How to Convexify your Neural Network

Sumedh and Sampad

May 10, 2022

Abstract

Regularization is common practice while training deep neural networks (DNNs) - with the overarching goal of reducing the complexity of DNNs by penalizing the magnitude of the weights of the network. This results in several benefits including lowering the likelihood of overfitting [8, 14], improving sample efficiency [6, 12], robustness [17, 9] and generalization [19, 18]. A recent line of research studies how a convex relaxation of the empirical risk minimization style of training can serve as a way to regularize the DNN [4, 1, 21, 22]. A parallel line of work studies the converse, i.e., how regularization reduces the primal-dual gap in DNNs [20], how regularized DNNs can be viewed as parsimonious convex models in high-dimensions [5] and how Gradient Descent finds global minima of DNNs [3]. Our project attempts to investigate this link and to generate insights into benefits of observing DNNs from a convex lens. We survey these works and highlight future directions in the convex relaxations of Empirical Risk Minimization style of DNN training.

1 Introduction

The training of Deep Neural Networks (DNNs) poses, at first glance, a highly non-convex problem. With non-linearities sandwiched between each of the tens of layers of the DNN, it is surprising that first order methods such as Gradient Descent, known to converge only on convex objectives yield any useful solutions at all. Empirically however, on large datasets such as Imagenet [13, 2], and even on control tasks from high dimensional visual inputs [16, 15], they converge remarkbly well and consistently across random seed experiments. There are 2 possible explanation for this - (1) that our understanding of what constitutes "optimizable" objectives needs rethinking or (2) that there exists mechanisms within DNNs that permit a more convex behavior during training. In this paper, we explore path (2) and study how extant methods convexify neural networks.

More specifically, we study what **(a) design choices** and **(b) modifications to training algorithms** make DNNs intrinsically less non-convex. Traditionally, approaches such as regularization, inspired from traditional learning theory are often used to reduce the number of learnable parameters of the networks and are touted for their ability to improve sample efficiency [6, 12] and generalization [19, 18]. We study how according to Ergen and Pilanci, convex relaxations can be viewed as a form of regularization of the training objective. Specifically, relaxing the gradient descent subroutine and subsequently modifying the training loss results in an objective that includes the original empirical risk minimization term in addition to a regularizer.

With regards to design choices, we study seminal works which derive some very important properties of gradient descent for DNNs - namely that gradient descent finds the global minimum of the training loss under some mild structural assumptions [3] and that DNNs with parallel subnetworks (encompasing the recent highly successful Residual Networks or ResNets and related works) are intrinsically less non-convex [20]. Specifically, according to Du et al., for neural networks with large enough width (defined in subsequent sections), the rate of convergence of DNN training is approximately linear and more crucially *does not depend on the specific values of the parameters of the network, but only the data it trains on!* This allows the authors to conclude that due to the linearity of convergence rate, the objective achieves its global minimum. Finally, we study how according to Zhang et al., modern DNN architectures are intrinsically less non-convex by showing that the primal-dual gap is smaller in comparison to their older fully-connected counterparts. They show that this is true empirically as well, by visualizing the loss surfaces in the parameter space. They also show that the duality gap for linear deeep neural networks is zero for a general regularized objective

Our paper is organized as follows: we study learning objective modifications from the point-of-view of Zhang et al. in Section 2.1 followed by the views of Ergen and Pilanci in Section 2.2. Subsequently, in Section 3.1, we study the effect of width on the convergence to the global minimum of the training loss during SGD training of DNNs as presented by Du et al.. Next we study the effect of parallel branches in networks and their ability to convexify DNN training as presented by Zhang et al. in Section 3.2. In each section, we present the theorems proven by the authors along with the assumptions made. We also present a brief proof sketch for theorems, and our own interpretation of the method from aligning viewpoints but leave out the full proofs in the interest of brevity.

