting G^* denote the EGOP of f^*

[2]For predictors with multivariate outputs, we consider the Jacobian instead of the gradient.

and G denote the AGOP of f, we define *AGOP alignment* using the cosine similarity between G, G^* as follows:

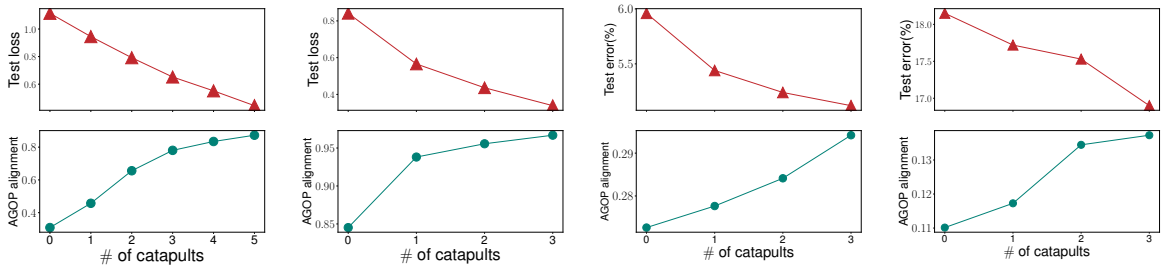
$$\textbf{AGOP alignment}: \cos(G, G^*) := \frac{\mathrm{Tr}\left(G^T G^*\right)}{\|G\|_F \|G^*\|_F}. \tag{4.6}$$

Remark 4.4.1. G^* captures the directions along which f^* varies the most and those along which it varies least. When training a predictor on data generated using low rank G^*, it is possible to improve sample efficiency by first estimating G^*. Indeed, this has been theoretically shown in the case of multi-index models, i.e., functions of the form $f^*(\boldsymbol{x}) = g(U\boldsymbol{x})$ where the index space U is a low-rank matrix [35, 105, 118]. Additionally, a recent line of work connected AGOP with feature learning in neural networks and further demonstrated that training predictors on data transformed by AGOP can lead to substantial increases in test performance [91, 13, 92]. Thus, we believe that AGOP alignment is a key measure for generalization, and we next corroborate our claim empirically across a broad class of network architectures and prediction tasks.

Experimental settings. We work with a total of seven datasets: three synthetic datasets and four real-world datasets. For synthetic datasets, we consider true functions $f^*(\boldsymbol{x}) = (1)\ x_1 x_2$ (rank-2), $(2)\ x_1 x_2 (\sum_{i=1}^{10} x_i)$ (rank-3) and $(3)\ \sum_{j=1}^{4} \prod_{i=1}^{j} x_i$ (rank-4) [1]. For the four real-world datasets, we use (1) CelebA [69], (2) SVHN dataset [79], (3) Fashion-MNIST [113] and (4) USPS dataset [44]. When the underlying model is not available, we use a state-of-the-art model as a substitute. We present the results for a selection of the datasets in this section and put the results for the remaining datasets in Appendix C.5. The experimental details can be found in Appendix C.6.

Improved test performance by catapults in GD. In Section 4.3, we showed that catapults can be generated multiple times. We now show that generating multiple catapults leads to improved test performance of neural networks trained with GD by leading to increased AGOP alignment. In Figure 4.8, we can see for all tasks, the test loss/error decreases as the number of catapults increases while AGOP alignment increases. This indicates that learning the EGOP

strongly correlates with test performance.



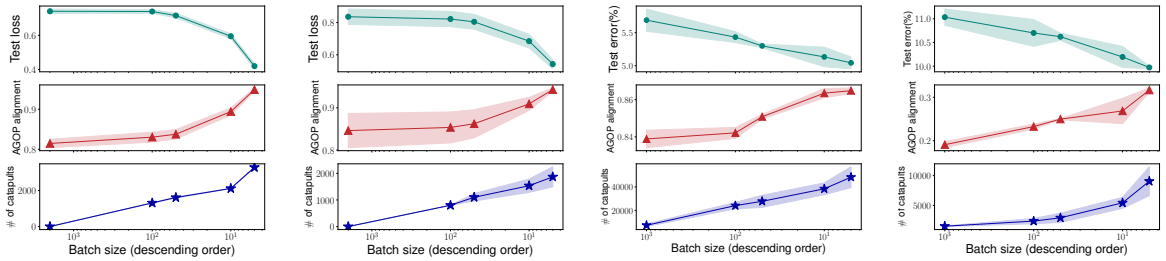
(a) Rank-2 regression **(b)** Rank-3 regression **(c)** SVHN **(d)** CelebA

Figure 4.8. Correlation between AGOP alignment and test performance in GD with multiple catapults. The learning rate is increased multiple times during training to generate multiple catapults. We train 2-layer FCN in Panel(a), 4-layer FCN in Panel(b,d) and 5-layer CNN in Panel(c). Experimental details can be found in Appendix C.6.

Remark 4.4.2. As discussed earlier, AGOP alignment is a means of improving sample efficiency when training on data from multi-index models with low-rank index space. Our results on synthetic datasets show that catapults increase AGOP alignment, thereby leading to improved test performance. Additionally, we show that when the index space is full-rank, which can be effectively learned by neural networks in the NTK regime, catapults do not improve the test performance as well as the AGOP alignment. See Figure C.17 in Appendix C.4.

Improved test performance by catapults in SGD. In Section 4.3, we have demonstrated the occurrence of catapults in SGD. We now show that decreasing batch size in SGD leads to better test performance as a result of an increase in the number of catapults and thus, increased AGOP alignment. We estimate the number of catapults during training by counting the number of the occurrence of the event $\eta - \eta_{\text{crit}}(X_{\text{batch}}) > \varepsilon$ with $\varepsilon = 10^{-8}$ until the best validation loss/error.

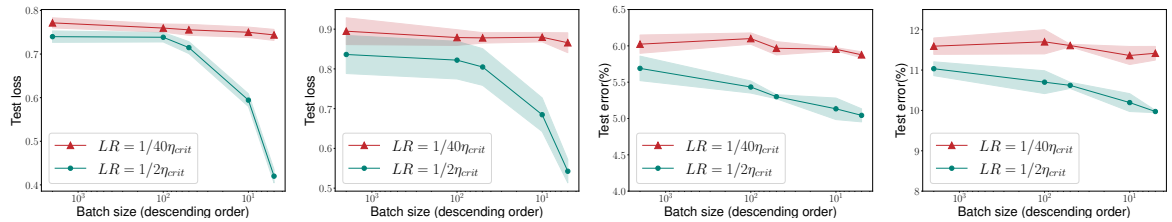
In Figure 4.9, we can see that across all tasks, as the batch size decreases, (1) the number of catapults increases, (2) the test loss/error decreases and (3) the AGOP alignment improves. These findings indicate that in SGD, a smaller batch size leads to more catapults which in turn improves the test performance through alignment with the AGOP. These observations are consistent with our findings in GD.



(a) Rank-2 regression **(b)** Rank-3 regression **(c)** SVHN **(d)** CelebA

Figure 4.9. Correlation between AGOP alignment and test performance in SGD. We train a 2-layer FCN in Panel(a), 4-layer FCN in Panel(b,d) and 5-layer CNN in Panel(c) by SGD with a constant learning rate. We report the results as the average of 3 independent runs. Experimental details can be found in Appendix C.6.

Batch size does not affect generalization when the learning rate is small. Given the discussion above, sufficiently small learning rates will result in no catapults for any batch size. Thus we expect that all batch sizes will provide similar generalization performance for sufficiently small learning rates. This, indeed, is what we observe in the experiments presented in Figure 4.10 where we keep the same experimental setting as for Figure 4.9 except for a smaller learning rate. Specifically, we observe that while decreasing batch size consistently improves generalization for large learning rates, it has little effect on generalization for small learning rates.

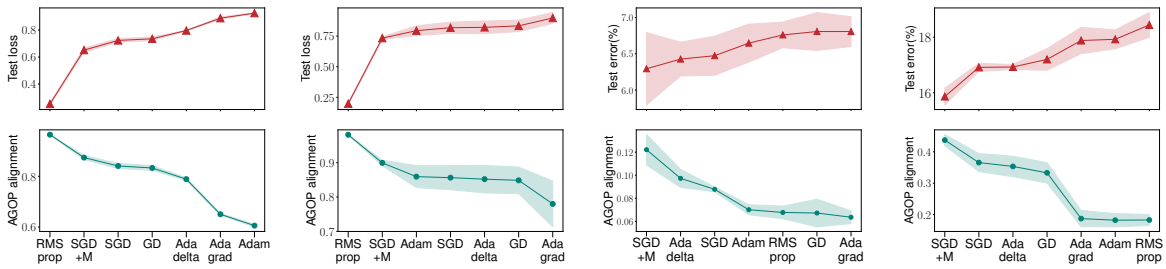


(a) Rank-2 regression **(b)** Rank-3 regression **(c)** SVHN **(d)** CelebA

Figure 4.10. "Large" vs. "small" learning rate on test performance with different batch sizes. We consider the same setting as in Figure 4.9 except for selecting a smaller learning rate $\eta_{\text{crit}}/40$ compared to $\eta_{\text{crit}}/2$ in Figure 4.9. Here η_{crit} is the critical learning rate for the whole dataset.

Generalization with different optimizers correlates with AGOP alignment. We further demonstrate the strong correlation between the test performance and AGOP alignment

by comparing the predictors trained on the same task with a number of different optimization algorithms. From the results shown in Figure 4.11, we can see that the AGOP alignment strongly correlates with the test performance, which suggests that models learning the AGOP is useful for learning the problem.



(a) Rank-2 regression **(b)** Rank-3 regression **(c)** SVHN **(d)** CelebA

Figure 4.11. Correlation between test performance and AGOP alignment for different optimization algorithms. We train a 2-layer FCN in Panel(a), a 4-layer FCN in Panel(b,d) and a 5-layer CNN in Panel(c). We use GD, SGD, SGD with Momentum [90](SGD+M), RMSprop [37], Adagrad [23], Adadelta [120] and Adam [57] for training. Experimental details can be found in Appendix C.6.

4.5 Conclusions

In this work, we addressed the following questions: (1) why do spikes in training loss occur during training with SGD and (2) how do the spikes relate to generalization? For the first question, we demonstrate that the spikes in the training loss are caused by the catapult dynamics in the top eigenspace of the tangent kernel. For the second question, we show that catapults lead to increased alignment between the AGOP of the model being trained and the AGOP of the underlying model or its state-of-the-art approximation. A consequence of our results is the explanation for the observation that SGD with small batch size often leads to improved generalization. This is due to an increase in the number of catapults for small batch sizes, presumably due to increased batch variability, and thus to improved AGOP alignment.

4.6 Acknowledgements

Chapter 4, in full, is a reprint of Libin Zhu, Chaoyue Liu, Adityanarayanan Radhakrishnan, and Mikhail Belkin. “Catapults in sgd: spikes in the training loss and their impact on generalization through feature learning.” ICML 2024. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Summary and discussion

Summary

This thesis investigates the complex dynamics of over-parameterized neural networks, structured into two parts. The first part focuses on demonstrating how wide neural networks exhibit a transition to linearity as their "width" increases, converging to the first-order Taylor expansion of parameters. This transition to linearity is important for the success of optimization for various architectures of feedforward neural networks corresponding to arbitrary directed acyclic graphs.

The second part shifts to an investigation of modern training regimes, specifically considering optimization using large learning rates. We focus on the catapult phase phenomenon, which happens when the neural networks are trained using gradient descent with a large learning rate. We show that quadratic models can be a useful tool to understand the catapult dynamics, as they similarly exhibit catapult dynamics when the learning rate is large, and also result in better generalization.

Additionally, we extend the study of catapult dynamics to stochastic gradient descent (SGD). We provide evidence that spikes in training loss observed during SGD are caused by catapult dynamics. We further show that catapults enhance feature learning and thereby improve generalization. We demonstrate that smaller batch sizes in SGD lead to a greater number of catapults, thus improving feature learning and test performance.

Overall, the thesis integrates theoretical analysis with empirical evidence to offer novel

insights into the complex dynamics that govern the training and generalization of neural networks.

Future directions

Understanding the feature learning caused by catapult dynamics. An interesting avenue is to build a theory of how catapult dynamics promote feature learning. In Chapter 4, we have empirically shown that catapult dynamics improve feature learning by enhancing the AGOP alignment. However, a comprehensive theoretical understanding remains to be established. As shown in Chapter 3, quadratic models, serving as a simpler proxy for neural networks, also exhibit catapult dynamics; therefore, they can be used to understand the feature learning induced by these dynamics. Additionally, in Chapter 4 we showed that catapult dynamics mainly occur in a subspace of low dimension, specifically the top eigenspace of the tangent kernel. This observation suggests that analyses confined to this subspace, which simplifies the complexity of the analysis, are sufficient for the task.

Connecting the implicit bias to the feature learning. The implicit bias introduced by optimization algorithms plays an important role in learning over-parameterized neural networks [33, 32, 101, 103]. Despite their impact, the relationship between these biases and the networks' generalization performance is not well understood. Feature learning is known to explicitly affect generalization [91], suggesting that it could serve as a crucial link between implicit biases and generalization. Therefore, it is an important direction to investigate how these implicit biases lead to improved feature learning for neural networks. Furthermore, understanding the nuances of this relationship can potentially guide the development of new more effective algorithms.

Promoting the generalization of neural networks by feature learning. Since catapults can effectively improve the generalization performance through AGOP alignment, it would be interesting to incorporate these insights into practical neural network training. A promising avenue is to construct regularization techniques based on the AGOP to promote feature learning of neural networks. For example, the constraint on the norm of the AGOP, which is close to the

norm of the weights by the neural ansatz in [91] can promote the smoothness of the predictor hence improving the generalization.

Another interesting application is to investigate how adaptive learning rate strategies interact with the catapult phase phenomenon and if they can further optimize the training process. As the catapult phase phenomenon occurs when the learning rate is larger than a critical learning rate (typically proportional to the inverse of the spectral norm of the tangent kernel), it would be promising to develop an algorithm that is aware of the critical learning rate and adapts the learning rate to be larger than the critical one. Therefore, a greater number of catapults will occur during training and possibly it leads to a better generalization.

Appendix A

Chapter 2 Supplementary

Notations for set of neurons. We extra define the following notations for the proof. For $0 \leq \ell \leq L-1$, $i \in [d_\ell]$, we use $\mathscr{F}_{\mathscr{I}_i^{(\ell)}}$ to denote the set of all the elements in the vector $f_{\mathscr{I}_i^{(\ell)}}$ (Eq. (2.5)):

$$\mathscr{F}_{\mathscr{I}_i^{(\ell)}} := \{f : f \in f_{\mathscr{I}_i^{(\ell)}}\}. \tag{A.1}$$

And we use $\mathscr{P}^{(\ell)}$ to denote the set of all neurons in ℓ'-th layer i.e., $f^{(\ell')}$ defined in Eq. (2.4), with $0 \leq \ell' \leq \ell$:

$$\mathscr{P}^{(\ell)} := \{f : f \in f^{(\ell')}, \ell' \leq \ell\}. \tag{A.2}$$

Activation functions. In Assumption 2.3.2, we assume the Lipschitz continuity and smoothness for all the activation functions. In the proof of lemmas, e.g., Lemma A.2.1 and A.2.2, we only use the fact that they are Lipschitz continuous and smooth, as well as bounded by a constant $\gamma_0 > 0$ at point 0, hence we use $\sigma(\cdot)$ to denote all the activation functions like what we do in Assumption 2.3.2 for simplicity.

Notations for derivatives. Additionally, in the following we introduce notations of the derivatives, mainly used in the proof of Lemma A.2.1 and Lemma A.2.2.

By definition of feedforward neural networks in Section 2.2, different from the standard

neural networks such as FCNs and CNNs in which the connection between neurons are generally only in adjacent layers, the neurons in feedforward neural networks can be arbitrarily connected as long as there is no loop.

To that end, we define $\partial f_{\mathscr{S}_i^{(\ell)}}/\partial f^{(\ell')}$ to be a mask matrix for any $\ell' < \ell$, $i \in [d_\ell]$ to indicate whether the neurons $f_{\mathscr{S}_i^{(\ell)}}$ appear in $f^{(\ell')}$:

$$\left(\frac{\partial f_{\mathscr{S}_i^{(\ell)}}}{\partial f^{(\ell')}}\right)_{j,k} = \mathbb{I}\left\{\left(f_{\mathscr{S}_i^{(\ell)}}\right)_k \in f_j^{(\ell')}\right\}. \tag{A.3}$$

And $\partial f_i^{(\ell)}/\partial f_{\mathscr{S}_i^{(\ell)}}$ and $\partial f_i^{(\ell)}/\partial \mathbf{w}_i^{(\ell)}$ are standard derivatives according to Eq. (2.5):

$$\frac{\partial f_i^{(\ell)}}{\partial f_{\mathscr{S}_i^{(\ell)}}} = \frac{1}{\sqrt{m_i^{(\ell)}}}\left(\mathbf{w}_i^{(\ell)}\right)^T (\sigma_i^{(\ell)})'(\tilde{f}_i^{(\ell)}),$$
$$\frac{\partial f_i^{(\ell)}}{\partial \mathbf{w}_i^{(\ell)}} = \frac{1}{\sqrt{m_i^{(\ell)}}}\left(f_{\mathscr{S}_i^{(\ell)}}\right)^T (\sigma_i^{(\ell)})'(\tilde{f}_i^{(\ell)}).$$

We give a table of notations that will be frequently used (See Table A.1). The same notations will be used for ResNets and CNNs with extra subscripts res and cnn respectively.

Table A.1. Table of notations

Symbol	Meaning
$f^{(\ell)}$	Vector of neurons in ℓ-th layer
d_ℓ	Number of neurons in ℓ-th layer, i.e., length of $f^{(\ell)}$
$f_{\mathscr{S}_i^{(\ell)}}$	Vector of in-coming neurons of $f_i^{(\ell)}$
$\mathbf{w}_i^{(\ell)}$	Weight vector corresponding to in-coming edges of $f_i^{(\ell)}$
$m_i^{(\ell)}$	Number of in-coming neurons of $f_i^{(\ell)}$, i.e., length of $f_{\mathscr{S}_i^{(\ell)}}$ and $\mathbf{w}_i^{(\ell)}$
$\sigma_i^{(\ell)}$	Activation function on $\tilde{f}_i^{(\ell)}$
$\mathbf{w}^{(\ell)}$	Weight vector corresponding to all incoming edges toward neurons at layer ℓ
$\mathscr{F}_{\mathscr{S}_i^{(\ell)}}$	Set of all the elements in the vector $f_{\mathscr{S}_i^{(\ell)}}$ (Eq. (A.1))
$\mathscr{P}^{(\ell)}$	Set of all neurons in $f^{(\ell')}$ with $0 \le \ell' \le \ell$ (Eq. (A.2))
$\mathsf{id}_{\ell_2,j}^{\ell_1,i}$	Index of $f_j^{(\ell_2)}$ in the vector $f_{\mathscr{S}_i^{(\ell_1)}}$

A.1 Examples of feedforward neural networks

Here we show that many common neural networks are special examples of the feedforward neural networks in Definition 2.2.2.

Fully-connected neural networks

Given an input $\boldsymbol{x} \in \mathbb{R}^d$, an L-layer fully-connected neural network is defined as follows:

$$\begin{aligned} &f^{(0)} = \boldsymbol{x}, \\ &f^{(\ell)} = \sigma\left(\frac{1}{\sqrt{m_{\ell-1}}} W^{(\ell)} f^{(\ell-1)}\right), \ \ \forall \ell \in [L-1], \\ &f(\mathbf{W};\boldsymbol{x}) := f^{(L)} = \frac{1}{\sqrt{m_{L-1}}} W^{(L)} f^{(L-1)}, \end{aligned} \tag{A.4}$$

where each $f^{(\ell)}$ is a m_ℓ-dimensional vector-valued function, and $\mathbf{W} := \left(W^{(1)}, ..., W^{(\ell)}\right)$, $W^{(\ell)} \in \mathbb{R}^{m_{\ell+1} \times m_\ell}$, is the collection of all the weight matrices. Here $\sigma(\cdot)$ is an element-wise activation function, e.g., sigmoid function.

For FCNs, the inputs are the 0-th layer neurons $f^{(0)} = \boldsymbol{x}$ and the outputs are the ℓ-th layer neurons $f^{(\ell)}$, which have zero in-degrees and zero out-degrees, respectively. For each non-input neuron, its in-degree is the number of neurons in its previous layer, $m_{\ell-1}$; the summation in Eq. (2.2) turns out to be over all the neurons in the previous layer, which is manifested in the matrix multiplication of $W^{(\ell)} f^{(\ell-1)}$. For this network, the activation functions are the same, except the ones on input and output neurons, where identity functions are used in the definition above.

DenseNets

Given an input $\boldsymbol{x} \in \mathbb{R}^d$, an L-layer DenseNet [42] is defined as follows:

$$
\begin{aligned}
& f^{(0)} = f^{(0)}_{\text{temp}} = \boldsymbol{x}, \\
& f^{(\ell)} = \sigma\left(\frac{1}{\sqrt{\sum_{l'=0}^{\ell-1} m_{\ell'}}} W^{(\ell)} f^{(\ell-1)}_{\text{temp}}\right), \qquad \text{(A.5)} \\
& f^{(\ell)}_{\text{temp}} = \left[\left(f^{(\ell-1)}_{\text{temp}}\right)^T, \left(f^{(\ell)}\right)^T\right]^T, \ \ \forall \ell \in [L-1], \\
& f(\mathbf{W};\boldsymbol{x}) := f^{(L)} = \frac{1}{\sqrt{\sum_{\ell'=0}^{L-1} m_{\ell'}}} W^{(L)} f^{(L-1)}_{\text{temp}}, \qquad \text{(A.6)}
\end{aligned}
$$

where $\mathbf{W} = \left(W^{(1)}, ..., W^{(L)}\right)$ is the collection of all the weight matrices. Here $\sigma(\cdot)$ is an element-wise activation function and for each $\ell \in [L]$, $W^{(\ell)} \in \mathbb{R}^{m_\ell \times \sum_{\ell'=0}^{\ell-1} m_{\ell'}}$.

The DenseNet shares much similarity with the fully-connected neural network, except that each non-input neuron depends on all the neurons in previous layers. This difference makes the in-degree of the neuron be $\sum_{\ell'=0}^{\ell-1} m_{\ell'}$.

Neural networks with randomly dropped edges

Given a network f built from a DAG, for any neuron f_v , where $v \in \mathscr{V} \setminus \mathscr{V}_{\text{input}}$, according to Eq. (A.4), it is defined by

$$
f_v = \sigma_v(\tilde{f}_v), \quad \tilde{f}_v = \frac{1}{\sqrt{\mathsf{in}(v)}} \sum_{u \in \mathscr{S}_{\text{in}}(v)} w_{(u,v)} f_u.
$$

If each edge (u,v) is randomly dropped with parameter $p \in (0,1)$, then the above equation becomes

$$
f_v = \sigma_v(\tilde{f}_v), \quad \tilde{f}_v = \frac{1}{\sqrt{\mathsf{in}(v)}} \sum_{u \in \mathscr{S}_{\text{in}}(v)} w_{(u,v)} f_u \cdot \mathbb{I}_{\{\xi_{u,v} \geq p\}},
$$

where $\xi_{u,v}$ is i.i.d. drawn from $\mathsf{Bernoulli}(p)$.

To interpret such an architecture, we can simply remove the edges (u,v) in the DAG where $\xi_{u,v} < p$. Then it is not hard to see that the new DAG network corresponds to the network with randomly dropped edges.

Similarly, for a neural network with randomly dropped edges in multiple layers, we can remove all the edges whose corresponding ξ is less than p. Then the resulting DAG can describe this network architecture.

We note the similarity of this network with the popularly used dropout layer [102], both of which have a mechanism of randomly dropping out neurons/edges. However, the major difference is that, neural networks with dropout layers dynamically remove (or put mask on) neurons/edges during training, while the networks we considered only here drop edges and are fixed during training.

A.2 Proof of Theorem 2.3.6

We will first compute the Hessian matrix of the network function then show how to bound the spectral norm of it.

We denote for each $\ell \in [L]$,

$$\underline{m}_\ell := \inf_{i\in[d_\ell]} m_i^{(\ell)}, \quad \overline{m}_\ell := \sup_{i\in[d_\ell]} m_i^{(\ell)}. \tag{A.7}$$

By Assumption 2.3.5, it is not hard to infer that $\overline{m}_\ell$ and $\underline{m}_\ell$ are also polynomial in m.

Fixing $k \in [d_\ell]$, to bound $\|H_{f_k}\|$, we will first bound the spectral norm of the each Hessian block $H_{f_k}^{(\ell_1,\ell_2)}$, which takes the form

$$H_{f_k}^{(\ell_1,\ell_2)} := \frac{\partial^2 f_k}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}}, \ \ k \in [d_\ell], \ \ \ell_1,\ell_2 \in [L].$$

Without lose of generality, we assume $1 \le \ell_1 \le \ell_2 \le L$ and we start with the simple case when $\ell_2 = L$.

If $\ell_1 = \ell_2 = L$, $H_{f_k}^{(L,L)}$ is simply a zero matrix since $f_k(\mathbf{w})$ is linear in $\mathbf{w}^{(\ell)}$.

If $1 \le \ell_1 < \ell_2 = L$, we will use the following Lemma:

Lemma A.2.1. *Given $\ell' \ge 1$, for any $\ell'+1 \le \ell \le L$, $\mathbf{w} \in \mathsf{B}(\mathbf{w}_0, R)$, and $j \in [d_\ell]$, we have, with probability at least $1 - \exp\Big(-C^f_{\ell,\ell'} \log^2 m\Big)$,*

$$\left\| \frac{\partial f_{S_j^{(\ell)}}}{\partial \mathbf{w}^{(\ell')}} \right\| = O\left(\max_{\ell'+1 \le p \le \ell} \frac{\sqrt{m_j^{(\ell)}}}{\sqrt{\underline{m}_p}} (\log m + R)^{\ell'} \right) = \tilde{O}\left(\max_{\ell'+1 \le p \le \ell} \frac{\sqrt{m_j^{(\ell)}}}{\sqrt{\underline{m}_p}} R^{\ell'} \right), \tag{A.8}$$

$$\left\| \frac{\partial f_{S_j^{(\ell)}}}{\partial \mathbf{w}^{(\ell')}} \right\|_F = O\left(\sqrt{m_j^{(\ell)}} (\log m + R)^{\ell-1} \right) = \tilde{O}\left(\sqrt{m_j^{(\ell)}} R^{\ell-1} \right), \tag{A.9}$$

where $C^f_{\ell,\ell'} > 0$ is a constant.

See the proof in Appendix A.8.

By Lemma A.2.1, with probability at least $1 - \exp\big(-\Omega(\log^2 m)\big)$,

$$\left\| H_{f_k}^{(\ell_1,L)} \right\| = \left\| \frac{1}{\sqrt{m_k^{(\ell)}}} \frac{\partial f_{\mathscr{S}_k^{(\ell)}}}{\partial \mathbf{w}^{(\ell_1)}} \right\| = O\left(\max_{\ell_1+1 \le \ell \le L} \frac{1}{\sqrt{\underline{m}_\ell}} (\log m + R)^{\ell_1} \right) = \tilde{O}(R^{\ell_1}/\sqrt{m}).$$

For the rest of blocks that $1 \le \ell_1 \le \ell_2 \le L-1$, we will use the following lemma to bound their spectral norms:

Lemma A.2.2. *Given $1 \le \ell_1 \le \ell_2 \le L-1$, for any $\ell_2 < \ell \le L$, $\mathbf{w} \in \mathsf{B}(\mathbf{w}_0, R)$, and $j \in [d_\ell]$, we have, with probability at least $1 - \exp\big(-\Omega(\log^2 m)\big)$,*

$$\left\| \frac{\partial^2 \tilde{f}_j^{(\ell)}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \right\| = O\left(\max_{\ell_1+1 \le p \le \ell} \frac{1}{\sqrt{\underline{m}_p}} (\log m + R)^{\ell^2} \right) = \tilde{O}\left(\max_{\ell_1+1 \le p \le \ell} \frac{R^{\ell^2}}{\sqrt{\underline{m}_p}} \right). \tag{A.10}$$

See the proof in Appendix A.9.

Remark A.2.3. Note that the above results hold for any $\ell \le L$. When $\ell = L$, $\tilde{f}_j^{(\ell)} = f_j$ which is what we need to show the transition to linearity of f_j. When $\ell < L$, as discussed before, we can

regard $\tilde{f}_j^{(\ell)}$ as a function of its parameters. We note that $\tilde{f}_j^{(\ell)}$ with $\ell < L$ will also transition to linearity by applying the same analysis for f_j, which is the result of Theorem 2.3.8.

By letting $\ell = L$ in Lemma A.2.2, for any $1 \le \ell_1 \le \ell_2 \le L-1$, with probability at least $1 - \exp\big(-\Omega(\log^2 m)\big)$,

$$\left\| H_{f_k}^{(\ell_1,\ell_2)} \right\| = O\left(\max_{\ell_1 \le \ell \le L} \frac{1}{\sqrt{\underline{m}_\ell}} (\log m + R)^{\ell^2} \right) = O((\log m + R)^{L^2}/\sqrt{m}) = \tilde{O}(R^{L^2}/\sqrt{m}).$$

Finally by Lemma A.11.1, the spectral norm of H_{f_k} can be bounded by the summation of the spectral norm of all the Hessian blocks, i.e.,$\|H_{f_k}\| \le \sum_{\ell_1,\ell_2} \|H_{f_k}^{(\ell_1,\ell_2)}\|$. Applying the union bound over the indices of layers ℓ_1, ℓ_2, we finish the proof.

A.3 Feedforward neural networks with multiple output

In cases of multiple output neurons, the network function is vector-valued and its Hessian is a three-order tensor. The spectral norm of Hessian is defined in a standard way, i.e.,

$$\|\mathbf{H}_f(\mathbf{w})\| := \sup_{\|\mathbf{v}\|=\|\mathbf{u}\|=\|\mathbf{s}\|=1} \sum_{i,j,k} \big(\mathbf{H}_f(\mathbf{w})\big)_{i,j,k} v_i u_j s_k,$$

where $\mathbf{s} \in \mathbb{R}^{d_\ell}$ and $\mathbf{v}$, $\mathbf{u}$ have the same dimension with $\mathbf{w}$. It is not hard to see that $\|\mathbf{H}_f(\mathbf{w})\| \le d_\ell \max_{k\in[d_\ell]} \|H_{f_k}(\mathbf{w})\|$.

If the number of output neurons d_ℓ is bounded (as in most practical cases), the spectral norm of the Hessian of f is also of the order $\tilde{O}(1/\sqrt{m})$, with high probability, as a direct consequence of Theorem 2.3.6.

Corollary A.3.1. *Suppose Assumption 2.3.1, 2.3.2 and 2.3.5 hold. Given a fixed radius $R > 0$, for all $\mathbf{w} \in \mathsf{B}(\mathbf{w}_0, R)$, with probability at least $1 - \exp\big(-\Omega(\log^2 m)\big)$ over the random initialization*

$\mathbf{w}_0$, *a vector-valued feedforward neural network* f *satisfies*

$$\left\|\mathbf{H}_f(\mathbf{w})\right\| = \tilde{O}\left(\frac{R^{L^2}}{\sqrt{m}}\right). \tag{A.11}$$

A.4 Feedforward neural networks with skip connections

In this section, we discuss the property of transition to linearity holds for networks with skip connection.

We formally define the skip connection in the following. We add a skip connection to each neuron then the neuron functions Eq. (2.5) become

$$f_{i,\mathrm{res}}^{(\ell)} = \sigma_i^{(\ell)}\left(\tilde{f}_{i,\mathrm{res}}^{(\ell)}\right) + f_{B(\ell,i),\mathrm{res}}^{(A(\ell,i))}, \;\; \tilde{f}_{i,\mathrm{res}}^{(\ell)} = \frac{1}{\sqrt{m_i^{(\ell)}}}\left(\mathbf{w}_i^{(\ell)}\right)^T f_{\mathscr{I}_i^{(\ell)},\mathrm{res}}, \tag{A.12}$$

where $1 \le \ell \le L-1$ and $i \in [d_\ell]$. Here $A(\ell,i) \in \{0,\cdots,\ell-1\}$ denotes the layer index of the connected neuron by skip connection with respect to $f_{i,\mathrm{res}}^{(\ell)}$ and $B(\ell,i) \in [d_{A(\ell,i)}]$.

And for the output layer $\ell = L$, we define

$$f_{i,\mathrm{res}}^{(L)} = \tilde{f}_{i,\mathrm{res}}^{(L)} = \frac{1}{\sqrt{m_i^{(L)}}}\left(\mathbf{w}_i^{(L)}\right)^T f_{\mathscr{I}_i^{(L)},\mathrm{res}},$$

where $i \in [d_L]$.

The following theorem shows the property of transition to linearity holds for networks with skip connections. The proof of the theorem follows the almost identical idea with the proof of Theorem 2.3.6, hence we present the proof sketch and focus on the arguments that are new for f_{res}.

Theorem A.4.1 (Scaling of the Hessian norm for f_{res})**.** *Suppose Assumption 2.3.1, 2.3.2 and 2.3.5 hold. Given a fixed radius* $R > 0$*, for all* $\mathbf{w} \in \mathsf{B}(\mathbf{w}_0, R)$*, with probability at least* $1 -$

$\exp\big(-\Omega(\log^2 m)\big)$ *over the random initliazation of* $\mathbf{w}_0$*, each output neuron* $f_{k,\mathrm{res}}$ *satisfies*

$$\left\|H_{f_{k,\mathrm{res}}}(\mathbf{w})\right\| = \tilde{O}\left(\frac{R^{L^2}}{\sqrt{m}}\right), \quad \ell \in [L], \;\; k \in [d_\ell]. \tag{A.13}$$

Proof sketch of Theorem A.4.1. For each output $f_{k,\mathrm{res}}$, where $k \in [d_\ell]$, similar to the proof of Theorem 2.3.6, we bound the spectral norm of each Hessian block, i.e., $\frac{\partial^2 f_{k,\mathrm{res}}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}}$. Without loss of generality, we assume $1 \le \ell_1 \le \ell_2 \le L$.

Similar to Eq.(2.13), we derive the expression of the Hessian block by definition:

$$\frac{\partial^2 f_{k,\mathrm{res}}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} = \sum_{\ell'=\ell_2}^{L} \sum_{i=1}^{d_{\ell'}} \frac{\partial^2 f_{i,\mathrm{res}}^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \frac{\partial f_{k,\mathrm{res}}}{\partial f_{i,\mathrm{res}}^{(\ell')}} := \sum_{\ell'=\ell_2}^{L} G_{k,\mathrm{res}}^{L,\ell'}.$$

And again by chain rule of derivatives, each $G_{k,\mathrm{res}}^{L,\ell'}$ can be written as

$$\begin{aligned}
G_{k,\mathrm{res}}^{L,\ell'} = & \underbrace{\frac{1}{\sqrt{m_k^{(L)}}} \sum_{r=\ell'}^{L-1} \sum_{s: f_{s,\mathrm{res}}^{(r)} \in \mathscr{F}_{\mathscr{S}_k^{(L)}}} \left(\mathbf{w}_k^{(L)}\right)_{\mathsf{id}_{r,s}^{L,k}} \sigma'\left(\tilde{f}_{s,\mathrm{res}}^{(r)}\right) G_{s,\mathrm{res}}^{r,\ell'}}_{T_1} \\
& + \underbrace{\frac{1}{\sqrt{m_k^{(L)}}} \sum_{r=\ell'}^{L-1} \sum_{s: f_{s,\mathrm{res}}^{(r)} \in \mathscr{F}_{\mathscr{S}_k^{(L)}}} \left(\mathbf{w}_k^{(L)}\right)_{\mathsf{id}_{r,s}^{L,k}} \sigma'\left(\tilde{f}_{B(\ell',s),\mathrm{res}}^{(A(\ell',s))}\right) G_{B(\ell',s),\mathrm{res}}^{A(\ell',s),\ell'}}_{T_2} \\
& + \underbrace{\frac{1}{\sqrt{m_k^{(L)}}} \sum_{i: f_{i,\mathrm{res}}^{(\ell')} \in \mathscr{F}_{\mathscr{S}_{k,\mathrm{res}}^{(L)}}} \left(\mathbf{w}_k^{(L)}\right)_{\mathsf{id}_{\ell',i}^{L,k}} \left(\sigma''\left(\tilde{f}_{i,\mathrm{res}}^{(\ell')}\right) \frac{\partial \tilde{f}_{i,\mathrm{res}}^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)}} \left(\frac{\partial \tilde{f}_{i,\mathrm{res}}^{(\ell')}}{\partial \mathbf{w}^{(\ell_2)}}\right)^T\right)}_{T_3} \\
& + \underbrace{\frac{1}{\sqrt{m_k^{(L)}}} \sum_{i: f_{i,\mathrm{res}}^{(\ell')} \in \mathscr{F}_{\mathscr{S}_{k,\mathrm{res}}^{(L)}}} \left(\mathbf{w}_k^{(L)}\right)_{\mathsf{id}_{\ell',i}^{L,k}} \left(\sigma''\left(\tilde{f}_{B(\ell',i),\mathrm{res}}^{(A(\ell',i))}\right) \frac{\partial \tilde{f}_{B(\ell',i),\mathrm{res}}^{(A(\ell',i))}}{\partial \mathbf{w}^{(\ell_1)}} \left(\frac{\partial \tilde{f}_{B(\ell',i),\mathrm{res}}^{(A(\ell',i))}}{\partial \mathbf{w}^{(\ell_2)}}\right)^T\right)}_{T_4},
\end{aligned}$$

where $\mathscr{F}_{\mathscr{S}^{(L)}_{k,\text{res}}} := \{f : f \in f_{\mathscr{S}^{(L)}_{k,\text{res}}}\}$ and $\mathsf{id}^{L,k}_{\ell',i} := \{p : \left(f_{\mathscr{S}^{(L)}_{k},\text{res}}\right)_p = f^{(\ell')}_{i,\text{res}}\}$.

Note that the new terms which are induced by the skip connection in the above equation are

$$T_2 = \frac{1}{\sqrt{m^{(L)}_k}} \sum_{r=\ell'}^{L-1} \sum_{s: f^{(r)}_{s,\text{res}} \in \mathscr{F}_{\mathscr{S}^{(L)}_k}} \left(\mathbf{w}^{(L)}_k\right)_{\mathsf{id}^{L,k}_{r,s}} \sigma'\left(\tilde{f}^{(A(\ell',s))}_{B(\ell',s),\text{res}}\right) G^{A(\ell',s),\ell'}_{B(\ell',s),\text{res}},$$

and

$$T_4 = \frac{1}{\sqrt{m^{(L)}_k}} \sum_{i: f^{(\ell')}_{i,\text{res}} \in \mathscr{F}_{\mathscr{S}^{(L)}_{k,\text{res}}}} \left(\mathbf{w}^{(L)}_k\right)_{\mathsf{id}^{L,k}_{\ell',i}} \left(\sigma''\left(\tilde{f}^{(A(\ell',i))}_{B(\ell',i),\text{res}}\right) \frac{\partial \tilde{f}^{(A(\ell',i))}_{B(\ell',i),\text{res}}}{\partial \mathbf{w}^{(\ell_1)}} \left(\frac{\partial \tilde{f}^{(A(\ell',i))}_{B(\ell',i),\text{res}}}{\partial \mathbf{w}^{(\ell_2)}}\right)^T \right).$$

These two new terms take the same form with the original two terms i.e., T_1 and T_3, which are matrix Gaussian series with respect to the random variables $\mathbf{w}^{(L)}_k$. Therefore, we can use the same method as T_1 and T_3 to bound the spectral norm of T_2 and T_4.

As $A(\ell',i) < \ell'$ by definition, the bound on T_2 and T_4 will be automatically included in our recursive analysis. Then the rest of the proof is identical to the one for feedforward neural networks, i.e., the proof of Theorem 2.3.6.

□

A.5 Feedforward neural networks with shared weights, e.g., convolutional neural networks

In this section, we consider the feedforward neural networks where weight parameters are shared, e.g., convolutional neural networks, as an extension to our result where we assume each weight parameter $w_e \in \mathscr{W}$ is initialized i.i.d. We will show that such feedforward neural networks in which the weight parameters are shared constant times, i.e., independent of the width m, the property of transition to linearity still holds.

We formally define the networks with shared weights in the following:

$$f_{i,j,\text{cnn}}^{(\ell)} = \sigma_i^{(\ell)}\left(\tilde{f}_{i,j,\text{cnn}}^{(\ell)}\right), \quad \tilde{f}_{i,j,\text{cnn}}^{(\ell)} = \frac{1}{\sqrt{m_{i,j}^{(\ell)}}}\left(\mathbf{w}_i^{(\ell)}\right)^T f_{\mathscr{I}_{i,j}^{(\ell)},\text{cnn}}, \tag{A.14}$$

where $1 \le \ell \le L$, $i \in [d_\ell]$. We introduce new index $j \in [D(\ell,i)]$ where $D(\ell,i)$ denotes the number of times that weights $\mathbf{w}_i^{(\ell)}$ are shared. Note that the element in $f_{\mathscr{I}_{i,j}^{(\ell)},\text{cnn}}$ is allowed to be 0, corresponding to the zero padding which is commonly used in CNNs.

We similarly denote the output of the networks $f_{i,j,\text{cnn}}^{(L)}$ by $f_{i,j,\text{cnn}}$.

To see how CNNs fit into this definition, we consider a CNN with 1-D convolution as a simple example.

Convolutional neural networks

Given input $\boldsymbol{x} \in \mathbb{R}^d$, an ℓ-layer convolutional neural network is defined as follows:

$$\begin{aligned} &f^{(0)} = \boldsymbol{x}, \\ &f^{(\ell)} = \sigma\left(\frac{1}{\sqrt{m_{\ell-1}\times p}} W^{(\ell)} * f^{(\ell-1)}\right), \ \ \forall l \in [L-1], \\ &f_i(\mathbf{W};\boldsymbol{x}) = \frac{1}{\sqrt{m_{\ell-1}\times d}}\left\langle W_{[i,:,:]}^{(\ell)}, f^{(\ell-1)}\right\rangle, \ \ \forall i \in [d_\ell], \end{aligned} \tag{A.15}$$

where $\mathbf{W} = \left(W^{(1)},...,W^{(\ell)}\right)$ is the collection of all the weight matrices.

We denote the size of the window by $p \times 1$, hence $W^{(\ell)} \in \mathbb{R}^{m_\ell \times m_{\ell-1}\times p}$ for $\ell \in [L-1]$. We assume the stride is 1 for simplicity, and we do the standard zero-padding to each $f^{(\ell)}$ such that for each $\ell \in [L-1]$, $f^{(\ell)} \in \mathbb{R}^{m_\ell \times d}$. At the last layer, as $f^{(\ell-1)} \in \mathbb{R}^{m_{\ell-1}\times d}$ and $W^{(\ell)} \in \mathbb{R}^{m_L \times m_{\ell-1}\times d}$, we do the matrix inner product for each $i \in [d_\ell]$.

Now we show how Eq. (A.15) fits into Eq. (A.14). For $\ell \in [L-1]$, in Eq. (A.15), each component of $f^{(\ell)} \in \mathbb{R}^{m_\ell \times d}$ is computed as

$$f_{i,j}^{(\ell)} = \sigma\left(\frac{1}{\sqrt{m_{\ell-1}\times p}}\left\langle W_{[i,:,:]}^{(\ell)}, f_{\left[:,j-\lceil\frac{p-1}{2}\rceil:j+\lceil\frac{p-1}{2}\rceil\right]}^{(\ell-1)}\right\rangle\right).$$

Therefore, $m_{i,j}^{(\ell)}$, $\mathbf{w}_i^{(\ell)}$ and $f_{\mathscr{S}_{i,j}^{(\ell)},\mathrm{cnn}}$ in Eq. (A.14) correspond to $m_{\ell-1}\times p$, $W_{[i,:,:]}^{(\ell)}$ and $f^{(\ell-1)}_{[:,j-\lceil\frac{p-1}{2}\rceil:j+\lceil\frac{p-1}{2}\rceil]}$ respectively. For $\ell=L$, $m_{i,j}^{(L)}$ corresponds to $m_{L-1}\times d$ and $f_{\mathscr{S}_{i,j}^{(L)},\mathrm{cnn}}$ corresponds to $f^{(L-1)}$. Then we can see our definition of networks with shared weights, i.e., Eq. (A.14) includes standard CNN as an special example.

