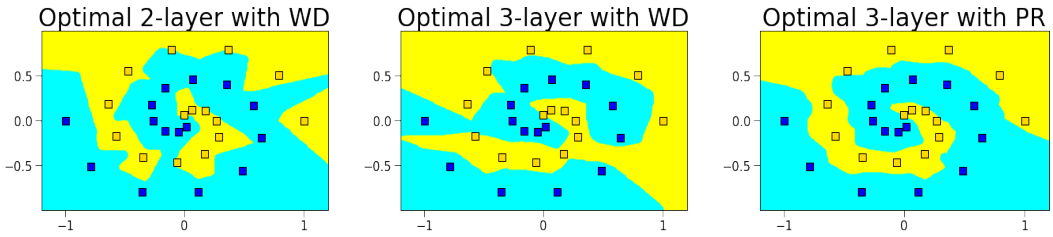
Path Regularization: A Convexity and Sparsity Inducing Regularization for Parallel ReLU Networks

Tolga Ergen & Mert Pilanci

Department of Electrical Engineering
Stanford University
Stanford, CA 94305, USA
{ergen,pilanci}@stanford.edu

Abstract

Understanding the fundamental principles behind the success of deep neural networks is one of the most important open questions in the current literature. To this end, we study the training problem of deep neural networks and introduce an analytic approach to unveil hidden convexity in the optimization landscape. We consider a deep parallel ReLU network architecture, which also includes standard deep networks and ResNets as its special cases. We then show that pathwise regularized training problems can be represented as an exact convex optimization problem. We further prove that the equivalent convex problem is regularized via a group sparsity inducing norm. Thus, a path regularized parallel ReLU network can be viewed as a parsimonious convex model in high dimensions. More importantly, since the original training problem may not be trainable in polynomial-time, we propose an approximate algorithm with a fully polynomial-time complexity in all data dimensions. Then, we prove strong global optimality guarantees for this algorithm. We also provide experiments corroborating our theory.



(a) 2-layer NN with WD (b) 3-layer NN with WD (c) 3-layer NN with PR (**Ours**)

Figure 1: Decision boundaries of 2-layer and 3-layer ReLU networks that are globally optimized with weight decay (WD) and path regularization (PR). Here, our convex training approach in (c) successfully learns the underlying spiral pattern for each class while the previously studied convex models in (a) and (b) fail (see Appendix A.1 for details).

1 Introduction

Deep Neural Networks (DNNs) have achieved substantial improvements in several fields of machine learning. However, since DNNs have a highly nonlinear and non-convex structure, the fundamental principles behind their remarkable performance is still an open problem. Therefore, advances in

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

this field largely depend on heuristic approaches. One of the most prominent techniques to boost the generalization performance of DNNs is regularizing layer weights so that the network can fit a function that performs well on unseen test data. Even though weight decay, i.e., penalizing the ℓ_2^2-norm of the layer weights, is commonly employed as a regularization technique in practice, recently, it has been shown that ℓ_2-path regularizer [1], i.e., the sum over all paths in the network of the squared product over all weights in the path, achieves further empirical gains [2]. Therefore, in this paper, we investigate the underlying mechanisms behind path regularized DNNs through the lens of convex optimization.

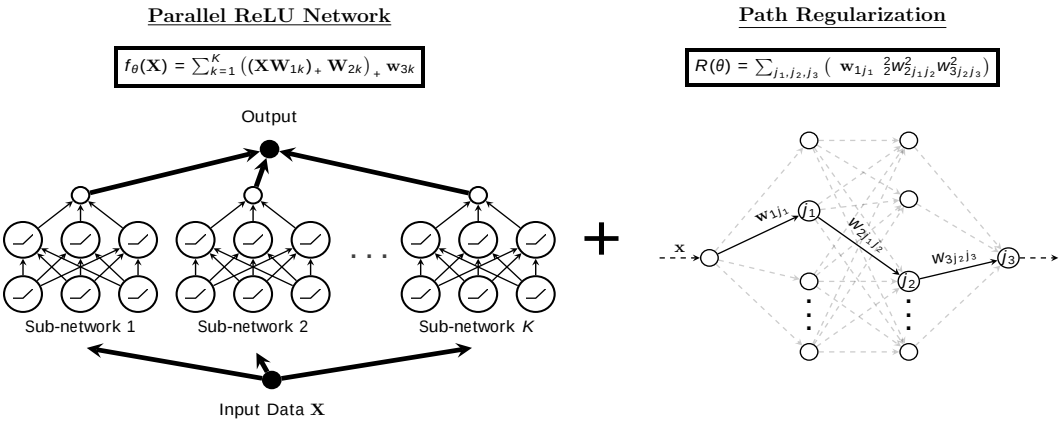


Figure 2: **(Left):** Parallel ReLU network in (1) with K sub-networks and three layers ($L = 3$) **(Right):** Path regularization for a three-layer network.

2 Parallel Neural Networks

Although DNNs are highly complex architectures due to the composition of multiple nonlinear functions, their parameters are often trained via simple first order gradient based algorithms, e.g., Gradient Descent (GD) and variants. However, since such algorithms only rely on local gradient of the objective function, they may fail to globally optimize the objective in certain cases [3, 4]. Similarly, [5, 6] showed that these pathological cases also apply to stochastic algorithms such as Stochastic GD (SGD). They further show that some of these issues can be avoided by increasing the number of trainable parameters, i.e., operating in an overparameterized regime. However, [7] reported the existence of more complicated cases, where SGD/GD usually fails. Therefore, training DNNs to global optimality remains a challenging optimization problem [8–10].

To circumvent difficulties in training, recent studies focused on models that benefit from overparameterization [11–14]. As an example, [15–19] considered a new architecture by combining multiple standard NNs, termed as sub-networks, in parallel. Evidences in [16–21] showed that this way of combining NNs yields an optimization landscape that has fewer local minima and/or saddle points so that SGD/GD generally converges to a global minimum. Therefore, many recently proposed NN-based architectures that achieve state-of-the-art performance in practice, e.g., SqueezeNet [22], Inception [23], Xception [24], and ResNext [25], are in this form.

Notation and preliminaries: Throughout the paper, we denote matrices and vectors as uppercase and lowercase bold letters, respectively. For vectors and matrices, we use subscripts to denote a certain column/element. As an example, $w_{lkj_l\ 1j_l}$ denotes the $j_l\ 1j_l^{th}$ entry of the matrix $\mathbf{W}_{lk}$. We use $\mathbf{I}_k$ and $\mathbf{0}$ (or $\mathbf{1}$) to denote the identity matrix of size $k \quad k$ and a vector/matrix of zeros (or ones) with appropriate sizes. We use $[n]$ for the set of integers from 1 to n. We use $k \quad k_2$ and $k \quad k_F$ to represent the Euclidean and Frobenius norms, respectively. Additionally, we denote the unit ℓ_p ball as $B_p := f\mathbf{u} \ 2 \ \mathbb{R}^d : k\mathbf{u}k_p \quad 1g$. We also use $\mathbb{1}[x \quad 0]$ and $(x)_+ = \max fx; 0g$ to denote the 0-1 valued indicator and ReLU, respectively.

