Optimization Landscape of Deep Neural Networks

A dissertation submitted towards the degree

Doctor of natural science

of the Faculty of Mathematics and Computer Science of

Saarland University

by

Quynh Nguyen

Saarbrücken, April 2019

Day of Colloquium	18.10.2019
Dean of the Faculty	Prof. Dr. Sebastian Hack

Examination Committee

Chair	Prof. Dr. Makus Bläser
Reviewer, Advisor	Prof. Dr. Matthias Hein
Reviewer	Prof. Dr. Marius Kloft
Reviewer	Prof. Dr. Matus Telgarsky
Academic Assistant	Dr. Pavel Kolev

Abstract

It has been empirically observed in deep learning that the training problem of deep over-parameterized neural networks does not seem to have a big problem with suboptimal local minima despite all hardness results proven in the literature. In many cases, local search algorithms such as (stochastic) gradient descent frequently converge to a globally optimal solution. In an attempt to better understand this phenomenon, this thesis studies sufficient conditions on the network architecture so that the landscape of the associated loss function is guaranteed to be well-behaved, which could be favorable to local search algorithms. Our analysis touches upon fundamental aspects of the problem such as existence of solutions with zero training error, global optimality of critical points, topology of level sets and sublevel sets of the loss. Gaining insight from this analysis, we come up with a new class of network architectures that are practically relevant and have a strong theoretical guarantee on the loss surface. We empirically investigate the generalization ability of these networks and other related phenomena observed in deep learning such as implicit bias of stochastic gradient descent. Finally, we study limitations of deep and narrow neural networks in learning connected decision regions, and draw connections to adversarial manipulation problems. Our results and analysis presented in this thesis suggest that having a sufficiently wide layer in the architecture is not only helpful to make the loss surface become well-behaved but also important to the expressive power of neural networks.

Zusammenfassung

Es wurde empirisch beobachtet, dass beim Trainieren von überparametrisierten tiefen, neuronalen Netzen keine Probleme mit lokalen Minima auftreten, trotz den Schwerheits-Resultaten in der Literatur. In vielen Fällen konvergieren lokale Suchalgorithmen wie (stochastischer) Gradientenabstieg oft zu einer global optimalen Lösung. In einem Versuch dieses Phänomen besser zu verstehen, diskutiert diese Arbeit hinreichende Bedingungen an die Netzwerkarchitektur, so dass die Funktionslandschaft der assozierten Verlustfunktion sich garantiert gut verhält, was günstig für lokale Suchalgorithmen ist. Unsere Analyse bezieht sich auf grundlegende Aspekte des Problems wie z.B. Existenz von Lösungen mit null Trainingsfehlern, globale Optimalität der kritischen Punkte und Topologie der Niveau- und Unterniveau-Mengen der Verlustfunktion. Aus den in dieser Analyse gewonnenen Erkenntnisse entwickeln wir eine neue Klasse von Netzwerkarchitekturen, die praxisrelevant sind und die starke theoretische Garantien über die Oberfläche der Verlustfunktion erlauben. Wir untersuchen empirisch die Generalisierungsfähigkeit dieser Netzwerke und anderer verwandter Phänomene, die beim tiefen Lernen beobachtet wurden, wie z.B. der implizite Bias des stochastischen Gradientenabstiegs. Weiter diskutieren wir Einschränkungen tiefer und schmaler neuronaler Netze beim Lernen von miteinander verbundenen Entscheidungsregionen und stellen eine Verbindung zum Problem der bösartigen Manipulation her. Unsere Ergebnisse und Analysen, die in dieser Arbeit vorgestellt werden, legen nahe, dass eine ausreichend breite Schicht in der Architektur nicht nur hilfreich ist, damit die Verlustoberfläche wohlbehalten ist, aber auch wichtig ist für die Ausdrucksstärke von neuronalen Netzen.

Acknowledgment

First of all, I would like to thank Matthias Hein for giving me the possibility to do my doctoral thesis with him. I am thankful to Matthias for giving me the freedom to look for my own lines of researches, for sharing his wisdom with me and showing me how to conduct research in a positive and sustainable manner. I am grateful to Matthias for all the skills which I have learned from him, among which my favorite are asking the right questions, writing good scientific papers, and how to view things from the perspective of other reviewers to improve our own research. I also very much appreciated all the time, advices and support that he has provided to me in times when it was needed.

I would like to sincerely thank Marius Kloft and Matus Telgarsky for agreeing to review my thesis. I am thankful to Matus for inviting me to apply for the Simons-Berkeley fellowship through whose support I can participate in the exciting program "Foundations of Deep Learning" in Berkeley where he is one of the organizers.

I would like to thank Joachim Weickert and Bernt Schiele for their excellent courses in image processing and computer vision through which I learned a lot of helpful knowledge before my PhD.

I thank all the present and former members of our lab for a nice atmosphere and for the mutual support of our small group. In particular, I would like to thank Francesco Tudisco for being a good friend of mine, Pedro Mercado Lopez for being a nice office mate with many subtle jokes, Antoine Gautier for pleasant and helpful discussions, and Nikita for a nice time in Saarbrucken.

Finally I would like to thank my parents for their unconditional help and encouragement, especially to my wife and our daughter, who are always an unlimited source of love and support for me.

Contents

Chapter 1

Introduction

Deep learning LeCun et al. (2015); Schmidhuber (2015) has led to breakthroughs in many application domains such as computer vision, natural language processing and speech recognition, yet the theoretical understanding of deep learning still lags far behind. To date, theoretical studies on deep learning mainly focus on three different topics: expressivity, optimization and generalization. Expressivity refers to the ability of neural networks to represent or approximate a function class, whereas optimization refers to the ability of practical learning algorithms such as (stochastic) gradient descent in finding a solution to fit the training data, and generalization is about understanding the performance of those models learned in this way on the new unseen data. This thesis contributes to the theory of the two former topics, with the main focus on optimization.

In terms of expressivity, universal approximation theorems Cybenko (1989); Hornik et al. (1989) state that any continuous function can be approximated to arbitrary precision on a compact set by a single hidden layer network with sufficient number of hidden neurons. It has been shown that this holds for neural networks with every non-polynomial activation function Leshno et al. (1993). This great expressive power of neural networks is further stressed in the regime of deep learning. In particular, several recent work Delalleau and Bengio (2011); Telgarsky (2016); Eldan and Shamir (2016); Cohen et al. (2016) have shown that there exist functions which can be computed efficiently by deep neural networks of linear or polynomial size but require exponential size for shallow networks to represent or approximate. More recently, Lin and Jegelka (2018) have shown that any Lebesgue integral function in d dimension can be uniformly approximated by a deep residual network with ReLU activation function and layers of alternating size 1 and d. This is in contrast to the standard feedforward neural networks as they cannot be universal function approximators if the widths of all hidden layers are not larger than the input dimension Johnson (2019); Beise et al. (2018).

However, merely knowing that a neural network can approximate any function is not sufficient to guarantee that popular algorithms such as (stochastic) gradient descent can always find a globally optimal solution on a fixed finite dataset in reasonable time. In fact, the problem of training neural networks has been shown to be NP-Hard, even when the network has just one or a few neurons (Sima, 2002; Blum and Rivest., 1989). Moreover, due to its non-convexity nature, the loss function can have potentially many distinct local minima (Auer et al., 1996), and so even if the learning algorithm is guaranteed to converge in polynomial time, it is hard to tell at the end of learning where the algorithm lands and what is the quality of the returned solution.

Surprisingly, despite all these hardness results, (stochastic) gradient descent algorithms still achieve excellent performance in practice Zhang et al. (2017), especially for training highly over-parameterized

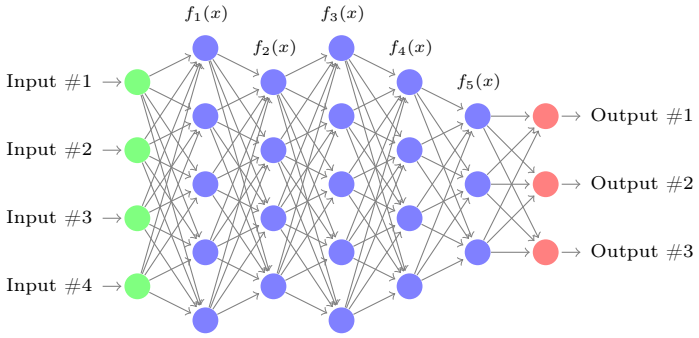


Figure 1.1: An example feedforward neural network with $L = 6$ layers and $d = 4, m = 3$.

networks [1]. Of course these successes are also fueled by massive amount of data and computational powers, but from an optimization perspective, it is still considered as perhaps one of the greatest mysteries in deep learning why such simple algorithms can frequently succeed in minimizing the highly non-convex function without facing any big problem with suboptimal local minima. This apparent gap between theory and practice seems to indicate that the training problem of practical deep networks is still very far from the worst-case scenario where it is known to be NP-Hard. This leads us to the question that whether these networks have any special structure which makes their associated loss function become favorable to local search algorithms like (stochastic) gradient descent? If so, can one identify several sufficient conditions on their architecture so that this is guaranteed? This is the kind of thinking that we would like to adopt in this thesis to overcome the problem of NP-Hardness and non-convexity.

This thesis focuses on the loss landscape of deep (convolutional) neural networks where one of the hidden layer is wide enough. We will analyze the global optimality of critical points of the loss function as they represent attractive solutions for gradient descent methods. Then we study connectivity and unboundedness of level sets and sublevel sets of the loss to gain more insights into the underlying geometric structure of the problem. Motivated by these results, we come up with a new class of convolutional neural network architectures that are more practically relevant and provably have a well-behaved loss surface. Lastly, we complement these results by studying limitations of neural networks in the regimes where our key conditions are (severely) violated.

As a foundation for the following chapters, we first briefly introduce below feedforward neural networks and their optimization problem. Then we give an overview of related work. We conclude this chapter with the summary of our main contributions to the recent advancements on theoretical understanding of deep over-parameterized neural networks.

1.1 Introduction to feedforward neural networks

A feedforward neural network consists of neurons that are ordered into multiple layers, as shown in Figure 1.1. The first layer and the last layer are refered to as the input layer and output layer respectively, and all the layers in between are called the hidden layers. The transformation between two consecutive layers is implemented by first performing an affine transformation on the output of the previous layer and then applying an elementwise nonlinear activation function on the outcome of that operation. The mathematical formulation of the network is described below.

Let L be the number of layers, d the input dimension, m the output dimension and n_k the number of neurons at the hidden layer k for every $1 \leq k \leq L-1$. By convention we assume that $n_0 = d$

[1] These are the networks which have more parameters than necessary to fit the training data.

and $n_L = m$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a continuous activation function. Amongst the most successful and popular activation functions in deep learning are the following ones

1. ReLU: $\sigma(x) = \max(0, x)$
2. Leaky-ReLU: $\sigma(x) = \max(\alpha x, x)$ for some $\alpha \in (0, 1)$
3. Sigmoid: $\sigma(x) = 1/(1 + e^{-x})$
4. Softplus: $\sigma(x) = \frac{1}{\alpha} \ln(1 + e^{\alpha x})$ for some $\alpha > 0$
5. Exponential linear units: $\sigma(x) = \begin{cases} e^x - 1 & x < 0 \\ x & x \geq 0 \end{cases}$

It is worth mentioning that ReLU and Leaky-ReLU are not differentiable at zero whereas the remaining functions are everywhere differentiable. Let $W_k \in \mathbb{R}^{n_{k-1} \times n_k}$ and $b_k \in \mathbb{R}^{n_k}$ be the weight matrix and bias vector of layer k. Let $f_k : \mathbb{R}^d \to \mathbb{R}^{n_k}$ be the function which maps every input $x \in \mathbb{R}^d$ to the output at layer k defined as

$$f_k(x) = \begin{cases} x & k = 0 \\ \sigma(W_k^T f_{k-1}(x) + b_k) & k \in [1, L-1] \\ W_L^T f_{L-1}(x) + b_L & k = L. \end{cases}$$

In the above formula and throughout this thesis, every (activation) function is applied elementwise. When σ is a linear function, e.g. $\sigma(x) = x$ for every $x \in \mathbb{R}$, the network is called a linear network since in this case the function computed by the network, given by $f_L(x)$, is just a linear function. In other cases, the network is called a nonlinear one. Let $(x_i)_{i=1}^N$ be the set of N training samples where $x_i \in \mathbb{R}^d$. Let $X = [x_1, \ldots, x_N]^T \in \mathbb{R}^{N \times d}$. By stacking the outputs of layer k for all the training samples into a matrix, we have

$$F_k = [f_k(x_1), f_k(x_2), \ldots, f_k(x_N)]^T \in \mathbb{R}^{N \times n_k}.$$

It follows from the above definition of f_k that

$$F_k = \begin{cases} X & k = 0 \\ \sigma(F_{k-1} W_k + \mathbf{1}_N b_k^T) & k \in [1, L-1] \\ F_{L-1} W_L + \mathbf{1}_N b_L^T & k = L. \end{cases}$$

Let $\theta := (W_l, b_l)_{l=1}^L$ be the set of all parameters of the network. By convention, we write $F_k(\theta)$ to denote the output of the network at layer k as a function θ, but sometimes we drop the argument and write just F_k if it is clear from the context. The output of the network is given by $F_L(\theta)$.

Fully connected networks and Convolutional neural networks. A feedforward neural network is called fully connected if every neuron on every hidden layer is connected to all neurons from the previous layer and the next layer. For instance, the network shown in Figure 1.1 is a fully connected one. On the other hand, if the neurons between two layers are only sparely connected and the weights of those connections are shared according to a special rule then we have a convolutional neural network (CNN). In this case, the sparsity patterns and weight sharing conditions are determined by a discrete convolution, that defines how the output of the previous layer is transformed to the next layer. A simple way to describe a CNN is to enforce the above weight matrices W_k to be sparse with shared coefficients. We note that in this case W_k only lies on a small subset of $\mathbb{R}^{n_{k-1} \times n_k}$ due to the additional structures, whereas in the case of fully connected networks W_k lies on the full space. In this thesis, a CNN is considered to be more general than a fully connected network since it might contain both convolutional layers and fully connected layers in the same architecture. In Chapter 2 we will give a more detailed description of CNNs as we analyze their optimization landscape.

1.1.1 Empirical risk minimization

The training problem of a deep network is usually framed as the following optimization problem

$$\min_{\theta \in \Omega} \Phi(\theta)$$

where Ω denotes the parameter space and $\Phi : \Omega \to \mathbb{R}$ the empirical risk defined as

$$\Phi(\theta) = \varphi(F_L(\theta))$$

where $\varphi : \mathbb{R}^{N \times m} \to \mathbb{R}$ is a convex loss function defined on the output of the network. In particular, for all the results in this thesis, if not stated otherwise, φ satisfies the following assumption.

Assumption 1.1.1 *The loss function $\varphi : \mathbb{R}^{N \times m} \to \mathbb{R}$ is convex.*

Assumption 1.1.1 implies that the loss Φ is convex w.r.t. the output of the network $F_L(\theta)$. Since F_L is an affine function of (W_L, b_L), it follows that Φ is also convex w.r.t. (W_L, b_L). However, we note that the loss Φ is generally non-convex w.r.t. all parameters of the network $\theta = (W_l, b_l)_{l=1}^L$. Typical loss functions in machine learning which satisfy Assumption 1.1.1 include:

1. The cross-entropy loss (for classification tasks)

$$\varphi(G) = \frac{1}{N} \sum_{i=1}^{N} -\log\Big(\frac{e^{G_{iy_i}}}{\sum_{k=1}^{m} e^{G_{ik}}}\Big), \tag{1.1}$$

   where $(x_i, y_i)_{i=1}^N$ is the training data with y_i being the ground-truth class of x_i.

2. The standard square loss (for classification/regression tasks)

$$\varphi(G) = \frac{1}{2} \|G - Y\|_F^2, \tag{1.2}$$

   where $Y \in \mathbb{R}^{N \times m}$ is the ground-truth matrix.

3. The multi-class Hinge-loss (for classification tasks)

$$\varphi(G) = \frac{1}{N} \sum_{i=1}^{N} \max_{j \neq y_i} \max(0, 1 - (G_{iy_i} - G_{ij})),$$

   where $(x_i, y_i)_{i=1}^N$ is the training data with y_i being the ground-truth class of x_i.

1.2 Related work

It is well known that the optimization problem of neural networks can have exponentially many local minima (Auer et al., 1996) and is in general NP-Hard (Blum and Rivest., 1989; Sima, 2002; Livni et al., 2014; Shamir, 2017; Shalev-Shwartz et al., 2017). However, it has been empirically observed (Dauphin et al., 2014; Goodfellow et al., 2015; Zhang et al., 2017) that the training problem of practical networks, which are often over-parameterized, is not hampered by suboptimal local minima.

In order to explain this apparent gap between hardness results and practical performance, many theoretical results have been recently developed in the literature. We can divide these work into two main categories. The first line of work is concerned with proving convergence guarantees for learning algorithms such as (stochastic) gradient descent and tensor power methods, which we review below in Section 1.2.1. The second line of work tries to characterize the structure of the loss function such as global optimality of critical points and/or local minima, topological properties of level sets and sublevel sets etc, which we refer to as the optimization landscape approach. This approach is more specific to the topic of the thesis which we review under Section 1.2.2.

1.2.1 Convergence theory of neural networks

Researches in this direction can be grouped into three different categories based on the network which they analyze: one hidden layer networks, deep linear networks and deep nonlinear networks.

Convergence results for one hidden layer networks Andoni et al. (2014) study shallow neural networks and show that randomly initialized gradient descent can learn any low degree polynomial given sufficiently many hidden units. Li and Liang (2018) study the learning problem of a two-layer over-parameterized ReLU network for multiclass problem where the data comes from well-separated distributions, and show that randomly initialized stochastic gradient descent with high probability will be biased towards a global minimum with better generalization error. Using the similar observation of Li and Liang (2018) that most of the pre-activation outputs do not change their signs over iterations, Du et al. (2019) show that gradient descent converges (with probability at least $1-\delta$) to a solution with zero training error if the number of hidden neurons scales as $\Omega(N^6/\delta^3)$. Instead of using gradient descent algorithms with random initialization like above, there are also work which use tensor initialization schemes Arora et al. (2014); Sedghi and Anandkumar (2015). In particular, Gautier et al. (2016) show that a particular class of feedforward neural networks with a regularized objective can be trained globally optimal with linear convergence rate using nonlinear spectral methods. Other work use optimal transport theory Chizat and Bach (2018) and mean field theory Mei et al. (2018); Nguyen (2018) to analyze the dynamics of (stochastic) gradient descent on over-parameterized models. Recently, Wang et al. (2018) propose a novel stochastic gradient descent algorithm to train one hidden layer ReLU networks with arbitrary number of hidden neurons to global optimality, despite the presence of potentially many bad local minima and saddle points. The authors claim that this is one of the first results of its kind which does not require any assumptions on the data distribution, network size or initialization.

There is a whole line of researches whose theoretical results are built upon the assumptions that the input distribution is Gaussian and the label is generated according to a planted neural network. Based on these (unrealistic) assumptions, they are able to show that (stochastic) gradient descent can recover the solution of the planted model. More specifically, Tian (2017) show that randomly initialized gradient descent can learn the underlying model for a two-layer ReLU network of the form $x \mapsto \sum_{i=1}^{K} \max(0, \langle w_i, x\rangle)$ with sufficiently number of hidden neurons. Brutzkus and Globerson (2017) consider a one hidden layer network with a single non-overlapping convolutional filter and ReLU activation function, and show that the problem is in general NP-Complete, but when the input distribution is Gaussian, gradient descent converges to the global optimum in polynomial time. Soltanolkotabi (2017) studies the problem of learning a ReLU network of the form $x \mapsto \max(0, \langle w, x\rangle)$ in the regime where the number of samples are less than the dimension of the weight vector w. By considering Gaussian inputs and a planted weight vector, they show that projected gradient descent, when initialized at zero, converges at a linear rate to the planted model. Li and Yuan (2017) consider a two-layer residual network of the form $x \mapsto \sum_{i=1}^{d} \max(0, x_i + \langle w_i, x\rangle)$ and show that if the inputs are Gaussian then randomly initialized gradient descent converges in polynomial time to the solution of a planted model. Going beyond Gaussian input distribution, Du et al. (2018b) show that (stochastic) gradient descent with random initialization can learn a single convolutional filter in polynomial time. Zhang et al. (2018) study the empirical risk of a one hidden layer ReLU network and show that gradient descent using a specific tensor initialization method can converge to the planted model at a linear rate given sufficiently large sample size. Similar study has also been done by Zhong et al. (2017) for smooth activation functions.

Convergence results for deep linear networks. Arora et al. (2018b) show that depth can accelerate the training of deep linear networks. Bartlett et al. (2018) study a particular class of deep linear networks similar to Hardt and Ma (2017) in which all the layers have the same width.

They show that if all the weight matrices are initialized to be the identity matrix, then gradient descent can converge to a global minimum with linear rate if the objective value at starting point is sufficient close to a global minimum, or a global minimum is attained when the product of all layers is positive definite. The follow-up work of Arora et al. (2019) has extended this results to more general deep linear networks. In particular, they show that gradient descent converges to a global minimum at linear rate if all the following conditions hold: 1) the dimensions of all hidden layer are not smaller than both the input and output dimension, 2) layers are initialized to be approximately balanced and 3) the initial loss is smaller than any loss obtainable with rank deficiencies. Ji and Telgarsky (2019) study the implicit regularization of gradient descent and gradient flow on linearly separable data. For logistic loss, they show that 1) every weight matrix asymptotically converges to its rank-1 approximation, 2) adjacent rank-1 weight matrix approximations are aligned and 3) the linear function induced by the network, namely the product of all weight matrices, converges to the same direction as the maximum margin solution.

Convergence results for deep nonlinear networks. Recently, Allen-Zhu et al. (2018a); Zou et al. (2018); Du et al. (2018a) have extended the result of Li and Liang (2018) to multiple layer networks of various architectures such as fully connected nets, convolutional neural nets and residual networks. Basically, they show that if the number of neurons of every hidden layer of a deep network scales as $\text{poly}(N, L)$, where N denotes the number of training samples and L the number of layers, then (stochastic) gradient descent will converge to a global minimum in polynomial time. In terms of the settings, while Du et al. (2018a) concentrates on gradient descent with square loss and smooth activation functions, the other two papers can deal with stochastic gradient descent, ReLU activation and different loss functions such as cross-entropy loss. On the other hand, Du et al. (2018a) can analyze the training of all layers in the network, whereas the work of Allen-Zhu et al. (2018a); Zou et al. (2018) fix the weights of the first hidden layer and/or the output layer, and only train the other hidden layers. In terms of the number of hidden neurons required for the result to hold, in the case of fully connected networks, Du et al. (2018a) require $\Omega(N^4 2^{O(L)})$ neurons per layer, whereas Allen-Zhu et al. (2018a) require $\Omega(N^{30}L^{30})$ and this is $\Omega(N^{26}L^{38})$ for Zou et al. (2018).

In summary, convergence theory for training algorithms of neural networks is a very active research area, and recent theoretical advancements along this research direction have certainly improved our theoretical understanding on how deep learning algorithms work under various scenarios. However, most of theoretical results so far are still quite limited in the sense that they often require prior knowledge about the data distribution, a modification of network structure and objective function, or they are for quite restricted networks, mostly one hidden layer networks. Interestingly, a few recent papers Allen-Zhu et al. (2018a); Zou et al. (2018); Du et al. (2018a) could show that gradient descent converges to a global minimum for certain deep nonlinear networks, but unfortunately these results require that the number of neurons "per hidden layer" has to grow polynomially large with the number of training samples and the depth.

At a high level, it's also unlikely that a pure convergence analysis of local search algorithms like gradient descent can help us explain the true reason why they succeed if there is no further information on the loss, especially given that the problem is highly non-convex. In many cases, it remains unclear whether this success comes from some special property of the algorithm, the loss function itself, or both. At this point, we believe that a comprehensive understanding of neural networks optimization landscape would shed light on this problem and help us understand better deep learning. There are a number of recent work in this direction which we review in the following.

1.2.2 Optimization landscape theory of neural networks

Theoretical work in this direction can be further divided into two smaller categories depending on the type of network architectures which they analyze: linear networks vs. non-linear networks.

Loss surface of linear networks. For linear neural networks, one has recently achieved a quite complete picture of the loss surface as it has been shown that every local minimum is a global minimum and every critical point that is not a global minimum is a saddle point. The result is first proved by Baldi and Hornik in 1988 for a one hidden layer autoencoder-decoder Baldi and Hornik (1988) with standard square loss and then extended to multiple layer networks by Kawaguchi (2016) under certain conditions on the training data. Since then there have been a series of follow-up work which try to improve this result or provide alternative (usually shorter) proofs. In particular, Lu and Kawaguchi (2017) simplify the proof of Kawaguchi (2016) and relax certain conditions on the training data. Hardt and Ma (2017) study deep linear networks (constant width per layer) with residual connections and show that every local minimum of an expected square loss for which the spectral norm of all the weight matrices is sufficiently small is a global minimum. Later, Yun et al. (2017) revise the work of Kawaguchi (2016) and provide necessary and sufficient conditions for a critical point to be a global minimum, which yields a simple checkable test for global optimality. The follow-up work of Zhou and Liang (2018) characterize the analytical forms of all critical points and global minima of deep linear networks with square loss. More recently, Laurent and v. Brecht (2018) extend the result of Kawaguchi (2016) to arbitrary convex differentiable loss using no assumption on the training data. Although all of these results are interesting from an optimization perspective, especially as the problem is still known to be non-convex Kawaguchi (2016), deep linear neural networks are not very helpful in practice as they learn just a linear classifier.

Loss surface of nonlinear networks. For one hidden layer networks, Yu and Chen (1995) show that if the activation function is sigmoid and the number of hidden neurons is $N-1$ then every local minimum is a global minimum. However, it seems that this result is not true in general given the recent findings of Yun et al. (2019) that bad local minima can exist for one hidden layer networks with generic training data and sigmoid activation. Soudry and Hoffer (2018) show that for piecewise linear activation functions, scalar output, quadratic loss, and standard normal distribution inputs, if the number of training samples N goes to infinity then the volume of differentiable regions of the empirical loss containing a suboptimal differentiable local minimum vanishes exponentially fast compared to the same volume of global minima.

For deep non-linear networks, the analysis of the optimization landscape becomes much more challenging and much less is known in the literature compared to the case of linear networks, or one hidden layer networks. Under this setting, the property that "every local minimum is a global minimum" does not hold anymore. In particular, Yun et al. (2019); Zhou and Liang (2018); Safran and Shamir (2018) have shown that bad local minima can exist for moderate-sized one hidden layer networks with many activation functions such as sigmoid, tanh and ReLU. Similar to above, merely knowing that a bad local minimum exists does not necessarily imply that (stochastic) gradient descent always converges to this solution, especially if the local minimum has very small basin of attraction and due to noise/stochasticity of the algorithm. At this point, we think that a better understanding of the loss landscape might be helpful. We review below some important prior work on the loss surface of deep nonlinear networks, which is the main topic of this thesis.

To our knowledge, one of the first work which analyze the loss surface of nonlinear and fully connected feedforward networks is the paper by Gori and Tesi (1992). They show that if the training data are linearly independent, that is $rank(X) = N$, and the network width is non-increasing from the first hidden layer to the output layer (i.e. $n_1 \geq \ldots \geq n_L$), and activation functions are continuously differentiable with non-zero derivative (e.g. sigmoid, softplus), then every critical point of the standard square loss for which all the weight matrices $(W_l)_{l=2}^L$ have full rank is a global minimum. This is a very neat result but relies on a strong assumption on the training data. In particular, the condition $rank(X) = N$ above implies that we cannot have more training samples than the input dimension (i.e. $N \leq d$), which is usually not the case in practice. In Chapter 2 we will show how to remove this condition by assuming that the network has a sufficiently wide hidden layer. Moreover, our results

can be applied to convolutional neural nets, which is one of the most successful architectures in deep learning and covers the fully connected nets of Gori and Tesi (1992) as a special case.

Choromanska et al. (2015a) studies the connection between the loss function of a fully connected feedforward neural network with ReLU activation function and the Hamiltonian of the spherical spin glass model from theoretical physics. By randomizing the nonlinear part of the network, the authors show that the critical values of the random loss function are located in a well-defined band lower bounded by the global minimum, and the number of bad local minima outside that band decreases exponentially with the size of the network. This is an inspiring result but is based on a number of unrealistic assumptions, which the authors have left as an open problem (Choromanska et al., 2015b), e.g. the condition that the activation outputs of all hidden neurons are independent from each other.

Safran and Shamir (2016) study fully connected networks with ReLU activation function and shows that under certain conditions there exists a continuous descent path between a pair of points in parameter space. In particular, let $\Phi : \Omega \to \mathbb{R}$ be the objective and $\theta_1, \theta_0 \in \Omega$ with $\Phi(\theta_0) > \Phi(\theta_1)$ so that there is a continuous path $\theta(\lambda), \lambda \in [0, 1]$ from θ_0 to θ_1 which satisfy the following conditions:

1. $\Phi(\theta_0) > \Phi(\mathbf{0})$
2. For some $\epsilon > 0$ and any $\lambda \in [0, 1]$, there exists $c_\lambda \geq 0$ such that $\Phi(c_\lambda \theta(\lambda)) \geq \Phi(\theta_0) + \epsilon$

Under above conditions, the authors show that there is a continuous path from θ_0 to some other point θ_2 where $\Phi(\theta_2) = \Phi(\theta_1)$ and the loss Φ is strictly monotonically decreasing along the path. They further show that if the number of hidden neurons of the last hidden layer goes to infinity and one draws the weights θ_0 of the network under a spherically symmetric distribution such as the standard normal distribution then the probability that the first condition is satisfied can be made close to $1/2$. One limitation of this result is that for cross-entropy loss and θ_1 having zero training error, the second condition is not fulfilled, which means that the result cannot tell us whether there is a continuous path from θ_0 to a solution with arbitrarily small objective value. To see that, by continuity of Φ there exists a sufficiently small neighborhood of θ_1 such that every point in this neighborhood also has zero training error and the objective value is smaller than θ_0. Now every continuous path from θ_0 to θ_1 mush go through some point θ_2 in this neighborhood. Since θ_2 has zero training error, the corresponding feature representations at the last hidden layer must be linearly separable. Since ReLU is positively homogeneous, any positive rescaling of θ_2 is the same as a positive rescaling of the output weight matrix, which thus can only lead to a decrease in the objective. That is, $\Phi(c\theta_2) \leq \Phi(\theta_2) < \Phi(\theta_0)$ for every $c \geq 0$, which violates the second condition. In Chapter 3, by studying sublevel sets of the loss function, we will show for large-sized networks that a continuous descent path indeed exists for any pair of points in parameter space and thus the above two conditions, which are technically hard to check, can be eliminated. In contrast to the probabilistic approach of Safran and Shamir (2016), all our results are for the deterministic setting (i.e. networks of fixed size).

Haeffele and Vidal (2017) study a special kind of network architectures in which the output of the network is given by a linear sum of the outputs of multiple parallel subnetworks which have exactly the same architecture but independent weights. For a class of regularized objective and positively homogeneous activation functions such as ReLU, the authors show that every local minimum of the objective for which all the weights of some subnetwork are identically zero is a global minimum. They further show that if the number of subnetworks is larger than Nm, with N being the number of training samples and m the output dimension, then there is a continuous descent path from any point in parameter space to a global minimum. This result has similar flavor to our results in this thesis in the sense that they both require the network to be sufficiently large compared to the size of the training set. However, as admitted by the authors in the paper, the conditions which their results rest upon are not realistic, especially as the result does not apply to standard network architectures. By checking their proof for the later result (see Theorem 2, Haeffele and Vidal (2017)), we found out that the proof is flawed since it relies on the fact that there is a continuous descent path from any

point in parameter space to a local minimum, which is not always true. Indeed, Figure 3.1 shows a counterexample where one can easily see that there is no such path to a local minimum, not even to a critical point, if one starts from any point far enough to the left of the negative real axis.

Liang et al. (2018) study the loss surface of fully connected networks with scalar output. Instead of minimizing the original objective $\Phi : \Omega \to \mathbb{R}$, they consider the following lifted function,

$$\tilde{\Phi}(\theta, w, a, b) = \sum_{i=1}^{N} l\Big(- y_i\big(f_L(x_i) + a\exp(w^T x_i + b)\big)\Big) + \lambda\frac{a^2}{2}.$$

where $l : \mathbb{R} \to \mathbb{R}$ is a polynomial hinge-loss such as $l(z) = \max(z+1, 0)^p$ for some $p \geq 3$. In this new objective, the standard output of the network $f_L(x_i)$ is modified by adding the exponential term $a\exp(w^T x_i + b)$. By analyzing the first/second order optimality conditions, they show that every local minimum of the new loss $\tilde{\Phi}$ is a global minimum. While this is an interesting result, it does not give us any insights on the landscape of the original loss function, which is the actual one used in practice. This can be important because in some cases as shown in Figure 3.1, merely knowing that "every local minimum is a global minimum" does not increase the chances or guarantee that gradient descent algorithms will converge to a global minimum. To complement this result, we study in Chapter 3 and Chapter 4 several sufficient conditions on the network architecture so that those bad local valleys as given in Figure 3.1 simply do not exist. Moreover, we do not require any non-standard modifications of the loss function as above.

Venturi et al. (2018) study the relationship between the intrinsic dimension of neural networks and the presence/absence of spurious valleys. The intrinsic dimension of a one hidden layer network is defined as the dimension of the following function space

$$q := \dim\big\{g(w, \cdot) \;\big|\; w \in \mathbb{R}^d\big\}$$

where $w \in \mathbb{R}^d$ is the incoming weight vector of a hidden neuron, and $g(w, \cdot) : \mathbb{R}^d \to \mathbb{R}$ is the function given by $g(w, x) = \sigma(\langle w, x\rangle)$ for every $x \in \mathbb{R}^d$. They analyze the training problem of one hidden layer networks with a population risk (assuming $\mathcal{D}$ is the data distribution) and standard square loss,

$$\min_{W_1, W_2} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \|W_2\sigma(W_1 x) - y\|_2^2 .$$

The key idea of the paper is that for appropriate choices of σ (e.g. polynomial activations), the above problem can be transformed into a linear one (for some mapping ϕ and ψ),

$$\min_{W_1, W_2} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \|W_2\psi(W_1)\phi(x) - y\|_2^2 .$$

In particular, by considering $(\psi(W_1), W_2)$ as the new variables of a linear network, the authors show that if the number of hidden neurons is greater than q then the loss has no spurious valley, and if it is greater than $2q$ then every sublevel set of the loss is connected. This result has similar flavor to our results in Chapter 3 where we study level sets and sublevel sets of the loss function. However, their result currently only applies to one hidden layer networks with expected square loss, and as admitted by the authors in the paper, an extension of their result, in particular the notion of intrinsic dimension, to multiple layer networks would require the number of hidden neurons to grow exponentially with depth. In contrast, our results in Chapter 3 are directly for the empirical risk, multiple layer networks of linear sizes (in terms of number of training samples) and for any standard convex loss function such as the above square loss, cross-entropy loss and Hinge-loss.

1.3 Summary of main contributions

This thesis consists of four main chapters. The content of each chapter is mainly taken from one of our publications which we report in the next section.

- Chapter 2: We analyze the optimization landscape of a class of deep convolutional neural networks with standard architecture, which include fully connected networks as a special case. By assuming that the network has a wide hidden layer with more neurons than the number of training samples, and the activation function is real analytic, we show that the set of training samples almost always become linearly independent in the feature space associated to the wide hidden layer, which significantly simplifies the optimization problem. Based on this, we derive a necessary and sufficient condition for a large subset of critical points to be globally optimal. Notably, this result is applicable to several existing CNN architectures such as VGG networks Simonyan and Zisserman (2015). Under a special case of the architecture where the wide layer is followed by a fully connected layer, we show that almost every critical point is a global minimum with zero training error, and there are uncountably many of them in parameter space.
- Chapter 3: We study sublevel sets of the loss function of a class of deep over-parameterized networks with fully connected architecture. We prove that if one of the hidden layers of the network has more neurons than the number of training samples, then essentially every sublevel set has to be connected and unbounded. One important consequence of this result is that the loss surface is well-behaved in the sense that there exists a continuous path from any point in parameter space on which the loss is non-increasing and gets arbitrarily close to the bottom of the loss surface. Another implication is that the loss surface has only one global valley, or equivalently, all global minima (if exist) are connected. Our results hold for standard deep fully connected networks with arbitrary convex losses, including the ones commonly used in deep learning such as cross-entropy loss, square loss, and hinge-loss, and a class of piecewise linear activation functions including Leaky-ReLU.
- Chapter 4: Gaining insights from the previous chapters, we propose in this chapter a new class of convolutional network architectures which are more practically relevant and provably have a well-behaved loss surface. The proposed networks can be seen as directed acyclic graphs from the input layer up to the last hidden layer, and allow for almost arbitrary weight sharing as in convolutional networks. In particular, we show that if a sufficient number of skip-connections are added from a random subset of hidden neurons (possibly from multiple hidden layers) to the output units, then the loss surface has no bad local valleys in the sense that there is a continuous path from any point in parameter space on which the loss is non-increasing and gets arbitrarily close to the minimal value of the loss. Our results are supported by experiments on standard benchmark datasets.
- Chapter 5: In contrary to the previous chapters which show that having a wide hidden layer in the architecture can help to make the loss landscape become well-behaved, in this chapter we study limitations of neural networks in terms of expressivity if they do not have any wide hidden layer. In particular, we show that if none of the hidden layers has more neurons than the input layer (plus some other mild conditions), then the network can only learn connected decision regions. This suggests that neural networks in general should be wide enough to learn disconnected decision regions – one of the aspects of expressivity. We further study the implication of this result on adversarial manipulation problems in deep learning.

1.3.1 Publication record

We list below all our papers (sorted by date of publication) that are publised during the work of this thesis. The content presented in the following chapters are mainly taken from the first four papers which contain our representative results that we have done on optimization landscape and expressivity of neural networks. The other papers are not presented in this thesis.

1. Q. Nguyen.
   On connected sublevel sets in deep learning.

International Conference on Machine Learning (ICML), 2019. Submitted.
See Chapter 3 in this thesis.

2. Q. Nguyen, M. Mukkamala and M. Hein.
On the loss landscape of a class of deep neural networks with no bad local valleys.
International Conference on Learning Representations (ICLR), 2019.
See Chapter 4 in this thesis.

3. Q. Nguyen and M. Hein.
Optimization landscape and expressivity of deep cnns.
International Conference on Machine Learning (ICML), 2018.
See Chapter 2 in this thesis.

4. Q. Nguyen, M. Mukkamala and M. Hein.
Neural networks should be wide enough to learn disconnected decision regions.
International Conference on Machine Learning (ICML), 2018.
See Chapter 5 in this thesis.

5. Q. Nguyen and M. Hein.
The loss surface of deep and wide neural networks.
International Conference on Machine Learning (ICML), 2017.

6. A. Gautier, Q. Nguyen and M. Hein.
Globally optimal training of generalized polynomial neural networks with nonlinear spectral methods.
Neural Information Processing Systems (NIPS) 2016.

7. Q. Nguyen, F. Tudisco, A. Gautier and M. Hein.
An efficient multilinear optimization framework for hypergraph matching.
IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI), 2017.

8. Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein and B. Schiele.
Latent embeddings for zero-shot classification.
IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

Chapter 2

Loss surface of fully connected and convolutional neural networks

2.1 Introduction

In this chapter, we analyze the optimization landscape of deep convolutional neural networks (CNNs), which cover fully connected networks described in Section 1.1 as a special case. All the main results in this chapter have been published at Nguyen and Hein (2018). To the best of our knowledge, this is one of the first work that theoretically analyzes the loss landscape of deep CNNs. CNNs are of high practical interest as they learn very useful representations with small number of parameters (Zeiler and Fergus, 2014; Mahendran and Vedaldi, 2015; Yosinski et al., 2015). Previously, we are only aware of Cohen and Shashua (2016) who study the expressiveness of CNNs with max-pooling layer and ReLU activation but with rather unrealistic filters (just 1×1) and no shared weights. In our setting we allow as well max pooling and general activation functions. Moreover, we can have an arbitrary number of filters and we study general convolutions as the filters need not be applied to regular structures like 3×3 but can be patch-based where the only condition is that all the patches have the size of the filter. Convolutional layers, fully connected layers and max-pooling layers can be combined in almost arbitrary order. Moreover, we study in this chapter the loss landscape and expressivity of a deep CNN where one layer is wide, in the sense that it has more neurons than the number of training points. While this assumption sounds at first quite strong, we want to emphasize that the popular VGG (Simonyan and Zisserman, 2015) and Inception networks (Szegedy et al., 2015b, 2016), see Table 2.1, fulfill this condition. We show that wide CNNs produce linearly independent feature representations at the wide layer and thus are able to fit the training data exactly (universal finite sample expressivity). This is even true with probability one when all the parameters up to the wide layer are chosen randomly[1]. We think that this partially explains the recent findings of Zhang et al. (2017) where they empirically show for several existing CNNs that they are able to fit random labels. Moreover, we provide necessary and sufficient conditions for global minima with zero training loss and show for a particular class of CNNs that almost all critical points are globally optimal, which to some extent explains why wide CNNs can be optimized so efficiently.

[1] for any probability measure on the parameter space which has a density with respect to the Lebesgue measure

Table 2.1: The maximum width of all hidden layers in several state-of-the-art CNN architectures compared to the size of ImageNet ($N \approx 1200K$) which represents by far one of the largest datasets in deep learning. All numbers are lower bounds on the true widths. Notations: n_k = width of layer k, e.g. if the output of a convolutional layer has size $H \times W \times C$ then $n_k = HWC$.