Similar to Theorem 2.3.6, we will show that the spectral norm of its Hessian can be controlled, hence the property of transition to linearity will hold for $f_{\mathrm{cnn}}^{(\ell)}$. The proof of the following theorem follows the almost identical idea with the proof of Theorem 2.3.6, hence we present the proof sketch and focus on the arguments that are new for f_{cnn}.

Theorem A.5.1. *Suppose Assumption 2.3.1, 2.3.2 and 2.3.5 hold. Given a fixed radius $R>0$, for all $\mathbf{w}\in\mathsf{B}(\mathbf{w}_0,R)$, with probability at least $1-\exp\big(-\Omega(\log^2 m)\big)$ over the random initliazation of $\mathbf{w}_0$, each output neuron $f_{i,j,\mathrm{cnn}}(\mathbf{w})$ satisfies*

$$\big\|H_{f_{i,j,\mathrm{cnn}}}(\mathbf{w})\big\| = O\left((\log m+R)^{\ell^2}/\sqrt{m}\right) = \tilde{O}\left(R^{\ell^2}/\sqrt{m}\right),\quad \ell\in[L],\ \ i\in[d_\ell],\ \ j\in[D(i,\ell)]. \tag{A.16}$$

Proof sketch of Theorem A.5.1. Similar to the proof of Theorem 2.3.6, by Lemma A.11.1, the spectral norm of $H_{f_{i,j,\mathrm{cnn}}}$ can be bounded by the summation of the spectral norm of all the Hessian blocks, i.e., $\|H_{f_{i,j,\mathrm{cnn}}}\|\le\sum_{\ell_1,\ell_2}\|H^{(\ell_1,\ell_2)}_{f_{i,j,\mathrm{cnn}}}\|$, where $H^{(\ell_1,\ell_2)}_{f_{i,j,\mathrm{cnn}}}:=\frac{\partial^2 f_k}{\partial\mathbf{w}^{(\ell_1)}\partial\mathbf{w}^{(\ell_2)}}$. Therefore, it suffices to bound the spectral norm of each block. Without lose of generality, we consider the block with $1\le\ell_1\le\ell_2\le L$.

By the chain rule of derivatives, we can write the Hessian block into:

$$\frac{\partial^2 f_{i,j,\mathrm{cnn}}}{\partial\mathbf{w}^{(\ell_1)}\partial\mathbf{w}^{(\ell_2)}} = \sum_{\ell'=\ell_2}^{L}\sum_{k=1}^{d_{\ell'}}\sum_{t=1}^{D(k,\ell')}\frac{\partial^2 f^{(\ell')}_{k,t,\mathrm{cnn}}}{\partial\mathbf{w}^{(\ell_1)}\partial\mathbf{w}^{(\ell_2)}}\frac{\partial f_{i,j,\mathrm{cnn}}}{\partial f^{(\ell')}_{k,t,\mathrm{cnn}}} := \sum_{\ell'=\ell_2}^{L} G^{L,\ell'}_{i,j,\mathrm{cnn}}. \tag{A.17}$$

For each $G^{L,\ell'}_{i,j,\mathrm{cnn}}$, since $f^{(\ell')}_{i,j,\mathrm{cnn}}=\sigma\left(\tilde{f}^{(\ell')}_{i,j,\mathrm{cnn}}\right)$, again by the chain rule of derivatives, we

have

$$
\begin{aligned}
G_{i,j,\mathrm{cnn}}^{L,\ell'} &= \sum_{k=1}^{d_{\ell'}} \sum_{t=1}^{D(k,\ell')} \frac{\partial^2 \tilde{f}_{k,t,\mathrm{cnn}}^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \frac{\partial f_{i,j,\mathrm{cnn}}}{\partial \tilde{f}_{k,t,\mathrm{cnn}}^{(\ell')}} \\
&\quad + \frac{1}{\sqrt{m_{i,j}^{(L)}}} \sum_{k,t: f_{k,t,\mathrm{cnn}}^{(\ell')} \in \mathscr{F}_{\mathscr{S}_{i,j,\mathrm{cnn}}^{(L)}}} \left(\mathbf{w}_i^{(L)}\right)_{\mathsf{id}_{\ell',k,t}^{L,i,j}} \sigma''\left(\tilde{f}_{k,t,\mathrm{cnn}}^{(\ell')}\right) \frac{\partial \tilde{f}_{k,t,\mathrm{cnn}}^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)}} \left(\frac{\partial \tilde{f}_{k,t,\mathrm{cnn}}^{(\ell')}}{\partial \mathbf{w}^{(\ell_2)}}\right)^T \\
&= \frac{1}{\sqrt{m_{i,j}^{(L)}}} \sum_{r=\ell'}^{L-1} \sum_{k,t: f_{k,t,\mathrm{cnn}}^{(r)} \in \mathscr{F}_{\mathscr{S}_{i,j,\mathrm{cnn}}^{(L)}}} \left(\mathbf{w}_i^{(L)}\right)_{\mathsf{id}_{r,k,t}^{L,i,j}} \sigma'\left(\tilde{f}_{k,t,\mathrm{cnn}}^{(r)}\right) G_{k,t,\mathrm{cnn}}^{r,\ell'} \\
&\quad + \frac{1}{\sqrt{m_{i,j}^{(L)}}} \sum_{k,t: f_{k,t,\mathrm{cnn}}^{(\ell')} \in \mathscr{F}_{\mathscr{S}_{i,j,\mathrm{cnn}}^{(L)}}} \left(\mathbf{w}_i^{(L)}\right)_{\mathsf{id}_{\ell',k,t}^{L,i,j}} \sigma''\left(\tilde{f}_{k,t,\mathrm{cnn}}^{(\ell')}\right) \frac{\partial \tilde{f}_{k,t,\mathrm{cnn}}^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)}} \left(\frac{\partial \tilde{f}_{k,t,\mathrm{cnn}}^{(\ell')}}{\partial \mathbf{w}^{(\ell_2)}}\right)^T,
\end{aligned}
$$

where $\mathscr{F}_{\mathscr{S}_{i,j,\mathrm{cnn}}^{(L)}} := \{f : f \in f_{\mathscr{S}_{i,j,\mathrm{cnn}}^{(L)}}\}$ and $\mathsf{id}_{\ell',k,t}^{L,i,j} := \{p : \left(f_{\mathscr{S}_{i,j}^{(L)},\mathrm{cnn}}\right)_p = f_{k,t,\mathrm{cnn}}^{(\ell')}\}$.

Compared to the derivation for standard feedforward neural networks, i.e., Eq. (2.14), there is an extra summation over the index t, whose carnality is at most $D(k,\ell')$. Recall that $D(k,\ell')$ denotes the number of times that the weight parameters $\mathbf{w}_k^{(\ell')}$ is shared. Therefore, as we assume $D(k,\ell')$ is independent of the width m, the norm bound will have the same order of m. Consequently, the spectral norm of each $G_{i,j,\mathrm{cnn}}^{L,\ell'}$ can be recursively bounded then Eq. (A.16) holds.

□

A.6 Feedforward neural networks with bottleneck neurons

In this section, we show that constant number of bottleneck neurons which serve as incoming neurons will not break the linearity.

We justify this claim based on the recursive relation in Eq. (2.13), which is used to prove the small spectral norm of the Hessian of the network function, hence proving the transition to linearity.

Recall that each Hessian block can be written into:

$$\frac{\partial^2 f_k}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} = \sum_{\ell'=\ell_2}^{L} \sum_{i=1}^{d_{\ell'}} \frac{\partial^2 f_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \frac{\partial f_k}{\partial f_i^{(\ell')}} := \sum_{\ell'=\ell_2}^{L} G_k^{L,\ell'}. \tag{A.18}$$

For each $G_k^{L,\ell'}$, we have a recursive form

$$\begin{aligned} G_k^{L,\ell'} &= \sum_{i=1}^{d_{\ell'}} \frac{\partial^2 \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \frac{\partial f_k}{\partial \tilde{f}_i^{(\ell')}} + \frac{1}{\sqrt{m_k^{(L)}}} \sum_{i: f_i^{(\ell')} \in \mathscr{F}_{\mathscr{S}_k^{(L)}}} \left(\mathbf{w}_k^{(L)}\right)_{\mathsf{id}_{\ell',i}^{L,k}} \sigma''\left(\tilde{f}_i^{(\ell')}\right) \frac{\partial \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)}} \left(\frac{\partial \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_2)}}\right)^T \\ &= \frac{1}{\sqrt{m_k^{(L)}}} \sum_{r=\ell'}^{L-1} \sum_{i: f_i^{(r)} \in \mathscr{F}_{\mathscr{S}_k^{(L)}}} \left(\mathbf{w}_k^{(L)}\right)_{\mathsf{id}_{r,i}^{L,k}} \sigma'\left(\tilde{f}_s^{(r)}\right) G_i^{r,\ell'} \\ &\quad + \frac{1}{\sqrt{m_k^{(L)}}} \sum_{i: f_i^{(\ell')} \in \mathscr{F}_{\mathscr{S}_k^{(L)}}} \left(\mathbf{w}_k^{(L)}\right)_{\mathsf{id}_{\ell',i}^{L,k}} \sigma''\left(\tilde{f}_i^{(\ell')}\right) \frac{\partial \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)}} \left(\frac{\partial \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_2)}}\right)^T, \end{aligned} \tag{A.19}$$

where $\mathscr{F}_{\mathscr{S}_k^{(L)}} := \{f : f \in f_{\mathscr{S}_k^{(L)}}\}$ and $\mathsf{id}_{\ell',i}^{L,k} := \{p : \left(f_{\mathscr{S}_k^{(L)}}\right)_p = f_i^{(\ell')}\}$.

As mentioned in Section 2.3, to prove the spectral norm of $G_k^{L,\ell'}$ is small, we need to bound the matrix variance, which suffices to bound the spectral norm of

$$\frac{1}{\sqrt{m_k^{(L)}}} \sum_{i: f_i^{(r)} \in \mathscr{F}_{\mathscr{S}_k^{(L)}}} G_i^{r,\ell'} \quad \text{and} \quad \frac{1}{\sqrt{m_k^{(L)}}} \sum_{i: f_i^{(\ell')} \in \mathscr{F}_{\mathscr{S}_k^{(L)}}} \frac{\partial \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)}} \left(\frac{\partial \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_2)}}\right)^T.$$

For the first quantity, if all $\tilde{f}_i^{(r)}$ are neurons with large in-degree, which is the case of our analysis by Assumption 2.3.5, then each $\tilde{f}_i^{(r)}$ will transition to linearity by Theorem 2.3.8. This is manifested as small spectral norm of $G_i^{r,\ell'}$ for all i. If some of $\tilde{f}_i^{(r)}$ are neurons with small in-degree, their corresponding $G_i^{r,\ell'}$ can be of a larger order, i.e., $O(1)$. However, note that the cardinally of the set $\mathscr{F}_{\mathscr{S}_k^{(L)}}$ is $m_k^{(L)}$. As long as the number of such neurons is not too large, i.e., $o\left(m_k^{(L)}\right)$, the order of the summation will be not affected. Therefore, the desired bound for the matrix variance will be the same hence the recursive argument can still apply.

The same analysis works for the second quantity as well. Neurons with small in-degree can make the norm of $\frac{\partial \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)}} \left(\frac{\partial \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_2)}} \right)^T$ be of a larger order. However, as long as the number of such neurons is not too large, the bound still holds.

For example, for the bottleneck neural network which has a narrow hidden layer (i.e., bottleneck layer) while the rest of hidden layers are wide, all neurons in the next layer to the bottleneck layer are bottleneck neurons. Such bottleneck neural networks were shown to break transition to linearity in [66]. However, we observe that for such bottleneck neural networks, the number of bottleneck neurons is large, a fixed fraction of all neurons. With our analysis, if we add trainable connections to the bottleneck neurons such that almost all (except a small number of) bottleneck neurons become neurons with sufficiently large in-degrees, then the resulting network can have the property of transition to linearity.

A.7 Proof of Proposition 2.4.4

Note that for any $k \in [d_\ell]$,

$$\|\nabla_{\mathbf{w}} f_k(\mathbf{w}_0)\| \geq \|\nabla_{\mathbf{w}^{(\ell)}} f_k(\mathbf{w}_0)\| = \left\| \frac{1}{\sqrt{m_k^{(\ell)}}} f_{\mathscr{S}_k^{(\ell)}} \right\| = \frac{1}{\sqrt{m_k^{(\ell)}}} \left\| f_{\mathscr{S}_k^{(\ell)}} \right\|.$$

Since $f_{\mathscr{S}_k^{(\ell)}}$ contains neurons from $\mathscr{P}^{(\ell)}$ (defined in Eq. (A.2)), in the following we prove $\mathbb{E}_{\mathbf{x},\mathbf{w}_0} \left| f_i^{(\ell)} \right|^2$ is uniformly bounded from 0 for any $\ell \in \{0,1,...,L-1\}$, $i \in [d_\ell]$.

Specifically, we will prove by induction that $\forall\ \ell \in \{0,1,...,L-1\}$, $\forall\ i \in [d_\ell]$,

$$\mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{w}_0} [|f_i^{(\ell)}|^2] \geq \min \left\{ 1, \min_{1 \leq j \leq \ell} C_\sigma^{\sum_{\ell'=0}^{j-1} r^{\ell'}} \right\}.$$

When $\ell = 0$, $\mathbb{E}_{\boldsymbol{x}} \left[|x_i|^2 \right] = 1$ for all $i \in [d_0]$ by Assumption 2.4.1.

Suppose for all $\ell \leq q-1$, $\mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{w}_0} [|f_i^{(\ell)}|^2] \geq \min \left(1, \min_{1 \leq j \leq q} C_\sigma^{\sum_{\ell'=0}^{j-1} r^{\ell'}} \right)$. When $\ell = q$,

$$\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{w}_0}[|f_i^{(q)}|^2] = \mathbb{E}_{\mathbf{w}_0}\left[\left|\sigma_i^{(q)}\left(\frac{1}{\sqrt{m_i^{(\ell)}}}(\mathbf{w}_i^{(q)})^T f_{\mathscr{S}_i^{(q)}}\right)\right|^2\right]$$
$$= \mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{w}_0}\mathbb{E}_{z\sim\mathscr{N}(0,1)}\left[\left|\sigma_i^{(q)}\left(\frac{\|f_{\mathscr{S}_i^{(q)}}\|}{\sqrt{m_i^{(q)}}}z\right)\right|^2\right].$$

By Assumption 2.4.2,

$$\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{w}_0}\mathbb{E}_{z\sim\mathscr{N}(0,1)}\left[\left|\sigma_i^{(q)}\left(\frac{\|f_{\mathscr{S}_i^{(q)}}\|}{\sqrt{m_i^{(q)}}}z\right)\right|^2\right] = \mathbb{E}_{z\sim\mathscr{N}(0,1)}[|\sigma_i^{(q)}(z)|^2]\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{w}_0}\left[\left(\frac{\|f_{\mathscr{S}_i^{(q)}}\|^2}{m_i^{(q)}}\right)^r\right]$$
$$\geq C_\sigma\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{w}_0}\left[\left(\frac{\|f_{\mathscr{S}_i^{(q)}}\|^2}{m_i^{(q)}}\right)^r\right]$$

We use Jensen's inequality,

$$C_\sigma\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{w}_0}\left[\left(\frac{\|f_{\mathscr{S}_i^{(q)}}\|^2}{m_i^{(q)}}\right)^r\right] \geq C_\sigma\left(\frac{\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{w}_0}\left[\|f_{\mathscr{S}_i^{(q)}}\|^2\right]}{m_i^{(q)}}\right)^r$$

Then according to inductive assumption, we have

$$C_\sigma\left(\frac{\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{w}_0}\left[\|f_{\mathscr{S}_i^{(q)}}\|^2\right]}{m_i^{(q)}}\right)^r \geq C_\sigma\left(\min\left(1, \min_{1\leq j\leq q} C_\sigma^{\sum_{\ell'=0}^{j-1} r^{\ell'}}\right)\right)^r$$
$$\geq \min_{1\leq j\leq q+1} C_\sigma^{\sum_{\ell'=0}^{j-1} r^{\ell'}}.$$

Hence for all $l \leq q$, $\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{w}_0}[|f_i^{(\ell)}|^2] \geq \min\left(1, \min_{1\leq j\leq q+1} C_\sigma^{\sum_{\ell'=0}^{j-1} r^{\ell'}}\right)$, which finishes the inductive step hence the proof.

Therefore,

$$\mathbb{E}_{\mathbf{x},\mathbf{w}_0}\left[\|\nabla_{\mathbf{w}^{(\ell)}} f_k(\mathbf{w}_0)\|\right] = \mathbb{E}_{\mathbf{x},\mathbf{w}_0}\left[\frac{1}{\sqrt{m_k^{(\ell)}}}\left\|f_{\mathscr{S}_k^{(\ell)}}\right\|\right] \geq \sqrt{\min\left(1, \min_{1\leq j\leq L} C_\sigma^{\sum_{\ell'=0}^{j-1} r^{\ell'}}\right)} = \Omega(1).$$

A.8 Proof of Lemma A.2.1

We prove the result by induction.

For the base case when $\ell = \ell' + 1$,

$$\begin{aligned}
\left\|\frac{\partial f_{S_j^{(\ell)}}}{\partial \mathbf{w}^{(\ell-1)}}\right\| &= \left\|\frac{\partial f^{(\ell-1)}}{\partial \mathbf{w}^{(\ell-1)}}\frac{\partial f_{S_j^{(\ell)}}}{\partial f^{(\ell-1)}}\right\| \\
&\leq \max_{i:f_i^{(\ell-1)}\in\mathscr{F}_{\mathscr{S}_j^{(\ell)}}} \frac{1}{\sqrt{m_i^{(\ell-1)}}}\left|\sigma'(\tilde{f}_i^{(\ell-1)})\right|\left\|f_{\mathscr{S}_i^{(\ell-1)}}\right\| \\
&\leq \max_{i:f_i^{(\ell-1)}\in\mathscr{F}_{\mathscr{S}_j^{(\ell)}}} \frac{\gamma_1}{\sqrt{m_i^{(\ell-1)}}}\left\|f_{\mathscr{S}_i^{(\ell-1)}}\right\|.
\end{aligned}$$

and

$$\begin{aligned}
\left\|\frac{\partial f_{S_j^{(\ell)}}}{\partial \mathbf{w}^{(\ell-1)}}\right\|_F &= \sqrt{\sum_{i:f_i^{(\ell-1)}\in\mathscr{F}_{\mathscr{S}_j^{(\ell)}}}\left\|\frac{\partial f_i^{(\ell-1)}}{\partial \mathbf{w}^{(\ell-1)}}\right\|^2} \\
&\leq \sqrt{m_j^{(\ell)}}\max_{i:f_i^{(\ell-1)}\in\mathscr{F}_{\mathscr{S}_j^{(\ell)}}} \frac{1}{\sqrt{m_i^{(\ell-1)}}}\left|\sigma'(\tilde{f}_i^{(\ell-1)})\right|\left\|f_{\mathscr{S}_i^{(\ell-1)}}\right\| \\
&\leq \sqrt{m_j^{(\ell)}}\max_{i:f_i^{(\ell-1)}\in\mathscr{F}_{\mathscr{S}_j^{(\ell)}}} \frac{\gamma_1}{\sqrt{m_i^{(\ell-1)}}}\left\|f_{\mathscr{S}_i^{(\ell-1)}}\right\|.
\end{aligned}$$

By Lemma A.11.3, with probability at least $1 - m_i^{(\ell-1)}\exp\left(-C_{\ell-1}^{\mathscr{P}}\log^2 m\right)$,

$$\left\|f_{\mathscr{S}_i^{(\ell-1)}}\right\| = O\left((\log m + R)^{\ell-2}\sqrt{m_i^{(\ell-1)}}\right) = \tilde{O}\left(R^{\ell-2}\sqrt{m_i^{(\ell-1)}}\right).$$

For the maximum norm $\max_i \left\| f_{\mathscr{S}_i^{(\ell-1)}} \right\| / \sqrt{m_i^{(\ell-1)}}$, we apply union bound over the indices i such that $f_i^{(\ell-1)} \in \mathscr{F}_{\mathscr{S}_j^{(\ell)}}$, the cardinality of which is at most $\left| \mathscr{F}_{\mathscr{S}_j^{(\ell)}} \right| = m_j^{(\ell)}$. Hence with probability at least $1 - m_i^{(\ell-1)} m_j^{(\ell)} \exp\left(-C_{\ell-1}^{\mathscr{P}} \log^2 m\right)$,

$$\max_i \left\| f_{\mathscr{S}_i^{(\ell-1)}} \right\| / \sqrt{m_i^{(\ell-1)}} = O\left((\log m + R)^{\ell-2} \right) = \tilde{O}(R^{\ell-2}).$$

Since $m_i^{(\ell-1)} \le \overline{m}_{\ell-1}$ and $m_j^{(\ell)} \le \overline{m}_\ell$ where $\overline{m}_{\ell-1}, \overline{m}_\ell$ are polynomial in m, we can find a constant $C_{\ell,\ell-1}^f > 0$ such that $\exp\left(-C_{\ell,\ell-1}^f \log^2 m\right) \ge \exp\left(-C_{\ell-1}^{\mathscr{P}} \log^2 m\right) \cdot \exp(\log(\overline{m}_{\ell-1} \cdot \overline{m}_\ell))$. As a result, with probability at least $1 - \exp\left(-C_{\ell,\ell-1}^f \log^2 m\right)$,

$$\begin{aligned}
\left\| \frac{\partial f_{S_j^{(\ell)}}}{\partial \mathbf{w}^{(\ell-1)}} \right\| &= O\left((\log m + R)^{\ell-1} \right) = \tilde{O}(R^{\ell-1}), \\
\left\| \frac{\partial f_{S_j^{(\ell)}}}{\partial \mathbf{w}^{(\ell-1)}} \right\|_F &= O\left(\sqrt{m_j^{(\ell)}} (\log m + R)^{\ell-1} \right) = \tilde{O}\left(\sqrt{m_j^{(\ell)}} R^{\ell-1} \right).
\end{aligned}$$

Suppose $\ell \le k$, Eq. (A.8) and (A.9) hold with probability at least $1 - \exp\left(-C_{k,\ell'}^f \log^2 m\right)$.

For $\ell = k+1$, since elements of $f_{\mathscr{S}_j^{(k+1)}}$ are from $\mathscr{P}^{(k)}$ where only $f^{(\ell')}, ..., f^{(k)}$ possibly depend on $\mathbf{w}^{(\ell')}$, we have

$$\frac{\partial f_{\mathscr{S}_j^{(k+1)}}}{\partial \mathbf{w}^{(\ell')}} = \sum_{q=\ell'+1}^{k} \sum_{i: f_i^{(q)} \in \mathscr{F}_{\mathscr{S}_j^{(k+1)}}} \frac{\partial f_{\mathscr{S}_i^{(q)}}}{\partial \mathbf{w}^{(\ell')}} \frac{\partial f_i^{(q)}}{\partial f_{\mathscr{S}_i^{(q)}}} \frac{\partial f_{\mathscr{S}_j^{(k+1)}}}{\partial f_i^{(q)}}. \tag{A.20}$$

With simple computation, we know that for any i s.t. $f_i^{(q)} \in \mathscr{F}_{\mathscr{S}_j^{(k+1)}}$:

$$\frac{\partial f_i^{(q)}}{\partial f_{\mathscr{S}_i^{(q)}}} \frac{\partial f_{\mathscr{S}_j^{(k+1)}}}{\partial f_i^{(q)}} = \frac{1}{\sqrt{m_i^{(q)}}} \sigma'(\tilde{f}_i^{(q)}) \mathbf{w}_i^{(q)} \frac{\partial f_{\mathscr{S}_j^{(k+1)}}}{\partial f_i^{(q)}},$$

where $\frac{\partial f_{\mathscr{S}_j^{(k+1)}}}{\partial f_i^{(q)}}$ is a mask matrix defined in Eq. (A.3).

Supposing that $\partial f_{\mathscr{S}_i^{(q)}}/\partial \mathbf{w}^{(\ell')}$, $i \in [d_q]$ in Eq. (A.20) is fixed, for each q, we apply Lemma A.11.6 to bound the spectral norm. Choosing $t = \sqrt{m_j^{(k+1)}}\log m$, with probability at least $1 - 2\exp\left(-m_j^{(k+1)}\log^2 m\right)$, for some absolute constant $C > 0$,

$$\begin{aligned}
&\left\| \sum_{i: f_i^{(q)} \in \mathscr{F}_{\mathscr{S}_j^{(k+1)}}} \frac{\partial f_{\mathscr{S}_i^{(q)}}}{\partial \mathbf{w}^{(\ell')}} \frac{\partial f_i^{(q)}}{\partial f_{\mathscr{S}_i^{(q-1)}}} \frac{\partial f_{\mathscr{S}_j^{(k+1)}}}{\partial f_i^{(q)}} \right\| \\
&\leq C\gamma_1 \left(\max_i \frac{1}{\sqrt{m_i^{(q)}}} \left\| \frac{\partial f_{\mathscr{S}_i^{(q)}}}{\partial \mathbf{w}^{(\ell')}} \right\| \left(\sqrt{m_j^{(k+1)}} + \sqrt{m_j^{(k+1)}}\log m + R \right) + \max_i \frac{1}{\sqrt{m_i^{(q)}}} \left\| \frac{\partial f_{S_i^{(q)}}}{\partial \mathbf{w}^{(\ell')}} \right\|_F \right).
\end{aligned} \tag{A.21}$$

To bound the Frobenious norm of Eq. (A.20) for each q, we apply Lemma A.11.7 and choose $t = \left\| \partial f_{S_i^{(q)}}/\partial \mathbf{w}^{(\ell')} \right\| \log m$. By union bound over indices i such that $f_i^{(q)} \in \mathscr{F}_{\mathscr{S}_j^{(k+1)}}$, then with probability at least $1 - 2m_j^{(k+1)}\exp\left(-c'\log^2 m\right)$, where $c' > 0$ is a constant, we have

$$\begin{aligned}
&\left\| \sum_{i: f_i^{(q)} \in \mathscr{F}_{\mathscr{S}_j^{(k+1)}}} \frac{\partial f_{\mathscr{S}_i^{(q)}}}{\partial \mathbf{w}^{(\ell')}} \frac{\partial f_i^{(q)}}{\partial f_{\mathscr{S}_i^{(q)}}} \frac{\partial f_{\mathscr{S}_j^{(k+1)}}}{\partial f_i^{(q)}} \right\|_F \\
&= \sqrt{\sum_{i: f_i^{(q)} \in \mathscr{F}_{\mathscr{S}_j^{(k+1)}}} \left\| \frac{\partial f_{\mathscr{S}_i^{(q)}}}{\partial \mathbf{w}^{(\ell')}} \frac{\partial f_i^{(q)}}{\partial f_{\mathscr{S}_i^{(q)}}} \right\|^2} \\
&\leq \sqrt{m_j^{(k+1)}} \max_i \left\| \frac{\partial f_{\mathscr{S}_i^{(q)}}}{\partial \mathbf{w}^{(\ell')}} \frac{1}{\sqrt{m_i^{(q)}}} \left(\mathbf{w}_i^{(q)} \right)^T \sigma'(\tilde{f}_i^{(q)}) \right\| \\
&\leq \gamma_1 \sqrt{m_j^{(k+1)}} \max_i \frac{1}{\sqrt{m_i^{(q)}}} \left(\left\| \frac{\partial f_{\mathscr{S}_i^{(q)}}}{\partial \mathbf{w}^{(\ell')}} \right\| (\log m + R) + \left\| \frac{\partial f_{\mathscr{S}_i^{(q)}}}{\partial \mathbf{w}^{(\ell')}} \right\|_F \right).
\end{aligned} \tag{A.22}$$

To bound the maximum of $\left\|\partial f_{\mathscr{S}_i^{(q)}}/\partial \mathbf{w}^{(\ell')}\right\|/\sqrt{m_i^{(q)}}$ and $\left\|\partial f_{\mathscr{S}_i^{(q)}}/\partial \mathbf{w}^{(\ell')}\right\|_F/\sqrt{m_i^{(q)}}$ that appear in Eq. (A.21) and (A.22), with the induction hypothesis, we apply union bound over indices i such that $f_i^{(q)} \in \mathscr{F}_{\mathscr{S}_j^{(k+1)}}$. With probability at least $1-m_j^{(k+1)}\exp\left(-C_{q,\ell'}^f \log^2 m\right)$,

$$\max_i \frac{1}{\sqrt{m_i^{(q)}}}\left\|\frac{\partial f_{\mathscr{S}_i^{(q)}}}{\partial \mathbf{w}^{(\ell')}}\right\| = O\left(\max_{\ell'+1\le p\le q}\frac{1}{\sqrt{\underline{m}_p}}(\log m+R)^{\ell'}\right) = \tilde{O}\left(\max_{\ell'+1\le p\le q}\frac{R^{\ell'}}{\sqrt{\underline{m}_p}}\right),$$
$$\max_i \frac{1}{\sqrt{m_i^{(q)}}}\left\|\frac{\partial f_{\mathscr{S}_i^{(q)}}}{\partial \mathbf{w}^{(\ell')}}\right\|_F = O\left((\log m+R)^{q-1}\right) = \tilde{O}\left(R^{q-1}\right).$$

Putting them in Eq. (A.21) and (A.22), we have

$$\left\|\sum_{i:f_i^{(q)}\in\mathscr{F}_{\mathscr{S}_j^{(k+1)}}}\frac{\partial f_{\mathscr{S}_i^{(q)}}}{\partial \mathbf{w}^{(\ell')}}\frac{\partial f_i^{(q)}}{\partial f_{\mathscr{S}_i^{(q)}}}\frac{\partial f_{\mathscr{S}_j^{(k+1)}}}{\partial f_i^{(q)}}\right\| = \tilde{O}\left(\max\left(\left(\max_{\ell'+1\le p\le q}\frac{\sqrt{m_j^{(k+1)}}}{\sqrt{\underline{m}_p}}\right),1\right)\right),$$
$$\left\|\sum_{i:f_i^{(q)}\in\mathscr{F}_{\mathscr{S}_j^{(k+1)}}}\frac{\partial f_{\mathscr{S}_i^{(q)}}}{\partial \mathbf{w}^{(\ell')}}\frac{\partial f_i^{(q)}}{\partial f_{\mathscr{S}_i^{(q)}}}\frac{\partial f_{\mathscr{S}_j^{(k+1)}}}{\partial f_i^{(q)}}\right\|_F = \tilde{O}\left(\sqrt{m_j^{(k+1)}}\right),$$

with probability at least

$$1-2\exp\left(-m_j^{(k+1)}\log^2 m\right)-m_j^{(k+1)}\exp\left(-C_{q,\ell'}^f\log^2 m\right)-2m_j^{(k+1)}\exp\left(-c'\log^2 m\right).$$

As the current result is for fixed q, applying the union bound over indices $q\in\{\ell'+1,...,k\}$, we have with probability at least

$$1-2(k-\ell')\exp\left(-m_j^{(k+1)}\right)-\sum_q m_j^{(k+1)}\exp\left(-C_{q,\ell'}^f\log^2 m\right)-2m_j^{(k+1)}\exp\left(-c'\log^2 m\right),$$

$$
\begin{aligned}
&\left\| \sum_{q=\ell'+1}^{k} \sum_{i:f_i^{(q)} \in \mathscr{F}_{\mathscr{S}_j^{(k+1)}}} \frac{\partial f_{\mathscr{S}_i^{(q)}}}{\partial \mathbf{w}^{(\ell')}} \frac{\partial f_i^{(q)}}{\partial f_{\mathscr{S}_i^{(q)}}} \frac{\partial f_{\mathscr{S}_j^{(k+1)}}}{\partial f_i^{(q)}} \right\| \\
&= O\left(\max\left(\max_{\ell'+1 \le p \le k} \frac{\sqrt{m_j^{(k+1)}}}{\sqrt{\underline{m}_p}}, 1 \right) (\log m + R)^{\ell'} \right) \\
&= \tilde{O}\left(\max_{\ell'+1 \le p \le k+1} \frac{\sqrt{m_j^{(k+1)}}}{\sqrt{\underline{m}_p}} R^{\ell'} \right), \\
&\left\| \sum_{q=\ell'+1}^{k} \sum_{i:f_i^{(q)} \in \mathscr{F}_{\mathscr{S}_j^{(k+1)}}} \frac{\partial f_{\mathscr{S}_i^{(q)}}}{\partial \mathbf{w}^{(\ell')}} \frac{\partial f_i^{(q)}}{\partial f_{\mathscr{S}_i^{(q)}}} \frac{\partial f_{\mathscr{S}_j^{(k+1)}}}{\partial f_i^{(q)}} \right\|_F \\
&= O\left(\sqrt{m_j^{(k+1)}} (\log m + R)^k \right) \\
&= \tilde{O}\left(\sqrt{m_j^{(k+1)}} R^k \right).
\end{aligned}
$$

Since $m_j^{(k+1)}$ is upper bounded by $\overline{m}_{k+1}$ which is polynomial in m, we can find a constant $C_{k+1,\ell'}^f > 0$ such that for each j, the result holds with probability at least $1 - \exp\left(-C_{k+1,\ell'}^f \log^2 m\right)$ for $\ell \le k+1$. Then we finish the inductive step which completes the proof.

A.9 Proof of Lemma A.2.2

Before the proof, by Assumption 2.3.5, we have the following proposition which is critical in the tail bound of the norm of the matrix Gaussian series, i.e., Lemma A.11.8. In the bound, there will be a dimension factor which is the number of parameters (see Eq. (A.32)). Note that the number of parameters at each layer can be exponentially large w.r.t. the width m. If we naively apply the bound, the bound will be useless. However, each neuron in fact only depends on polynomial in m number of parameters, which is the dimension factor we should use.

Proposition A.9.1. *Fixed $\ell' \in [L]$, we denote the maximum number of elements in $\mathbf{w}^{(\ell')}$ that $f_i^{(\ell)}$ depends on for all $\ell \in [L], i \in [d_\ell]$ by $m_{\ell'}^*$, which is polynomial in m.*

The proof the proposition can be found in Appendix A.10.

Now we start the proof of the lemma. In fact, we will prove a more general result which includes the neurons in output layer, i.e. ℓ-th layer. And we will use the result of Lemma A.2.1 in the proof. Specifically, we will prove the following lemma:

Lemma A.9.2. *Given $1 \le \ell_1 \le \ell_2 \le L$, for any $\ell_2 \le \ell \le L$, $\mathbf{w} \in \mathsf{B}(\mathbf{w}_0, R)$, and $j \in [d_\ell]$, we have, with probability at least $1 - \exp\big(-\Omega(\log^2 m)\big)$,*

$$\left\| \frac{\partial^2 \tilde{f}_j^{(\ell)}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \right\| = O\left(\max_{\ell_1+1 \le p \le \ell} \frac{1}{\sqrt{\underline{m}_p}} (\log m + R)^{\ell^2} \right) = \tilde{O}\left(\max_{\ell_1+1 \le p \le \ell} \frac{R^{\ell^2}}{\sqrt{\underline{m}_p}} \right). \tag{A.23}$$

We will prove the results by induction.

For the base case that $\ell = \ell_2$,

$$\left\| \frac{\partial^2 \tilde{f}_j^{(\ell_2)}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \right\| = \left\| \frac{1}{\sqrt{m_j^{(\ell_2)}}} \frac{\partial f_{S_j^{(\ell_2)}}}{\partial \mathbf{w}^{(\ell_1)}} \right\|.$$

By Lemma A.2.1, we can find a constant $M_{\ell_1,\ell_2}^{(\ell_2),j} > 0$ such that with probability at least $1 - \exp\left(-M_{\ell_1,\ell_2}^{(\ell_2),j} \log^2 m\right)$,

$$\left\| \frac{1}{\sqrt{m_j^{(\ell_2)}}} \frac{\partial f_{S_j^{(\ell_2)}}}{\partial \mathbf{w}^{(\ell_1)}} \right\| = O\left(\max_{\ell_1+1 \le p \le \ell_2} \frac{1}{\sqrt{\underline{m}_p}} (\log m + R)^{\ell_1} \right).$$

Suppose for $\ell_2 \le \ell' \le \ell$, with probability at least $1 - \exp\left(-M_{\ell_1,\ell_2}^{(\ell),j} \log^2 m\right)$ for some constant $M_{\ell_1,\ell_2}^{(\ell),j} > 0$, Eq. (A.10) holds.

When $\ell' = \ell + 1$, we have

$$\left\| \frac{\partial^2 \tilde{f}_j^{(\ell+1)}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \right\| = \left\| \sum_{\ell'=\ell_2}^{\ell} \sum_{i=1}^{d_{\ell'}} \frac{\partial^2 f_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \frac{\partial \tilde{f}_j^{(\ell+1)}}{\partial f_i^{(\ell')}} \right\| \le \sum_{\ell'=\ell_2}^{\ell} \left\| \sum_{i=1}^{d_{\ell'}} \frac{\partial^2 f_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \frac{\partial \tilde{f}_j^{(\ell+1)}}{\partial f_i^{(\ell')}} \right\|. \tag{A.24}$$

We will bound every term in the above summation. For each term, by definition,

$$\begin{aligned}
&\sum_{i=1}^{d_{\ell'}} \frac{\partial^2 f_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \frac{\partial \tilde{f}_j^{(\ell+1)}}{\partial f_i^{(\ell')}} \\
&= \sum_{i=1}^{d_{\ell'}} \frac{\partial^2 \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \frac{\partial \tilde{f}_j^{(\ell+1)}}{\partial \tilde{f}_i^{(\ell')}} \\
&\quad + \frac{1}{\sqrt{m_j^{(\ell+1)}}} \sum_{i: f_i^{(\ell')} \in \mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}} (\mathbf{w}_j^{(\ell+1)})_{\mathsf{id}_{\ell',i}^{\ell+1,j}} \sigma''(\tilde{f}_i^{(\ell')}) \frac{\partial \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)}} \left(\frac{\partial \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_2)}} \right)^T . \qquad \text{(A.25)}
\end{aligned}$$

For the first term in Eq. (A.25), we use Lemma A.11.10. Specifically, we view $U_i = \frac{\partial^2 \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}}$, hence with probability at least $1 - \sum_{k=1}^{\ell-\ell'+1} k(m_{\ell_1}^* + m_{\ell_2}^*) \exp\big(-\log^2 m/2\big)$,

$$\left\| \sum_{i=1}^{d_{\ell'}} \frac{\partial^2 \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \frac{\partial \tilde{f}_j^{(\ell+1)}}{\partial \tilde{f}_i^{(\ell')}} \right\| = O\left(\max_{i: f_i^{(\ell')} \in \mathscr{F}_{\mathscr{S}_j^{(\ell'+1)}}} \left\| \frac{\partial^2 \tilde{f}_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \right\| (\log m + R)^{\ell-\ell'+1} \right).$$

Here we'd like to note that from Lemma A.11.8, the tail bound depends on the dimension of $\mathbf{w}^{(\ell_1)}$ and $\mathbf{w}^{(\ell_2)}$ which are $\sum_{i=1}^{d_{\ell_1}} m_i^{(\ell)}$ and $\sum_{i=1}^{d_{\ell_2}} m_i^{(\ell)}$ respectively. By Assumption 2.3.5, for any ℓ, $m_i^{(\ell)}$ is polynomial in m. Therefore, the number of elements in $\mathbf{w}^{(\ell)}$ that $f_j^{(\ell+1)}$ depends on is polynomial in m by Proposition A.9.1. And the matrix variance $\tilde{v}^{(\ell')}$ in Lemma A.11.10 is equivalent to the matrix variance that we only consider the elements in $\mathbf{w}^{(\ell_1)}$ and $\mathbf{w}^{(\ell_2)}$ that $f_j^{(\ell+1)}$ depends on, in which case the dimension is polynomial in m. Therefore we can use m_ℓ^* here. It is the same in the following when we apply matrix Gaussian series tail bound.