In this paper, we particularly consider a parallel ReLU network with K sub-networks and each sub-network is an L-layer ReLU network (see Figure 2) with layer weights $\mathbf{W}_{lk} \ 2 \ \mathbb{R}^{m_l \ 1 \ m_l}$,

Table 1: Complexity comparison with prior works (n: # of data samples, d: feature dimension, m_l: # of hidden neurons in layer l, : approximation accuracy, r: rank of the data, r: chosen according to (10) such that accuracy is achieved)

	Loss function	2-layer complexity	L-layer complexity	Globally optimal
[26]	Convex loss	$O\left(2^{m_1} n^{dm_1} poly(n; d; m_1)\right)$	-	✓(Brute-force)
[27]	ℓ_2-loss	$2^{O(\frac{m}{2})} poly(n; d)$	-	✗(NP-hard)
[28]	ℓ_p-loss	$O\left(2^{m_1} n^{dm_1} poly(n; d; m_1)\right)$	-	✓(Brute-force)
[29]	Convex loss	$O(n^r poly(d; r))$	-	✓(Convex, exact)
Ours	Convex loss	$O(n^r poly(d; r))$	$O\left(n^{r \prod_{j=1}^{L-2} m_j} poly(d; r; \prod_{j=1}^{L-2} m_l)\right)$	✓(Convex, exact)
Ours	Convex loss	$O(n \ poly(d; \ ))$	$O\left(n^{\prod_{j=1}^{L-2} m_j} poly(d; \ ; \prod_{j=1}^{L-2} m_l)\right)$	✓(Convex, -opt)

$8l \in [L \quad 1]$ and $\mathbf{w}_{Lk} \in \mathbb{R}^{m_{L-1}}$, where $m_0 = d$, $m_L = 1$[1], and m_l denotes the number of neurons in the l^{th} hidden layer. Then, given a data matrix $\mathbf{X} \in \mathbb{R}^{n \quad d}$, the output of the network is as follows

$$f(\mathbf{X}) := \sum_{k=1}^{K} \left((\mathbf{X}\mathbf{W}_{1k})_+ ::: \mathbf{W}_{(L \quad 1)k}\right)_+ \mathbf{w}_{Lk}; \tag{1}$$

where we compactly denote the parameters as $:= \bigcup_k \{\mathbf{W}_{lk}\}_{l=1}^{L}$ with the parameter space as and each sub-network represents a standard deep ReLU network.

Remark 1. *Most commonly used neural networks in practice can be classified as special cases of parallel networks, e.g., standard NNs and ResNets [30] see Appendix A.2 for details.*

2.1 Our Contributions

- We prove that training the path regularized parallel ReLU networks (1) is equivalent to a convex optimization problem that can be approximately solved in polynomial-time by standard convex solvers (see Table 1). Therefore, we generalize the two-layer results in [29] to multiple nonlinear layer without any strong assumptions in contrast to [17] and a much broader class of NN architectures including ResNets.
- As already observed by [17, 29], regularized deep ReLU network training problems require exponential-time complexity when the data matrix is full rank, which is unavoidable. However, in this paper, we develop an approximate training algorithm which has fully polynomial-time complexity in all data dimensions and prove global optimality guarantees in Theorem 2. To the best of our knowledge, this is the first convex optimization based and polynomial-time complexity (in data dimensions) training algorithm for ReLU networks with global approximation guarantees.
- We show that the equivalent convex problem is regularized by a group norm regularization where grouping effect is among the sub-networks. Therefore the equivalent convex formulation reveals an implicit regularization that promotes group sparsity among sub-networks and generalizes prior works on linear networks such as [31] to ReLU networks.
- We derive a closed-form mapping between the parameters of the non-convex parallel ReLU networks and its convex equivalent in Proposition 1. Therefore, instead of solving the challenging non-convex problem, one can globally solve the equivalent convex problem and then construct an optimal solution to the original non-convex network architecture via our closed-form mapping.

2.2 Overview of Our Results

Given data $\mathbf{X} \in \mathbb{R}^{n \quad d}$ and labels $\mathbf{y} \in \mathbb{R}^n$, we consider the following regularized training problem

$$p_L := \min_{\in} L(f(\mathbf{X}); \mathbf{y}) + \quad R(\ ); \tag{2}$$

where $:= \{f \quad \in \quad : \mathbf{W}_{lk} \in \mathbb{R}^{m_{l-1} \quad m_l}; 8l \in [L]; 8k \in [K]\}$ is the parameter space, $L(\ ;\ )$ is an arbitrary convex loss function, $R(\ )$ represents the regularization on the network weights, and > 0 is the regularization coefficient.

[1] We analyze scalar outputs, however, our derivations extend to vector outputs as shown in Appendix A.11.

For the rest of the paper, we focus on a scalar output regression/classification framework with arbitrary loss functions, e.g., squared loss, cross entropy or hinge loss. We also note that our derivations can be straightforwardly extended to vector outputs networks as proven in Appendix A.11. More importantly, we use ℓ_2-path regularizer studied in [1, 2], which is defined as

$$R(\theta) := \sum_{k=1}^{K} \sqrt{\sum_{j_1, j_2, \ldots, j_L} \left(\|\mathbf{w}_{1kj_1}\|_2^2 \prod_{l=2}^{L} w_{lkj_{l-1}j_l}^2 \right)},$$

where $w_{lkj_{l-1}j_l}$ is the $j_{l-1}j_l^{th}$ entry of $\mathbf{W}_{lk}$. The above regularizer sums the square of all the parameters along each possible path from input to output of each sub-network k (see Figure 2) and then take the squared root of the summation. Therefore, we penalize each path in each sub-network and then group them based on the sub-network index k.

We now propose a scaling to show that (2) is equivalent to a group ℓ_1 regularized problem.