CNN Architecture	$M = \max_k n_k$	$M > N$
VGG(A-E) (Simonyan and Zisserman, 2015)	3000K($k = 1$)	yes
InceptionV3 (Szegedy et al., 2015b)	1300K($k = 3$)	yes
InceptionV4 (Szegedy et al., 2016)	1300K($k = 3$)	yes
SqueezeNet (Iandola et al., 2016)	1180K($k = 1$)	no
Enet (Paszke et al., 2016)	1000K($k = 1$)	no
GoogLeNet (Szegedy et al., 2015a)	800K($k = 1$)	no
ResNet (He et al., 2016)	800K($k = 1$)	no
Xception (Chollet, 2016)	700K($k = 1$)	no

2.2 Basic mathematical concepts

We start by reviewing some mathematical concepts from real analysis which are essential to the derivation of our results in this chapter. First is the concept of real analytic activation functions which will be used with our CNNs.

2.2.1 Real analytic function

Definition 2.2.1 *A real-valued function $f : \mathbb{R}^n \to \mathbb{R}$ is called analytic if for every $x \in \mathbb{R}^n$ the function f can be represented by a convergent power series in some neighborhood of x.*

It is well-known that all the elementary functions such as polynomials, exponential functions, trigonometric functions, logarithm and power functions are all real analytic. We refer to Krantz and Parks (2002) for a detailed treatment on this specific class of functions. Typical examples of activation functions that are real analytic in deep learning include:

1. sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$.
2. softplus: $\sigma(x) = \frac{1}{\alpha}\ln(1 + e^{\alpha x})$ for some $\alpha > 0$.
3. tanh: $\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

The following standard property of real analytic functions, see e.g. Mityagin (2015); Nguyen (2015), is particularly helpful for proving our key results (e.g. Theorem 2.4.5).

Lemma 2.2.2 *If $f : \mathbb{R}^n \to \mathbb{R}$ is a real analytic function which is not identically zero then the set $\{x \in \mathbb{R}^n \mid f(x) = 0\}$ has Lebesgue measure zero.*

Essentially Lemma 2.2.2 implies that if a real analytic function is constant on any open subset of the domain then it must be constant on the whole domain.

2.3 Convolutional neural networks

As briefly introduced at the end of Section 1.1, a CNN can be easily obtained from the standard formulation of feedforward neural networks by enforcing the weight matrices to have a specific sparsity structure and shared coefficients. However, CNNs employed in practice are often slightly more general

than that as they might contain pooling layers. To capture this feature, we find it helpful to discuss CNNs in this chapter using a more refined notation than in Section 1.1, which we briefly introduce in the following.

Let N be the number of training samples and denote by $X = [x_1, \dots, x_N]^T \in \mathbb{R}^{N \times d}, Y = [y_1, \dots, y_N]^T \in \mathbb{R}^{N \times m}$ the input resp. output matrix for the training data $(x_i, y_i)_{i=1}^N$, where d is the input dimension and m the output dimension.

Let L be the number of layers of the network, where each layer can be a convolutional layer, max-pooling layer or fully connected layer. The layers are indexed from $k = 0, 1, \dots, L$ which corresponds to the input layer, 1st hidden layer, ..., and the output layer. Let n_k be the width of layer k with the convention that $n_0 = d$ and $n_L = m$. Let $f_k : \mathbb{R}^d \to \mathbb{R}^{n_k}$ the function that maps every input to a feature vector at layer k.

Each convolutional layer consists of a number of patches of equal size, where every patch is a subset of neurons from the same layer. By convention, we assume that all the patches of a convolutional layer form a cover of the layer, that is, every neuron belongs to at least one of the patches, and that there exist no pair of patches that contain exactly the same subset of neurons. This implies that if there is one patch that covers the whole layer then it must be the only patch of the layer. Let P_k and l_k be the number of patches resp. the size of each patch at layer k for every $0 \leq k < L$. For every input $x \in \mathbb{R}^d$, let $\left\{x^1, \dots, x^{P_0}\right\} \in \mathbb{R}^{l_0}$ denote the set of patches at the input layer and $\left\{f_k^1(x), \dots, f_k^{P_k}(x)\right\} \in \mathbb{R}^{l_k}$ the set of patches at layer k. Each filter of the layer consists of the same set of patches. We denote by T_k the number of convolutional filters and by $W_k = [w_k^1, \dots, w_k^{T_k}] \in \mathbb{R}^{l_{k-1} \times T_k}$ the corresponding parameter matrix of the convolutional layer k for every $1 \leq k < L$. Each column of W_k corresponds to one filter. Furthermore, $b_k \in \mathbb{R}^{n_k}$ denotes the bias vector of layer k and $\sigma : \mathbb{R} \to \mathbb{R}$ the activation function. Note that one can use the same activation function for all layers but we use the general form to highlight the role of different layers. In this chapter, all functions are applied componentwise, and we denote by $[a]$ the set of integers $\{1, 2, \dots, a\}$ and by $[a, b]$ the set of integers from a to b.

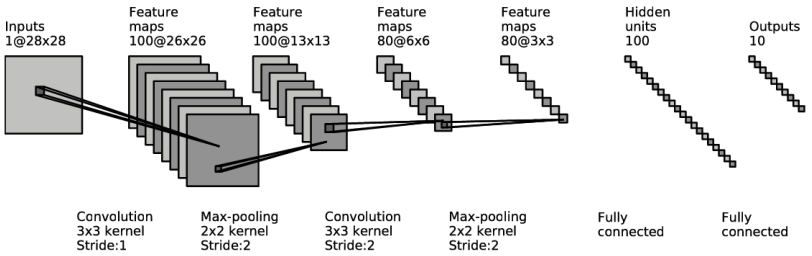


Figure 2.1: An example CNN architecture with layer widths $d = n_0 = 784, n_1 = 67600, n_2 = 16900, n_3 = 2880, n_4 = 720, n_5 = 100, n_6 = m = 10$. This network has pyramidal structure from layer 2 till the output layer, that is, $n_2 \geq \ldots \geq n_6$.

Definition 2.3.1 (Convolutional layer) *A layer k is called a convolutional layer if its output $f_k(x) \in \mathbb{R}^{n_k}$ is defined for every $x \in \mathbb{R}^d$ as*

$$f_k(x)_h = \sigma\Big(\left\langle w_k^t, f_{k-1}^p(x)\right\rangle + (b_k)_h\Big) \tag{2.1}$$

for every $p \in [P_{k-1}], t \in [T_k], h := (p-1)T_k + t$.

The value of each neuron at layer k is computed by first taking the inner product between a filter of layer k and a patch at layer $k-1$, adding the bias and then applying the activation function. The number of neurons at layer k is thus $n_k = T_k P_{k-1}$, which we denote as the width of layer k.

Our definition of a convolutional layer is quite general as every patch can be an arbitrary subset of neurons of the same layer and thus covers most of existing variants in practice.

Definition 2.3.1 includes the fully connected layer as a special case by using $P_{k-1} = 1, l_{k-1} = n_{k-1}, f^1_{k-1}(x) = f_{k-1}(x) \in \mathbb{R}^{n_{k-1}}, T_k = n_k, W_k \in \mathbb{R}^{n_{k-1} \times n_k}, b_k \in \mathbb{R}^{n_k}$. Thus we have only one patch which is the whole feature vector at this layer.

Definition 2.3.2 (Fully connected layer) *A layer k is called a fully connected layer if its output $f_k(x) \in \mathbb{R}^{n_k}$ is defined for every $x \in \mathbb{R}^d$ as*

$$f_k(x) = \sigma\Big(W_k^T f_{k-1}(x) + b_k\Big). \tag{2.2}$$

For some results in this chapter we also allow the network to have max-pooling layers.

Definition 2.3.3 (Max-pooling layer) *A layer k is called a max-pooling layer if its output $f_k(x) \in \mathbb{R}^{n_k}$ is defined for every $x \in \mathbb{R}^d$ and $p \in [P_{k-1}]$ as*

$$f_k(x)_p = \max\Big((f^p_{k-1}(x))_1, \ldots, (f^p_{k-1}(x))_{l_{k-1}}\Big). \tag{2.3}$$

A max-pooling layer just computes the maximum element of every patch from the previous layer. Since there are P_{k-1} patches at layer $k-1$, the number of output neurons at layer k is $n_k = P_{k-1}$.

Reformulation of Convolutional Layers: For each convolutional or fully connected layer, we denote by $\mathcal{M}_k : \mathbb{R}^{l_{k-1} \times T_k} \to \mathbb{R}^{n_{k-1} \times n_k}$ the linear map that returns for every parameter matrix $W_k \in \mathbb{R}^{l_{k-1} \times T_k}$ the corresponding full weight matrix $U_k = \mathcal{M}_k(W_k) \in \mathbb{R}^{n_{k-1} \times n_k}$. For convolutional layers, one can see that U_k plays the same role as the weight matrix W_k defined in Section 1.1. If layer k is fully connected then these two matrices are the same, and thus in this case we define $U_k = \mathcal{M}_k(W_k) = W_k$. Note that the mapping $\mathcal{M}_k$ depends on the patch structure of each convolutional layer k. For example, suppose that layer k has two filters of length 3, that is, $W_k = [w^1_k, w^2_k] = \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}$, and $n_{k-1} = 5$ and patches given by a 1D-convolution with stride 1 and no padding then:

$$U_k^T = \mathcal{M}_k(W_k)^T = \begin{bmatrix} a & b & c & 0 & 0 \\ d & e & f & 0 & 0 \\ 0 & a & b & c & 0 \\ 0 & d & e & f & 0 \\ 0 & 0 & a & b & c \\ 0 & 0 & d & e & f \end{bmatrix}.$$

The above ordering of the rows of U_k^T of a convolutional layer is determined by (2.1), in particular, the row index h of U_k^T is calculated as $h = (p-1)T_k + t$, which means for every given patch p one has to loop over all the filters t and compute the corresponding value of the output unit by taking the inner product of the h-th row of U_k^T with the whole feature vector of the previous layer. By ignoring max-pooling layers for the moment, the function $f_k : \mathbb{R}^d \to \mathbb{R}^{n_k}$ from above can be written as

$$f_k(x) = \begin{cases} x & k = 0 \\ \sigma\big(g_k(x)\big) & 1 \leq k \leq L-1 \\ g_L(x) & k = L \end{cases}$$

where $g_k : \mathbb{R}^d \to \mathbb{R}^{n_k}$ is the pre-activation output given as

$$g_k(x) = U_k^T f_{k-1}(x) + b_k, \quad \forall 1 \leq k \leq L$$

By stacking the outputs of layer k for all the training samples into a matrix, we have:

$$F_k = [f_k(x_1), \ldots, f_k(x_N)]^T \in \mathbb{R}^{N \times n_k},$$
$$G_k = [g_k(x_1), \ldots, g_k(x_N)]^T \in \mathbb{R}^{N \times n_k}.$$

It follows from above that

$$F_k = \begin{cases} X & k = 0 \\ \sigma(G_k) & 1 \leq k \leq L-1 \\ G_L & k = L \end{cases}, \quad \text{where } G_k = F_{k-1}U_k + \mathbf{1}_N b_k^T, \forall\, k \in [1, L]. \tag{2.4}$$

In the following, we refer to F_k as the output matrix at layer k.

2.3.1 Universal Assumptions

Throughout this chapter we assume that all the convolutional layers satisfy the following condition.

Assumption 2.3.4 (Convolutional Structure) *For every convolutional layer k, there exists a set of convolutional filters, represented by the matrix $W_k \in \mathbb{R}^{l_{k-1} \times T_k}$, for which the corresponding weight matrix $U_k = \mathcal{M}_k(W_k) \in \mathbb{R}^{n_{k-1} \times n_k}$ has full rank.*

Note that this condition is satisfied if every neuron belongs to at least one patch and there are no identical patches, which is often the case in practice. As the set of full rank matrices is dense in the space of all possible matrices, the following result follows immediately.

Lemma 2.3.5 *Let Assumption 2.3.4 holds. Then for every convolutional layer k, the set of $W_k \in \mathbb{R}^{l_{k-1} \times T_k}$ for which $U_k = \mathcal{M}_k(W_k) \in \mathbb{R}^{n_{k-1} \times n_k}$ does not have full rank has Lebesgue measure zero.*

Proof: Since $U_k = \mathcal{M}_k(W_k) \in \mathbb{R}^{n_{k-1} \times n_k}$ and $\mathcal{M}_k$ is a linear map, every entry of U_k is a linear function of the entries of W_k. Let $m = \min(n_{k-1}, n_k)$, then the set of low rank matrices U_k is characterized by a system of equations where the $\binom{\max(n_{k-1}, n_k)}{m}$ determinants of all $m \times m$ sub-matrices of U_k are zero. As the determinant is a polynomial in the entries of the matrix and thus a real analytic function, and the composition of analytic functions is again analytic, we get that each determinant is a real analytic function of W_k. By Assumption 2.3.4, there exists at least one W_k such that one of these determinants is non-zero. Thus by Lemma 2.2.2, the set of W_k where this determinant is zero has Lebesgue measure zero. As all the submatrices need to have low rank in order that U_k has low rank, we get that the set of W_k where U_k has low rank has Lebesgue measure zero. □

2.4 Main results

The organization of this section is as follows. In Section 2.4.1 we show that if CNNs have a sufficiently wide hidden layer then the set of training samples almost always become linearly independent in the feature space associated to this layer. We then discuss an implication of this result on capabilities of CNNs in memorizing all the training data, which is known as finite sample expressivity. In Section 2.4.2, we apply the above result to derive necessary and sufficient conditions for the global optimality of a large subset of critical points of the loss function. We further show that if the wide hidden layer is followed by a fully connected layer then all of these critical points are globally optimal.

2.4.1 CNNs learn linearly independent features

This sections shows that a class of standard CNN architectures with convolutional layers, fully connected layers and max-pooling layers plus standard activation functions like ReLU, sigmoid, softplus, etc are able to learn linearly independent features at every wide hidden layer if it has more neurons than the number of training samples. Our assumption on training data is the following.

Assumption 2.4.1 (Training data) *The patches of different training samples are non-identical, that is, $x_i^p \neq x_j^q$ for every $p, q \in [P_0], i, j \in [N], i \neq j$.*

Assumption 2.4.1 is quite weak, especially if the size of the input patches is large. If the assumption does not hold, one can add a small perturbation to the training samples: $\{x_1 + \epsilon_1, \ldots, x_N + \epsilon_N\}$. The set of $\{\epsilon_i\}_{i=1}^N$ where Assumption 2.4.1 is not fulfilled for the new dataset has measure zero. Moreover, $\{\epsilon_i\}_{i=1}^N$ can be chosen arbitrarily small so that the influence of the perturbation is negligible. Our main assumptions on the activation function of the hidden layers are the following.

Assumption 2.4.2 (Activation function) *The activation function σ is continuous, non-constant, and satisfies one of the following conditions:*

- *There exist $\mu_+, \mu_- \in \mathbb{R}$ s.t. $\lim_{t\to-\infty} \sigma(t) = \mu_-$ and $\lim_{t\to\infty} \sigma(t) = \mu_+$ and $\mu_+\mu_- = 0$*
- *There exist $\rho_1, \rho_2, \rho_3, \rho_4 \in \mathbb{R}_+$ s.t. $|\sigma(t)| \leq \rho_1 e^{\rho_2 t}$ for $t < 0$ and $|\sigma(t)| \leq \rho_3 t + \rho_4$ for $t \geq 0$*

Assumption 2.4.2 covers several standard activation functions, as shown by the following lemma.

Lemma 2.4.3 *The following activation functions satisfy Assumption 2.4.2:*

- *ReLU: $\sigma(t) = \max(0, t)$*
- *Sigmoid: $\sigma(t) = \frac{1}{1+e^{-t}}$*
- *Softplus: $\sigma_\alpha(t) = \frac{1}{\alpha}\ln(1 + e^{\alpha t})$ for some $\alpha > 0$*

We note that the softplus function is a smooth approximation of ReLU, in particular:

$$\lim_{\alpha\to\infty} \sigma_\alpha(t) = \lim_{\alpha\to\infty} \frac{1}{\alpha}\ln(1 + e^{\alpha t}) = \max(0, t). \tag{2.5}$$

One of the key results of this chapter is the following.

Theorem 2.4.4 (Linearly Independent Features) *Let Assumption 2.4.1 hold for the training sample. Consider a deep CNN architecture for which there exists some layer $1 \leq k \leq L-1$ such that*

1. *Layer 1 and layer k are convolutional or fully connected while all the other layers can be convolutional, fully connected or max-pooling*
2. *The width of layer k is larger than the number of training samples, $n_k = T_k P_{k-1} \geq N$*
3. *σ satisfy Assumption 2.4.2*

Then there exists a set of parameters of the first k layers $(W_l, b_l)_{l=1}^k$ such that the set of feature vectors $\{f_k(x_1), \ldots, f_k(x_N)\}$ are linearly independent. Moreover, $(W_l, b_l)_{l=1}^k$ can be chosen in such a way that all the weight matrices $U_l = \mathcal{M}_l(W_l)$ have full rank for every $1 \leq l \leq k$.

Theorem 2.4.4 implies that a large class of CNNs employed in practice with standard activation functions like ReLU, sigmoid or softplus can produce linearly independent features at any hidden layer if its width is larger than the size of training set. Figure 2.1 shows an example of a CNN architecture that satisfies the conditions of Theorem 2.4.4 at the first convolutional layer. Note that if a set of vectors is linearly independent then they are also linearly separable. In this sense, Theorem 2.4.4 suggests that CNNs can produce linearly separable features at every wide hidden layer.

Linear separability in neural networks has been recently studied by An et al. (2015), where the authors show that a two-hidden-layer fully connected network with ReLU activation function can transform any training set to be linearly separable while approximately preserving the distances of the training data at the output layer. Compared to An et al. (2015) our Theorem 2.4.4 is derived for CNNs with a wider range of activation functions. Moreover, our result shows even linear independence of features which is stronger than linear separability.

We want to stress that, in contrast to fully connected networks, for CNNs the condition $n_k \geq N$ of Theorem 2.4.4 does not imply that the network has a huge number of parameters as the layers k and $k+1$ can be chosen to be convolutional. In particular, the condition $n_k = T_k P_{k-1} \geq N$ can be fulfilled by increasing the number of filters T_k or by using a large number of patches P_{k-1} (however P_{k-1} is upper bounded by n_k), which is however only possible if l_{k-1} is small as otherwise our condition on the patches cannot be fulfilled. In total the CNN has only $l_{k-1}T_k + l_k T_{k+1}$ parameters versus $n_k(n_{k-1} + n_{k+1})$ for the fully connected network from and to layer k. Interestingly, the VGG-Net (Simonyan and Zisserman, 2015), where in the first layer small 3×3 filters and stride 1 is used, fulfills for ImageNet the condition $n_k \geq N$ for $k = 1$, as well as the Inception networks (Szegedy et al., 2015b, 2016), see Table 2.1.

One might ask now how difficult it is to find such parameters which generate linearly independent features at a hidden layer? Our next result shows that once analytic activation function such as sigmoid and softplus are used, the linear independence of features at layer k would hold with probability one even if one draws the parameters of the first k layers $(W_l, b_l)^k$ totally randomly [2].

Theorem 2.4.5 *Let Assumption 2.4.1 hold for the training samples. Consider a deep CNN for which there exists some layer* $1 \leq k \leq L-1$ *such that*

1. *Every layer from* 1 *to* k *is convolutional or fully connected*
2. *The width of layer* k *is larger than number of training samples, that is,* $n_k = T_k P_{k-1} \geq N$
3. σ *is real analytic and satisfies Assumption 2.4.2.*

Then the set of parameters of the first k *layers* $(W_l, b_l)_{l=1}^k$ *for which the set of feature vectors* $\{f_k(x_1), \ldots, f_k(x_N)\}$ *are* ***not*** *linearly independent has Lebesgue measure* ***zero****.*

Proof: Any linear function is real analytic and the set of real analytic functions is closed under addition, multiplication and composition, see e.g. Prop. 2.2.2 and Prop. 2.2.8 in Krantz and Parks (2002). As we assume that the activation function is real analytic, we get that the function f_k is a real analytic function of $(W_l, b_l)_{l=1}^k$ as it is the composition of real analytic functions. Now, we recall from our definition that $F_k = [f_k(x_1), \ldots, f_k(x_N)]^T \in \mathbb{R}^{N \times n_k}$ is the output matrix at layer k for all training samples. One observes that the set of low rank matrices F_k can be characterized by a system of equations such that all the $\binom{n_k}{N}$ determinants of all $N \times N$ sub-matrices of F_k are zero. As the determinant is a polynomial in the entries of the matrix and thus an analytic function of the entries and composition of analytic functions are again analytic, we conclude that each determinant is an analytic function of the network parameters of the first k layers. By Theorem 2.4.4, there exists at least one set of parameters of the first k layers such that one of these determinant functions is not

[2] for any probability measure on the parameter space which has a density with respect to the Lebesgue measure.

Table 2.2: The smallest singular value of the output matrix F_1 and F_3 at the first and second convolutional layer respectively of the trained network given in Figure 2.1 with varying number of convolutional filters T_1 and number of training samples $N = 60000$. The rank of a matrix $A \in \mathbb{R}^{m \times n}$ is estimated (see Chapter 2.6.1 in Press (2007)) by computing the singular value decomposition of A and counting the singular values which exceed the threshold $\frac{1}{2}\sqrt{m+n+1}\,\sigma_{\max}(A)\epsilon$, where ϵ is machine precision. For all filter sizes the output matrix F_1 has full rank. Zero training error is attained for $T_1 \geq 30$.

T_1	SIZE(F_1)	*rank*(F_1)	$\sigma_{\text{MIN}}(F_1)$	SIZE(F_3)	*rank*(F_3)	$\sigma_{\text{MIN}}(F_3)$	LOSS ($\times 10^{-5}$)	TRAIN ERROR	TEST ERROR
10	$N \times 6760$	**6760**	3.7×10^{-6}	$N \times 2880$	**2880**	2.0×10^{-2}	2.4	8	151
20	$N \times 13520$	**13520**	2.2×10^{-6}	$N \times 2880$	**2880**	7.0×10^{-4}	1.2	1	132
30	$N \times 20280$	**20280**	1.5×10^{-6}	$N \times 2880$	**2880**	2.4×10^{-4}	0.24	**0**	174
40	$N \times 27040$	**27040**	2.0×10^{-6}	$N \times 2880$	**2880**	2.2×10^{-3}	0.62	**0**	124
50	$N \times 33800$	**33800**	1.3×10^{-6}	$N \times 2880$	**2880**	3.9×10^{-5}	0.02	**0**	143
60	$N \times 40560$	**40560**	1.1×10^{-6}	$N \times 2880$	**2880**	4.0×10^{-5}	0.57	**0**	141
70	$N \times 47320$	**47320**	7.5×10^{-7}	$N \times 2880$	**2880**	7.1×10^{-3}	0.12	**0**	120
80	$N \times 54080$	**54080**	5.4×10^{-7}	$N \times 2880$	2875	4.9×10^{-18}	0.11	**0**	140
89	$N \times 60164$	**60000**	2.0×10^{-8}	$N \times 2880$	**2880**	8.9×10^{-10}	0.35	**0**	117
100	$N \times 67600$	**60000**	1.1×10^{-6}	$N \times 2880$	2856	8.5×10^{-27}	0.04	**0**	139

identically zero and thus by Lemma 2.2.2, the set of network parameters where this determinant is zero has Lebesgue measure zero. But as all submatrices need to have low rank in order that *rank*(F_k) $< N$, it follows that the set of parameters where *rank*(F_k) $< N$ has Lebesgue measure zero. □

Theorem 2.4.5 is a much stronger statement than Theorem 2.4.4, as it shows that for almost all weight configurations one gets linearly independent features at the wide layer. While Theorem 2.4.5 does not hold for the ReLU activation function as it is not an analytic function, we note again that one can approximate the ReLU function arbitrarily well using the softplus function (see 2.5), which is analytic function for any $\alpha > 0$ and thus Theorem 2.4.5 applies. It is an open question if the result holds also for the ReLU activation function itself. The condition $n_k \geq N$ is not very restrictive as several state-of-the art CNNs , see Table 2.1, fulfill the condition. Furthermore, we would like to stress that Theorem 2.4.5 is *not* true for deep linear networks. The reason is simply that the rank of a product of matrices can at most be the minimal rank among all the matrices. The nonlinearity of the activation function is thus critical (note that the identity activation function, $\sigma(x) = x$, does not fulfill Assumption 2.4.2).

To illustrate Theorem 2.4.5 we plot the rank of the feature matrices of the network in Figure 2.1. We use the MNIST dataset with $N = 60000$ training and 10000 test samples. We add small Gaussian noise $\mathcal{N}(0, 10^{-5})$ to every training sample so that Assumption 2.4.1 is fulfilled. We then vary the number of convolutional filters T_1 of the first layer from 10 to 100 and train the corresponding network with squared loss and sigmoid activation function using Adam (Kingma and Ba, 2015) and decaying learning rate for 2000 epochs. In Table 2.2 we show the smallest singular value of the feature matrices together with the corresponding training loss, training and test error. If number of convolutional filters is large enough (*i.e.* $T_1 \geq 89$), one has $n_1 = 26 \times 26 \times T_1 \geq N = 60000$, and the second condition of Theorem 2.4.5 is satisfied for $k = 1$. Table 2.2 shows that the feature matrices F_1 have full rank in all cases (and F_3 in almost all cases), in particular for $T_1 \geq 89$ as shown in Theorem 2.4.5. As expected when the feature maps of the training samples are linearly independent after the first layer (F_1 has rank 60000 for $T \geq 89$) the training error is zero and the training loss is close to zero (the GPU uses single precision). However, as linear independence is stronger than linear separability one can achieve already for $T < 89$ zero training error. It is interesting to note that Theorem 2.4.5 explains previous empirical observations. In particular, Czarnecki et al. (2017) have shown empirically that linear separability is often obtained already in the first few hidden layers of

the trained networks. This is done by attaching a linear classifier probe (Alain and Bengio, 2016) to every hidden layer in the network after training the whole network with backpropagation. The fact that Theorem 2.4.5 holds even if the parameters of the bottom layers up to the wide layer k are chosen randomly is also in line with recent empirical observations for CNN architectures that one has little loss in performance if the weights of the initial layers are chosen randomly without training (Jarrett et al., 2009; Saxe et al., 2011; Yosinski et al., 2014).

As a corollary of Theorem 2.4.4 we get the following universal finite sample expressivity for CNNs. In particular, a deep CNN with scalar output can perfectly fit any scalar-valued function for a finite number of inputs if the width of the last hidden layer is larger than the number of training samples.

Corollary 2.4.6 (Universal Finite Sample Expressivity) *Let Assumption 2.4.1 hold for the training samples. Consider a standard CNN with scalar output which satisfies the conditions of Theorem 2.4.4 at the last hidden layer $k = L - 1$. Let $f_L : \mathbb{R}^d \to \mathbb{R}$ be the network output given as*

$$f_L(x) = \sum_{j=1}^{n_{L-1}} \lambda_j f_{(L-1)j}(x) \quad \forall x \in \mathbb{R}^d$$

where $\lambda \in \mathbb{R}^{n_{L-1}}$ is the weight vector of the last layer. Then for every target $y \in \mathbb{R}^N$, there exists $\left\{\lambda, (W_l, b_l)_{l=1}^{L-1}\right\}$ so that it holds $f_L(x_i) = y_i$ for every $i \in [N]$.

Proof: Since the network satisfies the conditions of Theorem 2.4.4 for $k = L - 1$, there exists a set of parameters $(W_l, b_l)_{l=1}^{L-1}$ such that $\mathit{rank}(F_{L-1}) = N$. Let $F_L = [f_L(x_1), \ldots, f_L(x_N)]^T \in \mathbb{R}^N$ then it follows that $F_L = F_{L-1}\lambda$. Pick $\lambda = F_{L-1}^T(F_{L-1}F_{L-1}^T)^{-1}y$ then it holds $F_L = F_{L-1}\lambda = y$. □

The work of Cohen and Shashua (2016) is one the first ones which studies expressivity of CNNs. They show that CNNs with max-pooling and ReLU activation function can approximate any continuous function if the size of the network is unlimited. However, the number of convolutional filters in this result has to grow exponentially with the number of patches and they do not allow shared weights in their result, which is a standard feature of CNNs. In contrast, what is shown by Corollary 2.4.6 is not about capabilities of CNNs in approximating continuous functions, but the ability to fit a training dataset of fixed (and finite) size. This property is referred to as the universal finite sample expressivity of neural networks. In this sense, the above corollary implies that even a single convolutional layer network (i.e. $L = 2, k = 1$) can already fit the training data perfectly as long as the total number of neurons of the convolutional layer is at least the number of training samples.

For fully connected networks, universal finite sample expressivity has been studied by Zhang et al. (2017); Nguyen and Hein (2017), where they show a similar result for one hidden layer networks. While the number of training parameters of a (scalar-output) one hidden layer CNN with N hidden neurons is just $2N + T_1 l_0$, where T_1 is the number of convolutional filters and l_0 is the size of each filter, it is $Nd + 2N$ for a one hidden layer fully connected network with the same width. Now if we set the width of the CNN as $n_1 = T_1 P_0 = N$ in order to fulfill the condition of Corollary 2.4.6, then the number of training parameters of the CNN becomes $2N + Nl_0/P_0$, which is less than $3N$ if $l_0 \leq P_0$ compared to $(d+2)N$ for the fully connected case. In practice one almost always has $l_0 \leq P_0$ as l_0 is typically a small integer and P_0 is on the order of the dimension of the input. Therefore, the number of training parameters to achieve universal finite sample expressivity for CNNs is significantly smaller than for fully connected networks.

Of course, in practice it is more important that the network generalizes well to the new unseen data rather than just being able to fit the training data. By incorporating shared weights and sparsity structure, CNNs seem to implicitly regularize the model to achieve good performance. Thus although they can fit random labels or noise (Zhang et al., 2017) due to the universal finite sample expressivity as shown in Corollary 2.4.6, they seem still to be able to generalize well (Zhang et al.,

2017). Understanding generalization performance of optimization algorithms in deep learning is an important and active area, which is however beyond the scope of this thesis.

Now, we know that CNNs are expressive and can fit any finite dataset with enough number of neurons, why do we need to study the optimization landscape of CNNs? As briefly mentioned in Section 1.2, merely knowing that the network is able to fit the training data or that a global minimum with zero training error exists does not imply that the commonly used algorithms such as (stochastic) gradient descent necessarily converges to an optimal solution. Motivated by this view, we study in the next section global optimality conditions for the critical points of the loss.

2.4.2 Global optimality of critical points

In this section, we restrict our analysis to the use of least squares loss. However, as we show later that the network can produce exactly the target output (*i.e.* $F_L = Y$) for some choice of parameters, all our results can also be extended to any other loss function where the global minimum is attained at $F_L = Y$, such as the standard Hinge loss specified in Section 1.1. Let Ω denote the space of all network parameters. The final training objective $\Phi : \Omega \to \mathbb{R}$ is defined as

$$\Phi\Big((W_l, b_l)_{l=1}^L\Big) = \frac{1}{2} \|F_L - Y\|_F^2 \tag{2.6}$$

where F_L is defined as in (2.4), which is also the same as

$$F_L = \sigma(\ldots \sigma(XU_1 + \mathbf{1}_N b_1^T)\ldots)U_L + \mathbf{1}_N b_L^T,$$

where $U_l = \mathcal{M}_l(W_l)$ for every $1 \leq l \leq L$. We require the following assumptions on the CNN.

Assumption 2.4.7 (CNN Architecture) *Every layer in the network is a convolutional layer or fully connected layer and the output layer is fully connected. Moreover, there exists some hidden layer $1 \leq k \leq L-1$ such that the following holds:*

- *The width of layer k is larger than number of training samples, that is, $n_k = T_k P_{k-1} \geq N$*
- *σ satisfies Assumption 2.4.2. Moreover, σ is strictly monotonic and differentiable*
- *The network has pyramidal structure starting from layer $k+1$, that is, $n_{k+1} \geq \ldots \geq n_L$*

A typical example of a CNN that satisfies Assumption 2.4.7 with $k = 1$ can be found in Figure 2.1 where one disregards max-pooling layers and uses *e.g.* sigmoid or softplus activation function.

In the following, let us define for every $1 \leq k \leq L-1$ the subset $S_k \subseteq \Omega$,

$$S_k = \left\{(W_l, b_l)_{l=1}^L \;\middle|\; F_k, U_{k+2}, \ldots, U_L \text{have full rank}\right\}.$$

Basically S_k is the set of all network parameters where the feature matrix at layer k and all the weight matrices from layer $k+2$ till the output layer have full rank. In the following, we examine global optimality conditions for the critical points inside S_k. The next lemma shows that S_k covers almost the whole parameter space under an additional mild condition on the activation function.

Lemma 2.4.8 *Let Assumption 2.4.1 hold for the training samples and a deep CNN satisfy Assumption 2.4.7 for some layer $1 \leq k \leq L-1$. If σ is real analytic, then the complementary set $\Omega \setminus S_k$ has Lebesgue measure zero.*

Proof: One can see that

$$\Omega \setminus S_k \subseteq \left\{(W_l, b_l)_{l=1}^L \;\middle|\; rank(F_k) < N\right\} \cup \left\{(W_l, b_l)_{l=1}^L \;\middle|\; U_l \text{ has low rank for some layer } l\right\}.$$

By Theorem 2.4.5, it holds that the set $\left\{(W_l, b_l)_{l=1}^L \;\middle|\; rank(F_k) < N\right\}$ has Lebesgue measure zero. Moreover, it follows from Lemma 2.3.5 that the set $\left\{(W_l, b_l)_{l=1}^L \;\middle|\; U_l \text{ has low rank for some layer } l\right\}$ also has measure zero. Thus, $\Omega \setminus S_k$ has Lebesgue measure zero. □

In the next key lemma, we bound the objective function in terms of its gradient magnitude w.r.t. the weight matrix of layer k for which $n_k \geq N$. For every matrix $A \in \mathbb{R}^{m\times n}$, let $\sigma_{\min}(A)$ and $\sigma_{\max}(A)$ denote the smallest and largest singular value of A. Let $\|A\|_F = \sqrt{\sum_{i,j} A_{ij}^2}$, $\|A\|_{\min} := \min_{i,j} |A_{ij}|$ and $\|A\|_{\max} := \max_{i,j} |A_{ij}|$. From (2.4), and (2.6), it follows that Φ can be seen as a function of $(U_l, b_l)_{l=1}^L$, and thus we can use $\nabla_{U_k}\Phi$. If layer k is fully connected then $U_k = \mathcal{M}_k(W_k) = W_k$ and thus $\nabla_{U_k}\Phi = \nabla_{W_k}\Phi$. Otherwise, if layer k is convolutional then we note that $\nabla_{U_k}\Phi$ is "not" the true gradient of the training objective because U_k is not the true optimization parameter but W_k. In this case, the true gradient of Φ w.r.t. to the true parameter matrix W_k which consists of convolutional filters can be computed via the chain rule as

$$\frac{\partial \Phi}{\partial (W_k)_{rs}} = \sum_{i,j} \frac{\partial \Phi}{\partial (U_k)_{ij}} \frac{\partial (U_k)_{ij}}{\partial (W_k)_{rs}}$$

Please note that even though we write the partial derivatives with respect to the matrix elements, $\nabla_{W_k}\Phi$ resp. $\nabla_{U_k}\Phi$ are the matrices of the same dimension as W_k resp. U_k in the following.

Lemma 2.4.9 *Let a deep CNN satisfy Assumption 2.4.7 for some hidden layer $1 \leq k \leq L-1$. Then the following inequalities hold:*

$$\left\|\nabla_{U_{k+1}}\Phi\right\|_F \geq \sigma_{min}(F_k)\Big(\prod_{l=k+1}^{L-1} \sigma_{min}(U_{l+1}) \left\|\sigma'(G_l)\right\|_{min}\Big) \left\|F_L - Y\right\|_F,$$

$$\left\|\nabla_{U_{k+1}}\Phi\right\|_F \leq \sigma_{max}(F_k)\Big(\prod_{l=k+1}^{L-1} \sigma_{max}(U_{l+1}) \left\|\sigma'(G_l)\right\|_{max}\Big) \left\|F_L - Y\right\|_F.$$

Our next main result is motivated by the fact that empirically when training over-parameterized neural networks with shared weights and sparsity structure like CNNs, there seem to be no problems with sub-optimal local minima. In many cases, even when training labels are completely random, local search algorithms like stochastic gradient descent can converge to a solution with almost zero training error (Zhang et al., 2017). To understand better this phenomenon, we first characterize in the following Theorem 2.4.10 the set of points in parameter space with zero loss, and then analyze in Theorem 2.4.11 the loss surface for a special case of the network. We emphasize that our results hold for standard deep CNNs with convolutional layers with shared weights and fully connected layers.

Theorem 2.4.10 (Conditions for Zero Training Error) *Let Assumption 2.4.1 hold for the training sample and suppose that the CNN architecture satisfies Assumption 2.4.7 for some hidden layer $1 \leq k \leq L-1$. Let $\Phi : \Omega \to \mathbb{R}$ be defined as in (2.6). Given any point $(W_l, b_l)_{l=1}^L \in S_k$. Then it holds that $\Phi\Big((W_l, b_l)_{l=1}^L\Big) = 0$ if and only if $\nabla_{U_{k+1}}\Phi\Big|_{(W_l,b_l)_{l=1}^L} = 0$.*

Proof: If $\Phi\Big((W_l, b_l)_{l=1}^L\Big) = 0$ then it follows from the upper bound of Lemma 2.4.9 that $\nabla_{U_{k+1}}\Phi = 0$. For reverse direction, one has $(W_l, b_l)_{l=1}^L \in S_k$ and thus $rank(F_k) = N$ and U_l has full rank for every $l \in [k+2, L]$. Thus it holds $\sigma_{\min}(F_k) > 0$ and $\sigma_{\min}(U_l) > 0$ for every $l \in [k+2, L]$. Moreover, σ has non-zero derivative by Assumption 2.4.7 and thus $\|\sigma'(G_l)\|_{\min} > 0$ for every $l \in [k+1, L-1]$. This combined with the lower bound in Lemma 2.4.9 leads to $\Phi\Big(W_l, b_l)_{l=1}^L\Big) = \|F_L - Y\|_F = 0$. □

Lemma 2.4.8 shows that the set of points which are not covered by Theorem 2.4.10 has measure zero if the activation function is real analytic. The necessary and sufficient condition of Theorem 2.4.10 is

rather intuitive as it requires the gradient of the training objective to vanish w.r.t. the full weight matrix of layer $k+1$ regardless of the architecture of this layer. It turns out that if layer $k+1$ is fully connected, then this condition is always satisfied at a critical point, in which case we obtain that every critical point in S_k is a global minimum with exact zero training error. This is shown in the next Theorem 2.4.11, where we consider a standard classification task with m classes, $Z \in \mathbb{R}^{m \times m}$ denotes the full rank class encoding matrix e.g. the identity matrix and (X, Y) the training data such that $Y_{i:} = Z_{j:}$ whenever the training sample x_i belongs to class j for every $i \in [N], j \in [m]$.

Theorem 2.4.11 (Loss Surface of CNNs) *Let (X, Y, Z) be a training set for which Assumption 2.4.1 holds, the CNN architecture satisfies Assumption 2.4.7 for some hidden layer $1 \leq k \leq L-1$, and layer $k+1$ is fully connected. Let $\Phi : \Omega \to \mathbb{R}$ be defined as in* (2.6)*. Then the following holds*

- *Every critical point $(W_l, b_l)_{l=1}^L \in S_k$ is a global minimum with $\Phi\Big((W_l, b_l)_{l=1}^L\Big) = 0$*
- *There exist infinitely many global minima $(W_l, b_l)_{l=1}^L \in S_k$ with $\Phi\Big((W_l, b_l)_{l=1}^L\Big) = 0$*

Proof:

1. Consider a critical point in S_k. By first order optimality condition, we have $\nabla_{W_{k+1}} \Phi = 0$. Since layer $k+1$ is fully connected, it follows that $W_{k+1} = U_{k+1}$ and thus $\nabla_{U_{k+1}} \Phi = 0$. This combined with Theorem 2.4.10 yields the result.

2. One basically needs to show that there exist $(W_l, b_l)_{l=1}^L$ such that it holds: $\Phi\Big((W_l, b_l)_{l=1}^L\Big) = 0$, $\mathit{rank}(F_k) = N$ and $U_l = \mathcal{M}_l(W_l)$ has full rank $\forall l \in [k+2, L]$ Note that the last two conditions are fulfilled by the fact that $(W_l, b_l)_{l=1}^L \in S_k$.

   By Assumption 2.4.7, the subnetwork consisting of the first k layers satisfies the condition of Theorem 2.4.4. Thus by applying Theorem 2.4.4 to this subnetwork, one obtains that there exist $(W_l, b_l)_{l=1}^k$ such that $\mathit{rank}(F_k) = N$. In the following, we fix these layers and show how to pick $(W_l, b_l)_{l=k+1}^L$ such that $F_L = Y$. The main idea now is to make the output of all the training samples of the same class become identical at layer $k+1$ and thus they will have the same network output. Since there are only m classes, there would be only m distinct outputs for all the training samples at layer $L-1$. Thus if one can make these m distinct outputs become linearly independent at layer $L-1$ then there always exists a weight matrix W_L that realizes the target output Y as the last layer is just a linear map by assumption. Moreover, we will show that, except for max-pooling layers, all the parameters of other layers in the network can be chosen in such a way that all the weight matrices $(U_l)_{l=k+2}^L$ achieve full rank. In the following, we will present the detailed construction of the proof.