Then we apply union bound over indices i such that $f_i^{(\ell')} \in \mathscr{F}_{\mathscr{S}_j^{(\ell'+1)}}$, whose cardinality is at most $m_j^{(\ell+1)}$. By the inductive hypothesis, with probability at least $1 - \sum_{k=1}^{\ell-\ell'+1} k(m_{\ell_1}^* +$

$m^*_{\ell_2})\exp\big(-\log^2 m/2\big) - m_j^{(\ell+1)}\exp\Big(-M^{(\ell),j}_{\ell_1,\ell_2}\log^2 m\Big)$,

$$\left\|\sum_{i=1}^{d_{\ell'}}\frac{\partial^2\tilde{f}_i^{(\ell')}}{\partial\mathbf{w}^{(\ell_1)}\partial\mathbf{w}^{(\ell_2)}}\frac{\partial\tilde{f}_j^{(\ell+1)}}{\partial\tilde{f}_i^{(\ell')}}\right\| = O\left(\max_{\ell_1+1\le p\le\ell'}\frac{1}{\sqrt{\underline{m}_p}}(\log m+R)^{(\ell')^2+\ell-\ell'+1}\right).$$

For the second term in Eq. (A.25), we view it as a matrix Gaussian series with respect to $\mathbf{w}_j^{(\ell+1)}$. The matrix variance takes the form

$$\begin{aligned}\nu^{(\ell'),j}_{\ell_1,\ell_2} &= \frac{1}{m_j^{(\ell+1)}}\max\\ &\left\{\left\|\sum_{i:f_i^{(\ell')}\in\mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}}\left(\sigma''(\tilde{f}_i^{(\ell')})\right)^2\left\|\frac{\partial\tilde{f}_i^{(\ell')}}{\partial\mathbf{w}^{(\ell_1)}}\right\|^2\frac{\partial\tilde{f}_i^{(\ell')}}{\partial\mathbf{w}^{(\ell_2)}}\left(\frac{\partial\tilde{f}_i^{(\ell')}}{\partial\mathbf{w}^{(\ell_2)}}\right)^T\right\|,\right.\\ &\left.\left\|\sum_{i:f_i^{(\ell')}\in\mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}}\left(\sigma''(\tilde{f}_i^{(\ell')})\right)^2\left\|\frac{\partial\tilde{f}_i^{(\ell')}}{\partial\mathbf{w}^{(\ell_2)}}\right\|^2\frac{\partial\tilde{f}_i^{(\ell')}}{\partial\mathbf{w}^{(\ell_1)}}\left(\frac{\partial\tilde{f}_i^{(\ell')}}{\partial\mathbf{w}^{(\ell_1)}}\right)^T\right\|\right\}.\end{aligned}$$

We use Lemma A.11.9 to bound $\nu^{(\ell'),j}_{\ell_1,\ell_2}$. With the definition in Eq. (A.33), we rewrite the above equation as $\nu^{(\ell'),j}_{\ell_1,\ell_2} = \max\left\{\mu^{(\ell'),j}_{\ell_1,\ell_2},\mu^{(\ell'),j}_{\ell_2,\ell_1}\right\}$. Hence with probability at least at least $1-\exp\left(-C^{(\ell'),j}_{\ell_1,\ell_2}\log^2 m\right)-\exp\left(-C^{(\ell'),j}_{\ell_2,\ell_1}\log^2 m\right)$ for some constant $C^{(\ell'),j}_{\ell_1,\ell_2}, C^{(\ell'),j}_{\ell_2,\ell_1}>0$, we have

$$\nu^{(\ell'),j}_{\ell_1,\ell_2} = O\left(\max\left(1/m_j^{(\ell+1)}, \max_{\ell_1+1\le p\le\ell}1/\underline{m}_p\right)(\log m+R)^{4\ell'-2}\right).$$

Using Lemma A.11.8 again and choosing $t=\log m\sqrt{\nu^{(\ell'),j}_{\ell_1,\ell_2}}$, we have with probability at least $1-(m^*_{\ell_2}+m^*_{\ell_1})\exp\big(-\log^2 m/2\big)$,

$$\left\|\frac{1}{\sqrt{m_j^{(\ell+1)}}}\sum_{i:f_i^{(\ell')}\in\mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}}\left(\mathbf{w}_j^{(\ell+1)}\right)_{\mathsf{id}^{\ell+1,j}_{\ell',i}}\sigma''\left(\tilde{f}_i^{(\ell')}\right)\frac{\partial\tilde{f}_i^{(\ell')}}{\partial\mathbf{w}^{(\ell_1)}}\left(\frac{\partial\tilde{f}_i^{(\ell')}}{\partial\mathbf{w}^{(\ell_2)}}\right)^T\right\| \le (\log m+R)\sqrt{\nu^{(\ell'),j}_{\ell_1,\ell_2}}.$$

Combined the bound on $\nu_{\ell_1,\ell_2}^{(\ell'),j}$, with probability at least

$$1-\exp\left(-C_{\ell_1,\ell_2}^{(\ell'),j}\log^2 m\right)-\exp\left(-C_{\ell_2,\ell_1}^{(\ell'),j}\log^2 m\right)-(m_{\ell_2}^*+m_{\ell_1}^*)\exp\left(-\log^2 m/2\right),$$

we have

$$\begin{aligned}
&\left\|\frac{1}{\sqrt{m_j^{(\ell+1)}}}\sum_{i:f_i^{(\ell')}\in\mathscr{F}_j^{(\ell+1)}}\left(\mathbf{w}_j^{(\ell+1)}\right)_{\mathsf{id}_{\ell',i}^{\ell+1,j}}\sigma''\left(\tilde{f}_i^{(\ell')}\right)\frac{\partial\tilde{f}_i^{(\ell')}}{\partial\mathbf{w}^{(\ell_1)}}\left(\frac{\partial\tilde{f}_i^{(\ell')}}{\partial\mathbf{w}^{(\ell_2)}}\right)^T\right\| \\
&= O\left(\max\left(1/\sqrt{m_j^{(\ell_2+1)}},\max_{\ell_1+1\le p\le\ell}1/\sqrt{\underline{m}_p}\right)(\log m+R)^{2\ell'}\right) \\
&= \tilde{O}\left(\max\left(1/\sqrt{m_j^{(\ell_2+1)}},\max_{\ell_1+1\le p\le\ell}1/\sqrt{\underline{m}_p}\right)R^{2\ell'}\right) \\
&= \tilde{O}\left(\max_{\ell_1+1\le p\le\ell+1}R^{2\ell'}/\sqrt{\underline{m}_p}\right).
\end{aligned}$$

Now we have bound both terms in Eq. (A.25). Combining the bounds, we have with probability at least $1-\sum_{k=1}^{\ell-\ell'+1}k(m_{\ell_1}^*+m_{\ell_2}^*)\exp\left(-\log^2 m/2\right)-m_j^{(\ell+1)}\exp\left(-M_{\ell_1,\ell_2}^{(\ell),j}\log^2 m\right)-\exp\left(-C_{\ell_1,\ell_2}^{(\ell'),j}\log^2 m\right)-\exp\left(-C_{\ell_2,\ell_1}^{(\ell'),j}\log^2 m\right)-2(m_{\ell_2}^*+m_{\ell_1}^*)\exp\left(-\log^2 m/2\right)$

$$\left\|\sum_{i=1}^{d_{\ell'}}\frac{\partial^2 f_i^{(\ell')}}{\partial\mathbf{w}^{(\ell_1)}\partial\mathbf{w}^{(\ell_2)}}\frac{\partial\tilde{f}_j^{(\ell+1)}}{\partial f_i^{(\ell')}}\right\|=O\left(\max_{\ell_1+1\le p\le\ell+1}1/\sqrt{\underline{m}_p}(\log m+R)^{(\ell'+1)^2+\ell-\ell'}\right).$$

With the above results, to bound Eq. (A.24), we apply the union bound over the layer indices $l'=\ell_2,...,\ell$. Then with probability at least

$$\begin{aligned}
&1-\sum_{\ell'=\ell_2}^{l}\sum_{k=1}^{\ell-\ell'+1}k(m_{\ell_1}^*+m_{\ell_2}^*)\exp\left(-\log^2 m/2\right)-(\ell-\ell_2+1)m_j^{(\ell+1)}\exp\left(-M_{\ell_1,\ell_2}^{(\ell),j}\log^2 m\right) \\
&-\sum_{\ell'=\ell_2}^{l}\exp\left(-C_{\ell_1,\ell_2}^{(\ell'),j}\log^2 m\right)-\sum_{\ell'=\ell_2}^{l}\exp\left(-C_{\ell_2,\ell_1}^{(\ell'),j}\log^2 m\right) \\
&-2(\ell-\ell_2+1)(m_{\ell_2}^*+m_{\ell_1}^*)\exp\left(-\log^2 m/2\right),
\end{aligned}$$

we have

$$\begin{aligned}\left\|\frac{\partial^2 \tilde{f}_j^{(\ell+1)}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}}\right\| &\leq \sum_{\ell'=\ell_2}^{\ell} \left\| \sum_{i=1}^{d_{\ell'}} \frac{\partial^2 f_i^{(\ell')}}{\partial \mathbf{w}^{(\ell_1)} \partial \mathbf{w}^{(\ell_2)}} \frac{\partial \tilde{f}_j^{(\ell+1)}}{\partial f_i^{(\ell')}} \right\| \\ &= O\left(\max_{\ell_1+1 \leq p \leq \ell+1} 1/\sqrt{\underline{m}_p} (\log m + R)^{(\ell+1)^2} \right) \\ &= \tilde{O}\left(\max_{\ell_1+1 \leq p \leq \ell+1} R^{(\ell+1)^2} / \sqrt{\underline{m}_p} \right).\end{aligned}$$

By Proposition A.9.1, $m_{\ell_1}^*, m_{\ell_2}^*$ are also polynomial in m. Hence, we can find a constant $M_{\ell_1,\ell_2}^{(\ell+1),j} > 0$ such that

$$\begin{aligned}&\exp\left(-M_{\ell_1,\ell_2}^{(\ell+1),j} \log^2 m\right) \\ &> \sum_{\ell'=\ell_2}^{\ell} \sum_{k=1}^{\ell-\ell'+1} k(m_{\ell_1}^* + m_{\ell_2}^*) \exp\left(-\log^2 m/2\right) - (\ell - \ell_2 + 1) m_j^{(\ell+1)} \exp\left(-M_{\ell_1,\ell_2}^{(\ell),j} \log^2 m\right) \\ &\quad - \sum_{\ell'=\ell_2}^{\ell} \exp\left(-C_{\ell_1,\ell_2}^{(\ell'),j} \log^2 m\right) - \sum_{\ell'=\ell_2}^{\ell} \exp\left(-C_{\ell_2,\ell_1}^{(\ell'),j} \log^2 m\right) \\ &\quad - 2(\ell - \ell_2 + 1)(m_{\ell_2}^* + m_{\ell_1}^*) \exp\left(-\log^2 m/2\right) + \exp\left(-M_{\ell_1,\ell_2}^{(\ell),j} \log^2 m\right).\end{aligned}$$

Then Eq. (A.25) holds with probability at least $1 - \exp\left(-M_{\ell_1,\ell_2}^{(\ell+1),j} \log^2 m\right)$ for any $\ell_2 \leq \ell + 1 \leq L$, $j \in [d_{\ell+1}]$, which finishes the induction step hence completes the proof.

A.10 Proof of Proposition A.9.1

Fixing $\ell' \in [L]$, for any $\ell \in \{\ell', ..., L\}$, $i \in [d_\ell]$, we first show $f_i^{(\ell)}$ depends on polynomial number of elements in $\mathbf{w}^{(\ell')}$. We prove the result by induction.

If $\ell = \ell'$, then the number of elements in $\mathbf{w}^{(\ell)}$ that $f_i^{(\ell)}$ depend on is $m_i^{(\ell)}$.

Suppose $\ell' \leq \ell \leq k$ that $f_i^{(\ell)}$ depends on polynomial number of elements in $\mathbf{w}^{(\ell')}$. Then

at $\ell = k+1$, we know

$$f_i^{(k+1)} = \sigma_i^{(k+1)} \left(\frac{1}{\sqrt{m_i^{(k+1)}}} \left\langle \mathbf{w}_i^{(k+1)}, f_{\mathscr{S}_i^{(k+1)}} \right\rangle \right).$$

As $f_{\mathscr{S}_i^{(k+1)}}$ contains $m_i^{(k+1)}$ neurons where each one depends on polynomial number of elements in $\mathbf{w}^{(\ell')}$ by the induction hypothesis. The composition of two polynomial functions is still polynomial, hence $f_i^{(k+1)}$ also depends on polynomial number of elements in $\mathbf{w}^{(\ell')}$.

The maximum number of elements in $\mathbf{w}^{(\ell')}$ that $f_i^{(\ell)}$ depends on among all layers ℓ is polynomial since it is the maximum of a finite sequence. By Assumption 2.3.5 that $\sup_{\ell \in \{2,\dots,L-1\}, i \in [d_\ell]} m_i^{(\ell)} = O(m^c)$, it is not hard to see that the maximum among all $i \in [d_\ell]$ is also polynomial.

A.11 Technical Lemmas and their proofs

Lemma A.11.1. *Spectral norm of a matrix H is upper bounded by the sum of the spectral norm of its blocks.*

Proof.

$$\begin{aligned}
\|H\| &= \left\| \begin{pmatrix} H^{(1,1)} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} + \begin{pmatrix} 0 & H^{(1,2)} & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} + \cdots + \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & H^{(L,L)} \end{pmatrix} \right\| \\
&\leq \sum_{\ell_1, \ell_2} \|H^{(\ell_1,\ell_2)}\|.
\end{aligned}$$

□

Lemma A.11.2. *For $\ell = 0, 1, .., L$, with probability at least $1 - \exp\left(-C_\ell^{\mathscr{P}} \log^2 m\right)$ for some constant $C_\ell^{\mathscr{P}} > 0$, the absolute value of all neurons in $\mathscr{P}^{(\ell)}$ Eq. (A.2) is of the order $\tilde{O}(1)$ in the*

ball $\mathsf{B}(\mathbf{w}_0, R)$.

Proof. We prove the result by induction.

When $\ell = 0$, $\mathscr{P}^{(0)} = f^{(0)} = \{x_1, ..., x_{d_0}\}$ therefore for all i, $|f_i^{(0)}| \leq C_{\boldsymbol{x}}$ surely by Assumption 2.3.1.

Suppose when $\ell = k$, with probability at least $1 - \exp\left(-C_k^{\mathscr{P}} \log^2 m\right)$, the absolute value of each neuron in $\mathscr{P}^{(k)}$ is of the order $O\left((\log m + R)^k\right)$ where $C_k^{\mathscr{P}} > 0$ is a constant. Then when $\ell = k+1$, there will be new neurons $f^{(k+1)}$ added to $\mathscr{P}^{(k)}$, where each $f_i^{(k+1)}$ can be bounded by

$$\begin{aligned} |f_i^{(k+1)}| &= \left| \sigma\left(\frac{1}{\sqrt{m_i^{(k+1)}}} \left(\mathbf{w}_i^{(k+1)}\right)^T f_{\mathscr{S}_i^{(k+1)}} \right) \right| \\ &\leq \frac{\gamma_1}{\sqrt{m_i^{(k+1)}}} \left(\mathbf{w}_i^{(k+1)}\right)^T f_{\mathscr{S}_i^{(k+1)}} + \sigma(0). \end{aligned}$$

By the union bound over all the elements in $f_{\mathscr{S}_i^{(k+1)}}$ which are in $\mathscr{P}^{(k)}$ and the induction hypothesis, with probability at least $1 - m_i^{(k+1)} \exp\left(-C_k^{\mathscr{P}} \log^2 m\right)$,

$$\|f_{\mathscr{S}_i^{(k+1)}}\| = \sqrt{\sum_{j=1}^{m_i^{(k+1)}} \left(f_{\mathscr{S}_i^{(k+1)}}\right)_j} = O\left(\sqrt{m_i^{(k+1)}} (\log m + R)^k\right).$$

By Lemma A.11.4, supposing $f_{\mathscr{S}_i^{(k+1)}}$ is fixed, choosing $t = \log m \left\| f_{\mathscr{S}_i^{(k+1)}} \right\|$, with probability at least $1 - 2\exp\left(-\log^2 m/2\right)$, in the ball $\mathsf{B}(\mathbf{w}_0, R)$,

$$\left| (\mathbf{w}_i^{(k+1)})^T f_{\mathscr{S}_i^{(k+1)}} \right| \leq (\log m + R) \left\| f_{\mathscr{S}_i^{(k+1)}} \right\|.$$

Combined with the bound on $\|f_{\mathscr{S}_i^{(k+1)}}\|$, with probability at least $1 - 2\exp\left(-\log^2 m/2\right) -$

$m_i^{(k+1)} \exp\big(-C_k^{\mathscr{P}} \log^2 m\big)$,

$$\left|f_i^{(k+1)}\right| \leq \frac{\gamma_1}{\sqrt{m_i^{(k+1)}}}(\log m + R)\left\|f_{\mathscr{S}_i^{(k+1)}}\right\| + \gamma_0 = O\left((\log m + R)^{k+1}\right) = \tilde{O}(R^{k+1}).$$

Since $m_i^{(k+1)} \leq \overline{m}_{k+1}$ which is polynomial in m, we can find a constant $C_{k+1}^{\mathscr{P}} > 0$ such that for all i, we have

$$\begin{aligned}\exp\Big(-C_{k+1}^{\mathscr{P}} \log^2 m\Big) \geq\ & 2\exp\big(-\log^2 m/2\big) + \exp\Big(-C_k^{\mathscr{P}} \log^2(m) + \log(\overline{m}_{k+1})\Big) \\ & + \exp\Big(-C_k^{\mathscr{P}} \log^2 m\Big).\end{aligned}$$

Hence the above results hold with probability $1 - \exp\big(-C_{k+1}^{\mathscr{P}} \log^2 m\big)$, which completes the proof. □

Lemma A.11.3. *For $\ell \in [L], i \in [d_\ell]$, with probability at least $1 - m_i^{(\ell)} \exp\big(-C_\ell^{\mathscr{P}} \log^2 m\big)$, in the ball $\mathsf{B}(\mathbf{w}_0, R)$,*

$$\left\|f_{\mathscr{S}_i^{(\ell)}}\right\| = O\left(\sqrt{m_i^{(\ell)}}(\log m + R)^{\ell-1}\right) = \tilde{O}\left(\sqrt{m_i^{(\ell)}} R^{\ell-1}\right)$$

Proof. By Lemma A.11.2, each neuron is of order $\tilde{O}(1)$. Then we apply union bound over $m_i^{(\ell)}$ neurons and we get the result. □

Lemma A.11.4. *Given a fixed vector $\boldsymbol{b} \in \mathbb{R}^n$ and a random vector $\mathbf{a}_0 \sim \mathscr{N}(0, I_n)$, for any $\mathbf{a}$ in the ball $\mathsf{B}(\mathbf{a}_0, R)$, we have with probability at least $1 - 2\exp\big(-t^2/(2\|\boldsymbol{b}\|^2)\big)$,*

$$|\mathbf{a}^T \boldsymbol{b}| \leq t + \|\boldsymbol{b}\| R. \tag{A.26}$$

Proof. We can write $\mathbf{a}^T \boldsymbol{b} = (\mathbf{a}_0 + \Delta\mathbf{a})^T \boldsymbol{b} = \mathbf{a}_0^T \boldsymbol{b} + \Delta\mathbf{a}^T \boldsymbol{b}$. Since $\mathbf{a}_0 \sim \mathscr{N}(0,1)$, we have $\mathbf{a}_0^T \boldsymbol{b} \sim \mathscr{N}(0, \|\mathbf{b}\|^2)$. By Proposition 2.5.2 in [107], for any $t > 0$, with probability at least $1 -$

$2\exp\left(-t^2/(2\|\boldsymbol{b}\|^2)\right)$,

$$|\mathbf{a}_0^T\boldsymbol{b}| \le t.$$

Therefore, with the same probability

$$|\mathbf{a}^T\boldsymbol{b}| \le |\mathbf{a}_0^T\boldsymbol{b}| + |\Delta\mathbf{a}^T\boldsymbol{b}| \le t + \|\boldsymbol{b}\|R.$$

□

Lemma A.11.5. *For a random $m \times n$ matrix $W = [B_1\boldsymbol{a}_1, B_2\boldsymbol{a}_2, ..., B_n\boldsymbol{a}_n]$ where $A = [\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_n]$ is an $N_i \times n$ random matrix whose entries i.i.d. follow $\mathcal{N}(0,1)$ and $B_1, B_2, ..., B_n$ is a sequence of $m \times N_i$ non-random matrices, we have for some absolute constant $C > 0$, for any $t \ge 0$*

$$\|W\| \le C\left(\max_i \|B_i\|(\sqrt{n}+t) + \max_i \|B_i\|_F\right) \tag{A.27}$$

with probability at least $1 - 2\exp\left(-t^2\right)$.

Proof. We prove the result using an ε-net argument. Choosing $\varepsilon = 1/4$, by Corollary 4.2.13 in [107], we can find an ε-net $\mathcal{N}$ of the sphere S^{n-1} with cardinalities $|\mathcal{N}| \le 9^n$.

By Lemma 4.4.1 in [107], $\|W\| \le 2\sup_{\boldsymbol{x}\in\mathcal{N}} \|W\boldsymbol{x}\|$.

Fix $\boldsymbol{x} \in \mathcal{N}$, it is nor hard to see that

$$W\boldsymbol{x} = \sum_{i=1}^n x_i B_i \boldsymbol{a}_i \sim \mathcal{N}\left(0, \sum_{i=1}^n x_i^2 B_i B_i^T\right),$$

which can be viewed as $B'\boldsymbol{z}$ where $B' = \sqrt{\sum_{i=1}^n x_i^2 B_i B_i^T}$ and $\boldsymbol{z} \sim \mathcal{N}(0, I_m)$.

By Theorem 6.3.2 in [107], we have

$$\left\| \|B'\boldsymbol{z}\| - \|B'\|_F \right\|_{\psi_2} \le CK^2\|B'\|,$$

where $K = \max_i \|z_i\|_{\psi_2}$ and $\|\cdot\|_{\psi_2}$ is the sub-guassian norm (see Definition 2.5.6 in [107]) and C is an absolute constant.

By the definition of sub-gaussian norm, we can use the tail bound. For some positive absolute constant c and for any $\mu > 0$,

$$\mathbb{P}\left\{\|B'\boldsymbol{z}\| - \|B'\|_F \geq u\right\} \leq 2\exp\left(-cu^2/(K^4\|B'\|^2)\right).$$

Then we unfix $\boldsymbol{x}$ using a union bound. We have with probability at least $1 - 9^n 2\exp\left(-cu^2/(K^4\|B'\|^2)\right)$,

$$\sup_{\boldsymbol{x}\in\mathcal{N}} \|B'\boldsymbol{z}\| - \|B'\|_F \leq \mu.$$

Choose $u = CK^2\|B'\|(\sqrt{n}+t)$. If the constant C is chosen sufficiently large, we can let $cu^2/K^4 \geq 3n + t^2$. Thus,

$$\mathbb{P}\left\{\sup_{\boldsymbol{x}\in\mathcal{N}} \|B'\boldsymbol{z}\| - \|B'\|_F \geq u\right\} \leq 9^n 2\exp\left(-3n - t^2\right) \leq 2\exp\left(-t^2\right).$$

Combined with $\|W\| \leq 2\sup_{\boldsymbol{x}\in\mathcal{N}} \|W\boldsymbol{x}\|$, we conclude that

$$\mathbb{P}\left\{\|W\| \geq 2CK^2\|B'\|(\sqrt{n}+t) + 2\|B'\|_F\right\} \leq 2\exp\left(-t^2\right).$$

Noticing that $\|B'\| \leq \max_i \|B_i\|$ and $\|B'\|_F \leq \max_i \|B_i\|_F$, we have

$$\mathbb{P}\left\{\|W\| \geq 2CK^2 \max_i \|B_i\|(\sqrt{n}+t) + 2\max_i \|B_i\|_F\right\} \leq 2\exp\left(-t^2\right).$$

We absorb K into C as K is a constant. With abuse of notation of C which is absolute, we have

$$\mathbb{P}\left\{\|W\| \geq C(\max_i \|B_i\|(\sqrt{n}+t) + \max_i \|B_i\|_F)\right\} \leq 2\exp\left(-t^2\right).$$

□

Lemma A.11.6. *For a random $m \times n$ matrix $W = [B_1\boldsymbol{a}_1, B_2\boldsymbol{a}_2, ..., B_n\boldsymbol{a}_n]$ where $A = [\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_n]$ and $B_1, B_2, ..., B_n$ is a sequence of $m \times N$ non-random matrices. Here $A = A_0 + \Delta A$ where A_0 is an $N \times n$ random matrix whose entries i.i.d. follow $\mathcal{N}(0,1)$ and ΔA is a fixed matrix with $\|\Delta A\|_F \leq R$ given constant $R > 0$. We have for some absolute constant $C > 0$, for any $t \geq 0$*

$$\|W\| \leq C\left(\max_i \|B_i\|(\sqrt{n} + R + t) + \max_i \|B_i\|_F\right) \tag{A.28}$$

with probability at least $1 - 2\exp\left(-t^2\right)$.

Proof. Comparing to Lemma A.11.5, we only need to bound the norm of ΔW:

$$\Delta W := [B_1\Delta\boldsymbol{a}_1, B_2\Delta\boldsymbol{a}_2, , ..., B_n\Delta\boldsymbol{a}_n],$$

where $\Delta A = [\Delta\boldsymbol{a}_1, \Delta\boldsymbol{a}_2, ..., \Delta\boldsymbol{a}_n]$.

By the definition that $\|A_0\|_F = \sqrt{\sum_{i=1}^n \|\Delta\boldsymbol{a}_i\|^2}$, we have

$$\|\Delta W\| \leq \|\Delta W\|_F = \sqrt{\sum_{i=1}^n \|B_i\Delta\boldsymbol{a}_i\|^2} \leq \max_i \|B_i\|\|\Delta A\|_F \leq \max_i \|B_i\|R.$$

Therefore, for any $t \geq 0$, with probability at least $1 - 2\exp\left(-t^2\right)$,

$$\|W\| \leq \|W - \Delta W\| + \|\Delta W\| \leq C\left(\max_i \|B_i\|(\sqrt{n} + R + t) + \max_i \|B_i\|_F\right).$$

□

Lemma A.11.7. *Consider a fixed matrix $B \in \mathbb{R}^{m\times n}$ and a random vector $\mathbf{a}_0 \sim \mathcal{N}(0, I_n)$. For any $\mathbf{a} \in \mathbb{R}^n$ in the ball $\mathsf{B}(\mathbf{a}_0, R)$ given constant $R > 0$, for any $t > 0$, we have with probability at*

least $1-2\exp\left(-ct^2/\|B\|^2\right)$*,where c is an absolute constant,*

$$\|B\mathbf{a}\| \leq t + \|B\|_F + \|B\|R. \tag{A.29}$$

Proof. By Theorem 6.3.2 in [107], for any $t > 0$,

$$\mathbb{P}\{|\|B\mathbf{a}_0\| - \|B\|_F| \geq t\} \leq 2\exp\left(-ct^2/\|B\|^2\right),$$

where $c > 0$ is an absolute constant.

Note that $\|B\mathbf{a}\| \leq \|B\mathbf{a}_0\| + \|B(\mathbf{a}-\mathbf{a}_0)\| \leq \|B\mathbf{a}_0\| + \|B\|R$. With probability at least $1-2\exp\left(-ct^2/\|B\|^2\right)$, we have

$$\|B\mathbf{a}\| \leq t + \|B\|_F + \|B\|R.$$

□

Lemma A.11.8 (Matrix Gaussian series)**.** *For a sequence of fixed matrices* $\{B_k\}_{k=1}^n$ *with dimension* $d_1 \times d_2$ *and a sequence of independent standard normal variables* $\{\gamma_k\}$*, we define* $Z = \sum_{k=1}^n (\gamma_k + \Delta\gamma_k)B_k$ *where* $\{\Delta\gamma_k\}_{k=1}^n$ *is a fixed sequence with* $\sum_{k=1}^n \Delta\gamma_k^2 \leq R^2$ *given constant* $R > 0$*. Then we have for any* $t \geq 0$*, with probability at least* $1-(d_1+d_2)\exp\left(-t^2/(2v)\right)$*,*

$$\|Z\| \leq t + Rv, \tag{A.30}$$

where

$$v = \max\left\{\left\|\sum_k B_k B_k^T\right\|, \left\|\sum_k B_k^T B_k\right\|\right\}. \tag{A.31}$$

Proof. By Theorem 4.1.1 in [106], for all $t \geq 0$,

$$\mathbb{P}(\left\| \sum_{k=1}^{n} \gamma_k B_k \right\| \geq t) \leq (d_1 + d_2) \exp\left(\frac{-t^2}{2\nu} \right). \tag{A.32}$$

Since

$$\begin{aligned} \left\| Z - \sum_{k=1}^{n} \gamma_k B_k \right\| &= \left\| \sum_{k=1}^{n} \Delta\gamma_k B_k \right\| \\ &\leq \sqrt{\sum_{k=1}^{n} (\Delta\gamma_k)^2} \sqrt{\left\| \sum_{k=1}^{n} B_k B_k^T \right\|} \\ &\leq R\sqrt{\nu}. \end{aligned}$$

Then for Z, we have

$$\mathbb{P}(\|Z\| \geq t + R\sqrt{\nu}) \leq (d_1 + d_2) \exp\left(\frac{-t^2}{2\nu} \right).$$

□

Lemma A.11.9 (Bound on matrix variance)**.** *For any $\ell \in [L], \ell_1, \ell_2 \in [\ell], j \in [d_{\ell+1}]$, with probability at least $1 - \exp\left(-C_{\ell_1,\ell_2}^{(\ell),j} \log^2 m\right)$ for some constant $C_{\ell_1,\ell_2}^{(\ell),j} > 0$, we have*

$$\begin{aligned} \mu_{\ell_1,\ell_2}^{(\ell),j} &:= \frac{1}{m_j^{(\ell+1)}} \left\| \sum_{i: f_i^{(\ell)} \in \mathscr{F}_j^{(\ell+1)}} \left(\sigma'' \left(\tilde{f}_i^{(\ell)} \right) \right)^2 \left\| \frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell_1)}} \right\|^2 \frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell_2)}} \left(\frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell_2)}} \right)^T \right\| \\ &= O\left(\max\left(1/m_j^{(\ell+1)}, \max_{\min(\ell_1,\ell_2)+1 \leq p \leq \ell} 1/\underline{m}_p \right) (\log m + R)^{4\ell-2} \right) \\ &= \tilde{O}\left(\max\left(1/m_j^{(\ell+1)}, \max_{\min(\ell_1,\ell_2)+1 \leq p \leq \ell} 1/\underline{m}_p \right) R^{4\ell-2} \right). \end{aligned} \tag{A.33}$$

Proof. Without lose of generality, we assume $\ell_1 \leq \ell_2 \leq \ell$.

We consider two scenarios, (a) $\ell_1 \leq \ell_2 = \ell$ and (b) $\ell_1 \leq \ell_2 < \ell$.

In the scenario (a), we analyze $\ell_1 = \ell_2 = \ell$ and $\ell_1 < \ell_2 = \ell$ respectively.

When $\ell_1 = \ell_2 = \ell$, by definition,

$$\frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell_1)}} = \frac{1}{\sqrt{m_i^{(\ell)}}} f_{\mathscr{S}_i^{(\ell)}}.$$

By Lemma A.11.3, with probability at least $1 - m_i^{(\ell)} \exp\left(-C_\ell^{\mathscr{P}} \log^2 m\right)$, $\left\| f_{\mathscr{S}_i^{(\ell)}} \right\| = O\left(\sqrt{m_i^{(\ell)}} (\log m + R)^{\ell-1}\right) = \tilde{O}\left(\sqrt{m_i^{(\ell)}}\right)$. Applying union bound over the indices i such that $f_i^{(\ell)} \in f_{\mathscr{S}_j^{(\ell+1)}}$, the carnality of which is at most $m_j^{(\ell+1)}$, we have with probability at least $1 - m_i^{(\ell)} m_j^{(\ell+1)} \exp\left(-C_\ell^{\mathscr{P}} \log^2 m\right)$,

$$\max_{i: f_i^{(\ell)} \in \mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}} \left\| \frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell_1)}} \right\| = \max_{i: f_i^{(\ell)} \in \mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}} \frac{\left\| f_{\mathscr{S}_i^{(\ell)}} \right\|}{\sqrt{m_i^{(\ell)}}} = O\left((\log m + R)^{\ell-1}\right) = \tilde{O}(R^{\ell-1}).$$

It is not hard to see that

$$\sum_{i: f_i^{(\ell)} \in \mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}} \frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell)}} \left(\frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell)}} \right)^T$$

is a block diagonal matrix with i-th block in the form $\frac{1}{m_i^{(\ell)}} f_{\mathscr{S}_i^{(\ell)}} \left(f_{\mathscr{S}_i^{(\ell)}} \right)^T \cdot \mathbb{I}\left\{ f_i^{(\ell)} \in f_{\mathscr{S}_j^{(\ell+1)}} \right\}$.

Therefore, $\mu_{\ell,\ell}^{(\ell),j}$ can be bounded by

$$\begin{aligned}
\mu_{\ell,\ell}^{(\ell),j} &\leq \frac{1}{m_j^{(\ell+1)}} \gamma_2^2 \left(\max_{i: f_i^{(\ell)} \in \mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}} \frac{\left\| f_{\mathscr{S}_i^{(\ell)}} \right\|}{\sqrt{m_i^{(\ell)}}} \right)^2 \left(\max_{i: f_i^{(\ell)} \in \mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}} \left\| \frac{1}{m_i^{(\ell)}} f_{\mathscr{S}_i^{(\ell)}} \left(f_{\mathscr{S}_i^{(\ell)}} \right)^T \right\| \right) \\
&= O\left((\log m + R)^{4\ell-4} / m_j^{(\ell+1)} \right) = \tilde{O}\left(R^{4\ell-4} / m_j^{(\ell+1)} \right),
\end{aligned}$$

with probability at least $1 - 2m_i^{(\ell)} m_j^{(\ell+1)} \exp\left(-C_\ell^{\mathscr{P}} \log^2 m\right)$, where we apply the union bound on $\left\| f_{\mathscr{S}_i^{(\ell)}} \right\|$ once again.

By definition Eq. (A.7), $m_i^{(\ell)} \le \overline{m}_\ell$ and $m_j^{(\ell+1)} \le \overline{m}_{\ell+1}$. By Assumption 2.3.5, $\overline{m}_\ell, \overline{m}_{\ell+1}$ are polynomial in m. If m is large enough, we can find a constant $C_{\ell,\ell}^{(\ell),j} > 0$ such that

$$\exp\Big(-C_{\ell,\ell}^{(\ell),j}\log^2 m\Big) > 2m_i^{(\ell)}m_j^{(\ell+1)}\exp\Big(-C_\ell^{\mathscr{P}}\log^2 m\Big),$$

thus the bound holds with probability $1-\exp\Big(-C_{\ell,\ell}^{(\ell),j}\log^2 m\Big)$.

When $\ell_1 < \ell_2 = \ell$, By Eq. (2.5), we compute the derivative:

$$\frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell_1)}} = \frac{1}{\sqrt{m_i^{(\ell)}}}\frac{\partial f_{\mathscr{S}_i^{(\ell)}}}{\partial \mathbf{w}^{(\ell_1)}}\mathbf{w}_i^{(\ell)}. \tag{A.34}$$

By Lemma A.2.1, with probability at least $1-\exp\Big(-C_{\ell,\ell_1}^f\log^2 m\Big)$, we have

$$\Big\|\partial f_{\mathscr{S}_i^{(\ell)}}/\partial \mathbf{w}^{(\ell_1)}\Big\| = \tilde{O}\left(\max_{\ell_1+1\le p\le \ell}\sqrt{m_i^{(\ell)}}/\sqrt{\underline{m}_p}\right)$$

and $\Big\|\partial f_{\mathscr{S}_i^{(\ell)}}/\partial \mathbf{w}^{(\ell_1)}\Big\|_F = \tilde{O}\left(\sqrt{m_i^{(\ell)}}\right)$. We use Lemma A.11.7 and choose

$$t = \log m\Big\|\partial f_{\mathscr{S}_i^{(\ell)}}/\partial \mathbf{w}^{(\ell_1)}\Big\|,$$

then with probability at least $1-2\exp\big(-c'\log^2 m\big)-\exp\Big(-C_{\ell,\ell_1}^f\log^2 m\Big)$ for some absolute constant $c'>0$,

$$\begin{aligned}
\left\|\frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell_1)}}\right\| &= \frac{1}{\sqrt{m_i^{(\ell)}}}\left\|\frac{\partial f_{\mathscr{S}_i^{(\ell)}}}{\partial \mathbf{w}^{(\ell_1)}}\mathbf{w}_i^{(\ell)}\right\| \\
&\le \frac{1}{\sqrt{m_i^{(\ell)}}}\left((\log m+R)\left\|\frac{\partial f_{\mathscr{S}_i^{(\ell)}}}{\partial \mathbf{w}^{(\ell_1)}}\right\| + \left\|\frac{\partial f_{\mathscr{S}_i^{(\ell)}}}{\partial \mathbf{w}^{(\ell_1)}}\right\|_F\right) \\
&= O\Big((\log m+R)^\ell\Big) = \tilde{O}(R^\ell).
\end{aligned} \tag{A.35}$$

Similar to the case when $\ell_1 = \ell_2 = \ell$,

$$\sum_{i:f_i^{(\ell)} \in \mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}} \frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell)}} \left(\frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell)}} \right)^T$$

is a block matrix.

Therefore,

$$\begin{aligned}
\mu_{\ell,\ell}^{(\ell),j} &\leq \frac{1}{m_j^{(\ell+1)}} \gamma_2^2 \left(\max_{i:f_i^{(\ell)} \in \mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}} \left\| \frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell_1)}} \right\| \right)^2 \left(\max_{i:f_i^{(\ell)} \in \mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}} \left\| \frac{1}{m_i^{(\ell)}} f_{\mathscr{S}_i^{(\ell)}} \left(f_{\mathscr{S}_i^{(\ell)}} \right)^T \right\| \right) \\
&= O\left((\log m + R)^{4\ell-2} / m_j^{(\ell+1)} \right) = \tilde{O}\left(R^{4\ell-2} / m_j^{(\ell+1)} \right),
\end{aligned}$$

with probability at least

$$1 - 2m_j^{(\ell+1)} \exp\left(-c' \log^2 m\right) - m_j^{(\ell+1)} \exp\left(-C_{\ell,\ell_1}^f \log^2 m\right) - 2m_i^{(\ell)} m_j^{(\ell+1)} \exp\left(-C_\ell^{\mathscr{P}} \log^2 m\right),$$

where we apply the union bound over the indices i for the maximum.

Similarly, we can find a constant $C_{\ell_1,\ell}^{(\ell),j} > 0$ such that the bound holds with probability $1 - \exp\left(-C_{\ell,\ell}^{(\ell),j} \log^2 m\right)$.

For scenario (b) that $\ell_1 \leq \ell_2 < \ell$, we apply Lemma A.11.6 to bound $\mu_{\ell_1,\ell_2}^{(\ell),j}$. Specifically, we view

$$B_i = \frac{1}{\sqrt{m_i^{(\ell)}}} \left| \sigma''(\tilde{f}_i^{(\ell)}) \right| \left\| \frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell_1)}} \right\| \frac{\partial f_{\mathscr{S}_i^{(\ell)}}}{\partial \mathbf{w}^{(\ell_2)}}, \tag{A.36}$$

$$\mathbf{a}_i = \mathbf{w}_i^{(\ell)}. \tag{A.37}$$

Choosing $t = \log m$ and supposing B_i is fixed, then with probability at least $1 - 2\exp\left(-\log^2 m\right)$,

for some constant $K_{\ell_1,\ell_2}^{\ell,j} > 0$,

$$\begin{aligned}
&\left\| \sum_{i:f_i^{(\ell)} \in f_{\mathscr{S}_j^{(\ell+1)}}} \left(\sigma''(\tilde{f}_i^{(\ell)})\right)^2 \left\| \frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell_1)}} \right\|^2 \frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell_2)}} \left(\frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell_2)}} \right)^T \right\| \\
&\leq (K_{\ell_1,\ell_2}^{\ell,j})^2 \left(\max_i \|B_i\| \left(\sqrt{m_j^{(\ell+1)}} + \log m + R \right) + \max_i \|B_i\|_F \right)^2 \\
&\leq (K_{\ell_1,\ell_2}^{\ell,j})^2 \gamma_2^2 \left(\max_i \frac{1}{\sqrt{m_i^{(\ell)}}} \left\| \frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell_1)}} \right\| \left\| \frac{\partial f_{\mathscr{S}_i^{(\ell)}}}{\partial \mathbf{w}^{(\ell_2)}} \right\| \left(\sqrt{m_j^{(\ell+1)}} + \log m + R \right) \right. \\
&\quad \left. + \max_i \frac{1}{\sqrt{m_i^{(\ell)}}} \left\| \frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell_1)}} \right\| \left\| \frac{\partial f_{\mathscr{S}_i^{(\ell)}}}{\partial \mathbf{w}^{(\ell_2)}} \right\|_F \right)^2
\end{aligned}$$

By Eq. (A.35), with probability at least $1 - 2\exp\left(-c' \log^2 m\right) - \exp\left(-C_{\ell,\ell_1}^f \log^2 m\right)$ for some absolute constant $c' > 0$,

$$\left\| \frac{\partial \tilde{f}_i^{(\ell)}}{\partial \mathbf{w}^{(\ell_1)}} \right\| = \tilde{O}(R^\ell). \tag{A.38}$$

By Lemma A.2.1, with probability at least $1 - \exp\left(-C_{\ell,\ell_2}^f \log^2 m\right)$, $\left\| \partial f_{\mathscr{S}_i^{(\ell)}} / \partial \mathbf{w}^{(\ell_2)} \right\| = \tilde{O}\left(\max_{\ell_2+1 \leq p \leq \ell} \sqrt{m_i^{(\ell)}} / \sqrt{\underline{m}_p} \right)$ and $\left\| \partial f_{\mathscr{S}_i^{(\ell)}} / \partial \mathbf{w}^{(\ell_2)} \right\|_F = \tilde{O}\left(\sqrt{m_i^{(\ell)}} \right)$.

Combined them together, with probability at least $1 - m_i^{(\ell)} m_j^{(\ell+1)} \exp\left(-C_\ell^{\mathscr{P}} \log^2 m\right) - 2m_j^{(\ell+1)} \exp\left(-c' \log^2 m\right) - m_j^{(\ell+1)} \exp\left(-C_{\ell,\ell_1}^f \log^2 m\right) - m_j^{(\ell+1)} \exp\left(-C_{\ell,\ell_2}^f \log^2 m\right)$,

$$\begin{aligned}
\mu_{\ell_1,\ell_2}^{(\ell),j} &= O\left(\max\left(1/m_j^{(\ell+1)}, \max_{\ell_1+1 \leq p \leq \ell} 1/\underline{m}_p \right) (\log m + R)^{4\ell-2} \right) \\
&= \tilde{O}\left(\max\left(1/m_j^{(\ell+1)}, \max_{\ell_1+1 \leq p \leq \ell} 1/\underline{m}_p \right) R^{4\ell-2} \right).
\end{aligned}$$

Similarly we can find a constant $C_{\ell_1,\ell_2}^{(\ell),j} > 0$ such that the above bound holds with probability at

least $1-\exp\left(-C_{\ell_1,\ell_2}^{(\ell),j}\log^2 m\right)$.

For $\ell_1 \geq \ell_2$, we similarly have

$$\mu_{\ell_2,\ell_1}^{(\ell),j} = \tilde{O}\left(\max\left(1/m_j^{(\ell+1)}, \max_{\ell_2+1\leq p\leq \ell} 1/\underline{m}_p\right) R^{4\ell-2}\right),$$

with probability at least $1-\exp\left(-C_{\ell_2,\ell_1}^{(\ell),j}\log^2 m\right)$. □

Lemma A.11.10. *For any $0<\ell'\leq \ell \leq L-1$, given fixed matrices $U_1,...,U_{d_{\ell'}} \in \mathbb{R}^{u_1\times u_2}$, with probability at least $1-\sum_{k=1}^{\ell-\ell'+1} k(u_1+u_2)\exp\left(-\log^2 m/2\right)$*

$$\begin{aligned}\sum_{i=1}^{d_{\ell'}} U_i \frac{\partial \tilde{f}_j^{(\ell+1)}}{\partial \tilde{f}_i^{(\ell')}} &= O\left(\max_{i:f_i^{(\ell')}\in \mathscr{F}_{\mathscr{S}_j^{(\ell'+1)}}} \|U_i\|(\log m+R)^{\ell-\ell'+1}\right)\\ &= \tilde{O}\left(\max_{i:f_i^{(\ell')}\in \mathscr{F}_{\mathscr{S}_j^{(\ell'+1)}}} \|U_i\|\right).\end{aligned}$$

Proof. We prove the result by induction.