Lemma 1. *The following problems are equivalent* [2]*:*

$$\min_{\beta} L(f(\mathbf{X}), \mathbf{y}) + \beta R(\theta) = \min_{\beta} L(f(\mathbf{X}), \mathbf{y}) + \beta \sum_{k=1}^{K} \|\mathbf{w}_{Lk}\|_2,$$

where $\mathbf{w}_{Lk} \in \mathbb{R}^{m_{L-1}}$ *are the last layer weights of each sub-network* k*, and* $\theta_s := \{f : \sum_{j_1, j_2, \ldots, j_{L-2}} \left(\|\mathbf{w}_{1kj_1}\|_2^2 \prod_{l=2}^{L-1} w_{lkj_{l-1}j_l}^2 \right) \leq 1, \forall j_{L-1}, \forall k \in [K]\}$ *denotes the parameter space after rescaling.*

The advantage of the form in Lemma 1 is that we can derive a dual problem with respect to the output layer weights $\mathbf{w}_{Lk}$ and then characterize the optimal layer weights via optimality conditions and the prior works on ℓ_1 regularization in infinite dimensional spaces [32]. Thus, we first apply the rescaling in Lemma 1 and then take the dual with respect to the output weights $\mathbf{w}_{Lk}$. To characterize the hidden layer weights, we then change the order of minimization for the hidden layer weights and the maximization for the dual parameter to get the following dual problem[3]

$$p_L \geq d_L := \max_{\mathbf{v}} -L^*(\mathbf{v}) \quad \text{s.t.} \quad \max_{\theta_s} \left\| \mathbf{v}^T \left((\mathbf{X}\mathbf{W}_1)_+ \; \vdots \; \mathbf{W}_{(L-1)} \right)_+ \right\|_2 \leq \beta, \tag{3}$$

where L^* is the Fenchel conjugate function of L, which is defined as [33]

$$L^*(\mathbf{v}) := \max_{\mathbf{z}} \mathbf{z}^T \mathbf{v} - L(\mathbf{z}, \mathbf{y}).$$

The dual problem in (3) is critical for our derivations since it provides us with an analytic perspective to characterize a set of optimal hidden layer weights for the non-convex neural network in (1). To do so, we first show that strong duality holds for the non-convex training problem in Lemma 1, i.e., $p_L = d_L$. Then, based on the exact dual problem in (3), we propose an equivalent analytic description for the optimal hidden layer weights via the KKT conditions.

We note that strong duality for two-layer ReLU networks has already been proved by previous studies [15, 17, 18, 29, 34, 35], however, this is the first work providing an exact characterization for path regularized deep ReLU networks via convex duality.

3 Parallel networks with three layers

Here, we consider a three-layer parallel network with K sub-networks, which is a special case of (1) when $L = 3$. Thus, we have the following training problem

$$p_3 = \min_{\beta} L(f(\mathbf{X}), \mathbf{y}) + \beta \sum_{k=1}^{K} \sqrt{\sum_{j_1, j_2} \|\mathbf{w}_{1kj_1}\|_2^2 w_{2kj_1j_2}^2 w_{3kj_2}^2}. \tag{4}$$

[2] All the proofs are presented in the supplementary file.

[3] We present the details in Appendix A.7.

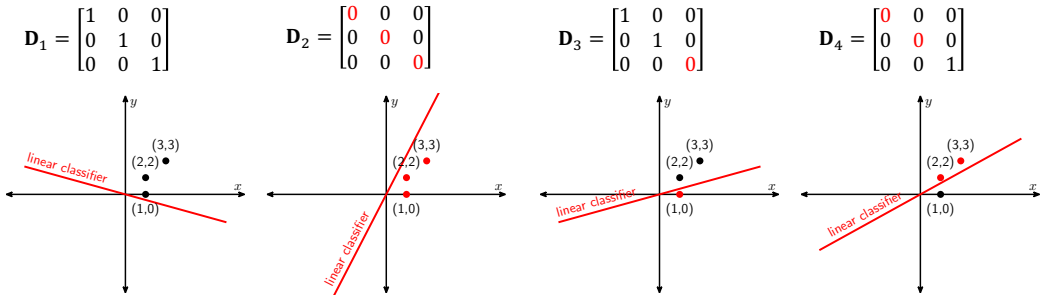


Figure 3: Illustration of possible hyperplane arrangements that determine the diagonal matrices $\mathbf{D}_i$. Here, we have three samples in two dimensions and we want to separate these samples with a linear classifier. $\mathbf{D}_i$ basically encodes information regarding which side of the linear classifier samples lie.

where $= f(\mathbf{W}_{1k}; \mathbf{W}_{2k}; \mathbf{w}_{3k}) : \mathbf{W}_{1k} \in \mathbb{R}^{d \times m_1}; \mathbf{W}_{2k} \in \mathbb{R}^{m_1 \times m_2}; \mathbf{w}_{3k} \in \mathbb{R}^{m_2}\}$. By Lemma 1,

$$p_3 = \min_{\beta} L(f(\mathbf{X}); \mathbf{y}) + \sum_{k=1}^{K} \|\mathbf{w}_{3k}\|_2 : \tag{5}$$

Then, taking the dual of (5) with respect to the output layer weights $\mathbf{w}_{3k} \in \mathbb{R}^{m_2}$ and then changing the order of the minimization for $\{\mathbf{W}_{1k}; \mathbf{W}_{2k}\}$ and the maximization for the dual variable $\mathbf{v}$ yields

$$p_3 \geq d_3 := \max_{\mathbf{v}} L(\mathbf{v}) \quad \text{s.t.} \quad \max_{\beta} \left\| \mathbf{v}^T \left((\mathbf{X}\mathbf{W}_1)_+ \mathbf{W}_2 \right)_+ \right\|_2 \leq : \tag{6}$$

Here, we remark that (4) is non-convex with respect to the layer weights, we may have a duality gap, i.e., $p_3 \geq d_3$. Therefore, we first show that strong duality holds in this case, i.e., $p_3 = d_3$ as detailed in Appendix A.4. We then introduce an equivalent representation for the ReLU activation as follows.

Since ReLU masks the negative entries of inputs, we have the following equivalent representation

$$\begin{aligned} \left((\mathbf{X}\mathbf{W}_1)_+ \mathbf{w}_2 \right)_+ &= \left(\sum_{j_1=1}^{m_1} (\mathbf{X}\mathbf{w}_{1j_1})_+ w_{2j_1} \right)_+ = \left(\sum_{j_1=1}^{m_1} (\mathbf{X}\mathbf{w}_{j_1})_+ I_{j_1} \right)_+ \\ &= \mathbf{D}_2 \sum_{j_1=1}^{m_1} I_{j_1} \mathbf{D}_{1j_1} \mathbf{X}\mathbf{w}_{j_1}; \end{aligned} \tag{7}$$

where $\mathbf{w}_{j_1} = \|w_{2j_1}\| \mathbf{w}_{1j_1}$, $I_{j_1} = \text{sign}(w_{2j_1}) \in \{-1; +1\}$, and we use the following alternative representation for ReLU (see Figure 3 for a two dimensional visualization)