1.1 Notation

We define our notations exactly as Du et al. do in their paper. Let $[n] = 1, 2, ..., n$. We use $\mathbf{N}(\mathbf{0}, \mathbf{I})$ to denote the standard Gaussian distribution. For a matrix $\mathbf{A}$, we use $\mathbf{A}_{ij}$ to denote its (i, j)-th entry. We will also use $\mathbf{A}_{i,:}$, to denote the i-th row vector of A and define $\mathbf{A}_{i,j:k} = (\mathbf{A}_{i,j}, \mathbf{A}_{i,j+1}, , \mathbf{A}_{i,j+k})$ as part of the vector. Similarly $\mathbf{A}_{:i}$ is the i-th column vector and $\mathbf{A}_{j:k,i}$ is a part of i-th column vector. For a vector $\mathbf{v}$, we use $||\mathbf{v}||_2$ to denote the Euclidean norm. For a matrix $\mathbf{A}$ we use $||\mathbf{A}||_F$ to denote the Frobenius norm and $||\mathbf{A}||_2$ to denote the operator norm. If a matrix A is positive semi-definite, we use $\lambda_{min}(\mathbf{A})$ to denote its smallest eigenvalue. We use $\langle \cdot, \cdot \rangle$ to denote the standard Euclidean inner product between two vectors or matrices. We let $O(\cdot)$ and $\Omega(\cdot)$ denote standard Big-O and Big-Omega notations, only hiding constants. We will use C and c to denote constants. $|| \cdot ||_{S_H} = \left(\sum_i \sigma_i^H(\cdot) \right)^{1/H}$ is the Schatten-H norm

1.2 Neural Network Notation

Let the neural network consist of H. Let the first layer weight matrix be denoted by $\mathbf{W}^{(1)}$. The subsequent h-th layers of the net are $\mathbf{W}^{(h)}$. The output layer of the network is $\mathbf{a} \in \mathbb{R}^{d_H}$ where d_H. Let the data matrix be $\mathbf{X} \in \mathbb{R}^{d_0 \times n}$ where there are d_0 features and n data points. Let $\mathbf{Y} \in \mathbb{R}^{d_H \times n}$ where d_H is the output dimension of the neural networks and there are n outputs for each data point. $\sigma(\cdot)$ represents the activation function applied element-wise between the layers of the neural network.

2 Convexification through Learning Objective

2.1 Strong Duality of Linear Neural Networks

Optimising parameters of linear neural networks with MSE loss is a non-convex optimisation problem which is formulated as

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \frac{1}{2} ||\mathbf{Y} - \mathbf{W}\mathbf{X}||_F^2 \tag{1}$$

where

$$\mathbf{W} = \prod_{i=1}^{H} \mathbf{W}^{(i)}$$

The problem being non-convex, Zhang et al. consider a more general regularized objective

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \frac{1}{2} ||\mathbf{Y} - \mathbf{W}\mathbf{X}||_F^2 + \frac{\gamma}{H} \left[||\mathbf{W}^{(1)} X||_{S_H}^H + \sum_{i=2}^{H} ||\mathbf{W}^{(i)}||_{S_H}^H \right] \tag{2}$$

Using explanations by Gabriel, $\mathbf{W}$ is the weights of linear combination of the rows of $\mathbf{X}$ and $\mathbf{W}\mathbf{X}$ can only span the vectors in $rowspace(\mathbf{X})$. Unless the rows of $\mathbf{Y}$ are in $rowspace(\mathbf{X})$, the lowest mean square error is non-zero. The lowest squared error (euclidean distance) that can be achieved in this case is when $\mathbf{W}\mathbf{X}$ is equal to the orthogonal projection of $\mathbf{Y}$ onto the $rowspace(\mathbf{X})$. Since $\mathbf{X}^\dagger\mathbf{X}$ gives us the projection operator onto $rowspace(\mathbf{X})$, the projections $\tilde{\mathbf{Y}}$ given by

$$\tilde{\mathbf{Y}}^T = \mathbf{X}^\dagger \mathbf{X} \mathbf{Y}^\mathbf{T} \implies \tilde{\mathbf{Y}} = (\mathbf{Y}^\mathbf{T})^\mathbf{T} (\mathbf{X}^\dagger \mathbf{X})^\mathbf{T} = \mathbf{Y}\mathbf{X}^\dagger\mathbf{X}$$

since $\mathbf{X}^\dagger\mathbf{X} = (\mathbf{V}\mathbf{\Sigma}^\dagger\mathbf{U}^\mathbf{T})(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathbf{T}) = \mathbf{V}\mathbf{\Sigma}^\dagger\mathbf{\Sigma}\mathbf{V}^\mathbf{T}$ is symmetric as $\mathbf{\Sigma}^\dagger$ and $\mathbf{\Sigma}$ are both diagonal and hence symmetric.