   Case 1: $k = L-1$
   It holds $\mathit{rank}(F_{L-1}) = N$. Pick $b_L = 0$ and $W_L = F_{L-1}^T (F_{L-1} F_{L-1}^T)^{-1} Y$. Since the output layer is fully connected, it follows from Definition 2.3.2 that $F_L = F_{L-1} W_L + \mathbf{1}_N b_L^T = Y$. Since $\mathit{rank}(F_k) = N$ and the full rank condition on $(W_l)_{l=k+2}^L$ is not active when $k = L-1$, it holds that $(W_l, b_l)_{l=1}^L \in S_k$ which finishes the proof.

   Case 2: $k = L-2$
   It holds $\mathit{rank}(F_{L-2}) = N$. Let $A \in \mathbb{R}^{m \times n_{L-1}}$ be a full row rank matrix such that $A_{ij} \in \mathrm{range}(\sigma)$. Note that $n_{L-1} \geq n_L = m$ due to Assumption 2.4.7. Let $D \in \mathbb{R}^{N \times n_{L-1}}$ be a matrix satisfying $D_{i:} = A_{j:}$ whenever x_i belongs to class j for every $i \in [N], j \in [m]$. By construction, F_{L-2} has full row rank, thus we can pick $b_{L-1} = 0, W_{L-1} = F_{L-2}^T (F_{L-2} F_{L-2}^T)^{-1} \sigma^{-1}(D)$. Since layer $L-1$ is fully connected, we have by Definition 2.3.2 that $F_{L-1} = \sigma(F_{L-2} W_{L-1} + \mathbf{1}_N b_{L-1}^T) = D$ and thus $(F_{L-1})_{i:} = D_{i:} = A_{j:}$ whenever x_i belongs to class j.

So far, our construction of the first $L-1$ layers lead to the fact that all the training samples belonging to the same class will have identical output at layer $L-1$. Since A has full row rank by our construction, we pick for the last layer $b_L = 0, W_L = A^T(AA^T)^{-1}Z$ where $Z \in \mathbb{R}^{m\times m}$ is the pre-defined class embedding matrix satisfying $rank(Z) = m$. One can easily check that $AW_L = Z$ and that $F_L = F_{L-1}W_L + \mathbf{1}_N b_L^T = F_{L-1}W_L$ where the later follows from Definition 2.3.2 as the output layer is fully connected. Now one can verify that $F_L = Y$. Indeed, for every x_i belongs to class j we have

$$(F_L)_{i:} = (F_{L-1})_{i:}^T W_L = (A)_{j:}^T W_L = Z_{j:} = Y_{i:}.$$

Moreover, since $rank(F_k) = N$ and $rank(W_L) = rank(A^T(AA^T)^{-1}Z) = m$, it holds that $(W_l, b_l)_{l=1}^L \in S_k$. Therefore, there exists $(W_l, b_l)_{l=1}^L \in S_k$ with $\Phi\Big((W_l, b_l)_{l=1}^L\Big) = 0$.

Case 3: $k \leq L-3$
It holds $rank(F_k) = N$. Let $E \in \mathbb{R}^{m\times n_{k+1}}$ be any matrix such that $E_{ij} \in \text{range}(\sigma)$ and $E_{ip} \neq E_{jq}$ for every $1 \leq i \neq j \leq N, 1 \leq p, q \leq n_{k+1}$. Let $D \in \mathbb{R}^{N\times n_{k+1}}$ satisfies $D_{i:} = E_{j:}$ for every x_i from class j. Pick $b_{k+1} = 0, W_{k+1} = F_k^T(F_kF_k^T)^{-1}\sigma^{-1}(D)$. Note that the matrix is invertible as F_k has been chosen to have full row rank. Since layer $k+1$ is fully connected by our assumption, it follows from Definition 2.3.2 that $F_{k+1} = \sigma(F_kW_{k+1} + \mathbf{1}_N b_{k+1}^T) = \sigma(F_kW_{k+1}) = D$ and thus

$$(F_{k+1})_{i:} = D_{i:} = E_{j:} \tag{2.7}$$

for every x_i from class j.

So far, our construction has led to the fact that all training samples belonging to the same class have identical output at layer $k+1$. The idea now is to see E as a new training data matrix of a subnetwork consisting of all layers from $k+1$ till the output layer L. In particular, layer $k+1$ can be seen as the input layer of this subnetwork and similarly, layer L can be seen as the output layer. Moreover, every row of $E \in \mathbb{R}^{m\times n_{k+1}}$ can be seen as a new training sample to this subnetwork. One can easily check that this subnetwork together with the new training data matrix E satisfy all the conditions of Theorem 2.4.4 at the last hidden layer $L-1$. In particular we have that:

- The rows of $E \in \mathbb{R}^{m\times n_{k+1}}$ are componentwise different from each other, and thus the input patches must be also distinct, and thus E satisfies Assumption 2.4.1
- Every layer from $k+1$ to $L-1$ is convolutional or fully connected by Assumption 2.4.7
- The width of layer $L-1$ is larger than the number of samples due to Assumption 2.4.7, that is, $n_{L-1} \geq n_L = m$
- σ satisfies Assumption 2.4.2 by Assumption 2.4.7

By applying Theorem 2.4.4 to this subnetwork and training data E, we obtain that there must exist $(W_l, b_l)_{l=k+2}^{L-1}$ for which all the weight matrices $(U_l)_{l=k+2}^{L-1}$ have full rank such that the set of corresponding m outputs at layer $L-1$ are linearly independent. In particular, let $A \in \mathbb{R}^{m\times n_{L-1}}$ be the corresponding outputs of E through this subnetwork then it holds that $rank(A) = m$. Intuitively, if one feeds $E_{j:}$ as an input at layer $k+1$ then one would get $A_{j:}$ as an output at layer $L-1$. This combined with (2.7) leads to the fact that if one now feeds $(F_{k+1})_{i:} = E_{j:}$ as an input at layer $k+1$ then one would get at layer $L-1$ the output $(F_{L-1})_{i:} = A_{j:}$ whenever x_i belongs to class j.

Last, we pick $b_L = 0, W_L = A^T(AA^T)^{-1}Z$. It follows that $AW_L = Z$. Since the output layer L is fully connected, it holds from Definition 2.3.2 that $F_L = F_{L-1}W_L + \mathbf{1}_N b_L^T = F_{L-1}W_L$.

One can verify now that $F_L = Y$. Indeed, for every sample x_i from class j it holds that

$$(F_L)_{i:} = (F_{L-1})_{i:}^T W_L = A_{j:}^T W_L = Z_{j:} = Y_{i:}.$$

Overall, we have shown that $\Phi = 0$. In addition, it holds $rank(F_k) = N$ from the construction of the first k layers. All the matrices $(U_l)_{l=k+2}^{L-1}$ have full rank by the construction of the subnetwork from $k+1$ till L. Moreover, $U_L = W_L$ also has full rank since $rank(W_L) = rank(A^T(AA^T)^{-1}Z) = m$. Therefore it holds $(W_l, b_l)_{l=1}^{L} \in S_k$.

□

Theorem 2.4.11 shows that the loss surface for this class of CNNs has a rather simple structure in the sense that every critical point lying inside S_k must be a global minimum with zero training error. Note that if the activation function is real analytic then the complement of S_k has measure zero (see Lemma 2.4.8), in which case S_k covers almost the whole parameter space. For the critical points lying outside S_k, it holds that either one of the weight matrices $\{U_{k+2}, \dots, U_L\}$ has low rank or the set of feature vectors at layer k is not linearly independent (*i.e.* F_k has low rank). Obviously, some of these critical points can also be global minima, but we conjecture that they cannot be suboptimal local minima due to the following reasons. First, it seems unlikely that a critical point with a low rank weight matrix is a suboptimal local minimum as this would imply that all possible full rank perturbations of the current solution must have larger/equal objective value. However, there is no term in the loss function which favors low rank solutions. Even for linear networks, it has been shown by Baldi and Hornik (1988) that all the critical points with low rank weight matrices have to be saddle points and thus cannot be suboptimal local minima. Second, a similar argument applies to the case where one has a critical point outside S_k such that the features are not linearly independent. In particular, any neighborhood of such a critical point contains points which have linearly independent features at layer k, from which it is easy to reach zero loss if one fixes the parameters of the first k layers and optimizes the loss w.r.t. the remaining ones. This implies that every small neighborhood of the critical point should contain points from which there exists a descent path that leads to a global minimum with zero loss, which contradicts the fact that the critical point is a suboptimal local minimum. All in all, if there are critical points lying outside the set S_k, then it is very "unlikely" that these are suboptimal local minima but rather also global minima, saddle points or local maxima. In the next chapter, we will see how the above arguments make sense as we show that there is indeed a continuous path from any low rank solution in parameter space on which the loss is non-increasing and gets arbitrarily close to a global minimum.

It remains an interesting open problem if the result of Theorem 2.4.11 can be transferred to the case where layer $k+1$ is also convolutional. In any case whether layer $k+1$ is fully connected or not, one might assume that a solution with zero training error still exists as it is usually the case for practical over-parameterized networks. However, note that Theorem 2.4.10 shows that at those points where the loss is zero, the gradient of Φ w.r.t. U_{k+1} must be zero as well. A special case of Theorem 2.4.11 is when all the layers of the network are fully connected, in which case we obtain a fully connected network and all the results of Theorem 2.4.11 hold without any modification. In particular, the combination of Theorem 2.4.11 and Lemma 2.4.8 immediately lead to the following corollary for fully connected networks.

Corollary 2.4.12 (Loss Surface of Fully Connected Nets) *Let (X, Y, Z) be a training set as in Theorem 2.4.11 with distinct training samples, that is, $x_i \neq x_j$ for every $i \neq j$. Suppose that a deep fully connected network has a wide hidden layer $k \in [L-1]$ so that $n_k \geq N$ and $n_{k+1} \geq \dots \geq n_L$, and the activation function $\sigma : \mathbb{R} \to \mathbb{R}$ is real analytic, strictly increasing and satisfies Assumption 2.4.2. Let $\Phi : \Omega \to \mathbb{R}$ be the training objective given by* (2.6)*. Then the following holds*

- *Every critical point $(W_l, b_l)_{l=1}^{L} \in S_k$ is a global minimum with $\Phi\Big((W_l, b_l)_{l=1}^{L}\Big) = 0$*
- *There exist infinitely many global minima $(W_l, b_l)_{l=1}^{L} \in S_k$ with $\Phi\Big((W_l, b_l)_{l=1}^{L}\Big) = 0$*
- *S_k covers almost the whole parameter space, i.e. $\Omega \setminus S_k$ has Lebesgue measure zero*

Corollary 2.4.12 shows that if one excludes from the whole parameter space a negligible set of measure zero then essentially every critical point is a global minimum with zero training error. This indicates that the loss surface restricted to S_k is rather well-behaved. For every point lying outside S_k, we will show in the next chapter (though with a slightly different setting) that there exists a continuous path connecting this point with some other point in S_k so that the loss is constant along the path.

2.5 Summary

We have analyzed the optimization landscape and expressivity of deep CNNs under realistic and practically relevant settings. For a class of CNNs with real analytic activation functions such as sigmoid and softplus, we prove that if the network has a sufficiently wide convolutional layer then for almost all configurations of network parameters, the set of feature representations at the wide layer for all the training samples are linearly independent. We use this result to derive a necessary and sufficient condition for a large subset of critical points to be globally optimal. In a special case where the wide layer is followed by a fully connected layer, we show almost every critical point of the loss is a global minimum with zero training error. Since state-of-the-art networks fulfill exactly or approximately our condition to have a wide convolutional layer, we believe that our results have shed some light on the reason why existing CNN architectures can be trained effectively in practice.

There are currently two limitations of our results. First, they only apply to smooth activation functions and thus cannot cover ReLU and Leaky-ReLU, which represent by far one of the most successful activation functions in deep learning. Second, we still lack a rigorous mathematical proof to address the case where a critical point lies outside of the set S_k as discussed below Theorem 2.4.11. The next chapter will address both of these issues through the analysis of level sets and sublevel sets. In particular, we will show for a class of fully connected networks with piecewise linear activation functions that if any given point in parameter space has low rank features or low rank weight matrices, then it can always be connected with a full-rank solution by a continuous path on which the loss is constant, and from this new point there exists always a continuous descent path leading to a global minimum.

2.6 Appendix

2.6.1 Proof of Lemma 2.4.3

- ReLU: It holds for every $t < 0$ that $\sigma(t) = \max(0, t) = 0 < e^t$, and for $t \geq 0$ that $\sigma(t) = t < t+1$. Thus ReLU satisfies the second condition of Assumption 2.4.2.
- Sigmoid: It holds that $\lim_{t\to-\infty} \frac{1}{1+e^{-t}} = 0$ and $\lim_{t\to\infty} \frac{1}{1+e^{-t}} = 1$. Thus σ satisfies the first condition of Assumption 2.4.2.
- Softplus: Since $1 + e^{\alpha t} \leq 2e^{\alpha t}$ for every $t \geq 0$, it holds for every $t \geq 0$ that

$$0 \leq \sigma_\alpha(t) = \frac{1}{\alpha}\log(1 + e^{\alpha t}) \leq \frac{1}{\alpha}\log(2e^{\alpha t}) = \frac{\log(2)}{\alpha} + t.$$

  Moreover, since $\log(1 + t) \leq t$ for $t > 0$, it holds $\log(1 + e^{\alpha t}) \leq e^{\alpha t}$ for every $t \in \mathbb{R}$. Thus it holds that $0 \leq \sigma_\alpha(t) \leq \frac{e^{\alpha t}}{\alpha}$ for every $t < 0$. Therefore σ_α satisfies the second condition of Assumption 2.4.2 for $\rho_1 = 1/\alpha, \rho_2 = \alpha, \rho_3 = 1, \rho_4 = \log(2)/\alpha$.

2.6.2 Proof of Theorem 2.4.4

To prove Theorem 2.4.4, we first show that Assumption 2.4.1 can be transported from the input to the output layer.

Lemma 2.6.1 *Let Assumption 2.4.1 hold for the training sample. Consider a standard deep CNN architecture which satisfies the following*

1. *The first layer is either convolutional or fully connected while all the other layers can be convolutional, fully connected or max-pooling*
2. *σ is a continuous and non-constant activation function*

Then for every layer $1 \leq k \leq L$, there exist a set of parameters of the first k layers $(W_l, b_l)_{l=1}^k$ such that it holds $f_k^p(x_i) \neq f_k^q(x_j)$ for every $p, q \in [P_k], i, j \in [N], i \neq j$. Moreover, $(W_l, b_l)_{l=1}^k$ can be chosen in such a way that, except for max-pooling layers, all the weight matrices $U_l = \mathcal{M}_l(W_l)$ have full rank for every $1 \leq l \leq k$.

Proof: The high-level idea of the proof is the following: since Assumption 2.4.1 holds for the training inputs, one can first pick (W_1, b_1) such that one transports the property of Assumption 2.4.1 to the feature maps of the training data at the next layer and thus to all higher layers by induction.

Since our definition of a convolutional layer includes fully connected layer as a special case, it is sufficient to prove the result for the general convolutional structure. Since the first layer is a convolutional layer by our assumption, we denote by $Q = [a^1, \dots, a^{T_1}] \in \mathbb{R}^{l_0 \times T_1}$ a matrix that contains the set of convolutional filters of the first layer. Note here that there are T_1 filters, namely $\left\{a^1, \dots, a^{T_1}\right\}$, where each filter $a^t \in \mathbb{R}^{l_0}$ for every $t \in [T_1]$. Let us define the set

$$S := \left\{Q \in \mathbb{R}^{l_0 \times T_1} \;\middle|\; \mathcal{M}_1(Q) \text{ has low rank}\right\} \cup \bigcup_{\substack{i \neq j \\ p,q \in [P_0] \\ t,t' \in [T_1]}} \left\{Q \in \mathbb{R}^{l_0 \times T_1} \;\middle|\; \left\langle a^t, x_i^p \right\rangle - \left\langle a^{t'}, x_j^q \right\rangle = 0\right\}.$$

Basically, S is the set of "true" parameter matrices of the first layer where the corresponding weight matrix $\mathcal{M}_1(Q)$ has low rank or there exists two patches of two different training samples that have the same inner product with some corresponding two filters. By Assumption 2.4.1 it holds that $x_i^p \neq x_j^q$ for every $p, q \in [P_0], i \neq j$, and thus the right hand side in the above formula of S is just the union of a finite number of hyperplanes which has Lebesgue measure zero. For the left hand side, it follows from Lemma 2.3.5 that the set of Q for which $\mathcal{M}_1(Q)$ does not have full rank has measure zero. Thus the left hand side of S is a set of measure zero. Since S is the union of two measure zero sets, it has also measure zero, and thus the complementary set $\mathbb{R}^{l_0 \times T_1} \setminus S$ must be non-empty and we choose $W_1 \in \mathbb{R}^{l_0 \times T_1} \setminus S$.

Since σ is a continuous and non-constant function, there exists an interval (μ_1, μ_2) such that σ is bijective on (μ_1, μ_2). We select and fix some matrix $Q = [a^1, \dots, a^{T_1}] \in \mathbb{R}^{l_0 \times T_1} \setminus S$ and select some $\beta \in (\mu_1, \mu_2)$. Let $\alpha > 0$ be a free variable and $W_1 = [w_1^1, \dots, w_1^{T_1}]$ where w_1^t denotes the t-th filter of the first layer. Let us pick

$$w_1^t = \alpha Q_{:t} = \alpha a^t, \quad (b_1)_h = \beta, \quad \forall t \in [T_1], h \in [n_1].$$

It follows that $W_1 = \alpha Q$ and thus $\mathcal{M}_1(W_1) = \mathcal{M}_1(\alpha Q) = \alpha \mathcal{M}_1(Q)$ as $\mathcal{M}_1$ is a linear map by our definition. Since $Q \notin S$ by construction, it holds that $\mathcal{M}_1(W_1)$ has full rank for every $\alpha \neq 0$. By Definition 2.3.1, it holds for every $i \in [N], p \in [P_0], t \in [T_1], h = (p-1)T_1 + t$ that

$$f_1(x_i)_h = \sigma(\left\langle w_1^t, x_i^p \right\rangle + (b_1)_h) = \sigma(\alpha \left\langle a^t, x_i^p \right\rangle + \beta).$$

Since $\beta \in (\mu_1, \mu_2)$, one can choose a sufficiently small positive α such that it holds $\alpha \langle a, x_i^p \rangle + \beta \in (\mu_1, \mu_2)$ for every $i \in [N], p \in [P_0], t \in [T_1]$. Under this construction, we will show that every entry of $f_1(x_i)$ must be different from every entry of $f_1(x_j)$ for $i \neq j$. Indeed, let us compare $f_1(x_i)_h$ and $f_1(x_j)_v$ for some $h = (p-1)T_1 + t, v = (q-1)T_1 + t'$ and $i \neq j$. It holds for sufficient small $\alpha > 0$ that

$$f_1(x_i)_h - f_1(x_j)_v = \sigma\left(\alpha \left\langle a^t, x_i^p \right\rangle + \beta\right) - \sigma\left(\alpha \left\langle a^{t'}, x_j^q \right\rangle + \beta\right) \neq 0 \tag{2.8}$$

where the last inequality follows from three facts. First, it holds $\langle a^t, x_i^p \rangle \neq \left\langle a^{t'}, x_j^q \right\rangle$ since $Q \notin S$. Second, for the chosen α the values of the arguments of the activation function σ lie within (μ_1, μ_2). Third, since σ is bijective on (μ_1, μ_2), it maps different inputs to different outputs.

Now, since the entries of $f_1(x_i)$ and that of $f_1(x_j)$ are already pairwise different from each other, their corresponding patches must be also different from each other no matter how the patches are organized in the architecture, that is,

$$f_1^p(x_i) \neq f_1^q(x_j) \quad \forall p, q \in [P_1], i, j \in [N], i \neq j.$$

Now, if the network has only one layer, *i.e.* $L = 1$, then we are done. Otherwise, we will prove via induction that this property can be translated to any higher layer. In particular, suppose that one has already constructed $(W_l, b_l)_{l=1}^k$ so that it holds

$$f_k(x_i)_h - f_k(x_j)_v \neq 0 \quad \forall h, v \in [n_k], i, j \in [N], i \neq j. \tag{2.9}$$

This is true for $k = 1$ due to (2.8). We will show below that (2.9) can also hold at layer $k+1$.

1. Case 1: Layer $k+1$ is convolutional or fully connected.

   Since (2.9) holds for k by our induction assumption, it must hold that

   $$f_k^p(x_i) \neq f_k^q(x_j) \quad \forall p, q \in [P_k], i, j \in [N], i \neq j.$$

   which means Assumption 2.4.1 also holds for the set of features at layer k. Thus one can follows the similar construction as done for layer 1 above by considering the output of layer k as input to layer $k+1$. Then one obtains that there exist (W_{k+1}, b_{k+1}) where $U_{k+1} = \mathcal{M}_{k+1}(W_{k+1})$ has full rank so that the similar inequality (2.8) now holds for layer $k+1$, which thus implies (2.9) holds for $k+1$.

2. Case 2: layer $k+1$ is max-pooling

   By Definition 2.3.3, it holds $n_{k+1} = P_k$ and one has for every $p \in [P_k]$

   $$f_{k+1}(x)_p = \max\left(\left(f_k^p(x)\right)_1, \ldots, \left(f_k^p(x)\right)_{l_k}\right).$$

   Since (2.9) holds at layer k by our induction assumption, every entry of every patch of $f_k(x_i)$ must be different from every entry of every patch of $f_k(x_j)$ for every $i \neq j$, that is, $\left(f_k^p(x_i)\right)_r \neq \left(f_k^q(x_j)\right)_s$ for every $r, s \in [l_k], p, q \in [P_k], i \neq j$. Therefore, their maximum elements cannot be the same, that is,

   $$f_{k+1}(x_i)_p \neq f_{k+1}(x_j)_q$$

   for every $p, q \in [n_{k+1}], i, j \in [N], i \neq j$, which proves (2.9) for layer $k+1$.

So far, we have proved that (2.9) holds for every $1 \leq k \leq L$. Thus it follows that for every layer k, there exists a set of parameters of the first k layers for which the patches at layer k of different training samples are pairwise different from each other, that is, $f_k^p(x_i) \neq f_k^q(x_j)$ for every $p, q \in [P_k], i \neq j$. Moreover, except for max-pooling layers, all the weight matrices up to layer k have been chosen to have full rank. □

Proof of Theorem 2.4.4 Let $A = F_k = [f_k(x_1)^T, \dots, f_k(x_N)^T] \in \mathbb{R}^{N \times n_k}$. Since our definition of a convolutional layer includes fully connected layer as a special case, it is sufficient to prove the result for convolutional structure. By Theorem's assumption, layer k is convolutional and thus it holds by Definition 2.3.1 that

$$A_{ij} = f_k(x_i)_j = \sigma\Big(\left\langle w_k^t, f_{k-1}^p(x_i) \right\rangle + (b_k)_j \Big)$$

for every $i \in [N], t \in [T_k], p \in [P_{k-1}]$ and $j = (p,t) := (p-1)T_k + t \in [n_k]$.

In the following, we show that there exists a set of parameters of the network such that $rank(A) = N$ and all the weight matrices $U_l = \mathcal{M}_l(W_l)$ have full rank.

First, one observes that the subnetwork consisting of all the layers from the input layer till layer $k-1$ satisfies the conditions of Lemma 2.6.1. Thus by applying Lemma 2.6.1 to this subnetwork, one obtains that there exist $(W_l, b_l)_{l=1}^{k-1}$ for which all the matrices $(U_l)_{l=1}^{k-1}$, except for max-pooling layers, have full rank and it holds that $f_{k-1}^p(x_i) \neq f_{k-1}^q(x_j)$ for every $p, q \in [P_{k-1}], i \neq j$. The main idea now is to fix the parameters of these layers and pick (W_k, b_k) such that $U_k = \mathcal{M}_k(W_k)$ has full rank and it holds $rank(A) = N$. Let us define the set

$$S = \bigcup_{\substack{i \neq j \\ p \in [P_{k-1}]}} \left\{ a \in \mathbb{R}^{l_{k-1}} \;\middle|\; \left\langle a, f_{k-1}^p(x_i) - f_{k-1}^p(x_j) \right\rangle = 0 \right\}.$$

From the above construction, it holds that $f_{k-1}^p(x_i) \neq f_{k-1}^p(x_j)$ for every $p \in [P_{k-1}], i \neq j$, and thus S is the union of a finite number of hyperplanes which thus has measure zero. Let us denote by $Q = [a^1, \dots, a^{T_k}] \in \mathbb{R}^{l_{k-1} \times T_k}$ a parameter matrix that contains all the convolutional filters of layer k in its columns. Pick $a^t \in \mathbb{R}^{l_{k-1}} \setminus S$ for every $t \in [T_k]$, so that it holds that $U_k = \mathcal{M}_k(Q)$ has full rank. Note here that such matrix Q always exists. Indeed, Q is chosen from a positive measure set as its columns (*i.e.* a^t) are picked from a positive measure set. Moreover, the set of matrices Q for which $\mathcal{M}_k(Q)$ has low rank has just measure zero due to Lemma 2.3.5. Thus there always exists at least one matrix Q so that all of its columns do not belong to S and that $\mathcal{M}_k(Q)$ has full rank. In the rest of the proof, the value of matrix Q is fixed. Let $\alpha \in \mathbb{R}$ be a free parameter. Since σ is a continuous and non-constant function, there exist a $\beta \in \mathbb{R}$ such that $\sigma(\beta) \neq 0$. Let the value of β be fixed as well. We construct the convolutional filters $W_k = [w_k^1, \dots, w_k^{T_k}]$ and the biases $b_k \in \mathbb{R}^{n_k}$ of layer k as follows. For every $p \in [P_{k-1}], t \in [T_k], j = (p,t)$, we define

$$w_k^t = -\alpha a^t, \quad (b_k)_j = \alpha \left\langle a^t, f_{k-1}^p(x_j) \right\rangle + \beta.$$

It follows that $W_k = -\alpha Q$ and thus $U_k = \mathcal{M}_k(W_k) = -\alpha \mathcal{M}_k(Q)$ as $\mathcal{M}_k$ is a linear map. Moreover, since $\mathcal{M}_k(Q)$ has full rank by construction, it holds that U_k has full rank for every $\alpha \neq 0$. As α varies, we get a family of matrices $A(\alpha) \in \mathbb{R}^{N \times n_k}$ where it holds for every $i \in [N], j = (p,t) \in [n_k]$ that

$$A(\alpha)_{ij} = \sigma\Big(\left\langle w_k^t, f_{k-1}^p(x_i) \right\rangle + (b_k)_j \Big) = \sigma\Big(\alpha \left\langle a^t, f_{k-1}^p(x_j) - f_{k-1}^p(x_i) \right\rangle + \beta \Big). \tag{2.10}$$

Note that each row of $A(\alpha)$ corresponds to one training sample and that permutations of the rows of $A(\alpha)$ do not change the rank of $A(\alpha)$. We construct a permutation γ of $\{1, 2, \dots, N\}$ as follows. For every $j = 1, 2, \dots, N$, let (p,t) be the tuple determined by j (the inverse transformation for given $j \in [n_k]$ is $p = \left\lceil \frac{j}{T_k} \right\rceil$ and $t = j - \Big(\left\lceil \frac{j}{T_k} \right\rceil - 1 \Big) T_k$) and define

$$\gamma(j) = \underset{i \in \{1,2,\dots,N\} \setminus \{\gamma(1), \dots, \gamma(j-1)\}}{\arg\min} \left\langle a^t, f_{k-1}^p(x_i) \right\rangle.$$

Note that $\gamma(j)$ is unique for every $1 \leq j \leq N$ since $a^t \notin S$. It is clear that γ constructed as above is a permutation of $\{1, 2, \dots, N\}$ since every time a different element is taken from the index set $[N]$. From the definition of $\gamma(j)$, it holds that for every $j = (p,t) \in [N], i \in [N], i > j$ that

$$\left\langle a^t, f_{k-1}^p(x_{\gamma(j)}) \right\rangle < \left\langle a^t, f_{k-1}^p(x_{\gamma(i)}) \right\rangle.$$

We can relabel the training data according to the permutation so that w.l.o.g we can assume that γ is the identity permutation, that is, $\gamma(j) = j$ for every $j \in [N]$, in which case it holds for every $j = (p,t) \in [N], i \in [N], i > j$ that

$$\left\langle a^t, f_{k-1}^p(x_j) \right\rangle < \left\langle a^t, f_{k-1}^p(x_i) \right\rangle . \tag{2.11}$$

Under the above construction of $(W_l, b_l)_{l=1}^k$, we are ready to show that there exist α for which $rank(A(\alpha)) = N$. Since σ satisfies Assumption 2.4.2, we consider the following cases.

1. Case 1: There are finite constants $\mu_+, \mu_- \in \mathbb{R}$ s.t. $\lim_{t\to-\infty} \sigma(t) = \mu_-$ and $\lim_{t\to\infty} \sigma(t) = \mu_+$ and $\mu_+\mu_- = 0$.
   Let us consider the first case where $\mu_- = 0$. From (2.10) and (2.11) one obtains

$$\lim_{\alpha\to+\infty} A(\alpha)_{ij} = \begin{cases} \sigma(\beta) & j = i \\ \mu_- = 0 & i > j \\ \eta_{ij} & i < j \end{cases} \tag{2.12}$$

   where η_{ij} is given for every $i < j$ where $j = (p,t)$ as

$$\eta_{ij} = \begin{cases} \mu_-, & \left\langle a^t, f_{k-1}^p(x_j) - f_{k-1}^p(x_i) \right\rangle < 0 \\ \mu_+, & \left\langle a^t, f_{k-1}^p(x_j) - f_{k-1}^p(x_i) \right\rangle > 0 \end{cases}$$

   Note that η_{ij} cannot be zero for $i \neq j$ because $a^t \notin S$. In the following, let us denote $A(\alpha)_{1:N,1:N}$ as a sub-matrix of $A(\alpha)$ that consists of the first N rows and columns. By the Leibniz-formula one has

$$\det(A(\alpha)_{1:N,1:N}) = \sigma(\beta)^N + \sum_{\pi\in S_N\setminus\{\gamma\}} \operatorname{sign}(\pi) \prod_{j=1}^N A(\alpha)_{\pi(j)j} \tag{2.13}$$

   where S_N is the set of all $N!$ permutations of the set $\{1,\dots,N\}$ and γ is the identity permutation. Now, one observes that for every permutation $\pi \neq \gamma$, there always exists at least one component j where $\pi(j) > j$ in which case it follows from (2.12) that

$$\lim_{\alpha\to\infty} \prod_{j=1}^N A(\alpha)_{\pi(j)j} = 0.$$

   Since there are only finitely many such terms in (2.13), one obtains

$$\lim_{\alpha\to\infty} \det(A(\alpha)_{1:N,1:N}) = \sigma(\beta)^N \neq 0$$

   where the last inequality follows from our choice of β. Since $\det(A(\alpha)_{1:N,1:N})$ is a continuous function of α, there exists $\alpha_0 \in \mathbb{R}$ such that for every $\alpha \geq \alpha_0$ it holds $\det(A(\alpha)_{1:N,1:N}) \neq 0$ and thus $rank(A(\alpha)_{1:N,1:N}) = N$ which further implies $rank(A(\alpha)) = N$. Thus the corresponding set of feature vectors $\{f_k(x_1),\dots,f_k(x_N)\}$ are linearly independent.

   For the case where $\mu_+ = 0$, one can argue similarly. The only difference is that one considers now the limit for $\alpha \to -\infty$. In particular, (2.10) and (2.11) lead to

$$\lim_{\alpha\to-\infty} A(\alpha)_{ij} = \begin{cases} \sigma(\beta) & i = j \\ \mu_+ = 0 & i > j \\ \eta_{ij} = 0 & i < j. \end{cases}$$

   For every permutation $\pi \neq \gamma$ there always exists at least one component j where $\pi(j) > j$, in which case it holds that

$$\lim_{\alpha\to-\infty} \prod_{j=1}^N A(\alpha)_{\pi(j)j} = 0.$$

and thus it follows from the Leibniz formula that

$$\lim_{\alpha\to-\infty} \det(A(\alpha)_{1:N,1:N}) = \sigma(\beta)^N \neq 0.$$

Since $\det(A(\alpha)_{1:N,1:N})$ is a continuous function of α, there exists $\alpha_0 \in \mathbb{R}$ such that for every $\alpha \leq \alpha_0$ it holds $\det(A(\alpha)_{1:N,1:N}) \neq 0$ and thus $rank(A(\alpha)_{1:N,1:N}) = N$ which further implies $rank(A(\alpha)) = N$. Thus the set of feature vectors at layer k are linearly independent.

2. Case 2: There are positive constants $\rho_1, \rho_2, \rho_3, \rho_4$ s.t. $|\sigma(t)| \leq \rho_1 e^{\rho_2 t}$ for $t < 0$ and $|\sigma(t)| \leq \rho_3 t + \rho_4$ for $t \geq 0$.
Our proof strategy is essentially similar to the previous case. Indeed, for every permutation $\pi \neq \gamma$ there always exist at least one component $j = (p,t) \in [N]$ where $\pi(j) > j$ in which case $\delta_j := \langle a^t, f_{k-1}^p(x_j) - f_{k-1}^p(x_{\pi(j)})\rangle < 0$ due to (2.11). For sufficiently large $\alpha > 0$, it holds that $\alpha\delta_j + \beta < 0$ and thus one obtains from (2.10) that

$$|A(\alpha)_{\pi(j)j}| = |\sigma(\alpha\delta_j + \beta)| \leq \rho_1 e^{\rho_2\beta} e^{-\alpha\rho_2|\delta_j|}.$$

If $\pi(j) = j$ then $|A(\alpha)_{\pi(j)j}| = |\sigma(\beta)|$ which is a constant. For $\pi(j) < j$, one notices that $\delta_j := \langle a^t, f_{k-1}^p(x_j) - f_{k-1}^p(x_{\pi(j)})\rangle$ can only be either positive or negative as $a^t \notin S$. In this case, if $\delta_j < 0$ then it can be bounded by the similar exponential term as above for sufficiently large α. Otherwise it holds $\alpha\delta_j + \beta > 0$ for sufficiently large $\alpha > 0$ and we get

$$|A(\alpha)_{\pi(j)j}| = |\sigma(\alpha\delta_j + \beta)| \leq \rho_3\delta_j\alpha + \rho_3\beta + \rho_4.$$

Overall, for sufficiently large $\alpha > 0$, there must exist positive constants P, Q, R, S, T such that it holds for every $\pi \in S_N \setminus \{\gamma\}$ that

$$\Big|\prod_{j=1}^{N} A(\alpha)_{\pi(j)j}\Big| \;\leq\; R(P\alpha + Q)^S e^{-\alpha T}.$$

The upper bound goes to zero as α goes to ∞. This combined with the Leibniz formula from (2.13), we get $\lim_{\alpha\to\infty} \det(A(\alpha)_{1:N,1:N}) = \sigma(\beta)^N \neq 0$. Since $\det(A(\alpha)_{1:N,1:N})$ is a continuous function of α, there exists $\alpha_0 \in \mathbb{R}$ such that for every $\alpha \geq \alpha_0$ it holds $\det(A(\alpha)_{1:N,1:N}) \neq 0$ and thus $rank(A(\alpha)_{1:N,1:N}) = N$ which implies $rank(A(\alpha)) = N$. Thus the set of feature vectors at layer k are linearly independent.

Overall, we have shown that there always exist $(W_l, b_l)_{l=1}^k$ such that the set of feature vectors $\{f_k(x_1), \ldots, f_k(x_N)\}$ at layer k are linearly independent. Moreover, $(W_l, b_l)_{l=1}^k$ have been chosen so that all the weight matrices $U_l = \mathcal{M}_l(W_l)$, except for max-pooling layers, have full rank for every $1 \leq l \leq k$.

2.6.3 Proof of Lemma 2.4.9

To prove Lemma 2.4.9, we first derive standard backpropagation in Lemma 2.6.2. In the following we use the Hadamard product $\circ$, which for $A, B \in \mathbb{R}^{m\times n}$ is defined as $A \circ B \in \mathbb{R}^{m\times n}$ with $(A \circ B)_{ij} = A_{ij}B_{ij}$. Let $\delta_{kj}(x_i) = \frac{\partial\Phi}{\partial g_{kj}(x_i)}$ be the derivative of Φ w.r.t. the value of unit j at layer k evaluated at a single sample x_i. We arrange these vectors for all training samples into a single matrix $\Delta_k = [\delta_{k:}(x_1), \ldots, \delta_{k:}(x_N)]^T \in \mathbb{R}^{N\times n_k}$.

Lemma 2.6.2 *The following hold:*

1. $\Delta_l = \begin{cases} F_L - Y, & l = L \\ (\Delta_{l+1}U_{l+1}^T) \circ \sigma'(G_l), & 1 \leq l \leq L-1 \end{cases}$

2. $\nabla_{U_l}\Phi = F_{l-1}^T\Delta_l$ *for every* $1 \le l \le L$

Proof:

1. By definition, it holds for every $i \in [N], j \in [n_L]$ that

$$(\Delta_L)_{ij} = \delta_{Lj}(x_i) = \frac{\partial \Phi}{\partial g_{Lj}(x_i)} = f_{Lj}(x_i) - y_{ij}$$

and hence, $\Delta_L = F_L - Y$.

For every $k+1 \le l \le L-1$, the output of the network for a single training sample can be written as the composition of differentiable functions (*i.e.* the outputs of all layers from $l+1$ till the output layer), and thus the chain rule yields for every $i \in [N], j \in [n_l]$ that

$$\begin{aligned}(\Delta_l)_{ij} = \delta_{lj}(x_i) = \frac{\partial \Phi}{\partial g_{lj}(x_i)} &= \sum_{s=1}^{n_{l+1}} \frac{\partial \Phi}{\partial g_{(l+1)s}(x_i)} \frac{\partial g_{(l+1)s}(x_i)}{\partial f_{lj}(x_i)} \frac{\partial f_{lj}(x_i)}{\partial g_{lj}(x_i)} \\ &= \sum_{s=1}^{n_{l+1}} \delta_{(l+1)s}(x_i)(U_{l+1})_{js}\sigma'(g_{lj}(x_i)) \\ &= \sum_{s=1}^{n_{l+1}} (\Delta_{(l+1)})_{is}(U_{l+1})_{sj}^T\sigma'((G_l)_{ij})\end{aligned}$$

and hence $\Delta_l = (\Delta_{l+1}U_{l+1}^T) \circ \sigma'(G_l)$.

2. For every $k+1 \le l \le L-1, r \in [n_{l-1}], s \in [n_l]$, one has

$$\frac{\partial \Phi}{\partial (U_l)_{rs}} = \sum_{i=1}^N \frac{\partial \Phi}{\partial g_{ls}(x_i)} \frac{\partial g_{ls}(x_i)}{\partial (U_l)_{rs}} = \sum_{i=1}^N \delta_{ls}(x_i) f_{(l-1)r}(x_i) = \sum_{i=1}^N (F_{l-1}^T)_{ri}(\Delta_l)_{is} = \left(F_{l-1}^T\Delta_l\right)_{rs}$$

and hence $\nabla_{U_l}\Phi = F_{l-1}^T\Delta_l$.

□

The following straightforward inequalities are also helpful to prove Lemma 2.4.9. Let $\lambda_{\min}(\cdot)$ and $\lambda_{\max}(\cdot)$ denotes the smallest and largest eigenvalue of a matrix.

Lemma 2.6.3 *Let* $A \in \mathbb{R}^{m\times n}$ *with* $m \ge n$. *Then* $\sigma_{max}(A)\|x\|_2 \ge \|Ax\|_2 \ge \sigma_{min}(A)\|x\|_2$ *for every* $x \in \mathbb{R}^n$.

Proof: Since $m \ge n$, it holds that $\sigma_{\min}(A) = \sqrt{\lambda_{\min}(A^TA)} = \sqrt{\min \frac{x^TA^TAx}{x^Tx}} = \min \frac{\|Ax\|_2}{\|x\|_2}$ and thus $\sigma_{\min}(A) \le \frac{\|Ax\|_2}{\|x\|_2}$ for every $x \in \mathbb{R}^n$. Similarly, it holds $\sigma_{\max}(A) = \sqrt{\lambda_{\max}(A^TA)} = \sqrt{\max \frac{x^TA^TAx}{x^Tx}} = \max \frac{\|Ax\|_2}{\|x\|_2}$ and thus $\sigma_{\max}(A) \ge \frac{\|Ax\|_2}{\|x\|_2}$ for every $x \in \mathbb{R}^n$. □

Lemma 2.6.4 *Let* $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times p}$ *with* $m \ge n$. *Then* $\sigma_{max}(A)\|B\|_F \ge \|AB\|_F \ge \sigma_{min}(A)\|B\|_F$.