$$(\mathbf{X}\mathbf{w})_+ = \mathbf{D}\mathbf{X}\mathbf{w} \iff \begin{matrix} \mathbf{D}\mathbf{X}\mathbf{w} \geq 0 \\ (\mathbf{I}_n - \mathbf{D})\mathbf{X}\mathbf{w} \leq 0 \end{matrix} \iff (2\mathbf{D} - \mathbf{I}_n)\mathbf{X}\mathbf{w} \geq 0;$$

where $\mathbf{D} \in \mathbb{R}^{n \times n}$ is a diagonal matrix of zeros/ones, i.e., $\mathbf{D}_{ii} \in \{0; 1\}$. Therefore, we first enumerate all possible signs and diagonal matrices for the ReLU layers and denote them as I_{j_1}, $\mathbf{D}_{1ij_1}$ and $\mathbf{D}_{2l}$ respectively, where $j_1 \in [m_1]; i \in [P_1]; l \in [P_2]$. Here, $\mathbf{D}_{1ij_1}$ and $\mathbf{D}_{2l}$ denotes the masking/diagonal matrices for the first and second ReLU layers, respectively and P_1 and P_2 are the number diagonal matrices in each layer as detailed in Section A.10. Then, we convert the non-convex constraints in (6) to convex constraints using $\mathbf{D}_{1ij_1}$, $\mathbf{D}_{2l}$ and I_{j_1}.

Using the representation in (7), we then take the dual of (6) to obtain the convex bidual of the primal problem (4) as detailed in the theorem below.

Theorem 1. *The non-convex training problem in (4) can be cast as the following convex program*

$$\min_{\mathbf{z}; \mathbf{z}' \in C} L\left(\mathbf{X}(\mathbf{z} - \mathbf{z}'); \mathbf{y} \right) + \beta \sqrt{m_2} (\|\mathbf{z}\|_{F;1} + \|\mathbf{z}'\|_{F;1}); \tag{8}$$

where $\|\cdot\|_{F;1}$ denotes a $d \times m_1$ dimensional group Frobenius norm operator such that given a vector $\mathbf{u} \in \mathbb{R}^{dm_1 P}$, $\|\mathbf{u}\|_{F;1} := \sum_{i=1}^{P} \|\mathbf{U}_i\|_F$, where $\mathbf{U}_i \in \mathbb{R}^{d \times m_1}$ are reshaped partitions of $\mathbf{u}$. Moreover,

the convex set C is defined as

$$C := \{\mathbf{z} : \mathbf{z}^s_{ij_1l} \in C^s_{il}, \forall i \in [P_1], l \in [P_2], s \in [M]\}$$

$$C^s_{il} := \left\{ \{\mathbf{w}_{j_1}, g_{j_1}\} : (2\mathbf{D}_{2l} - \mathbf{I}_n) \sum_{j_1=1}^{m_1} I^s_{ij_1l}\mathbf{D}_{1ij_1}\mathbf{X}\mathbf{w}_{j_1} \geq 0, (2\mathbf{D}_{1ij_1} - \mathbf{I}_n)\mathbf{X}\mathbf{w}_{j_1} \geq 0, \forall j_1 \in [m_1] \right\}$$

where $I^s_{ij_1l} \in \{+1, -1\}$, $M = 2^{m_1}$, and $\mathbf{z}, \mathbf{z}^0 \in \mathbb{R}^{dm_1MP_1P_2}$ are constructed by stacking $\mathbf{z}^s_{ij_1l}, \mathbf{z}^{s^0}_{ij_1l} \in \mathbb{R}^d$, $\forall i \in [P_1], l \in [P_2], j_1 \in [m_1], s \in [M]$, respectively. Also, the effective data matrix $\mathbf{X} \in \mathbb{R}^{n \times dm_1MP_1P_2}$ is defined as $\mathbf{X} := \mathbf{I}_M \otimes \mathbf{X}_s$ and

$$\mathbf{X}_s := [\mathbf{D}_{21}\mathbf{D}_{111}\mathbf{X} \ \vdots\vdots\vdots \ \mathbf{D}_{2l}\mathbf{D}_{1ij_1}\mathbf{X} \ \vdots\vdots\vdots \ \mathbf{D}_{2P_2}\mathbf{D}_{1P_1m_1}\mathbf{X}] .$$

We next derive a mapping between the convex program (8) and the non-convex architecture (4).

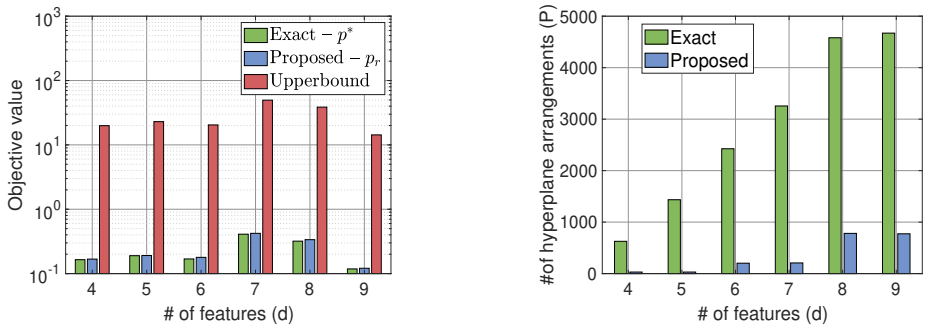
Proposition 1. *An optimal solution to the non-convex parallel network training problem in (4), denoted as $\{\mathbf{W}_{1k}, \mathbf{w}_{2k}, w_{3k}\}_{k=1}^K$, can be recovered from an optimal solution to the convex program in (8), i.e., $\{\mathbf{z}^*, \mathbf{z}^{0*}\}$ via a closed-form mapping. Therefore, we prove a mapping between the parameters of the parallel network in Figure 2 and its convex equivalent.*

Next, we prove that the convex program in (8) can be globally optimized with a polynomial-time complexity given $\mathbf{X}$ has fixed rank, i.e., $\text{rank}(\mathbf{X}) = r < \min\{n, d\}$.

Proposition 2. *Given a data matrix such that $\text{rank}(\mathbf{X}) = r < \min\{n, d\}$, the convex program in (8) can be globally optimized via standard convex solvers with $O(d^3 m_1^3 m_2^3 2^{3(m_1+1)m_2} n^{3(m_1+1)r})$ complexity, which is a polynomial-time complexity in terms of n, d. Note that here globally optimizing the training objective means to achieve the exact global minimum up to any arbitrary machine precision or solver tolerance.*

Below, we show that the complexity analysis in Proposition 2 extends to arbitrarily deep networks.