Since the rank of the matrix $\mathbf{W}$ is restricted by the minimum of the widths of the layers of the neural network, the rank of $\tilde{\mathbf{Y}} \leq min\{d_0, d_1, ..., d_H, n\} = d_{min}$. The authors claim in a theorem that as long as $H \geq 2$ and $0 \leq \gamma \leq \sigma_{min}(\tilde{\mathbf{Y}})$, strong duality holds for the deep linear neural network where the dual of the formulation above is given by

$$\mathbf{\Lambda}^* = \underset{rowspace(\mathbf{\Lambda}) \subseteq rowspace(\mathbf{X})}{\text{argmax}} \quad -\frac{1}{2}||\tilde{\mathbf{Y}} - \mathbf{\Lambda}||^2_{d_{min}} + \frac{1}{2}||\mathbf{Y}||^2_F \tag{3}$$

$$\text{s.t. } ||\mathbf{\Lambda}|| \leq \gamma \tag{4}$$

where $||\cdot||^2_{d_{min}} = \sum_{i=1}^{d_{min}} \sigma_i^2(\cdot)$ is a convex function. The globally optimal solution to the original is recovered from the solution of the dual problem as

$$\mathbf{W}_*^{(\mathbf{i})} = \begin{bmatrix} \mathbf{\Sigma^{1/H}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad \forall\, i \in \{2, ..., H-1\}$$

$$\mathbf{W}_*^{(\mathbf{H})} = \begin{bmatrix} \mathbf{U\Sigma^{1/H}} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{W}_*^{(\mathbf{1})} = \begin{bmatrix} \mathbf{\Sigma^{1/H}V^T} \\ \mathbf{0} \end{bmatrix} \mathbf{X}^\dagger$$

where $\mathbf{U\Sigma V^T} = svd_{d_{min}}(\tilde{\mathbf{Y}} - \mathbf{\Lambda}^*)$ is the truncated SVD.

It is important to point out that the smallest singular value $\sigma_{min}(\tilde{\mathbf{Y}})$ often captures random noise level. When $\gamma = 0$, the formulation reduces to the non-regularized deep linear neural networks. Strong duality in non-convex problems is rare and the other non-convex problems that enjoy strong duality are matrix-completion, Fantope, quadratic optimisation with two quadratic constraints. Empirical observations support the notion that convex relaxations of deep neural network are easier to train. Doughlas-Rachford convex solver algorithm can be applied to the dual problem 3 (since dual is convex) followed by a SVD to get the weights $\{\mathbf{W}^{*H}_{\mathbf{i}\ i=1}\}$ form the truncated SVD $svd_{d_{min}}(\tilde{\mathbf{Y}} - \mathbf{\Lambda}^*)$.

What is interesting here is that strong duality still holds as long as linear layers are replaced with linear operations (convolution, linear operator with bias term - affine transforms) with a suitable group of kernels. Also since the result holds as long as $0 \leq \gamma \leq \sigma_{min}(\tilde{\mathbf{Y}})$, for $\gamma = 0$, strong duality for the non-regularised deep linear neural networks holds. A vast majority of results on deep learning study the generalisation and expressive power and do so by analyzing the complicated non-convex form - with strong duality the analysis can be done with simple convex forms (relaxations/surrogates). This result highlights that the relaxations/surrogates are tight to the original deep linear neural networks and hence the evidence for results on the relaxed versions, both theoretical and empirical, are strengthened. Earlier work by Kawaguchi has only addressed the non-regularised forms of the linear neural networks for non-existence of local minima. However, non-existence of local minima doesn't prevent gradient descent from being stuck in a bad saddle point while solving the primal. In contrast the results here work for a general regularized version, and have zero duality gap. Hence, solving the convex dual problem with a convex solver gets us the optimal error and the solution of the dual from which we can recover the solution of the primal - thereby finding the best weights $\mathbf{W}$ on the training data.

One shortcomings of the above method is that there is a limit to the amount of bias we can add to the model via regularisation since $\gamma \leq \sigma_{min}(\tilde{\mathbf{Y}})$.

2.2 Convex Relaxations of Shallow Neural Networks

Above, in the case of linear neural networks, the activation function $\sigma(\cdot)$ was the identity function $\sigma(x) = x$. However, in practice non-linear functions are applied element-wise and neural networks take the form $f(x) = \sigma(\mathbf{W}^{(H)}(\sigma(\mathbf{W}^{(H-1)}(\cdots\sigma(\mathbf{W}^{(1)})))))$. We study a special case of the DNNs, shallow neural networks which consists of a single layer neural network with a single matrix of weights that maps inputs from $\mathbb{R}^{d_0}$ to $\mathbb{R}$ defined as $\mathbf{W} \in \mathbb{R}^{d_0}$ with the data matrix $\mathbf{X} \in \mathbb{R}^{d_0 \times n}$. The activation function is applied element-wise to the output and is defined as $\sigma(x) = max\{0, x\}$. The ground truth labels are $\mathbf{Y} \in \mathbb{R}^n$. The following optimization problem is solved from the lens of Ergen and Pilanci.