For the base case that $\ell=\ell'$,

$$\begin{aligned}\sum_{i=1}^{d_{\ell'}} U_i \frac{\partial \tilde{f}_j^{(\ell'+1)}}{\partial \tilde{f}_i^{(\ell')}} &= \sum_{i=1}^{d_{\ell'}} U_i \sigma'\left(\tilde{f}_i^{(\ell')}\right)\frac{\partial f_{\mathscr{S}_j^{(\ell'+1)}}}{\partial f_i^{(\ell')}}\frac{\partial \tilde{f}_j^{(\ell'+1)}}{\partial f_{\mathscr{S}_j^{(\ell'+1)}}}\\ &= \frac{1}{\sqrt{m_j^{(\ell'+1)}}}\sum_{i:f_i^{(\ell')}\in \mathscr{F}_{\mathscr{S}_j^{(\ell'+1)}}} U_i\sigma'\left(\tilde{f}_i^{(\ell')}\right)\left(\mathbf{w}_j^{(\ell'+1)}\right)_{\mathsf{id}_{\ell',i}^{\ell'+1,j}}.\end{aligned}$$

We view the above equation as a matrix Gaussian series with respect to $\mathbf{w}_j^{(\ell'+1)}$. Its matrix

variance $v^{(\ell')}$ can be bounded by

$$\begin{aligned} v^{(\ell')} &:= \frac{1}{m_j^{(\ell'+1)}} \left\| \sum_{i: f_i^{(\ell')} \in \mathscr{F}_{\mathscr{S}_j^{(\ell'+1)}}} U_i \sigma'\left(\tilde{f}_i^{(\ell')}\right) \right\|^2 \\ &\leq \max_{i: f_i^{(\ell')} \in \mathscr{F}_{\mathscr{S}_j^{(\ell'+1)}}} \gamma_1^2 \|U_i\|^2. \end{aligned}$$

Using Lemma A.11.8 and choosing $t = \log m \sqrt{v^{(\ell')}}$, we have with probability at least $1 - (u_1 + u_2)\exp\bigl(-\log^2 m/2\bigr)$,

$$\sum_{i=1}^{d_{\ell'}} U_i \frac{\partial \tilde{f}_j^{(\ell'+1)}}{\partial \tilde{f}_i^{(\ell')}} \leq (\log m + R)\sqrt{v^{(\ell')}} \leq \max_i (\log m + R)\gamma_1 \|U_i\| = O((\log m + R)\max_i \|U_i\|).$$

Suppose with probability at least $1 - \sum_{k=1}^{\ell-\ell'+1} k(u_1+u_2)\exp\bigl(-\log^2 m/2\bigr)$, for all $\ell' \leq k \leq \ell$,

$$\sum_{i=1}^{d_{\ell'}} U_i \frac{\partial \tilde{f}_j^{(k)}}{\partial \tilde{f}_i^{(\ell')}} = O\left(\max_{i: f_i^{(\ell')} \in \mathscr{F}_{\mathscr{S}_j^{(\ell'+1)}}} \|U_i\| (\log m + R)^{k-\ell'} \right).$$

Then when $k = \ell + 1$, we have

$$\begin{aligned}
\sum_{i=1}^{d_{\ell'}} U_i \frac{\partial \tilde{f}_j^{(\ell+1)}}{\partial \tilde{f}_i^{(\ell')}} &= \sum_{r=\ell'}^{\ell} \sum_{i=1}^{d_{\ell'}} U_i \frac{\partial f^{(r)}}{\partial \tilde{f}_i^{(\ell')}} \frac{\partial f_{\mathscr{S}_j^{(\ell+1)}}}{\partial f^{(r)}} \frac{\partial \tilde{f}_j^{(\ell+1)}}{\partial f_{\mathscr{S}_j^{(\ell+1)}}} \\
&= \sum_{r=\ell'}^{\ell} \sum_{s=1}^{d_r} \sum_{i=1}^{d_{\ell'}} U_i \frac{\partial f_s^{(r)}}{\partial \tilde{f}_i^{(\ell')}} \frac{\partial f_{\mathscr{S}_j^{(\ell+1)}}}{\partial f_s^{(r)}} \frac{\partial \tilde{f}_j^{(\ell+1)}}{\partial f_{\mathscr{S}_j^{(\ell+1)}}} \\
&= \sum_{r=\ell'}^{\ell} \sum_{s=1}^{d_r} \sum_{i=1}^{d_{\ell'}} U_i \frac{\partial \tilde{f}_s^{(r)}}{\partial \tilde{f}_i^{(\ell')}} \sigma'\left(\tilde{f}_s^{(r)}\right) \frac{\partial f_{\mathscr{S}_j^{(\ell+1)}}}{\partial \tilde{f}_s^{(r)}} \frac{\partial \tilde{f}_j^{(\ell+1)}}{\partial f_{\mathscr{S}_j^{(\ell+1)}}} \\
&= \sum_{r=\ell'}^{\ell} \sum_{s: f_s^{(r)} \in \mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}} \left(\sum_{i=1}^{d_{\ell'}} U_i \frac{\partial \tilde{f}_s^{(r)}}{\partial \tilde{f}_i^{(\ell')}} \right) \sigma'\left(\tilde{f}_s^{(r)}\right) \frac{1}{\sqrt{m_j^{(\ell+1)}}} \left(\mathbf{w}_j^{(\ell+1)}\right)_{\mathsf{id}_{r,s}^{\ell+1,j}}
\end{aligned}$$

For each $r \in \{\ell', ..., \ell\}$, we view

$$\sum_{s: f_s^{(r)} \in \mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}} \left(\sum_{i=1}^{d_{\ell'}} U_i \frac{\partial \tilde{f}_s^{(r)}}{\partial \tilde{f}_i^{(\ell')}} \right) \sigma'\left(\tilde{f}_s^{(r)}\right) \frac{1}{\sqrt{m_j^{(\ell+1)}}} \left(\mathbf{w}_j^{(\ell+1)}\right)_{\mathsf{id}_{r,s}^{\ell+1,j}}$$

as a matrix Gaussian series with respect to $\mathbf{w}_j^{(\ell+1)}$.

By the inductive hypothesis, for all r, its matrix variance can be bounded by

$$\begin{aligned}
v^{(r)} &:= \frac{1}{m_j^{(\ell+1)}} \left\| \sum_{s: f_s^{(r)} \in \mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}} \left(\sum_{i=1}^{d_{\ell'}} U_i \frac{\partial \tilde{f}_s^{(r)}}{\partial \tilde{f}_i^{(\ell')}} \right) \sigma'\left(\tilde{f}_s^{(r)}\right) \right\|^2 \\
&= O\left(\max_{i: f_i^{(\ell')} \in \mathscr{F}_{\mathscr{S}_j^{(\ell'+1)}}} \|U_i\|^2 (\log m + R)^{2r-2\ell'} \right).
\end{aligned}$$

Then we use Lemma A.11.8 and choose $t = \log m \sqrt{v^{(r)}}$. With probability at least

$1-(u_1+u_2)\exp\bigl(-\log^2 m/2\bigr)$,

$$\begin{aligned}
&\left\| \sum_{s:f_s^{(r)}\in\mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}} \left(\sum_{i=1}^{d_{\ell'}} U_i \frac{\partial \tilde{f}_s^{(r)}}{\partial \tilde{f}_i^{(\ell')}}\right) \sigma'\left(\tilde{f}_s^{(r)}\right) \frac{1}{\sqrt{m_j^{(\ell+1)}}} \left(\mathbf{w}_j^{(\ell+1)}\right)_{\mathsf{id}_{r,s}^{\ell+1,j}} \right\| \\
&\le (\log m+R)\sqrt{\nu^{(r)}} \\
&\le \max_{i:f_i^{(\ell')}\in\mathscr{F}_{\mathscr{S}_j^{(\ell'+1)}}} (\log m+R)\gamma_1\|U_i\| \\
&= O\left(\max_{i:f_i^{(\ell')}\in\mathscr{F}_{\mathscr{S}_j^{(\ell'+1)}}} \|U_i\|(\log m+R)^{r-\ell'+1} \right).
\end{aligned}$$

We apply union bound over indices $r=\ell',...,\ell$ and add the probability from the induction hypothesis. With probability at least $1-\sum_{k=1}^{\ell-\ell'+1} k(u_1+u_2)\exp\bigl(-\log^2 m/2\bigr)$,

$$\begin{aligned}
\left\| \sum_{i=1}^{d_{\ell'}} U_i \frac{\partial \tilde{f}_j^{(\ell+1)}}{\partial \tilde{f}_i^{(\ell')}} \right\| &\le \sum_{r=\ell'+1}^{\ell} \left\| \sum_{s:f_s^{(r)}\in\mathscr{F}_{\mathscr{S}_j^{(\ell+1)}}} \left(\sum_{i=1}^{d_{\ell'}} U_i \frac{\partial \tilde{f}_s^{(r)}}{\partial \tilde{f}_i^{(\ell')}}\right) \sigma'\left(\tilde{f}_s^{(r)}\right) \frac{1}{\sqrt{m_j^{(\ell+1)}}} \left(\mathbf{w}_j^{(\ell+1)}\right)_{\mathsf{id}_{r,s}^{\ell+1,j}} \right\| \\
&= O\left(\max_{i:f_i^{(\ell')}\in\mathscr{F}_{\mathscr{S}_j^{(\ell'+1)}}} \|U_i\|(\log m+R)^{\ell-\ell'+1} \right) \\
&= \tilde{O}\left(\max_{i:f_i^{(\ell')}\in\mathscr{F}_{\mathscr{S}_j^{(\ell'+1)}}} \|U_i\|R^{\ell-\ell'+1} \right).
\end{aligned}$$

Then we finish the induction step which completes the proof. □

Appendix B

Chapter 3 Supplementary

B.1 Derivation of NQM

We will derive the NQM that approximate the two-layer fully connected ReLU activated neural networks based on Eq. (3.2).

The first derivative of f can be computed by:

$$\frac{\partial f}{\partial \mathbf{u}_i} = \frac{1}{\sqrt{md}} v_i \mathbb{1}_{\{\mathbf{u}_i^T \boldsymbol{x} \geq 0\}} \boldsymbol{x}^T, \quad \frac{\partial f}{\partial v_i} = \frac{1}{\sqrt{m}} \sigma\left(\frac{1}{\sqrt{d}} \mathbf{u}_i^T \boldsymbol{x}\right), \quad \forall i \in [m].$$

And each entry of the Hessian of f, i.e., H_f, can be computed by

$$\frac{\partial^2 f}{\partial \mathbf{u}_i^2} = \mathbf{0}, \ \frac{\partial^2 f}{\partial v_i^2} = 0, \quad \frac{\partial^2 f}{\partial \mathbf{u}_i v_i} = \frac{1}{\sqrt{md}} \mathbb{1}_{\{\mathbf{u}_i^T \boldsymbol{x} \geq 0\}} \boldsymbol{x}^T, \quad \forall i \in [m].$$

Now we get f_{quad} taking the following form

$$\begin{aligned} \textbf{NQM}: \ f_{\text{quad}}(\mathbf{u}, \mathbf{v}; \boldsymbol{x}) = f(\mathbf{u}_0, \mathbf{v}_0; \boldsymbol{x}) &+ \frac{1}{\sqrt{md}} \sum_{i=1}^{m} (\mathbf{u}_i - \mathbf{u}_{0,i})^T \boldsymbol{x} \mathbb{1}_{\left\{\mathbf{u}_{0,i}^T \boldsymbol{x} \geq 0\right\}} v_{0,i} \\ &+ \frac{1}{\sqrt{m}} \sum_{i=1}^{m} (v_i - v_{0,i}) \sigma\left(\frac{1}{\sqrt{d}} \mathbf{u}_{0,i}^T \boldsymbol{x}\right) \\ &+ \frac{1}{\sqrt{md}} \sum_{i=1}^{m} (\mathbf{u}_i - \mathbf{u}_{0,i})^T \boldsymbol{x} \mathbb{1}_{\left\{\mathbf{u}_{0,i}^T \boldsymbol{x} \geq 0\right\}} (v_i - v_{0,i}). \end{aligned} \tag{B.1}$$

B.2 Derivation of dynamics equations

For simplicity of notation, we denote f_{quad} by g. Note that at initialization, the first and second derivatives of f with respect to parameters are the same as those of g.

Single training example

The NQM can be equivalently written as:

$$\begin{aligned} g(\mathbf{u},\mathbf{v};\boldsymbol{x}) = g(\mathbf{u}_0,\mathbf{v}_0;\boldsymbol{x}) &+ \left\langle \mathbf{u}-\mathbf{u}_0,\, \nabla_{\mathbf{u}} g(\mathbf{u},\mathbf{v};\boldsymbol{x})\Big|_{\mathbf{u}=\mathbf{u}_0,\mathbf{v}=\mathbf{v}_0} \right\rangle \\ &+ \left\langle \mathbf{v}-\mathbf{v}_0,\, \nabla_{\mathbf{v}} g(\mathbf{u},\mathbf{v};\boldsymbol{x})\Big|_{\mathbf{u}=\mathbf{u}_0,\mathbf{v}=\mathbf{v}_0} \right\rangle \\ &+ \left\langle \mathbf{u}-\mathbf{u}_0,\, \frac{\partial^2 g(\mathbf{u},\mathbf{v};\boldsymbol{x})}{\partial \mathbf{u}\partial \mathbf{v}}\Big|_{\mathbf{u}=\mathbf{u}_0,\mathbf{v}=\mathbf{v}_0} (\mathbf{v}-\mathbf{v}_0) \right\rangle, \end{aligned}$$

since $\frac{\partial^2 g}{\partial \mathbf{u}^2} = 0$ and $\frac{\partial^2 g}{\partial \mathbf{v}^2} = 0$.

And the tangent kernel $\lambda(\mathbf{u},\mathbf{v};\boldsymbol{x})$ takes the form

$$\begin{aligned} \lambda(\mathbf{u},\mathbf{v};\boldsymbol{x}) = &\left\| \nabla_{\mathbf{u}} g(\mathbf{u},\mathbf{v};\boldsymbol{x})\Big|_{\mathbf{u}=\mathbf{u}_0,\mathbf{v}=\mathbf{v}_0} + \frac{\partial^2 g(\mathbf{u},\mathbf{v};\boldsymbol{x})}{\partial \mathbf{u}\partial \mathbf{v}}\Big|_{\mathbf{u}=\mathbf{u}_0} (\mathbf{v}-\mathbf{v}_0) \right\|_F^2 \\ &+ \left\| \nabla_{\mathbf{v}} g(\mathbf{u},\mathbf{v};\boldsymbol{x})\Big|_{\mathbf{u}=\mathbf{u}_0,\mathbf{v}=\mathbf{v}_0} + (\mathbf{u}-\mathbf{u}_0)^T \frac{\partial^2 g(\mathbf{u},\mathbf{v};\boldsymbol{x})}{\partial \mathbf{u}\partial \mathbf{v}}\Big|_{\mathbf{u}=\mathbf{u}_0,\mathbf{v}=\mathbf{v}_0} \right\|^2. \end{aligned}$$

Here

$$\begin{aligned} \nabla_{\mathbf{u}_i} g(\mathbf{u},\mathbf{v};\boldsymbol{x})\Big|_{\mathbf{u}=\mathbf{u}_0,\mathbf{v}=\mathbf{v}_0} &= \frac{1}{\sqrt{md}} \sum_{i=1}^{m} v_{0,i} \mathbb{1}_{\left\{\mathbf{u}_{0,i}^T \boldsymbol{x} \geq 0\right\}} \boldsymbol{x}, \quad \forall i \in [m], \\ \nabla_{\mathbf{v}} g(\mathbf{u},\mathbf{v};\boldsymbol{x})\Big|_{\mathbf{u}=\mathbf{u}_0,\mathbf{v}=\mathbf{v}_0} &= \frac{1}{\sqrt{md}} \sigma\left(\mathbf{u}_0^T \boldsymbol{x}\right). \end{aligned}$$

In the following, we will consider the dynamics of g and λ with GD, hence for simplicity

of notations, we denote

$$\nabla_{\mathbf{u}} g(0) := \nabla_{\mathbf{u}} g(\mathbf{u},\mathbf{v};\boldsymbol{x})\Big|_{\mathbf{u}=\mathbf{u}_0,\mathbf{v}=\mathbf{v}_0},$$
$$\nabla_{\mathbf{v}} g(0) := \nabla_{\mathbf{v}} g(\mathbf{u},\mathbf{v};\boldsymbol{x})\Big|_{\mathbf{u}=\mathbf{u}_0,\mathbf{v}=\mathbf{v}_0},$$
$$\frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}} := \frac{\partial^2 g(\mathbf{u},\mathbf{v};\boldsymbol{x})}{\partial \mathbf{u}\partial \mathbf{v}}\Big|_{\mathbf{u}=\mathbf{u}_0,\mathbf{v}=\mathbf{v}_0}.$$

By gradient descent with learning rate η, at iteration t, we have the update equations for weights $\mathbf{u}$ and $\mathbf{v}$:

$$\mathbf{u}(t+1) = \mathbf{u}(t) - \eta(g(t)-y)\left(\nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}}(\mathbf{v}(t)-\mathbf{v}(0))\right),$$
$$\mathbf{v}(t+1) = \mathbf{v}(t) - \eta(g(t)-y)\left(\nabla_{\mathbf{v}} g(0) + (\mathbf{u}(t)-\mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}}\right).$$

Then we plug them in the expression of $\lambda(t+1)$ and we get

$$\begin{aligned}
&\lambda(t+1)\\
&= \left\|\nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}}(\mathbf{v}(t+1)-\mathbf{v}(0))\right\|_F^2 + \left\|\nabla_{\mathbf{v}} g(0) + (\mathbf{u}(t+1)-\mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}}\right\|^2\\
&= \left\|\nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}}\left(\mathbf{v}(t) - \eta(g(t)-y)\left(\nabla_{\mathbf{v}} g(0) + (\mathbf{u}(t)-\mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}}\right) - \mathbf{v}(0)\right)\right\|_F^2\\
&\quad + \left\|\nabla_{\mathbf{v}} g(0) + \left(\mathbf{u}(t) - \eta(g(t)-y)\left(\nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}}(\mathbf{v}(t)-\mathbf{v}(0))\right) - \mathbf{u}(0)\right)^T \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}}\right\|^2.
\end{aligned}$$

Expanding the terms in the norms we have

$$\begin{aligned}
\lambda(t+1) = \lambda(t) &+ \eta^2(g(t)-y)^2 \left\| \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}} \left(\nabla_{\mathbf{v}} g(0) + (\mathbf{u}(t)-\mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}} \right) \right\|_F^2 \\
&+ \eta^2(g(t)-y)^2 \left\| \left(\nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}} (\mathbf{v}(t)-\mathbf{v}(0)) \right)^T \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}} \right\|^2 \\
&- 2\eta(g(t)-y) \left\langle \nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}} (\mathbf{v}(t)-\mathbf{v}(0)), \right. \\
&\qquad \left. \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}} \left(\nabla_{\mathbf{v}} g(0) + (\mathbf{u}(t)-\mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}} \right) \right\rangle \\
&- 2\eta(g(t)-y) \left\langle \nabla_{\mathbf{v}} g(0) + (\mathbf{u}(t)-\mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}}, \right. \\
&\qquad \left. \left(\nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}} (\mathbf{v}(t)-\mathbf{v}(0)) \right)^T \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}} \right\rangle.
\end{aligned}$$

Due to the structure of $\frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}}$, we have

$$\begin{aligned}
\left\| \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}} \left(\nabla_{\mathbf{v}} g(0) + (\mathbf{u}(t)-\mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}} \right) \right\|_F^2 &= \frac{\|\boldsymbol{x}\|^2}{md} \left\| \nabla_{\mathbf{v}} g(0) + (\mathbf{u}(t)-\mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}} \right\|^2 \\
&= \frac{\|\boldsymbol{x}\|^2}{md} \|\nabla_{\mathbf{v}} g(t)\|^2,
\end{aligned}$$

and

$$\begin{aligned}
\left\| \left(\nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}} (\mathbf{v}(t)-\mathbf{v}(0)) \right)^T \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}} \right\|^2 &= \frac{\|\boldsymbol{x}\|^2}{md} \left\| \nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u}\partial \mathbf{v}} (\mathbf{v}(t)-\mathbf{v}(0)) \right\|_F^2 \\
&= \frac{\|\boldsymbol{x}\|^2}{md} \|\nabla_{\mathbf{u}} g(t)\|_F^2.
\end{aligned}$$

Furthermore,

$$\begin{aligned}
&\left\langle \nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} (\mathbf{v}(t) - \mathbf{v}(0)), \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} \left(\nabla_{\mathbf{v}} g(0) + (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right) \right\rangle \\
&= \frac{\|\boldsymbol{x}\|^2}{md} \langle \mathbf{v}(t) - \mathbf{v}(0), \nabla_{\mathbf{v}} g(0) \rangle + \frac{\|\boldsymbol{x}\|^2}{md} \langle \nabla_{\mathbf{u}} g(0), \mathbf{u}(t) - \mathbf{u}(0) \rangle + \left\langle \nabla_{\mathbf{u}} g(0), \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} \nabla_{\mathbf{v}} g(0) \right\rangle \\
&\quad + \left\langle \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} (\mathbf{v}(t) - \mathbf{v}(0)), \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right\rangle \\
&= \frac{\|\boldsymbol{x}\|^2}{md} \langle \mathbf{v}(t) - \mathbf{v}(0), \nabla_{\mathbf{v}} g(0) \rangle + \frac{\|\boldsymbol{x}\|^2}{md} \langle \nabla_{\mathbf{u}} g(0), \mathbf{u}(t) - \mathbf{u}(0) \rangle + g(0) \\
&\quad + \frac{\|\boldsymbol{x}\|^2}{md} \left\langle \mathbf{v}(t) - \mathbf{v}(0), \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} (\mathbf{u}(t) - \mathbf{u}(0))^T \right\rangle \\
&= g(t) \|\boldsymbol{x}\|^2 / md. \qquad \text{(B.2)}
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
&\left\langle \nabla_{\mathbf{v}} g(0) + (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}}, \left(\nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} (\mathbf{v}(t) - \mathbf{v}(0)) \right)^T \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right\rangle \\
&\quad = g(t) \|\boldsymbol{x}\|^2 / md.
\end{aligned}$$

As a result,

$$\begin{aligned}
\lambda(t+1) &= \lambda(t) + \frac{\|\boldsymbol{x}\|^2}{md} \eta^2 (g(t) - y)^2 \lambda(t) - \frac{4\|\boldsymbol{x}\|^2}{md} \eta (g(t) - y) g(t) \\
&= \lambda(t) + \eta \frac{\|\boldsymbol{x}\|^2}{md} (g(t) - y)^2 \left(\eta \lambda(t) - 4 \frac{g(t)}{g(t) - y} \right).
\end{aligned}$$

For g, we plug the update equations for $\mathbf{u}$ and $\mathbf{v}$ in the expression of $g(t+1)$ and we can

get

$$
\begin{aligned}
g(t+1) &= g(0) + \langle \mathbf{u}(t+1) - \mathbf{u}(0), \nabla_{\mathbf{u}} g(0) \rangle + \langle \mathbf{v}(t+1) - \mathbf{v}(0), \nabla_{\mathbf{v}} g(0) \rangle \\
&\quad + \left\langle \mathbf{u}(t+1) - \mathbf{u}(0), \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} (\mathbf{v}(t+1) - \mathbf{v}(0) \right\rangle \\
&= g(0) + \left\langle \mathbf{u}(t) - \eta(g(t)-y) \left(\nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} (\mathbf{v}(t) - \mathbf{v}(0)) \right) - \mathbf{u}(0), \nabla_{\mathbf{u}} g(0) \right\rangle \\
&\quad + \left\langle \mathbf{v}(t) - \eta(g(t)-y) \left(\nabla_{\mathbf{v}} g(0) + (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right) - \mathbf{v}(0), \nabla_{\mathbf{v}} g(0) \right\rangle \\
&\quad + \left\langle \mathbf{u}(t) - \eta(g(t)-y) \left(\nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} (\mathbf{v}(t) - \mathbf{v}(0)) \right) - \mathbf{u}(0) \right. , \\
&\quad \left. \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} \left(\mathbf{v}(t) - \eta(g(t)-y) \left(\nabla_{\mathbf{v}} g(0) + (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right) - \mathbf{v}(0) \right) \right\rangle .
\end{aligned}
$$

Writing out the inner product we have

$$
\begin{aligned}
g(t+1) &= g(t) - \eta(g(t)-y) \left\langle \nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} (\mathbf{v}(t) - \mathbf{v}(0)), \nabla_{\mathbf{u}} g(0) \right\rangle \\
&\quad - \eta(g(t)-y) \left\langle \nabla_{\mathbf{v}} g(0) + (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}}, \nabla_{\mathbf{v}} g(0) \right\rangle \\
&\quad + \eta^2 (g(t)-y)^2 \left\langle \nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} (\mathbf{v}(t) - \mathbf{v}(0)), \right. \\
&\qquad \left. \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} \left(\nabla_{\mathbf{v}} g(0) + (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right) \right\rangle \\
&\quad - \eta(g(t)-y) \left\langle \mathbf{u}(t) - \mathbf{u}(0), \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} \left(\nabla_{\mathbf{v}} g(0) + (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right) \right\rangle \\
&\quad - \eta(g(t)-y) \left\langle \nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} (\mathbf{v}(t) - \mathbf{v}(0)), \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} (\mathbf{v}(t) - \mathbf{v}(0)) \right\rangle .
\end{aligned}
$$

Using the definition of $\lambda(t)$ and Eq.(B.2) we have

$$\begin{aligned} g(t+1) &= g(t) - \eta(g(t)-y)\lambda(t) \\ &\quad + \eta^2(g(t)-y)^2 \left\langle \nabla_{\mathbf{u}} g(0) + \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}}(\mathbf{v}(t)-\mathbf{v}(0)), \right. \\ &\qquad \left. \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} \left(\nabla_{\mathbf{v}} g(0) + (\mathbf{u}(t)-\mathbf{u}(0))^T \frac{\partial^2 g(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right) \right\rangle \\ &= g(t) - \eta(g(t)-y)\lambda(t) + \frac{\|\boldsymbol{x}\|^2}{md}\eta^2(g(t)-y)^2 g(t) \end{aligned}$$

Therefore,

$$g(t+1)-y = \left(1 - \eta\lambda(t) + \frac{\|\boldsymbol{x}\|^2}{md}\eta^2(g(t)-y)g(t)\right)(g(t)-y).$$

Multiple training examples

We follow the similar notation on the first and second order derivative of g with Appendix B.2. Specifically, for $k \in [n]$, we denote

$$\begin{aligned} \nabla_{\mathbf{u}} g_k(0) &:= \nabla_{\mathbf{u}} g(\mathbf{u}, \mathbf{v}; x_k)\Big|_{\mathbf{u}=\mathbf{u}_0, \mathbf{v}=\mathbf{v}_0}, \\ \nabla_{\mathbf{v}} g_k(0) &:= \nabla_{\mathbf{v}} g(\mathbf{u}, \mathbf{v}; x_k)\Big|_{\mathbf{u}=\mathbf{u}_0, \mathbf{v}=\mathbf{v}_0}, \\ \frac{\partial^2 g_k(0)}{\partial \mathbf{u} \partial \mathbf{v}} &:= \frac{\partial^2 g(\mathbf{u}, \mathbf{v}; x_k)}{\partial \mathbf{u} \partial \mathbf{v}}\Big|_{\mathbf{u}=\mathbf{u}_0, \mathbf{v}=\mathbf{v}_0}. \end{aligned}$$

By GD with learning rate η, we have the update equations for weights $\mathbf{u}$ and $\mathbf{v}$ at iteration t:

$$\begin{aligned} \mathbf{u}(t+1) &= \mathbf{u}(t) - \eta \sum_{k=1}^{n} (g_k(t)-y_k)\left(\nabla_{\mathbf{u}} g_k(0) + \frac{\partial^2 g_k(0)}{\partial \mathbf{u} \partial \mathbf{v}}(\mathbf{v}(t)-\mathbf{v}(0))\right), \\ \mathbf{v}(t+1) &= \mathbf{v}(t) - \eta \sum_{k=1}^{n} (g_k(t)-y_k)\left(\nabla_{\mathbf{v}} g_k(0) + (\mathbf{u}(t)-\mathbf{u}(0))^T \frac{\partial^2 g_k(0)}{\partial \mathbf{u} \partial \mathbf{v}}\right). \end{aligned}$$

We consider the evolution of $K(t)$ first. Plugging in the expression of $\mathbf{u}(t)$ and $\mathbf{v}(t+1)$ we have

$$
\begin{aligned}
K_{i,j}(t+1) &= \left\langle \nabla_{\mathbf{u}} g_i(0) + \frac{\partial^2 g_i(0)}{\partial \mathbf{u} \partial \mathbf{v}}(\mathbf{v}(t+1) - \mathbf{v}(0)), \nabla_{\mathbf{u}} g_j(0) + \frac{\partial^2 g_j(0)}{\partial \mathbf{u} \partial \mathbf{v}}(\mathbf{v}(t+1) - \mathbf{v}(0)) \right\rangle \\
&\quad + \left\langle \nabla_{\mathbf{v}} g_i(0) + (\mathbf{u}(t+1) - \mathbf{u}(0))^T \frac{\partial^2 g_i(0)}{\partial \mathbf{u} \partial \mathbf{v}}, \nabla_{\mathbf{v}} g_j(0) + (\mathbf{u}(t+1) - \mathbf{u}(0))^T \frac{\partial^2 g_j(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right\rangle \\
&= K_{i,j}(t) - \left\langle \eta \frac{\partial^2 g_i(0)}{\partial \mathbf{u} \partial \mathbf{v}} \sum_{k=1}^{n} (g_k(t) - y_k) \left(\nabla_{\mathbf{v}} g_k(0) + (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g_k(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right), \right. \\
&\qquad \left. \nabla_{\mathbf{u}} g_j(0) + \frac{\partial^2 g_j(0)}{\partial \mathbf{u} \partial \mathbf{v}}(\mathbf{v}(t) - \mathbf{v}(0)) \right\rangle \\
&\quad - \left\langle \eta \frac{\partial^2 g_j(0)}{\partial \mathbf{u} \partial \mathbf{v}} \sum_{k=1}^{n} (g_k(t) - y_k) \left(\nabla_{\mathbf{v}} g_k(0) + (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g_k(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right), \right. \\
&\qquad \left. \nabla_{\mathbf{u}} g_i(0) + \frac{\partial^2 g_i(0)}{\partial \mathbf{u} \partial \mathbf{v}}(\mathbf{v}(t) - \mathbf{v}(0)) \right\rangle \\
&\quad + \left\langle \eta \frac{\partial^2 g_i(0)}{\partial \mathbf{u} \partial \mathbf{v}} \sum_{k=1}^{n} (g_k(t) - y_k) \left(\nabla_{\mathbf{v}} g_k(0) + (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g_k(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right), \right. \\
&\qquad \left. \eta \frac{\partial^2 g_j(0)}{\partial \mathbf{u} \partial \mathbf{v}} \sum_{k=1}^{n} (g_k(t) - y_k) \left(\nabla_{\mathbf{v}} g_k(0) + (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g_k(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right) \right\rangle \\
&\quad - \left\langle \eta \frac{\partial^2 g_j(0)}{\partial \mathbf{u} \partial \mathbf{v}} \sum_{k=1}^{n} (g_k(t) - y_k) \left(\nabla_{\mathbf{u}} g_k(0) + \frac{\partial^2 g_k(0)}{\partial \mathbf{u} \partial \mathbf{v}}(\mathbf{v}(t) - \mathbf{v}(0)) \right), \right. \\
&\qquad \left. \nabla_{\mathbf{v}} g_i(0) + (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g_i(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right\rangle \\
&\quad - \left\langle \eta \frac{\partial^2 g_i(0)}{\partial \mathbf{u} \partial \mathbf{v}} \sum_{k=1}^{n} (g_k(t) - y_k) \left(\nabla_{\mathbf{u}} g_k(0) + \frac{\partial^2 g_k(0)}{\partial \mathbf{u} \partial \mathbf{v}}(\mathbf{v}(t) - \mathbf{v}(0)) \right), \right. \\
&\qquad \left. \nabla_{\mathbf{v}} g_j(0) + (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g_j(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right\rangle \\
&\quad + \left\langle \eta \frac{\partial^2 g_i(0)}{\partial \mathbf{u} \partial \mathbf{v}} \sum_{k=1}^{n} (g_k(t) - y_k) \left(\nabla_{\mathbf{u}} g_k(0) + \frac{\partial^2 g_k(0)}{\partial \mathbf{u} \partial \mathbf{v}}(\mathbf{v}(t) - \mathbf{v}(0)) \right), \right. \\
&\qquad \left. \eta \frac{\partial^2 g_j(0)}{\partial \mathbf{u} \partial \mathbf{v}} \sum_{k=1}^{n} (g_k(t) - y_k) \left(\nabla_{\mathbf{u}} g_k(0) + \frac{\partial^2 g_k(0)}{\partial \mathbf{u} \partial \mathbf{v}}(\mathbf{v}(t) - \mathbf{v}(0)) \right) \right\rangle.
\end{aligned}
$$

We separate the data into two sets according to their sign:

$$\mathscr{S}_+ := \{i : x_i \geq 0, i \in [n]\}, \quad \mathscr{S}_- := \{i : x_i < 0, i \in [n]\}.$$

We consider two scenarios: (1) x_i and x_j have different signs; (2) x_i and x_j have the same sign.

(1)

With simple calculation, we get if x_i and x_j have different signs, i.e., $i \in \mathscr{S}_+, j \in \mathscr{S}_-$ or $i \in \mathscr{S}_-, j \in \mathscr{S}_+$,

$$\frac{\partial^2 g_i(0)}{\partial \mathbf{u} \partial \mathbf{v}} \frac{\partial^2 g_j(0)}{\partial \mathbf{u} \partial \mathbf{v}} = 0, \quad \frac{\partial^2 g_i(0)}{\partial \mathbf{u} \partial \mathbf{v}} \nabla_{\mathbf{u}} g_j(0) = 0, \quad \frac{\partial^2 g_i(0)}{\partial \mathbf{u} \partial \mathbf{v}} \nabla_{\mathbf{v}} g_j(0) = 0.$$

Without lose of generality, we assume $i \in \mathscr{S}_+$, $j \in \mathscr{S}_-$. Then we have

$$K_{i,j}(t+1) = K_{i,j}(t).$$

(2)

If x_i and x_j have the same sign, i.e., $i, j \in \mathscr{S}_+$ or $i, j \in \mathscr{S}_-$,

$$\frac{\partial^2 g_i(0)}{\partial \mathbf{u} \partial \mathbf{v}} \frac{\partial^2 g_j(0)}{\partial \mathbf{u} \partial \mathbf{v}} = \frac{1}{\sqrt{m}} \frac{\partial^2 g_i(0)}{\partial \mathbf{u} \partial \mathbf{v}} x_j, \quad \frac{\partial^2 g_i(0)}{\partial \mathbf{u} \partial \mathbf{v}} \nabla_{\mathbf{u}} g_j(0) = \frac{1}{\sqrt{m}} \nabla_{\mathbf{u}} g_i(0) x_j,$$
$$\text{and} \quad \frac{\partial^2 g_i(0)}{\partial \mathbf{u} \partial \mathbf{v}} \nabla_{\mathbf{v}} g_j(0) = \frac{1}{\sqrt{m}} \nabla_{\mathbf{v}} g_i(0) x_j.$$

For $i, j \in \mathscr{S}_+$, we have

$$
\begin{aligned}
&K_{i,j}(t+1) \\
&= K_{i,j}(t) - \frac{2\eta}{\sqrt{m}} \sum_{k \in \mathscr{S}_+} (g_k(t) - y_k) x_i \left\langle \nabla_{\mathbf{v}} g_k(0) + (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g_k(0)}{\partial \mathbf{u} \partial \mathbf{v}} , \right. \\
&\qquad\qquad \left. \nabla_{\mathbf{u}} g_j(0) + \frac{\partial^2 g_j(0)}{\partial \mathbf{u} \partial \mathbf{v}} (\mathbf{v}(t) - \mathbf{v}(0)) \right\rangle \\
&\quad - \frac{2\eta}{\sqrt{m}} \sum_{k \in \mathscr{S}_+} (g_k(t) - y_k) x_i \left\langle \nabla_{\mathbf{u}} g_k(0) + \frac{\partial^2 g_k(0)}{\partial \mathbf{u} \partial \mathbf{v}} (\mathbf{v}(t) - \mathbf{v}(0)), \right. \\
&\qquad\qquad \left. \nabla_{\mathbf{v}} g_j(0) + (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g_j(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right\rangle \\
&\quad + \frac{\eta^2}{m} x_i x_j \left\| \sum_{k \in \mathscr{S}_+} (g_k(t) - y_k) \left(\nabla_{\mathbf{v}} g_k(0) + (\mathbf{u}(t) - \mathbf{u}(0))^T \frac{\partial^2 g_k(0)}{\partial \mathbf{u} \partial \mathbf{v}} \right) \right\|^2 \\
&\quad + \frac{\eta^2}{m} x_i x_j \left\| \sum_{k \in \mathscr{S}_+} (g_k(t) - y_k) \left(\nabla_{\mathbf{u}} g_k(0) + \frac{\partial^2 g_k(0)}{\partial \mathbf{u} \partial \mathbf{v}} (\mathbf{v}(t) - \mathbf{v}(0)) \right) \right\|^2 \\
&= K_{i,j}(t) - \frac{4\eta}{m} x_i x_j \sum_{k \in \mathscr{S}_+} (g_k(t) - y_k) g_k(t) + \frac{\eta^2}{m} x_i x_j \left((\mathbf{g}(t) - \boldsymbol{y}) \odot \boldsymbol{m}_+ \right)^T K(t) \left((\mathbf{g}(t) - \boldsymbol{y}) \odot \boldsymbol{m}_+ \right) \\
&= K_{i,j}(t) - \frac{4\eta}{m} x_i x_j \left((\mathbf{g}(t) - \boldsymbol{y}) \odot \boldsymbol{m}_+ \right)^T \left(\mathbf{g}(t) \odot \boldsymbol{m}_+ \right) \\
&\quad + \frac{\eta^2}{m} x_i x_j \left((\mathbf{g}(t) - \boldsymbol{y}) \odot \boldsymbol{m}_+ \right)^T K(t) \left((\mathbf{g}(t) - \boldsymbol{y}) \odot \boldsymbol{m}_+ \right).
\end{aligned}
$$

Similarly, for $i, j \in \mathscr{S}_-$, we have

$$
\begin{aligned}
K_{i,j}(t+1) = K_{i,j}(t) &- \frac{4\eta}{m} x_i x_j \left((\mathbf{g}(t) - \boldsymbol{y}) \odot \boldsymbol{m}_- \right)^T \left(\mathbf{g}(t) \odot \boldsymbol{m}_- \right) \\
&+ \frac{\eta^2}{m} x_i x_j \left((\mathbf{g}(t) - \boldsymbol{y}) \odot \boldsymbol{m}_- \right)^T K(t) \left((\mathbf{g}(t) - \boldsymbol{y}) \odot \boldsymbol{m}_- \right).
\end{aligned}
$$

Combining the results together, we have

$$
\begin{aligned}
K(t+1) = K(t) &+ \frac{\eta^2}{m}\left((\mathbf{g}(t)-\boldsymbol{y})\odot \boldsymbol{m}_+\right)^T K(t)\left((\mathbf{g}(t)-\boldsymbol{y})\odot \boldsymbol{m}_+\right)\boldsymbol{p}_1\boldsymbol{p}_1^T \\
&+ \frac{\eta^2}{m}\left((\mathbf{g}(t)-\boldsymbol{y})\odot \boldsymbol{m}_-\right)^T K(t)\left((\mathbf{g}(t)-\boldsymbol{y})\odot \boldsymbol{m}_-\right)\boldsymbol{p}_2\boldsymbol{p}_2^T \\
&- \frac{4\eta}{m}\left((\mathbf{g}(t)-\boldsymbol{y})\odot \boldsymbol{m}_+\right)^T \left(\mathbf{g}(t)\odot \boldsymbol{m}_+\right)\boldsymbol{p}_1\boldsymbol{p}_1^T \\
&- \frac{4\eta}{m}\left((\mathbf{g}(t)-\boldsymbol{y})\odot \boldsymbol{m}_-\right)^T \left(\mathbf{g}(t)\odot \boldsymbol{m}_-\right)\boldsymbol{p}_2\boldsymbol{p}_2^T.
\end{aligned}
$$

Now we derive the evolution of $\mathbf{g}(t)-\boldsymbol{y}$. Suppose $i \in \mathscr{S}_+$. Then we have

$$
\begin{aligned}
& g_i(t+1) \\
&= g_i(0) + \langle \mathbf{u}(t+1)-\mathbf{u}(0), \nabla_{\mathbf{u}} g_i(0)\rangle + \langle \mathbf{v}(t+1)-\mathbf{v}(0), \nabla_{\mathbf{v}} g_i(0)\rangle \\
&\quad + \left\langle \mathbf{u}(t+1)-\mathbf{u}(0), \frac{\partial^2 g_i(0)}{\partial \mathbf{u}\partial \mathbf{v}}(\mathbf{v}(t+1)-\mathbf{v}(0)\right\rangle \\
&= g_i(t) - \eta\left\langle \sum_{k=1}^n (g_k(t)-y_k)\left(\nabla_{\mathbf{u}} g_k(0) + \frac{\partial^2 g_k(0)}{\partial \mathbf{u}\partial \mathbf{v}}(\mathbf{v}(t)-\mathbf{v}(0))\right), \nabla_{\mathbf{u}} g_i(0)\right\rangle \\
&\quad - \eta\left\langle \sum_{k=1}^n (g_k(t)-y_k)\left(\nabla_{\mathbf{v}} g_k(0) + (\mathbf{u}(t)-\mathbf{u}(0))^T\frac{\partial^2 g_k(0)}{\partial \mathbf{u}\partial \mathbf{v}}\right), \nabla_{\mathbf{v}} g_i(0)\right\rangle \\
&\quad - \eta\left\langle \sum_{k=1}^n (g_k(t)-y_k)\left(\nabla_{\mathbf{u}} g_k(0) + \frac{\partial^2 g_k(0)}{\partial \mathbf{u}\partial \mathbf{v}}(\mathbf{v}(t)-\mathbf{v}(0))\right), \frac{\partial^2 g_i(0)}{\partial \mathbf{u}\partial \mathbf{v}}(\mathbf{v}(t)-\mathbf{v}(0)\right\rangle \\
&\quad - \eta\left\langle \sum_{k=1}^n (g_k(t)-y_k)\left(\nabla_{\mathbf{v}} g_k(0) + (\mathbf{u}(t)-\mathbf{u}(0))^T\frac{\partial^2 g_k(0)}{\partial \mathbf{u}\partial \mathbf{v}}\right), (\mathbf{u}(t)-\mathbf{u}(0)^T\frac{\partial^2 g_i(0)}{\partial \mathbf{u}\partial \mathbf{v}}\right\rangle \\
&\quad + \eta^2\left\langle \sum_{k=1}^n (g_k(t)-y_k)\left(\nabla_{\mathbf{u}} g_k(0) + \frac{\partial^2 g_k(0)}{\partial \mathbf{u}\partial \mathbf{v}}(\mathbf{v}(t)-\mathbf{v}(0))\right),\right. \\
&\qquad \left. \frac{\partial^2 g_i(0)}{\partial \mathbf{u}\partial \mathbf{v}}\sum_{k=1}^n (g_k(t)-y_k)\left(\nabla_{\mathbf{v}} g_k(0) + (\mathbf{u}(t)-\mathbf{u}(0))^T\frac{\partial^2 g_k(0)}{\partial \mathbf{u}\partial \mathbf{v}}\right)\right\rangle \\
&= g_i(t) - \eta\sum_{k\in\mathscr{S}_+}(g_k(t)-y_k)K_{k,i}(t) + \frac{\eta^2}{m}\sum_{k\in\mathscr{S}_+}\sum_{j\in\mathscr{S}_+}(g_k(t)-y_k)(g_j(t)-y_j)g_j(t)x_k x_i.
\end{aligned}
$$

Similarly, for $i \in \mathscr{S}_-$, we have

$$g_i(t+1) = g_i(t) - \eta \sum_{k \in \mathscr{S}_-} (g_k(t) - y_k) K_{k,i}(t) + \frac{\eta^2}{m} \sum_{k \in \mathscr{S}_-} \sum_{f\, j \in \mathscr{S}_-} (g_k(t) - y_k)(g_j(t) - y_j) g_j(t) x_k x_i.$$

Combining the results together, we have

$$\begin{aligned} \mathbf{g}(t+1) - \boldsymbol{y} = \Bigg(I - \eta K(t) + \frac{\eta^2}{m} ((\mathbf{g}(t) - \boldsymbol{y}) \odot \boldsymbol{m}_+)^T (\mathbf{g}(t) \odot \boldsymbol{m}_+) \boldsymbol{p}_1 \boldsymbol{p}_1^T \\ + \frac{\eta^2}{m} ((\mathbf{g}(t) - \boldsymbol{y}) \odot \boldsymbol{m}_-)^T (\mathbf{g}(t) \odot \boldsymbol{m}_-) \boldsymbol{p}_2 \boldsymbol{p}_2^T \Bigg) (\mathbf{g}(t) - \boldsymbol{y}). \end{aligned}$$

B.3 Optimization with sub-critical learning rates

Theorem B.3.1. *Consider training the NQM Eq. (3.10), with squared loss on a single training example by GD. With a sub-critical learning rate* $\eta \in [\varepsilon, \frac{2-\varepsilon}{\lambda_0}]$ *with* $\varepsilon = \Theta\left(\frac{\log m}{\sqrt{m}}\right)$*, the loss decreases exponentially with*

$$\mathscr{L}(t+1) \leq \left(1 - \delta + O\left(\frac{1}{m\delta}\right)\right)^2 \mathscr{L}(t) = (1 - \delta + o(\delta))^2 \mathscr{L}(t),$$

where $\delta = \min(\eta\lambda_0, 2 - \eta\lambda_0)$.

Furthermore, $\sup_t |\lambda(t) - \lambda(0)| = O\left(\frac{1}{m\delta}\right)$.