Corollary 1. *The same analysis can be readily applied to arbitrarily deep networks. Therefore, given $\text{rank}(\mathbf{X}) = r < \min\{n, d\}$, we prove that L-layer architectures can be globally optimized with $O\left(d^3 \left(\prod_{j=1}^{L-2} m_j^3\right) 2^{3 \sum_{l=1}^{L-1} \prod_{j=1}^{l} m_j} n^{3r(1+\sum_{l=1}^{L-2} \prod_{j=1}^{l} m_j)}\right)$, which is polynomial in n, d.*



(a) Objective value (#) (b) # of hyperplane arrangements (P)

Figure 4: Verification of Theorem 2 and Remark 2. We train a parallel network using the convex program in Theorem 1 with ℓ_2 loss on a toy dataset with $n = 15$, $\beta = 0.1$, $m_2 = 1$, and the low-rank approximation $r = \lfloor \frac{d}{2} \rfloor$. To obtain a low-rank model, we first sample a data matrix from a standard normal Gaussian distribution and then set $\sigma_{r+1} = \cdots = \sigma_d = 1$. Please see the subsection for low rank models in Section 4 for more details.

3.1 Polynomial-time training for arbitrary data

Based on Corollary 1, exponential complexity is unavoidable for deep networks when the data matrix is full rank, i.e., $\text{rank}(\mathbf{X}) = \min\{n, d\}$. Thus, we propose a low rank approximation to the model in (4). We first denote the rank-r approximation of $\mathbf{X}$ as $\hat{\mathbf{X}}_r$ such that $\|\mathbf{X} - \hat{\mathbf{X}}_r\|_2 \leq \sigma_{r+1}$, where σ_{r+1} represents the $(r+1)^{th}$ largest singular value of $\mathbf{X}$. Then, we have the following result.

Theorem 2. *Given an R-Lipschitz convex loss function $L(\,;\mathbf{y})$, the regularized training problem*

$$p_3 = \min_{\beta} L(f(\mathbf{X});\mathbf{y}) + \sum_{k=1}^{K} \sqrt{\sum_{j_1;j_2} k\mathbf{w}_{1kj} k_2^2 w_{2kj_1j_2}^2 w_{3kj_2}^2}; \tag{9}$$

can be solved using the data matrix $\hat{\mathbf{X}}_r$ to achieve the following optimality guarantee

$$p_3 \quad p_r \quad p_3 \left(1 + \frac{\sqrt{m_1 m_2} R \;\; r+1}{}\right)^2; \tag{10}$$

where p_r denotes the objective value achieved by the parameters trained using $\hat{\mathbf{X}}_r$.

Remark 2. *Theorem 1 and 2 imply that for a given arbitrary rank data matrix $\mathbf{X}$, the regularized training problem in (4) can be approximately solved by convex solvers to achieve a worst-case approximation $p_3 \left(1 + \frac{\sqrt{m_1 m_2} R \;\; r+1}{}\right)^2$ with complexity $O(d^3 m_1^3 2^{3(m_1+1)} n^{3(m_1+1)r})$, where $r \min\{n; d\}$. Therefore, even for full rank data matrices where the complexity is exponential in n or d, one can approximately solve the convex program in (8) in polynomial-time. Moreover, we remark that the approximation error proved in Theorem 2 can be arbitrarily small for practically relevant problems. As an example, consider a parallel network training problem with ℓ_2 loss function, then the upperbound becomes $(1 + \frac{\sqrt{m_1 m_2} \;\; r+1}{})^2$, which is typically close to one due to fast decaying singular values in practice (see Figure 4).*

3.2 Representational power: Two versus three layers

Here, we provide a complete explanation for the representational power of three-layer networks by comparing with the two-layer results in [29]. We first note that three-layer networks have substantially higher expressive power due to the non-convex interactions between hidden layers as detailed in [36, 37]. Furthermore, [38] show that layerwise training of three-layer networks can achieve comparable performance to deeper models, e.g., VGG-11, on Imagenet. There exist several studies analyzing two-layer networks, however, despite their empirical success, a full theoretical understanding and interpretation of three-layer networks is still lacking in the literature. In this work, we provide a complete characterization for three-layer networks through the lens of convex optimization theory. To understand their expressive power, we compare our convex program for three-layer networks in (8) with its two-layer counterpart in [29].

[29] analyzes two-layer networks with one ReLU layer, so that the data matrix $\mathbf{X}$ is multiplied with a single diagonal matrix (or hyperplane arrangement) $\mathbf{D}_i$. Thus, the effective data matrix is in the form of $\mathbf{X}_s = [\mathbf{D}_1\mathbf{X} \;\vdots\; \mathbf{D}_P\mathbf{X}]$. However, since our convex program in (8) has two nonlinear ReLU layers, the composition of these two-layer can generate substantially more complex features via locally linear variables $\{\mathbf{w}_{ij_1}^s\}$ multiplying the d-dimensional blocks of the columns of the effective data matrix $\mathbf{X}_s$ in Theorem 1. Although this may seem similar to the features in [17], here, we have 2^{m_1} variables for each linear region unlike [17] which employ 2 variables per linear region. Moreover, [17] only considers the case where the second hidden layer has only one neuron, i.e., $m_2 = 1$, therefore do not consider standard three layer or deeper networks. Hence, we exactly describe the impact of having one more ReLU layer and its contribution to the representational power of the network.

4 Experiments

In this section[4], we present numerical experiments corroborating our theory.

Low rank model in Theorem 2: To validate our claims, we generate a synthetic dataset as follows. We first randomly generate a set of layer weights for a parallel ReLU network with $K = 5$ sub-networks by sampling from a standard normal distribution. We then obtain the labels as $\mathbf{y} = \sum_k \left((\mathbf{X}\mathbf{W}_{1k})_+ \mathbf{W}_{2k}\right)_+ \mathbf{w}_{3k} + 0.1\,\varepsilon$, where $\varepsilon \sim N(0; \mathbf{I}_n)$. To promote a low rank structure in the data, we first sample a matrix from the standard normal distribution and then set $_{r+1} = \vdots = {}_{d} = 1$.

[4] Details on the experiments can be found in Appendix A.1.

Table 2: Training objective of a three-layer parallel network trained with non-convex SGD (5 independent initialization trials) on a toy dataset with $(n; d; m_1; m_2; \quad ;$ batch size$) = (5; 2; 3; 1; 0.002; 5)$, where the convex program in (8) are solved via the interior point solvers in CVX/CVXPY.