$$\arg\min_x \frac{1}{2}||\sigma(\mathbf{W^T X}) - \mathbf{Y}||_2^2 \tag{5}$$

We can solve Equation 5 using gradient descent as

$$\mathbf{W}(k) = \mathbf{W}(k) - \eta(\sigma(\mathbf{W}^\intercal \mathbf{X}) - \mathbf{Y})\mathbf{D}(k)\mathbf{X}^\intercal \tag{6}$$

where η is the learning rate, k represents the training epoch and $\mathbf{D}(k)$ is a diagonal matrix with the i-th diagonal entry equal to 1 if $\mathbf{W}^\intercal \mathbf{x}_i \geq 0$. We can view squared error loss of Equation 5 as the following by rearranging terms:

$$L(\mathbf{W}) = ||\sigma(\mathbf{W}^\intercal \mathbf{X})||_2^2 - 2\sigma(\mathbf{W}^\intercal \mathbf{X})\mathbf{Y}^\intercal + ||\mathbf{Y}||_2^2 \tag{7}$$

This is non-convex due to the second term. We can however relax the loss to make it convex in the following manner:

$$L_r(\mathbf{W}) = ||\sigma(\mathbf{W}^\intercal \mathbf{X})||_2^2 - 2\mathbf{W}^\intercal \mathbf{X}\mathbf{Y}^\intercal + ||\mathbf{Y}||_2^2 \tag{8}$$

The authors subsequently prove the following theorem:

Theorem: Let $\tilde{\mathbf{W}}$ be the global minimizer of the loss $L_r(\mathbf{W})$ in Equation 8. Then, $\tilde{\mathbf{W}}$ is the global minimizer of $L(\mathbf{W})$ in Equation 7.

This relaxation allows the authors to train the Shallow Neural Network using a convex objective. The authors subsequently show that convexified Shallow networks converge faster on toy datasets.

3 Convexification through Design Choices

3.1 Effect of Width

Du et al. consider a DNN with with H layers and width of m neurons. The network is assumed to have been trained on n data points. They assume that the activation function is Lipschitz and smooth. Let the first layer weight matrix be denoted by $\mathbf{W}^{(1)} \in \mathbb{R}^{m \times d}$. The subsequent h-th layers of the net are $\mathbf{W}^{(h)} \in \mathbb{R}^{m \times m}\ \forall\ 2 \leq h \leq H$. The output layer of the network is $\mathbf{a} \in \mathbb{R}^m$. Their contributions can be summarized as follows:

- **Fully-connected Feed-forward networks:** For vanilla fully connected DNNs, if the width of the network m is $\Omega(poly(n), 2^{O(H)})$, then the training error reduces to zero at a linear rate.
- **Residual Networks (or ResNets):** The current state of the art in DNNs for a variety of tasks, ResNets enjoy a smaller width requirement. If the width of the network m is $\Omega(poly(n, H))$, then the training error reduces to zero at a linear rate.

Due to the linear convergence rate, the authors posit that DNNs find the global minimum of the training loss.

3.1.1 Proof Sketch

Consider the training data matrix to be $\mathbf{X} \in \mathbb{R}^{n \times d_0}$ with n datapoints in d dimensions. Let $\mathbf{Y} \in \mathbb{R}^n$ be the output label for the data. The authors define the prediction of the neural network for the i-th datapoint $\mathbf{x}_i$ after k epochs of training to be

$$u_i(k) = f(\theta(k), \mathbf{x}_i) \tag{9}$$

where $\theta(k)$ is unified representation for the parameters of the neural network after k epochs of training. Thus, the vector of outputs $\mathbf{u}(k) = (u_1(k), \cdots, u_n(k))^T$. The key technique of the paper is that dynamics of the loss i.e., the dynamics of $\{\mathbf{y} - \mathbf{u}(k)\}_{k=0}^{\infty}$ admits the following form:

$$\mathbf{y} - \mathbf{u}(k+1) = (\mathbf{I} - \eta\mathbf{G}(k))(\mathbf{y} - \mathbf{u}(k)) \tag{10}$$

where,

$$\begin{aligned}
\mathbf{G}_{i,j}(k) &= \langle \frac{\partial u_i(k)}{\partial \theta(k)}, \frac{\partial u_j(k)}{\partial \theta(k)} \rangle \\
&= \sum_{h=1}^{H} \langle \frac{\partial u_i(k)}{\partial \mathbf{W}^{(h)}(k)}, \frac{\partial u_j(k)}{\partial \mathbf{W}^{(h)}(k)} \rangle + \langle \frac{\partial u_i(k)}{\partial \mathbf{a}(k)}, \frac{\partial u_j(k)}{\partial \mathbf{a}(k)} \rangle \\
&\triangleq \sum_{h=1}^{H} \mathbf{G}_{i,j}^{(h)}(k)
\end{aligned} \tag{11}$$