Proof: Since $m \ge n$, it holds that $\lambda_{\min}(A^TA) = \sigma_{\min}(A)^2$ and $\lambda_{\max}(A^TA) = \sigma_{\max}(A)^2$. Thus we have $\|AB\|_F^2 = \operatorname{tr}(B^TA^TAB) \ge \lambda_{\min}(A^TA)\operatorname{tr}(B^TB) = \sigma_{\min}(A)^2\|B\|_F^2$. Similarly, it holds $\|AB\|_F^2 = \operatorname{tr}(B^TA^TAB) \le \lambda_{\max}(A^TA)\operatorname{tr}(B^TB) = \sigma_{\max}(A)^2\|B\|_F^2$. □

Proof of Lemma 2.4.9 We first prove the lower bound. Let $\mathbb{I}_m$ denotes an m-by-m identity matrix and $\otimes$ the Kronecker product. From Lemma 2.6.2 it holds $\nabla_{U_{k+1}}\Phi = F_L^T\Delta_{k+1}$ and thus

$$vec(\nabla_{U_{k+1}}\Phi) = (\mathbb{I}_{n_{k+1}} \otimes F_k^T)\, vec(\Delta_{k+1}).$$

It follows that

$$\left\|\nabla_{U_{k+1}}\Phi\right\|_F = \left\|(\mathbb{I}_{n_{k+1}} \otimes F_k^T)\, vec(\Delta_{k+1})\right\|_2 \geq \sigma_{\min}(F_k)\, \|vec(\Delta_{k+1})\|_2 = \sigma_{\min}(F_k)\, \|\Delta_{k+1}\|_F \quad (2.14)$$

where the inequality follows from Lemma 2.6.3 for the matrix $(\mathbb{I}_{n_{k+1}} \otimes F_k^T) \in \mathbb{R}^{n_k n_{k+1} \times N n_{k+1}}$ with $n_k \geq N$ by Assumption 2.4.7. Using Lemma 2.6.2 again, one has

$$\begin{aligned}
\|\Delta_{k+1}\|_F &= \left\|(\Delta_{k+2}U_{k+2}^T) \circ \sigma'(G_{k+1})\right\|_F \\
&\geq \|\sigma'(G_{k+1})\|_{\min} \left\|\Delta_{k+2}U_{k+2}^T\right\|_F \\
&= \|\sigma'(G_{k+1})\|_{\min} \left\|U_{k+2}\Delta_{k+2}^T\right\|_F \\
&\geq \|\sigma'(G_{k+1})\|_{\min} \sigma_{\min}(U_{k+2}) \|\Delta_{k+2}\|_F
\end{aligned}$$

where the last inequality follows from Lemma 2.6.4 for the matrices $U_{k+2} \in \mathbb{R}^{n_{k+1}\times n_{k+2}}$ and Δ_{k+2}^T with $n_{k+1} \geq n_{k+2}$ by Assumption 2.4.7. By repeating this for $\|\Delta_{k+2}\|_F, \ldots, \|\Delta_{L-1}\|_F$, one gets

$$\|\Delta_{k+1}\|_F \geq \Big(\prod_{l=k+1}^{L-1} \|\sigma'(G_l)\|_{\min} \sigma_{\min}(U_{l+1})\Big) \|\Delta_L\|_F = \Big(\prod_{l=k+1}^{L-1} \|\sigma'(G_l)\|_{\min} \sigma_{\min}(U_{l+1})\Big) \|F_L - Y\|_F \quad (2.15)$$

From (2.14), (2.15), one obtains

$$\left\|\nabla_{U_{k+1}}\Phi\right\|_F \geq \sigma_{\min}(F_k)\Big(\prod_{l=k+1}^{L-1} \|\sigma'(G_l)\|_{\min} \sigma_{\min}(U_{l+1})\Big) \|F_L - Y\|_F$$

which proves the lower bound.

The proof for upper bound is similar. Indeed one has

$$\left\|\nabla_{U_{k+1}}\Phi\right\|_F = \left\|(\mathbb{I}_{n_{k+1}} \otimes F_k^T)\, vec(\Delta_{k+1})\right\|_2 \leq \sigma_{\max}(F_k)\, \|vec(\Delta_{k+1})\|_2 = \sigma_{\max}(F_k)\, \|\Delta_{k+1}\|_F \quad (2.16)$$

where the inequality follows from Lemma 2.6.3 Now, one has

$$\begin{aligned}
\|\Delta_{k+1}\|_F &= \left\|(\Delta_{k+2}U_{k+2}^T) \circ \sigma'(G_{k+1})\right\|_F \\
&\leq \|\sigma'(G_{k+1})\|_{\max} \left\|\Delta_{k+2}U_{k+2}^T\right\|_F \\
&= \|\sigma'(G_{k+1})\|_{\max} \left\|U_{k+2}\Delta_{k+2}^T\right\|_F \\
&\leq \|\sigma'(G_{k+1})\|_{\max} \sigma_{\max}(U_{k+2}) \|\Delta_{k+2}\|_F
\end{aligned}$$

where the last inequality follows from Lemma 2.6.4. By repeating this chain of inequalities for $\|\Delta_{k+2}\|_F, \ldots, \|\Delta_{L-1}\|_F$, one obtains

$$\|\Delta_{k+1}\|_F \leq \Big(\prod_{l=k+1}^{L-1} \|\sigma'(G_l)\|_{\max} \sigma_{\max}(U_{l+1})\Big) \|\Delta_L\|_F = \Big(\prod_{l=k+1}^{L-1} \|\sigma'(G_l)\|_{\max} \sigma_{\max}(U_{l+1})\Big) \|F_L - Y\|_F\,. \quad (2.17)$$

From (2.16), (2.17), one obtains that

$$\left\|\nabla_{U_{k+1}}\Phi\right\|_F \leq \sigma_{\max}(F_k)\Big(\prod_{l=k+1}^{L-1} \|\sigma'(G_l)\|_{\max} \sigma_{\max}(U_{l+1})\Big) \|F_L - Y\|_F$$

which proves the upper bound.

Chapter 3

Topology of level sets and sublevel sets of the training loss function

In Chapter 2 we have analyzed the loss surface of CNNs by studying global optimality of critical points and existence of zero training error solutions. In the conclusion, we have left it as an open problem how to analyze the loss surface of neural networks with non-smooth activation functions, and how to deal with situations where the weight matrices have low rank or the set of feature representations at the wide layer are not linearly independent.

In this chapter, we address both of these problems through the analysis of level sets and sublevel sets. In particular, for a class of deep fully connected networks, we show that if the network has a sufficiently wide hidden layer then every sublevel set of the loss is connected. This not only implies that there is a continuous descent path from any point in parameter space to a global minimum, but also that all the global minima are connected. In light of the second problem mentioned above, we show that from any point in parameter space where the feature representations are not full rank or one of the weight matrices have low rank, there always exists a continuous path to some other point where all the full-rank conditions are satisfied, and in particular, the loss is constant along the path. Concerning the first problem, our results in this chapter are directly applicable to a class of piecewise linear activation functions including Leaky-ReLU, which was not possible before due to the assumption on real analytic activation functions.

At a high level, our main results in this chapter shed more light on the underlying geometrical structure of the optimization landscape of deep and wide networks. They complement the results of the previous work Choromanska et al. (2015a); Haeffele and Vidal (2017); Nguyen and Hein (2018); Liang et al. (2018) which so far purely focus on the analysis of critical points/local minima and thus often miss the overall picture of the loss landscape, and extend the previous work of Venturi et al. (2018) from one hidden layer to arbitrary deep networks with a wide range of convex losses and activation functions. We refer to Section 1.2 for more details on the previous work. A preliminary version of our paper which contains the main results of this chapter can be found at Nguyen (2019).

3.1 Introduction

It is commonly observed in deep learning that over-parameterization can be helpful for optimizing neural networks. Theoretically, several recent work Allen-Zhu et al. (2018b); Du et al. (2018a);

Zou et al. (2018) have also established convergence of gradient descent for non-linear networks under excessive over-parameterization regimes. For instance the above work require $\Omega(N^4)$ neurons (or more) per hidden layer, where N denotes the number of training samples, to guarantee that (stochastic) gradient descent converges to a global minimum with zero training error. This is an inspiring result given the hardness of the problem, but the question that which properties of the loss underpinning to these successes remains unanswered. We are interested in the following question:

Is there any underlying structure of the loss function that can "intuitively" supports for the success of local search algorithms like gradient descent under excessive over-parameterization regimes?

In this chapter, we almost settle this question by showing that every sublevel set of the loss function is connected if the network has a sufficiently wide hidden layer. Our key idea is to show that, for linearly independent training data and under a relatively mild condition on the architecture, every sublevel set of the loss is connected and unbounded. This allows us to obtain similar results for deep and wide neural nets with arbitrary data. In particular, we first show that if one of the hidden layers has more neurons than the number of training samples, then the loss has no bad local valleys in the sense that there is a continuous path from anywhere in parameter space on which the loss is non-increasing and gets arbitrarily close to the minimal value of the loss. In a special case where the first hidden layer is a wide layer with twice more neurons than the number of training samples, then we show that every sublevel set is connected, and thus there is a unique global valley. All our results hold for deep fully connected networks with standard architecture, for arbitrary convex losses and strictly monotonic and/or piecewise linear activation functions such as Leaky-ReLU.

3.2 Problem setting

We consider a class of feedforward fully connected networks. The extension of our results to convolutional neural networks as analyzed in Chapter 2 is left for future work. Our basic notations and definition of the network mostly follow from Section 1.1. For the convenience of the reader, we briefly review them below, and then introduce some other new notations.

Let L be the number of layers, d the input dimension, m the output dimension and n_k the width of layer k. Throughout this chapter we assume that the network has at least one hidden layer, that is, $L \geq 2$. Let $X = [x_1, \ldots, x_N]^T \in \mathbb{R}^{N \times d}$ be our training data. By convention we assume that $n_0 = d$ and $n_L = m$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a continuous activation function which we will specify later. Let $W_k \in \mathbb{R}^{n_{k-1} \times n_k}$ and $b_k \in \mathbb{R}^{n_k}$ be the weight matrix and bias vector of layer k. The function $f_k : \mathbb{R}^d \to \mathbb{R}^{n_k}$ which maps every input $x \in \mathbb{R}^d$ to the output at layer k is defined as

$$f_k(x) = \begin{cases} x & k = 0 \\ \sigma(W_k^T f_{k-1}(x) + b_k) & k \in [1, L-1] \\ W_L^T f_{L-1}(x) + b_L & k = L. \end{cases}$$

Let $F_k = [f_k(x_1), f_k(x_2), \ldots, f_k(x_N)]^T \in \mathbb{R}^{N \times n_k}$ then we have

$$F_k = \begin{cases} X & k = 0 \\ \sigma(F_{k-1} W_k + \mathbf{1}_N b_k^T) & k \in [1, L-1] \\ F_{L-1} W_L + \mathbf{1}_N b_L^T & k = L. \end{cases}$$

Let $\theta := (W_l, b_l)_{l=1}^L$ be the set of all parameters of the network. Let Ω_l be the parameter space of layer $l \in [1, L]$, and $\Omega = \Omega_1 \times \ldots \times \Omega_L$ the whole parameter space. Let $\Omega_l^* \subset \Omega_l$ be the subset of parameters of layer l for which the corresponding weight matrix has full rank, that is $\Omega_l^* = \{(W_l, b_l) \mid W_l \text{ has full rank}\}$. In this chapter, we write $F_k(\theta)$ to denote the network output at layer k as a function θ, but sometimes we drop the argument and just write F_k if it is clear from the

context. We also use the notations $F_k\Big((W_1,b_1),\dots,(W_L,b_L)\Big), F_k\Big((W_1,b_1),(W_l,b_l)_{l=2}^L\Big)$ to denote the same quantity. The output of the network is given as $F_L(\theta)$.

The training problem of a deep fully connected network is framed as an optimization problem, in which one minimizes the empirical loss $\Phi : \Omega \to \mathbb{R}$ defined as

$$\Phi(\theta) = \varphi(F_L(\theta)) \tag{3.1}$$

where $\varphi : \mathbb{R}^{N\times m} \to \mathbb{R}$ is any convex loss function defined on the output of the network, such as the standard square loss or cross-entropy loss as described in Section 1.1.

In the following, we let

$$p^* = \inf_{G\in\mathbb{R}^{N\times m}} \varphi(G),$$

which serves as a lower bound on Φ. Note that p^* is fully determined by the choice of $\varphi(\cdot)$ and thus independent of the training data. In this chapter, we make no assumption on p^* but for most of practical losses as mentioned above one has $p^* = 0$.

3.3 Preliminaries

This section reviews some basic mathematical concepts that are used throughout the chapter.

3.3.1 Connected sets

We first recall the standard definition of connected sets and some basic properties from topology. This is essential for proving our main results on connectedness of sublevel sets of the loss function.

Definition 3.3.1 (Connected set) *A subset $S \subseteq \mathbb{R}^d$ is called connected if for every $x, y \in S$, there exists a continuous curve $r : [0,1] \to S$ such that $r(0) = x$ and $r(1) = y$.*

The following standard result shows that the image of every connected set under a continuous mapping is again connected.

Proposition 3.3.2 *Let $f : U \to V$ be a continuous function. If $A \subseteq U$ is a connected set then the set $f(A) := \{f(x) \mid x \in A\}$ is also a connected set.*

Proof: Pick some $a, b \in f(A)$. Then there must exist some $x, y \in A$ such that $f(x) = a$ and $f(y) = b$. Since A is a connected set, it holds by Definition 3.2 that there exists a continuous curve $r : [0,1] \to A$ such that $r(0) = x, r(1) = y$. Consider the curve $f \circ r : [0,1] \to f(A)$, then it holds that $f(r(0)) = a, f(r(1)) = b$. Moreover, $f \circ r$ is continuous as both f and r are continuous. Thus it holds that $f(A)$ is a connected set by Definition 3.2. □

The following property follows immediately from the definition of connected sets.

Proposition 3.3.3 *The Minkowski sum of two connected subsets $U, V \subseteq \mathbb{R}^n$, defined as $U + V = \{u + v \mid u \in U, v \in V\}$, is a connected set.*

Proof: Let $x, y \in U+V$, then there exists $a, b \in U$ and $c, d \in V$ such that $x = a+c, y = b+d$. Since U and V are connected sets, there exist two continuous curves $p : [0,1] \to U$ and $q : [0,1] \to V$ such that $p(0) = a, p(1) = b$ and $q(0) = c, q(1) = d$. Consider the continuous curve $r(t) := p(t)+q(t)$ then it holds that $r(0) = a+c = x, r(1) = b+d = y$ and $r(t) \in U+V$ for every $t \in [0,1]$. This implies that every two elements in $U+V$ can be connected by a continuous curve and thus $U+V$ must be a connected set. □

3.3.2 Pre-Image

The hierarchical structure of a deep network allows us to write its output as the composition of a number of linear and nonlinear transformations. In the derivation of our key results (see e.g. Lemma 3.4.6) we need to take the pre-image of a composition function, which we briefly discuss next.

Definition 3.3.4 (Pre-Image) *The pre-image of a function $f : U \to V$ is the set-valued function $f^{-1} : V \to U$ defined for every $y \in V$ as $f^{-1}(y) = \{x \in U \mid f(x) = y\}$. Similarly, for every subset $A \subseteq V$, let $f^{-1}(A) = \{x \in U \mid f(x) \in A\}$.*

By definition, it holds that $f(x) \in A$ if and only if $x \in f^{-1}(A)$. We note that the above definition does not require $V = \text{range}(f) := f(U)$ but the inverse function f^{-1} is always well-defined. In particular, we have $f^{-1}(y) = \emptyset$ for every $y \notin \text{range}(f)$. Thus it holds for every $A \subseteq V$ that

$$f^{-1}(A) = f^{-1}(A \cap \text{range}(f)) \ \cup \ f^{-1}(A \setminus \text{range}(f)) = f^{-1}(A \cap \text{range}(f)).$$

The following proposition shows how to take the pre-image of a composition map.

Proposition 3.3.5 *Let $f : U \to V$ and $g : V \to Q$. Then it holds $(g \circ f)^{-1} = f^{-1} \circ g^{-1}$.*

Proof: By definition, it holds for every $A \subseteq Q$ that

$$\begin{aligned}(g \circ f)^{-1}(A) = \{x \in U \mid g(f(x)) \in A\} = \left\{x \in U \;\middle|\; f(x) \in g^{-1}(A)\right\} &= \left\{x \in U \;\middle|\; x \in f^{-1}(g^{-1}(A))\right\} \\ &= (f^{-1} \circ g^{-1})(A).\end{aligned}$$

□

Similar to above, we have for every $A \subseteq Q$ that

$$(g \circ f)^{-1}(A) = f^{-1}\Big(g^{-1}\big(A \cap \text{range}(g)\big) \cap \text{range}(f)\Big).$$

As we will see later in the derivation of our results (e.g. Lemma 3.4.6), we can avoid the above intersection steps by making certain conditions on the network so that these functions have full range.

3.3.3 Level sets and sublevel sets

The key concept studied in this chapter is level sets and sublevel sets of the loss function.

Definition 3.3.6 *For every $\alpha \in \mathbb{R}$, the α-level set of $\Phi : \Omega \to \mathbb{R}$ is the preimage $\Phi^{-1}(\alpha) = \{\theta \in \Omega \mid \Phi(\theta) = \alpha\}$, and the α-sublevel set of Φ is given as $L_\alpha = \{\theta \in \Omega \mid \Phi(\theta) \leq \alpha\}$.*

3.3.4 Local valleys and bad local valleys

Besides connectedness of sublevel sets, we also study in this chapter sufficient conditions on the architecture so that the loss surface is guaranteed to have no bad local valleys.

Definition 3.3.7 *A local valley is a nonempty connected component of some strict sublevel set $L^s_\alpha := \{\theta \mid \Phi(\theta) < \alpha\}$. A bad local valley is a local valley on which the training loss Φ cannot be made arbitrarily close to p^*.*

A trivial example of bad local valley is a small neighborhood around a sub-optimal strict local minimum. More generally, a bad local valley can be any open set of the parameter space where points on the border have strictly larger objective value than points in the interior. Intuitively, bad local valleys represent regions where gradient descent with infinitesimal step sizes cannot escape.

3.4 Main results

The organization of this section is as follows. Section 3.4.2 shows our first motivating result that if all the training samples are linearly independent and the network has decreasing width from the first hidden layer till the output layer then every sublevel set of the loss must be connected and unbounded. The following sections will extend this result to deep and wide networks with "arbitrary" training data. In particular, Section 3.4.3 shows that if one of the hidden layers has more neurons than the number of training samples, then the network has no bad local valleys. In Section 3.4.4 we further shows that if the first hidden layer has at least twice more neurons than the number of training samples then every sublevel set of the loss is connected and unbounded. Finally, Section 3.4.5 shows an extension of our previous results to the case of ReLU activation function.

3.4.1 Assumptions

This section presents all the assumptions on the activation function and training data. We will refer to each individual assumption accordingly in the next sections where we present our main results.

Assumption 3.4.1 *σ is strictly monotonic and $\sigma(\mathbb{R}) = \mathbb{R}$.*

Assumption 3.4.1 implies that σ has a continuous inverse $\sigma^{-1} : \mathbb{R} \to \mathbb{R}$. One can see that this is satisfied by the Leaky-ReLU $\sigma(x) = \max(\alpha x, x)$ where $\alpha \in (0, 1)$ but not by ReLU $\sigma(x) = \max(0, x)$.

The following assumption states that the activation function σ cannot be written as a weighted sum of its shifted variants.

Assumption 3.4.2 *There do not exist non-zero coefficients $(\lambda_i, a_i)_{i=1}^p$ with $a_i \neq a_j \, \forall \, i \neq j$ such that $\sigma(x) = \sum_{i=1}^p \lambda_i \sigma(x - a_i)$ for every $x \in \mathbb{R}$.*

It turns out that Assumption 3.4.2 is satisfied for a large class of non-smooth activation functions, including ReLU and Leaky-ReLU, as shown by the following lemma. This is also one of the key advantages of our results in this chapter compared to the previous Chapter 2 where we can only deal with smooth activation functions such as sigmoid and softplus.

Lemma 3.4.3 *Assumption 3.4.2 is satisfied for any continuous piecewise linear activation function with at least two pieces such as ReLU and Leaky-ReLU, and for the exponential linear unit $\sigma(x) = \begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$ where $\alpha > 0$.*

Proof: A function $\sigma : \mathbb{R} \to \mathbb{R}$ is continuous piecewise linear if it can be represented as

$$\sigma(x) = a_i x + b_i, \quad \forall \, x \in (x_{i-1}, x_i), \, \forall \, i \in [1, n+1].$$

for some $x_0 = -\infty < x_1 < \ldots < x_n < x_{n+1} = \infty$ and $(a_i, b_i)_{i=1}^{n+1}$. Since σ has at least two pieces we have $n \geq 1$. We can assume that all the linear pieces agree at their intersection and there are no consecutive pieces with the same slope: $a_i \neq a_{i+1}$ for every $i \in [1, n]$. Suppose by contradiction that σ does not satisfy Assumption 3.4.2, then there are non-zero coefficients $(\lambda_i, y_i)_{i=1}^m$ with $y_i \neq y_j (i \neq j)$ such that $\sigma(x) = \sum_{i=1}^m \lambda_i \sigma(x - y_i)$ for every $x \in \mathbb{R}$. We assume w.l.o.g. that $y_1 < \ldots < y_m$.

Case 1: $y_1 > 0$. For every $x \in (-\infty, x_1)$ we have $\sigma(x) = a_1 x + b_1 = \sum_{i=1}^m \lambda_i (a_1(x - y_i) + b_1)$ and thus by comparing the coefficients on both sides we obtain $\sum_{i=1}^m \lambda_i a_1 = a_1$. Moreover, for every $x \in \big(x_1, \min(x_1 + y_1, x_2)\big)$ it holds $\sigma(x) = a_2 x + b_2 = \sum_{i=1}^m \lambda_i (a_1(x - y_i) + b_1)$ and so $\sum_{i=1}^m \lambda_i a_1 = a_2$. Thus $a_1 = a_2$, which is a contradiction.

Case 2: $y_1 < 0$. By definition, for $x \in (-\infty, x_1+y_1)$ we have $\sigma(x) = a_1x+b_1 = \sum_{i=1}^m \lambda_i(a_1(x-y_i)+b_1)$ and thus by comparing the coefficients on both sides we obtain $\sum_{i=1}^m \lambda_i a_1 = a_1$. Moreover, for every $x \in \big(x_1+y_1, \min(x_1+y_2, x_1, x_2+y_1)\big)$ we have

$$\sigma(x) = a_1x + b_1 = \lambda_1(a_2(x-y_1)+b_2) + \sum_{i=2}^m \lambda_i(a_1(x-y_i)+b_1)$$

and thus by comparing the coefficients we have $\lambda_1 a_2 + \sum_{i=2}^m \lambda_i a_1 = a_1$. This combined with the first result above leads to $\lambda_1 a_1 = \lambda_1 a_2$, and thus $a_1 = a_2$ (since $\lambda_1 \neq 0$) which is a contradiction.

One can prove similarly for ELU Clevert et al. (2016)

$$\sigma(x) = \begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases} \text{ where } \alpha > 0.$$

Suppose by contradiction that there exist non-zero coefficients $(\lambda_i, y_i)_{i=1}^m$ with $y_i \neq y_j (i \neq j)$ such that $\sigma(x) = \sum_{i=1}^m \lambda_i \sigma(x-y_i)$, and assume w.l.o.g. that $y_1 < \ldots < y_m$. If $y_m > 0$ then for every $x \in (\max(0, y_{m-1}), y_m)$ it holds

$$\sigma(x) = x = \lambda_m \alpha(e^{x-y_m} - 1) + \sum_{i=1}^{m-1} \lambda_i(x-y_i) \quad \Longrightarrow \quad e^x = \frac{xe^{y_m} - \sum_{i=1}^{m-1} \lambda_i(x-y_i)e^{y_m}}{\lambda_m \alpha} + e^{y_m}$$

which is a contradiction since e^x cannot be identical to any affine function on any open interval. Thus it must hold that $y_m < 0$. For every $x \in (y_m, 0)$ we have

$$\sigma(x) = \alpha(e^x - 1) = \sum_{i=1}^m \lambda_i(x-y_i) \quad \Longrightarrow \quad e^x = \frac{1}{\alpha}\sum_{i=1}^m \lambda_i(x-y_i) + 1$$

which is a contradiction for the same reason above. □

In the remainder of this chapter, if not stated otherwise we will always assume the following universal condition on the training data.

Assumption 3.4.4 *All the training samples are distinct, that is, $x_i \neq x_j$ for every $i \neq j$. Equivalently, the training data matrix $X \in \mathbb{R}^{N \times d}$ has distinct rows.*

3.4.2 Linearly independent data leads to connected sublevel sets

This section presents our key results for linearly independent data, which lays the foundation for our further results in the next sections where we analyze deep over-parameterized neural nets with arbitrary data. In this section, we assume that the widths of all the hidden layers are decreasing, i.e. $n_1 > \ldots > n_L$. Note that it is still possible to have $n_1 \geq d$ or $n_1 < d$. It turns out that this condition is quite natural as for many practical architectures (see Table 2.1) the first hidden layer often has the most number of neurons, and then it starts decreasing towards the output layer, which is helpful for the network to learn more compact representations at higher layers. For convenience, we introduce the following property of $\theta = (W_l, b_l)_{l=1}^L \in \Omega$ and will refer to it later during our presentation.

Property 3.4.5 *W_l has full rank for every $l \in [2, L]$.*

The derivation of our key result relies on several helpful lemmas, which we present next.

First, we show that any changes in the output space of every layer $k \in [2, L]$ or any changes in the weight matrices $(W_l)_{l=2}^L$ can also be reflected onto the first layer, as long as these matrices have full

rank. As shown below, this holds when the training data X has full row rank, and the activation function is invertible by Assumption 3.4.1, because in which case the network is able to realize arbitrary outputs at every hidden layer just by adapting the value of (W_1, b_1). The following lemma shows that these changes can be done in a continuous manner.

Lemma 3.4.6 *Let Assumption 3.4.1 hold, rank$(X) = N$ and $n_1 > \ldots > n_L$. Given some layer $k \in [2, L]$. Then there is a continuous map $h : \Omega_2^* \times \ldots \times \Omega_k^* \times \mathbb{R}^{N \times n_k} \to \Omega_1$ which satisfy that:*

1. *For every $\Big((W_2, b_2), \ldots, (W_k, b_k), A\Big) \in \Omega_2^* \times \ldots \times \Omega_k^* \times \mathbb{R}^{N \times n_k}$ it holds that*

$$F_k\Big(h\Big((W_l, b_l)_{l=2}^k, A\Big), (W_l, b_l)_{l=2}^k\Big) = A.$$

2. *For every $\theta = (W_l^*, b_l^*)_{l=1}^L$ where all the matrices $(W_l^*)_{l=2}^k$ have full rank, there is a continuous curve from θ to $\Big(h\Big((W_l^*, b_l^*)_{l=2}^k, F_k(\theta)\Big), (W_l^*, b_l^*)_{l=2}^L\Big)$ on which the loss Φ is constant.*

Proof: We recall that Ω_l is the parameter space of layer l and $\Omega_l^* = \{(W_l, b_l) \mid W_l \text{ has full rank}\} \subset \Omega_l$. For every $\Big((W_2, b_2), \ldots, (W_k, b_k), A\Big) \in \Omega_2^* \times \ldots \times \Omega_k^* \times \mathbb{R}^{N \times n_k}$, let us define the value of h as

$$h\Big((W_l, b_l)_{l=2}^k, A\Big) = (W_1, b_1),$$

where (W_1, b_1) is given by the following recursive formula

$$\begin{cases} \begin{bmatrix} W_1 \\ b_1^T \end{bmatrix} = [X, \mathbf{1}_N]^\dagger \sigma^{-1}(B_1), \\ B_l = \Big(\sigma^{-1}(B_{l+1}) - \mathbf{1}_N b_{l+1}^T\Big) W_{l+1}^\dagger, \ \forall\, l \in [1, k-2], \\ B_{k-1} = \begin{cases} (A - \mathbf{1}_N b_L^T)\, W_L^\dagger & k = L \\ \Big(\sigma^{-1}(A) - \mathbf{1}_N b_k^T\Big) W_k^\dagger & k \in [2, L-1] \end{cases} \end{cases}.$$

By our assumption $n_1 > \ldots > n_L$, it follows from the domain of h that all the matrices $(W_l)_{l=2}^k$ have full column rank, and so they have a left inverse. Similarly, $[X, \mathbf{1}_N]$ has full row rank due to our assumption that *rank*$(X) = N$, and so it has a right inverse. Moreover σ has a continuous inverse by Assumption 3.4.1. Thus h is a continuous map as it is a composition of continuous functions. In the following, we prove that h satisfies the two statements of the lemma.

1. Let $\Big((W_2, b_2), \ldots, (W_k, b_k), A\Big) \in \Omega_2^* \times \ldots \times \Omega_k^* \times \mathbb{R}^{N \times n_k}$. Since all the matrices $(W_l)_{l=2}^k$ have full column rank and $[X, \mathbf{1}_N]$ has full row rank, it holds that $W_l^\dagger W_l = \mathbb{I}$ and $[X, \mathbf{1}_N][X, \mathbf{1}_N]^\dagger = \mathbb{I}$ and thus we easily obtain from the above definition of h that

$$\begin{cases} B_1 = \sigma\Big([X, \mathbf{1}_N] \begin{bmatrix} W_1 \\ b_1^T \end{bmatrix}\Big), \\ B_{l+1} = \sigma(B_l W_{l+1} + \mathbf{1}_N b_{l+1}^T), \ \forall\, l \in [1, k-2], \\ A = \begin{cases} B_{L-1} W_L + \mathbf{1}_N b_L^T & k = L, \\ \sigma(B_{k-1} W_k + \mathbf{1}_N b_k^T) & k \in [2, L-1]. \end{cases} \end{cases}$$

   One can easily check that the above formula of A is exactly the definition of F_k from (2.4) and thus it holds $F_k\Big(h\Big((W_l, b_l)_{l=2}^k, A\Big), (W_l, b_l)_{l=2}^k\Big) = A$ for every $\Big((W_2, b_2), \ldots, (W_k, b_k), A\Big) \in \Omega_2^* \times \ldots \times \Omega_k^* \times \mathbb{R}^{N \times n_k}$.

2. Let $G_l : \mathbb{R}^{N\times n_{l-1}} \to \mathbb{R}^{N\times n_l}$ be defined as

$$G_l(Z) = \begin{cases} ZW_L^* + \mathbf{1}_N(b_L^*)^T & l = L \\ \sigma\Big(ZW_l^* + \mathbf{1}_N(b_l^*)^T\Big) & l \in [2, L-1]. \end{cases}$$

For convenience, let us group the parameters of the first layer into a matrix, say $U = [W_1^T, b_1]^T \in \mathbb{R}^{(d+1)\times n_1}$. Similarly, let $U^* = [(W_1^*)^T, b_1^*]^T \in \mathbb{R}^{(d+1)\times n_1}$. Let $f : \mathbb{R}^{(d+1)\times n_1} \to \mathbb{R}^{N\times n_k}$ be a function of (W_1, b_1) defined as

$$\begin{aligned} &f(U) = G_k \circ G_{k-1} \ldots G_2 \circ G_1(U), \text{ where} \\ &G_1(U) = \sigma([X, \mathbf{1}_N]U),\ U = [W_1^T, b_1]^T. \end{aligned}$$

We note that this definition of f is exactly F_k from (2.4), but here we want to exploit the fact that f is a function of (W_1, b_1) as all other parameters are fixed to the corresponding values of θ. Let $A = F_k(\theta)$. By definition we have $f(U^*) = A$ and thus $U^* \in f^{-1}(A)$. Let us denote

$$(W_1^h, b_1^h) = h\Big(W_l^*, b_l^*)_{l=2}^k, A\Big), \quad U^h = [(W_1^h)^T, b_1^h]^T.$$

By applying the first statement of the lemma to $\Big((W_2^*, b_2^*), \ldots, (W_k^*, b_k^*), A\Big)$ we have

$$A = F_k\Big((W_1^h, b_1^h), (W_l^*, b_l^*)_{l=2}^k\Big) = f(U^h)$$

which implies $U^h \in f^{-1}(A)$. So far both U^* and U^h belong to $f^{-1}(A)$. The idea now is that if one can show that $f^{-1}(A)$ is a connected set then there would exist a connected path between U^* and U^h (and thus a path between (W_1^*, b_1^*) and (W_1^h, b_1^h)) on which the output at layer k is identical to A and hence the loss is invariant, which concludes the proof.

In the following, we show that $f^{-1}(A)$ is indeed connected. First, one observes that $\text{range}(G_l) = \mathbb{R}^{N\times n_l}$ for every $l \in [2, k]$ since all the matrices $(W_l^*)_{l=2}^k$ have full column rank and $\sigma(\mathbb{R}) = \mathbb{R}$ by Assumption 3.4.1. Similarly, it follows from our assumption $rank(X) = N$ that $\text{range}(G_1) = \mathbb{R}^{N\times n_1}$. By applying Proposition 3.3.5 to the composition map f, we obtain

$$f^{-1}(A) = G_1^{-1} \circ G_2^{-1} \circ \ldots \circ G_k^{-1}(A).$$

where all the maps $G_1, \ldots, G_k$ have full range as shown above. Now it holds that

$$G_k^{-1}(A) = \begin{cases} (A - \mathbf{1}_N b_L^T)(W_L^*)^\dagger + \{B \mid BW_L^* = 0\} & k = L \\ \Big(\sigma^{-1}(A) - \mathbf{1}_N b_k^*\Big)(W_k^*)^\dagger + \{B \mid BW_k^* = 0\} & \text{else} \end{cases}$$

which is a connected set in either case due to the following reasons: 1) the kernel of any matrix is connected, 2) the Minkowski-sum of two connected sets is connected by Proposition 3.3.3, and 3) the image of a connected set under a continuous map is connected by Proposition 3.3.2. Once $G_k^{-1}(A)$ is connected, we can repeat exactly the same argument as above for $k-1, \ldots, 2$ to conclude that $V := G_2^{-1} \circ \ldots \circ G_k^{-1}(A)$ is a connected set. Lastly, we have

$$G_1^{-1}(V) = [X, \mathbf{1}_N]^\dagger \sigma^{-1}(V) + \{B \mid [X, \mathbf{1}_N]B = 0\}$$

which is also connected by the same arguments above. Thus $f^{-1}(A)$ is a connected set.

Overall, we have shown in this proof that the set of (W_1, b_1) which realizes the same output at layer k (given the parameters of other layers in between are fixed) is a connected set. Since both (W_1^*, b_1^*) and $h\Big((W_l^*, b_l^*)_{l=2}^k, F_k(\theta)\Big)$ belong to this solution set, there must exist a continuous path between them on which the loss Φ is constant.

□

Using the previous lemma, we can show that from any point in parameter space where the weight matrices $(W_l)_{l=2}^L$ are potentially low rank, we can find a continuous path which leads to some other point where all the corresponding matrices obtain full rank, and most importantly, the loss is constant along the path. This is done by the next lemma.

Lemma 3.4.7 *Let Assumption 3.4.1 hold, rank$(X) = N$ and $n_1 > \ldots > n_L$. Let $\theta = (W_l, b_l)_{l=1}^L$ be any point in parameter space. Then there is a continuous curve which starts from θ and ends at some $\theta' = (W_l', b_l')_{l=1}^L$ so that θ' satisfies Property 3.4.5 and the loss Φ is constant on the curve.*

Proof: The idea is to make each of the weight matrices $W_2, \ldots, W_L$ full rank, one at a time (i.e. keeping other matrices fixed, except W_1). At each step, we find a continuous path from the current starting point to some other point where the rank condition is fulfilled while keeping the loss constant on the path. We then restart our starting point to the end point of the path and proceed to the next weight matrix. This is repeated until all the matrices $(W_l)_{l=2}^L$ have full rank, which satisfies 3.4.5.

Step 1: Make W_2 full rank. If W_2 has full rank then we proceed to W_3. Otherwise, let *rank*$(W_2) = r < n_2 < n_1$. Let $\mathcal{I} \subset \{1, \ldots, n_1\}, |\mathcal{I}| = r$ denote the set of indices of linearly independent rows of W_2 so that *rank*$(W_2(\mathcal{I}, :)) = r$. Let $\bar{\mathcal{I}}$ denote the remaining rows of W_2. Let $E \in \mathbb{R}^{(n_1 - r) \times r}$ be a matrix such that $W_2(\bar{\mathcal{I}}, :) = EW_2(\mathcal{I}, :)$. Let $P \in \mathbb{R}^{n_1 \times n_1}$ be a permutation matrix which permutes the rows of W_2 according to $\mathcal{I}$ so that we can write

$$PW_2 = \begin{bmatrix} W_2(\mathcal{I}, :) \\ W_2(\bar{\mathcal{I}}, :) \end{bmatrix}.$$

We recall that $F_1(\theta)$ is the output of the network at the first layer, evaluated at θ. Below we drop θ and just write F_1 as it is clear from the context. By construction of P, we have

$$F_1 P^T = [F_1(:, \mathcal{I}), F_1(:, \bar{\mathcal{I}})].$$

The first step is to turn W_1 into a canonical form. In particular, the set of all possible solutions of W_1 which realizes the same the output F_1 at the first hidden layer is characterized by $X^\dagger\big(\sigma^{-1}(F_1) - \mathbf{1}_N b_1^T\big) + \ker(X)$ where we denote, by abuse of notation, $\ker(X) = \big\{A \in \mathbb{R}^{d \times n_1} \;\big|\; XA = 0\big\}$. This solution set is connected because $\ker(X)$ is a connected set and the Minkowski-sum of two connected sets is known to be connected, and so there exists a continuous path between every two solutions in this set on which the output F_1 is invariant. Obviously the current W_1 and $X^\dagger(\sigma^{-1}(F_1) - \mathbf{1}_N b_1^T)$ are elements of this set, thus they must be connected by a continuous path on which the loss is invariant. So we can assume now that $W_1 = X^\dagger(\sigma^{-1}(F_1) - \mathbf{1}_N b_1^T)$. Next, we consider the curve:

$$W_1(\lambda) = X^\dagger\Big(\sigma^{-1}(A(\lambda)) - \mathbf{1}_N b_1^T\Big), \text{ where } A(\lambda) = [F_1(:, \mathcal{I}) + \lambda F_1(:, \bar{\mathcal{I}})E, (1-\lambda)F_1(:, \bar{\mathcal{I}})]\, P.$$

This curves starts at θ since $W_1(0) = W_1$, and it is continuous as σ has a continuous inverse by Assumption 3.4.1. Using $XX^\dagger = \mathbb{I}$, one can compute the pre-activation output (without bias term) at the second layer as

$$\sigma\big(XW_1(\lambda) + \mathbf{1}_N b_1^T\big)\, W_2 = A(\lambda)\, W_2 = F_1 W_2,$$

which implies that the loss is invariant on this curve, and so we can take its end point $W_1(1)$ as a new starting point:

$$W_1 = X^\dagger\Big(\sigma^{-1}(A) - \mathbf{1}_N b_1^T\Big), \text{ where } A = [F_1(:, \mathcal{I}) + F_1(:, \bar{\mathcal{I}})E,\, \mathbf{0}]\, P.$$

Now, the output at second layer above, given by AW_2, is independent of $W_2(\bar{\mathcal{I}},:)$ because it is canceled by the zero component in A. Thus one can easily change $W_2(\bar{\mathcal{I}},:)$ so that W_2 has full rank while still keeping the loss invariant.

Step 2: Using induction to make $W_3,\dots,W_L$ full rank. Let $\theta = (W_l,b_l)_{l=2}^L$ be our current point. Suppose that all the matrices $(W_l)_{l=2}^k$ already have full rank for some $k \geq 2$ then we show below how to make W_{k+1} full rank. We write F_k to denote $F_k(\theta)$. By the second statement of Lemma 3.4.6, we can follow a continuous path (with invariant loss) to drive θ to the following point:

$$\theta := \Big(h\Big((W_l,b_l)_{l=2}^k, F_k\Big), (W_l,b_l)_{l=2}^L\Big) \tag{3.2}$$

where $h : \Omega_2^* \times \ldots \times \Omega_k^* \times \mathbb{R}^{N\times n_k}$ is the continuous map from Lemma 3.4.6 which satisfies for every $A \in \mathbb{R}^{N\times n_k}$,

$$F_k\Big(h\big((W_l,b_l)_{l=2}^k, A\big), (W_l,b_l)_{l=2}^k\Big) = A. \tag{3.3}$$

Now, if W_{k+1} already has full rank then we are done, otherwise we follow the similar steps as before. Indeed, let $r = rank(W_{k+1}) < n_{k+1} < n_k$ and $\mathcal{I} \subset \{1,\dots,n_k\}, |\mathcal{I}| = r$ the set of indicies of r linearly independent rows of W_{k+1}. Then there is a permutation matrix $P \in \mathbb{R}^{n_k\times n_k}$ and some matrix $E \in \mathbb{R}^{(n_k-r)\times r}$ so that

$$PW_{k+1} = \begin{bmatrix} W_{k+1}(\mathcal{I},:) \\ W_{k+1}(\bar{\mathcal{I}},:) \end{bmatrix}, W_{k+1}(\bar{\mathcal{I}},:) = EW_{k+1}(\mathcal{I},:). \tag{3.4}$$

Moreover it holds

$$F_kP^T = [F_k(:,\mathcal{I}), F_k(:,\bar{\mathcal{I}})]. \tag{3.5}$$

Consider the following curve $c : [0,1] \to \Omega$ which continuously update (W_1,b_1) while keeping other layers fixed:

$$\begin{aligned} c(\lambda) &= \Big(h\Big((W_l,b_l)_{l=2}^k, A(\lambda)\Big), (W_2,b_2),\dots,(W_L,b_L)\Big), \\ &\text{where } A(\lambda) = [F_k(:,\mathcal{I}) + \lambda F_k(:,\bar{\mathcal{I}})E, (1-\lambda)F_k(:,\bar{\mathcal{I}})]\, P. \end{aligned}$$

Since h is continuous, c is also continuous. Moreover, it follows from (3.5) and (3.2) that $c(0) = \theta$. The pre-activation output (without bias term) at layer $k+1$ for every point on this curve is given by

$$F_k(c(\lambda))\, W_{k+1} = A(\lambda)W_{k+1} = F_kW_{k+1},\ \ \forall\, \lambda \in [0,1],$$

where the first equality follows from (3.3) and the second follows from (3.4) and (3.5). As the loss is invariant on this curve, we can take its end point $c(1)$ as a new starting point:

$$\theta := \Big(h\Big((W_l,b_l)_{l=2}^k, A\Big), (W_2,b_2),\dots,(W_L,b_L)\Big) \text{ where } A = [F_k(:,\mathcal{I}) + F_k(:,\bar{\mathcal{I}})E,\, \mathbf{0}]\, P.$$

At this point, the output at layer $k+1$ as mentioned above is given by AW_{k+1}, which is independent of $W_{k+1}(\bar{\mathcal{I}},:)$ since it is canceled out by the zero component in A, and thus one can easily change the submatrix $W_{k+1}(\bar{\mathcal{I}},:)$ so that W_{k+1} has full rank while leaving the loss invariant.