We use the following transformation of the variables to simplify notations.

$$u(t) = \frac{\|\boldsymbol{x}\|^2}{md} \eta^2 (g(t) - y)^2, \quad w(t) = \frac{\|\boldsymbol{x}\|^2}{md} \eta^2 (g(t) - y) y, \quad v(t) = \eta\lambda(t).$$

Then the Eq. (3.11) and Eq. (3.12) are reduced to

$$\begin{aligned} u(t+1) &= (1 - v(t) + u(t) + w(t))^2 u(t) := \kappa(t) u(t) \\ v(t+1) &= v(t) - u(t)(4 - v(t)) - 4w(t). \end{aligned}$$

At initialization, since $|g(0)| = O(1)$, we have $u(0) \leq C_u/m$ for some constant $C_u > 0$. As $|w(t)| = \frac{C\sqrt{u(t)}}{\sqrt{m}}$. where $C := \frac{\eta \|\boldsymbol{x}\| |y|}{\sqrt{d}} > 0$, we have $|w(0)| \leq C_u C/m^{3/2}$. Note that by definition, for all $t \geq 0$, $u(t) \geq 0$ we have $v(t) \geq 0$ since $\lambda(t)$ is the tangent kernel for a single training example. From the definition of δ, we can infer that $\delta < 1$.

In the following, we will show that if $v(0) \in [\varepsilon, 2-\varepsilon]$ with $\varepsilon = \Theta\left(\frac{\log m}{\sqrt{m}}\right)$, then there exist constant $C_\kappa, C_v > 0$ such that for all $t \geq 0$,

$$\kappa(t) \leq \left(1 - \delta + \frac{C_\kappa}{m\delta}\right)^2 < 1, |v(t) - v(0)| \leq \frac{C_v}{m\delta},$$

if $C_\kappa \geq 9C_u + (C_u + C_u C)\delta$ and m satisfies

$$m > \max\left\{\frac{12C_\kappa}{\delta^2}, \frac{\sqrt{6}C_\kappa}{\delta^{3/2}}, C^2\right\}.$$

Given the condition on m, we have $(1 - \delta + \frac{C_\kappa}{m\delta})^2 < 1$.

We will prove the result by induction. When $t = 0$, we have

$$\kappa(0) = (1 - v(0) + u(0) + w(0))^2 < \left(1 - \delta + \frac{C_u}{m} + \frac{C_u C}{m^{3/2}}\right)^2 < \left(1 - \delta + \frac{C_\kappa}{m\delta}\right)^2,$$

where we use the assumption $C_\kappa \geq (C_u + C_u C)\delta$.

Therefore the result holds at $t = 0$.

Suppose when $t = T$ the results hold. Then at $t = T + 1$, by the inductive hypothesis that $u(t)$ decreases exponentially with $\kappa(t) < \left(1 - \delta + \frac{C_\kappa}{m\delta}\right)^2$, we can bound the change of $v(T+1)$

from $v(0)$:

$$
\begin{aligned}
&|v(T+1)-v(0)| \\
&= \sum_{t=0}^{T} |u(t)(4-v(t))+4w(t)| \\
&\le \sum_{t=0}^{T} |u(t)||4-v(t)| + \sum_{t=0}^{T} 4|w(t)| \\
&\le \max_{t\in[0,T]} |v(t)-4| \cdot \frac{u(0)-u(T)\max_{t\in[0,T]}\kappa(t)}{1-\max_{t\in[0,T]}\kappa(t)} + \frac{w(0)-w(T)\max_{t\in[0,T]}\sqrt{\kappa(t)}}{1-\max_{t\in[0,T]}\sqrt{\kappa(t)}} \\
&\le 4\cdot \frac{u(0)}{1-\max_{t\in[0,T]}\kappa(t)} + \frac{w(0)}{1-\max_{t\in[0,T]}\sqrt{\kappa(t)}} \\
&\le 4\cdot \frac{C_u/m}{\delta/2} + \frac{C_u C/m^{3/2}}{\delta/2} \\
&\le \frac{9C_u}{m\delta}.
\end{aligned}
$$

For the summation of the "geometric sequence" i.e., $\{u(0),u(1),\cdots,u(T)\}$ where $u(0)$ and $u(T)$ have the determined order but the ratio has an upper bound, we use the maximum ratio, i.e., $\max\kappa(t)$ in the denominator to upper bound the summation.

For $1-\max_{t\in[0,T]}\kappa(t)$, we use the bound that

$$
\begin{aligned}
1-\left(1-\delta+\frac{C_\kappa}{m\delta}\right)^2 &= 2\delta-\delta^2-\frac{C_\kappa^2}{m^2\delta^2}-\frac{2C_\kappa}{m\delta}+\frac{2C_\kappa}{m} \\
&\ge \delta-\frac{\delta}{6}-\frac{\delta}{6}-\frac{\delta}{6} \\
&\ge \frac{\delta}{2},
\end{aligned}
$$

where we use the assumption on m.

Furthermore,

$$\begin{aligned}\kappa(T+1) &= (1-v(T+1)+u(T+1)+w(T+1))^2 \\ &= (1-v(0)+v(0)-v(T+1)+u(T+1)+w(T+1))^2 \\ &\le \left(1-\delta+\frac{9C_u}{m\delta}+\frac{C_u}{m}+\frac{C_uC}{m^{3/2}}\right)^2 \\ &\le \left(1-\delta+\frac{C_\kappa}{m\delta}\right)^2.\end{aligned}$$

Here we use the assumption $C_\kappa \ge 9C_u + (C_u + C_uC)\delta$.

Therefore, we finish the inductive step hence finishing the proof.

B.4 Proof of Lemma 3.3.2

We present the formal statement of Lemma 3.3.2:

Lemma 3.3.2. *Consider constants $C_u, C_u', C_v, C_\kappa, C_\varepsilon > 0$ which satisfies $C_\kappa \ge 28C_u' + C_u'\delta + C\sqrt{C_u'}$ where $C = \eta \|\boldsymbol{x}\| |y|/\sqrt{d}$, and $C_v \ge 28C_u'$. If m satisfies*

$$m \ge \max\left\{\frac{C_\kappa\delta}{C_u(C+1)}, \left(\frac{2C_u+4C\sqrt{C_u}}{C_\kappa C_\varepsilon}\right)^2, \frac{576C^2}{C_u'\delta^2}, \exp(C_v\delta), \exp(4C_\kappa)\right\},$$

then with high probability over random initialization of the weights, the following holds: for $T > 0$ such that $\sup_{t\in[0,T]} u(t) \le \frac{C_u'\delta^2}{\log m}$, $u(t)$ increases exponentially with ratio $\inf_{t\in[0,T]} \kappa(t) \ge \left(1+\delta-\frac{C_\kappa\delta}{\log m}\right)^2 > 1$ and $\sup_{t\in[0,T]} |v(t)-v(0)| \le \frac{C_v\delta}{\log m}$.

Proof. Due to the random initialization of the weights, we have with probability, there exists constant $C_u > 0$ such that $|u(0)| \le C_u/m$. As $|w(t)| = \frac{C\sqrt{u(t)}}{\sqrt{m}}$, where $C := \frac{\eta\|\boldsymbol{x}\||y|}{\sqrt{d}} > 0$, we have $|w(0)| \le \frac{C\sqrt{C_u}}{m^{3/2}}$.

We prove the results by induction.

Recall that $\delta := \eta\lambda_0 - 2 \in [\varepsilon, 2-\varepsilon]$ where $\varepsilon \in [C_\varepsilon \log m/\sqrt{m}, C_\varepsilon' \log m/\sqrt{m}]$ for some constant $0 < C_\varepsilon < C_\varepsilon'$.

When $t = 0$, as $v(0) = \eta\lambda_0 = \delta + 2$, we have

$$\begin{aligned}\kappa(0) &= (1 - v(0) + u(0) + w(0))^2 \\ &= (1 - (\delta + 2) + u(0) + w(0))^2 \\ &= (1 + \delta - u(0) - w(0))^2 .\end{aligned}$$

Based on the condition on m that $m \geq \frac{C_\kappa \delta}{C_u(C+1)}$, we have,

$$\begin{aligned}(1 + \delta - u(0) - w(0))^2 &\geq \left(1 + \delta - \frac{C_u}{m} - \frac{C_u C}{m^{3/2}}\right)^2 \\ &\geq \left(1 + \delta - \frac{C_u}{m} - \frac{C_u C}{m}\right)^2 \\ &\geq \left(1 + \delta - \frac{C_\kappa \delta}{\log m}\right)^2 .\end{aligned}$$

And by the condition $m \geq \left(\frac{2C_u + 4c\sqrt{C_u}}{C_\kappa C_\varepsilon}\right)^2$, we get

$$|v(1) - v(0)| \leq |u(0)||2 - \delta| + 4|w(0)| \leq 2C_u/m + \frac{4C\sqrt{C_u}}{m^{3/2}} \leq C_\kappa \delta / \log m.$$

Therefore the results hold at $t = 0$.

Suppose when $t = T'$ the results hold. Then at $t = T' + 1$, by the inductive hypothesis that $u(t)$ increases exponentially with a rate at least $\left(1 + \delta - \frac{C_\kappa \delta}{\log m}\right)^2$ from $u(0) \leq C_u/m$ to

$u(T') \leq \frac{C'_u \delta^2}{\log m}$, we can bound the change of $v(t)$:

$$
\begin{aligned}
|v(T'+1) - v(0)| &= \left| \sum_{t=1}^{T'} u(t)(v(t)-4) + 4w(t) \right| \\
&\leq \max_{t\in[0,T']} |v(t)-4| \sum_{t=1}^{T'} u(t) + 4 \sum_{t=1}^{T'} |w(t)| \\
&\leq \max_{t\in[0,T']} |v(t)-4| \frac{u(T') \min_{t\in[0,T']} \kappa(t)}{\min_{t\in[0,T']} \kappa(t) - 1} + 4 \frac{|w(T')| \min_{t\in[0,T']} \sqrt{\kappa(t)}}{\min_{t\in[0,T']} \sqrt{\kappa(t)} - 1} \\
&\leq \left(\max_{t\in[0,T']} |v(t) - v(0)| + |v(0)-4| \right) \frac{\left(\frac{C'_u \delta^2}{\log m}\right) \cdot (1+\delta)^2}{\left(1+\delta - \left(\frac{C_\kappa \delta}{\log m}\right)\right)^2 - 1} \\
&\quad + \frac{4\left(\frac{C\sqrt{C'_u}\delta}{\sqrt{m \log m}}\right) \cdot (1+\delta)}{\left(1+\delta - \left(\frac{C_\kappa \delta}{\log m}\right)\right) - 1} \\
&\leq \left(2 - \delta + \frac{C_v \delta}{\log m}\right) \cdot \left(\frac{9C'_u \delta}{\log m}\right) + \frac{24C\sqrt{C'_u}}{\sqrt{m \log m}} \\
&\leq \frac{28C'_u \delta}{\log m}.
\end{aligned} \tag{B.3}
$$

Here are the techniques we used for the above inequalities: for the summation of the "geometric sequence" i.e., $\{u(0), u(1), \cdots, u(T')\}$ where $u(0)$ and $u(T')$ have the determined order but the ratio has a lower bound, we use the smallest ratio, i.e., $\inf \kappa(t)$ to upper bound the summation. Specifically, we apply the following inequality to bound the summation:

$$
\begin{aligned}
\sum_{t=1}^{T'} u(t) &\leq \sum_{t=1}^{T'} \frac{u(T')}{\left(\min_{t\in[0,T']} \kappa(t)\right)^{t-1}} \\
&= u(T') \sum_{t=1}^{T'} \frac{1}{\left(\min_{t\in[0,T']} \kappa(t)\right)^{t-1}} \\
&\leq u(T') \frac{\min_{t\in[0,T']} \kappa(t)}{\min_{t\in[0,T']} \kappa(t) - 1}.
\end{aligned}
$$

Additionally, sine $m \geq \exp(4C_\kappa\delta)$, we used the inequality

$$\begin{aligned}\left(1+\delta-\frac{C_\kappa\delta}{\log m}\right)^2-1 &= \left(1-\frac{C_\kappa\delta}{\log m}\right)^2\delta^2+2\left(1-\frac{C_\kappa\delta}{\log m}\right)\delta \\ &\geq 2\left(1-\frac{C_\kappa\delta}{\log m}\right)\delta \geq \delta,\end{aligned}$$

and $\left(1+\delta-\left(\frac{C_\kappa\delta}{\log m}\right)\right)-1 \geq \frac{\delta}{2}$ to bound the denominator of the summation of the geometric sequence.

And we further used the inequality $0<\delta<2$ and $\frac{24C\sqrt{C_u'}}{\sqrt{m}\log m} \leq \frac{C_u'\delta}{\log m}$ by the condition on m to get the final upper bound.

Consequently, by the assumption $C_v \geq 28C_u'\delta$, we have $|v(T'+1)-v(0)| \leq \frac{C_v\delta}{\log m}$.

Now we bound the ratio $\kappa(T'+1)$. By our assumption, $u(T'+1) \leq u(T) \leq \frac{C_u'\delta^2}{\log m}$, and we can similarly bound $|w(T'+1)| \leq \frac{C\sqrt{C_u'}\delta}{\sqrt{m}\log m}$ as $|w(T'+1)| = \frac{C\sqrt{u(T'+1)}}{\sqrt{m}}$.

And the rate $\kappa(T'+1)$ satisfies

$$\begin{aligned}\kappa(T'+1) &= (1-v(T'+1)+u(T'+1)+w(T'+1))^2 \\ &= \big(1-v(0)+v(0)-v(T'+1)+u(T'+1)+w(T'+1)\big)^2 \\ &= \big(1+\delta+v(T'+1)-v(0)-u(T'+1)-w(T'+1)\big)^2.\end{aligned}$$

Note that $|v(T'+1)-v(0)| \leq \frac{28C_u'\delta}{\log m}$ by Eq. (B.3). By the assumption that $m \geq \exp(4C_\kappa)$ and $C_\kappa \geq 28C_u'+C_u'\delta+C\sqrt{C_u'}$, we have $\delta > |v(T'+1)-v(0)|+u(T'+1)+|w(T'+1)|$.

Consequently, we can get

$$\begin{aligned}\kappa(T'+1) &= \big(1+\delta+v(T'+1)-v(0)-u(T'+1)-w(T'+1)\big)^2 \\ &\geq \big(1+\delta-|v(T'+1)-v(0)|-u(T'+1)-|w(T'+1)|\big)^2 \\ &\geq \left(1+\delta-\frac{28C_u'\delta}{\log m}-\frac{C_u'\delta^2}{\log m}-\frac{C\sqrt{C_u'}\delta}{\sqrt{m\log m}}\right)^2 \\ &\geq \left(1+\delta-\frac{C_\kappa\delta}{\log m}\right)^2.\end{aligned}$$

Since $m \geq \exp(4C_\kappa)$, we have $\left(1+\delta-\frac{C_\kappa\delta}{\log m}\right)^2 \geq \left(1+\frac{3}{4}\delta\right)^2 > 1$.

Then we finish the inductive step hence finishing the proof.

□

B.5 Proof of Lemma 3.3.3

A formal statement of Lemma 3.3.3 is as follows:

Lemma 3.3.3. *Under the condition of Lemma 3.3.2, if we further assume that m satisfies*

$$m > \max\left\{\exp(2C_v\delta), \frac{256C^2}{(C_\varepsilon-C_v')^2{C_u'}^2}, \exp\Big(5(C_u'+4C\sqrt{C_u'})\Big), \exp\left(\frac{C_u'(C_\varepsilon-2C_v')-8C\sqrt{C_u'}}{20CC_u'}\right)\right\},$$

where $C_v' := 18C_u'+2C_v$, and $C_v \geq 4C\sqrt{C_u'}$, $C_\varepsilon > 2C_v'$, then with high probability over random initialization of the weights, the following holds: there exists $T^* > 0$ such that $u(T^*) = O\left(\frac{1}{m}\right)$.

Proof. The main idea of the proof is the following: as $u(t)$ increases, $v(t)$ decreases since $u(t)(4-v(t)) \gg w(t) = \Theta(\sqrt{u(t)}/\sqrt{m})$ in Eq. (3.14) and $u(t)(4-v(t)) < 0$. Furthermore, the increase of $u(t)$ speeds up the decrease of $v(t)$. However, $v(t)$ cannot decrease infinitely as $v(t) \geq 0$ by definition. Therefore, $u(t)$ has to stop increasing at some point and decrease to a small value.

We first show that by the choice of the learning rate that $4-v(0) \geq \varepsilon$ where $\varepsilon = \Theta\left(\frac{\log m}{\sqrt{m}}\right)$, we will have $4-v(t) > 0$ for all t in the increasing phase. Recall that $\delta := \eta\lambda_0 - 2$.

Proposition B.5.1. *Under the condition in Lemma 3.3.2, if we further assume that* $m > \exp\left(\frac{48C\sqrt{C_u'}}{C_\varepsilon}\right)^{\frac{2}{3}}$, *then for* $T > 0$ *such that* $\sup_{t\in[0,T]} u(t) \leq \frac{C_u'\delta^2}{\log m}$, *we have* $v(T) < 4 - \frac{C_\varepsilon \log m}{2\sqrt{m}}$.

See the proof in Appendix B.8

Given the constant C_u' in Lemma 3.3.2, we define the end of the increasing phase by T_1, i.e.,

$$T_1 := \sup\left\{t : u(t) \leq \frac{C_u'\delta^2}{\log m}\right\}. \tag{B.4}$$

We further show that there exists $T_2 \geq T_1$ such that $v(T_2) \leq 3$.

Note that we indeed can show that there exists T_2 such that $v(T_2) < \overline{C}$ where $\overline{C} \in (2,4)$ is a constant independent of m. Here for the simplicity of the presentation, we take C as 3. Furthermore, we note that T_1, T_2 depends on m.

Before that, we present a useful result that controls the decrease of $v(t)$:

Proposition B.5.2. *For* t *such that* $v(t) < 4$, *if* $u(t) > \frac{4C}{m(4-v(t))^2}$, *then* $v(t+1) < v(t)$.

See the proof in Appendix B.8.

Now we are ready to show the existence of T_2 such that $v(T_2) \leq 3$.

Proposition B.5.3. *Under the condition of Lemma 3.3.2, if we further assume that m satisfies*

$$m > \max\left\{\exp\left(\frac{768^2C^2}{C_u'C_\varepsilon^2}\right), \exp\left(2C_u' + C_\varepsilon\right), \exp\left(\frac{48C\sqrt{C_u'}}{C_\varepsilon}\right)^{2/3}, \frac{16C^2}{{C_u'}^2}\right\},$$

and $C_u' \geq 4C^2$, *there exists* $T_2 \geq T_1$ *such that* $v(T_2) \leq 3$.

See the proof in Appendix B.8

Since $v(T_2) < 3$ hence $4 - v(T_2) \geq 1$. Simply using Proposition B.5.2, we get

Proposition B.5.4. *$v(t)$ keeps decreasing after T_2 until $u(t) = O\left(\frac{1}{m}\right)$.*

By definition $v(t) = \eta\lambda(t)$ where $\lambda(t) \geq 0$, $v(t)$ will not keep decreasing for $t \to \infty$ hence there exists T^* such that $u(T^*) = O\left(\frac{1}{m}\right)$. And it indicates that the loss will decrease to the order of $O(1)$.

B.6 Proof of Theorem 3.3.4

We compute the steady-state equilibria of Eq. (3.13) and (3.14). By letting $u(t+1) = u(t)$ and $v(t+1) = v(t)$, we have the steady-state equilibria (u^*, v^*) satisfy one of the following:

(1) $u^* = 0$, $v^* \in \mathbb{R}$;

(2) $|1 - v^* + u^* + w^*| = 1$, $u^*(4 - v^*) + 4w^* = 0$.

As $w(t)^2 = \frac{C^2 u(t)}{m}$ where $C := \frac{\eta\|\boldsymbol{x}\|\|y\|}{\sqrt{d}} > 0$, we write w as a function of u for simplicity, hence $w^* = w(u^*)$.

As the dynamics equations are non-linear, we analyze the local stability of the steady-state equilibria. We consider the Jacobian matrix of the dynamical systems:

$$J(u,v) = \begin{bmatrix} 2(1-v+u+w)(1+\frac{dw}{du})u + (1-v+u+w)^2 & -2(1-v+u+w)u \\ v - 4 - 4\frac{dw}{du} & 1+u \end{bmatrix}.$$

We analyze the stability of two equilibria separately.

For Scenario (1), we evaluate $J(u,v)$ at the steady-state equilibrium (u^*, v^*) then we get

$$J(u^*, v^*) = \begin{bmatrix} (1-v^*)^2 & 0 \\ v^* - 4 - 4\frac{dw}{du} & 1 \end{bmatrix}.$$

We get the two eigenvalues of $J(u^*, v^*)$ are 1 and $(1-v^*)^2$. We will show the Lyapunov stability

of the equilibrium (u^*, v^*). Specifically, we apply Theorem 1.2 in [14]. We find the domain

$$D = \{(u,v) : u \leq C_1, |v - v^*| \leq \min(|C_2 - v^*|, |2 - C_2 - v^*|\},$$

where $C_1 = \Theta(1/m)$ and $C_2 = \Theta(1/\sqrt{m})$, and the Lyapunov function $V(u,v) = u + (v - v^*)^2$. It is not hard to verify that V is locally Lipschitz in D as V is continuous in a compact domain. Furthermore, we can see that (u^*, v^*) with $u^* = 0, v^* \in [\varepsilon, 2 - \varepsilon]$ where $\varepsilon = \Theta(\log m/\sqrt{m})$ satisfies the condtions Eq. (3,4) in Theorem 1.2 in [14]. Therefore, (u^*, v^*) with $u^* = 0$ and $v^* \in [\varepsilon, 2 - \varepsilon]$ is a stable equilibrium point.

For Scenario (2), we again evaluate $J(u,v)$ at the steady-state equilibrium (u^*, v^*) then we get

$$J(u^*, v^*) = \begin{bmatrix} -2u^* + \frac{C\sqrt{u^*}}{\sqrt{m}} + 1 & 2u^* \\ -\frac{2C}{\sqrt{mu^*}} & 1 + u^* \end{bmatrix},$$

where we replace v^* by $4 + 4w^*/u^*$ based on the second equality in Scenario (2). Note that $u^*(4 - v^*) > 0$ since $v < 4$ during the whole training process, therefore we have $w^* < 0$ to achieve the equilibrium.

We can compute the eigenvalue of $J(u^*, v^*)$ then we get

$$\lambda_J = 1 + \frac{C}{2\sqrt{m}}\sqrt{u^*} - \frac{u^*}{2} \pm \frac{1}{2}(u^*)^{1/4}\sqrt{16\frac{C}{\sqrt{m}} - \frac{C^2\sqrt{u^*}}{m} + 6\frac{Cu^*}{\sqrt{m}} - 9(u^*)^{3/2}i}.$$

Note that when Scenario (2) holds, there are only two possible cases

(2.1) $u^* = \Theta(1/m)$, $|w^*| = \Theta(1/m)$ and $v^* = \Theta(1)$;

(2.2) $u^* = \Theta(1/m)$, $|w^*| = \Theta(1/m)$ and $v^* = \Theta(1/m)$.

For (2.1), by the first equality $v^* = 2 - u^* + w^* \in (1,2)$. Then plugging v^* into the second equality yields $u^* \in \left(\frac{4}{3}\frac{C^2}{m}, 2\frac{C^2}{m}\right)$.

For (2.2), by the second equality that $u^*(4-v^*)+4w^*=0$, we have $u^*=\frac{C^2}{m}+o(1/m)$. By computing the modulo of λ_J, we have

$$|\lambda_J| = 1+\frac{5C}{\sqrt{m}}\sqrt{u^*}-u^*+o\left(\frac{1}{m}\right).$$

Therefore, for both (2.1) and (2.2) we have $|\lambda_J|>1$ which indicates (u^*,v^*) is unstable.

□

B.7 Optimization with $\eta > \eta_{\max}$

Theorem B.7.1. *Consider training the NQM Eq. (3.10), with squared loss on a single training example by GD. If the learning rate satisfies* $\eta \in \left[\frac{4+\varepsilon}{\lambda_0},\infty\right)$ *with* $\varepsilon=\Theta\left(\frac{\log m}{\sqrt{m}}\right)$*, then GD diverges.*

Proof. We similarly use the transformation transformation of the variables to simplify notations.

$$u(t)=\frac{\|\boldsymbol{x}\|^2}{md}\eta^2(g(t)-y)^2,\quad w(t)=\frac{\|\boldsymbol{x}\|^2}{md}\eta^2(g(t)-y)y,\quad v(t)=\eta\lambda(t).$$

Then the Eq. (3.11) and Eq. (3.12) are reduced to

$$\begin{aligned}u(t+1)&=(1-v(t)+u(t)+w(t))^2u(t):=\kappa(t)u(t)\\ v(t+1)&=v(t)-u(t)(4-v(t))-4w(t).\end{aligned}$$

We similarly consider the interval $[0,T]$ such that $\sup_{t\in[0,T]}u(t)=O(1/\log m)$. By Lemma 3.3.2, in $[0,T]$, $u(t)$ increases exponentially with a rate $\sup_{t\in[0,T]}\kappa(t)>9$. We assume $|w(t)|>|u(t)(4-v(t))|$ for all $t\in[0,T]$, which is the worst case as $v(t)$ will increase the least. By Lemma 3.3.2, we have $\sum_{t=0}^{T}|w(t)|=O\left(\frac{1}{\sqrt{m\log m}}\right)$, which is less than ε. Therefore, we have $v(T)>4$.

Then at the end of the increasing phase, we have $|u(T_1)(4-v(T_1))|=\Omega(1/\sqrt{m})$ is of a greater order than $|w(T_1)|=O(1/\sqrt{m\log m})$, hence $v(t)$ will increase at T_1. Note that

$\kappa(T_1) = (1-4+o(1))^2 = 9+o(1)$, hence $u((t)$ also increases at T_1.

It is not hard to see that $v(t)$ will keep increasing unless $u(t)$ decreases to a smaller order. Specifically, if $|u(t)(4-v(t))| = |4w(t)|$, it requires $u(t)$ to be of the order at least $O(1/\sqrt{\log m})$ (by letting $\varepsilon u(t) = \Theta(w(t)) = \Theta(\sqrt{u(t)/m})$), which will not happen as $\kappa(t) = (1-v(t)+o(1))^2 > 1$ and it contradicts the decrease of $u(t)$.

Therefore, both $u(t)$ and $v(t)$ keep increasing which leads to the divergence of GD.

□

B.8 Proof of propositions

Proof of Proposition B.5.1

Proof. Note that $4-v(0) = 2-\delta \geq \frac{C_\varepsilon \log m}{\sqrt{m}}$ by definition, where $C_\varepsilon > 0$ is a constant. To show $4-v(T) > \frac{C_\varepsilon \log m}{2\sqrt{m}}$, a sufficient condition is $v(T)-v(0) < \frac{C_\varepsilon \log m}{2\sqrt{m}}$.

Specifically, we will prove for $T > 0$ such that $\sup_{t\in[0,T]} u(t) \leq \frac{C_u' \delta^2}{\log m}$, the following holds:

$$v(T)-v(0) < 4\sum_{t=0}^{T} |w(t)| \leq \frac{24C\sqrt{C_u'}}{\sqrt{m\log m}},$$

where C, C_u' are the same constants defined in Lemma 3.3.2. Then by the condition that $m > \exp\left(\frac{48C\sqrt{C_u'}}{C_\varepsilon}\right)^{2/3}$, we have $v(T)-v(0) < \frac{C_\varepsilon \log m}{2\sqrt{m}}$.

We will prove the result by induction.

When $T = 0$, the result holds trivially.

Suppose $T = T'$ the result holds. When $T = T'+1$, since $v(T')-v(0) < 0$, we have $v(T') < 4$. Therefore, by the update equation of $v(t)$ Eq. (3.14), we have

$$\begin{aligned} v(T'+1) &= v(T') - u(T')(4-v(T')) - 4w(T') \\ &\leq v(T') - 4w(T') \\ &\leq v(T') + 4|w(T')|. \end{aligned}$$

Then $v(T'+1)-v(0) = v(T'+1)-v(T')+v(T')-v(0) \le \sum_{t=0}^{T'+1}|w(t)|$.

By Lemma 3.3.2, we have $\sum_{t=0}^{T'+1}|w(t)| \le \frac{24C\sqrt{C'_u}}{\sqrt{m\log m}}$. Indeed, this inequality holds for any $T'+1$ such that $\sup_{t\in[0,T'+1]} u(t) \le \frac{C'_u\delta^2}{\log m}$.

Therefore, we finish the inductive step hence finish the proof.

□

Proof of Proposition B.5.2

Proof. A sufficient condition for $v(t)$ to decrease is

$$u(t)(4-v(t)) > 4|w(t)| = \frac{4C\sqrt{u(t)}}{\sqrt{m}}.$$

If $u(t) > \frac{4C}{m(4-v(t))^2}$, then the above condition is satisfied. □

Proof of Proposition B.5.3

Proof. Note that for $t \in [0, T_1]$, by Lemma 3.3.2, the change of $v(t)$ satisfies $\sup_t |v(t)-v(0)| \le \frac{C_v\delta}{\log m}$.

For $\delta < 1 - \frac{C_v\delta}{\log m}$, i.e., $v(0) < 3 - \frac{C_v\delta}{\log m}$, we have $v(T_1) < v(0) + |v(T_1)-v(0)| = 2+\delta+|v(0)-v(T_1)| < 3$. Therefore, the existence of T_2 can be guaranteed by simply letting $T_2 = T_1$.

For $\delta \ge 1 - \frac{C_v\delta}{\log m}$, i.e., $v(0) \ge 3 - \frac{C_v\delta}{\log m}$, we will show there exists $T_2 \ge T_1$ which depends on m such that $v(T_2) < 3$.

We prove the existence of T_2 by contradiction. Suppose that for all $t \ge T_1+1$ we have $v(t) \ge 3$.

For the simple case that if all $u(t) > \frac{4C}{m(4-v(t))^2}$, then by Proposition B.5.2, $v(t)$ keeps decreasing which will ultimately lead to $v(t) < 3$.

Suppose there is an iteration $t \ge T_1+1$ such that $u(t) \le \frac{4C}{m(4-v(t))^2}$. The following Proposition guarantees that $v(t)$ will decrease to a smaller value after t once such t occurs. Therefore, we can find T_2.

Proposition B.8.1. *Under the condition of Lemma 3.3.2, suppose m further satisfies*

$$m > \max\left\{\exp\left(\frac{768^2C^2}{C_u'C_\varepsilon^2}\right), \frac{16C^2}{{C_u'}^2}, \exp\left(2C_u' + C_\varepsilon\right)\right\},$$

where $C_u' \geq 4C^2$.

Then if there is $T \geq 0$ *such that* $u(T) \leq \frac{4C}{m(4-v(T))^2}$ *and* $v(T) > 3$, *we have* $v(T'+2) < v(T)$ *and* $u(T'+1) > \frac{4C}{m(4-v(T'))^2}$, *where* T' *is the end of the increasing phase starting from* T.

See the proof in Appendix B.8.

□

Proof of proposition B.8.1

Proof. Since $u(T) \leq \frac{C_u'}{\log m}$ by the assumption that $m > \frac{16C^2}{{C_u'}^2}$, the training dynamics falls into the increasing phase from T. We denote the end of the increasing phase starting from T by T', i.e.,

$$T' = \sup\left\{t : u(t) \leq \frac{C_u'\delta^2}{\log m}, t \geq T\right\}.$$

We will prove the result by induction.

Suppose T' is the end of the first increasing phase, i.e., $T' = T_1$. By proposition B.5.1, $v(T_1) < 4 - \frac{C_\varepsilon \log m}{2\sqrt{m}}$. And the magnitude of $u(T_1+1)$ can be lower bounded by

$$u(T_1+1) \geq \frac{C_u'}{4\log m}, \tag{B.5}$$

where we use $\delta \geq \frac{1}{2}$ by the assumption on m that and plug δ in $\frac{C_u'\delta^2}{\log m}$.

Note that when $m > \exp(28C_u')$, $\delta \geq \frac{1}{2}$ is necessary for $v(T_1) > 3$ as $|v(T_1) - v(0)| \leq \frac{28C_u'\delta}{\log m}$ by Inequality (B.3).

Furthermore, with the above bound on $u(T_1+1)$, we have

$$
\begin{aligned}
v(T_1+1) &= v(T_1) - u(T_1)(4 - v(T_1)) - 4w(T_1) \\
&\leq 4 - \frac{C_\varepsilon \log m}{2\sqrt{m}} + \frac{C'_u}{4\log m}\frac{C_\varepsilon \log m}{2\sqrt{m}} + 4|w(T_1)| \\
&\leq 4 - \frac{C_\varepsilon \log m}{2\sqrt{m}} + \frac{C'_u C_\varepsilon}{8\sqrt{m}} + \frac{2\sqrt{C'_u}}{\sqrt{m\log m}} \\
&\leq 4 - \frac{C_\varepsilon \log m}{4\sqrt{m}}.
\end{aligned} \tag{B.6}
$$

Consequently, when $C'_u \geq 4C^2$ and $m > \exp(2C'_u + C_\varepsilon)$, we have

$$
\begin{aligned}
u(T_1+1) = \kappa(T_1)u(T_1) &= (1 - v(T_1) + u(T_1) + w(T_1))^2 u(T_1) \\
&\leq \left(1 - 4 + \frac{C_\varepsilon \log m}{4\sqrt{m}} + \frac{C'_u}{4\log m} + \frac{C\sqrt{C'_u}}{2\sqrt{m\log m}}\right)^2 \frac{C'_u}{4\log m} \\
&\leq \frac{9C'_u}{4\log m}.
\end{aligned}
$$

Therefore, at $T_1 + 1$, we have

$$
\begin{aligned}
v(T_1+1) - v(T_1+2) &\geq u(T_1+1)(4 - v(T_1+1)) - \frac{4C\sqrt{u(T_1+1)}}{\sqrt{m}} \\
&\geq \frac{C'_u}{4\log m}\frac{C_\varepsilon \log m}{4\sqrt{m}} - \frac{8C\sqrt{C'_u}}{\sqrt{m\log m}} \\
&= \frac{C'_u C_\varepsilon}{16\sqrt{m}} - \frac{6C\sqrt{C'_u}}{\sqrt{m\log m}} \\
&\geq \frac{C'_u C_\varepsilon}{32\sqrt{m}},
\end{aligned}
$$

where we use the assumption that $m > \exp\left(\frac{256C^2}{9C'_u C_\varepsilon^2}\right)$.

Note that the increase is caused by the term $w(t)$ since for all $t \in [T, T_1 + 1]$ we have

$u(t)(4-v(t)) < 0$. Then we have the maximum increase during $[T, T_1+1]$ be bounded by

$$v(T_1+1) - v(T) \le \sum_{t=T}^{T_1+1} 4|w(t)| \le \frac{24C\sqrt{C_u'}}{\sqrt{m\log m}},$$

where we use Eq. (B.3) in the proof of Lemma 3.3.2.

By the assumption on m that $m > \exp\left(\frac{768^2C^2}{C_u'C_\varepsilon^2}\right)$, we have $v(T_1+2) < v(T)$.

If there is $\widetilde{T} > 0$ such that $u(\widetilde{T}) \le \frac{4C}{m(4-v(\widetilde{T}))^2}$ while $v(\widetilde{T}) > 3$, there is another increasing phase. Since $v(\widetilde{T}) < v(T)$, we can apply the same analysis under the same condition to show $v(\widetilde{T}_1+2) < v(\widetilde{T})$, where $\widetilde{T}_1$ is the end of the increasing phase starting from $\widetilde{T}$. Therefore, we finish the inductive step hence finish the proof.

□

Proof of Proposition B.11.1

Restate Proposition B.11.1: *For any* $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$, $\mathrm{rank}(K) \le 2$. *Furthermore,* $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ *are eigenvectors of K, where* $p_{1,i} = x_i \mathbb{1}_{\{i \in \mathscr{S}_+\}}$, $p_{2,i} = x_i \mathbb{1}_{\{i \in \mathscr{S}_-\}}$, *for* $i \in [n]$.

Proof. By Definition 3.2.1,

$$K_{i,j} = \frac{1}{m}\sum_{k=1}^{m} (v_k^2 + u_k^2) x_i x_j \mathbb{1}_{\{u_k x_i \ge 0\}} \mathbb{1}_{\{u_k x_j \ge 0\}}, \quad i, j \in [n].$$

By definition of eigenvector, we can see

$$\begin{aligned}
\sum_{j=1}^{n} K_{i,j} p_{1,j} &= \frac{1}{m}\sum_{j=1}^{n}\sum_{k=1}^{m} (v_k^2 + u_k^2) x_i x_j^2 \mathbb{1}_{\{u_k x_i \ge 0\}} \mathbb{1}_{\{u_k x_j \ge 0\}} \mathbb{1}_{\{j \in \mathscr{S}_+\}} \\
&= \sum_{j=1}^{n} x_j^2 \mathbb{1}_{\{j \in \mathscr{S}_+\}} \frac{1}{m}\sum_{k=1}^{m} (v_k^2 + u_k^2) x_i \mathbb{1}_{\{u_k x_i \ge 0\}} \mathbb{1}_{\{u_k x_j \ge 0\}} \\
&= x_i \mathbb{1}_{\{x_i \in S_+\}} \sum_{j=1}^{n} x_j^2 \mathbb{1}_{\{j \in \mathscr{S}_+\}} \frac{1}{m}\sum_{k=1}^{m} (v_k^2 + u_k^2) \mathbb{1}_{\{u_k x_j \ge 0\}},
\end{aligned}$$

where we use the fact that if $x_i x_j < 0$, $K_{i,j} = 0$.

As $p_{1,i} = x_i \mathbb{1}_{\{x_i \in S_+\}}$ and $\sum_{j=1}^{n} x_j^2 \mathbb{1}_{\{j \in \mathscr{S}_+\}} \frac{1}{m}\sum_{k=1}^{m} (v_k^2 + u_k^2) \mathbb{1}_{\{u_k x_j \ge 0\}}$ does not depend on i, we can

see $\boldsymbol{p}_1$ is an eigenvector of K with corresponding eigenvalue $\lambda_1 = \sum_{j=1}^n x_j^2 \mathbb{1}_{\{j\in\mathscr{S}_+\}} \frac{1}{m}\sum_{k=1}^m (v_k^2 + u_k^2)\mathbb{1}_{\{u_k x_j \geq 0\}}$.

The same analysis can be applied to show $\boldsymbol{p}_2$ is another eigenvector of K with corresponding $\lambda_2 = \sum_{j=1}^n x_j^2 \mathbb{1}_{\{j\in\mathscr{S}_-\}} \frac{1}{m}\sum_{k=1}^m (v_k^2 + u_k^2)\mathbb{1}_{\{u_k x_j \geq 0\}}$.

For the rank of K, it is not hard to verify that $K = \lambda_1 \boldsymbol{p}_1\boldsymbol{p}_1^T + \lambda_2 \boldsymbol{p}_2\boldsymbol{p}_2^T$ hence the rank of K is at most 2. □

B.9 Scale of the tangent kernel for single training example

Proposition B.9.1 (Scale of tangent kernel)**.** *For any $\delta \in (0,1)$, if $m \geq c' \log(4/\delta)$ where c' is an absolute constant, with probability at least $1-\delta$, $\|\boldsymbol{x}\|^2/(2d) \leq \lambda(0) \leq 3\|\boldsymbol{x}\|^2/(2d)$.*

Proof. Note that when $t = 0$,

$$\lambda(0) = \frac{1}{md}\sum_{i=1}^m \left(\mathbf{u}_{0,i}^T \boldsymbol{x} \mathbb{1}_{\left\{\mathbf{u}_{0,i}^T \boldsymbol{x} \geq 0\right\}}\right)^2 + \frac{1}{md}\sum_{i=1}^m (v_{0,i})^2 \|\boldsymbol{x}\|^2 \left(\mathbb{1}_{\left\{\mathbf{u}_{0,i}^T \boldsymbol{x} \geq 0\right\}}\right)^2.$$

According to NTK initialization, for each $i \in [m]$, $v_{0,i} \sim \mathscr{N}(0,1)$ and $\mathbf{u}_{0,i} \sim \mathscr{N}(0, I)$. We consider the random variable

$$\zeta_i := \mathbf{u}_{0,i}^T \boldsymbol{x} \mathbb{1}_{\left\{\mathbf{u}_{0,i}^T \boldsymbol{x} \geq 0\right\}}, \quad \xi_i := v_{0,i} \mathbb{1}_{\left\{\mathbf{u}_{0,i}^T \boldsymbol{x} \geq 0\right\}}.$$

it is not hard to see that ζ_i and ξ_i are sub-guassian since $\mathbf{u}_{0,i}^T \boldsymbol{x}$ and $v_{0,i}$ are sub-gaussian. Specifically, for any $t \geq 0$,

$$\mathbb{P}\{|\zeta_i| \geq t\} \leq \mathbb{P}\{|\mathbf{u}_{0,i}^T \boldsymbol{x}| \geq t\} \leq 2\exp\left(-t^2/(2\|\boldsymbol{x}\|^2)\right),$$

$$\mathbb{P}\{|\xi_i| \geq t\} \leq \mathbb{P}\{|v_{0,i}| \geq t\} \leq 2\exp\left(-t^2/2\right),$$

where the second inequality comes from the definition of sub-gaussian variables.

Since ξ_i is sub-gaussian, by definition, ξ^2 is sub-exponential, and its sub-exponential norm is bounded:

$$\|\xi_i^2\|_{\psi_1} \leq \|\xi_i\|_{\psi_2}^2 \leq C,$$

where $C > 0$ is a absolute constant. Similarly we have $\|\zeta_i\|_{\psi_2}^2 \leq C\|\boldsymbol{x}\|^2$.

By Bernstein's inequality, for every $t \geq 0$, we have

$$\mathbb{P}\left\{\left|\sum_{i=1}^m \xi_i^2 - \frac{m}{2}\right| \geq t\right\} \leq 2\exp\left(-c\min\left(\frac{t^2}{\sum_{i=1}^m \|\xi_i^2\|_{\psi_1}^2}, \frac{t}{\max_i \|\xi_i^2\|_{\psi_1}}\right)\right),$$

where $c > 0$ is an absolute constant.

Letting $t = m/4$, we have with probability at least $1 - 2\exp(-m/c')$,

$$\frac{m}{4} \leq \sum_{i=1}^m \xi_i^2 \leq \frac{3m}{4},$$

where $c' = c/(4C)$.

Similarity, we have with probability at least $1 - 2\exp(-m/c')$,

$$\frac{m}{4}\|\boldsymbol{x}\|^2 \leq \sum_{i=1}^m \zeta_i^2 \leq \frac{3m}{4}\|\boldsymbol{x}\|^2.$$

As a result, using union bound, we have probability at least $1 - 4\exp(-m/c')$,

$$\frac{\|\boldsymbol{x}\|^2}{2d} \leq \lambda(0) \leq \frac{3\|\boldsymbol{x}\|^2}{2d}.$$

□

B.10 Scale of the tangent kernel for multiple training examples

Proof. As shown in Proposition B.11.1, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ are eigenvectors of K, hence we have two eigenvalues:

$$\lambda_1(0) = \frac{\boldsymbol{p}_1^T K(0) \boldsymbol{p}_1}{\|\boldsymbol{p}_1\|^2}, \quad \lambda_2(0) = \frac{\boldsymbol{p}_2^T K(0) \boldsymbol{p}_2}{\|\boldsymbol{p}_2\|^2}.$$

Take $\lambda_1(0)$ as an example:

$$\begin{aligned}
\lambda_1(0)\|\boldsymbol{p}_1\|^2 &= \sum_{i,j=1}^{n} x_i x_j \mathbb{1}_{\{x_i \geq 0\}} \mathbb{1}_{\{x_j \geq 0\}} \sum_{k=1}^{m} (u_{0,k}^2 + v_{0,k}^2) x_i x_j \mathbb{1}_{\{u_{0,k} x_i \geq 0\}} \mathbb{1}_{\{u_{0,k} x_j \geq 0\}} \\
&= \sum_{k=1}^{m} (u_{0,k}^2 + v_{0,k}^2) \left(\mathbb{1}_{\{u_{0,k} \geq 0\}} \right)^2 \sum_{i,j=1}^{n} x_i^2 x_j^2 \mathbb{1}_{\{x_i \geq 0\}} \mathbb{1}_{\{x_j \geq 0\}}.
\end{aligned}$$

Similar to the proof of Proposition B.9.1, we consider $\xi_k := v_{0,k} \mathbb{1}_{\{u_{0,k} \geq 0\}}$ which is a sub-gaussian random variable. Hence ξ_k^2 is sub-exponential so that $\|\xi_k^2\|_{\psi_1} \leq C$ where $C > 0$ is an absolute constant. By Bernstein's inequality, for every $t \geq 0$, we have

$$\mathbb{P}\left\{ \left| \sum_{i=1}^{m} \xi_i^2 - \frac{m}{2} \right| \geq t \right\} \leq 2\exp\left(-c \min\left(\frac{t^2}{\sum_{i=1}^{m} \|\xi_i^2\|_{\psi_1}^2}, \frac{t}{\max_i \|\xi_i^2\|_{\psi_1}} \right) \right),$$

where $c > 0$ is an absolute constant.