Note that for each entry of $\mathbf{G}^{(h)}(k)$ is an inner product and thus $\forall\ h, \mathbf{G}^{(h)}(k)$ is a Positive Semi-definite (PSD) matrix. The authors show that for large enough width m, for each $k = 0, 1, \cdots$, $\mathbf{G}^{(h)}(k)$ is close to fixed to matrix $\mathbf{K}^{(h)}$ which depends on input data, the neural network architecture and activation function *but does not depend on the neural network parameters* θ! Subsequently, they use the analysis of power method to show that as long as $\mathbf{K}^{(h)}$ is positive definite, then gradient descent will have a linear convergence rate. They also prove that as long as the data is not degenerate, $\mathbf{K}^{(h)}$ will be positive definite.

To show that $\mathbf{K}^{(h)}$ is not dependent on the DNN parameters θ, the authors show that averaged Frobenius norm of each of the weight matrices of the layers is small for every iteration $k = 0, 1, \cdots$, i.e.,

$$\frac{1}{\sqrt{m}}||\mathbf{W}^{(h)}(k) - \mathbf{W}^{(h)}(0)||_2 \tag{12}$$

is small. Put simply, the change between the initialization weight matrix and the trained matrix is small for every layer and for every iteration.

Definition 1: The Gram Matrix $\mathbf{K}^{(h)}$ is defined recursively for layers $h = 1, 2, \cdots, H-1$ for $(i, j) \in [n] \times [n]$:

$$\mathbf{K}_{ij}^{(0)} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$
$$\mathbf{A}_{ij}^{(h)} = \begin{pmatrix} \mathbf{K}_{ii}^{(h-1)} & \mathbf{K}_{ij}^{(h-1)} \\ \mathbf{K}_{ji}^{(h-1)} & \mathbf{K}_{jj}^{(h-1)} \end{pmatrix}$$
$$\mathbf{K}_{ij}^{(h)} = c_\sigma \mathbb{E}_{(u,v)\sim N(\mathbf{0}, \mathbf{A}_{\mathbf{ij}}^{(\mathbf{h})})}[\sigma(u)\sigma(v)]$$
$$\mathbf{K}_{ij}^{(H)} = c_\sigma \mathbf{K}_{ij}^{(H-1)} \mathbb{E}_{(u,v)\sim N(\mathbf{0}, \mathbf{A}_{\mathbf{ij}}^{(\mathbf{H-1})})}[\sigma'(u)\sigma'(v)]$$

where $\mathbf{x}_i$ is the ith data point, σ is the activation function and σ' is the first derivative. $c_\sigma = \mathbb{E}_{N(0,1)}[\sigma(x)^2]^{-1}$ is a normalizer. We now formally state the result proved by Du et al..

Theorem: Convergence Rate for all Deep Fully Connected Neural Networks: Assume for $\forall\ i \in [n], ||x||_2 = 1, |y| = O(1)$ and the number of hidden layers

$$m = \Omega(2^{O(H)} max(\frac{n^4}{\lambda_{min}^4(\mathbf{K}^{(H)})}, \frac{n}{\delta}, \frac{n^2 log(\frac{Hn}{\delta})}{\lambda_{min}^2(\mathbf{K}^{(H)})})) \tag{13}$$

where $\mathbf{K}^{(H)}$ is as in Definition 1. If we set step size of gradient descent to be:

$$\eta = O(\frac{\lambda_{min}(\mathbf{K}^{(H)})}{n^2 2^{O(H)}}) \tag{14}$$

then with probability at least $1 - \delta$ over random initialization, the loss at each iteration $k = 1, 2, \cdots$ satisfies:

$$L(\theta(k)) \leq (1 - \frac{\eta \lambda_{min}(\mathbf{K}^{(H)})}{2})^k L(\theta(0) \tag{15}$$

Intuitively this means that the training loss of the DNN during gradient descent decreases linearly at a rate determined by the data, architecture and the activation functions but independent of the parameter values of the network. The main assumption is that the width of the network should be high enough. The dependence on the number of layers however is exponential.