Method		**Non-convex SGD**					**Convex(Ours)**
		Run #1	**Run #2**	**Run #3**	**Run #4**	**Run #5**	
$K = 5$	Training objective	0.0221	0.0239	0.0031	0.0027	0.0027	**0.0007**
	Time(s)	11:62	11:62	11:62	11:62	11:62	**4:947**
$K = 20$	Training objective	0.0010	0.0027	0.0009	0.0010	0.0027	**0.0007**
	Time(s)	44:37	44:37	44:37	11:55	44:37	**4:947**
$K = 40$	Training objective	0.0008	0.0009	0.0009	0.0009	0.0008	**0.0007**
	Time(s)	91:87	91:87	91:87	91:87	91:87	**4:947**

We consider a regression framework with ℓ_2 loss and $(n; r; \quad ; m_1; m_2) = (15; d{=}2; 0.1; 1; 1)$ and present the numerical results in Figure 4. Here, we observe that the low rank approximation of the objective p_r is closer to p_3 than the worst-case upper-bound predicted by Theorem 2. However, in Figure 4b, the low rank approximation provides a significant reduction in the number of hyperplane arrangements, and therefore in the complexity of solving the convex program.

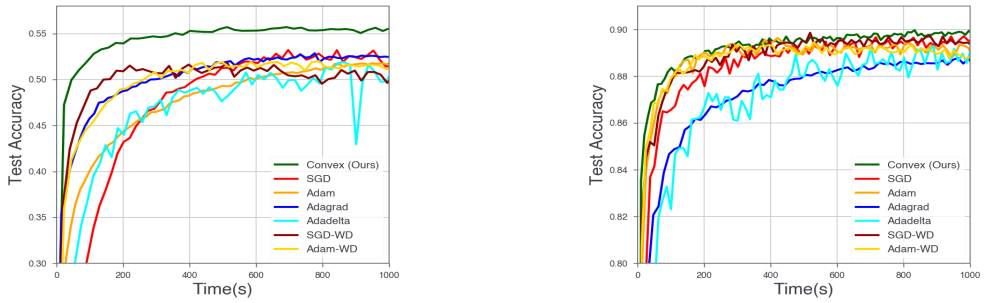
Toy dataset: We use a toy dataset with 5 samples and 2 features, i.e., $(n; d) = (5; 2)$. To generate the dataset, we forward propagate i.i.d. samples from a standard normal distribution, i.e., $\mathbf{x}_i \quad N(\mathbf{0}; \mathbf{I}_d)$, through a parallel network with 3 layers, 5 sub-networks, and 3 neurons, i.e., $(L; K; m_1; m_2) = (3; 5; 3; 1)$. We then train the parallel network in (4) on this toy dataset using both our convex program (8) and non-convex SGD. We provide the training objective and wall-clock time in Table 2, where we particularly include 5 initialization trials for SGD. This experiment shows that when the number of sub-networks K is small, SGD trials fail to converge to the global minimum achieved by our convex program. However, as we increase K, the number of trials converging to global minimum gradually increases. Therefore, we show the benign overparameterization impact.

Image classification: We conduct experiments on benchmark image datasets, namely CIFAR-10 [39] and Fashion-MNIST [40]. We particularly consider a ten class classification task and use a parallel network with 40 sub-networks and 100 hidden neurons, i.e., $(K; m_1; m_2) = (40; 100; 1)$. In Figure 5, we plot the test accuracies against wall-clock time, where we include several different optimizers as well as SGD. Moreover, we include a parallel network trained with SGD/Adam and Weight Decay (WD) regularization to show the effectiveness of path regularization in (4). We first note that our convex approach achieves both faster convergence and higher final test accuracies for both dataset. However, the performance gain for Fashion-MNIST seems to be significantly less compared to the CIFAR-10 experiment. This is due to the nature of these datasets. More specifically, since CIFAR-10 is a much more challenging dataset, the baseline accuracies are quite low (around 50%) unlike Fashion-MNIST with the baseline accuracies around 90%. Therefore, the accuracy improvement achieved by the convex program seems low in Figure 5b. We also observe that weight decay achieves faster convergence rates however path regularization yields higher final test accuracies. It is normal to have faster convergence with weight decay since it can be incorporated into gradient-based updates without any computational overhead.

5 Related Work

Parallel neural networks were previously investigated by [18, 19]. Although these studies provided insights into the solutions, they require assumptions, e.g., sparsity among sub-networks in Theorem 1 of [19]) and linear activations and hinge loss assumptions in [18], which invalidates applications in practice.

Recently, [29] studied weight decay regularized two-layer ReLU network training problems and introduced polynomial-time trainable convex formulations. However, their analysis is restricted to standard two-layer ReLU networks, i.e., in the form of $f(\mathbf{X}) = (\mathbf{X}\mathbf{W}_1)_+ \mathbf{w}_2$. The reasons for this restriction is that handling more than one ReLU layer is a substantially more challenging optimization problem. As an example, a direct extension of [29] to three-layer NNs will yield doubly exponential complexity, i.e., $O(n^{rn^r})$ for a rank-r data matrix, due to the combinatorial behavior of multiple



(a) CIFAR-10 (b) Fashion-MNIST

Figure 5: Accuracy of a three-layer architecture trained using the non-convex formulation (4) and the proposed convex program (8), where we use **(a)** CIFAR-10 with $(n; d; m_1; m_2; K; \ ;\text{batch size}) = (5\text{x}10^4; 3072; 100; 1; 40; 10^{-3}; 10^3)$ and **(b)** Fashion-MNIST with $(n; d; m_1; m_2; K; \ ;\text{batch size}) = (6\text{x}10^4; 784; 100; 1; 40; 10^{-3}; 10^3)$. We note that the convex model is trained using **(a)** SGD and **(b)** Adam.

ReLU layers. Thus, they only examined the case with a single ReLU layer (see Table 1 for details and the other relevant references in [29]). In addition, since they only considered standard two-layer ReLU networks, their analysis is not valid for a broader range of NN-based architectures as detailed in Remark 1. Later on, [17] extended this approach to three-layer ReLU networks. However, since they analyzed ℓ_2-norm regularized training problem, they had to put unit Frobenius norm constraints on the first layer weights, which does not reflect the settings in practice. In addition, their analysis is restricted to the networks with a single neuron in the second layer (i.e. $m_2 = 1$) that is in the form of $f(\mathbf{X}) = \sum_k \left((\mathbf{X}\mathbf{W}_{1k})_+ \mathbf{w}_{2k}\right)_+ w_{3k}$. Since this architecture only allows a single neuron in the second layer, each sub-network k has an expressive power that is equivalent to a standard two-layer network rather than three-layer. This can also be realized from the definition of the constraint set $\mathcal{C}$ in Theorem 1. Specifically, the convex set $\mathcal{C}$ in [17] has decoupled constraints across the hidden layer index j_1 whereas our formulation sums the responses over hidden neurons before feeding through the next layer as standard deep networks do. Therefore, this analysis does not reflect the true power of deep networks with $L > 2$. Moreover, the approach in [17] has exponential complexity when the data matrix has full rank, which is unavoidable.