Overall, by induction we can make all the weight matrices $W_2,\dots,W_L$ full rank by following several continuous paths on which the loss is constant, which finishes the proof. □

The following result is due to Evard and Jafari (1994), which later allows us to build connected paths between solutions in the parameter space where all the weight matrices $(W_l)_{l=2}^L$ have full rank.

Proposition 3.4.8 *The set of full rank matrices $A \in \mathbb{R}^{m\times n}$ is connected for $m \neq n$.*

We are now ready to state our main result in this section.

Theorem 3.4.9 *Let Assumption 3.4.1 hold, rank$(X) = N$ and $n_1 > \ldots > n_L$. Then it holds:*

1. *Every sublevel set of Φ is connected. Moreover, Φ can attain any value arbitrarily close to p^*.*
2. *Every non-empty connected component of every level set of Φ is unbounded.*

Proof:

1. Let L_α be some sublevel set of Φ. Let $\theta = (W_l, b_l)_{l=1}^L$ and $\theta' = (W'_l, b'_l)_{l=1}^L$ be arbitrary points in L_α. Let $F_L = F_L(\theta)$ and $F'_L = F_L(\theta')$. These two quantities are computed in the beginning and will never change during this proof. But when we write $F_L(\theta'')$ for some θ'' we mean the network output evaluated at θ''. The main idea is to construct two different continuous paths which simultaneously start from θ and θ' and are entirely contained in L_α (this is done by making the loss on each individual path non-increasing), and then show that they meet at a common point in L_α, which then implies that L_α is a connected set.

   First of all, we can assume that both θ and θ' satisfy Property 3.4.5, because otherwise by Lemma 3.4.7 one can follow a continuous path from each point to arrive at some other point where this property holds and the loss on each path is invariant, meaning that we still stay inside L_α. As θ and θ' satisfy Property 3.4.5, all the weight matrices $(W_l)_{l=2}^L$ and $(W'_l)_{l=2}^L$ have full rank, and thus by applying the second statement of Lemma 3.4.6 with $k = L$ and using the similar argument above, we can simultaneously drive θ and θ' to the following points,

$$\begin{aligned}\theta &= \Big(h\Big((W_l, b_l)_{l=2}^L, F_L\Big), (W_2, b_2), \ldots, (W_L, b_L)\Big),\\ \theta' &= \Big(h\Big((W'_l, b'_l)_{l=2}^L, F'_L\Big), (W'_2, b'_2), \ldots, (W'_L, b'_L)\Big)\end{aligned} \tag{3.6}$$

   where $h : \Omega_2^* \times \ldots \times \Omega_L^* \times \mathbb{R}^{N\times m} \to \Omega_1$ is the continuous map from Lemma 3.4.6 which satisfies

$$F_L\Big(h\Big((\hat{W}_l, \hat{b}_l)_{l=2}^L, A\Big), (\hat{W}_l, \hat{b}_l)_{l=2}^L\Big) = A \tag{3.7}$$

   for every $\Big((\hat{W}_l, \hat{b}_l), \ldots, (\hat{W}_L, \hat{b}_L), A\Big) \in \Omega_2^* \times \ldots \times \Omega_L^* \times \mathbb{R}^{N\times n_k}$.

   Next, we construct a continuous path starting from θ on which the loss is constant and it holds at the end point of the path that all parameters from layer 2 till layer L are equal to the corresponding parameters of θ'. Indeed, by applying Proposition 3.4.8 to the pairs of full rank matrices (W_l, W'_l) for every $l \in [2, L]$, we obtain continuous curves $W_2(\lambda), \ldots, W_L(\lambda)$ so that $W_l(0) = W_l, W_l(1) = W'_l$ and $W_l(\lambda)$ has full rank for every $\lambda \in [0, 1]$. For every $l \in [2, L]$, let $c_l : [0, 1] \to \Omega_l^*$ be the curve of layer l defined as

$$c_l(\lambda) = \Big(W_l(\lambda), (1-\lambda)b_l + \lambda b'_l\Big).$$

   We consider the curve $c : [0, 1] \to \Omega$ given by

$$c(\lambda) = \Big(h\Big((c_l(\lambda))_{l=2}^L, F_L\Big), c_2(\lambda), \ldots, c_L(\lambda)\Big).$$

   Then one can easily check that $c(0) = \theta$ and c is continuous as all the functions $h, c_2, \ldots, c_l$ are continuous. Moreover, we have $\Big(c_2(\lambda), \ldots, c_L(\lambda)\Big) \in \Omega_2^* \times \ldots \times \Omega_L^*$ and thus it follows from (3.7) that $F_L(c(\lambda)) = F_L$ for every $\lambda \in [0, 1]$, which leaves the loss invariant on c.

   Since the curve c above starts at θ and has constant loss, we can reset θ to the end point of this curve, by setting $\theta = c(1)$, while keeping θ' from (3.6), which together give us

$$\begin{aligned}\theta &= \Big(h\Big((W'_l, b'_l)_{l=2}^L, F_L\Big), (W'_2, b'_2), \ldots, (W'_L, b'_L)\Big),\\ \theta' &= \Big(h\Big((W'_l, b'_l)_{l=2}^L, F'_L\Big), (W'_2, b'_2), \ldots, (W'_L, b'_L)\Big).\end{aligned}$$

Now the parameters of θ and θ' already coincide at all the layers, except at the first layer. In the following, we will construct two continuous paths in L_α, say $c_1(\cdot)$ and $c_2(\cdot)$, which starts from θ and θ' respectively, and then show that they meet at a common point in L_α. Let $\hat{Y} \in \mathbb{R}^{N\times m}$ be any matrix so that

$$\varphi(\hat{Y}) \leq \min(\Phi(\theta), \Phi(\theta')). \tag{3.8}$$

Consider the curve $c_1 : [0,1] \to \Omega$ defined as

$$c_1(\lambda) = \Big(h\Big((W_l', b_l')_{l=2}^L, (1-\lambda)F_L + \lambda\hat{Y}\Big), (W_l', b_l')_{l=2}^L\Big).$$

Note that c_1 is continuous as h is continuous, and it holds:

$$c_1(0) = \theta, \quad c_1(1) = \Big(h\Big((W_l', b_l')_{l=2}^L, \hat{Y}\Big), (W_l', b_l')_{l=2}^L\Big).$$

It follows from the definition of Φ, $c_1(\lambda)$ and (3.7) that

$$\Phi(c_1(\lambda)) = \varphi(F_L(c_1(\lambda))) = \varphi((1-\lambda)F_L + \lambda\hat{Y})$$

and thus by convexity of φ,

$$\Phi(c_1(\lambda)) \leq (1-\lambda)\varphi(F_L) + \lambda\varphi(\hat{Y}) \leq (1-\lambda)\Phi(\theta) + \lambda\Phi(\theta) = \Phi(\theta),$$

which implies that $c_1[0,1]$ is entirely contained in L_α. Similarly, we can also construct a curve $c_2(\cdot)$ inside L_α which starts at θ' and satisfies

$$c_2(0) = \theta', \; c_2(1) = \Big(h\Big((W_l', b_l')_{l=2}^L, \hat{Y}\Big), (W_l', b_l')_{l=2}^L\Big).$$

Till now, we have constructed the curves c_1 and c_2 which start at θ and θ' respectively and meet at the same point $c_1(1) = c_2(1)$.

To summary, we have shown that starting from any two points in L_α we can find two continuous curves so that the loss is non-increasing on each curve, and these curves meet at a common point in L_α, and so L_α has to be connected. Moreover, the point where they meet achieves the loss value $\Phi(c_1(1)) = \varphi(\hat{Y})$. From (3.8), $\varphi(\hat{Y})$ can be chosen arbitrarily small, and thus we conclude that Φ can attain any value arbitrarily close to p^*.

2. Let C be a non-empty connected component of some level set, i.e. $C \subseteq \Phi^{-1}(\alpha)$ for some $\alpha \in \mathbb{R}$. Let $\theta = (W_l, b_l)_{l=1}^L \in C$. Similar as above, we first use Lemma 3.4.7 to find a continuous path from θ to some other point where W_2 attains full rank, and the loss is invariant on the path. From that point, we apply Lemma 3.4.6 with $k = 2$ to obtain another continuous path (with constant loss) which leads us to $\theta' := \Big(h\Big((W_2, b_2), F_2(\theta)\Big), (W_2, b_2), \ldots, (W_L, b_L)\Big)$ where $h : \Omega_2^* \to \Omega_1$ is a continuous map satisfying that

$$F_2\Big(h\Big((\hat{W}_2, \hat{b}_2), A\Big), (\hat{W}_l, \hat{b}_l)_{l=2}^L\Big) = A,$$

for every point $(\hat{W}_l, \hat{b}_l)_{l=1}^L$ such that $\hat{W}_2$ has full rank, and every $A \in \mathbb{R}^{N\times n_2}$. Note that $\theta' \in C$ as the loss is constant on the above paths. Consider the following continuous curve

$$c(\lambda) = \Big(h\Big((\lambda W_2, b_2), F_2(\theta)\Big), (\lambda W_2, b_2), \ldots, (W_L, b_L)\Big)$$

for every $\lambda \geq 1$. This curve starts at θ' since $c(1) = \theta'$. We have $F_2(c(\lambda)) = F_2(\theta)$ for every $\lambda \geq 1$ and thus the loss is constant on this curve, meaning that the entire curve belongs to C. Lastly, the curve $c[1,\infty)$ is unbounded as λ goes to infinity, and thus C has to be unbounded.

□

By the decomposition of sublevel set, $\Phi^{-1}((-\infty,\alpha]) = \Phi^{-1}(\alpha) \cup \Phi^{-1}((-\infty,\alpha))$, it follows that if Φ has unbounded level sets then its sublevel sets must also be unbounded. We note that the reverse is not true, e.g. the standard Gaussian distribution function has unbounded sublevel sets but its level sets are bounded. Given that, the statements of Theorem 3.4.9 together imply that every sublevel set of Φ must be an unbounded and connected set. While connected sublevel sets of Φ implies that Φ has a well-behaved loss surface with no bad local valleys, its unbounded level sets could further imply that all the valleys have to be unbounded, regardless of whether they contain a local/global minimum or not. Clearly this also implies that Φ has no strict local minima or local maxima.

3.4.3 Large width of one of hidden layers leads to no bad local valleys

In the previous section, we show that if the training samples are linearly independent then every sublevel set of the loss is connected and unbounded. In this section, we show the first application of this result in proving absence of bad local valleys on the loss landscape of deep and wide neural nets with "arbitrary" training data. Same as before, we first prove a few intermediate results before presenting our main theorem.

In the following, we frequently encounter situations where we are given the product of two matrices $F \in \mathbb{R}^{N\times n}$ and $W \in \mathbb{R}^{n\times p}$ and we want to modify some columns of F or rows of W without changing the value of the product. The following lemma shows that this is always possible when the matrix F does not have full column rank, that is, $rank(F) < n$.

Lemma 3.4.10 *Let $(F, W, \mathcal{I})$ be such that $F \in \mathbb{R}^{N\times n}, W \in \mathbb{R}^{n\times p}, rank(F) < n$ and $\mathcal{I} \subset \{1,\dots,n\}$ be a subset of columns of F so that $rank(F(:,\mathcal{I})) = rank(F)$ and $\bar{\mathcal{I}}$ the remaining columns. Then there exists a continuous curve $c : [0,1] \to \mathbb{R}^{n\times p}$ which satisfies the following:*

1. *$c(0) = W$ and $Fc(\lambda) = FW$, $\forall\, \lambda \in [0,1]$.*
2. *The product $Fc(1)$ is independent of $F(:,\bar{\mathcal{I}})$.*

Proof: Let $r = rank(F) < n$. Since $\mathcal{I}$ contains r linearly independent columns of F, the remaining columns must lie on their span. In other words, there exists $E \in \mathbb{R}^{r\times(n-r)}$ so that $F(:,\bar{\mathcal{I}}) = F(:,\mathcal{I})\,E$. Let $P \in \mathbb{R}^{n\times n}$ be a permutation matrix which permutes the columns of F according to $\mathcal{I}$ so that we can write $F = [F(:,\mathcal{I}), F(:,\bar{\mathcal{I}})]\,P$. Consider the continuous curve $c : [0,1] \to \mathbb{R}^{n\times p}$ defined as

$$c(\lambda) = P^T \begin{bmatrix} W(\mathcal{I},:) + \lambda E\, W(\bar{\mathcal{I}},:) \\ (1-\lambda) W(\bar{\mathcal{I}},:) \end{bmatrix}, \forall\, \lambda \in [0,1].$$

It holds $c(0) = P^T \begin{bmatrix} W(\mathcal{I},:) \\ W(\bar{\mathcal{I}},:) \end{bmatrix} = W$. For every $\lambda \in [0,1]$:

$$Fc(\lambda) = [F(:,\mathcal{I}), F(:,\bar{\mathcal{I}})]\, PP^T \begin{bmatrix} W(\mathcal{I},:) + \lambda E\, W(\bar{\mathcal{I}},:) \\ (1-\lambda) W(\bar{\mathcal{I}},:) \end{bmatrix} = F(:,\mathcal{I})W(\mathcal{I},:) + F(:,\bar{\mathcal{I}})W(\bar{\mathcal{I}},:) = FW.$$

Lastly, we have

$$Fc(1) = [F(:,\mathcal{I}), F(:,\bar{\mathcal{I}})]\, PP^T \begin{bmatrix} W(\mathcal{I},:) + EW(\bar{\mathcal{I}},:) \\ \mathbf{0} \end{bmatrix} = F(:,\mathcal{I})W(\mathcal{I},:) + F(:,\mathcal{I})E\, W(\bar{\mathcal{I}},:)$$

which is independent of $F(:,\bar{\mathcal{I}})$. □

The next lemma will be helpful to prove our main Theorem 3.4.13 where we want to modify the biases of a subset of neurons at layer k so that the output of this layer (i.e. F_k) achieves full rank.

Lemma 3.4.11 *Given $v \in \mathbb{R}^n$ with $v_i \neq v_j \, \forall \, i \neq j$, and $\sigma : \mathbb{R} \to \mathbb{R}$ satisfies Assumption 3.4.2. Let $S \subseteq \mathbb{R}^n$ be defined as $S = \{\sigma(v + b\mathbf{1}_n) \mid b \in \mathbb{R}\}$. Then it holds Span$(S) = \mathbb{R}^n$.*

Proof: Suppose by contradiction that $\dim(\text{Span}(S)) < n$. Then there exists $\lambda \in \mathbb{R}^n, \lambda \neq 0$ such that $\lambda \perp \text{Span}(S)$, and thus it holds $\sum_{i=1}^n \lambda_i \sigma(v_i + b) = 0$ for every $b \in \mathbb{R}$. We assume w.l.o.g. that $\lambda_1 \neq 0$ then it holds

$$\sigma(v_1 + b) = -\sum_{i=2}^{n} \frac{\lambda_i}{\lambda_1} \sigma(v_i + b), \quad \forall \, b \in \mathbb{R}.$$

By a change of variable, we have

$$\sigma(c) = -\sum_{i=2}^{n} \frac{\lambda_i}{\lambda_1} \sigma(c + v_i - v_1), \quad \forall \, c \in \mathbb{R},$$

which contradicts Assumption 3.4.2. Thus $\text{Span}(S) = \mathbb{R}^n$. □

Lastly we recall a standard result from topology (e.g., see Apostol (1974), Theorem 4.23, p. 82).

Proposition 3.4.12 *Let $f : \mathbb{R}^m \to \mathbb{R}^n$ be a continuous function. If $U \subseteq \mathbb{R}^n$ is an open set then $f^{-1}(U)$ is also open.*

We are now ready to state our main result in this section.

Theorem 3.4.13 *Let Assumption 3.4.1 and Assumption 3.4.2 hold. Suppose that there exists a layer $k \in [1, L-1]$ such that $n_k \geq N$ and $n_{k+1} > \ldots > n_L$. Then the following hold:*

1. *The loss Φ has no bad local valleys.*
2. *If $k \leq L-2$ then every local valley of Φ is unbounded.*

Proof:

1. The high level idea is that inside every local valley of the loss one can find a point where the feature representations of all the training samples are linearly independent at the wide layer k, and thus an application of Theorem 3.4.9 to the subnetwork from layer k to the output layer yields the result. We present the proof details below. Let C be a connected component of some strict sublevel set $L_\alpha^s = \Phi^{-1}((-\infty, \alpha))$, for some $\alpha > p^*$. By Proposition 5.1.1, L_α^s is an open set and thus C must be open.

   Step 1: Finding a point inside C where F_k has full rank. Let $\theta \in C$ be such that the pre-activation outputs at the first hidden layer are distinct for all training samples. Note that such θ always exist since Assumption 3.4.4 implies that the set of W_1 where this does not hold has Lebesgue measure zero, whereas C has positive measure. This combined with Assumption 3.4.1 implies that the (post-activation) outputs at the first hidden layer are distinct for all training samples. Now one can view these outputs at the first layer as inputs to the next layer and argue similarly. By repeating this argument and using the fact that C has positive measure, we conclude that there exists $\theta \in C$ such that the outputs at layer $k-1$ are distinct for all training samples, *i.e.* $(F_{k-1})_{i:} \neq (F_{k-1})_{j:}$ for every $i \neq j$. Let V be the pre-activation output (without bias term) at layer k, in particular $V = F_{k-1}W_k = [v_1, \ldots, v_{n_k}] \in \mathbb{R}^{N \times n_k}$. Since F_{k-1} has distinct rows, one can easily perturb W_k so that every column of V has distinct entries. Note here that the set of W_k where this does not hold has measure zero whereas C has positive

measure. From here we conclude that C must contain a point where every v_j has distinct entries. To simplify notation, let $a = b_k \in \mathbb{R}^{n_k}$, then by definition,

$$F_k = [\sigma(v_1 + \mathbf{1}_N a_1), \ldots, \sigma(v_{n_k} + \mathbf{1}_N a_{n_k})]. \tag{3.9}$$

Suppose that F_k has low rank, otherwise we are done. Let $r = \mathit{rank}(F_k) < N \leq n_k$ and $\mathcal{I} \subset \{1, \ldots, n_k\}, |\mathcal{I}| = r$ be the subset of columns of F_k so that $\mathit{rank}(F_k(:, \mathcal{I})) = \mathit{rank}(F_k)$, and $\bar{\mathcal{I}}$ the remaining columns. By applying Lemma 3.4.10 to $(F_k, W_{k+1}, \mathcal{I})$, we can follow a continuous path with invariant loss (*i.e.* entirely contained inside C) to arrive at some point where $F_k W_{k+1}$ is independent of $F_k(:, \bar{\mathcal{I}})$. It remains to show how to change $F_k(:, \bar{\mathcal{I}})$ by modifying certain parameters so that F_k has full rank. Let $p = |\bar{\mathcal{I}}| = n_k - r$ and $\bar{\mathcal{I}} = \{j_1, \ldots, j_p\}$. From (3.9) we have

$$F_k(:, \bar{\mathcal{I}}) = [\sigma(v_{j_1} + \mathbf{1}_N a_{j_1}), \ldots, \sigma(v_{j_p} + \mathbf{1}_N a_{j_p})].$$

Let $\mathrm{col}(\cdot)$ denotes the column space of a matrix. Then $\dim(\mathrm{col}(F_k(:, \mathcal{I}))) = r < N$. Since v_{j_1} has distinct entries and σ satisfies Assumption 3.4.2, Lemma 3.4.11 implies that there must exist $a_{j_1} \in \mathbb{R}$ so that $\sigma(v_{j_1} + \mathbf{1}_N a_{j_1}) \notin \mathrm{col}(F_k(:, \mathcal{I}))$, because otherwise we have

$$\mathrm{Span}\,\{\sigma(v_{j_1} + \mathbf{1}_N a_{j_1}) \mid a_{j_1} \in \mathbb{R}\} \in \mathrm{col}(F_k(:, \mathcal{I}))$$

whose dimension is strictly smaller than N and thus contradicts Lemma 3.4.11. So now, we can pick one such value for a_{j_1} and follow a direct line segment between its current value and the new value. Note that the loss is invariant on this segment since any changes on a_{j_1} only affects $F_k(:, \bar{\mathcal{I}})$ which however has no influence on the loss by above construction. Moreover, at the new value of a_{j_1} the rank of F_k has increased by 1. Since $n_k \geq N$ by assumption, we have $p \geq N - r$ and thus one can choose $\{a_{j_2}, \ldots, a_{j_{N-r}}\}$ in a similar way to increase the rank of F_k, and finally obtain $\mathit{rank}(F_k) = N$.

Step 2: Applying Theorem 3.4.9 to the subnetwork above k. Suppose that we have found from previous step a $\theta = ((W_l^*, b_l^*)_{l=1}^L) \in C$ so that F_k has full rank. Let $g : \Omega_{k+1} \times \ldots \times \Omega_L \to \mathbb{R}$ be the function defined as

$$g\Big((W_l, b_l)_{l=k+1}^L\Big) = \Phi\Big((W_l^*, b_l^*)_{l=1}^k, (W_l, b_l)_{l=k+1}^L\Big) \tag{3.10}$$

We recall that C is a connected component of L_α^s. It holds $g\Big((W_l^*, b_l^*)_{l=k+1}^L\Big) = \Phi(\theta) \leq \alpha$. Now one can view g as a loss function for the subnetwork from layer k till layer L where F_k now plays the role of the training data to this subnetwork. Since $\mathit{rank}(F_k) = N$ and $n_{k+1} > \ldots > n_L$, an application of Theorem 3.4.9 to this subnetwork yields that g has connected sublevel sets and g can attain any value arbitrarily close to p^*. Let $\epsilon \in (p^*, \alpha)$ and $(W_l', b_l')_{l=k+1}^L$ be any point such that $g\Big((W_l', b_l')_{l=k+1}^L\Big) \leq \epsilon$. Since both $(W_l^*, b_l^*)_{l=k+1}^L$ and $(W_l', b_l')_{l=k+1}^L$ belongs to the α-sublevel set of g, which is a connected set, there must exist a continuous path from $(W_l^*, b_l^*)_{l=k+1}^L$ to $(W_l', b_l')_{l=k+1}^L$ on which the value of g is not larger than α. This combined with (3.10) implies that there is also a continuous path from $\theta = \Big((W_l^*, b_l^*)_{l=1}^k, (W_l^*, b_l^*)_{l=k+1}^L\Big)$ to $\theta' := \Big((W_l^*, b_l^*)_{l=1}^k, (W_l', b_l')_{l=k+1}^L\Big)$ on which the loss Φ is not larger than α. Since C is connected, it must hold $\theta' \in C$. Moreover, we have $\Phi(\theta') = g\Big((W_l', b_l')_{l=k+1}^L\Big) \leq \epsilon$. Since ϵ can be chosen arbitrarily small and close to p^*, we conclude that the loss Φ can be made arbitrarily small inside C, and thus Φ has no bad local valleys.

2. Let C be a local valley, which by Definition 3.3.7 is a connected component of some strict sublevel set $L_\alpha^s = \Phi^{-1}((-\infty, \alpha))$. According the the proof of the first statement above, one can find a $\theta = (W_l^*, b_l^*)_{l=1}^L \in C$ so that $F_k(\theta)$ has full rank. Now one can view $F_k(\theta)$ as the training data for the subnetwork from layer k till layer L. The new loss is defined for this subnetwork as

$$g\Big((W_l, b_l)_{l=k+1}^L\Big) = \Phi\Big((W_l^*, b_l^*)_{l=1}^k, (W_l, b_l)_{l=k+1}^L\Big).$$

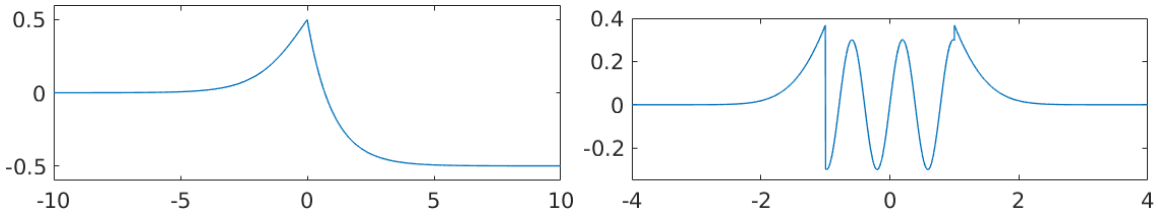


Figure 3.1: Left: an example function with exponential tails where local minima do NOT exist but local/global valleys still exist. **Right**: a different function which satisfies every local minimum is a global minimum, but bad local valleys still exist at both infinities (exponential tails) where local search algorithms easily get stuck.

By our assumptions, σ satisfies Assumption 3.4.1 and $n_{k+1} > \ldots > n_L$, thus the above subnetwork with the new loss g and training data $F_k(\theta)$ satisfy all the conditions of Theorem 3.4.9, and so it follows that g has unbounded level set components. Let $\beta := g\Big((W_l^*, b_l^*)_{l=k+1}^L\Big) = \Phi(\theta) < \alpha$. Let E be a connected component of the level set $g^{-1}(\beta)$ which contains $(W_l^*, b_l^*)_{l=k+1}^L$. Let $D = \left\{ \Big((W_l^*, b_l^*)_{l=1}^k, (W_l, b_l)_{l=k+1}^L\Big) \;\middle|\; (W_l, b_l)_{l=k+1}^L \in E \right\}$. Then D is connected and unbounded since E is connected and unbounded. It holds for every $\theta' \in D$ that $\Phi(\theta') = \beta$, and thus $D \subseteq \Phi^{-1}(\beta) \subseteq L_\alpha^s$, where the last inclusion follows from $\beta < \alpha$. Moreover, we have $\theta = \Big((W_l^*, b_l^*)_{l=1}^k, (W_l^*, b_l^*)_{l=k+1}^L\Big) \in D$ and also $\theta \in C$, it follows that $D \subseteq C$ since C is already the maximal connected component of L_α^s. Since D is unbounded, C must also be unbounded, which finishes the proof.

□

The conditions of Theorem 3.4.13 are satisfied for any strictly monotonic and piecewise linear activation function such as Leaky-ReLU (see Lemma 3.4.3). We note that for Leaky-ReLU and other similar homogeneous activation functions, the second statement of Theorem 3.4.13 is quite straightforward. Indeed, if one scales all parameters of one hidden layer by some arbitrarily large factor $k > 0$ and the weight matrix of the following layer by $1/k$ then the network output will be unchanged, and so every connected component of every level set (also sublevel set) must extend to infinity and thus be unbounded. However, for general non-homogeneous activation functions, the second statement is non-trivial. The first statement of Theorem 3.4.13 implies that there is a continuous path from any point in parameter space on which the loss is non-increasing and gets arbitrarily close to p^*. At this point, one might wonder if a function satisfies "every local minimum is a global minimum" would automatically contain no bad local valleys. Unfortunately this is not true in general. Indeed, Figure 3.1 shows two counter-examples where a function does not have any bad local mimina, but bad local valleys still exist. The reason for this lies at the fact that bad local valleys generally need not contain any critical point though in theory they can have very large volume or even be unbounded. Thus any pure results on global optimality of local minima with no further information on the loss would not be sufficient to guarantee convergence of local search algorithms to a global minimum, especially if they are initialized in such regions. Similar to the second statement of Theorem 3.4.13, the first statement on one hand can guarantee absence of strict local minima, but on the other hand cannot rule out the possibility of non-strict bad local minima. This suggests that it might be desirable to have in practice both properties for the loss surface of neural nets, that is, there are no bad local valleys and every local minimum is a global minimum. Overall, the statements of Theorem 3.4.13 altogether imply that every local valley must be an "unbounded" global valley in which the loss can attain any value arbitrarily close to p^*.

3.4.4 Large width of first hidden layer leads to connected sublevel sets

In the previous section (see Theorem 3.4.13), we prove that if one of the hidden layers has at least N neurons then the loss surface has no bad local valleys. In this section, we analyze a special case of the network where the first hidden layer has at least $2N$ neurons, that is, $n_1 \geq 2N$. Under this regime, we show that every sublevel set of the loss Φ must be connected.

Given two points in parameter space, the first step of our proof is show that there exist two continuous paths starting from each point separately so that these paths lead us to two new points where the output matrices at the first hidden layer $F_1 = \sigma(XW_1 + \mathbf{1}_N b_1^T) \in \mathbb{R}^{N \times n_1}$ have full row rank and the loss is constant on each path. This step can be done by the following lemma, in which the second property shows that the pre-activation output at the second layer $\sigma(XW_1 + \mathbf{1}_1 b_1^T)W_2$ is invariant on the path and so the loss stays constant.

Lemma 3.4.14 *Let $(X, W, b, V) \in \mathbb{R}^{N \times d} \times \mathbb{R}^{d \times n} \times \mathbb{R}^n \times \mathbb{R}^{n \times p}$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ satisfy Assumption 3.4.2. Suppose that $n \geq N$ and X has distinct rows. Let $Z = \sigma(XW + \mathbf{1}_N b^T)\, V$. There is a continuous curve $c : [0, 1] \to \mathbb{R}^{d \times n} \times \mathbb{R}^n \times \mathbb{R}^{n \times p}$ with $c(\lambda) = (W(\lambda), b(\lambda), V(\lambda))$ satisfying:*

1. *$c(0) = (W, b, V)$.*
2. *$\sigma\Big(XW(\lambda)) + \mathbf{1}_N b(\lambda)^T\Big)\, V(\lambda) = Z$, $\forall\, \lambda \in [0, 1]$.*
3. *$rank\left(\sigma\Big(XW(1) + \mathbf{1}_N b(1)^T\Big)\right) = N$.*

Proof: Let $F = \sigma(XW + \mathbf{1}_N b^T) \in \mathbb{R}^{N \times n}$. If F already has full rank then we are done. Otherwise let $r = rank(F) < N \leq n$. Let $\mathcal{I}$ denote a set of column indices of F so that $rank(F(:, \mathcal{I})) = r$ and $\bar{\mathcal{I}}$ the remaining columns. By applying Lemma 3.4.10 to $(F, V, \mathcal{I})$, we can find a continuous path $V(\lambda)$ so that we will arrive at some point where $FV(\lambda)$ is invariant on the path and it holds at the end point of the path that FV is independent of $F(:, \bar{\mathcal{I}})$. This means that we can arbitrarily change the values of $W(:, \bar{\mathcal{I}})$ and $b(\bar{\mathcal{I}})$ without affecting the value of Z, because any changes of these variables are absorbed into $F(:, \bar{\mathcal{I}})$ which anyway has no influence on FV. Thus it is sufficient to show that there exist $W(:, \bar{\mathcal{I}})$ and $b(\bar{\mathcal{I}})$ for which F has full rank. Let $p = n - r$ and $\bar{\mathcal{I}} = \{j_1, \ldots, j_p\}$. Let $A = XW$ then $A(:, \bar{\mathcal{I}}) := [a_{j_1}, \ldots, a_{j_p}] = XW(:, \bar{\mathcal{I}})$. By assumption X has distinct rows, one can choose $W(:, \bar{\mathcal{I}})$ so that each $a_{j_k} \in \mathbb{R}^N$ has distinct entries. Then we have

$$F(:, \bar{\mathcal{I}}) = [\sigma(a_{j_1} + \mathbf{1}_N b_{j_1}), \ldots, \sigma(a_{j_p} + \mathbf{1}_N b_{j_p})].$$

Let $\mathrm{col}(\cdot)$ denotes the column space of a matrix. It holds $\dim(\mathrm{col}(F(:, \mathcal{I}))) = r < N$. Since a_{j_1} has distinct entries, Lemma 3.4.11 implies that there must exist $b_{j_1} \in \mathbb{R}$ so that $\sigma(a_{j_1} + \mathbf{1}_N b_{j_1}) \notin \mathrm{col}(F(:, \mathcal{I}))$, because otherwise $\mathrm{Span}\,\{\sigma(a_{j_1} + \mathbf{1}_N b_{j_1}) \mid b_{j_1} \in \mathbb{R}\} \in \mathrm{col}(F(:, \mathcal{I}))$ whose dimension is strictly smaller than N, which contradicts Lemma 3.4.11. So it means that there is $b_{j_1} \in \mathbb{R}$ so that $rank(F)$ increases by 1. By assumption $n \geq N$, it follows that $p \geq N - r$, and thus we can choose $\left\{b_{j_2}, \ldots, b_{j_{N-r}}\right\}$ similarly to obtain $rank(F) = N$. □

Given two points θ, θ' in parameter space where the matrices $F_1(\theta), F_1(\theta')$ have full row rank, our next step is to make the parameters of the first layer of these two points identical to each other. This can be done by finding a continuous path from θ to some other point where the weights of the first layer are equal to that of θ' and the loss is constant on the path, which is shown next.

Lemma 3.4.15 *Let $(X, W, V, W') \in \mathbb{R}^{N \times d} \times \mathbb{R}^{d \times n} \times \mathbb{R}^{n \times p} \times \mathbb{R}^{d \times n}$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ satisfy Assumption 3.4.2. Suppose that $n \geq 2N$ and $rank(\sigma(XW)) = N$, $rank(\sigma(XW')) = N$. Then there is a continuous curve $c : [0, 1] \to \mathbb{R}^{d \times n} \times \mathbb{R}^{n \times p}$ with $c(\lambda) = (W(\lambda), V(\lambda))$ which satisfies the following:*

1. $c(0) = (W, V)$.
2. $\sigma(XW(\lambda))\, V(\lambda) = \sigma(XW)\, V, \quad \forall\, \lambda \in [0, 1]$.
3. $W(1) = W'$.

Proof: We need to show that there is a continuous path from (W, V) to (W', V') for some $V' \in \mathbb{R}^{n \times p}$, so that the output function, defined by $Z := \sigma(XW)V$, is invariant along the path. Let $F = \sigma(XW) \in \mathbb{R}^{N \times n}$ and $F' = \sigma(XW')$. It holds $Z = FV$. Let I resp. I' denote the maximum subset of linearly independent columns of F resp. F' so that $\mathit{rank}(F(:, I)) = \mathit{rank}(F(:, I')) = N$, and $\bar{I}$ and $\bar{I}'$ be their complements. By the rank condition, we have $|I| = |I'| = N$. Since $\mathit{rank}(F) = N < n$, we can apply Lemma 3.4.10 to the tuple (F, V, I) to arrive at some point where the output Z is independent of $F(:, \bar{I})$. From here, we can update $W(:, \bar{I})$ arbitrarily so that it does not affect Z because any change to these weights only lead to changes on $F(:, \bar{I})$ which however has no influence on Z. So by taking a direct line segment from the current value of $W(:, \bar{I})$ to $W'(:, I')$, we achieve $W(:, \bar{I}) = W'(:, I')$. We refer to this step below as a copy step. Note here that since $n \geq 2N$ by assumption, we must have $|\bar{I}| \geq |I'|$. Moreover, if $|\bar{I}| > |I'|$ then we can simply ignore the redundant space in $W(:, \bar{I})$.

Now we already copy $W'(:, I')$ into $W(:, \bar{I})$, so it holds that $\mathit{rank}(F(:, \bar{I})) = \mathit{rank}(F'(:, I')) = N$. Let $K = I' \cap \bar{I}$ and $J = I' \cap I$ be disjoint subsets so that $I' = K \cup J$. Suppose w.l.o.g. that the above copy step has been done in such a way that $W(:, \bar{I} \cap I') = W'(:, K)$. Now we apply Lemma 3.4.10 to $(F, V, \bar{I})$ to arrive at some point where Z is independent of $F(:, I)$, and thus we can easily obtain $W(:, I \cap I') = W'(:, J)$ by taking a direct line segment between these weights. So far, all the rows of $W'(:, K \cup J)$ have been copied into $W(:, I')$ at the right positions so we obtain that $W(:, I') = W'(:, I')$. It follows that $\mathit{rank}(F(:, I')) = \mathit{rank}(F'(:, I')) = N$ and thus we can apply Lemma 3.4.10 to (F, V, I') to arrive at some other point where Z is independent of $F(:, \bar{I}')$. From here we can easily obtain $W(:, \bar{I}') = W'(:, \bar{I}')$ by taking a direct line segment between these variables. Till now we already have $W = W'$. Moreover, all the paths which we have followed leave the output Z invariant. □

We are now ready to state the main result of this section.

Theorem 3.4.16 *Let Assumption 3.4.1 and Assumption 3.4.2 hold. Suppose that $n_1 \geq 2N$ and $n_2 > \ldots > n_L$. Then the following hold:*

1. *Every sublevel set of Φ is connected.*
2. *Every connected component of every level set of Φ is unbounded.*

Proof:

1. Let $\theta = (W_l, b_l)_{l=1}^L, \theta' = (W_l', b_l')_{l=1}^L$ be arbitrary points in some sublevel set L_α. It is sufficient to show that there is a connected path between θ and θ' on which the loss is not larger than α. The output at the first layer is given by

$$F_1(\theta) = \sigma([X, \mathbf{1}_N][W_1^T, b_1]^T), \quad F_1(\theta') = \sigma([X, \mathbf{1}_N][W_1'^T, b_1']^T).$$

   First, by applying Lemma 3.4.14 to (X, W_1, b_1, W_2), we can assume that $F_1(\theta)$ has full rank, because otherwise there is a continuous path starting from θ to some other point where the rank condition is fulfilled and the loss is invariant on the path, and so we can reset θ to this new point. Similarly, we can assume that $F_1(\theta')$ has full rank.

   Next, by applying Lemma 3.4.15 to $\left([X, \mathbf{1}_N], [W_1^T, b_1]^T, W_2, [W_1'^T, b_1']^T\right)$, and with the similar argument above, we can drive θ to some other point where the parameters of the first hidden

layer agree with the corresponding ones of θ'. So we can assume w.l.o.g. that $(W_1, b_1) = (W_1', b_1')$. Note that we do not modify θ' but θ at this step, and thus $F_1(\theta')$ still has full rank.

Once the first hidden layer of θ and θ' coincide, one can view the output of this layer, say $F_1 := F_1(\theta) = F_1(\theta')$ with $rank(F_1) = N$, as the new training data for the subnetwork from layer 1 till layer L (given that (W_1, b_1) is fixed). This subnetwork and the new data F_1 satisfy all the conditions of Theorem 3.4.9, and so it follows that the loss Φ restricted to this subnetwork has connected sublevel sets, which implies that there is a connected path between $(W_l, b_l)_{l=2}^L$ and $(W_l', b_l')_{l=2}^L$ on which the loss is not larger than α. This indicates that there is also a connected path between θ and θ' in L_α and so L_α must be connected.

2. Let $\theta \in \Omega$ be an arbitrary point. Denote $F_1 = F_1(\theta)$ and let $\mathcal{I} \subset \{1, \dots, N\}$ be such that $rank(F_1(:, \mathcal{I})) = rank(F_1)$. Since $rank(F_1) \leq \min(N, n_1) < n_1$, we can apply Lemma 3.4.10 to the tuple $(F_1, W_2, \mathcal{I})$ to find a continuous path $W_2(\lambda)$ which drives θ to some other point where the output at 2nd layer $F_1 W_2$ is independent of $F_1(:, \bar{\mathcal{I}})$. Note that the network output at 2nd layer is invariant on this path and hence the entire path belongs to the same level set component with θ. From that point, one can easily scale $(W_1(:, \bar{\mathcal{I}}), b_1(\bar{\mathcal{I}}))$ to arbitrarily large values without affecting the output. Since this path has constant loss and is unbounded, it follows that every level set component of Φ is unbounded.

□

Theorem 3.4.16 shows a stronger result than Theorem 3.4.13 as it not only implies that there are no bad local valleys but also there is a unique global valley. Equivalently, all finite global minima (if exist) must be connected. This can be seen as a generalization of previous result Venturi et al. (2018) from one hidden layer networks and square loss to arbitrary deep networks and convex losses. Interestingly, recent work Draxler et al. (2018); Garipov et al. (2018) have empirically shown that different global minima of several existing CNN architectures can be connected by a continuous path on which the loss has similar values. While our results are not directly applicable to these models, we consider this as a stepping stone for such an extension in future work. Similar to our main results in the previous sections, the unboundedness property of level sets as shown in Theorem 3.4.16 implies that Φ has no bounded local valleys nor strict local extrema.

3.4.5 Extensions to ReLU activation function

In this section, we show an extension of our results from the previous sections to the ReLU activation function. This is done by removing Assumption 3.4.1 from Theorem 3.4.13 and Theorem 3.4.16.