Width Requirements for ResNets: In the context of ResNets, the authors derive a similar result to that of Theorem 1 with the exception that for linear convergence, the size of the neural network should be **polynomial in width** instead of exponential as is the case with fully connected DNNs. This reveals the importance of the multibranch architecture in ResNets.

3.2 Effect of Parallel Branches

Zhang et al. also studied the effect of parallel branches on the loss landscape of the network. In this work, they showed that the deep neural network architectures with parallel branches have smaller duality gap which goes to zero with increase in number of branches. Single hidden layer neural networks with small number of hidden units are reported to get stuck in poor local minima. Significant evidence also support existence of bad saddle points in deep neural networks which are hard to escape. Due to computational obstacles, over parametrization and multi-branch architectures have evolved which are observed to mitigate the problems to a large extent since first order methods like gradient descent converge to global minimum empirically. This supports the folklore that these architectures make the loss surface "look more convex". A lot of the SOTA architectures like ResNeXT, Inception, Xception, Squeezenet and Wide ResNet fall into the category of both over-parametrization and multi-branch. Hence the motivation in this work is to show that multi-branching reduces non-convexity of the loss. There are probably many metrics to capture non-convexity but the metric used here is duality gap. Although there are a lot of non-convex problems such as PCA and quadratic programming where duality gap is zero/small, the authors argue that the duality gap is, regardless, a measure of intrinsic non-convexity with smaller dual gaps relating to lesser non-convexity. Their arguments are that the optimal value of the dual problem has zero duality gap. The convex relaxation of the original problem refers to relaxing the non-convex objective to convexity and taking the feasible region as the convex hull of the original problem. Hence the duality gap measures the discrepancy between the original problem and the relaxed problem. Duality gap is zero for most convex problems but not all as mentioned in Jonathan M. Borwein and Yao. Secondly the authors show in their main result that the duality gap is a lower bound of the discrepancy between the primal and its convex relaxation - which means lower duality gap implies a possibly smaller discrepancy.

The authors analyze a classification setting and the multi-branch neural net is modelled as follows

$$f(w;x) = \frac{1}{I}\sum_{i=1}^{I} f_i(w_{(i)};x)\ ,\ w_{(i)} \in W_i \tag{16}$$

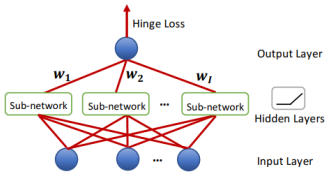


Figure 1: Multi-branch architecture

where W_i is a convex set, w is the concatenation of the network parameters $\{w_{(i)}\}_{i=1}^{I}$ of the branches, $x \in \mathbb{R}^{d_0}$ is a data point and $f_i(w_{(i)};\cdot) : \mathbb{R}^{d_0} \to \mathbb{R}$ is a continuous map by an sub-network with any arbitrary architecture (CNN, RNN, etc.). The activation functions in each layers of a sub-network needs to be continuous and sub-networks can have arbitrary depth H_i. The case $H_i = 1$ and $d_{H_i} = 1$ corresponds to each sub-network representing one hidden unit and the architecture $f(w;x)$ reduces to a one-hidden-layer network.

The τ-hinge loss is used on top of the network output for label $y \in \{-1,+1\}$:

$$l_\tau(w;x,y) := max\left(0, 1 - \frac{y \cdot f(w;x)}{\tau}\right)\ ,\ \tau > 0 \tag{17}$$

With $\tau = 0$, we get the classic hinge loss. However, in their analysis, τ needs to be sufficiently large. The analysis is built upon tools from non-convex geometric analysis - Shapely-Folkman lemma (analogous to central limit theorem in Probability). The lemma states that the sum of constrained non-convex functions is close to being convex. Thee authors analyze the duality gap of the risk w.r.t the above loss with an extra regularization constraint.

$$\min_{w \in W_1 \times W_2 \times \ldots W_I} \mathbb{E}_{(x,y)\sim P}[l_\tau(w;x,y)]$$
$$\text{s.t}\ \frac{1}{I}\sum_{i=1}^{I} h_i(w_{(i)}) \leq K$$

where $h_i(\cdot), i \in [I]$ are convex regularization functions (e.g. the weight decay) and K can be arbitrary as long as the problem is feasible. The dual problem is given by

$$\max_{\lambda>0} Q(\lambda) - \lambda K,$$

$$Q(\lambda) := \mathbb{E}_{\substack{(x,y)\sim P \\ w\in W_1\times\ldots\times W_I}}[l_\tau(w;x,y)] + \frac{\lambda}{I}\sum_{i=1}^{I} h_i(w_{(i)})$$

There are some more details which we skip here. Essentially they find the duality gap for each of $f_i(w_{(i)}, x)$ using restrictions on the original objective which is captured in the Δ_i defined below. The main result about multi-branch neural networks follows.