However, we analyze deep neural networks in (1) without any assumption on the weights. Furthermore, we develop an algorithm which has fully polynomial-time complexity in data dimensions and prove strong global optimality guarantees for this algorithm in Theorem 2.

6 Concluding Remarks

We studied the training problem of path regularized deep parallel ReLU networks, which includes ResNets and standard deep ReLU networks as its special cases. We first showed that the non-convex training problem can be equivalently cast as a single convex optimization problem. Therefore, we achieved the following advantages over the training on the original non-convex formulation: 1) Since our model is convex, it can be globally optimized via standard convex solvers whereas the non-convex formulation trained with optimizers such as SGD might be stuck at a local minimum, 2) Thanks to convexity, our model does not require any sort of heuristics and additional tricks such as learning rate schedule and initialization scheme selection or dropout. More importantly, we proposed an approximation to the convex program to enable fully polynomial-time training in terms of the number of data samples n and feature dimension d. Thus, we proved the polynomial-time trainability of deep ReLU networks without requiring any impractical assumptions unlike [17, 29]. Notice that we derive an exact convex program only for three-layer networks, however, recently [15] proved that strong duality holds for arbitrarily deep parallel networks. Therefore, a similar analysis can be extended to deeper networks to achieve an equivalent convex program, which is quite promising for future work. Additionally, although we analyzed fully connected networks in this paper, our approach can be

directly extended to various NN architectures, e.g., convolution networks [41], generative adversarial networks [42], NNs with batch normalization [43], and autoregressive models [44].

7 Acknowledgements

This work was supported in part by the National Science Foundation (NSF) CAREER Award under Grant CCF-2236829, Grant DMS-2134248 and Grant ECCS-2037304; in part by the U.S. Army Research Office Early Career Award under Grant W911NF-21-1-0242; in part by the Stanford Precourt Institute; and in part by the ACCESS—AI Chip Center for Emerging Smart Systems through InnoHK, Hong Kong, SAR.

References

[1] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In Peter Grünwald, Elad Hazan, and Satyen Kale, editors, *Proceedings of The 28th Conference on Learning Theory*, volume 40 of *Proceedings of Machine Learning Research*, pages 1376–1401, Paris, France, 03–06 Jul 2015. PMLR.

[2] Behnam Neyshabur, Russ R Salakhutdinov, and Nati Srebro. Path-sgd: Path-normalized optimization in deep neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[3] Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah. Failures of gradient-based deep learning. *arXiv preprint arXiv:1703.07950*, 2017.

[4] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[5] Rong Ge, Jason D. Lee, and Tengyu Ma. Learning one-hidden-layer neural networks with landscape design, 2017.

[6] Itay Safran and Ohad Shamir. Spurious local minima are common in two-layer relu neural networks. In *International Conference on Machine Learning*, pages 4433–4441. PMLR, 2018.

[7] Animashree Anandkumar and Rong Ge. Efficient approaches for escaping higher order saddle points in non-convex optimization. In *Conference on learning theory*, pages 81–102, 2016.

[8] Bhaskar DasGupta, Hava T Siegelmann, and Eduardo Sontag. On the complexity of training neural networks with continuous activation functions. *IEEE Transactions on Neural Networks*, 6(6):1490–1504, 1995.

[9] Avrim Blum and Ronald L Rivest. Training a 3-node neural network is np-complete. In *Advances in neural information processing systems*, pages 494–501, 1989.

[10] Peter Bartlett and Shai Ben-David. Hardness results for neural network approximation problems. In *European Conference on Computational Learning Theory*, pages 50–62. Springer, 1999.

[11] Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. SGD learns over-parameterized networks that provably generalize on linearly separable data. *CoRR*, abs/1710.10174, 2017.

[12] Simon S Du and Jason D Lee. On the power of over-parametrization in neural networks with quadratic activation. *arXiv preprint arXiv:1803.01206*, 2018.

[13] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *35th International Conference on Machine Learning, ICML 2018*, pages 372–389. International Machine Learning Society (IMLS), 2018.

[14] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.

[15] Yifei Wang, Tolga Ergen, and Mert Pilanci. Parallel deep neural networks have zero duality gap. In *The Eleventh International Conference on Learning Representations*, 2023.

[16] Tolga Ergen and Mert Pilanci. Revealing the structure of deep neural networks via convex duality. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3004–3014. PMLR, 18–24 Jul 2021.

[17] Tolga Ergen and Mert Pilanci. Global optimality beyond two layers: Training deep relu networks via convex programs. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2993–3003. PMLR, 18–24 Jul 2021.

[18] 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, 2019.

[19] Benjamin D Haeffele and René Vidal. Global optimality in neural network training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7331–7339, 2017.

[20] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[21] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in neural information processing systems*, pages 550–558, 2016.

[22] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[23] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[24] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[25] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[26] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

[27] Surbhi Goel, Adam Klivans, Pasin Manurangsi, and Daniel Reichman. Tight Hardness Results for Training Depth-2 ReLU Networks. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:14, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[28] Vincent Froese, Christoph Hertrich, and Rolf Niedermeier. The computational complexity of relu network training parameterized by data dimensionality, 2021.

[29] Mert Pilanci and Tolga Ergen. Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7695–7705. PMLR, 13–18 Jul 2020.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[31] Zhen Dai, Mina Karzand, and Nathan Srebro. Representation costs of linear neural networks: Analysis and design. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 26884–26896. Curran Associates, Inc., 2021.

[32] Saharon Rosset, Grzegorz Swirszcz, Nathan Srebro, and Ji Zhu. L1 regularization in infinite dimensional feature spaces. In *International Conference on Computational Learning Theory*, pages 544–558. Springer, 2007.

[33] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[34] Tolga Ergen and Mert Pilanci. Convex geometry and duality of over-parameterized neural networks. *Journal of Machine Learning Research*, 22(212):1–63, 2021.

[35] Francis Bach. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.

[36] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[37] Huy Tuan Pham and Phan-Minh Nguyen. Global convergence of three-layer neural networks in the mean field regime. In *International Conference on Learning Representations*, 2021.

[38] Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. In *International conference on machine learning*, pages 583–593. PMLR, 2019.

[39] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The CIFAR-10 dataset. http://www.cs.toronto.edu/kriz/cifar.html, 2014.

[40] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[41] Tolga Ergen and Mert Pilanci. Implicit convex regularizers of cnn architectures: Convex optimization of two- and three-layer networks in polynomial time. In *International Conference on Learning Representations*, 2021.