Theorem 3.4.17 *All the following hold under Assumption 3.4.2:*

1. *If* $\min\{n_1, \dots, n_{L-1}\} \geq N$ *then the loss function* Φ *has no bad local valleys.*
2. *If* $\min\{n_1, \dots, n_{L-1}\} \geq 2N$ *then every sublevel set of* Φ *is connected.*

Proof:

1. Let $\theta = (W_l, b_l)_{l=1}^L$ be an arbitrary point of some strict sublevel set L_α^s, for some $\alpha > p^*$. We will show that there is a continuous descent path starting from θ on which the loss is non-increasing and gets arbitrarily close to p^*. Indeed, for every ϵ arbitrarily close to p^* and $\epsilon \leq \alpha$, let $\hat{Y} \in \mathbb{R}^{N \times m}$ be such that $\varphi(\hat{Y}) \leq \epsilon$. Since X has distinct rows, $n_1 \geq N$, and the activation σ satisfies Assumption 3.4.2, an application of Lemma 3.4.14 to (X, W_1, b_1, W_2) shows that there is a continuous path with constant loss which leads θ to some other point where the output at the first hidden layer is full rank. So we can assume w.l.o.g. that it holds

for θ that $rank(F_1) = N$. By assumption $n_1 \geq N$ and $F_1 \in \mathbb{R}^{N \times n_1}$, it follows that F_1 must have distinct rows, and thus by applying Lemma 3.4.14 again to (F_1, W_2, b_2, W_3) we can assume w.l.o.g. that $rank(F_2) = N$. By repeating this argument to higher layers using our assumption on the width, we can eventually arrive at some $\theta = (W_l, b_l)_{l=1}^L$ where $rank(F_{L-1}) = N$. Thus there must exist $W_{L-1}^* \in \mathbb{R}^{n_{L-1} \times m}$ so that $F_{L-1}W_L^* = \hat{Y} - \mathbf{1}_N b_L^T$. Consider the line segment $W_L(\lambda) = (1-\lambda)W_L + \lambda W_L^*$, then it holds by convexity of φ that

$$\begin{aligned}
\Phi\Big((W_l, b_l)_{l=1}^{L-1}, (W_L(\lambda), b_L)\Big) &= \varphi\Big(F_{L-1}W_L(\lambda) + \mathbf{1}_N b_L^T\Big) \\
&= \varphi\Big((1-\lambda)(F_{L-1}W_L + \mathbf{1}_N b_L^T) + \lambda(F_{L-1}W_L^* + \mathbf{1}_N b_L^T)\Big) \\
&\leq (1-\lambda)\varphi(F_L) + \lambda\varphi(\hat{Y}) \\
&< (1-\lambda)\alpha + \lambda\epsilon \\
&\leq \alpha.
\end{aligned}$$

Thus the whole line segment is contained in L_α^s. For $\lambda = 1$ it follows that $\Big((W_l, b_l)_{l=1}^{L-1}, (W_L^*, b_L)\Big) \in L_\alpha^s$. Moreover, it holds $\Phi\Big((W_l, b_l)_{l=1}^{L-1}, (W_L^*, b_L)\Big) = \varphi(\hat{Y}) \leq \epsilon$. Since ϵ can be chosen arbitrarily close to p^*, we conclude that Φ can be made arbitrarily close to p^* in every strict sublevel set which implies that Φ has no bad local valleys.

2. Our first step is similar to the first step in the proof of Theorem 3.4.16, which we repeat below for completeness. Let $\theta = (W_l, b_l)_{l=1}^L, \theta' = (W_l', b_l')_{l=1}^L$ be arbitrary points in some sublevel set L_α. It is sufficient to show that there is a connected path between θ and θ' on which the loss is not larger than α. In the following, we denote F_k and F_k' as the output at a layer k for θ and θ' respectively. The output at the first layer is:

$$F_1 = \sigma([X, \mathbf{1}_N][W_1^T, b_1]^T), \quad F_1' = \sigma([X, \mathbf{1}_N][W_1'^T, b_1']^T).$$

By applying Lemma 3.4.14 to (X, W_1, b_1, W_2) and (X, W_1', b_1', W_2') we can assume w.l.o.g. that both F_1 and F_1' have full rank, since otherwise there is a continuous path starting from each point and leading to some other point where the rank condition is fulfilled and the network output at second layer is invariant on the path. Once F_1 and F_1' have full rank, we can apply Lemma 3.4.15 to $\Big([X, \mathbf{1}_N], [W_1^T, b_1]^T, W_2, [W_1'^T, b_1']^T\Big)$ in order to drive θ to some other point where the parameters of the first layer are all equal to the corresponding ones of θ'. So we can assume w.l.o.g. that $(W_1, b_1) = (W_1', b_1')$.

Once the network parameters of θ and θ' coincide at the first hidden layer, we can view the output of this layer, which is equal for both points (i.e., $F_1 = F_1'$), as the new training data for the subnetwork from layer 2 till layer L. Same as before, we first apply Lemma 3.4.14 to (F_1, W_2, b_2, W_3) and (F_1', W_2', b_2', W_3') to drive θ and θ' respectively to other new points where both F_2 and F_2' have full rank. Note that this path only acts on (W_2, b_2, W_3) and thus leaves everything else below layer 2 invariant, in particular we still have $F_1 = F_1'$. Then we can apply Lemma 3.4.15 again to the tuple $\Big([F_1, \mathbf{1}_N], [W_2^T, b_2]^T, W_3, [W_2'^T, b_2']^T\Big)$ to drive θ to some other point where $(W_2, b_2) = (W_2', b_2')$.

By repeating the above argument to the last hidden layer, we can make all network parameters of θ and θ' coincide for all layers, except the output layer. In particular, the path that each θ and θ' has followed has invariant loss. The output of the last hidden layer for these points is $A := F_{L-1} = F_{L-1}'$. The loss at these two points can be rewritten as

$$\Phi(\theta) = \varphi\Big([A, \mathbf{1}_N]\begin{bmatrix} W_L \\ b_L^T \end{bmatrix}\Big), \quad \Phi(\theta') = \varphi\Big([A, \mathbf{1}_N]\begin{bmatrix} W_L' \\ b_L'^T \end{bmatrix}\Big).$$

Since φ is convex, the line segment

$$(1-\lambda)\begin{bmatrix} W_L \\ b_L^T \end{bmatrix} + \lambda \begin{bmatrix} W'_L \\ b'^T_L \end{bmatrix}$$

must yield a continuous descent path between (W_L, b_L) and (W'_L, b'_L), and so the loss of every point on this path cannot be larger than α. Moreover, this path connects θ and θ' together, and thus L_α has to be connected.

□

It is clear that the conditions of Theorem 3.4.17 are still far from the practical settings. These conditions are also significantly stronger than that of Theorem 3.4.16 as they requires all the hidden layers to be wide enough. Nevertheless, we note that the similar conditions have also been used by recent theoretical work Allen-Zhu et al. (2018b); Du et al. (2018a) in proving convergence guarantees of gradient descent methods to a zero-training-error solution. At the moment, we find these results interesting as they seem to suggest that Leaky-ReLU might lead to a much easier optimization landscape than ReLU.

3.5 Summary

For deep fully connected networks having a sufficiently wide hidden layer and piecewise linear activation functions such as Leaky-ReLU, we show that every sublevel set of the loss function is connected, and every connected component of every level set is unbounded. This leads to several interesting implications on their optimization landscape such as:

1. There is no bad (or bounded) local valley, no strict local minima nor strict local maxima
2. There is a continuous descent path from any point in parameter space on which the loss is non-increasing and gets arbitrarily close to the infimum of the loss.
3. There is a unique global valley, and thus all finite global minima (if exist) are connected.

There are two open problems that emerge from this result: 1) how to extend our current analysis to the case of convolutional neural networks as analyzed in Chapter 2, and 2) whether it is possible to relax the current critical assumption on the wide layer (i.e. $n_k \geq N$) and bring it closer to practice? In the next chapter, we will address the second problem by identifying a new class of network architectures which are practically relevant and provably have no bad local valleys. By adding skip-connections from a random subset of N hidden neurons (possibly from multiple hidden layers) to the output layer, we show that the above critical condition can be relaxed to $n_1 + \ldots + n_{L-1} \geq N$.

Chapter 4

CNNs with skip-connections to the output layer have no bad local valleys

4.1 Introduction

Having gained insights from the previous chapters, we are motivated to come up with a class of network architectures which are more practically relevant and provably have a nice loss surface. In this chapter, we identify a family of deep networks with skip connections to the output layer whose loss landscape has no bad local valleys, that is, there is a continuous path from anywhere in parameters space on which the loss is non-increasing and gets arbitrarily close to a global minimum. (see Figure 4.1 for the illustration). Our setting is for the empirical loss and there are no distributional assumptions on the training data. Moreover, we study directly the standard loss functions such as cross-entropy loss in deep learning for multi-class problems. There are little assumptions on the network structure which can be arbitrarily deep and can have convolutional layers (weight sharing) and skip-connections between hidden layers. From a practical perspective, one can generate an architecture which fulfills our conditions by taking an existing CNN architecture and then adding skip-connections from a random subset of N neurons (N is the number of training samples), possibly from multiple hidden layers, to the output layer (see Figure 4.2 for an illustration). For these networks we show that there always exists a continuous path from any point in parameter space on which the loss is non-increasing and gets arbitrarily close to zero. We note that this implies the loss landscape has no strict local minima, but theoretically non-strict local minima can still exist. Beside that, we show that the loss has also no local maxima.

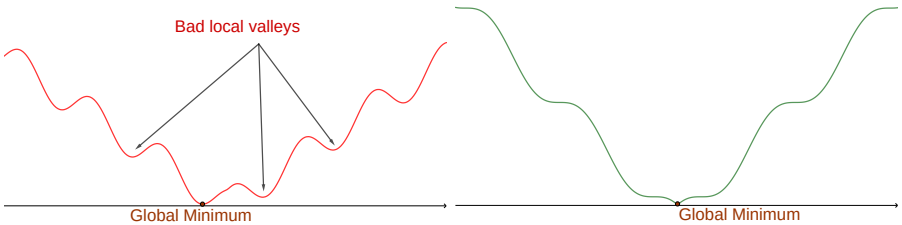


Figure 4.1: An example function with bad local valleys (left) and without bad local valley (right).

Beside the theoretical analysis, we show in experiments that despite achieving zero training error, the presented class of networks generalize well in practice when trained with stochastic gradient descent (SGD) whereas an alternative training procedure guaranteed to achieve zero training error has significantly worse generalization performance and is overfitting. Thus the presented class of networks offers an interesting test bed for future work to theoretically study the implicit bias of SGD. All our main results in this chapter have been published at Nguyen et al. (2019).

4.2 Description of network architecture

We consider a family of deep neural networks which have d input units, H hidden units, m output units and satisfy the following conditions:

1. Every hidden unit of the first layer can be connected to an arbitrary subset of input units.
2. Every hidden unit at higher layers can take as input an arbitrary subset of hidden units from (multiple) lower hidden layers.
3. Any subgroup of hidden units lying on the same layer can have non-shared or shared weights, in the later case their number of incoming units have to be equal.
4. There exist N hidden units which are connected to the output nodes with independent weights (N denotes the number of training samples).
5. The output of every hidden unit j in the network, denoted as $f_j : \mathbb{R}^d \to \mathbb{R}$, is given as

$$f_j(x) = \sigma_j\Big(b_j + \sum_{k:k\to j} f_k(x) u_{k\to j}\Big)$$

   where $x \in \mathbb{R}^d$ is an input vector of the network, $\sigma_j : \mathbb{R} \to \mathbb{R}$ is the activation function of unit j, $b_j \in \mathbb{R}$ is the bias of unit j, and $u_{k\to j} \in \mathbb{R}$ the weight from unit k to unit j.

This definition covers a class of deep fully connected and convolutional neural networks with an additional condition on the number of connections to the output layer. In particular, while conventional architectures have just connections from the last hidden layer to the output, we require in our setting that there must exist at least N neurons, "regardless" of their hidden layer, that are connected to the output layer. Essentially, this means that if the last hidden layer of a traditional network has just $L < N$ neurons then one can add connections from $N - L$ neurons in the hidden layers below it to the output layer so that the network fulfills our conditions.

Similar skip-connections have been used in DenseNet (Huang et al., 2017) which are different from identity skip-connections as used in ResNets (He et al., 2016). In Figure 4.2 we illustrate a network with and without skip connections to the output layer which is analyzed in this chapter. We note that several architectures like DenseNets Huang et al. (2017) already have skip-connections between hidden layers in their original architecture, whereas our special skip-connections go from hidden layers directly to the output layer. As our framework allow both kinds to exist in the same network (see Figure 4.2 for an example), we would like to separate them from each other by making the convention that in the following skip-connections, if not stated otherwise, always refer to ones which connect hidden neurons to output neurons.

We denote by d the dimension of the input and index all neurons in the network from the input layer to the output layer as $1, 2, \ldots, d, d+1, \ldots, d+H, d+H+1, \ldots, d+H+m$ which correspond to d input units, H hidden units and m output units respectively. As we only allow directed arcs from lower layers to upper layers, it follows that $k < j$ for every $k \to j$. Let N be the number of training samples. Suppose that there are M hidden neurons which are directly connected to the output with

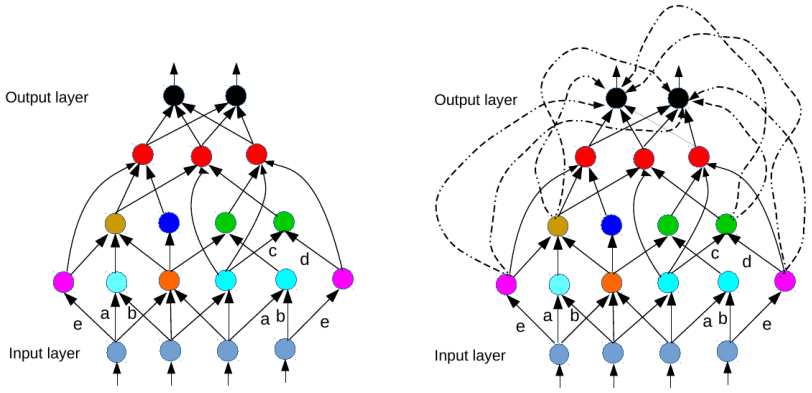


Figure 4.2: Left: An example neural network represented as directed acyclic graph. **Right**: The same network with skip connections added from a subset of hidden neurons to the output layer. All neurons with the same color can have shared or non-shared weights.

independent weights where it holds $N \leq M \leq H$. Let $\{p_1, \ldots, p_M\}$ with $p_j \in \{d+1, \ldots, d+H\}$ be the set of hidden units which are directly connected to the output units. Let $\text{in}(j)$ be the set of incoming nodes to unit j and $u_j = [u_{k\to j}]_{k\in\text{in}(j)}$ the weight vector of the j-th unit. Let $U = (u_{d+1}, \ldots, u_{d+H}, b_{d+1}, \ldots, b_{d+H})$ denote the set of all weights and biases of all hidden units in the network. Let $V \in \mathbb{R}^{M\times m}$ be the weight matrix which connects the M hidden neurons to the m output units of the network.

An important quantity in the following is the matrix $\Psi \in \mathbb{R}^{N\times M}$ defined as

$$\Psi = \begin{bmatrix} f_{p_1}(x_1) & \ldots & f_{p_M}(x_1) \\ \vdots & & \vdots \\ f_{p_1}(x_N) & \ldots & f_{p_M}(x_N) \end{bmatrix} \tag{4.1}$$

where $(x_i)_{i=1}^N$ is the set of training samples. As Ψ depends on U, we write Ψ_U or $\Psi(U)$ as a function of U. Let $G \in \mathbb{R}^{N\times m}$ be the output of the network for all training samples. In particular, G_{ij} is the value of the j-th output neuron for the training sample x_i. By definition we have

$$G_{ij} = \langle \Psi_{i:}, V_{:j} \rangle = \sum_{k=1}^{M} f_{p_k}(x_i) V_{kj}, \quad \forall i \in [N], j \in [m]$$

4.2.1 Empirical risk minimization

We consider the same general empirical risk as defined in Section 1.1, that is,

$$\Phi(U, V) = \varphi(G(U, V))$$

where $G(U, V) = \Psi(U)V \in \mathbb{R}^{N\times m}$ represents the output of the network for all training samples at the given set of network parameters (U, V), and $\varphi : \mathbb{R}^{N\times m} \to \mathbb{R}$ is any loss function satisfying Assumption 1.1.1 such as the standard cross-entropy loss, square loss and multiclass Hinge-loss. Similar to the previous chapters, let us define in the following,

$$p^* := \inf_{G\in\mathbb{R}^{N\times m}} \varphi(G)$$

which serves as a lower bound on our training objective Φ.

4.3 Main results

The following conditions are required for our main result to hold.

Assumption 4.3.1 *1. All activation functions $\{\sigma_{d+1}, \ldots, \sigma_{d+H}\}$ are real analytic and strictly increasing*

2. Among M neurons $\{p_1, \ldots, p_M\}$ which are connected to the output units, there exist $N \leq M$ neurons, say w.l.o.g. $\{p_1, \ldots, p_N\}$, such that one of the following conditions hold:

- *For every $1 \leq j \leq N : \sigma_{p_j}$ is bounded and $\lim_{t\to-\infty} \sigma_{p_j}(t) = 0$*
- *For every $1 \leq j \leq N : \sigma_{p_j}$ is the softplus activation (4.2), and there exists a backward path from p_j to the first hidden layer s.t. on this path there is no neuron which has skip-connections to the output or shared weights with other skip-connection neurons.*

3. The input patches of different training samples are distinct. In particular, let n_1 be the number of units in the first hidden layer and denote by S_i for $i \in [d+1, d+n_1]$ their input support, then for all $r \neq s \in [N]$, and $i \in [d+1, d+n_1]$, it holds $x_r|_{S_i} \neq x_s|_{S_i}$.

The first condition of Assumption 4.3.1 is satisfied for softplus, sigmoid, tanh, etc, whereas the second condition is fulfilled for sigmoid and softplus. For softplus activation function (smooth approximation of ReLU),

$$\sigma_\gamma(t) = \frac{1}{\gamma}\log(1 + e^{\gamma t}), \quad \text{for some } \gamma > 0, \tag{4.2}$$

we require an additional assumption on the network architecture. The third condition is always satisfied for fully connected networks if the training samples are distinct. For CNNs, this condition means that the corresponding input patches across different training samples are distinct. This could be potentially violated if the first convolutional layer has very small receptive fields. However, if this condition is violated for the given training set then after an arbitrarily small random perturbation of all training samples it will be satisfied with probability 1. Note that the M neurons which are directly connected to the output units can lie on different hidden layers in the network. Moreover, there is no condition on the width of every individual hidden layer as long as the total number of hidden neurons in the network satisfies $n_1 + \ldots + n_{L-1} \geq N$ so that our condition $M \geq N$ is feasible.

Overall, we would like to stress that Assumption 4.3.1 covers a quite large class of interesting network architectures but nevertheless allows us to show quite strong results on their empirical loss landscape.

The following key lemma shows that for almost all U, the matrix $\Psi(U)$ has full rank.

Lemma 4.3.2 *Under Assumption 4.3.1, the set of U such that $\Psi(U)$ has not full rank N has Lebesgue measure zero.*

Proof: We assume w.l.o.g. that $\{p_1, \ldots, p_N\}$ is a subset of the neurons with skip connections to the output layer and satisfy Assumption 4.3.1. In the following, we will show that there exists a weight configuration U such that the submatrix $\Psi_{1:N,1:N}$ has full rank. Using then that the determinant is an analytic function together with Lemma 2.2.2, we will conclude that the set of weight configurations U such that Ψ has *not* full rank has Lebesgue measure zero.

We remind that all the hidden units in the network are indexed from the first hidden layer till the higher layers as $d+1, \ldots, d+H$. For every hidden neuron $j \in [d+1, d+H]$, u_j denotes the associated weight vector

$$u_j = [u_{k\to j}]_{k\in \mathrm{in}(j)} \in \mathbb{R}^{|\mathrm{in}(j)|}, \quad \text{where } \mathrm{in}(j) = \text{the set of incoming units to unit } j.$$

Let n_1 be the number of units of the first hidden layer. For every neuron j from the first hidden layer, let us define the pre-activation output g_j,

$$g_j(x_i) = \sum_{k \to j} (x_i)_k u_{k \to j}.$$

Due to Assumptions 4.3.1 (condition 3), we can always choose the weights $\{u_{d+1}, \ldots, u_{d+n_1}\}$ so that the output of every neuron in the first layer is distinct for different training samples, that is $g_j(x_i) \neq g_j(x_{i'})$ for every $j \in [d+1, d+n_1]$ and $i \neq i'$. For every neuron $j \in [d+n_1+1, d+H]$ in the higher layers we choose the weight vector u_j such that it has exactly one 1 and 0 elsewhere. According to our definition of network in Section 4.2, the weight vectors of neurons of the same layer need not have the same dimension, but any subgroup of these neurons can still have shared weights as long as the dimensions among them agree. Thus the above choice of u is always possible. In the following, let $c(j)$ denote the neuron below j such that $u_{c(j) \to j} = 1$. This leads to

$$\sum_{k \to j} f_k(x) u_{k \to j} = f_{c(j)}(x).$$

Let $\alpha := (\alpha_{d+1}, \ldots, \alpha_{d+H})$ be a tuple of positive scalars. Let $\beta \in \mathbb{R}$ such that $\sigma_{p_j}(\beta) \neq 0$ for every $j \in [N]$. We consider a family of configurations of network parameters of the form $(\alpha_j u_j, b_j)_{j=d+1}^{d+H}$, where the biases are chosen as

$$\begin{aligned}
b_{p_j} &= \beta - \alpha_{p_j} g_{p_j}(x_j) \quad \forall\, j \in [N], p_j \in [d+1, d+n_1] \\
b_{p_j} &= \beta - \alpha_{p_j} f_{c(p_j)}(x_j) \quad \forall\, j \in [N], p_j \notin [d+1, d+n_1] \\
b_j &= 0 \quad \forall\, j \in \{d+1, \ldots, d+H\} \setminus \{p_1, \ldots, p_N\}
\end{aligned}$$

Note that the assignment of biases can be done via a forward pass through the network. By the above choice of biases and our definition of neurons in Section 4.2, we have

$$\begin{aligned}
f_{p_j}(x_i) &= \sigma_{p_j}\Big(\beta + \alpha_{p_j}\big(g_{p_j}(x_i) - g_{p_j}(x_j)\big)\Big), \quad \forall\, j \in [N], p_j \in [d+1, d+n_1], \\
f_{p_j}(x_i) &= \sigma_{p_j}\Big(\beta + \alpha_{p_j}\big(f_{c(p_j)}(x_i) - f_{c(p_j)}(x_j)\big)\Big) \quad \forall j \in [N], p_j \notin [d+1, d+n_1], \\
f_j(x_i) &= \sigma_j\Big(\alpha_j f_{c(j)}(x_i)\Big) \quad \forall j \in \{d+n_1+1, \ldots, d+H\} \setminus \{p_1, \ldots, p_N\}, \\
f_j(x_i) &= \sigma_j\Big(\alpha_j g_j(x_i)\Big) \quad \forall j \in \{d+1, \ldots, d+n_1\} \setminus \{p_1, \ldots, p_N\}.
\end{aligned} \tag{4.3}$$

One notes that the output of every skip-connection neuron p_j is given by the first equation if p_j lies on the first layer and by the second equation if p_j lies on higher layers. In the following, to reduce notational complexity we make a convention that: $f_{c(p_j)} = g_{p_j}$ for every p_j lies on the first layer. This allows us to use the second equation for every skip-connection neuron, that is,

$$f_{p_j}(x_i) = \sigma_{p_j}\Big(\beta + \alpha_{p_j}\big(f_{c(p_j)}(x_i) - f_{c(p_j)}(x_j)\big)\Big) \, \forall\, j \in [N]. \tag{4.4}$$

Now, since $\alpha > 0$ and all activation functions are strictly increasing by Assumption 4.3.1, one can easily show from the above recursive definitions that if p_j is a skip-connection neuron which does not lie on the first hidden layer then one has the relation: $f_{c(p_j)}(x_i) < f_{c(p_j)}(x_j)$ if and only if $g_{q_j}(x_i) < g_{q_j}(x_j)$, where q_j is some neuron in the first hidden layer. This means if one sorts the elements of the set $\big\{f_{c(p_j)}(x_1), \ldots, f_{c(p_j)}(x_N)\big\}$ in increasing order then for every positive tuple α, the order is fully determined by the corresponding order of $\big\{g_{q_j}(x_1), \ldots, g_{q_j}(x_N)\big\}$ for some neuron q_j in the first layer. Note that this order can be different for different neurons q_j in the first layer, and thus can be different for different skip-connection neurons p_j. Let π be a permutation such that it holds for every $j = 1, 2, \ldots, N$ that

$$\pi(j) = \underset{i \in \{1, \ldots, N\} \setminus \{\pi(1), \ldots, \pi(j-1)\}}{\arg\max} f_{c(p_j)}(x_i) \tag{4.5}$$

It follows from above that π is fully determined by the values of g at the first layer. By definition one has $f_{c(p_j)}(x_{\pi_i}) < f_{c(p_j)}(x_{\pi_j})$ for every $i > j$. Since π is independent of every positive tuple α and fully determined by the values of g, it can be fixed in the beginning. One can assume w.l.o.g. that π is the identity permutation as otherwise one can reorder the training samples according to π so that the rank of Ψ does not change. Thus it holds for every $\alpha > 0$ that

$$\delta_{ij} := f_{c(p_j)}(x_i) - f_{c(p_j)}(x_j) < 0 \quad \forall\, i, j \in [N], i > j \tag{4.6}$$

Now, we are ready to show that there exists a positive tuple α for which Ψ has full rank. We consider two cases of the activation functions of skip-connection neurons as stated in Assumption 4.3.1:

- In the first case, the activation functions $\sigma_{p_j} : \mathbb{R} \to \mathbb{R}$ for every $j \in [N]$ are strictly increasing, bounded and $\lim_{t\to-\infty} \sigma_{p_j}(t) = 0$. In the following, let $l(j)$ denote the layer index of the hidden unit j. For every hidden unit $j \in \{d+1, \ldots, d+H\}$ we set α_j to be the maximum of certain bounds (explained later in (4.8)) associated to all skip-connection neurons p_k lying on the same layer, that is,

$$\alpha_j = \max\left\{1,\ \max_{k\in[N]|l(p_k)=l(j)} \max_{i>k} \frac{\sigma_{p_k}^{-1}(\epsilon) - \beta}{f_{c(p_k)}(x_i) - f_{c(p_k)}(x_k)}\right\} \tag{4.7}$$

  where $\epsilon > 0$ is an arbitrarily small constant which will be specified later. There are a few remarks we want to make for Eq. (4.7) before proceeding with our proof. First, the second term in (4.7) can be empty if there is no skip-connection unit p_k which lies on the same layer as unit j, in which case α_j is simply set to 1. Second, α_j's are well-defined by constructing the values $f_{c(p_k)}(x_r)$, $r = 1, \ldots, N$ by a forward pass through the network (note that the network is a directed, acyclic graph; in particular, in the formula of α_j, one has $l(c(p_k)) < l(p_k) = l(j)$ and thus the computation of α_j is feasible given the values of hidden units lying below the layer of unit j, namely $f_{c(p_k)}$). Third, if j and j' are two neurons from the same layer, *i.e.* $l(j) = l(j')$, then it follows from (4.7) that $\alpha_j = \alpha_{j'}$, meaning that their corresponding weight vectors are scaled by the same factor, thus any potential weight sharing conditions imposed on these neurons can still be satisfied.

  The main idea of choosing the above values of α is to obtain

$$\Psi_{ij} = f_{p_j}(x_i) \leq \epsilon \quad \forall\, i, j \in [N], i > j. \tag{4.8}$$

  To see this, one first observes that the inequality (4.6) holds for the constructed values of α since they are all positive. From (4.7) it holds for every skip-connection unit p_j that

$$\alpha_{p_j} > \max_{i>j} \frac{\sigma_{p_j}^{-1}(\epsilon) - \beta}{f_{c(p_j)}(x_i) - f_{c(p_j)}(x_j)} \quad \forall\, j \in [N]$$

  which combined with (4.6) leads to

$$\alpha_{p_j}(f_{c(p_j)}(x_i) - f_{c(p_j)}(x_j)) \leq \sigma_{p_j}^{-1}(\epsilon) - \beta \quad \forall\, i, j \in [N], i > j.$$

  and thus using (4.4) we obtain (4.8).

  Coming back to the main proof of the lemma, since $\sigma_{p_j}(j \in [N])$ are bounded there exists a finite positive constant C such that it holds that

$$|\Psi_{ij}| \leq C \quad \forall\, i, j \in [N] \tag{4.9}$$

  By the Leibniz-formula one has

$$\det(\Psi_{1:N,1:N}) = \prod_{j=1}^{N} \sigma_{p_j}(\beta) + \sum_{\pi\in S_N\setminus\{\gamma\}} \operatorname{sign}(\pi) \prod_{j=1}^{N} \Psi_{\pi(j)j} \tag{4.10}$$

where S_N is the set of all $N!$ permutations of the set $\{1, \ldots, N\}$ and γ is the identity permutation. Now, one observes that for every permutation $\pi \neq \gamma$, there always exists at least one component j where $\pi(j) > j$ in which case it follows from (4.8) and (4.9) that

$$\Big| \sum_{\pi \in S_N \setminus \{\gamma\}} \operatorname{sign}(\pi) \prod_{j=1}^{N} \Psi_{\pi(j)j} \Big| \;\leq\; N!\, C^{N-1}\, \epsilon$$

By choosing $\epsilon = \frac{\left|\prod_{j=1}^{N} \sigma_{p_j}(\beta)\right|}{2N! C^{N-1}}$, we get that

$$\det(\Psi_{1:N,1:N}) \geq \prod_{j=1}^{N} \sigma_{p_j}(\beta) - \frac{1}{2} \prod_{j=1}^{N} \sigma_{p_j}(\beta) = \frac{1}{2} \prod_{j=1}^{N} \sigma_{p_j}(\beta) \neq 0$$

and thus Ψ has full rank.

- In the second case we consider the softplus activation function which satisfies our Assumption 4.3.1 that there exists a backward path from every skip-connection neuron p_j to the first hidden layer s.t. on this path there is no neuron which has skip-connections to the output or shared weights with other skip-connection neurons.

  We choose all the weights and biases similarly to the first case. The only difference is that for every skip-connection neuron $p_j (1 \leq j \leq N)$, the position of 1 in its weight vector u_{p_j} is chosen s.t. the value of neuron p_j is determined by the first neuron on the corresponding backward path as stated in Assumption 4.3.1, that is,

  $$\sum_{k \to p_j} f_k(x_i) u_{k \to p_j} = f_{c(p_j)}(x_i).$$

  For skip-connection neurons we set all $\{\alpha_{p_1}, \ldots, \alpha_{p_N}\}$ to some scalar variable α, and for non-skip connection neurons j we set $\alpha_j = 1$. From (4.4) and equations of (4.3) we have

  $$\begin{aligned} f_{p_j}(x_i) &= \sigma_{p_j}\Big(\beta + \alpha\big(f_{c(p_j)}(x_i) - f_{c(p_j)}(x_j)\big)\Big) \quad \forall j \in [N], \\ f_j(x_i) &= \sigma_j\big(f_{c(j)}(x_i)\big) \quad \forall j \in \{d+1, \ldots, d+H\} \setminus \{p_1, \ldots, p_N\}. \end{aligned} \tag{4.11}$$

  Note that with above construction of u and α, the only case where our weight sharing conditions can be potentially violated is between a skip-connection neuron ($\alpha_j = \alpha$) with a neuron on a backward path ($\alpha_j = 1$). However, this is not possible because our assumption in this case states that there is no weight sharing between a skip-connection neuron and a neuron on one of the backward paths.

  Next, by our assumption the recursive backward path $c^{(k)}(p_j)$ does not contain any skip-connection unit and thus will eventually end up at some neuron $q_j \in [d+1, d+n_1]$ in the first hidden layer after some finite number of steps. Thus we can write for every $j \in [N]$

  $$f_{p_j}(x_i) = \sigma_{p_j}\Big(\beta + \alpha\big(f_{c(p_j)}(x_i) - f_{c(p_j)}(x_j)\big)\Big),$$

  where

  $$f_{c(p_j)}(x_i) = \sigma_{c(p_j)}(\sigma_{c(c(p_j))}(\ldots(g_{q_j}(x_i))\ldots)) \quad \forall\, i \in [N].$$

  Moreover, we have from (4.6) that $f_{c(p_j)}(x_i) < f_{c(p_j)}(x_j)$ for every $i > j$. Note that softplus fulfills for $t < 0$, $\sigma_\gamma(t) \leq \frac{1}{\gamma} e^{\gamma t}$, whereas for $t > 0$ one has $\sigma_\gamma(t) \leq \frac{1}{\gamma} + t$. The latter property implies $\sigma^{(K)}(t) \leq \frac{K}{\gamma} + t$. Finally, this together implies that there exist positive constants c_1, c_2, c_3, c_4 such that it hods

  $$|\prod_{j=1}^{N} \Psi_{\pi(j)j}| \;\leq\; c_1 e^{-\alpha c_2} (c_3 + \alpha)^{N-1}.$$

This can be made arbitrarily small by increasing α. Thus we get

$$\lim_{\alpha\to\infty} \det(\Psi_{1:N,1:N}) = \prod_{j=1}^{N} \sigma_{p_j}(\beta) \neq 0$$

So far, we have shown that there always exist U such that Ψ has full rank. Since every activation function is real analytic by Assumption 4.3.1, every entry of Ψ is also a real analytic function of the network parameters where Ψ depends on. The set of low rank matrices Ψ can be characterized by a system of equations such that all the $\binom{M}{N}$ determinants of all $N \times N$ sub-matrices of Ψ are zero. As the determinant is a polynomial in the entries of the matrix and thus an analytic function of the entries and composition of analytic functions are again analytic, we conclude that each determinant is an analytic function of U. As shown above, there exists at least one U such that one of these determinant functions is not identically zero and thus by Lemma 2.2.2, the set of U where this determinant is zero has measure zero. But as all submatrices need to have low rank in order that Ψ has low rank, it follows that the set of U where Ψ has low rank has just measure zero. □

While we conjecture that the result of Lemma 4.3.2 holds for softplus activation function without the additional condition as mentioned in Assumption 4.3.1, the proof of this is considerably harder for such a general class of neural networks since one has to control the output of neurons with skip connection from different layers which depend on each other. However, we note that the condition is also not too restrictive as it just might require more connections from lower layers to upper layers but it does not require that the network is wide.

Before presenting our main result, we first recall the following definition of bad local valleys from Definition 3.3.7.

Definition 4.3.3 *A local valley is a nonempty connected component of some strict sublevel set* $L_\alpha^s := \{(U,V) \mid \Phi(U,V) < \alpha\}$. *A bad local valley is a local valley on which the training loss* Φ *cannot be made arbitrarily close to* p^*.

We are now ready the state the main result of this chapter. A solution is called to have zero training error if it can realize a pre-defined target output $Y \in \mathbb{R}^{N\times m}$ at the output layer of the network.

Theorem 4.3.4 *The following holds under Assumption 4.3.1 and Assumption 1.1.1:*

1. *There exist uncountably many solutions with zero training error.*
2. *The loss surface of* Φ *does not have any bad local valley.*
3. *There exists no suboptimal strict local minimum.*
4. *For cross-entropy loss* (1.1) *and square loss* (1.2) *there exists no local maximum.*

Proof:

1. Given an arbitrary target output $Y \in \mathbb{R}^{N\times m}$. By Lemma 4.3.2 the set of U such that $\Psi(U)$ has not full rank N has measure zero. Pick a U such that Ψ has full rank, then the linear system $\Psi(U)V = Y$ has at least one solution V. As this is possible for almost all U, there exist uncountably many solutions achieving zero training error.
2. Let C be a non-empty, connected component of some strict sublevel set L_α^s where $\alpha > p^*$. Given any $\epsilon \in (p^*, \alpha)$, we will show that C always contains a point (U,V) s.t. $\Phi(U,V) \leq \epsilon$ as this would imply that the loss Φ restricted to C can attain any value arbitrarily close to p^*.

We note that the strict sublevel set $L^s_\alpha = \Phi^{-1}((-\infty, \alpha))$ is an open set according to Proposition 5.1.1. Since C is a non-empty connected component of L^s_α, C must also be an open set with non-zero Lebesgue measure. By Lemma 4.3.2 the set of U where $\Psi(U)$ has not full rank has measure zero and thus C must contain a point (U, V) such that $\Psi(U)$ has full rank. By Assumption 1.1.1, φ attains its infimum at $p^* < \epsilon$, and thus by continuity of φ, there exists $G^* \in \mathbb{R}^{N\times m}$ such that $p^* \leq \varphi(G^*) \leq \epsilon$. Since $\Psi(U)$ has full row rank, there always exist V^* such that $\Psi(U)V^* = G^*$. Now, one notes that the loss $\Phi(U, V) = \varphi(\Psi(U)V)$ is convex in V, and that $\Phi(U, V) < \alpha$, thus we have for the line segment $V(\lambda) = \lambda V + (1 - \lambda)V^*$ for $\lambda \in [0, 1]$,

$$\Phi(U, V(\lambda)) \leq \lambda\Phi(U, V) + (1 - \lambda)\Phi(U, V^*) < \lambda\alpha + (1 - \lambda)\epsilon < \alpha.$$

Thus the whole line segment from (U, V) to (U, V^*) is contained in L^s_α. Since C is a connected component of L^s_α which contains (U, V), it follows that $(U, V^*) \in C$. Moreover, one has $\Phi(U, V^*) = \varphi(G^*) \leq \epsilon$ and since ϵ can be chosen to be arbitrarily close to p^*, we conclude that the loss restricted to C can always be made arbitrarily close to p^*.

3. Let (U_0, V_0) be a strict suboptimal local minimum, then there exists $r > 0$ such that $\Phi(U, V) > \Phi(U_0, V_0) > p^*$ for all $(U, V) \in B((U_0, V_0), r) \setminus \{(U_0, V_0)\}$ where $B(\cdot, r)$ denotes a closed ball of radius r. Let $\alpha = \min_{(U,V)\in\partial B\big((U_0,V_0),r\big)} \Phi(U, V)$ which exists as Φ is continuous and the boundary $\partial B\big((U_0, V_0), r\big)$ of $B\big((U_0, V_0), r\big)$ is compact. Note that $\Phi(U_0, V_0) < \alpha$ as (U_0, V_0) is a strict local minimum, and thus $(U_0, V_0) \in L^s_\alpha$. Let E be the connected component of L^s_α which contains (U_0, V_0), that is, $(U_0, V_0) \in E \subseteq L^s_\alpha$. Since the loss of every point inside E is strictly smaller than α, whereas the loss of every point on the boundary $\partial B\big((U_0, V_0), r\big)$ is greater than or equal to α, E must be contained in the interior of the ball, that is $E \subset B\big((U_0, V_0), r\big)$. Moreover, $\Phi(U, V) \geq \Phi(U_0, V_0) > p^*$ for every $(U, V) \in E$ and thus the values of Φ restricted to E can not be arbitrarily close to p^*, meaning that E is a bad local valley, which contradicts the second statement of the theorem.

4. Case 1: cross-entropy loss.

$$\Phi(U, V) = \varphi(G) = \frac{1}{N}\sum_{i=1}^{N} -\log\Big(\frac{e^{G_{iy_i}}}{\sum_{k=1}^m e^{G_{ik}}}\Big) \text{ where } G = \Psi(U)V.$$

Suppose by contradiction that (U, V) is a local maximum. Then the Hessian of Φ is negative semi-definite. However, as principal submatrices of negative semi-definite matrices are again negative semi-definite, then also the Hessian of Φ w.r.t V must be negative semi-definite. However, Φ is always convex in V and thus its Hessian restricted to V is positive semi-definite. The only matrix which is both p.s.d. and n.s.d. is the zero matrix. It follows that $\nabla^2_V\Phi(U, V) = 0$. One can easily show that

$$\nabla^2_{V_{:j}}\Phi = \sum_{i=1}^{N} \frac{e^{G_{ij}}}{\sum_{k=1}^m e^{G_{ik}}}\Big(1 - \frac{e^{G_{ij}}}{\sum_{k=1}^m e^{G_{ik}}}\Big)\Psi_{i:}\Psi_{i:}^T$$

From Assumption 4.3.1 it holds that there exists $j \in [N]$ s.t. σ_{p_j} is strictly positive, and thus some entries of $\Psi_{i:}$ must be strictly positive. Moreover, one has $\frac{e^{G_{ij}}}{\sum_{k=1}^m e^{G_{ik}}} \in (0, 1)$. It follows that some entries of $\nabla^2_{V_{:j}}\Phi$ must be strictly positive. Thus $\nabla^2_{V_{:j}}\Phi$ cannot be identically zero, leading to a contradiction. Therefore Φ has no local maximum.

Case 2: square loss.

$$\Phi(U, V) = \frac{1}{2}\left\|\Psi(U)V - Y\right\|_F^2 = \frac{1}{2}\left\|(\mathbb{I}_m \otimes \Psi(U))\, vec(V) - vec(Y)\right\|_2^2$$

where $\otimes$ denotes Kronecker product, and $\mathbb{I}_m$ an $m \times m$ identity matrix. The hessian of Φ w.r.t. V is $\nabla^2_{vec(V)}\Phi = (\mathbb{I}_m \otimes \Psi(U))^T(\mathbb{I}_m \otimes \Psi(U))$. From here we use exactly the same argument as

above to derive that if (U, V) is a local maximum then $\Psi(U) = 0$ which leads to a contradiction due to Assumption 4.3.1. Therefore Φ has no local maximum.

□

Theorem 4.3.4 shows that there are infinitely many solutions which achieve zero training error, and the loss landscape is nice in the sense that from any point in the parameter space there exists a continuous path that drives the loss arbitrarily close to zero (and thus a solution with zero training error) on which the loss is non-increasing.