Letting $t = m/4$, we have with probability at least $1 - 2\exp(-m/c')$,

$$\frac{m}{4} \leq \sum_{i=1}^{m} \xi_i^2 \leq \frac{3m}{4},$$

where $c' = c/(4C)$.

The same analysis applies to $\zeta_k := u_{0,k} \mathbb{1}_{\{u_{0,k} \geq 0\}}$ as well and we have with probability at

least $1-2\exp(-m/c')$,

$$\frac{m}{4} \leq \sum_{i=1}^{m} \zeta_i^2 \leq \frac{3m}{4}.$$

As a result, we have probability at least $1-4\exp(-m/c')$,

$$\begin{aligned}\lambda_1(0)\|\boldsymbol{p}_1\|^2 &= \frac{1}{m}\sum_{i=k}^{m}(u_{0,k}^2+v_{0,k}^2)\left(\mathbb{1}_{\{u_k(0)\geq 0\}}\right)^2 \sum_{i,j=1}^{n} x_i^2 x_j^2 \mathbb{1}_{\{x_i\geq 0\}}\mathbb{1}_{\{x_j\geq 0\}} \\ &\in \left[\frac{1}{2}\sum_{i,j=1}^{n} x_i^2 x_j^2 \mathbb{1}_{\{x_i\geq 0\}}\mathbb{1}_{\{x_j\geq 0\}}, \frac{3}{2}\sum_{i,j=1}^{n} x_i^2 x_j^2 \mathbb{1}_{\{x_i\geq 0\}}\mathbb{1}_{\{x_j\geq 0\}}\right].\end{aligned}$$

Applying the same analysis to $\lambda_2(0)$, we have with probability $1-4\exp(-m/c')$,

$$\begin{aligned}\lambda_2(0)\|\boldsymbol{p}_2\|^2 &= \frac{1}{m}\sum_{i=k}^{m}(u_{0,k}^2+v_{0,k}^2)\left(\mathbb{1}_{\{u_k(0)\leq 0\}}\right)^2 \sum_{i,j=1}^{n} x_i^2 x_j^2 \mathbb{1}_{\{x_i\leq 0\}}\mathbb{1}_{\{x_j\leq 0\}} \\ &\in \left[\frac{1}{2}\sum_{i,j=1}^{n} x_i^2 x_j^2 \mathbb{1}_{\{x_i\leq 0\}}\mathbb{1}_{\{x_j\leq 0\}}, \frac{3}{2}\sum_{i,j=1}^{n} x_i^2 x_j^2 \mathbb{1}_{\{x_i\leq 0\}}\mathbb{1}_{\{x_j\leq 0\}}\right].\end{aligned}$$

The largest eigenvalue is $\max\{\lambda_1(0),\lambda_2(0)\}$. Combining the results together, we have with probability at least $1-4\exp(-m/c')$,

$$\frac{1}{2}M \leq \|K(0)\| \leq \frac{3}{2}M,$$

where $M = \max\left\{\frac{\sum_{i,j=1}^{n} x_i^2 x_j^2 \mathbb{1}\{x_i\geq 0\}\mathbb{1}\{x_j\geq 0\}}{\sum_{i=1}^{n} x_i^2 \mathbb{1}\{x_i\geq 0\}}, \frac{\sum_{i,j=1}^{n} x_i^2 x_j^2 \mathbb{1}\{x_i\leq 0\}\mathbb{1}\{x_j\leq 0\}}{\sum_{i=1}^{n} x_i^2 \mathbb{1}\{x_i\leq 0\}}\right\}$. □

B.11 Analysis on optimization dynamics for multiple training examples

In this section, we discuss the optimization dynamics for multiple training examples. We will see that by confining the dynamics into each eigendirection of the tangent kernel, the

training dynamics is similar to that for a single training example.

Since x_i is a scalar for all $i \in [n]$, with the homogeneity of ReLU activation function, we can compute the exact eigenvectors of $K(t)$ for all $t \geq 0$. To that end, we group the data into two sets $\mathscr{S}_+$ and $\mathscr{S}_-$ according to their sign:

$$\mathscr{S}_+ := \{i : x_i \geq 0, i \in [n]\}, \quad \mathscr{S}_- := \{i : x_i < 0, i \in [n]\}.$$

Now we have the proposition for the tangent kernel K(the proof is deferred to Appendix B.8):

Proposition B.11.1 (Eigenvectors and low rank structure of K)**.** *For any* $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$, $\operatorname{rank}(K) \leq 2$. *Furthermore,* $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ *are eigenvectors of K, where* $p_{1,i} = x_i \mathbb{1}_{\{i \in \mathscr{S}_+\}}$, $p_{2,i} = x_i \mathbb{1}_{\{i \in \mathscr{S}_-\}}$, *for* $i \in [n]$.

Note that when all x_i are of the same sign, $\operatorname{rank}(K) = 1$ and K only has one eigenvector (either $\boldsymbol{p}_1$ or $\boldsymbol{p}_2$ depending on the sign). It is in fact a simpler setting since we only need to consider one direction, whose analysis is covered by the one for $\operatorname{rank}(K) = 2$. Therefore, in the following we will assume $\operatorname{rank}(K) = 2$. We denote two eigenvalues of $K(t)$ by $\lambda_1(t)$ and $\lambda_2(t)$ corresponding to $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ respectively, i.e., $K(t)\boldsymbol{p}_1 = \lambda_1(t)\boldsymbol{p}_1$, $K(t)\boldsymbol{p}_2 = \lambda_2(t)\boldsymbol{p}_2$. Without loss of generality, we assume $\lambda_1(0) \geq \lambda_2(0)$.

By Eq. (3.5), the tangent kernel K at step t is defined as:

$$\begin{aligned} K_{i,j}(t) &= \langle \nabla_{\mathbf{v}} g_i(t), \nabla_{\mathbf{v}} g_j(t) \rangle + \langle \nabla_{\mathbf{u}} g_i(t), \nabla_{\mathbf{u}} g_j(t) \rangle \\ &= \frac{1}{m} \sum_{k=1}^{m} \left((u_k(t))^2 + (v_k(t))^2 \right) x_i x_j \mathbb{1}_{\{u_k(0) x_i \geq 0\}} \mathbb{1}_{\left\{u_k(0) x_j \geq 0\right\}}, \quad \forall i, j \in [n]. \end{aligned}$$

Similar to single example case, the largest eigenvalue of of tangent kernel is bounded from 0:

Proposition B.11.2. *For any* $\delta \in (0,1)$, *if* $m \geq c' \log(4/\delta)$ *where* c' *is an absolute constant, with*

probability at least $1-\delta$, $M/2 \le \lambda_{\max}(K(0)) \le 3M/2$ *where*

$$M = \max\left\{ \frac{\sum_{i,j=1}^n x_i^2 x_j^2 \mathbb{1}_{\{x_i \ge 0\}} \mathbb{1}_{\{x_j \ge 0\}}}{\sum_{i=1}^n x_i^2 \mathbb{1}_{\{x_i \ge 0\}}}, \frac{\sum_{i,j=1}^n x_i^2 x_j^2 \mathbb{1}_{\{x_i \le 0\}} \mathbb{1}_{\{x_j \le 0\}}}{\sum_{i=1}^n x_i^2 \mathbb{1}_{\{x_i \le 0\}}} \right\}.$$

The proof can be found in Appendix B.10.

For the simplicity of notation, given $\boldsymbol{p}, \boldsymbol{m} \in \mathbb{R}^n$, we define the matrices $K_{\boldsymbol{p},\boldsymbol{m}}$and $Q_{\boldsymbol{p},\boldsymbol{m}}$:

$$K_{\boldsymbol{p},\boldsymbol{m}}(t) := \left((\mathbf{g}(t)-\boldsymbol{y}) \odot \boldsymbol{m}\right)^T K(t) \left((\mathbf{g}(t)-\boldsymbol{y}) \odot \boldsymbol{m}\right) \boldsymbol{p}\boldsymbol{p}^T,$$
$$Q_{\boldsymbol{p},\boldsymbol{m}}(t) := \left((\mathbf{g}(t)-\boldsymbol{y}) \odot \boldsymbol{m}\right)^T \left(\mathbf{g}(t) \odot \boldsymbol{m}\right) \boldsymbol{p}\boldsymbol{p}^T$$

It is not hard to see that for all t, $K_{\boldsymbol{p},\boldsymbol{m}}$ and $Q_{\boldsymbol{p},\boldsymbol{m}}$ are rank-1 matrices. Specially, $\boldsymbol{p}$ is the only eigenvector of $K_{\boldsymbol{p},\boldsymbol{m}}$ and $Q_{\boldsymbol{p},\boldsymbol{m}}$.

With the above notations, we can write the update equations for $\mathbf{g}(t)-\boldsymbol{y}$ and $K(t)$ during gradient descent with learning rate η:

Dynamics equations.

$$\mathbf{g}(t+1)-\boldsymbol{y} = \left(I - \eta K(t) + \underbrace{\frac{\eta^2}{m}\left(Q_{\boldsymbol{p}_1,\boldsymbol{m}_+}(t) + Q_{\boldsymbol{p}_2,\boldsymbol{m}_-}(t)\right)}_{R_{\mathbf{g}}(t)} \right) (\mathbf{g}(t)-\boldsymbol{y}), \tag{B.7}$$

$$K(t+1) = K(t) + \underbrace{\frac{\eta^2}{m}\left(K_{\boldsymbol{p}_1,\boldsymbol{m}_+}(t) + K_{\boldsymbol{p}_2,\boldsymbol{m}_-}(t)\right) - \frac{4\eta}{m}\left(Q_{\boldsymbol{p}_1,\boldsymbol{m}_+}(t) + Q_{\boldsymbol{p}_2,\boldsymbol{m}_-}(t)\right)}_{R_K(t)}, \tag{B.8}$$

where $\boldsymbol{m}_+, \boldsymbol{m}_- \in \mathbb{R}^n$ are mask vectors:

$$m_{+,i} = \mathbb{1}_{\{i \in \mathscr{S}_+\}}, \quad m_{-,i} = \mathbb{1}_{\{i \in \mathscr{S}_-\}}.$$

Now we are ready to discuss different three optimization dynamics for multiple training examples case, similar to the single training example case in the following.

Monotonic convergence: sub-critical learning rates ($\eta < 2/\lambda_1(0)$)

We use the key observation that when $\|\mathbf{g}(t)\|$ is small, i.e., $O(1)$, and $\|K(t)\|$ is bounded, then $\|R_{\mathbf{g}}(t)\|$ and $\|R_K(t)\|$ are of the order $o(1)$. Then the dynamics equations approximately reduce to the ones of linear dynamics for multiple training examples:

$$\mathbf{g}(t+1) - \boldsymbol{y} = (I - \eta K(t) + o(1))(\mathbf{g}(t) - \boldsymbol{y}),$$
$$K(t+1) = K(t) + o(1).$$

At initialization, $\|\mathbf{g}(0)\| = O(1)$ with high probability over random initialization. By the choice of the learning rate, we will have for all $t \geq 0$, $\|I - \eta K(t)\| < 2$, hence $\|\mathbf{g}(t) - \boldsymbol{y}\|$ decreases exponentially. The cumulative change on the norm of tangent kernel is $o(1)$ since $\|R_K(t)\| = O(1/m)$ and the loss decreases exponentially hence $\sum \|R_K(t)\| = O(1/m) \cdot \log O(1) = o(1)$.

Catapult convergence: super-critical learning rates $\left(\frac{2}{\lambda_1(0)} < \eta < \min\left(\frac{2}{\lambda_2(0)}, \frac{4}{\lambda_1(0)}\right)\right)$

We summarize the catapult dynamics in the following:

Restate Theorem 3.3.6(Catapult dynamics on multiple training examples). *Supposing Assumption 3.3.5 holds, consider training the NQM Eq. (3.10) with squared loss on multiple training examples by GD. Then,*

1. *with $\eta \in \left[\frac{2+\varepsilon}{\lambda_1(0)}, \frac{2-\varepsilon}{\lambda_2(0)}\right]$, the catapult only occurs in eigendirection $\boldsymbol{p}_1$: $\Pi_1 \mathscr{L}$ increases to the order of $\Omega\left(\frac{m(\eta - 2/\lambda_1(0))^2}{\log m}\right)$ then decreases to $O(1)$;*
2. *with $\eta \in \left[\frac{2+\varepsilon}{\lambda_2(0)}, \frac{4-\varepsilon}{\lambda_1(0)}\right]$, the catapult occurs in both eigendirections $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$: $\Pi_i \mathscr{L}$ for $i = 1, 2$ increases to the order of $\Omega\left(\frac{m(\eta - 2/\lambda_i(0))^2}{\log m}\right)$ then decreases to $O(1)$,*

where $\varepsilon = \Theta\left(\frac{\log m}{\sqrt{m}}\right)$.

The proof can be found in Appendix B.12.

For the remaining eigendirections $\boldsymbol{p}_3, \cdots, \boldsymbol{p}_n$, i.e., the basis of the subspace orthogonal to $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, we can show that the loss projected to this subspace does not change during training in the following proposition. It follows from the fact that K, $R_{\mathbf{g}}(t)$ and $R_K(t)$ are orthogonal to $\boldsymbol{p}_i \boldsymbol{p}_i^T$ for $i = 3, \cdots, n$.

Proposition B.11.3. *$\forall t \geq 0$, $\Pi_i \mathscr{L}(t) = \Pi_i \mathscr{L}(0)$ for $i = 3, \cdots, n$.*

Once the catapult finishes as the loss decreases to the order of $O(1)$, we generally have $\eta > 2/\lambda_1$ and $\eta > 2/\lambda_2$. Therefore the training dynamics fall into linear dynamics, and we can use the same analysis for sub-critical learning rates for the remaining training dynamics.

Divergence: ($\eta > \eta_{\max} = 4/\lambda_1(0)$)

Similar to the increasing phase in the catapult convergence, initially $\|\mathbf{g}(t) - \boldsymbol{y}\|$ increases in direction $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ since linear dynamics dominate and the learning rate is chosen to be larger than η_{crit}. Also, we approximately have $\eta > 4/\lambda_1(t)$ at the end of the increasing phase, by a similar analysis for the catapult convergence. We consider the evolution of $K(t)$ in the direction $\boldsymbol{p}_1$. Note that when $\|\mathbf{g}(t)\|$ increases to the order of $\Theta(\sqrt{m})$, $\mathbf{g}(t) \odot \boldsymbol{m}_+$ will be aligned with $\boldsymbol{p}_1$, hence with simple calculation, we approximately have

$$\boldsymbol{p}_1^T R_K(t) \boldsymbol{p}_1 \approx \frac{\|\mathbf{g}(t)\|^2 \|\boldsymbol{p}_1\|^2}{m} \eta(\lambda_1(t) - 4\eta) > 0.$$

Therefore, $\lambda_1(t)$ increases since $\boldsymbol{p}_1^T K(t+1) \boldsymbol{p}_1 = \boldsymbol{p}_1^T K(t) \boldsymbol{p}_1 + \boldsymbol{p}_1^T R_K(t) \boldsymbol{p}_1 > \boldsymbol{p}_1^T K(t) \boldsymbol{p}_1$. As a result, $\|I - \eta K(t) + R_{\mathbf{g}}(t)\|$ becomes even larger which makes $\|\mathbf{g}(t) - \boldsymbol{y}\|$ grows faster, and ultimately leads to divergence of the optimization.

B.12 Proof of Theorem 3.3.6

As the tangent kernel K has rank 2 by Proposition B.11.1, the update of weight parameters $\mathbf{w}$ is in a subspace with dimension 2. Specifically,

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \frac{\partial \mathbf{g}}{\partial \mathbf{w}} \frac{\partial \mathscr{L}}{\partial \mathbf{g}}(t),$$

where $\partial \mathbf{g}/\partial \mathbf{w}$ has rank 2. Therefore, to understand the whole training dynamics, it is sufficient to analyze the dynamics of the loss in eigendirection $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$.

We will analyze the dynamics of the loss $\mathscr{L}$ and the tangent kernel K confined to $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$. It turns out that the dynamics in each eigen direction is almost independent on the other hence can be reduced to the same training dynamics for a single training example.

We start with eigendirection $\boldsymbol{p}_1$. For dynamics equations Eq. (B.7) and (B.8), we consider the training dynamics confined to direction $\boldsymbol{p}_1$ and we have

$$\begin{aligned}
\Pi_1 \mathscr{L}(t) &= \left(1 - \eta \lambda_1(t) + \boldsymbol{p}_1^T R_{\mathbf{g}}(t) \boldsymbol{p}_1\right)^2 \Pi_1 \mathscr{L}(t) := \kappa_1(t) \Pi_1 \mathscr{L}(t), \\
\lambda_1(t+1) &= \lambda_1(t) + \boldsymbol{p}_1^T R_K(t) \boldsymbol{p}_1,
\end{aligned}$$

where we use the notation $\Pi_1 \mathscr{L}(t) = \frac{1}{2} \langle \mathbf{g}(t) - \boldsymbol{y}, \boldsymbol{p}_1 \rangle^2$.

We further expand $\boldsymbol{p}_1^T R_{\mathbf{g}}(t) \boldsymbol{p}_1$ and $\boldsymbol{p}_1^T R_K(t) \boldsymbol{p}_1$ and we have

$$\begin{aligned}
\boldsymbol{p}_1^T R_{\mathbf{g}}(t) \boldsymbol{p}_1 &= \frac{2\eta^2}{m} \Pi_1 \mathscr{L}(t) + \frac{\eta^2}{m} \left\langle (\mathbf{g}(t) - \boldsymbol{y}) \odot \boldsymbol{m}_+, \boldsymbol{y} \odot \boldsymbol{m}_+ \right\rangle, \\
\boldsymbol{p}_1^T R_K(t) \boldsymbol{p}_1 &= \frac{2\eta}{m} \Pi_1 \mathscr{L}(t) (\eta \lambda_1(t) - 4) - \frac{4\eta}{m} \left\langle (\mathbf{g}(t) - \boldsymbol{y}) \odot \boldsymbol{m}_+, \boldsymbol{y} \odot \boldsymbol{m}_+ \right\rangle.
\end{aligned}$$

Analogous to the transformation for Eq. (3.11) and (3.12) as we have done in the proof of

Theorem 3.3.1, we let

$$u_1(t) = \frac{2\eta^2}{m}\Pi_1 \mathscr{L}(t), \quad w_1(t) = \frac{\eta^2}{m}\left\langle (\mathbf{g}(t) - \boldsymbol{y}) \odot \boldsymbol{m}_+, \boldsymbol{y} \odot \boldsymbol{m}_+ \right\rangle, \quad v_1(t) = \eta \lambda_1(t).$$

Then the dynamic equations can be written as:

$$u_1(t+1) = (1 - v_1(t) + u_1(t) + w_1(t))^2 u_1(t), \tag{B.9}$$

$$v_1(t+1) = v_1(t) - u_1(t)(4 - v_1(t)) - 4w_1(t). \tag{B.10}$$

Note that at initialization, $\|\mathbf{g}(t)\| = O(\sqrt{1})$ with high probability, hence we have $u_1(0) = O\left(\frac{1}{m}\right)$ and $w_1(0) = O\left(\frac{1}{m}\right)$ (we omit the factor n as n is a constant). Furthermore, $|w_1(t)| = \Theta\left(\frac{\sqrt{u_1(t)}}{\sqrt{m}}\right)$. Therefore, both the dynamic equations and the initial condition are exactly the same with the ones for a single training example (Eq. (3.13) and (3.14)). Then we can follow the same idea of the proof of Theorem 3.3.1 to show the catapult in eigendirection $\boldsymbol{p}_1$.

Similarly, when we consider the training dynamics confined to $\boldsymbol{p}_2$, we have

$$u_2(t+1) = (1 - v_2(t) + u_2(t) + w_2(t))^2 u_2(t), \tag{B.11}$$

$$v_2(t+1) = v_2(t) - u_2(t)(4 - v_2(t)) - 4w_2(t), \tag{B.12}$$

where

$$u_2(t) = \frac{2\eta^2}{m}\Pi_2 \mathscr{L}(t), \quad w_2(t) = \frac{\eta^2}{m}\left\langle (\mathbf{g}(t) - \boldsymbol{y}) \odot \boldsymbol{m}_-, \boldsymbol{y} \odot \boldsymbol{m}_- \right\rangle, \quad v_2(t) = \eta \lambda_2(t).$$

Then the same analysis with Theorem 3.3.1 can be used to show the catapult in direction $\boldsymbol{p}_2$.

Note that when $2/\lambda_2(0) > 4/\lambda_1(0)$, the learning rate is only allowed to be less than $4/\lambda_1(0)$ otherwise GD will diverge, therefore, there will be no catapult in direction $\boldsymbol{p}_2$.

B.13 Special case of quadratic models when $\phi(\boldsymbol{x}) = 0$

In this section we will show under some special settings, the catapult phase phenomenon also happens and how two layer linear neural networks fit in our quadratic model.

We consider one training example $(\boldsymbol{x}, y)$ with label $y = 0$ and assume the initial tangent kernel $\lambda(0) = \Omega(1)$. Letting the feature vector $\phi(\boldsymbol{x}) = 0$, the quadratic model Eq.(3.3) becomes:

$$g(\mathbf{w}) = \frac{1}{2}\gamma \mathbf{w}^T \Sigma(\boldsymbol{x}) \mathbf{w}.$$

For this quadratic model, we have the following proposition:

Proposition B.13.1. *With learning rate* $\frac{2}{\lambda(0)} < \eta < \frac{4}{\lambda(0)}$, *if* $\Sigma(\boldsymbol{x})^2 = \|\boldsymbol{x}\|^2 \cdot I$, $g(\mathbf{w})$ *exhibits catapult phase.*

Proof. With simple computation, we get

$$g(t+1) = \left(1 - \eta\lambda(t) + \gamma\eta^2 \|\boldsymbol{x}\|^2 (g(t))^2\right) g(t),$$
$$\lambda(t+1) = \lambda(t) - \gamma\|\boldsymbol{x}\|^2 (g(t))^2 (4 - \eta\lambda(t)).$$

We note that the evolution of g and λ is almost the same with Eq. (3.11) and Eq. (3.12) if we regard $\gamma = 1/m$. Hence we can apply the same analysis to show the catapult phase phenomenon. □

It is worth pointing out that the two-layer linear neural network with input $\boldsymbol{x} \in \mathbb{R}^d$ analyzed in [65] that

$$f(\mathbf{U}, \mathbf{v}; x) = \frac{1}{\sqrt{m}} \mathbf{v}^T \mathbf{U} \boldsymbol{x},$$

where $\mathbf{v} \in \mathbb{R}^m, \mathbf{U} \in \mathbb{R}^{m \times d}$ is a special case of our model with $\mathbf{w} = \left[\mathrm{Vec}(\mathbf{U})^T, \mathbf{v}^T\right]^T$, $\gamma = 1/\sqrt{m}$

and

$$\Sigma = \begin{pmatrix} 0 & I_m \otimes \boldsymbol{x} \\ I_m \otimes \boldsymbol{x}^T & 0 \end{pmatrix} \in \mathbb{R}^{md+m}.$$

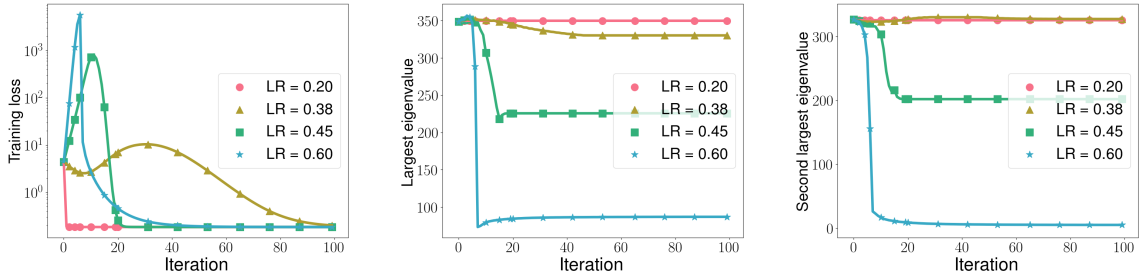
B.14 Experimental settings and additional results

Verification of non-linear training dynamics of NQMs, i.e., Figure 3.3

We train the NQM which approximates the two-layer fully-connected neural network with ReLU activation function on 128 data points where each input is drawn i.i.d. from $\mathcal{N}(-2,1)$ if the label is -1 or $\mathcal{N}(2,1)$ if the label is 1. The network width is $5,000$.

Experiments for training dynamics of wide neural networks with multiple examples

We train a two-layer fully-connected neural network with ReLU activation function on 128 data points where each input is drawn i.i.d. from $\mathcal{N}(-2,1)$ if the label is -1 or $\mathcal{N}(2,1)$ if the label is 1. The network width is $5,000$. See the results in Figure B.1.



(a) Training loss

(b) Largest eigenvalue of tangent kernel

(c) Second largest eigenvalue of tangent kernel

Figure B.1. Training dynamics of wide neural networks for multiple examples case with different learning rates. Compared to the training dynamics of NQMs, i.e., Figure 3.3, the behaviour of of top eigenvalues is almost the same with different learning rates: when $\eta < 0.37$, the kernel is nearly constant; when $0.37 < \eta < 0.39$, only $\lambda_1(t)$ decreases; when $0.39 < \eta < \eta_{\max}$, both $\lambda_1(t)$ and $\lambda_2(t)$ decreases. See the experiment setting in Appendix B.14.

Training dynamics confined to top eigenspace of the tangent kernel

We consider the corresponding dynamics equations (3.15) and (3.16) for neural networks:

$$\mathbf{f}(t+1) - \boldsymbol{y} = (I - \eta K(t) + R_{\mathbf{f}}(t))\,(\mathbf{f}(t) - \boldsymbol{y}), \tag{B.13}$$

$$K(t+1) = K(t) - R_K(t). \tag{B.14}$$

Note that for NQMs, $R_{\mathbf{f}}(t)$ and $R_K(t)$ have closed-form expressions but generally for neural networks they do not have.

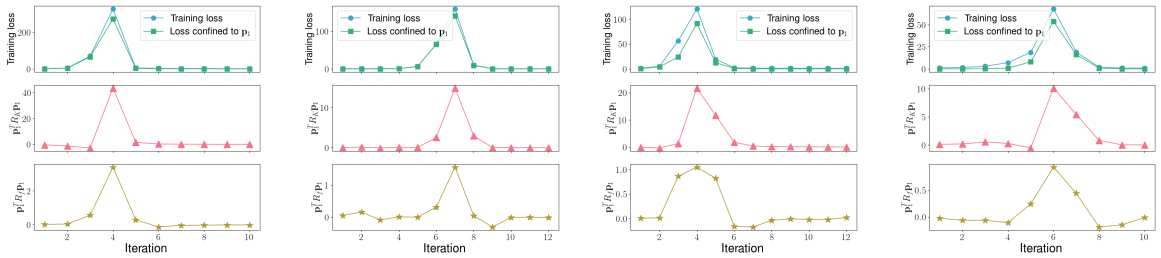
We consider the training dynamics confined to the top eigenvector of the tangent kernel $\boldsymbol{p}_1(t)$:

$$\langle \boldsymbol{p}_1(t), \mathbf{f}(t+1) - \boldsymbol{y} \rangle = \left(I - \eta \lambda_1(t) + \boldsymbol{p}_1(t)^T R_{\mathbf{f}}(t) \boldsymbol{p}_1(t)\right) \langle \boldsymbol{p}_1(t), \mathbf{f}(t) - \boldsymbol{y} \rangle,$$

$$\boldsymbol{p}_1(t)^T K(t+1) \boldsymbol{p}_1(t) = \lambda_1(t) - \boldsymbol{p}_1(t)^T R_K(t) \boldsymbol{p}_1(t).$$

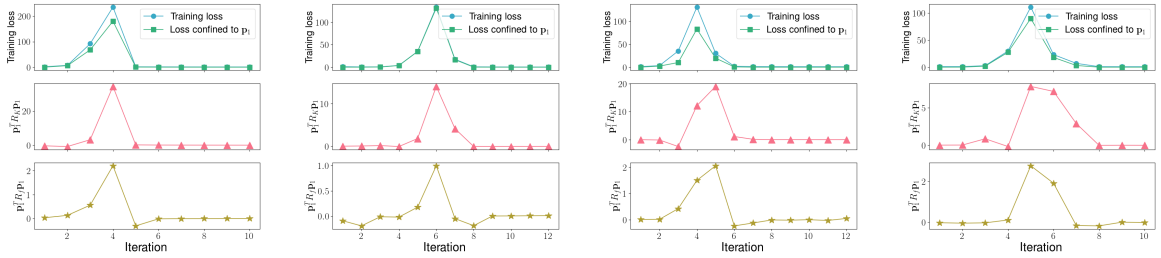
We conduct experiments to show that $\boldsymbol{p}_1(t)^T R_{\mathbf{f}}(t) \boldsymbol{p}_1(t)$ and $\boldsymbol{p}_1(t)^T R_K(t) \boldsymbol{p}_1(t)$ scale with the loss and remain positive when the loss is large. Furthermore, the loss confined to $\boldsymbol{p}_1$ can almost capture the spike in the training loss.

In the experiments, we train a two-layer FC and CNN with width 2048 and 1024 respectively on 128 points from CIFAR-2 (2 class subset of CIFAR-10) and SVHN-2 (2 class subset from SVHN-10). The results for NQM can be seen in Figure B.2 and for neural networks can be seen in Figure B.3.



(a) FC on CIFAR-2 **(b)** CNN on CIFAR-2 **(c)** FC on SVHN-2 **(d)** CNN on SVHN-2

Figure B.2. Training dynamics confined to the top eigenspace of the tangent kernel for NQMs.



(a) FC on CIFAR-2 **(b)** CNN on CIFAR-2 **(c)** FC on SVHN-2 **(d)** CNN on SVHN-2

Figure B.3. Training dynamics confined to the top eigenspace of the tangent kernel for wide neural networks.

Training dynamics of general quadratic models and neural networks

As discussed at the end of Section 3.3, a more general quadratic model can exhibit the catapult phase phenomenon. Specifically, we consider a general quadratic model:

$$g(\mathbf{w};\boldsymbol{x}) = \mathbf{w}^T \boldsymbol{\phi}(\boldsymbol{x}) + \frac{1}{2}\gamma \mathbf{w}^T \Sigma(\boldsymbol{x})\mathbf{w}.$$

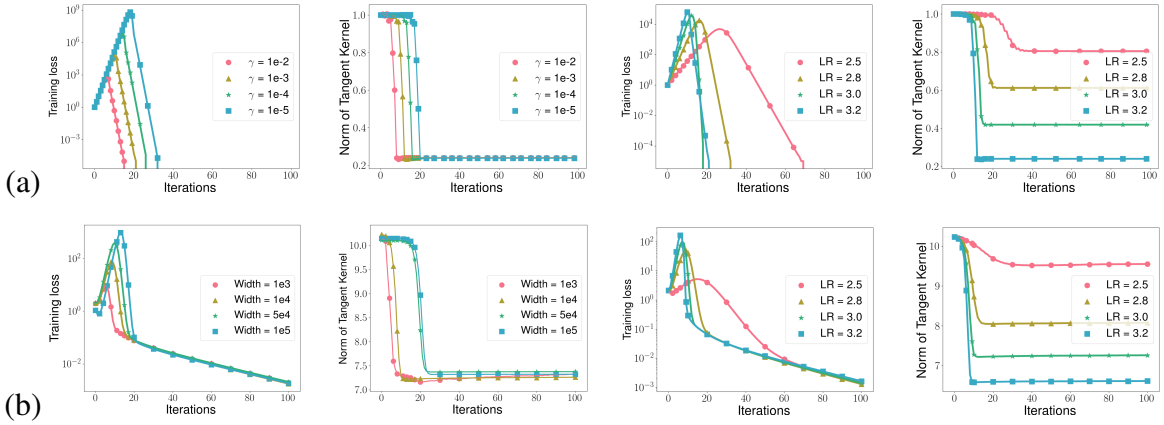
We will train the general quadratic model with different learning rates, and different γ respectively, to see how the catapult phase phenomenon depends on these two factors. For comparison, we also implement the experiments for neural networks. See the experiment setting in the following:

General quadratic models. We set the dimension of the input $d = 100$. We let the feature vector $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{x}/\|\boldsymbol{x}\|$ where $x_i \sim \mathcal{N}(0,1)$ i.i.d. for each $i \in [d]$. We let Σ be a diagonal

matrix with $\Sigma_{i,i} \in \{-1, 1\}$ randomly and independently. The weight parameters $\mathbf{w}$ are initialized by $\mathcal{N}(0, I_d)$. Unless stated otherwise, $\gamma = 10^{-3}$, and the learning rate is set to be 2.8.

Neural networks. We train a two-layer fully-connected neural networks with ReLU activation function on 20 data points of CIFAR-2. Unless stated otherwise, the network width is 10^4, and the learning rate is set to be 2.8.

See the results in Figure B.4.



(a) Loss (log scaled) vs. γ/width

(b) Tangent kernel norm vs. γ/width

(c) Loss vs. learning rate

(d) Tangent kernel norm vs. learning rate

Figure B.4. General quadratic models have similar training dynamics with neural networks when trained with super-critical learning rates. (a): experiments on general quadratic models. Smaller γ or larger learning rates lead to larger training loss at the peak. Larger learning rates make tangent kernel decrease more. (b): experiments on two-layer neural networks. Larger width (corresponding to smaller γ) and larger learning rates have similar effect on the training loss at the peak and decrease of tangent kernel norm with quadratic models. Note that width or γ seems to have no effect on the tangent kernel norm at convergence.

Test performance of f, f_{lin} and f_{quad}, i.e., Figure 3.2(b) and Figure 3.4

For the architectures of two-layer fully connected neural network and two-layer convolutional neural network, we set the width to be $5,000$ and $1,000$ respectively. Specific to Figure 3.2(b), we use the architecture of a two-layer fully connected neural network.

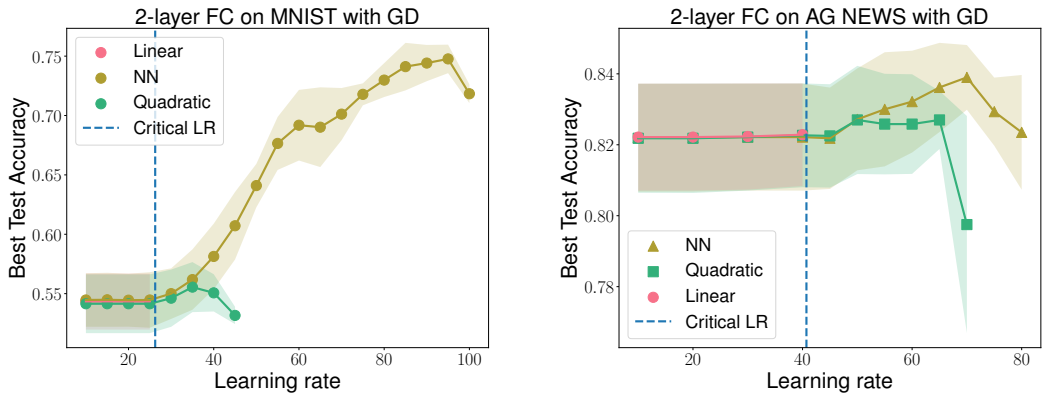
Due to the large number of parameters in NQMs, we choose a small subset of all the datasets. We use the first class (airplanes) and third class (birds) of CIFAR-10, which we call CIFAR-2, and select 256 data points out of it as the training set. We use the number 0 and 2 of

SVHN, and select 256 data points as the training set. We select 128, 256, 128 data points out of MNIST, FSDD and AG NEWS dataset respectively as the training sets. The size of testing set is $2,000$ for all. When implementing SGD, we choose batch size to be 32.

For each setting, we report the average result of 5 independent runs.

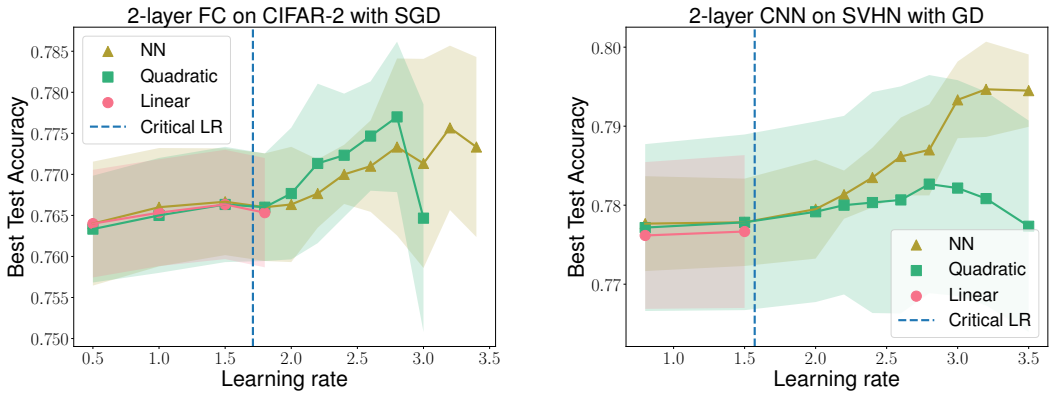
Test performance of f, f_{lin} and f_{quad} in terms of accuracy

In this section, we report the best test accuracy for f, f_{lin} and f_{quad} corresponding to the best test loss in Figure 3.4. We use the same setting as in Appendix B.14.



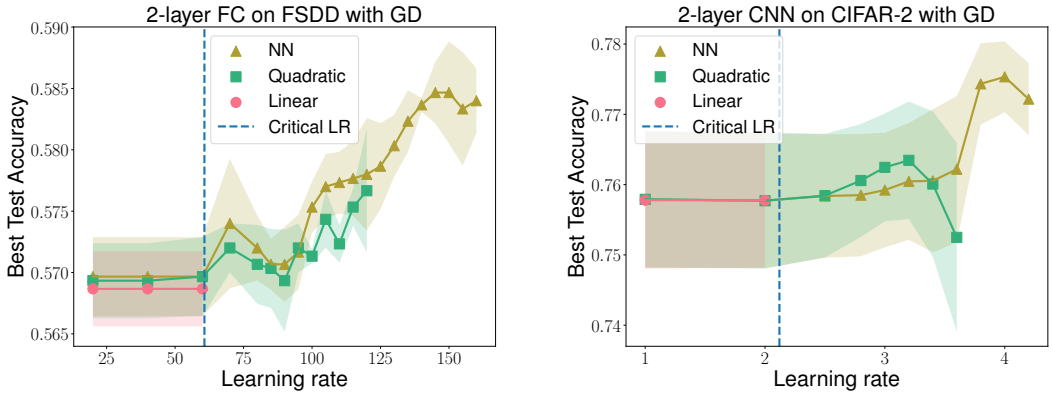
(a) **(b)**

Figure B.5. Best test accuracy plotted against different learning rates for f_{quad}, f, and f_{lin}. (a): 2-layer FC on MNIST trained with GD. (b): 2-layer FC on AG NEWS trained with GD.



(a) **(b)**

Figure B.6. Best test accuracy plotted against different learning rates for f_{quad}, f, and f_{lin}. (a): 2-layer FC on CIFAR-2 trained with SGD. (b): 2-layer CNN on SVHN trained with GD.



(a) **(b)**

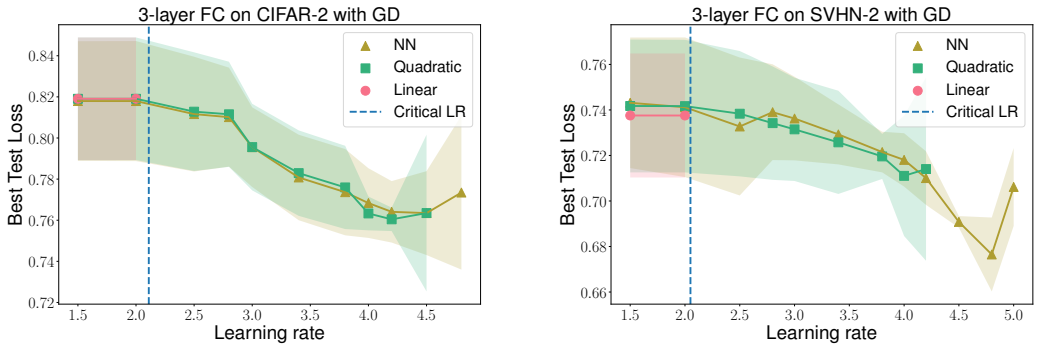
Figure B.7. Best test accuracy plotted against different learning rates for f_{quad}, f, and f_{lin}. (a): 2-layer FC on FSDD trained with GD. (b): 2-layer CNN on CIFAR-2 trained with GD.

Test performance of f, f_{lin} and f_{quad} with architecture of 3-layer FC

In this section, we extend our results for shallow neural networks discussed in Section 3.4 to 3-layer fully connected neural networks. In the same way, we compare the test performance of three models, f, f_{lin} and f_{quad} upon varying learning rate. We observe the same phenomenon for 3-layer ReLU activated FC with shallow neural networks. See Figure B.8 and B.9.

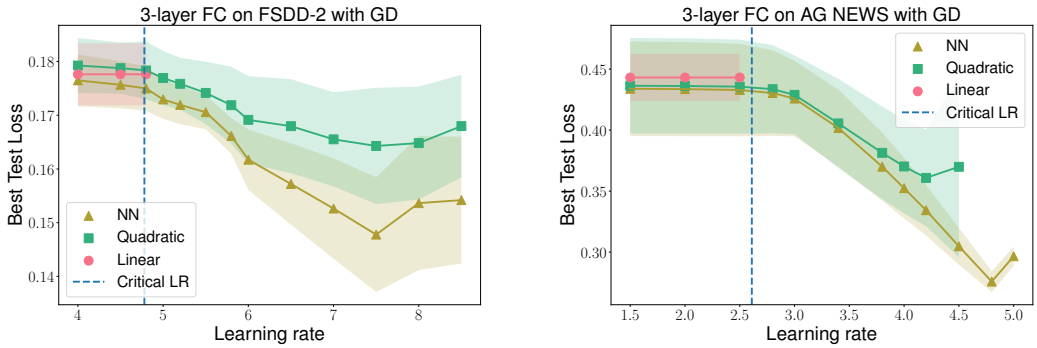
We use the first class (airplanes) and third class (birds) of CIFAR-10, which we call CIFAR-2, and select 100 data points out of it as the training set. We use the number 0 and 2 of SVHN, and select 100 data points as the training set. We select 100 data points out of AG NEWS dataset as the training set. For the speech data set FSDD, we select 100 data points in class 1 and 3 as the training set. The size of testing set is 500 for all.

For each setting, we report the average result of 5 independent runs.



(a) **(b)**

Figure B.8. Best test accuracy plotted against different learning rates for f_{quad}, f, and f_{lin}. (a): 3-layer FC on CIFAR-2 trained with GD. (b): 3-layer FC on SVHN-2 trained with GD.