Let $inf(P)$ be the minimum of the primal and $sup(D)$ be the maximum of the dual problem. Let $\Delta_i := sup_{w\in W_i}\left\{\hat{f}_i(w) - \tilde{f}_i(w)\right\} \geq 0$ and $\Delta_{worst} := max_{i\in I}\Delta_i$.

If there exist a feasible solution for the problem (P), then under some assumptions, the duality gap can bee bounded by

$$0 \leq \frac{inf(P) - sup(D)}{\Delta_{worst}} \leq \frac{2}{I}$$

The normalized duality gap above helps in avoiding the trivialities in characterizing the degree of non-convexity: scaling the objective function by any constant does not change the value of the normalized duality gap.

The Δ_{worst} does not grow as fast as I and hence the duality gap $0 \leq inf(P) - sup(D) \leq \frac{2\Delta_{worst}}{I}$ gets squeezed to 0. Recall that I is the number of parallel sub-networks (branches).

Setting K infinitely large implies that the above theorem holds for unconstrained deep neural networks. The key result from using the above measure of non-convexity is that the wider the neural network, the smaller the duality gap and hence the lesser the non-convexity of the loss.

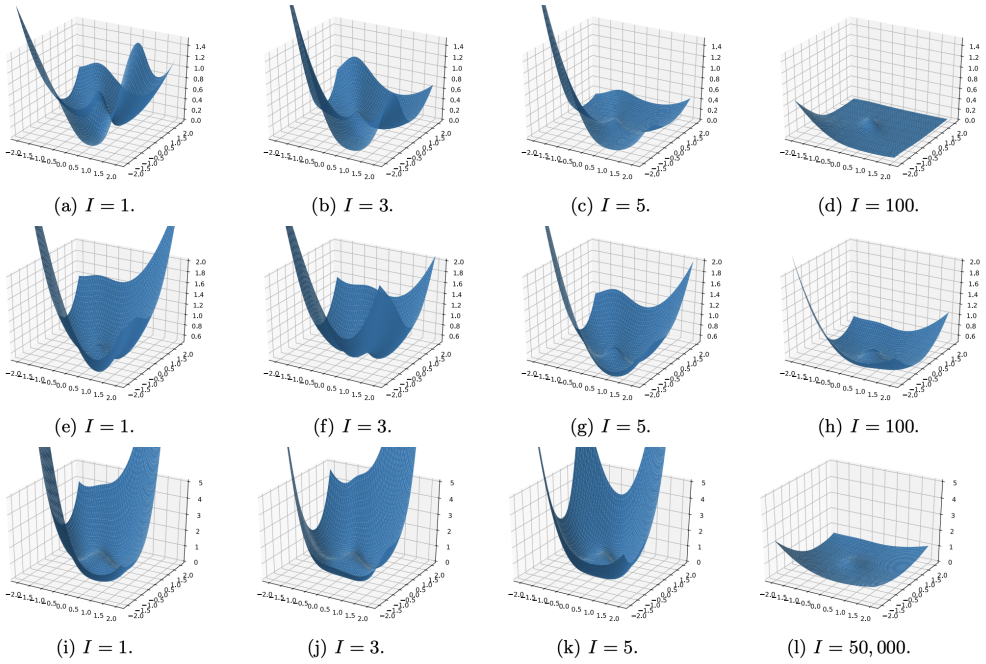
4 Conclusion and Future Directions

It appears that convexity for DNNs brings with it several important things. It makes loss surfaces easier to optimize over. As can be seen in Figure 2, the loss of networks with more parallel branches are increasingly less non-convex. It also makes convergence during training easier as can be seen from Figure 3. Thus, making neural networks more convex seems to benefit stability during training.