While the networks are over-parameterized, we show in the next Section 4.4 that the modification of standard networks so that they fulfill our conditions leads nevertheless to good generalization performance, often even better than the original network. We would like to note that the proof of Theorem 4.3.4 also suggests a different algorithm to achieve zero training error: one initializes all weights, except the weights to the output layer, randomly (e.g. Gaussian weights), denoted as U, and then just solves the linear system $\Psi(U)V = Y$ to obtain the weights V to the output layer. Basically, this algorithm uses the network as a random feature generator and fits the last layer directly to achieve zero training error. The algorithm is successful with probability 1 due to Lemma 4.3.2. Note that from a solution with zero training error one can drive the cross-entropy loss to zero by upscaling to infinity but this does not change the classifier. We will see, that this simple algorithm shows bad generalization performance and overfitting, whereas training the full network with SGD leads to good generalization performance. This might seem counter-intuitive as our networks have more parameters than the original networks but is in line with recent observations in Zhang et al. (2017) that state-of-the art networks, also heavily over-parameterized, can fit even random labels but still generalize well on the original problem. Due to this qualitative difference of SGD and the simple algorithm which both can find solutions with zero training error, we think that our class of networks is an ideal test bed to study the implicit regularization/bias of SGD, see e.g. Soudry et al. (2018).

4.4 Experiments

The main purpose of this section is to investigate the generalization ability of practical neural networks with skip-connections added to the output layer to fulfill Assumption 4.3.1.

Datasets. We consider MNIST and CIFAR10 datasets.

- MNIST Lecun et al. is a standard benchmark dataset which is widely used for image classification/recognition tasks in machine learning. The dataset contains 55000 training images and 10000 test images. Each image has size 28×28 pixel, and contains a handwritten digit ranging from 0 to 9, meaning that there are 10 classes in total.
- CIFAR10 Krizhevsky (2009) is yet another standard dataset in machine learning for image classification tasks. The dataset consists of 60000 32×32 color images in 10 classes, with 6000 images per class. Each class can be one of the following objects: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. The classes are completely mutually exclusive. For instance, there is no overlap between automobiles and trucks. There are 50000 training images and 10000 test images.

In all our experiments described below, we do not use any data pre-processing or data-augmentation techniques. However, some results with data-augmentation are available in the appendix.

Layer	Output size	#neurons
Input: 28×28	$28 \times 28 \times 1$	
3×3 conv $-$ 64, stride 1	$28 \times 28 \times 64$	50176
3×3 conv $-$ 64, stride 1	$28 \times 28 \times 64$	50176
3×3 conv $-$ 64, stride 2	$14 \times 14 \times 64$	12544
3×3 conv $-$ 128, stride 1	$14 \times 14 \times 128$	25088
3×3 conv $-$ 128, stride 1	$14 \times 14 \times 128$	25088
3×3 conv $-$ 128, stride 2	$7 \times 7 \times 128$	6272
3×3 conv $-$ 256, stride 1	$7 \times 7 \times 256$	12544
1×1 conv $-$ 256, stride 1	$7 \times 7 \times 256$	12544
3×3 conv $-$ 256, stride 2	$4 \times 4 \times 256$	4096
3×3 conv $-$ 256, stride 1	$4 \times 4 \times 256$	4096
3×3 conv $-$ 256, stride 2	$2 \times 2 \times 256$	1024
3×3 conv $-$ 256, stride 1	$2 \times 2 \times 256$	1024
3×3 conv $-$ 256, stride 2	$1 \times 1 \times 256$	256
Fully connected, 10 output units		

Table 4.1: The architecture of CNN13 for MNIST dataset, with 179,840 hidden neurons in total.

Network architectures. For MNIST, we use a plain CNN architecture with 13 layers, denoted as CNN13. We refer to Table 4.1 for the detailed description about this architecture. For CIFAR10 we use VGG11, VGG13, VGG16 (Simonyan and Zisserman, 2015) and DenseNet121 (Huang et al., 2017). As the VGG models were originally proposed for ImageNet and have very large fully connected layers, we adapted these layers for CIFAR10 by reducing their width from 4096 to 128. For each network, we create the corresponding skip-model by adding skip-connections to the output so that our condition $M \geq N$ from the main theorem is satisfied. In particular, we aggregate all neurons of all the hidden layers in a pool and randomly choose from there a subset of N neurons to connect to the output layer (see Figure 4.2 for an illustration). Since the above CNN architectures have a large number of feature maps per convolutional layer, the total number of neurons is often very large compared to number of training samples, and thus it is easy to choose from there a subset of N neurons to connect to the output. For each network and their skip-variants, we test both sigmoid and softplus activation function (from Equation (4.2) with $\gamma = 20$).

Training procedure. We train all the networks with the standard cross-entropy loss and SGD+Nesterov momentum for 300 epochs. The initial learning rate is set to 0.1 for Densenet121 and 0.01 for the other architectures. Following Huang et al. (2017), we also divide the learning rate by 10 after 50% and 75% of the total number of training epochs. Note that we do not use any explicit regularization like weight decay or dropout. We report the test accuracy for the original models and the ones with skip-connections to the output layer. For the latter ones we consider two different training algorithms: 1) standard SGD for training the full network as described above (**SGD**) and 2) the randomized procedure (**rand**). In the second approach, we randomly initialize the weights of the network U up to the output layer by drawing each of them from a truncated Gaussian distribution with zero mean and variance $\frac{2}{d}$ where d is the number of weight parameters and the truncation is done after ± 2 standard deviations (standard Keras initialization), and then use SGD to optimize the weights V for a linear classifier with fixed features $\Psi(U)$ which is a convex optimization problem.

4.4.1 Discussion of generalization performance

Our experimental results are summarized in Table 4.2 for MNIST and Table 4.3 for CIFAR10. For skip-models, we report mean and standard deviation over 8 random choices of the subset of N neurons connected to the output.

	Sigmoid activation function	Softplus activation function
CNN13	11.35	99.20
CNN13-skip (SGD)	98.40 ± 0.07	99.14 ± 0.04

Table 4.2: Test accuracy (%) of CNN13 on MNIST dataset. CNN13 denotes the original architecture from Table 4.1 while CNN13-skip denotes the corresponding skip-model. There are in total $179,840$ hidden neurons from the original CNN13 (see Table 4.1), out of which we choose a random subset of $N = 55,000$ neurons to connect to the output layer to obtain CNN13-skip.

	Sigmoid activation function		Softplus activation function	
Model	Test acc (%)	Train acc (%)	Test acc (%)	Train acc (%)
VGG11	10	10	78.92	100
VGG11-skip (rand)	62.81 ± 0.39	100	64.49 ± 0.38	100
VGG11-skip (SGD)	**72.51** ± 0.35	100	**80.57** ± 0.40	100
VGG13	10	10	80.84	100
VGG13-skip (rand)	61.50 ± 0.34	100	61.42 ± 0.40	100
VGG13-skip (SGD)	**70.24** ± 0.39	100	**81.94** ± 0.40	100
VGG16	10	10	81.33	100
VGG16-skip (rand)	61.57 ± 0.41	100	61.46 ± 0.34	100
VGG16-skip (SGD)	**70.61** ± 0.36	100	**81.91** ± 0.24	100
Densenet121	**86.41**	100	**89.31**	100
Densenet121-skip (rand)	52.07 ± 0.48	100	55.39 ± 0.48	100
Densenet121-skip (SGD)	81.47 ± 1.03	100	86.76 ± 0.49	100

Table 4.3: Training and test accuracy of several CNN architectures with/without skip-connections on CIFAR10 (no data-augmentation). For every A $\in$ {VGG11, VGG13, VGG16, Densenet121}, A-skip denotes the corresponding skip-model in which a subset of N randomly selected neurons are connected to the output units. For Densenet121, these neurons are randomly chosen from the first dense block. The names in open brackets (rand/SGD) specify how the networks are trained: **rand** (U is randomized and fixed while V is learned with SGD), **SGD** (both U and V are optimized with SGD). See Table 4.4 in the appendix for the corresponding results with data-augmentation.

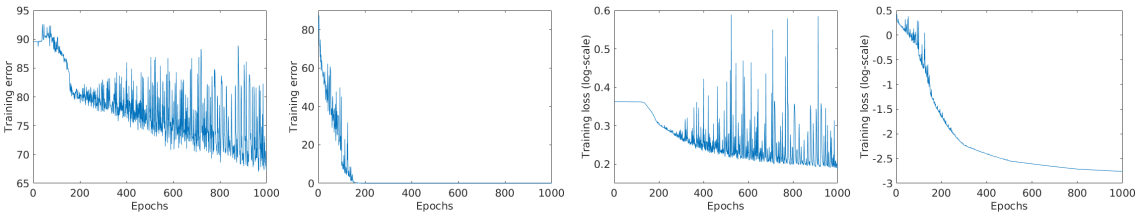
First of all, we note that adding skip connections to the output improves the test accuracy in almost all networks (with the exception of Densenet121) when the full network is trained with SGD. In particular, for the sigmoid activation function the skip connections allow for all models except Densenet121 to get reasonable performance whereas training the original model fails. This effect can be directly related to our result of Theorem 4.3.4 that the loss landscape of skip-networks has no bad local valley and thus it is not difficult to reach a solution with zero training error (see Section 4.4.2 for more detailed discussions on this issue, as well as Section 4.4.3 for a visual example which shows why the skip-models can succeed while the original models fail). The exception is Densenet121 which gets already good performance for the sigmoid activation function for the original model. We think that the reason is that the original Densenet121 architecture has already quite a lot of skip-connections between the hidden layers which thus improves the loss surface already so that the additional connections added to the output units are not necessary anymore.

The second interesting observation is that we do not see any sign of overfitting for the SGD version even though we have increased for all models the number of parameters by adding skip connections to the output layer and we know from Theorem 4.3.4 that for all the skip-models one can easily achieve zero training error. This is in line with the recent observation of Zhang et al. (2017) that

modern heavily over-parameterized networks can fit everything (random labels, random input) but nevertheless generalize well on the original training data when trained with SGD. This is currently an active research area to show that SGD has some implicit bias (Neyshabur et al., 2017; Brutzkus et al., 2018; Soudry et al., 2018) which leads to a kind of regularization effect similar to the linear least squares problem where SGD converges to the minimum norm solution. Our results confirm that there is an implicit bias as we see a strong contrast to the (skip-rand) results obtained by using the network as a random feature generator and just fitting the connections to the output units (*i.e.* V) which also leads to solutions with zero training error with probability 1 as shown in Lemma 4.3.2 and the proof of Theorem 4.3.4. For this version we see that the test accuracy gets worse as one is moving from simpler networks (VGG11) to more complex ones (VGG16 and Densenet121) which is a sign of overfitting. Thus we think that our class of networks is also an interesting test bed to understand the implicit regularization effect of SGD. It seems that SGD selects from the infinite pool of solutions with zero training error one which generalizes well, whereas the randomized feature generator selects one with much worse generalization performance.

4.4.2 Discussion of training performance

As shown in Table 4.3, the training error is zero in all cases, except when the original VGG models are used with sigmoid activation function. The reason, as noticed in our experiments, is because the learning of these sigmoidal networks converges quickly to a constant zero classifier (i.e. the output of the last hidden layer converges to zero), which makes both training and test accuracy converge to 10% and the loss in Equation (1.1) converges to $-\log(1/10)$. While we are not aware of a theoretical explanation for this behavior, it is not restricted to the specific architecture of VGGs but hold in general for plain sigmoidal networks with depth>5 as pointed out earlier by Glorot and Bengio (2010). As shown in Table 4.3, Densenets however do not suffer from this phenomenon, probably because they already have skip-connections between all the hidden layers of a dense block, thus gradients can easily flow from the output to every layer of a dense block, which makes the training of this network with sigmoid activation function become feasible.



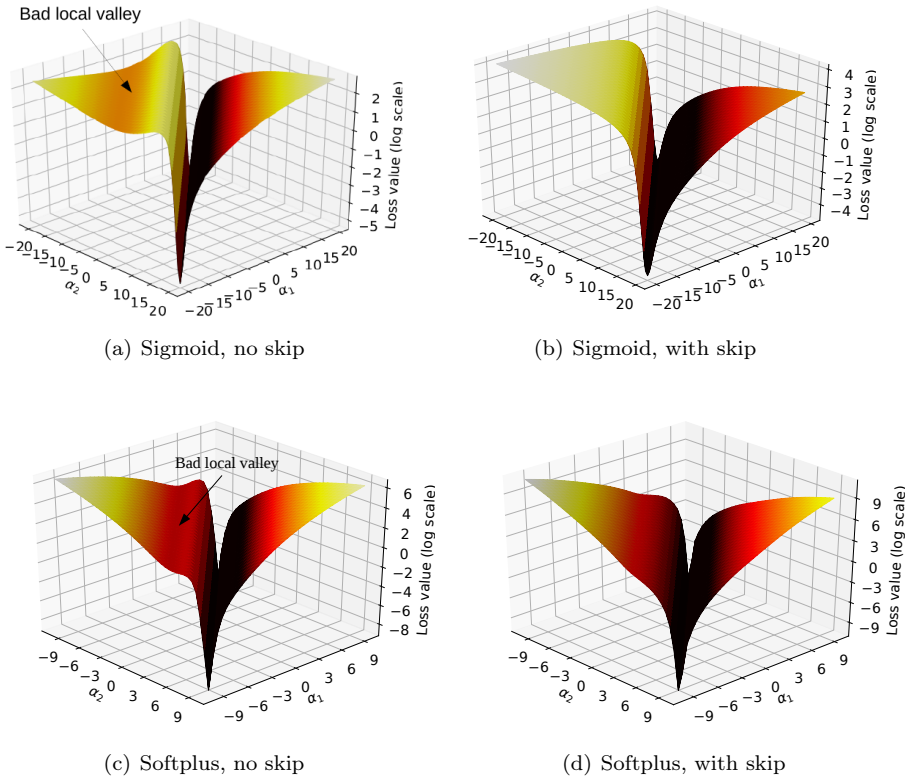
(a) Softplus, no skip (b) Softplus, with skip (c) Softplus, no skip (d) Softplus, with skip

Figure 4.3: Training progress of a 150-layer neural network with and without skip-connections. a),b) plots of training error and c),d) plots of training loss.

Skip-connections are helpful for training extremely deep neural networks. Our experimental results from Table 4.3 have shown that skip-connections are helpful for training deep sigmoidal networks. In this part, we show a similar result for softplus activation function. For the purpose of illustration, we create a small dataset with $N = 1000$ training images randomly chosen from CIFAR10 dataset. We use a very deep network with 150 fully connected layers, each of width 10, and softplus activation. A skip-model is created by adding skip-connections from N randomly chosen neurons to the output units. We train both networks with SGD. The best learning rate for each model is empirically chosen from $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. We report the training loss and training error of both models in Figure 4.3. One can see that the skip-network easily converge to zero training error

within 200 epochs, whereas the original network has stronger fluctuations and fails to converge after 1000 epochs. This is directly related to our result of Theorem 4.3.4 in the sense that skip-connections can help to smooth the loss landscape and enable effective training of very deep networks.

4.4.3 Visualization of the loss surface



(a) Sigmoid, no skip

(b) Sigmoid, with skip

(c) Softplus, no skip

(d) Softplus, with skip

Figure 4.4: Loss surface of a two-hidden-layer neural network with and without skip-connections on a small MNIST dataset with cross-entropy loss and sigmoid/softplus activation function.

Similar to Li et al. (2018); Goodfellow et al. (2015), we visualize the loss surface restricted to a two dimensional subspace of the parameter space. The subspace is chosen to go through some point (U_0, V_0) learned by SGD and spanned by two random directions (U_1, V_1) and (U_2, V_2). For the purpose of illustration, we train with SGD a two-hidden-layer fully connected network with 784 and 300 hidden units respectively, followed by a 10-way softmax classifier. The training set consists of 1024 images, which are randomly selected from MNIST dataset. After adding skip-connections to the output, the network fulfills $M = N = 1024$. Figure 4.4 shows the heat map of the loss surface before and after adding skip-connections. One can see a visible effect that skip-connections helps to smooth the loss landscape near a small sub-optimal region and allows gradient descent to flow directly from there to the lower regions of the landscape with smaller objective value.

4.5 Summary

We have identified a class of deep neural networks whose loss landscape has no bad local valleys. While our networks are over-parameterized and can easily achieve zero training error, they generalize

well in practice when trained with SGD. Interestingly, a simple different algorithm using the network as random feature generator also achieves zero training error but has significantly worse generalization performance. Thus we think that our class of networks is an interesting test bed for studying the implicit regularization effect of SGD.

4.6 Appendix

4.6.1 Data-augmentation results for Table 4.3

The following Table 4.4 shows additional results to Table 4.3 where data-augmentation is used now. For data-augmentation, we follow the procedure as described in (Zagoruyko and Komodakis, 2016) by considering random crops of size 32×32 after 4 pixel padding on each side of the training images and random horizontal flips with probability 0.5. For the convenience of the reader, we also repeat the results of Table 4.3 in the new Table 4.4.

	Sigmoid activation function		**Softplus activation function**	
Model	C-10	C-10$^+$	C-10	C-10$^+$
VGG11	10	10	78.92	88.62
VGG11-skip (rand)	62.81 ± 0.39	-	64.49 ± 0.38	-
VGG11-skip (SGD)	$\mathbf{72.51} \pm 0.35$	$\mathbf{85.55} \pm 0.09$	$\mathbf{80.57} \pm 0.40$	$\mathbf{89.32} \pm 0.16$
VGG13	10	10	80.84	90.58
VGG13-skip (rand)	61.50 ± 0.34	-	61.42 ± 0.40	-
VGG13-skip (SGD)	$\mathbf{70.24} \pm 0.39$	$\mathbf{86.48} \pm 0.32$	$\mathbf{81.94} \pm 0.40$	$\mathbf{91.06} \pm 0.12$
VGG16	10	10	81.33	90.68
VGG16-skip (rand)	61.57 ± 0.41	-	61.46 ± 0.34	-
VGG16-skip (SGD)	$\mathbf{70.61} \pm 0.36$	$\mathbf{86.42} \pm 0.31$	$\mathbf{81.91} \pm 0.24$	$\mathbf{91.00} \pm 0.22$
Densenet121	**86.41**	**90.93**	**89.31**	**94.20**
Densenet121-skip (rand)	52.07 ± 0.48	-	55.39 ± 0.48	-
Densenet121-skip (SGD)	81.47 ± 1.03	90.32 ± 0.50	86.76 ± 0.49	93.23 ± 0.42

Table 4.4: Test accuracy (%) of several CNN architectures with/without skip-connections on CIFAR10 ($^+$ denotes data augmentation). All other notations are similar to Table 4.3.

4.6.2 Additional experiments: how did we deal with pooling layers?

The original VGG Simonyan and Zisserman (2015) and Densenet Huang et al. (2017) contain pooling layers in their architecture. In particular, original VGGs have max-pooling layers, and original Densenets have averaging pooling layers. In the following, we will clarify how/if these pooling layers have been used in our experiments in Table 4.3, and whether and how our theretical results are appicable to this case, as well as presenting additional experimental results in this regard.

First of all, we note that Densenets Huang et al. (2017) contain pooling layers only after the first dense block. Meanwhile, as noted in Table 4.3, our experiments with Densenets only use skip-connections from hidden units of the first dense block, and thus Lemma 4.3.2 and Theorem 4.3.4 are applicable. The reason is that one can restrict the full-rank analysis of matrix Ψ in Lemma 4.3.2 to the hidden units of the first dense block, so that it follows that the set of parameters of the first dense block where Ψ has not full rank has measure zero, from which the results of Theorem 4.3.4 follow immediately.

However for VGGs in Table 4.3, we kept their max-pooling layers similar to the original architecture as we wanted to have a fair comparison between our skip-models and the original models. In this setting, our results are not directly applicable because we lose the analytic property of the entries of Ψ w.r.t. its dependent parameters, which is crucial to prove Lemma 4.3.2. Therefore in this section, we would like to present additional results to Table 4.3 in which we replace all max-pooling layers of all VGG models from Table 4.3 with 2x2 convolutional layers of stride 2. In this case, the whole network consists of just convolutional/fully connected layers, hence our theoretical results hold.

	Sigmoid activation function		**Softplus activation function**	
Model	C-10	C-10$^+$	C-10	C-10$^+$
VGG11-mp2conv	10	10	74.53	88.80
VGG11-mp2conv-skip (rand)	53.65 ± 0.66	-	55.51 ± 0.43	-
VGG11-mp2conv-skip (SGD)	**64.45** ± 0.31	**80.15** ± 0.59	**76.18** ± 0.58	**89.93** ± 0.19
VGG13-mp2conv	10	10	74.04	90.37
VGG13-mp2conv-skip (rand)	53.45 ± 0.23	-	53.33 ± 0.67	-
VGG13-mp2conv-skip (SGD)	**63.53** ± 0.37	**82.40** ± 0.23	**75.58** ± 0.77	**91.04** ± 0.20
VGG16-mp2conv	10	10	74.00	90.38
VGG16-mp2conv-skip (rand)	53.83 ± 0.30	-	55.34 ± 0.64	-
VGG16-mp2conv-skip (SGD)	**65.77** ± 0.67	**83.06** ± 0.34	**76.52** ± 0.78	**91.00** ± 0.23

Table 4.5: Test accuracy (%) of VGG networks from Table 4.3 where max-pooling layers are replaced by 2x2 convolutional layers of stride 2 (denoted as mp2conv).

The experimental results are presented in Table 4.5. Overall, our main observations are similar as before. The performance gap between original models and their corresponding skip-variants are approximately the same as in Table 4.3 or slightly more pronounced in some cases. A one-to-one comparison with Table 4.3 also shows that the performance of skip-models themselves have decreased by $4-7\%$ after the replacement of max-pooling layers with 2x2 convolutional layers. This is perhaps not so surprising because the problem gets potentially harder when the network has more layers to be learned, especially in case of sigmoid activation where the decrease is sharper. Similar to Table 4.3, adding skip-connections to the output units still prove to be very helpful – it improves the result for softplus while making the training of deep networks with sigmoid activation become possible at all. Finally, the training of full network with SGD still yields significantly better solutions in terms of generalization error than the random feature approach. This confirms once again the implicit bias of SGD towards high quality solutions among infinitely many solutions with zero training error.

Chapter 5

Connectivity of decision regions of deep neural networks

In Chapter 2 and Chapter 3, we have shown that the loss surface of neural networks is well-behaved if there is a wide hidden layer with more neurons than the number of training samples. This condition has been improved and brought closer to practice in Chapter 4 using random and flexible skip-connections to the output. All of these results so far have highlighted the important benefit of width in deep learning. In this chapter, we address the following opposite question:

What limitations that neural networks might have if none of their hidden layers is wide enough?

By focusing on standard classification tasks, we prove that if the network does not have any hidden layer with more neurons than the input dimension, then it can only learn connected decision regions – which refer to the subsets of the input space where the network predict a specific class. We further discuss the implication of this result on the adversarial manipulation problem of neural networks. All our main results of this chapter have been published at Nguyen et al. (2018).

5.1 Introduction

While deep learning has become state of the art in many application domains such as computer vision and natural language processing and speech recognition, the theoretical understanding of this success is steadily growing but there are still plenty of questions where there is little or no understanding. In particular, for the question how one should construct the network e.g. choice of activation function, number of layers, number of hidden units per layer etc., there is little guidance and only limited understanding on the implications of the choice e.g. "The design of hidden units is an extremely active area of research and does not yet have many definitive guiding theoretical principles." is a quote from the recent book on deep learning (Goodfellow et al., 2016, p. 191). Nevertheless there is recently progress in the understanding of these choices.

The first important results are the universal approximation theorems (Leshno et al., 1993; Cybenko, 1989; Hornik et al., 1989) which show that a single hidden layer network with non-polynomial activation functions can approximate every continuous function on any compact domain. In order to explain the success of deep learning, much of the recent effort has been spent on analyzing the representation power of neural networks from the perspective of depth (Delalleau and Bengio, 2011;

Telgarsky, 2016; Eldan and Shamir, 2016; Safran and Shamir, 2017; Yarotsky, 2016; Poggio et al., 2016; Liang and Srikant, 2017; Mhaskar and Poggio, 2016). Basically, they show that there exist functions that can be computed efficiently by deep networks of linear or polynomial size but require exponential size for shallow networks. To further highlight the power of depth, Montufar et al. (2014); Pascanu et al. (2014) show that the number of linear regions that a ReLU network can form in the input space grows exponentially with depth. Tighter bounds on the number of linear regions are later on developed by Arora et al. (2018a); Serra et al. (2018); Charisopoulos and Maragos (2018). Another measure of expressivity so-called trajectory length is proposed by Raghu et al. (2017). who show that the complexity of functions computed by the network on a curve in the input space also grows exponentially with depth.

While most of previous work can only show the existence of depth efficiency (*i.e.* there exist certain functions that can be efficiently represented by deep networks but not effectively represented or even approximated by shallow networks) but cannot show how often this holds for all functions of interest, Cohen et al. (2016) have taken the first step to address this problem. In particular, by studying a special type of networks called convolutional arithmetic circuits – also known as Sum-Product networks (Poon and Domingos, 2011), the authors show that besides a set of measure zero, all functions that can be realized by a deep network of polynomial size require exponential size in order to be realized, or even approximated by a shallow network. Later, Cohen and Shashua (2016) show that this property however no longer holds for convolutional rectifier networks, which represents so far the empirically most successful deep learning architecture in practice.

Unlike most of previous work which focuses on the power of depth, (Lu et al., 2017; Hanin and Sellke, 2017) have recently shown that neural networks with ReLU activation function have to be wide enough in order to have the universal approximation property as depth increases. In particular, the authors show that the class of continuous functions on a compact set cannot be arbitrarily well approximated by an arbitrarily deep network if the maximum width of the network is not larger than the input dimension d. Moreover, it has been shown recently, that the loss surface of fully connected networks (Nguyen and Hein, 2017) and for convolutional neural networks (Nguyen and Hein, 2018) is well behaved, in the sense that almost all local minima are global minima, if there exists a layer which has more hidden units than the number of training points.

In this chapter we study the question under which conditions on the network the decision regions of a neural network are connected respectively can potentially be disconnected. The decision region of a class is the subset of $\mathbb{R}^d$, where the network predicts this class. A similar study has been in Makhoul et al. (1989, 1990) for feedforward networks with threshold activation functions, where they show that the initial layer has to have width $d+1$ in order that one can get disconnected decision regions. On an empirical level it has recently been argued Fawzi et al. (2017) that the decision regions of the Caffe Network Jia et al. (2014) on ImageNet are connected. In this chapter we analyze feedforward networks with continuous activation functions as currently used in practice. We show in line with previous work that almost all networks which have a pyramidal structure up to the last hidden layer, that is the width of all hidden layers is smaller than the input dimension d, can only produce connected decision regions. We show that the result is tight by providing explicit counterexamples for the case $d+1$. We conclude that a guiding principle for the construction of neural networks should be that there is a layer which is wider than the input dimension as it would be a strong assumption that the Bayes optimal classifier must have connected decision regions. Interestingly, our result holds for leaky ReLU, that is $\sigma(t) = \max(t, \alpha t)$ for $0 < \alpha < 1$, whereas the result of Hanin and Sellke (2017) is for ReLU, that is $\sigma(t) = \max(t, 0)$, but "the generalization is not straightforward, even for activations of the form $\sigma(t) = \max(l_1(t), l_2(t))$, where l_1, l_2 are affine functions with different slopes." We discuss also the implications of connected decision regions regarding the generation of adversarial samples, which will provide another argument in favor of larger width for neural network architectures.

We consider in the following activation functions $\sigma : \mathbb{R} \to \mathbb{R}$ which are continuous and strictly monotonically increasing. This is true for most of proposed activation functions, but does not hold

for ReLU, $\sigma(t) = \max(t, 0)$. On the other hand, it has been argued in the recent literature, that the following variants are to be preferred over ReLU as they deal better with the vanishing gradient problem and outperform ReLU in prediction performance He et al. (2015); Clevert et al. (2016). This is the leaky ReLU Maas et al. (2013) defined as $\sigma(t) = \max(t, \alpha t) \quad \text{for} \quad 0 < \alpha < 1$, where typically α is fixed, but it has also been optimized together with the network weights He et al. (2015) and ELU (exponential linear unit) (Clevert et al., 2016) given as

$$\sigma(t) = \begin{cases} e^t - 1 & t < 0 \\ t & t \geq 0. \end{cases}.$$

Note that image of the activation function σ, $\sigma(\mathbb{R}) = \{\sigma(t) \,|\, t \in \mathbb{R}\}$, is equal to $\mathbb{R}$ for leaky ReLU and $(-1, \infty)$ for the exponential linear unit.

In the following, we introduce some basic definitions and terminologies used in this chapter. For a function $f : U \to V$, where $\text{dom}(f) = U \subseteq \mathbb{R}^m$ and $V \subseteq \mathbb{R}^n$, we denote for every subset $A \subseteq U$, the image $f(A)$ as $f(A) := \{f(x) \mid x \in A\} = \bigcup_{x \in A} f(x)$. Let $\text{range}(f) = f(U)$.

We recall the following standard result from topology (see *e.g.* Apostol, 1974, Theorem 4.23, p. 82).

Proposition 5.1.1 *Let $f : \mathbb{R}^m \to \mathbb{R}^n$ be a continuous function. If $U \subseteq \mathbb{R}^n$ is an open set then $f^{-1}(U)$ is also open.*

We now recall a standard result from calculus showing that under certain, restricted conditions the inverse of a continuous mapping exists and is as well continuous.

Proposition 5.1.2 *Let $f : \mathbb{R} \to f(\mathbb{R})$ be continuous and strictly monotonically increasing. Then the inverse mapping $f^{-1} : f(\mathbb{R}) \to \mathbb{R}$ exists and is continuous.*

5.2 Main results

We first recall some basic notations of feedfoward neural networks from Section 1.1 Let L be the number of layers, d the input dimension, m the output dimension and n_k the width of layer k. By convention we assume that $n_0 = d$ and $n_L = m$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be some activation function which we will specify later. Let $W_k \in \mathbb{R}^{n_{k-1} \times n_k}$ and $b_k \in \mathbb{R}^{n_k}$ be the weight matrix and bias vector of layer k. The function $f_k : \mathbb{R}^d \to \mathbb{R}^{n_k}$ which maps every input $x \in \mathbb{R}^d$ to the output at layer k is defined as

$$f_k(x) = \begin{cases} x & k = 0 \\ \sigma(W_k^T f_{k-1}(x) + b_k) & k \in [1, L-1] \\ W_L^T f_{L-1}(x) + b_L & k = L. \end{cases}$$

Next, we define the deicison region of a class.

Definition 5.2.1 (Decision region) *The decision region of a given class $1 \leq j \leq m$, denoted by C_j, is defined as $C_j = \left\{x \in \mathbb{R}^d \;\middle|\; (f_L)_j(x) > (f_L)_k(x),\ \forall k \neq j\right\}$.*

The following lemma will be helpful to derive our results. Basically it shows that the pre-image of an open connected subset under the composition of an affine transformation and an elementwise nonlinear function is again open and connected. This later on will allow us to transfer the idea easily to multiple layer networks in a recursive manner.

Lemma 5.2.2 *Let $m \geq n$ and $f : \mathbb{R}^m \to \mathbb{R}^n$ be a function defined as $f = \hat{\sigma} \circ h$ where $\hat{\sigma} : \mathbb{R}^n \to \mathbb{R}^n$ is defined as*

$$\hat{\sigma}(x) = \begin{pmatrix} \sigma(x_1) \\ \vdots \\ \sigma(x_n) \end{pmatrix}, \tag{5.1}$$

and $\sigma : \mathbb{R} \to \mathbb{R}$ is bijective, continuous and $\sigma(\mathbb{R}) = \mathbb{R}$, and $h : \mathbb{R}^m \to \mathbb{R}^n$ is a linear map defined as $h(x) = W^T x + b$ where $W \in \mathbb{R}^{m \times n}$ has full rank and $b \in \mathbb{R}^n$. Let $V \subseteq \mathbb{R}^n$ be an open connected set. Then $f^{-1}(V) \subseteq \mathbb{R}^m$ is an open connected set.

Proof: By Proposition 3.3.5, it holds that $f^{-1}(V) = h^{-1}(\hat{\sigma}^{-1}(V))$. As $\hat{\sigma}$ is a componentwise function, the inverse mapping $\hat{\sigma}^{-1}$ is given by the inverse mappings of the components

$$\hat{\sigma}^{-1} : \mathbb{R}^n \to \mathbb{R}^n, \ \hat{\sigma}^{-1}(x) = \begin{pmatrix} \sigma^{-1}(x_1) \\ \vdots \\ \sigma^{-1}(x_n) \end{pmatrix},$$

where under the stated assumptions the inverse mapping $\sigma^{-1} : \mathbb{R} \to \mathbb{R}$ exists by Lemma 5.1.2 and is continuous. Since $V \subseteq \mathbb{R}^n = \mathrm{dom}(\hat{\sigma}^{-1})$, $\hat{\sigma}^{-1}(V)$ is the image of the connected set V under the continuous map $\hat{\sigma}^{-1}$. Thus by Proposition 3.3.2, $\hat{\sigma}^{-1}(V)$ is connected. Moreover, $\hat{\sigma}^{-1}(V)$ is an open set by Proposition 5.1.1. It holds for every $y \in \mathbb{R}^n$ that

$$h^{-1}(y) = \begin{cases} \emptyset & y \notin \mathrm{range}(h) \\ W(W^T W)^{-1}(y - b) + \ker(W^T) & y \in \mathrm{range}(h), \end{cases}$$

where the inverse of $W^T W$ exists as W has full rank n (note that we assume $n \leq m$). As W has full rank and $m \geq n$, it holds that $\mathrm{range}(h) = \mathbb{R}^n$ and thus

$$h^{-1}(y) = W(W^T W)^{-1}(y - b) + \ker(W^T), \quad \forall\, y \in \mathbb{R}^n.$$

Therefore it holds for $\hat{\sigma}^{-1}(V) \subseteq \mathbb{R}^n$ that

$$h^{-1}\big(\hat{\sigma}^{-1}(V)\big) = W(W^T W)^{-1}\big(\hat{\sigma}^{-1}(V) - b\big) + \ker(W^T),$$

where the first term is the image of the connected set $\hat{\sigma}^{-1}(V)$ under an affine mapping and thus is again connected by Proposition 3.3.2, the second term $\ker(W^T)$ is a linear subspace which is also connected. By Proposition 3.3.3, the Minkowski sum of two connected sets is connected. Thus $f^{-1}(V) = h^{-1}(\hat{\sigma}^{-1}(V))$ is a connected set. Moreover, as $f^{-1}(V)$ is the pre-image of the open set V under the continuous function f, it must be also an open set by Proposition 5.1.1. Thus $f^{-1}(V)$ is an open and connected set. □

Note that in Lemma 5.2.2, if $m < n$ and W has full rank then $\mathrm{range}(h) \subsetneq \mathbb{R}^n$ and the linear equation $h(x) = y$ has a unique solution $x = (WW^T)^{-1}W(y - b)$ for every $y \in \mathrm{range}(h)$ and thus

$$\begin{aligned} f^{-1}(V) &= h^{-1}\big(\sigma^{-1}(V)\big) \\ &= h^{-1}\big(\sigma^{-1}(V) \cap \mathrm{range}(h)\big) \\ &= (WW^T)^{-1}W\big((\sigma^{-1}(V) \cap \mathrm{range}(h)) - b\big). \end{aligned}$$

In this case, even though $\sigma^{-1}(V)$ is a connected set, the intersection $\sigma^{-1}(V) \cap \mathrm{range}(h)$ can be disconnected which can imply that $f^{-1}(V)$ is disconnected and thus the decision region becomes disconnected. We illustrate this with a simple example, where $m = 1$ and $n = 2$ with $\sigma(x) = x^3$ and $W^T = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$ and $b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. In this case it holds that

$$f(x) = \hat{\sigma}(W^T x + b) = \begin{pmatrix} \sigma(-x) \\ \sigma(x) \end{pmatrix} = \begin{pmatrix} -x^3 \\ x^3 \end{pmatrix}. \tag{5.2}$$

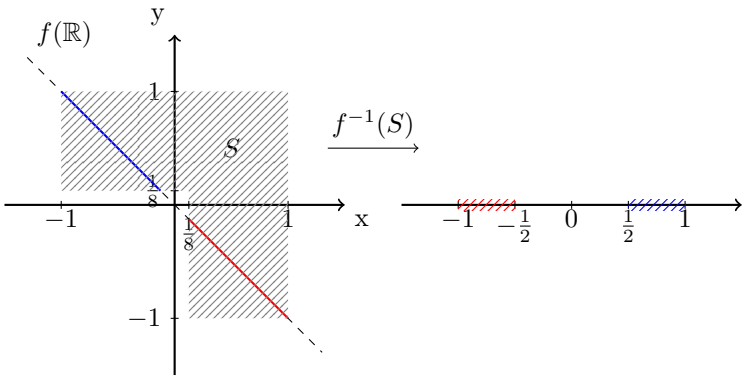


Figure 5.1: Left: The image of $\mathbb{R}$ under the mapping f, denoted as $f(\mathbb{R}) \subset \mathbb{R}^2$ for the toy example from Equation (5.2) which maps into a lower-dimensional subspace (the diagonal line). Right: The pre-image $f^{-1}(S) \subset \mathbb{R}$ of the connected S becomes disconnected.

Figure 5.1 shows that $f(\mathbb{R})$ is a one-dimensional submanifold (in this case subspace) of $\mathbb{R}^2$ and provides an example of a set $S \subset \mathbb{R}^2$ where the pre-image $f^{-1}(S)$ is disconnected.

We are now ready to state our main result which shows that the decisions regions of a deep neural network with pyramidal structure and have maximal width at most the input dimension d can only produce connected decision regions. We assume for the activation function that $\sigma(\mathbb{R}) = \mathbb{R}$, which is fulfilled by leaky ReLU.

Theorem 5.2.3 *Let the width of the layers of the feedforward network network satisfy $d = n_0 \geq n_1 \geq \ldots \geq n_{L-1}$ and let $\sigma : \mathbb{R} \to \mathbb{R}$ be a continuous and bijective activation function with $\sigma(\mathbb{R}) = \mathbb{R}$ and all the weight matrices $(W_l)_{l=1}^{L-1}$ have full rank. Then every decision region C_j is an open and connected subset of $\mathbb{R}^d$ for every $1 \leq j \leq m$.*

Proof: From Definition 5.2.1, it holds for every $1 \leq j \leq m$

$$C_j = \left\{ x \in \mathbb{R}^d \;\middle|\; f_{Lj}(x) - f_{Lk}(x) > 0, \forall k \neq j \right\}$$

where $f_{Lj}(x) - f_{Lk}(x) = \langle (W_L)_{:j} - (W_L)_{:k}, f_{L-1}(x) \rangle + (b_L)_j - (b_L)_k$. Let us define the set

$$V_j = \left\{ y \;\middle|\; \langle (W_L)_{:j} - (W_L)_{:k}, y \rangle > (b_L)_k - (b_L)_j, \forall k \neq j \right\}$$

then it holds $C_j = \left\{ x \in \mathbb{R}^d \;\middle|\; f_{L-1}(x) \in V_j \right\} = f_{L-1}^{-1}(V_j)$. If V_j is an empty set then we are done, otherwise one observes that V_j is the intersection of a finite number of open half-spaces (or the whole space), which is thus an open and connected set. Moreover, it holds $V_j \cap \hat{\sigma}(\mathbb{R}) = V_j$, where $\hat{\sigma}$ is defined as in (5.1). It follows from Proposition 5.1.1 that C_j must be an open set as it is the pre-image of the open set V_j under the continuous mapping f_{L-1}. To show that C_j is a connected set, one first observes that

$$f_{L-1} = \hat{\sigma} \circ h_{L-1} \circ \hat{\sigma} \circ h_{L-2} \ldots \circ \hat{\sigma} \circ h_1$$

where $h_k : \mathbb{R}^{n_{k-1}} \times \mathbb{R}^{n_k}$ is an affine mapping between layer $k-1$ and layer k defined as $h_k(x) = W_k^T x + b_k$ for every $1 \leq k \leq L-1, x \in \mathbb{R}^{n_{k-1}}$, and $\hat{\sigma} : \mathbb{R}^{n_k} \to \mathbb{R}^{n_k}$ is the activation mapping of layer k defined as in (5.1). By Proposition 3.3.5 it holds that

$$f_{L-1}^{-1}(V_j) = (h_1^{-1} \circ \hat{\sigma}^{-1} \circ \ldots \circ h_{L-1}^{-1} \circ \hat{\sigma}^{-1})(V_j)$$

Since $\sigma : \mathbb{R} \to \mathbb{R}$ is a continuous bijection by our assumption, it follows that $\hat{\sigma} : \mathbb{R}^{n_k} \to \mathbb{R}^{n_k}$ is also a continuous bijection. Moreover, it holds that W_k has full rank and $n_{k-1} \geq n_k$ for every $1 \leq k \leq L-1$

and V_j is a connected set. Thus one can apply Lemma 5.2.2 subsequently for the composed functions $(\hat{\sigma} \circ h_k)$ for every $k = L-1, L-2, \ldots, 1$ and obtains that $C_j = f_{L-1}^{-1}(V_j)$ is a connected set. Thus C_j is an open and connected set for every $1 \leq j \leq m$. □

The next theorem holds just for networks with one hidden layer but allows general activation functions which are continuous and strictly monotonically increasing, that is leaky ReLU, ELU, softplus or sigmoid activation functions. Again the decision regions are connected if the hidden layer has maximal width smaller than $d+1$.