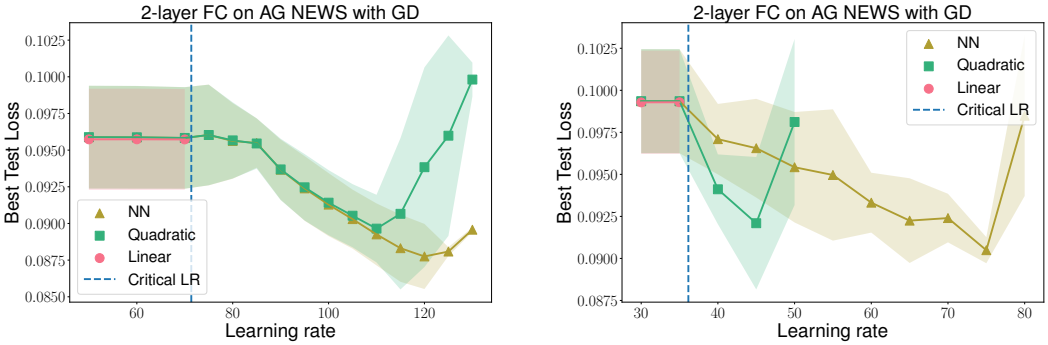


(a) **(b)**

Figure B.9. Best test accuracy plotted against different learning rates for f_{quad}, f, and f_{lin}. (a): 3-layer FC on FSDD-2 trained with GD. (b): 3-layer FC on AG NEWS trained with GD.

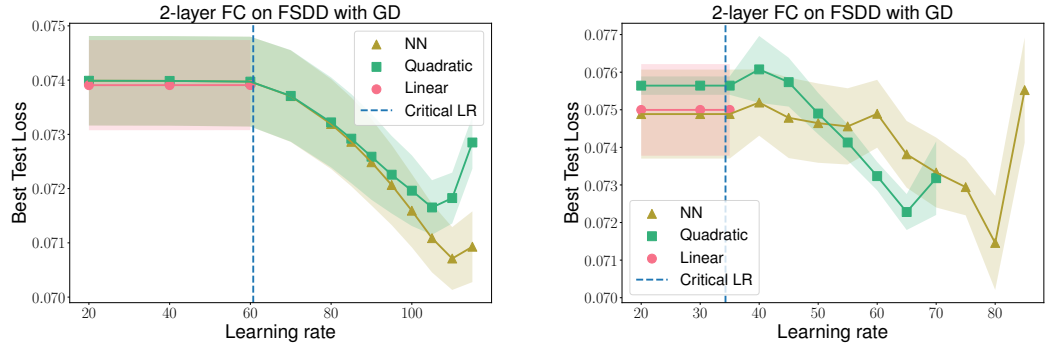
Test performance with Tanh and Swish activation functions

We replace ReLU by Tanh and Swish activation functions to train the models with the same setting as Figure 3.4. We observe the same phenomenon as we describe in Section 3.4.



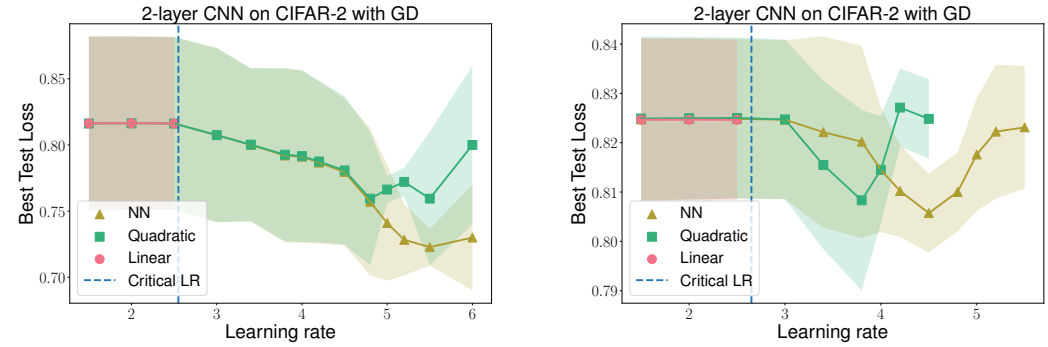
(a) Swish activation function **(b)** Tanh activation function

Figure B.10. Best test loss plotted against different learning rates for f_{quad}, f, and f_{lin}. We use 2-layer FC as the architecture and train all the models on AG NEWS with GD.



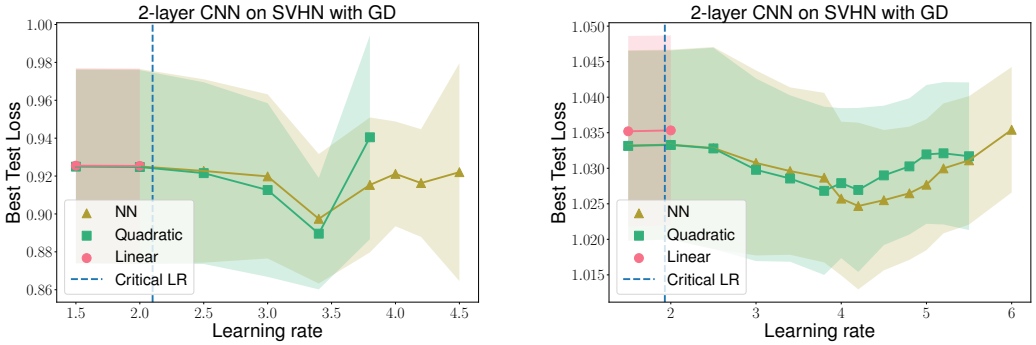
(a) Swish activation function **(b)** Tanh activation function

Figure B.11. Best test loss plotted against different learning rates for f_{quad}, f, and f_{lin}. We use 2-layer FC as the architecture and train all the models on FSDD with GD.



(a) Swish activation function **(b)** Tanh activation function

Figure B.12. Best test loss plotted against different learning rates for f_{quad}, f, and f_{lin}. We use 2-layer CNN as the architecture and train all the models on CIFAR-2 with GD.



(a) Swish activation function **(b)** Tanh activation function

Figure B.13. Best test loss plotted against different learning rates for f_{quad}, f, and f_{lin}. We use 2-layer CNN as the architecture and train all the models on SVHN with GD.

Appendix C

Chapter 4 Supplementary

C.1 The critical learning rate can be well approximated using NTK for wide neural networks

In this section, we show that the critical learning rate $\eta_{\text{crit}} := \frac{2}{\lambda_{\max}(H_{\mathscr{L}})}$ can be well approximated using NTK, i.e., $\tilde{\eta}_{\text{crit}} := \frac{n}{\lambda_{\max}(K)(\mathbf{w})}$. Note that $\|K\| = \lambda_{\max}(K)(\mathbf{w})$.

Approximation of the critical learning rate using NTK during training with a small constant learning rate

For MSE $\mathscr{L}(\mathbf{w};X) = \frac{1}{n}\sum_{i=1}^{n}(f(\mathbf{w};\mathbf{x}_i) - y_i)^2$, we can compute its $H_{\mathscr{L}}$ by the chain rule:

$$H_{\mathscr{L}}(\mathbf{w}) = \underbrace{\frac{2}{n}\sum_{i=1}^{n}\left(\frac{\partial f(\mathbf{w};\mathbf{x}_i)}{\partial \mathbf{w}}\right)^T \frac{\partial f(\mathbf{w};\mathbf{x}_i)}{\partial \mathbf{w}}}_{\mathscr{A}(\mathbf{w})} + \underbrace{\frac{2}{n}\sum_{i=1}^{n}(f(\mathbf{w};\mathbf{x}_i) - y_i)\frac{\partial^2 f(\mathbf{w};\mathbf{x}_i)}{\partial \mathbf{w}^2}}_{\mathscr{B}(\mathbf{w})}.$$

Assume $\|\boldsymbol{x}_i\| = O(1)$ and $|y_i| = O(1)$ for all $i \in [n]$. For $\mathscr{B}(\mathbf{w}_0)$, by random initialization of weights $\mathbf{w}_0$, with high probability, we have $|f(\mathbf{w}_0;\mathbf{x}_i) - y_i| = O(\log m)$, and $\left\|\frac{\partial^2 f(\mathbf{w}_0;\mathbf{x}_i)}{\partial \mathbf{w}^2}\right\|_2 = \tilde{O}(1/\sqrt{m})$ [66, 123] where m denotes the width of the network. Therefore, by the union bound, with high probability, we have $\mathscr{B}(\mathbf{w}_0) = \tilde{O}(1/\sqrt{m})$.

Note that $\lambda_{\max}(\mathscr{A}(\mathbf{w})) = \frac{2}{n}\lambda_{\max}(K(\mathbf{w}))$ for any $\mathbf{w}$. Combining all the bounds together,

we have $\left|\lambda_{\max}(H_{\mathscr{L}})(\mathbf{w}_0) - \frac{2}{n}\lambda_{\max}(K)(\mathbf{w}_0)\right| = \tilde{O}(1/\sqrt{m})$. Then we have

$$|\eta_{\text{crit}} - \tilde{\eta}_{\text{crit}}| = \left| \frac{2}{\lambda_{\max}(H_{\mathscr{L}})(\mathbf{w}_0)} - \frac{n}{\lambda_{\max}(K)(\mathbf{w}_0)} \right| = \tilde{O}(1/\sqrt{m})$$

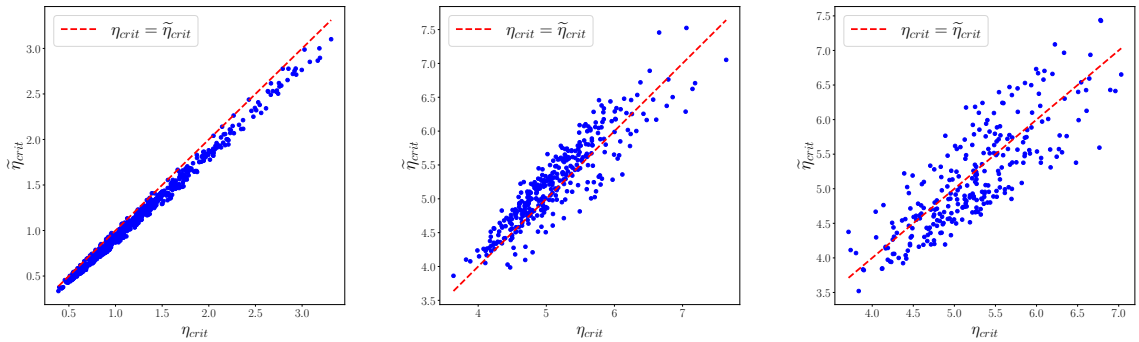
as long as $\lambda_{\max}(K)(\mathbf{w}_0) = \Omega(1)$, which is true with high probability over random initialization for wide networks [82, 12].

For wide neural networks trained with a small constant learning rate, $\left\| \frac{\partial^2 f(\mathbf{w}_0;\mathbf{x}_i)}{\partial \mathbf{w}^2} \right\|_2 = \tilde{O}(1/\sqrt{m})$ holds during the whole training process of GD/SGD, hence this approximation holds [66].

Approximation of the critical learning rate using NTK during training with a large learning rate

In this section, we provide further evidence for SGD that $\tilde{\eta}_{\text{crit}}$ approximates η_{crit} during training even with a large learning rate. Recall that $\tilde{\eta}_{\text{crit}} = b/\lambda_{\max}(K(\mathbf{w};X_{\text{batch}}))$ where b is the batch size. We consider the same network architectures as the shallow network in Figure 4.6 and deep networks in Figure 4.7.

We can see Figure C.1 shows that η_{crit} is close to $\tilde{\eta}_{\text{crit}}$ during training with SGD.



(a) Shallow network **(b)** 5-layer FCN **(c)** 5-layer CNN

Figure C.1. Validation of $\eta_{\text{crit}} \approx \tilde{\eta}_{\text{crit}}$ during SGD with catapults. Plot of points $(\eta_{\text{crit}}, \tilde{\eta}_{\text{crit}})$ at each iteration of SGD for the shallow network, 5-layer FCN and CNN. The models are trained on 128 data points from CIFAR-10 by SGD with batch size 32. The settings are the same with Table 4.2.

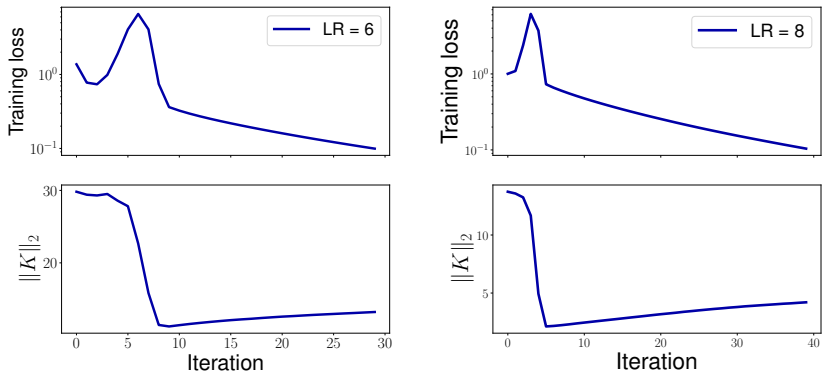
C.2 Additional experiments for the catapult in GD

Catapults occur in the top eigenspace of NTK

In this section, we provide additional empirical evidence to justify Claim 1. In particular, we consider three neural network architectures: a 5-layer Fully Connected Neural Network (FCN), a 5-layer Convolutional Neural Network (CNN), and Wide ResNets 10-10; and three datasets CIFAR-10, SVHN, and a synthetic dataset. For the synthetic dataset, we consider the rank-2 regression task with training size 128.

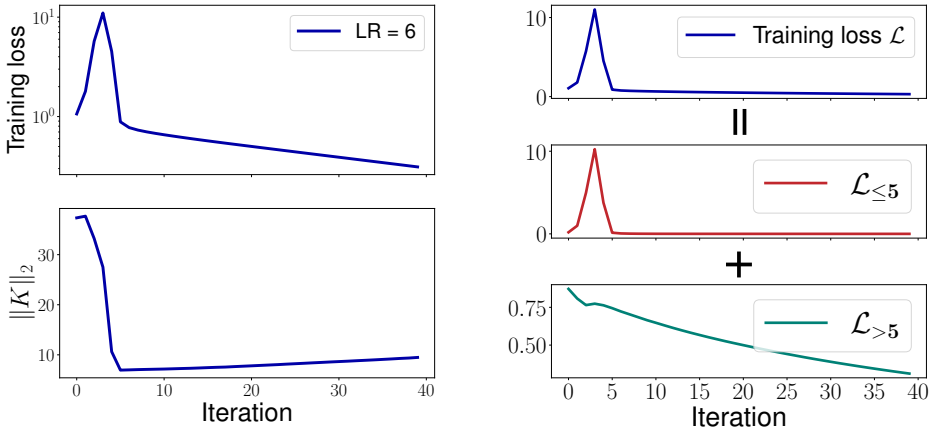
From the experimental results, we can see that for a large learning rate that causes catapult dynamics, the loss spike occurs in the top eigenspace of the tangent kernel. See Figure C.2 for 5-layer FCN and CNN on CIFAR-10 dataset and C.3 one SVHN dataset, and C.4 for Wide-ResNets on CIFAR-10 dataset.

We further show Claim 1 holds for multidimensional outputs in Figure C.5. In particular, for k-class classification tasks, we project the flattened vector of predictions of size kn to the top eigenspaces of the empirical NTK, which is of size $kn \times kn$. Correspondingly, we empirically observe that catapults occur in the top ks eigenspace with a small s.



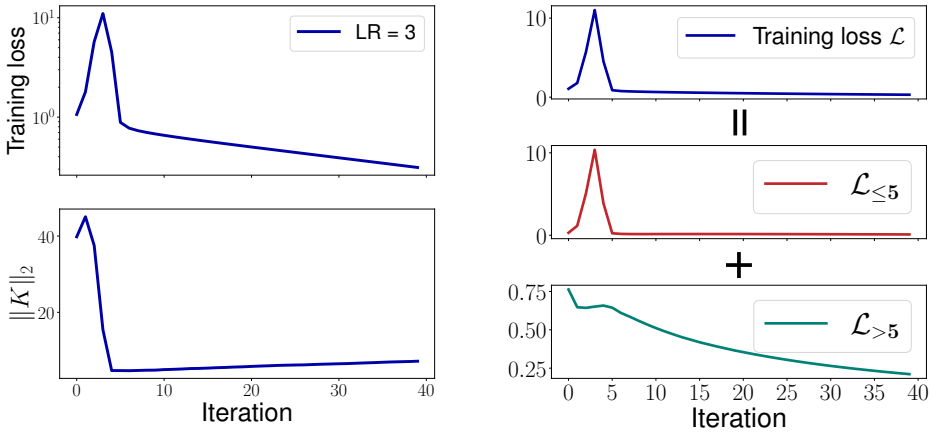
(a) 5-layer FCN **(b)** 5-layer CNN

Figure C.2. The training loss and the spectral norm of the tangent kernel during catapult for 5-layer FCN (a) and CNN (b) on CIFAR-10 dataset. Both networks are trained under the same experimental setting with Figure 4.3.



(a) Training loss(FCN)

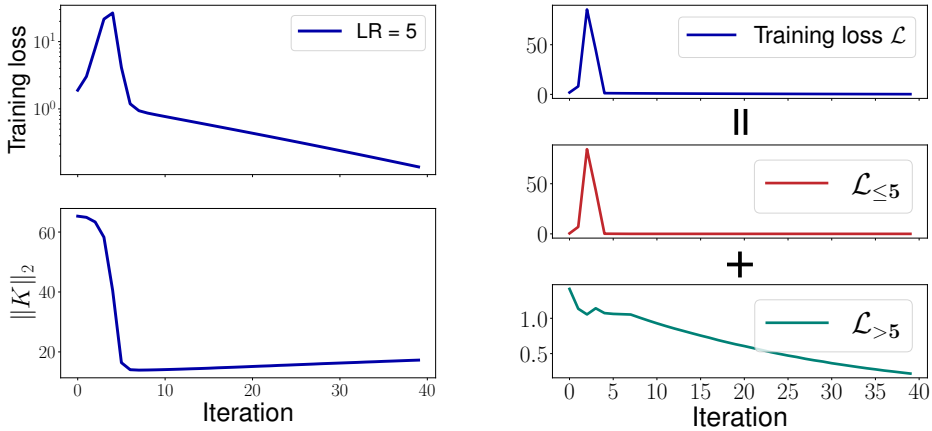
(b) Loss decomposition



(c) Training loss(CNN)

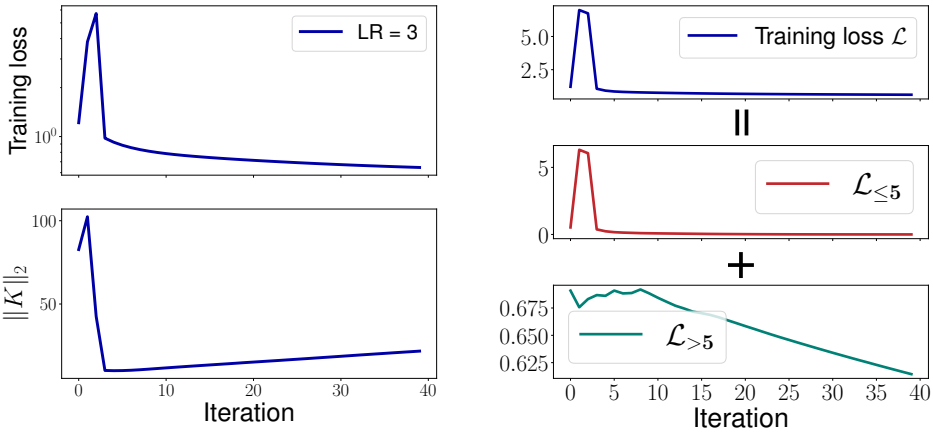
(d) Loss decomposition

Figure C.3. Catapult dynamics for 5-layer FCN (a-b) and CNN (c-d) on SVHN dataset. Panel (a) and (c) are the training loss and the spectral norm of the tangent kernel with learning rate 6.0 and 3.0 respectively, and Panel (b) and (d) are the training loss decomposed into the top eigendirections of the tangent kernel, $\mathscr{L}_{\leq 5}$ and the remaining eigendirections, $\mathscr{L}_{>5}$. All the networks are trained on a subset of SVHN with 128 data points. In this experiment, the critical learning rates for FCN and CNN are 3.4 and 1.6 respectively.



(a) Training loss(FCN)

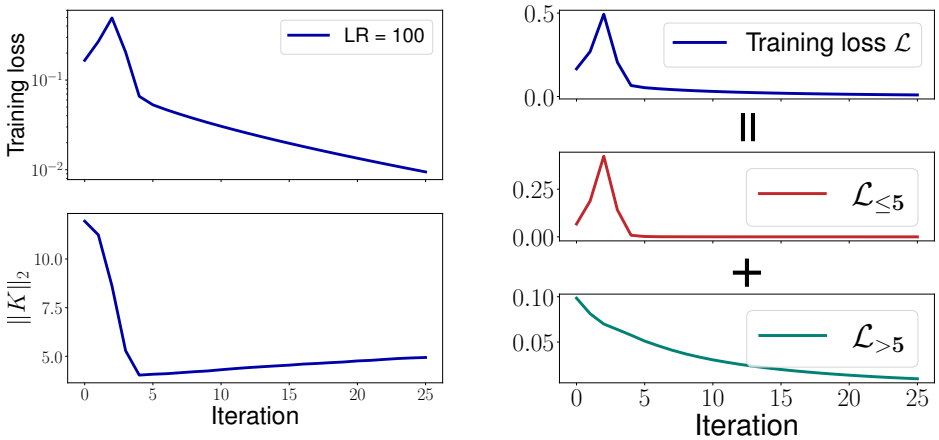
(b) Loss decomposition



(c) Training loss(WRN)

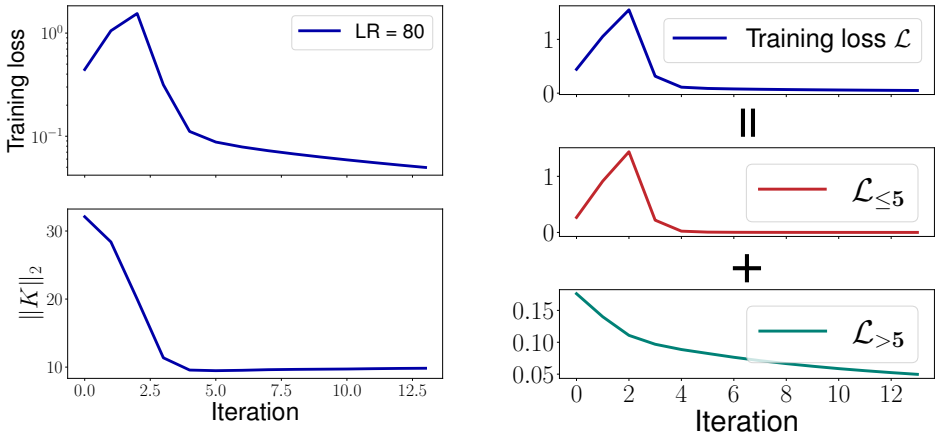
(d) Loss decomposition

Figure C.4. Catapult dynamics for FCN (a-b) on a synthetic dataset and Wide ResNets 10-10 (c-d) on CIFAR-10 dataset. Panel (a) and (c) are the training loss and the spectral norm of the tangent kernel with learning rates 5.0 and 3.0 respectively, and Panel (b) and (d) are the training loss decomposed into the top eigendirections of the tangent kernel, $\mathscr{L}_{\leq 5}$ and the remaining eigendirections, $\mathscr{L}_{>5}$. For the synthetic dataset, we use the rank-2 regression task considered in Section 4.4. The size of the training set is 128. In this experiment, the critical learning rates for FCN and WRN are 1.9 and 1.5 respectively.



(a) Training loss(FCN)

(b) Loss decomposition



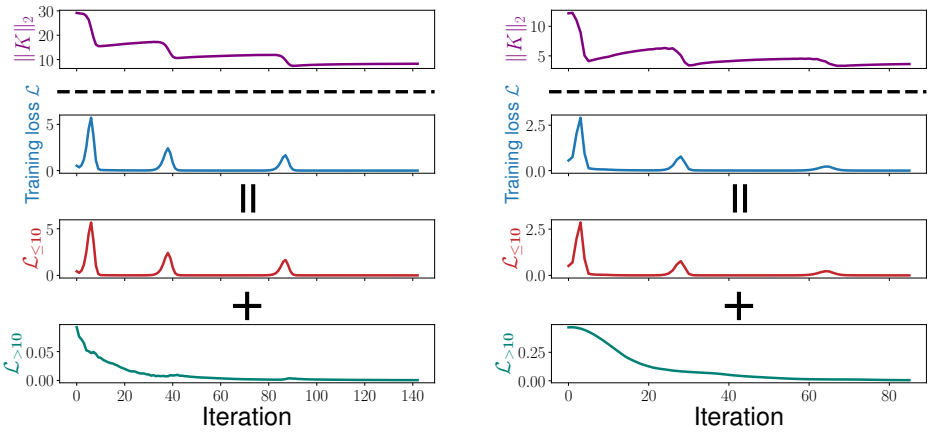
(c) Training loss(CNN)

(d) Loss decomposition

Figure C.5. Catapult dynamics for 5-layer FCN (a-b) and CNN (c-d) on multiclass classification tasks. Panel (a) and (c) are the training loss and the spectral norm of the tangent kernel with learning rate 100 and 80 respectively, and Panel (b) and (d) are the training loss decomposed into the top eigendirections of the tangent kernel, $\mathscr{L}_{\leq 5}$ and the remaining eigendirections, $\mathscr{L}_{>5}$. All the networks are trained on a subset of CIFAR-10 with 10 classes. Here the dimension of the eigenspace $s = 1, 3, 5$ refers to $10, 30, 50$ respectively due to the output dimension 10. The critical learning rate for FCN and CNN are 34 and 16 respectively.

Multiple catapults in GD occur in the top eigenspace of NTK

For the multiple catapults shown in Figure 4.4, similar to a single catapult, we show that the catapults occur in the top eigenspace of NTK. See Figure C.6.



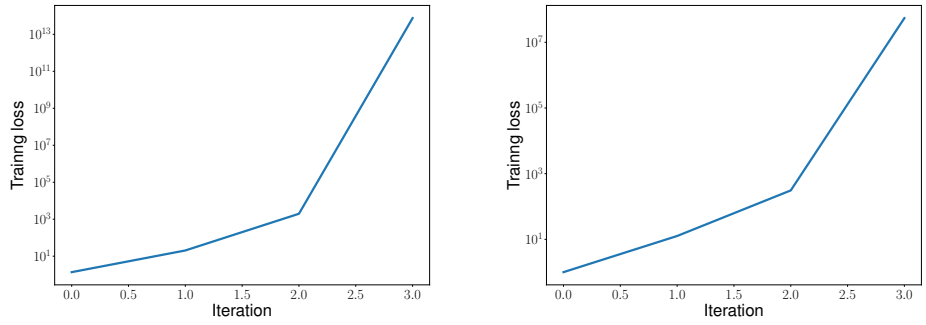
(a) 5-layer FCN **(b)** 5-layer CNN

Figure C.6. Multiple catapults in GD with increased learning rates. With the same setting of Figure 4.4, the training loss is decomposed into the top eigendirections of the tangent kernel, $\mathscr{L}_{\leq 10}$ and the remaining eigendirections, $\mathscr{L}_{>10}$

.

Multiple catapults allow a larger learning rate at convergence

Corresponding to Figure 4.4 in Section 4.3, we show that if the neural networks are trained with the learning rate at the convergence, i.e., after multiple catapults, the GD will diverge.



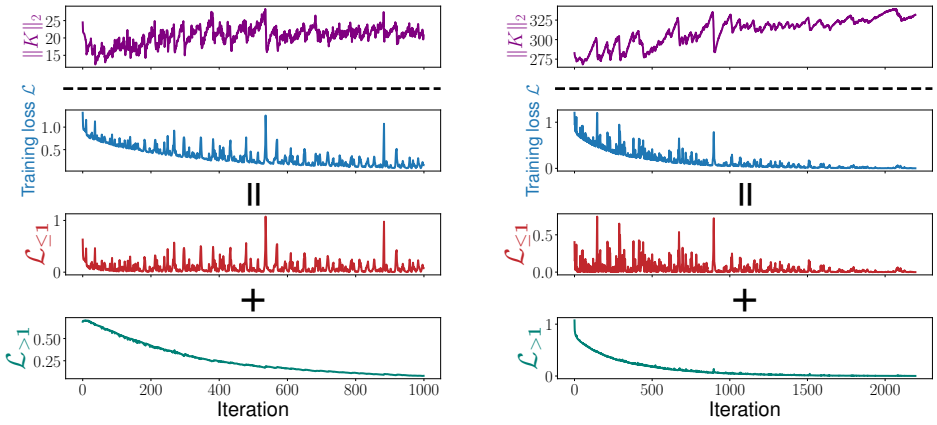
(a) 5-layer FCN **(b)** 5-layer CNN

Figure C.7. GD diverges when trained with the learning rate after multiple catapults. Corresponding to Figure 4.4, we train the model using GD with learning rate at convergence, 60 and 40 respectively for the 5-layer FCN and CNN.

C.3 Additional experiments for catapults in SGD

Full training process visualization corresponding to Figure 4.7

We present the complete training loss and the spectrum norm of the NTK corresponding to Figure 4.7(c,d) in Figure C.8.

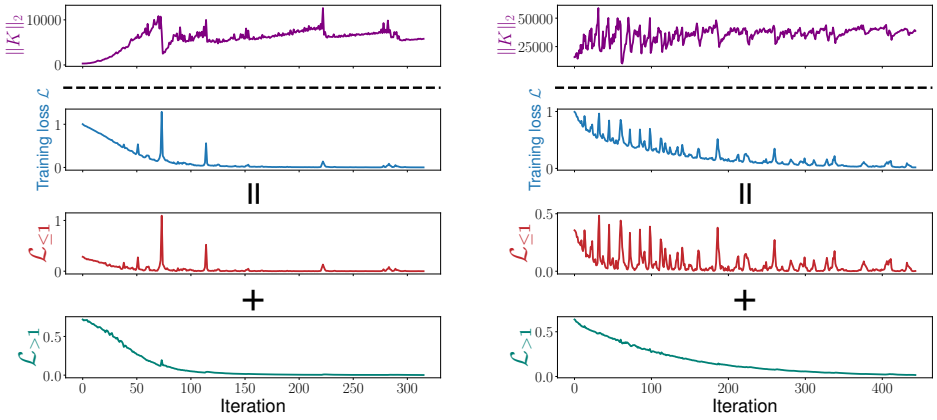


(a) Wide ResNets 10-10 **(b)** ViT-4

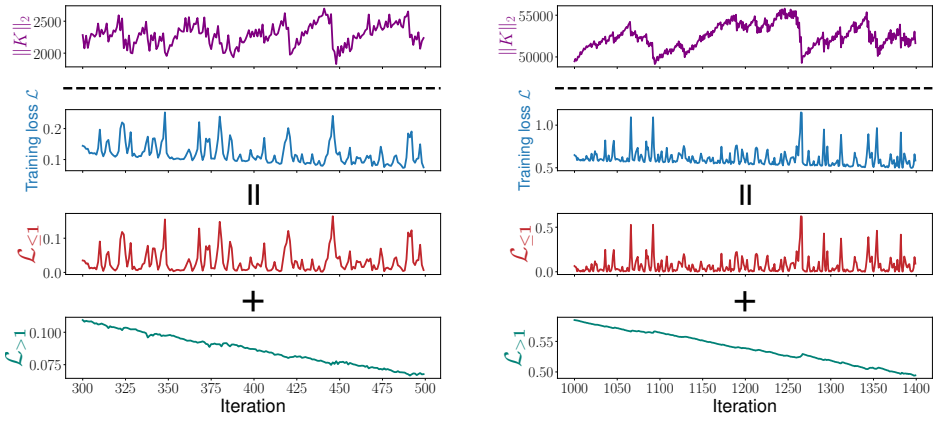
Figure C.8. Catapult dynamics in SGD for modern deep architectures corresponding to the whole training process. These are complete versions corresponding to Figure 4.7(c,d). The training loss is decomposed based on the eigendirections of the NTK: $\mathscr{L}_{\leq 1}$ and $\mathscr{L}_{>1}$.

Catapults in SGD with Pytorch default parameterization

In Figure 4.7, we used NTK parameterization (see the definition in Appendix C.6) for the neural networks. We further validate our empirical observations on (1) the occurrence of the loss spikes of SGD in the top eigenspace of the tangent kernel and (2) the decrease in the spectral norm of the tangent kernel during loss spikes in the setting with Pytorch default parameterization, under which the wide networks are still close to their linear approximations [66, 116] in Figure C.9.



(a) FCN **(b)** CNN

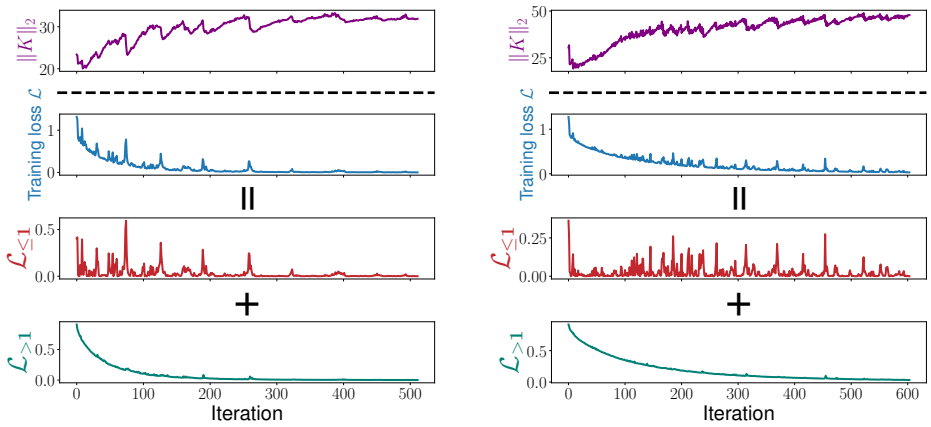


(c) Wide ResNets 10-10 **(d)** ViT-4

Figure C.9. Cataput dynamics in SGD for modern deep architectures with Pytorch default parameterization. The tasks are the same with Figure 4.7 except that we use Pytorch default parameterization. The training loss is decomposed based on the eigendirections of the NTK: $\mathscr{L}_{\leq 1}$ and $\mathscr{L}_{>1}$.

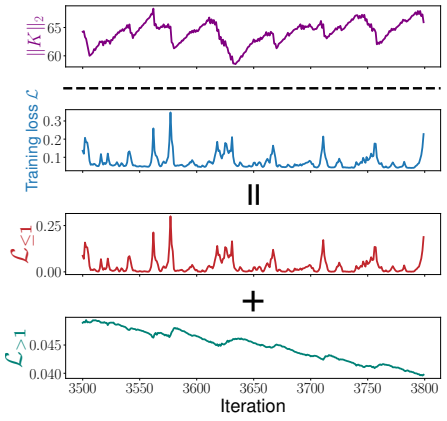
Catapults in SGD with additional datasets

We show that the findings in Figure 4.7 hold for a subset of SVHN dataset (see Figure C.10) and for a larger dataset (5,000 data points from CIFAR-2) and for multi-class classification problems (see Figure C.11).

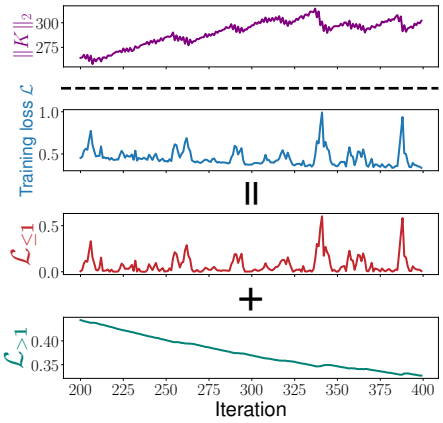


(a) 5-layer FCN

(b) 5-layer CNN

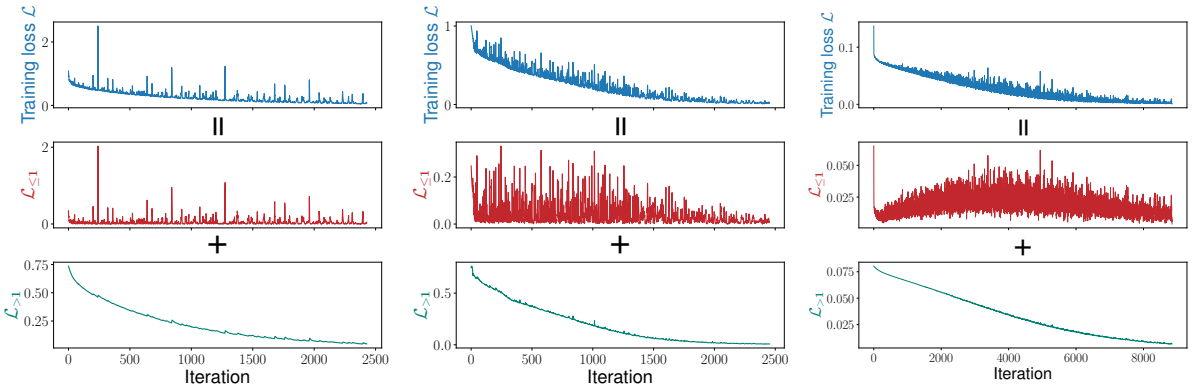


(c) Wide ResNets 10-10 (zoomed-in)



(d) ViT-4 (zoomed-in)

Figure C.10. Cataput dynamics in SGD for modern deep architectures on 2-class SVHN. The tasks are the same with Figure 4.7 except that we train the neural networks on a subset of SVHN dataset. The training loss is decomposed into the top eigenspace of the tangent kernel $\mathscr{L}_{\leq 1}$ and its complement $\mathscr{L}_{>1}$. Here $\mathscr{L} = \mathscr{L}_{\leq 1} + \mathscr{L}_{>1}$.

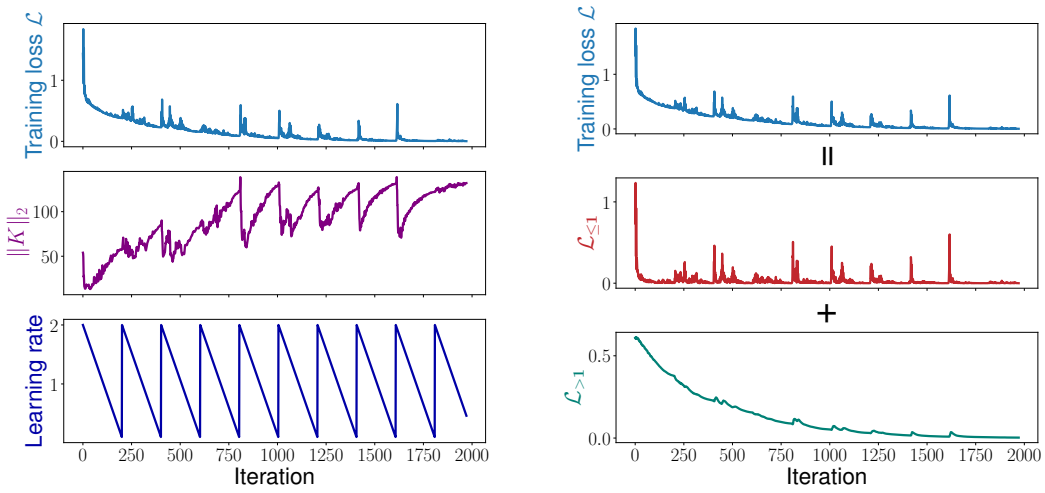


(a) 5-layer FCN **(b)** 5-layer CNN **(c)** Myrtle network [77]

Figure C.11. Catapult dynamics in SGD for large datasets (Panel (a) and (b)) and multi-class classification problems (Panel(c)). Panel(a,b): The networks are trained on $5,000$ data points from CIFAR-2. Panel(c): The network is trained on 128 points from CIFAR-10. The training loss is decomposed into the top eigenspace of the tangent kernel $\mathscr{L}_{\leq 1}$ and its complement $\mathscr{L}_{>1}$. Here $\mathscr{L} = \mathscr{L}_{\leq 1} + \mathscr{L}_{>1}$.

Catapults occur in training with cyclical learning rates

In this section, we show that catapults occur in SGD with a cyclical learning rate schedule. Specifically, we show that loss spikes occur in the top eigenspace of the tangent kernel and there is a decrease in the spectral norm of the tangent kernel according to each loss spike.



(a) Training loss, $\|K\|_2$ and learning rate

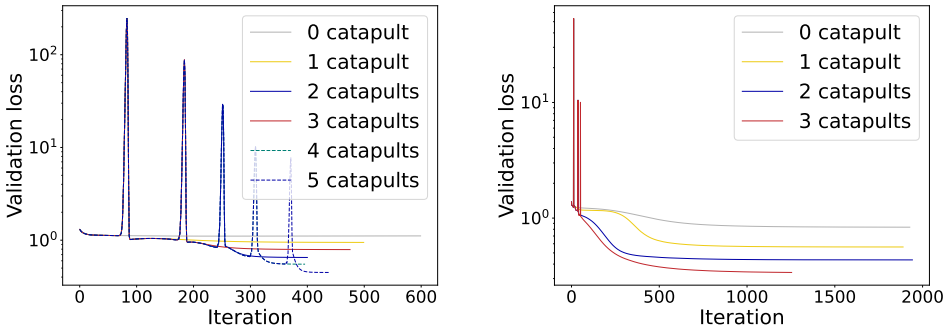
(b) Loss decomposition

Figure C.12. Catapults in SGD with cyclical learning rates. Panel (a): The plot of the training loss and the spectral norm of the tangent kernel corresponding to the whole training set with a cyclic learning rate schedule. Panel (b): The training loss is decomposed into the top and non-top eigenspace of the tangent kernel, i.e.,$\mathscr{L}_{\leq 1}$ and $\mathscr{L}_{>1}$. Here $\mathscr{L} = \mathscr{L}_{\leq 1} + \mathscr{L}_{>1}$. We train Wide ResNets 10-10 on a subset of CIFAR-10. The setting is the same with Figure 4.7c.

C.4 Additional experiments for feature learning in GD

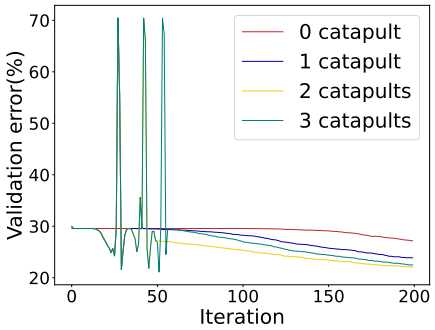
Validation loss/error for multiple catapults corresponding to Figure 4.8

We present the validation loss/error in Figure C.13 for the tasks corresponding to Figure 4.8. The learning rate is increased during training to generate multiple catapults.

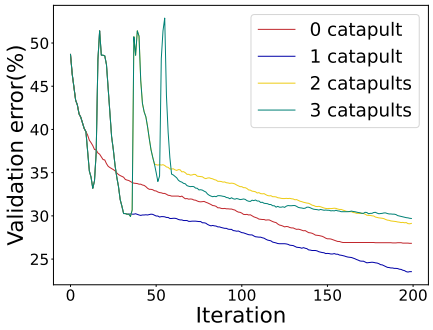


(a) Rank-2 regression

(b) Rank-3 regression



(c) SVHN-2

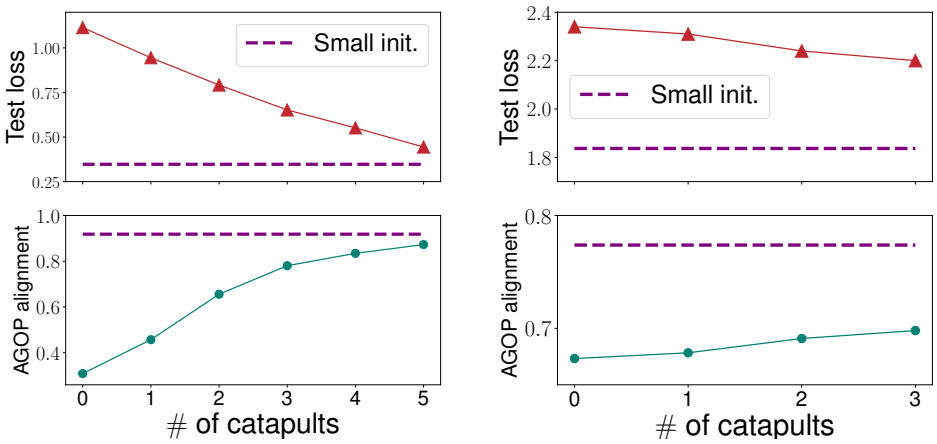


(d) CelebA

Figure C.13. Validation loss/error of multiple catapults in GD corresponding to Figure 4.8. Panel(c)&(d) only present first 200 iterations.