One key observation we were able to make is that it might be that convexity improves the learning capacity of DNNs. Consider the work from Zhang et al. explained in Section 3.2. Clearly, adding branches as is done in the case of ResNets reduces the primal-dual gap. Additionally, as is shown from the work by Du et al., ResNets have a smaller width requirement to guarantee linear convergence (fully connected networks require an **exponential** width while Resnet require a polynomial width in the number of layers). Similarly, Ergen and Pilanci show that a convex relaxation improves convergence for networks with the same number of parameters. Simply put, it appears that architectures/loss functions that are less non-convex require fewer parameters for high performance than more non-convex ones. Thus the first question we ask is ***Does convexifying increasing the per parameter train-ability of neural networks?***

Our next endeavour is to look carefully at the proof structure from Du et al. and study at what juncture the proofs for ResNets (with polynomial width requirements for convergence) differ from those of vanilla DNNs (which require exponential width for convergence). We would like to utilize the proof as a starting point for motivation for new architectures that share the properties of ResNets (or possibly even surpass them), i.e., ***are there other architectures that would admit the same polynomial requirements as ResNets?***

Another direction is to see if the skip connections in the ResNets are equivalent to adding parallel branches which would then explain the fact that ResNets are intrinsically less non-convex using results from Zhang et al..

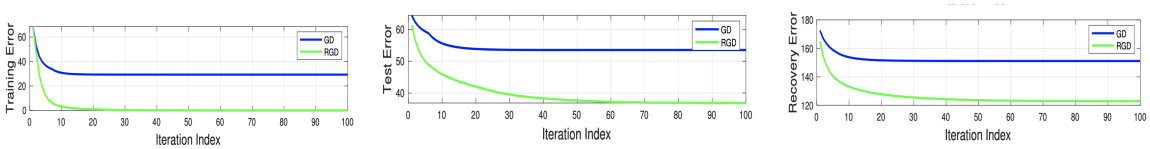


(a) $I = 1$. (b) $I = 3$. (c) $I = 5$. (d) $I = 100$.

(e) $I = 1$. (f) $I = 3$. (g) $I = 5$. (h) $I = 100$.

(i) $I = 1$. (j) $I = 3$. (k) $I = 5$. (l) $I = 50,000$.

Figure 2: **Training Loss Surfaces for various values of number of parallel branches.** Taken from Zhang et al.. Top Row: Landscape of one-hidden-layer network on MNIST. Middle Row: Landscape of one hidden-layer network on CIFAR-10. Bottom Row: Landscape of three-hidden-layer, multi-branch network on CIFAR-10 dataset. From left to right, the landscape looks less non-convex



(a) Training Error (b) Test Error (c) Recovery Error

Figure 3: **Convex Relaxed Gradient Descent vs Vanilla Gradient Descent.** Taken from Ergen and Pilanci. The relaxed version of gradient descent converges to a lower value of training, test and recovery error in comparison to the vanilla GD.

References

[1] Burak Bartan and Mert Pilanci. Convex relaxations of convolutional neural nets. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4928–4932. IEEE, 2019.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[3] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR, 2019.

[4] Tolga Ergen and Mert Pilanci. Convex optimization for shallow neural networks. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 79–83. IEEE, 2019.

[5] Tolga Ergen and Mert Pilanci. Path regularization: A convexity and sparsity inducing regularization for parallel relu networks. *arXiv preprint arXiv:2110.09548*, 2021.

[6] Jesse Farebrother, Marlos C Machado, and Michael Bowling. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

[7] K. R. Gabriel. Least squares approximation of matrices by additive and multiplicative models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1978.

[8] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[9] Daniel Jakubovitz and Raja Giryes. Improving dnn robustness to adversarial attacks using jacobian regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 514–529, 2018.

[10] Regina S. Burachik Jonathan M. Borwein and Liangjin Yao. Conditions for zero duality gap in convex programming. *arXiv)*, 2013.

[11] K. Kawaguchi. Deep learning without poor local minima. *Advances in Neural Information Processing Systems,)*, 2016.

[12] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[14] Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9078–9086, 2019.

[18] Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving generalization in reinforcement learning with mixture regularization. *Advances in Neural Information Processing Systems*, 33: 7968–7978, 2020.

[19] Colin Wei, Jason D Lee, Qiang Liu, and Tengyu Ma. Regularization matters: Generalization and optimization of neural nets vs their induced kernel. *Advances in Neural Information Processing Systems*, 32, 2019.

[20] Hongyang Zhang, Junru Shao, and Ruslan Salakhutdinov. Deep neural networks with multi-branch architectures are intrinsically less non-convex. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1099–1109. PMLR, 2019.

[21] Yi Zhang, Samuel Burer, W Nick Street, Kristin P Bennett, and Emilio Parrado-Hernández. Ensemble pruning via semi-definite programming. *Journal of machine learning research*, 7(7), 2006.

[22] Yuchen Zhang, Percy Liang, and Martin J Wainwright. Convexified convolutional neural networks. In *International Conference on Machine Learning*, pages 4044–4053. PMLR, 2017.