Theorem 5.2.4 *Let the one hidden layer network satisfy $d = n_0 \geq n_1$ and let $\sigma : \mathbb{R} \to \mathbb{R}$ be a continuous, strictly monotonically increasing function and the hidden layer's weight matrix W_1 has full rank. Then every decision region C_j is an open connected subset of $\mathbb{R}^d$ for every $1 \leq j \leq m$.*

Proof: We note that in the proof of Theorem 5.2.3 the V_j is a finite intersection of open half-spaces and thus a convex set. Moreover, $\hat{\sigma}(\mathbb{R}^{n_1})$ is an open convex set (it is just an axis-aligned open box), as σ is strictly monotonically increasing. Thus it holds

$$C_j = \left\{ x \in \mathbb{R}^d \;\middle|\; f_1(x) \in V_j \cap \hat{\sigma}(\mathbb{R}^{n_1}) \right\} = f_1^{-1}\big(V_j \cap \hat{\sigma}(\mathbb{R}^{n_1})\big).$$

As both sets are open convex sets, the intersection $V_j \cap \hat{\sigma}(\mathbb{R}^{n_1})$ is again convex and open as well. Thus $V_j \cap \hat{\sigma}(\mathbb{R}^{n_1})$ is a connected set. The rest of the argument follows then by using Lemma 5.2.2, noting that by Proposition 5.1.2 $\hat{\sigma}^{-1} : \hat{\sigma}(\mathbb{R}^{n_1}) \to \mathbb{R}^{n_1}$ is a continuous mapping. □

Note that Theorem 5.2.3 and Theorem 5.2.4 make no assumption on the structure of all layers in the network. Thus they can be applied to neural networks with both fully connected layers and convolutional layers. Moreover, the results hold regardless of how the parameters of the network $(W_l, b_l)_{l=1}^L$ have been attained, trained or otherwise, as long as all the weight matrices of hidden layers have full rank. This is a quite weak condition in practice as the set of low rank matrices has just Lebesgue measure zero. Even if the optimal weight parameters for the data generating distribution would be low rank (we discuss such an example below), then it is very unlikely that the trained weight parameters are low rank, as one has statistical noise by the training sample, "optimization noise" from the usage of stochastic gradient descent (SGD) and its variants and finally in practice one often uses early stopping and thus even if the optimal solution for the training set is low rank, one will not find it.

Theorem 5.2.3 covers activation functions like leaky ReLU but not sigmoid, ELU or softplus. At the moment it is unclear for us if the result might hold also for the more general class of activation functions treated in Theorem 5.2.4. The problem is that then in Lemma 5.2.2 one has to compute the pre-image of $V \cap \hat{\sigma}(\mathbb{R}^n)$. Even though both sets are connected, the intersection of connected sets need not be connected. This is avoided in Theorem 5.2.4 by using that the initial set V_j and $\hat{\sigma}(\mathbb{R}^{n_{L-1}})$ are both convex and the intersection of convex sets is convex and thus connected.

We show below that the result is tight by giving an empirical example of a neural network with a single hidden layer of $d+1$ hidden units which produces disconnected regions. Note that our result complements the result of Hanin and Sellke (2017), where they show the universal approximation property (for ReLU) only if one considers networks of width at least $d+1$ for arbitrary depth. Theorem 5.2.3 and Theorem 5.2.4 indicate that this result could also hold for leaky ReLU as approximation of arbitrary functions implies approximation of arbitrary decisions regions, which clearly requires that one is able to get disconnected decision regions. Taking both results together, it seems rather obvious that as a general guiding principle for the construction of hidden layers in neural networks one should use, at least for the first hidden layer, more units than the input dimension, as it is rather unlikely that the Bayes optimal decision regions are connected. Indeed, if the true decision regions are disconnected then using a network of smaller width than $d+1$ might still perfectly fit the finite training data but since the learned decision regions are connected there exists a path between the

true decision regions which then can be used for potential adversarial manipulation. This is discussed in the next section where we show empirical evidence for the existence of such adversarial examples.

5.3 Discussions

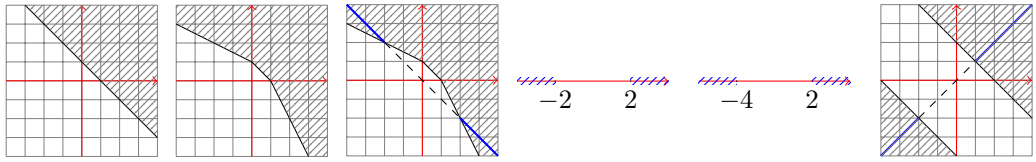
In this section we discuss with analytical examples as well as trained networks that the result is tight and the conditions of the theorem cannot be further relaxed. Moreover, we argue that connected decision regions can be problematic as they open up the possibility to generate adversarial examples.

5.3.1 Why pyramidal structure is a necessary condition?

In Theorem 5.2.3, if the network does not have pyramidal structure up to the last hidden layer, *i.e.* the condition $d_1 \geq \ldots \geq d_{L-1}$ is not fulfilled, then the statement of the theorem might not hold as the decision regions can be disconnected. We illustrate this via a counter-example below. Let us consider a non-pyramidal network with layer widths 2-1-2-2 defined as

$$W_3^T \hat{\sigma}(W_2^T \hat{\sigma}(W_1^T x + b_1) + b_2) + b_3 \tag{5.3}$$

where $\sigma(t) = \max(0.5\,t, t)$, and $W_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, b_1 = 0, W_2 = \begin{bmatrix} 1 & -1 \end{bmatrix}, b_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, W_3 = \begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix}, b_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. We can show that this network has disconnected decision regions. Indeed, the decision region of the



(a) V_1 (dashed) (b) P (dashed) (c) Q (blue) (d) R (dashed) (e) S (dashed) (f) C_1 (dashed)

Figure 5.2: Construction of the decision region C_1 for the example given in Equation (5.3).

first class C_1 can be computed recursively as

1. $V_1 := \left\{(y_1, y_2) \in \mathbb{R}^2 \;\middle|\; y_1 + y_2 - 1 > 0\right\}$
2. $P := \hat{\sigma}^{-1}(V_1)$
3. $Q := P \cap \text{range}(W_2^T)$
4. $R := (W_2 W_2^T)^{-1} W_2 (Q - b_2)$
5. $S := \hat{\sigma}^{-1}(R)$
6. $C_1 = W_1 (W_1^T W_1)^{-1} (S - b_1) + \ker(W_1^T)$

where the output at each step is illustrated in Figure 5.2. One can easily check that

$$C_1 = \left\{x \in \mathbb{R}^2 \;\middle|\; x_1 + x_2 - 2 > 0 \text{ and } x_1 + x_2 + 4 < 0\right\}$$

which is thus a disconnected set.

Overall, the above counter-example shows that the pyramidal structure of the network, provided that other conditions of Theorem 5.2.3, is a necessary condition to get connected decision regions.

5.3.2 Why full rank weight matrices is a necessary condition?

Similar to Section 5.3.1, we show that if the weight matrices of hidden layers are not full rank while the other conditions are still satisfied, then the decision regions can be disconnected. The reason is simply that low rank matrices, in particular in the first layer, reduce the effective dimension of the input. We illustrate this effect with a small analytical example and then argue that nevertheless in practice it is extremely difficult to get low rank weight matrices.

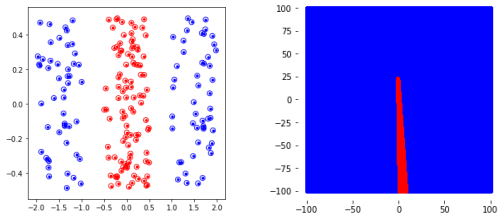


Figure 5.3: Left: A training dataset sampled from the distribution given in Equation (5.4). Right: Decision regions of a trained classifier for the example on the left, which are connected as the learned weight matrix W_1 has full rank.

Suppose that one has a two-class classification problem on $\mathbb{R}^2$ (see Figure 5.3) with equal class probabilities $P(\text{red}) = P(\text{blue})$, and the conditional distribution is given as

$$\begin{aligned} p(x_1, x_2|\text{blue}) &= \frac{1}{2}, \forall\, x_1 \in [-2,-1] \cup [1,2], x_2 \in [-\frac{1}{2}, \frac{1}{2}] \\ p(x_1, x_2|\text{red}) &= 1, \forall\, x_1 \in [-1,1], x_2 \in [-\frac{1}{2}, \frac{1}{2}]. \end{aligned} \tag{5.4}$$

Note that the Bayes optimal decision region for class blue is disconnected. Moreover, it is easy to verify that a one hidden layer network with leaky ReLU $\sigma(t) = \max(t, \alpha t)$ for $0 < \alpha < 1$ can perfectly fit the data with

$$W_1^T = \begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix}, b_1 = \begin{pmatrix} -1 \\ -1 \end{pmatrix}, W_2^T = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, b_2 = \begin{pmatrix} 0 \\ -2\alpha \end{pmatrix}$$

Note that W_1 has low rank. Suppose that the first output unit corresponds to the blue class and second output unit corresponds to the red class. Then it holds $(f_2)_{\text{red}}(x_1, x_2) = -2\alpha$, $(f_2)_{\text{blue}}(x_1, x_2) = \max(x_1 - 1, \alpha(x_1 - 1)) + \max(-(x_1 + 1), -\alpha(x_1 + 1))$ and thus

$$(f_2)_{\text{blue}}(x_1, x_2) = \begin{cases} (1-\alpha)x_1 - (1+\alpha) & x_1 \geq 1 \\ -2\alpha & -1 \leq x_1 \leq 1 \\ -(1-\alpha)x_1 - (1+\alpha) & x_1 \leq -1 \end{cases}$$

which implies that $(f_2)_{\text{blue}}(x_1, x_2) > (f_2)_{\text{red}}(x_1, x_2)$ for every $x_1 \in (-\infty, -1) \cup (1, +\infty)$ and thus the decision region for class blue has two disconnected decision regions. This implies that Theorems 5.2.3 and 5.2.4 do indeed not hold if the weight matrices do not have full rank. Nevertheless in practice, it is unlikely that one will get such low rank weight matrices, which we illustrate in Figure 5.3 that the decision regions of the trained classifier has indeed connected decision regions. This is due to statistical noise in the training set as well as through the noise in the optimization procedure (SGD) and the common practice of early stopping in training of neural networks.

5.3.3 Does the result hold for ReLU activation function?

As the conditions of Theorem 5.2.3 are not fulfilled for ReLU, one might ask whether the decision regions of a pyramidal network with full rank weight matrices can be potentially disconnected for

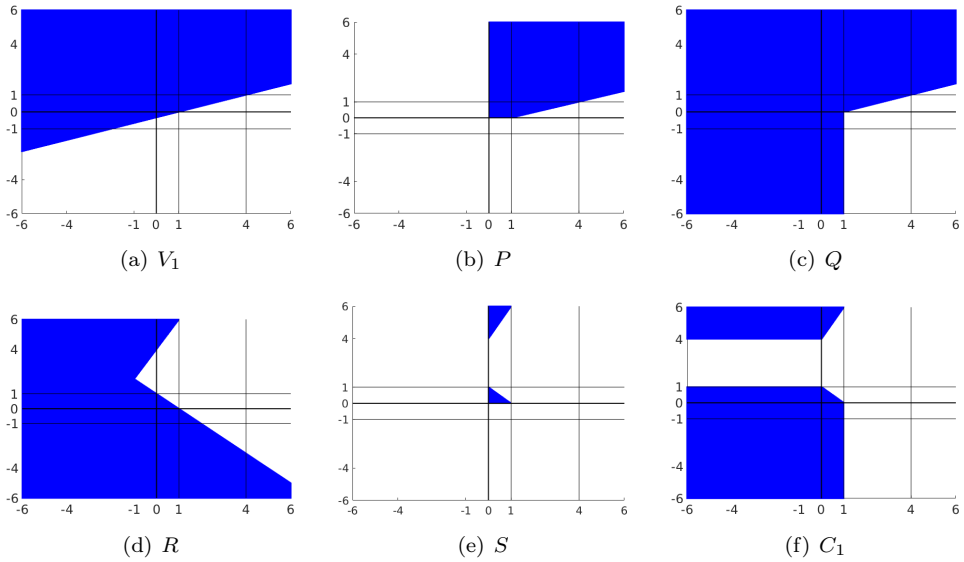
ReLU activation function. We show via the following example that this is indeed possible. Let a two hidden layer network with layer widths 2-2-2-2 be defined as

$$W_3^T \hat{\sigma}(W_2^T \hat{\sigma}(W_1^T x + b_1) + b_2) + b_3 \tag{5.5}$$

where $\sigma(t) = \max(0, t)$ and

$$W_1^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, W_2^T = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}, W_3^T = \begin{bmatrix} -1 & 0 \\ 0 & -3 \end{bmatrix},$$

and $b_1 = [0, 0]^T, b_2 = \frac{1}{\sqrt{2}}[\sqrt{2} - 1, -3]^T, b_3 = [1, 0]^T$. The decision region of C_1 is given recursively as



(a) V_1 (b) P (c) Q

(d) R (e) S (f) C_1

Figure 5.4: Output of each step in the computation of C_1 for the example given in Equation (5.5).

1. $V_1 := \{y \in \mathbb{R}^2 \mid (W_3^T y + b_3)_1 > (W_3^T y + b_3)_2\} = \{(y_1, y_2) \in \mathbb{R}^2 \mid y_1 - 3y_2 - 1 < 0\}$
2. $P := V_1 \cap \text{range}(\hat{\sigma}) = V_1 \cap \mathbb{R}^2_+$
3. $Q := \hat{\sigma}^{-1}(P)$
4. $R := (W_2^T)^{-1}(Q - b_2)$
5. $S := R \cap \text{range}(\hat{\sigma}) = R \cap \mathbb{R}^2_+$
6. $C_1 = \hat{\sigma}^{-1}(S)$

The output at each step is visualized in Figure 5.4. Note that the ReLU function $\sigma(t) = \max(t, 0)$ has an inverse, given by $\sigma^{-1}(t) = t$ for every $t > 0$ and $\sigma^{-1}(t) = \mathbb{R}_-$ for $t = 0$. By following the above steps, one can easily check that

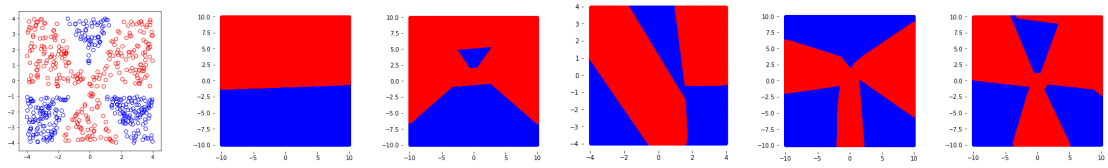
$$C_1 = \{x \in \mathbb{R}^2 \mid x_1 < 1,\ x_2 < 1,\ x_1 + x_2 < 1\} \cup \{x \in \mathbb{R}^2 \mid x_2 > 4,\ 2x_1 - x_2 + 4 < 0\}$$

which is a disconnected set as illustrated in Figure 5.4.

In this example, except for the activation function, all the other conditions of Theorem 5.2.3 are satisfied, that is, the network has pyramidal structure (2-2-2-2) and all the weight matrices $(W_l)_{l=1}^2$ have full rank. Thus the statement of Theorem 5.2.3 does not hold for ReLU.

5.3.4 Are our theorems tight in terms of number of hidden units?

We consider a binary classification task in $\mathbb{R}^2$ where the data points are generated so that the blue class has disconnected components on the square $[-4,4]\times[-4,4]$, see Figure 5.5 (a) for an illustration. We use a one hidden layer network with varying number of hidden units, two output units, leaky ReLU activation function and cross-entropy loss. We then train this network by using SGD with momentum for 1000 epochs and learning rate 0.1 and reduce the it by a factor of 2 after every 50 epochs. For all the attempts with different starting points that we have done in our experiment, the resulting weight matrices always have full rank. We show the training error and the decision regions



(a) Training data (b) $n_1 = 2(122)$ (c) $n_1 = 3(72)$ (d) $n_1 = 4(45)$ (e) $n_1 = 7(12)$ (f) $n_1 = 50(0)$

Figure 5.5: Decision region of a one hidden layer network trained with SGD for varying number of hidden units for the toy example as shown in (a). As shown by Theorem 5.2.3 the decision region for $n_1 = d = 2$ is connected, however already for $n_1 > d = 2$ one gets disconnected decision regions which shows that Theorem 5.2.3 is tight. The numbers in bracket show the number of misclassified training points.

of trained network in Figure 5.5. The grid size in each case of Figure 5.5 has been manually chosen so that one can see clearly the connected/disconnected components in the decision regions. First, we observe that for two hidden units ($n_1 = 2$), the network satisfies the condition of Theorem 5.2.3 and thus can only learn connected regions, which one can also clearly see in the figure, where one basically gets a linear separator. However, for three hidden units ($n_1 = 3$), one can see that the network can produce disconnected decision regions, which shows that both our Theorems 5.2.3 and 5.2.4 are tight, in the sense that width $d+1$ is already sufficient to produce disconnected components, whereas the results say that for width less than $d+1$ the decision regions have to be connected. As the number of hidden units increases, we observe that the network produces more easily disconnected decision regions as expected.

5.3.5 Relation to adversarial manipulation problems

It has been shown by Szegedy et al. (2014) that several machine learning models, including state-of-the-art deep neural networks are vulnerable to "adversarial examples". That is, it is possible to fool a trained classifier by manipulating a correctly classified input with an imperceptible perturbation so that the classifier now misclassifies this input with high confidence. Adversarial examples have the potential to be dangerous, especially as they enter critical safety systems such as autonomous vehicles. In this section, we show empirical evidence indicating that narrow neural networks which learn connected decision regions are susceptible to adversarial examples.

We use a single image of digit 1 from the MNIST dataset Lecun et al. to create a new artificial dataset where the underlying data generation probability measure has a similar one-dimensional structure as in (5.4) but now embedded in the pixel space $\mathbb{R}^{28\times 28}$. This is achieved by using rotation as the one-dimensional degree of freedom. We generate 2000 training images for each red/blue class by rotating the chosen digit 1 with angles ranging from $[-5°, 5°]$ for the read class, and $[-20°, -15°] \cup [15°, 20°]$ for the blue class, see Figure 5.6.

Figure 5.6: Training examples for the digit-1 dataset with two classes. The color red/blue denotes the class of the corresponding example.

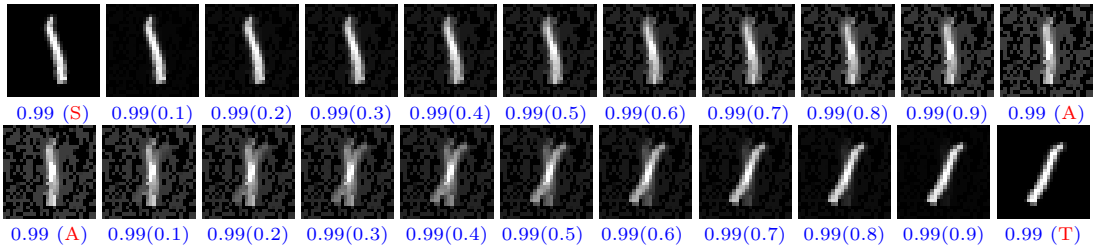


Figure 5.7: Digit-1 dataset with two classes. Top row: Trajectory from a source image (S) to an adversarial image (A) parameterized by λ (shown inside brackets). Bottom row: Trajectory from the adversarial image (A) to a target image (T). Each number outside bracket shows the confidence level (from 0 to 1) that the corresponding image was predicted to be in blue class.

Note that this is a binary classification task where the dataset has just one effective degree of freedom and the Bayes optimal decision regions are disconnected. We train a one hidden layer network with 784 hidden units which is equal to the input dimension and leaky ReLU as activation function with $\alpha = 0.1$. The training error is zero and the resulting weight matrices have full rank, thus the conditions of Theorem 5.2.3 are satisfied and the decision region of class blue should be connected even though the Bayes optimal decision region is disconnected. This can only happen by establishing a connection around the other red class. We test this by sampling a source image from the $[-20°, -15°]$ part of the blue class and a target image from the other part $[15°, 20°]$. Next, we generate an adversarial image [1] from the red class using the one step target class method Kurakin et al. (2016, 2017) and consider the path between the source image to the adversarial image and subsequently from the adversarial image to the target one. For each path, we simply consider the line segment $\lambda s + (1 - \lambda)t$ for $\lambda \in [0, 1]$ between the two endpoint images s and t and sample it very densely by dividing $[0, 1]$ into 10^4 equidistant parts. Figure 5.7 shows the complete path from the source image to the target image where the color indicates that all the intermediate images are classified as blue with high confidence (note that we turned the output of the network into probabilities by using the softmax function). Moreover, the intermediate images from Figure 5.7 look very much like images from the red class thus could be seen as adversarial samples for the red class. The point we want to make here is that one might think that in order to avoid adversarial manipulation the solution is to use a simple classifier of low capacity. We think that rather the opposite is true in the sense that only if the classifier is rich enough to model the true underlying data generating distribution it will be able to model the true decision boundaries. In particular, the classifier should be able to realize disconnected decision regions in order to avoid paths through the input space which connect different disconnected regions of the Bayes optimal classifier. Now one could argue that the problem of our synthetic example is that the corresponding digits obviously do not fill the whole image space, nevertheless the classifier has to do a prediction for all possible images. This could be handled by introducing a background class, but then it would be even more important that the classifier can produce disconnected decision regions which naturally requires a minimal width of $d + 1$ of the network.

In Figure 5.8, we show another similar experiment on MNIST dataset, but now for all the 10 image

[1] This is essentially a small perturbation of an image from the red class which is classified as blue class

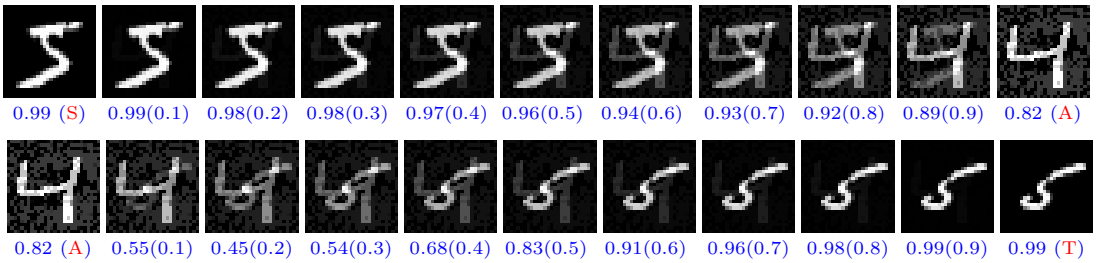


Figure 5.8: Standard MNIST dataset with 10 classes. Top row: Trajectory from a source image (S) to an adversarial image (A) parameterized by λ (shown inside brackets). Bottom row: Trajectory from the adversarial image (A) to a target image (T). Each number outside bracket shows the confidence level (from 0 to 1) that the corresponding image was predicted to be in blue class (digit 5) out of 10 classes.

classes. We train a network with 200 hidden units, leaky ReLU and softmax cross-entropy loss to zero training error. Once again, one can see that there exists a continuous path that connects two different-looking images of digit 5 (blue class) where every image along this path is classified as blue class with high confidence. Moreover this path goes through a pre-constructed adversarial image of the red class (digit 4).

5.4 Summary

We have shown that deep neural networks need to have in general width larger than the input dimension in order to learn disconnected decision regions. Moreover, our experiments show that too narrow networks produce high confidence predictions on a continuous path connecting the true disconnected components of decision regions, which then could be used to attack these networks using adversarial manipulation. While our result does not resolve the question how to choose the network architecture in practice, it provides at least a guideline how to choose the width of the network.

Chapter 6

Summary and outlook

In this thesis, we study the landscape of the empirical risk of deep over-parameterized neural networks in an attempt to improve our theoretical understanding of why local search algorithms such as (stochastic) gradient descent can frequently converge to a global minimum in training these networks. Our analysis touches upon fundamental and important aspects of the problem such as existence of solutions with zero training error, global optimality of critical points, connectivity and unboundedness of level sets and sublevel sets, and connectivity of decision regions. We proved several results for fully connected networks and convolutional neural networks, showing that a sufficiently wide layer in the architecture is key to ensure both expressive power of the network and well-behaved property of the loss surface. Finally, we devised a new class of CNN architectures with skip-connections to the output, which are practically relevant and have a strong theoretical guarantee on the loss landscape.

Our research in this thesis leads to some open problems. First, our current results on well-behaved loss surface (Chapter 2 and Chapter 3) crucially rely on the key assumption that the network contains a wide hidden layer with at least N neurons. While this condition has been relaxed to $n_1 + \ldots + n_{L-1} \geq N$ in Chapter 4 by using random skip-connections, the network still has an implicitly over-parameterized output layer whose total number of parameters can amount to Nm which can be problematic if the output dimension m is large. For instance, for ImageNet dataset where $N = 10^6$ and $m = 1000$, the network will have at least 10^9 parameters just for those skip-connections to the output units. Thus from a practical point of view, it would be helpful to know if there are other ways to improve our current condition on the wide layer without using skip-connections as done in Chapter 4. One approach which we think can be interesting is to extend results from expressivity of neural networks to optimization landscape. Along this line, Yun et al. (2018) have shown that a three hidden layer network with $4\sqrt{N} + 4m$ neurons can already memorize arbitrary datasets. However this result does not mean that local search algorithms like gradient descent can find a global solution efficiently, as the landscape of the loss function is still unknown. In fact, one can easily construct a function whose global minima have zero loss but the function itself can have arbitrarily bad local valleys with large volume where local search algorithms might get stuck. In contrast, while our results of this thesis require more neurons than necessary (i.e. N neurons at a wide layer), it allows us to show that the network can not only fit arbitrary datasets but also have a well-behaved loss surface, which seems to agree with common empirical observations in deep learning Zhang et al. (2017).

Second, while our results on connectivity of sublevel sets from Chapter 3 might shed light on the geometric structure of the loss function of deep over-parameterized neural nets, it still remains unclear what these imply for optimization. In particular, while the property of connected sublevel sets might help us understand why local search algorithms like gradient descent do not easily get trapped at

bad local valleys, it cannot fully explain why these algorithms work so well in practice as the loss surface might still contain large plateaus (i.e. the loss is almost flat in a large area) where learning becomes difficult. At this point, we also want to understand if the continuous trajectory of gradient descent is (often) a strict descent path, or is it necessary that every continuous path has to cross some regions with constant loss in parameter space before it can reach a global minimum.

Another interesting direction is to understand the implicit bias of SGD as shown in Section 4.4, as well as the generalization performance of different (global) solutions with zero training error. In particular, we have shown that for a class of deep CNNs with enough skip-connections to the output layer, one can have two training algorithms which both achieve zero training error: 1) a random feature approach where we randomize and fix all weights of the network during training except the skip-connection weights which are trained with SGD, and 2) the standard approach where all parameters are optimized with SGD. In our experiments, both of these algorithms can find solutions which fit perfectly the training data, but have significantly different test error. In particular, training the whole network with SGD frequently converges to a global minimum with 10-20% better test error than the ones found by random feature approach. This is a very strong sign that SGD is implicitly biased towards the better solutions among infinitely many points with zero training error. In the literature, there are several recent work which try to understand this phenomenon by showing that (stochastic) gradient descent might have some implicit regularization under specific settings, such as for deep linear networks Ji and Telgarsky (2019), deep linear convolutional networks Gunasekar et al. (2018) and shallow nets with linearly separable data Brutzkus et al. (2018); Soudry et al. (2018). It remains an interesting open question whether these results could be extended to the case of deep non-linear networks as analyzed in this thesis, which might help us to understand better generalization in deep learning.

Bibliography

G. Alain and Y. Bengio. Understanding intermediate layers using linear classifier probes. In *International Conference on Learning Representations (ICLR Workshop)*, 2016.

Z. Allen-Zhu, Y. Li, and Y. Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. *arXiv:1811.04918*, 2018a.

Z. Allen-Zhu, Y. Li, and Z. Song. A convergence theory for deep learning via over-parameterization. *arXiv:1811.03962*, 2018b.

S. An, F. Boussaid, and M. Bennamoun. How can deep rectifier networks achieve linear separability and preserve distances? In *International Conference on Machine Learning (ICML)*, 2015.

A. Andoni, R. Panigrahy, G. Valiant, and L. Zhang. Learning polynomials with neural networks. In *International Conference on Machine Learning (ICML)*, 2014.

T. M. Apostol. *Mathematical analysis*. Addison Wesley, Reading, Massachusetts, 1974.

R. Arora, A. Basu, P. Mianjy, and A. Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations (ICLR)*, 2018a.

S. Arora, A. Bhaskara, R. Ge, and T. Ma. Provable bounds for learningsome deep representation. In *International Conference on Machine Learning (ICML)*, 2014.

S. Arora, N. Cohen, and E. Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *ICML*, 2018b.

S. Arora, N. Cohen, N. Golowich, and W. Hu. A convergence analysis of gradient descent for deep linear neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

P. Auer, M. Herbster, and M. K. Warmuth. Exponentially many local minima for single neurons. In *Advanced in Neural Information Processsing Systems (NIPS)*, 1996.

P. Baldi and K. Hornik. Neural networks and principle component analysis: Learning from examples without local minima. *Neural Networks*, pages 53–58, 1988.

P. Bartlett, D. Helmbold, and P. Long. Gradient descent with identity initialization efficiently learns positive definite linear transformations by deep residual networks. In *International Conference on Machine Learning (ICML)*, 2018.

H. P. Beise, S. D. Cruz, and U. Schröder. On decision regions of narrow deep neural networks. *arXiv:1807.01194*, 2018.

A. Blum and R. L Rivest. Training a 3-node neural network is np-complete. In *Advanced in Neural Information Processsing Systems (NIPS)*, 1989.

A. Brutzkus and A. Globerson. Globally optimal gradient descent for a convnet with gaussian inputs. In *International Conference on Machine Learning (ICML)*, 2017.

A. Brutzkus, A. Globerson, E. Malach, and S. Shalev-Shwartz. Sgd learns over-parameterized networks that provably generalize on linearly separable data. In *International Conference on Learning Representations (ICLR)*, 2018.

V. Charisopoulos and P. Maragos. A tropical approach to neural networks with piecewise linear activations. *arXiv:1805.08749*, 2018.

L. Chizat and F. Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Advanced in Neural Information Processsing Systems (NIPS)*, 2018.

F. Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv:1610.02357*, 2016.

A. Choromanska, M. Hena, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015a.

A. Choromanska, Y. LeCun, and G. B. Arous. Open problem: The landscape of the loss surfaces of multilayer networks. In *Annual Conference on Learning Theorey (COLT)*, 2015b.

D. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations (ICLR)*, 2016.

N. Cohen and A. Shashua. Convolutional rectifier networks as generalized tensor decompositions. In *International Conference on Machine Learning (ICML)*, 2016.

N. Cohen, O. Sharir, and A. Shashua. On the expressive power of deep learning: A tensor analysis. In *Annual Conference on Learning Theorey (COLT)*, 2016.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, pages 303–314, 1989.

W. M. Czarnecki, G. Swirszcz, M. Jaderberg, S. Osindero, O. Vinyals, and K. Kavukcuoglu. Understanding synthetic gradients and decoupled neural interfaces. In *International Conference on Machine Learning (ICML)*, 2017.

Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advanced in Neural Information Processsing Systems (NIPS)*, 2014.

O. Delalleau and Y. Bengio. Shallow vs. deep sum-product networks. In *Advanced in Neural Information Processsing Systems (NIPS)*, 2011.

F. Draxler, K. Veschgini, M. Salmhofer, and F. Hamprecht. Essentially no barriers in neural network energy landscape. In *International Conference on Machine Learning (ICML)*, 2018.

S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. *arXiv:1811.03804*, 2018a.

S. S. Du, J. D. Lee, and Y. Tian. When is a convolutional filter easy to learn? In *International Conference on Learning Representations (ICLR)*, 2018b.

S. S. Du, X. Zhai, B. Poczos, and A. Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

R. Eldan and O. Shamir. The power of depth for feedforward neural networks. In *Annual Conference on Learning Theorey (COLT)*, 2016.

J. C. Evard and F. Jafari. The set of all mxn rectangular real matrices of rank-r is connected by analytic regular arcs. In *Proceedings of American Mathematical Society*, 1994.

A. Fawzi, S. Mohsen M. Dezfooli, P. Frossard, and S. Soatto. Classification regions of deep neural networks. *arXiv:1705.09552*, 2017.

T. Garipov, P. Izmailov, D. Podoprikhin, D. Vetrov, and A. G. Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advanced in Neural Information Processsing Systems (NIPS)*, 2018.

A. Gautier, Q. Nguyen, and M. Hein. Globally optimal training of generalized polynomial neural networks with nonlinear spectral methods. In *Advanced in Neural Information Processsing Systems (NIPS)*, 2016.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Machine Learning (ICML)*, 2010.

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

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

M. Gori and A. Tesi. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 76–86, 1992.

S. Gunasekar, J. D. Lee, D. Soudry, and N. Srebro. Implicit bias of gradient descent on linear convolutional networks. In *Advanced in Neural Information Processsing Systems (NIPS)*, 2018.

B. D. Haeffele and R. Vidal. Global optimality in neural network training. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

B. Hanin and M. Sellke. Approximating continuous functions by relu nets of minimal width. *arXiv:1710.11278*, 2017.

M. Hardt and T. Ma. Identity matters in deep learning. In *International Conference on Learning Representations (ICLR)*, 2017.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International Conference on Computer Vision (ICCV)*, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, pages 359–366, 1989.

G. Huang, Z. Liu, L. Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$ 0.5mb model size. *arXiv:1602.07360*, 2016.

K. Jarrett, K. Kavukcuoglu, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Z. Ji and M. Telgarsky. Gradient descent aligns the layers of deep linear networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Grishick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Int. Conference on Multimedia*, 2014.

J. Johnson. Deep, skinny neural networks are not universal approximators. In *International Conference on Learning Representations (ICLR)*, 2019.

K. Kawaguchi. Deep learning without poor local minima. In *Advanced in Neural Information Processsing Systems (NIPS)*, 2016.

D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

S. G. Krantz and H. R. Parks. *A Primer of Real Analytic Functions*. Birkhäuser, Boston, second edition, 2002.

A. Krizhevsky. Learning multiple layers of features from tiny images. *Technical report*, 2009.

A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations (ICLR)*, 2016.

A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. In *International Conference on Learning Representations (ICLR Workshop)*, 2017.

T. Laurent and J. v. Brecht. Deep linear networks with arbitrary loss: All local minima are global. In *International Conference on Machine Learning (ICML)*, 2018.

Y. Lecun, C. Cortes, and C.J.C. Burges. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, pages 436–444, 2015.

M. Leshno, Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, pages 861–867, 1993.

H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. In *International Conference on Learning Representations (ICLR Workshop)*, 2018.

Y. Li and Y. Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *NIPS*, 2018.

Y. Li and Y. Yuan. Convergence analysis of two-layer neural networks with relu activation. *arXiv:1705.09886*, 2017.

S. Liang and R. Srikant. Why deep neural networks for function approximation? In *International Conference on Learning Representations (ICLR)*, 2017.

S. Liang, R. Sun, J. D. Lee, and R. Srikant. Adding one neuron can eliminate all bad local minima. In *Advanced in Neural Information Processsing Systems (NIPS)*, 2018.

H. Lin and S. Jegelka. Resnet with one-neuron hidden layers is a universal approximator. In *NIPS*, 2018.

R. Livni, S. Shalev-Shwartz, and O. Shamir. On the computational efficiency of training neural networks. In *Advanced in Neural Information Processsing Systems (NIPS)*, 2014.

H. Lu and K. Kawaguchi. Depth creates no bad local minima. *arXiv:1702.08580*, 2017.

Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The expressive power of neural networks: A view from the width. In *Advanced in Neural Information Processsing Systems (NIPS)*, 2017.

A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning (ICML)*, 2013.

A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

John Makhoul, Amro El-Jaroudi, and Richard Schwartz. Formation of disconnected decision regions with a single hidden layer. *International Joint Conference on Neural Networks*, pages 455–460, 1989.

John Makhoul, Amro El-Jaroudi, and Richard Schwartz. Partitioning capbabilities of two-layer neural networks. *IEEE Trans. Signal Processing*, pages 1435–1440, 1990.

S. Mei, A. Montanari, and P. M. Nguyen. A mean field view of the landscape of two-layer neural networks. In *Proceedings of the National Academy of Sciences (PNAS)*, 2018.

H. Mhaskar and T. Poggio. Deep vs. shallow networks : An approximation theory perspective. *arXiv:1608.03287*, 2016.

B. Mityagin. The zero set of a real analytic function. *arXiv:1512.07276*, 2015.

G. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *Advanced in Neural Information Processsing Systems (NIPS)*, 2014.

B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. Exploring generalization in deep learning. In *Advanced in Neural Information Processsing Systems (NIPS)*, 2017.

P. M. Nguyen. Mean field limit of the learning dynamics of multilayer neural networks. *arXiv:1902.02880*, 2018.

Q. Nguyen. On connected sublevel sets in deep learning. *arXiv:1901.07417*, 2019.

Q. Nguyen and M. Hein. The loss surface of deep and wide neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

Q. Nguyen and M. Hein. Optimization landscape and expressivity of deep cnns. In *International Conference on Machine Learning (ICML)*, 2018.

Q. Nguyen, M.C. Mukkamala, and M. Hein. Neural networks should be wide enough to learn disconnected decision regions. In *International Conference on Machine Learning (ICML)*, 2018.

Q. Nguyen, M. C. Mukkamala, and M. Hein. On the loss landscape of a class of deep neural networks with no bad local valleys. In *International Conference on Learning Representations (ICLR)*, 2019.

V. D. Nguyen. Complex powers of analytic functions and meromorphic renormalization in qft. *arXiv:1503.00995*, 2015.

R. Pascanu, G. Montufar, and Y. Bengio. On the number of response regions of deep feedforward networks with piecewise linear activations. In *International Conference on Learning Representations (ICLR)*, 2014.

A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv:1606.02147*, 2016.

T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao. Why and when can deep – but not shallow – networks avoid the curse of dimensionality: a review. *arXiv:1611.00740*, 2016.

H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *International Conference on Computer Vision (ICCV Workshop)*, 2011.

W. H. Press. *Numerical Recipes: The art of scientific computing.* Cambridge University Press, 2007.

M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein. On the expressive power of deep neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

I. Safran and O. Shamir. On the quality of the initial basin in overspecified networks. In *International Conference on Machine Learning (ICML)*, 2016.

I. Safran and O. Shamir. Depth-width tradeoffs in approximating natural functions with neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

I. Safran and O. Shamir. Spurious local minima are common in two-layer relu neural networks. In *International Conference on Machine Learning (ICML)*, 2018.

A. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng. On random weights and unsupervised feature learning. In *International Conference on Machine Learning (ICML)*, 2011.

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, pages 85–117, 2015.

H. Sedghi and A. Anandkumar. Provable methods for training neural networks with sparse connectivity. In *International Conference on Learning Representations (ICLR Workshop)*, 2015.

T. Serra, C. Tjandraatmadja, and S. Ramalingam. Bounding and counting linear regions of deep neural networks. *arXiv:1711.02114*, 2018.

S. Shalev-Shwartz, O. Shamir, and S. Shammah. Failures of gradient-based deep learning. In *International Conference on Machine Learning (ICML)*, 2017.

O. Shamir. Distribution-specific hardness of learning neural networks. *arXiv:1609.01037*, 2017.

J. Sima. Training a single sigmoidal neuron is hard. *Neural Computation*, pages 2709–2728, 2002.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

M. Soltanolkotabi. Learning relus via gradient descent. *Advanced in Neural Information Processsing Systems (NIPS)*, 2017.

D. Soudry and E. Hoffer. Exponentially vanishing sub-optimal local minima in multilayer neural networks. In *International Conference on Learning Representations (ICLR Workshop)*, 2018.

D. Soudry, E. Hoffer, M. S. Nacson, and N. Srebro. The implicit bias of gradient descent on separable data. In *International Conference on Learning Representations (ICLR)*, 2018.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015a.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *arXiv:1512.00567*, 2015b.

C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv:1602.07261*, 2016.

M. Telgarsky. Benefits of depth in neural networks. In *Annual Conference on Learning Theorey (COLT)*, 2016.

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

L. Venturi, A. S. Bandeira, and J. Bruna. Spurious valleys in two-layer neural network optimization landscapes. *arXiv:1802.06384v2*, 2018.

G. Wang, G. B. Giannakis, and J. Chen. Learning relu networks on linearly separable data: Algorithm, optimality, and generalization. *arXiv:1808.04685*, 2018.

D. Yarotsky. Error bounds for approximations with deep relu networks. *arXiv:1610.01145*, 2016.

J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advanced in Neural Information Processsing Systems (NIPS)*, 2014.

J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. In *International Conference on Machine Learning (ICML)*, 2015.

X. Yu and G. Chen. On the local minima free condition of backpropagation learning. *IEEE Transaction on Neural Networks*, pages 1300–1303, 1995.

C. Yun, S. Sra, and A. Jadbabaie. Global optimality conditions for deep neural networks. In *International Conference on Learning Representations (ICLR)*, 2017.

C. Yun, S. Sra, and A. Jadbabaie. Finite sample expressive power of small-width relu networks. *arXiv:1810.07770*, 2018.

C. Yun, S. Sra, and A. Jadbabaie. Small nonlinearities in activation functions create bad local minima in neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

S. Zagoruyko and N. Komodakis. Wide residual networks. In *British Machine Vision Conference (BMVC)*, 2016.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, 2014.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and Oriol Vinyals. Understanding deep learning requires re-thinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

X. Zhang, Y. Yu, L. Wang, and Q. Gu. Learning one-hidden-layer relu networks via gradient descent. *arXiv:1806.07808*, 2018.

K. Zhong, Z. Song, P. Jain, P. Bartlett, and I. Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

Y. Zhou and Y. Liang. Critical points of neural networks: Analytical forms and landscape properties. In *International Conference on Learning Representations (ICLR)*, 2018.

D. Zou, Y. Cao, D. Zhou, and Q. Gu. Stochastic gradient descent optimizes over-parameterized deep relu networks. *arXiv:1811.08888*, 2018.