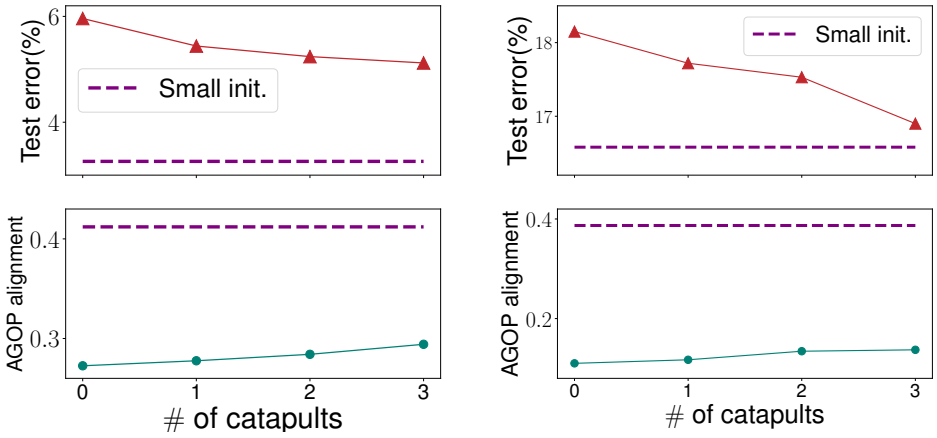
Feature learning with near zero initialization

We compare the performance of networks exhibiting multiple catapults with those initialized using near zero initialization scheme, i.e., each weight is sampled i.i.d. from $\mathcal{N}(0, \sigma^2)$ with $\sigma = 0.1$. This is in contrast to the NTK parameterization where we use $\sigma = 1$. It was argued in [116] that feature learning occurs with near zero initialization. We can see that small initialization achieves the smallest test loss/error as well as the best AGOP alignment, which indicates that learning AGOP correlates strongly with the test performance.



(a) Rank-2 regression

(b) Rank-4 regression



(c) SVHN-2

(d) CelebA

Figure C.14. Multiple catapults in GD compared to the near zero initialization scheme. We train a 2-layer FCN in Panel(a), a 4-layer FCN in Panel(b,d) and a 5-layer CNN in Panel(c). For near zero initialization, each weight parameter is i.i.d. from $\mathcal{N}(0, \sigma^2)$ with $\sigma = 0.1$. The experimental setup is the same as Figure 4.8.

For the Rank-2 regression task, we visualize the AGOP in the following Figure C.15, where we can see that the features are learned better, i.e., closer to the True AGOP, with a greater number of catapults.

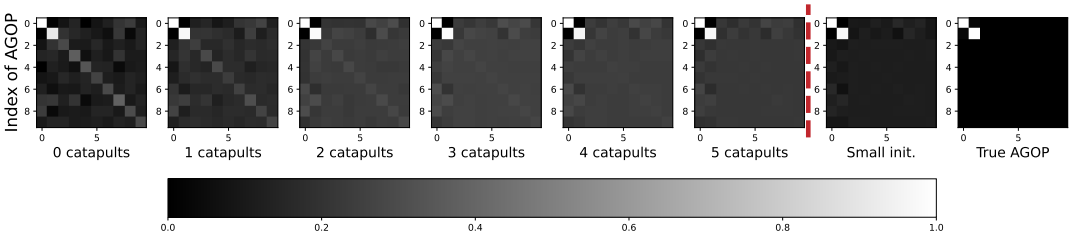
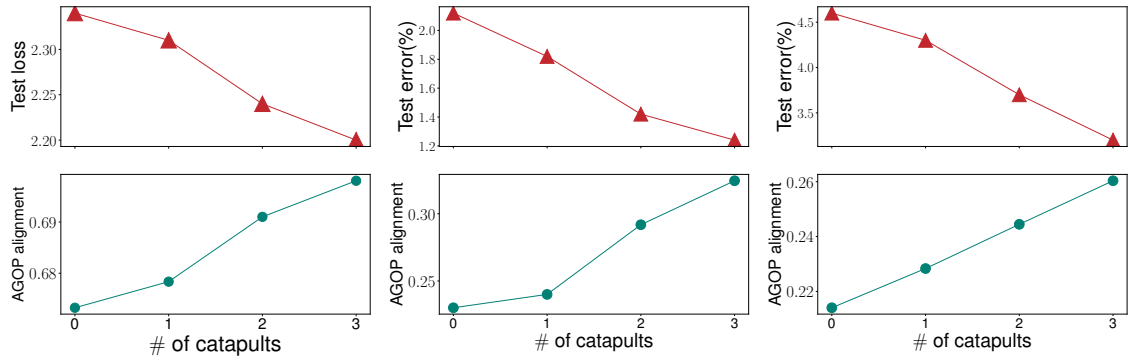


Figure C.15. Visualization of AGOP for rank-2 regression task. All pixels are normalized to the range $[0, 1]$ and the top 10 rows and columns of the AGOP are plotted.

Feature learning in GD for additional datasets

In this section, we show the findings observed in Figure 4.8 hold for Rank-4 regression, USPS dataset and Fashion MNIST dataset. See Figure C.16.

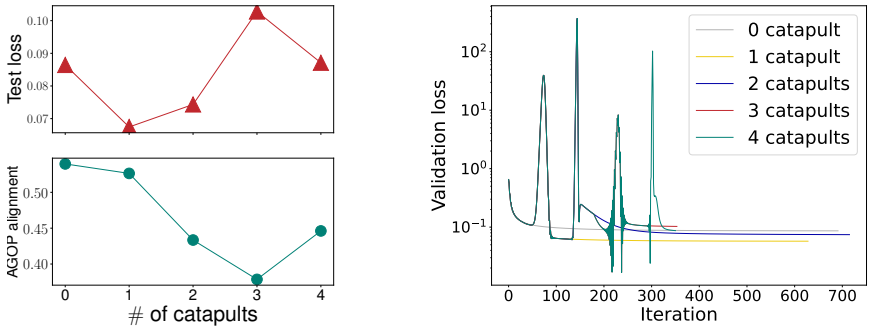


(a) Rank-4 regression **(b)** USPS **(c)** Fashion MNIST

Figure C.16. Correlation between AGOP alignment and test performance in GD with multiple catapults on additional datasets. We train a 4-layer FCN using GD for all tasks. The learning rate is increased multiple times during training to generate multiple catapults. Experimental details can be found in Appendix C.6.

No feature learning for full rank task

In Figure C.17, we show that for a full-rank task where the target function is $f^*(\boldsymbol{x}) = \frac{1}{\sqrt{d}} \|\boldsymbol{x}\|$, catapults do not improve the test performance or the AGOP alignment.



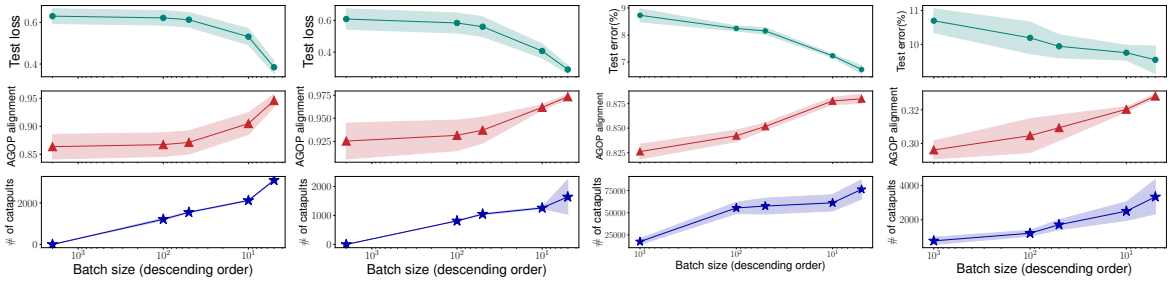
(a) Validation loss vs. iteration **(b)** Test loss and AGOP alignment

Figure C.17. Multiple catapults in GD for a full rank task. We train a 2-layer FCN on a synthetic dataset with a full rank target function $f^*(\boldsymbol{x}) = \frac{1}{\sqrt{d}} \|\boldsymbol{x}\|$ using GD.The learning rate is increased multiple times during training to generate multiple catapults. The experimental details can be found in Appendix C.6.

C.5 Additional experiments for feature learning in SGD

Feature learning of catapults in SGD with Pytorch parameterization

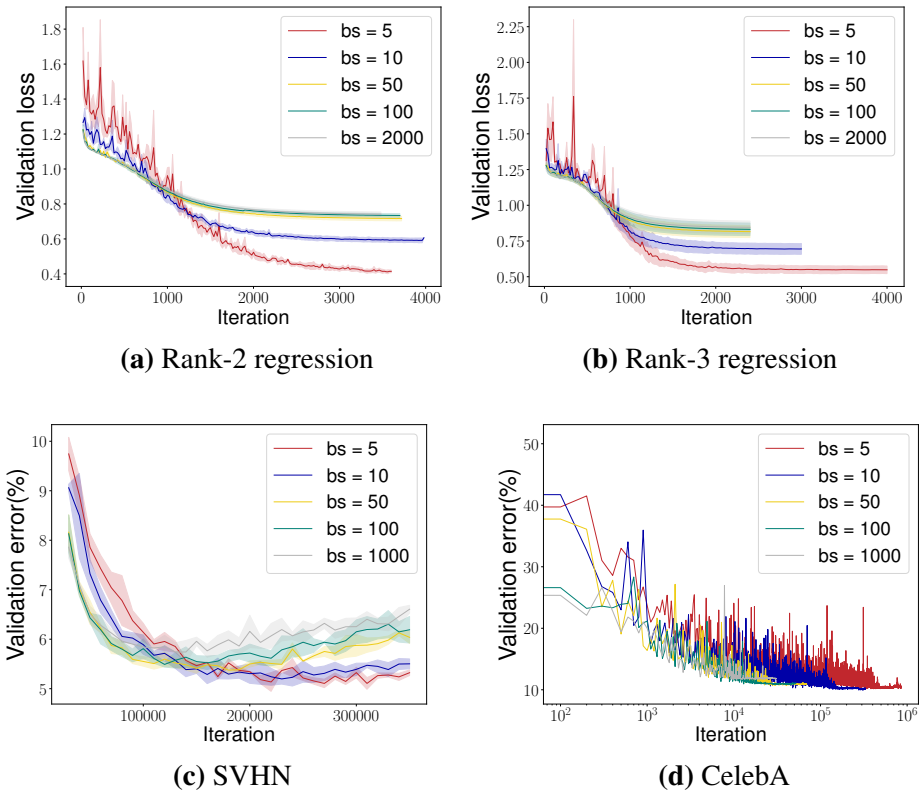
In this section, we further verify our observation on the feature learning of SGD with Pytorch default parameterization on the same tasks with Figure 4.9 in Section 4.4.



(a) Rank-2 regression **(b)** Rank-3 regression **(c)** SVHN **(d)** CelebA

Figure C.18. Correlation between AGOP alignment and test performance in SGD with Pytorch default parameterization. The tasks are the same with Figure 4.9 except that we use Pytorch default parameterization.

Validation loss/error of SGD corresponding to Figure 4.9



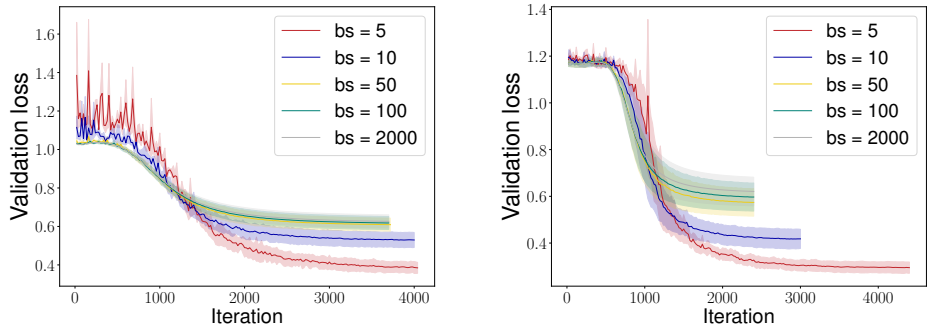
(a) Rank-2 regression

(b) Rank-3 regression

(c) SVHN

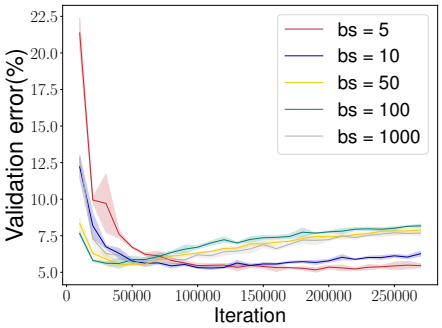
(d) CelebA

Figure C.19. Validation loss/error corresponding to Figure 4.9. Panel(c) presents the validation error from iteration 4000.

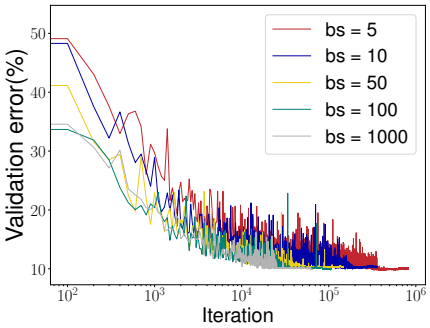


(a) Rank-2

(b) Rank-3



(c) SVHN

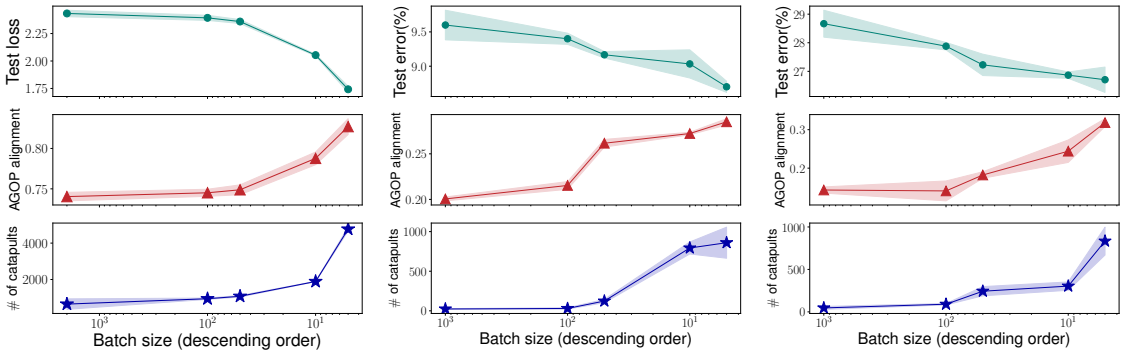


(d) CelebA

Figure C.20. Validation loss/error with Pytorch default parameterization corresponding to Figure C.18. Panel(c) presents the validation error from iteration 2000.

Feature learning in SGD for additional datasets

In this section, we show the findings observed in Figure 4.9 hold for Rank-4 regression, USPS dataset and Fashion MNIST dataset. See Figure C.21.

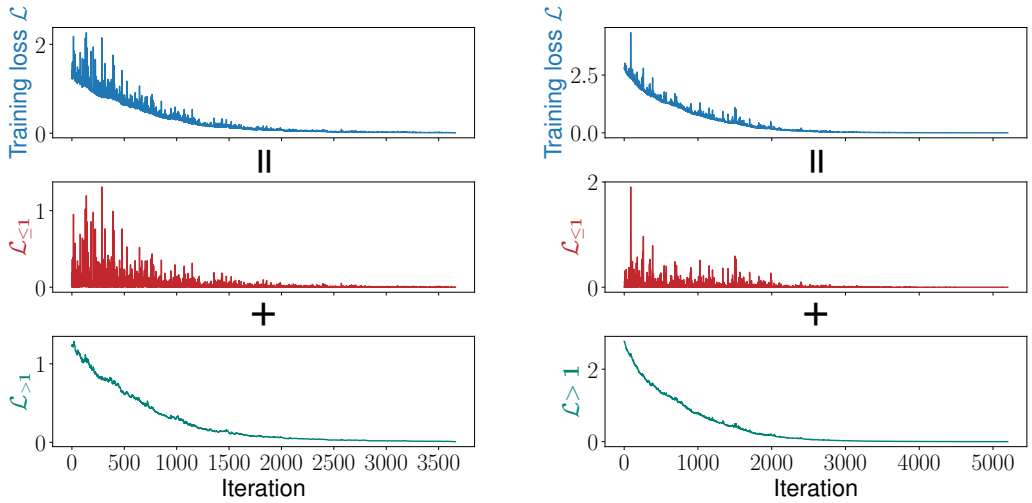


(a) Rank-4 regression **(b)** USPS **(c)** Fashion MNIST

Figure C.21. Correlation between AGOP alignment and test performance in SGD. We train a 4-layer fully connected neural network using SGD.

Verification of catapults in SGD

In Figure C.22, we verify that the spikes in the training loss of SGD with small batch sizes are caused by catapult dynamics. Specifically, we show that the spikes occur in the top eigendirection of the NTK.

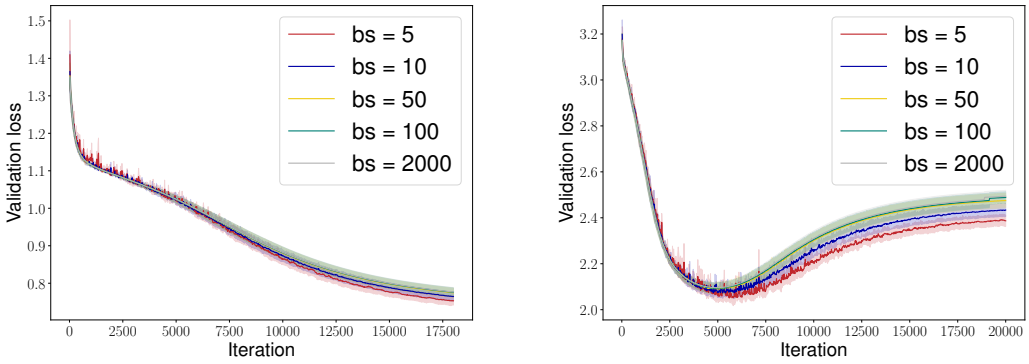


(a) Loss decomposition (Rank-2) **(b)** Loss decomposition (Rank-4)

Figure C.22. Verification of catapult dynamics: loss decomposition of Rank-2 and Rank-4 regression tasks corresponding to Figure 4.9 with batch size 5. The training loss is decomposed into the top eigenspace of the tangent kernel $\mathscr{L}_{\leq 1}$ and its complement $\mathscr{L}_{>1}$. Here $\mathscr{L} = \mathscr{L}_{\leq 1} + \mathscr{L}_{>1}$.

No feature learning with a small learning rate for SGD

In Figure 4.9, we have shown that a smaller batch size leads to more catapults, hence resulting in better test performance. In this section, we show that the test performance with different batch sizes is similar when training with a small learning rate, where no catapults occur. This further verifies that a greater number of catapults accounts for better test performance for small batch sizes. See Figure C.23.



(a) Rank-2 **(b)** Rank-4

Figure C.23. The networks are trained with a smaller learning rate corresponding to Figure C.19a &b. We train the network with a learning rate 0.1.

C.6 Experimental details

For all the networks considered in this paper, we use ReLU activation functions. We parameterize the networks by NTK parameterization [46]. Note that NTK parametrization is widely used for understanding neural networks [63, 21, 65]. We also verify our results with Pytorch [87] default parameterization for the experiments shown in Figure C.9 and C.18.

NTK parameterization. Given a neural network with NTK parameterization, all the trainable weight parameters are i.i.d. drawn from $\mathcal{N}(0,1)$. For a fully connected layer, it takes the form

$$f^{\ell+1} = \mathsf{ReLU}\left(\frac{1}{\sqrt{m_\ell}} W^\ell f^\ell + b^\ell\right),$$

where $W^\ell \in \mathbb{R}^{m_{\ell+1} \times m_\ell}, f^\ell \in \mathbb{R}^{m_\ell}, b^\ell \in \mathbb{R}^{m_{\ell+1}}$. For a convolutional layer, it takes the form

$$f_{i,j,k}^{\ell+1} = \mathsf{ReLU}\left(\frac{1}{\sqrt{m_\ell s^2}} \sum_{p=0}^{\lceil \frac{s+1}{2} \rceil} \sum_{q=0}^{\lceil \frac{s+1}{2} \rceil} \sum_{o=1}^{m_\ell} W_{p,q,o,k}^\ell f_{i-\lceil \frac{s-1}{2} \rceil, j-\lceil \frac{s-1}{2} \rceil, o}^\ell + b_k^\ell \right),$$

where $W^\ell \in \mathbb{R}^{s \times s \times m_\ell \times m_{\ell+1}}, f^\ell \in \mathbb{R}^{h \times w \times m_\ell}, b^\ell \in \mathbb{R}^{m_{\ell+1}}$. Note that s is the filter size and we assume the stride to be 1 in this case. For f^ℓ with negative indices, we let it be 0, i.e., zero padding. For the output layer, we use a linear layer without activation functions.

Dataset. For the synthetic datasets, we generate data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ by i.i.d. $\boldsymbol{x}_i \sim \mathcal{N}(0, I_{100})$ and $y_i = f^*(\boldsymbol{x}) + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, 0.1^2)$. For two real-world datasets, we consider a subset of CelebA dataset with glasses as the label, the Street View House Numbers (SVHN) dataset, USPS dataset and Fashion MNIST dataset. Due to computational limitations with GD, for some tasks, we select two classes (number 0 and 2) of SVHN dataset, USPS dataset and Fashion MNIST dataset.

EGOP (Epexcted Gradient Outer Product). Note that for these low-rank polynomial regression tasks, we know the analytical form of target functions hence we can calculate the EGOP by $G^* = \mathbb{E}_{\boldsymbol{x}} \frac{\partial f^*}{\partial \boldsymbol{x}} \frac{\partial f^*}{\partial \boldsymbol{x}}^T$. For real-world datasets, we estimate the EGOP by using the AGOP of one of the state-of-the-art models $\hat{f}$ that achieve high test accuracy: $\hat{G} = \frac{1}{n} \sum_{i=1}^n \frac{\partial \hat{f}}{\partial \boldsymbol{x}_i} \frac{\partial \hat{f}}{\partial \boldsymbol{x}_i}^T$.

In the following, we provide the detailed experimental setup for each experiment. Note that in the classification tasks, i.e. CelebA and SVHN datasets, the test error refers to the classification error on the test split.

Experiments in Section 4.3

Figure 4.3: We use a 2-class subset of CIFAR-10 dataset [59] (class 7 and class 9) and randomly select 128 data points out of it. For the network architectures, we use a 5-layer FCN with width 1024 and 5-layer CNN with 512 channels per layer. For CNN, we flatten the image into a one-dimensional vector before the last fully connected layer.

Figure 4.4: We use the same training tasks as in Figure 4.3. For FCN, we start with a

learning rate 6 and we increase the learning rate to $[10, 15]$ at iteration $[15, 60]$. For CNN, we start with a learning rate 8 and we increase the learning rate to $[15, 20]$ at iteration $[10, 40]$.

Figure 4.5: We consider a synthetic dataset (x_i, y_i) where x_i are sampled i.i.d. on unit sphere and $y_i = 1$ with training size and dimension both equal to 100. We train a wide two-layer ReLU network with second-layer weights fixed, using SGD with batch size one. The critical learning rate $\eta_{\text{crit}}(x_i)$ for (a minibatch of size one) x_i is proportional to $1/\|x_i\|_2^2 = const$. We select one data point (x_*, y_*) from the training set and multiply both x_* and y_* by 2. This makes the critical learning rate corresponding to the sample x_* four times smaller. We choose the (constant) learning rate η between the critical learning rate of (the mini-batch) x_* and the critical learning rates of the rest of the data points. Thus SGD with learning rate η induces catapult on x_* but not on any other data points.

Figure 4.6: For the shallow network, we use a 2-layer FCN with width 1024. We train the model on 128 data points from CIFAR 2 using SGD with batch size 32. We use a constant learning rate 0.8. We stop training when the training loss is less than 10^{-3}.

Table 4.2: The 5-layer FCN and CNN are the same as in Figure 4.3. We train the model on 128 data points from CIFAR 2 using SGD with batch size 32. We use a constant learning rate 6 and 8 for 5-layer FCN and CNN respectively. We stop training when the training loss is less than 10^{-3}.

Figure 4.7: The 5-layer FCN and CNN are the same as in Figure 4.3. And we use the standard Wide ResNets 10-10 and ViT-4 architectures. The learning rates for 5-layer FCN, 5-layer CNN, are $6, 8, 3, 0.2$ respectively. We train the model with a constant learning rate, and we stop training when the training loss is less than 10^{-3}. All the models are trained on 128 data points from CIFAR-2 using SGD with batch size 32.

Experiments in Section 4.4

Figure 4.8: For rank-2 task, we train a 2-layer FCN with width 1024. The size of the training set, testing set and validation set are $2000, 5000$ and 5000 respectively.

For rank-3 task, CelebA tasks, we train a 4-layer FCN with width 256. The size of the training set, testing set and validation set are $1000, 5000$ and 5000 respectively.

For SVHN-2 tasks, we train a 5-layer CNN with width 256. We select class 0 and class 2 out of the full SVHN datasets as SVHN-2. The size of the training set, testing set and validation set are $1000, 5000$ and 5000 respectively.

We increase the learning rate during training. For Rank-2 task, we increase the learning rate to $[8, 16, 30, 50, 75, 80]$ at iteration $[50, 150, 220, 280, 350, 400]$. For Rank-3 task, we increase the learning rate to $[40, 100, 150]$ at iteration $[20, 60, 80]$. For SVHN-2 task, we increase the learning rate to $[30, 60, 90]$ at iteration $[10, 35, 50]$. For CelebA task, we increase the learning rate to $[40, 70, 100]$ at iteration $[10, 35, 50]$. We decay the learning rate if necessary after the catapult to avoid extra catapults until the end of training.

Figure 4.9: For both Rank-2 and Rank-3 tasks, we let the size of training set, testing set and validation set be $2000, 5000$ and 5000. For the SVHN task, we train the full SVHN using the 5-layer Myrtle network. For the CelebA task, we train the full 2-class CelebA dataset with glasses feature using 4-layer FCN with width 256. To obtain the true AGOP, we use one of the SOTA models (WideResNet 16-2) which achieves 97.2% test accuracy on SVHN and 5-layer Myrtle network which achieves 95.7% test accuracy on CelebA.

We use the same learning rate across batch sizes for each task. The learning rate is chosen as $\frac{1}{2}\eta_{\text{crit}}$ corresponding to the whole training set. For SVHN and CelebA tasks, we estimate η_{crit} using a subset with size 5000 of the whole training set. We train the model with batch size $[5, 10, 50, 100, 2000]$. For all tasks, we stop training when the training loss is less than 10^{-3}. We report the average of 3 independent runs.

Figure 4.11: We use the same network architectures and training/validation/testing sets as in Figure 4.8.

For all the tasks, except for GD, all the optimizers use a mini-batch size 100.

We stop training when the training loss is less than 10^{-3}. We report the average of 3 independent runs.

For the rank-2 task and rank-4 task, we know the target function hence we can analytically compute the exact true AGOP. For SVHN-2 task and CelebA task, to estimate the true AGOP, we use one of the SOTA models, Myrtle-5 which achieves 98.4% test accuracy on two-class SVHN dataset and 95.7% test accuracy on CelebA dataset.

The following table is the learning rate we choose for the experiments:

Table C.1. Choice of learning rates for Figure 4.11.

Task	SGD	GD	SGD+M	Adadelta	Adagrad	RMSprop	Adam
Rank-2	2.0	2.0	2.0	2.0	0.1	10^{-2}	10^{-2}
Rank-3	2.0	2.0	2.0	2.0	10^{-2}	10^{-2}	10^{-3}
Rank-4	1.0	1.0	1.0	1.0	5×10^{-3}	10^{-3}	10^{-3}
SVHN-2	5.0	5.0	5.0	5.0	5×10^{-3}	10^{-4}	10^{-3}
CelebA	10.0	10.0	10.0	10.0	5×10^{-3}	10^{-3}	10^{-3}

The experiment is to demonstrate the correlation between AGOP alignment and test performance. For this reason, we did not fine-tune the learning rate to achieve the best test performance.

Experiments in Appendix C.3

Figure C.5: We use the same network architectures as in Figure 4.3 and we train 128 data point from CIFAR-10.

Figure C.10: We use the same setting as Figure 4.7, except we train the networks on 128 data points from SVHN-2(number 0 and 2).

Figure C.11: For panel(a) and panel(b), we train the same 5-layer FCN and CNN as in Figure 4.3 and on 5,000 data points from CIFAR-2. For panel(c), we train a 5-layer Myrtle network on 128 points from CIFAR-10.

Experiments in Appendix C.4

Figure C.16: For rank-4 task, USPS dataset and Fashion MNIST dataset, we train a 4-layer FCN with width 256. The size of the training set, testing set and validation set are $1000, 5000$ and 5000 respectively.

For rank-4 task, we increase the learning rate to $[15, 40, 60]$ at iteration $[50, 75, 110]$. For USPS dataset, we increase the learning rate to $[15, 30, 40]$ at iteration $[10, 30, 45]$. For Fashion MNIST dataset, we increase the learning rate to $[10, 40, 55]$ at iteration $[6, 20, 30]$.

Figure C.17: We train a 2-layer FCN with width 1024. We consider a synthetic dataset, where $f^*(\boldsymbol{x}) = \frac{1}{\sqrt{d}} \|\boldsymbol{x}\|$. The size of the training set and validation set is $128, 2000$ respectively. During training, we start with lr=6 and increase the learning rate to $[7, 12, 40, 80]$ at iteration $[30, 120, 180, 280]$.

Experiments in Appendix C.5

Figure C.18: We use the same setup with Figure 4.9 except that all the networks are parameterized with Pytorch default parameterization. The learning rates are 0.01, 0.01, 0.05 and 1.0 for each task.

Figure C.21: For rank-4 task, USPS dataset and Fashion MNIST dataset, we train a 4-layer FCN with width 256. The size of the training set, testing set and validation set are $2000, 5000$ and 5000 respectively. We add 10% label noise for the USPS dataset and Fashion MNIST dataset. To obtain the true AGOP, we use one of the SOTA models (5-layer CNN) which achieves 99.2% test accuracy on USPS and 5-layer Myrtle network which achieves 91.8% test accuracy on Fashion MNIST.

Bibliography

[1] Emmanuel Abbe, Enric Boix-Adsera, Matthew S Brennan, Guy Bresler, and Dheeraj Nagaraj. The staircase property: How hierarchical structure can guide deep learning. *Advances in Neural Information Processing Systems*, 34:26989–27002, 2021.

[2] Atish Agarwala and Yann Dauphin. Sam operates far from home: eigenvalue regularization as a dynamical phenomenon. In *International Conference on Machine Learning*, pages 152–168. PMLR, 2023.

[3] Atish Agarwala, Fabian Pedregosa, and Jeffrey Pennington. Second-order regression models exhibit progressive sharpening to the edge of stability. In *International Conference on Machine Learning*, pages 169–195. PMLR, 2023.

[4] Kwangjun Ahn, Jingzhao Zhang, and Suvrit Sra. Understanding the unstable convergence of gradient descent. In *International Conference on Machine Learning*, pages 247–257. PMLR, 2022.

[5] Sina Alemohammad, Zichao Wang, Randall Balestriero, and Richard Baraniuk. The recurrent neural tangent kernel. In *International Conference on Learning Representations*, 2021.

[6] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 242–252. PMLR, 09–15 Jun 2019.

[7] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*, 2023.

[8] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in neural information processing systems*, 32, 2019.

[9] Sanjeev Arora, Zhiyuan Li, and Abhishek Panigrahi. Understanding gradient descent on the edge of stability in deep learning. In *International Conference on Machine Learning*, pages 948–1024. PMLR, 2022.

[10] Alexander Atanasov, Blake Bordelon, and Cengiz Pehlevan. Neural networks as kernel learners: The silent alignment effect. In *International Conference on Learning Representations*, 2022.

[11] Yu Bai and Jason D. Lee. Beyond linearization: On quadratic and higher-order approximation of wide neural networks. In *International Conference on Learning Representations*, 2020.

[12] Arindam Banerjee, Pedro Cisneros-Velarde, Libin Zhu, and Mikhail Belkin. Neural tangent kernel at initialization: linear width suffices. In *Uncertainty in Artificial Intelligence*, pages 110–118. PMLR, 2023.

[13] Daniel Beaglehole, Adityanarayanan Radhakrishnan, Parthe Pandit, and Mikhail Belkin. Mechanism of feature learning in convolutional neural networks. *arXiv preprint arXiv:2309.00570*, 2023.

[14] Nicoletta Bof, Ruggero Carli, and Luca Schenato. Lyapunov theory for discrete time systems. *arXiv preprint arXiv:1809.05289*, 2018.

[15] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, 32, 2019.

[16] Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021.

[17] Alex Damian, Eshaan Nichani, and Jason D. Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. In *The Eleventh International Conference on Learning Representations*, 2023.

[18] Lijun Ding, Dmitriy Drusvyatskiy, Maryam Fazel, and Zaid Harchaoui. Flat minima generalize for low-rank matrix recovery. *Information and Inference: A Journal of the IMA*, 13(2):iaae009, 2024.

[19] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR, 2017.

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain

Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[21] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685, 2019.

[22] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2018.

[23] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[24] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[25] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.

[26] Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33:5850–5861, 2020.

[27] Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The early phase of neural network training. In *International Conference on Learning Representations*, 2020.

[28] Jonas Geiping, Micah Goldblum, Phil Pope, Michael Moeller, and Tom Goldstein. Stochastic training is not necessary for generalization. In *International Conference on Learning Representations*, 2022.

[29] Justin Gilmer, Behrooz Ghorbani, Ankush Garg, Sneha Kudugunta, Behnam Neyshabur, David Cardoze, George Dahl, Zachary Nado, and Orhan Firat. A loss curvature perspective on training instability in deep learning. *arXiv preprint arXiv:2110.04369*, 2021.

[30] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[31] Antonio Gulli. AG's corpus of news articles. http://groups.di.unipi.it/~gulli/AG_corpus_

of_news_articles.html.

[32] Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry. In *International Conference on Machine Learning*, pages 1832–1841. PMLR, 2018.

[33] Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. *Advances in neural information processing systems*, 30, 2017.

[34] Boris Hanin and Mihai Nica. Finite depth and width corrections to the neural tangent kernel. In *International Conference on Learning Representations*, 2020.

[35] Wolfgang Härdle and Thomas M Stoker. Investigating smooth multiple regression by the method of average derivatives. *Journal of the American statistical Association*, 84(408):986–995, 1989.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[37] Geoffrey Hinton, 2014.

[38] Sepp Hochreiter and Jürgen Schmidhuber. Simplifying neural nets by discovering flat minima. *Advances in neural information processing systems*, 7, 1994.

[39] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.

[40] Marian Hristache, Anatoli Juditsky, Jorg Polzehl, and Vladimir Spokoiny. Structure adaptive approach for dimension reduction. *Annals of Statistics*, pages 1537–1566, 2001.

[41] Jiri Hron, Yasaman Bahri, Jascha Sohl-Dickstein, and Roman Novak. Infinite attention: Nngp and ntk for deep attention networks. In *International Conference on Machine Learning*, pages 4376–4386. PMLR, 2020.

[42] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[43] Jiaoyang Huang and Horng-Tzer Yau. Dynamics of deep neural networks and neural tangent hierarchy. In *International Conference on Machine Learning*, pages 4542–4551. PMLR, 2020.

[44] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.

[45] P Izmailov, AG Wilson, D Podoprikhin, D Vetrov, and T Garipov. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 876–885, 2018.

[46] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[47] Jakobovski. Free-Spoken-Digit-Dataset. https://github.com/Jakobovski/free-spoken-digit-dataset.

[48] Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

[49] Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho*, and Krzysztof Geras*. The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations*, 2020.

[50] Stanisław Jastrzębski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amost Storkey. On the relation between the sharpest directions of DNN loss and the SGD step length. In *International Conference on Learning Representations*, 2019.

[51] Ziwei Ji and Matus Telgarsky. Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow relu networks. In *International Conference on Learning Representations*, 2020.

[52] Yiding Jiang*, Behnam Neyshabur*, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020.

[53] Dayal Singh Kalra and Maissam Barkeshli. Phase diagram of early training dynamics in deep neural networks: effect of the learning rate, depth, and width. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[54] Ibrahem Kandel and Mauro Castelli. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT express*, 6(4):312–315, 2020.

[55] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and

Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

[56] Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*, 2017.

[57] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[58] Bobby Kleinberg, Yuanzhi Li, and Yang Yuan. An alternative view: When does sgd escape local minima? In *International conference on machine learning*, pages 2698–2707. PMLR, 2018.

[59] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

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

[61] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[62] Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 2002.

[63] Jaehoon Lee, Samuel Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. *Advances in Neural Information Processing Systems*, 33:15156–15172, 2020.

[64] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in neural information processing systems*, pages 8570–8581, 2019.

[65] Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

[66] Chaoyue Liu, Libin Zhu, and Mikhail Belkin. On the linearity of large non-linear models: when and why the tangent kernel is constant. *Advances in Neural Information Processing Systems*, 33, 2020.

[67] Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational*

Harmonic Analysis, 2022.

[68] Chaoyue Liu, Libin Zhu, and Misha Belkin. Transition to linearity of wide neural networks is an emerging property of assembling weak models. In *International Conference on Learning Representations*, 2022.

[69] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[70] Stanislaw Lojasiewicz. A topological property of real analytic subsets. *Coll. du CNRS, Les équations aux dérivées partielles*, 117:87–89, 1963.

[71] Philip M Long. Properties of the after kernel. *arXiv preprint arXiv:2105.10585*, 2021.

[72] Noel Loo, Ramin Hasani, Alexander Amini, and Daniela Rus. Evolution of neural tangent kernels under benign and adversarial training. *Advances in Neural Information Processing Systems*, 35:11642–11657, 2022.

[73] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

[74] James L McClelland, David E Rumelhart, PDP Research Group, et al. *Parallel Distributed Processing, Volume 2: Explorations in the Microstructure of Cognition: Psychological and Biological Models*, volume 2. MIT press, 1987.

[75] David Meltzer and Junyu Liu. Catapult dynamics and phase transitions in quadratic nets. *arXiv preprint arXiv:2301.07737*, 2023.

[76] Andrea Montanari and Yiqiao Zhong. The interpolation phase transition in neural networks: Memorization and generalization under lazy training. *The Annals of Statistics*, 50(5):2816–2847, 2022.

[77] myrtle.ai. Page, d., 2018.

[78] Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady AN USSR*, volume 269, pages 543–547, 1983.

[79] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[80] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

[81] Quynh Nguyen, Marco Mondelli, and Guido F Montufar. Tight bounds on the smallest eigenvalue of the neural tangent kernel for deep relu networks. In *International Conference on Machine Learning*, pages 8119–8129. PMLR, 2021.

[82] Quynh Nguyen, Mahesh Chandra Mukkamala, and Matthias Hein. On the loss landscape of a class of deep neural networks with no bad local valleys. In *International Conference on Learning Representations*, 2019.

[83] Eshaan Nichani, Adityanarayanan Radhakrishnan, and Caroline Uhler. Increasing depth leads to U-shaped test risk in over-parameterized convolutional networks. In *International Conference on Machine Learning Workshop on Over-parameterization: Pitfalls and Opportunities*, 2021.

[84] OpenAI. Chatgpt. https://www.openai.com/chatgpt, 2020. Version 4.

[85] Guillermo Ortiz-Jiménez, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. What can linearized neural networks actually say about generalization? *Advances in Neural Information Processing Systems*, 34, 2021.

[86] Vardan Papyan. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. In *International Conference on Machine Learning*, pages 5012–5021. PMLR, 2019.

[87] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[88] Boris T Polyak. Introduction to optimization. *Optimization Software, Inc, New York*, 1987.

[89] Boris Teodorovich Polyak. Gradient methods for minimizing functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 3(4):643–653, 1963.

[90] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[91] Adityanarayanan Radhakrishnan, Daniel Beaglehole, Parthe Pandit, and Mikhail Belkin. Mechanism for feature learning in neural networks and backpropagation-free machine learning models. *Science*, 383(6690):1461–1467, 2024.

[92] Adityanarayanan Radhakrishnan, Mikhail Belkin, and Dmitriy Drusvyatskiy. Linear recursive feature machines provably recover low-rank matrices. *arXiv preprint arXiv:2401.04553*, 2024.

[93] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[94] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[95] Daniel A Roberts, Sho Yaida, and Boris Hanin. *The principles of deep learning theory*. Cambridge University Press, 2022.

[96] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[97] Pedro Savarese, Itay Evron, Daniel Soudry, and Nathan Srebro. How do infinite width bounded norm networks look in function space? In *Conference on Learning Theory*, pages 2667–2690. PMLR, 2019.

[98] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[99] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE, 2019.

[100] Samuel L Smith, Benoit Dherin, David Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*, 2021.

[101] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19(70):1–57, 2018.

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

[103] Matus Telgarsky. Margins, shrinkage, and boosting. In *International Conference on Machine Learning*, pages 307–315. PMLR, 2013.

[104] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[105] Shubhendu Trivedi, Jialei Wang, Samory Kpotufe, and Gregory Shakhnarovich. A

consistent estimator of the expected gradient outerproduct. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, pages 819–828, 2014.

[106] Joel A Tropp et al. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230, 2015.

[107] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[108] Zixuan Wang, Zhouzi Li, and Jian Li. Analyzing sharpness along gd trajectory: Progressive sharpening and edge of stability. *Advances in Neural Information Processing Systems*, 35:9983–9994, 2022.

[109] Francis Williams, Matthew Trager, Daniele Panozzo, Claudio Silva, Denis Zorin, and Joan Bruna. Gradient dynamics of shallow univariate relu networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[110] Mitchell Wortsman, Ali Farhadi, and Mohammad Rastegari. Discovering neural wirings. *Advances in Neural Information Processing Systems*, 32, 2019.

[111] Lei Wu, Zhanxing Zhu, et al. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv preprint arXiv:1706.10239*, 2017.

[112] Yingcun Xia, Howell Tong, Wai Keung Li, and Li-Xing Zhu. An adaptive estimation of dimension reduction space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(3):363–410, 2002.

[113] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[114] Zeke Xie, Issei Sato, and Masashi Sugiyama. A diffusion theory for deep learning dynamics: Stochastic gradient descent exponentially favors flat minima. In *International Conference on Learning Representations*, 2021.

[115] Chen Xing, Devansh Arpit, Christos Tsirigotis, and Yoshua Bengio. A walk with sgd. *arXiv preprint arXiv:1802.08770*, 2018.

[116] Greg Yang and Edward J Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pages 11727–11737. PMLR, 2021.

[117] Jiaxuan You, Jure Leskovec, Kaiming He, and Saining Xie. Graph structure of neural networks. In *International Conference on Machine Learning*, pages 10881–10891. PMLR, 2020.

[118] Gan Yuan, Mingyue Xu, Samory Kpotufe, and Daniel Hsu. Efficient estimation of the central mean subspace via smoothed gradient outer products. *arXiv preprint arXiv:2312.15469*, 2023.

[119] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

[120] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[121] Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger B Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. *Advances in neural information processing systems*, 32, 2019.

[122] Zhongwang Zhang and Zhi-Qin John Xu. Loss spike in training neural networks. *arXiv preprint arXiv:2305.12133*, 2023.

[123] Libin Zhu, Chaoyue Liu, and Misha Belkin. Transition to linearity of general neural networks with directed acyclic graph architecture. *Advances in Neural Information Processing Systems*, 35:5363–5375, 2022.

[124] Libin Zhu, Chaoyue Liu, Adityanarayanan Radhakrishnan, and Mikhail Belkin. Quadratic models for understanding catapult dynamics of neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.

[125] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Gradient descent optimizes over-parameterized deep relu networks. *Machine Learning*, 109(3):467–492, 2020.

[126] Difan Zou and Quanquan Gu. An improved analysis of training over-parameterized deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2053–2062, 2019.