[42] Arda Sahiner, Tolga Ergen, Batu Ozturkler, Burak Bartan, John M. Pauly, Morteza Mardani, and Mert Pilanci. Hidden convexity of wasserstein GANs: Interpretable generative models with closed-form solutions. In *International Conference on Learning Representations*, 2022.

[43] Tolga Ergen, Arda Sahiner, Batu Ozturkler, John M. Pauly, Morteza Mardani, and Mert Pilanci. Demystifying batch normalization in reLU networks: Equivalent convex optimization models and implicit regularization. In *International Conference on Learning Representations*, 2022.

[44] Vikul Gupta, Burak Bartan, Tolga Ergen, and Mert Pilanci. Convex neural autoregressive models: Towards tractable, expressive, and theoretically-backed models for sequential forecasting and generation. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3890–3894, 2021.

[45] Daniel Smilkov Carter and Shan.

[46] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, March 2014.

[47] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[48] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[49] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[50] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15(90):3133–3181, 2014.

[51] Maurice Sion. On general minimax theorems. *Pacific J. Math.*, 8(1):171–176, 1958.

[52] Piyush C Ojha. Enumeration of linear threshold functions from the lattice of hyperplane intersections. *IEEE Transactions on Neural Networks*, 11(4):839–850, 2000.

[53] Richard P Stanley et al. An introduction to hyperplane arrangements. *Geometric combinatorics*, 13:389–496, 2004.

[54] RO Winder. Partitions of n-space by hyperplanes. *SIAM Journal on Applied Mathematics*, 14(4):811–818, 1966.

[55] Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, (3):326–334, 1965.

[56] Arda Sahiner, Tolga Ergen, John M. Pauly, and Mert Pilanci. Vector-output relu neural network problems are copositive programs: Convex analysis of two layer networks and polynomial-time algorithms. In *International Conference on Learning Representations*, 2021.

Supplementary Material

Table of Contents

A Appendix

A.1 Additional numerical results and details

In this section, we provide new numerical results and detailed information about our experiments in the main paper.

Decision boundary plots in Figure 1: In order to visualize the capabilities of our convex training approach, we perform an experiment on the spiral dataset which is known to be challenging for 2-layer networks while 3-layer networks can readily interpolate the training data (i.e. exactly fit the training labels) [45]. As the baselines of our analysis, we include the two-layer convex training approach in [29] and recently introduced three-layer convex training approach (with weight decay regularization and several architectural and parametric assumptions) in [17]. As our experimental setup, we consider a binary classification task with $y \in \{-1, 1\}^n$ and squared loss. We choose $(n, m_1, m_2, P_1, P_2, \ldots) = (30, 5, 1, 11, 11, 1e4)$ and for [29] we use $P = 50$ neurons/hyperplane arrangements. We also use CVPXY with MOSEK solver [46–48] to globally solve the convex programs. As demonstrated in Figure 1, baselines methods, especially [17], fit a function that is significantly different than the underlying spiral data distribution. This clearly shows that since [29] is restricted two-layer networks and [17] have multiple assumptions, i.e., unit Frobenius norm constraints on the layer weights ($\|\mathbf{W}_{1k}\|_F \leq 1, \forall l \in [L-2]$) and last hidden layer weights cannot be matrices ($\mathbf{w}_{(L-1)k} \in \mathbb{R}^{m_{L-2}}$), both baseline approaches fail to reflect true expressive power of deep networks ($L > 2$). On the other hand, our convex training approach for path regularized networks fits a model that successfully describes the underlying data distribution for this challenging task.

Additional experiments: We also conduct experiments on several datasets available in UCI Machine Learning Repository [49], where we particularly selected the datasets from [50] such that $n \leq 500$. For these datasets, we consider a conventional binary classification framework with $(m_1, m_2, K, \ldots) = (100, 1, 40, 0.5)$ and compare the test accuracies of non-convex architectures trained with SGD and Adam with their convex counter parts in (8). For these experiments, we use the 80% - 20% splitting ratio for the training and test sets. Furthermore, we train each algorithm long enough to reach training accuracy one. As shown in Table 3, our convex approach achieves higher or the same test accuracy compared to the standard non-convex training approach for most of the datasets (precisely 20 and 19 out of 21 datasets for SGD and Adam, respectively). We also note that for this experiment, we used the unconstrained form in (11) with the approximate version in Remark 3.3 of [29].

Details for the experiments in Table 2 and Figure 5: We first note that for the experiments in Table 2, we use CVX/CVPXY [46–48] to globally solve the proposed convex program in (8). For these

Table 3: Test accuracies for UCI experiments ($(m_1; m_2; K; \ ) = (100; 1; 40; 0{:}5)$ and 80% 20% training-test split). Here, we present the standard non-convex architectures and the proposed convex counterpart trained with SGD and Adam optimizers. If one approach achieves higher accuracy on a certain dataset, we display the corresponding accuracy value in bold font. We observe that our convex approach achieves either higher or the same accuracy for 20 and 19 datasets (out of 21 datasets) when trained with SGD and Adam, respectively

			SGD		**Adam**	
Dataset	n	d	**Non-convex**	**Convex(Ours)**	**Non-convex**	**Convex(Ours)**
acute-inflammation	120	6	1:000	1:000	1:000	1:000
acute-nephritis	120	6	1:000	1:000	1:000	1:000
balloons	16	4	1:000	1:000	1:000	1:000
breast-cancer	286	9	0:690	**0:707**	0:655	**0:672**
breast-cancer-wisc-prog	198	33	0:750	**0:800**	0:800	**0:825**
congressional-voting	435	16	0:667	0:667	0:551	**0:597**
conn-bench-sonar-mines-rocks	208	60	0:714	**0:786**	0:738	**0:833**
echocardiogram	131	10	0:704	0:704	0:666	**0:703**
fertility	100	9	0:750	**0:800**	0:800	0:800
haberman-survival	306	3	0:710	**0:774**	0:677	**0:709**
heart-hungarian	294	12	0:831	0:831	0:779	**0:813**
hepatitis	155	19	0:645	**0:710**	0:709	**0:741**
ionosphere	351	33	0:887	**0:901**	**0:929**	0:887
molec-biol-promoter	106	57	0:818	0:818	0:727	**0:772**
musk-1	476	166	**0:958**	0:927	**0:947**	0:927
parkinsons	195	22	0:974	**1:000**	0:923	**1:000**
pittsburg-bridges-T-OR-D	102	7	0:952	0:952	0:809	**0:857**
planning	182	12	0:541	**0:568**	0:649	**0:703**
statlog-heart	270	13	0:759	**0:796**	0:759	**0:833**
trains	10	29	1:000	1:000	1:000	1:000
vertebral-column-2clases	310	6	0:806	**0:839**	0:758	**0:822**
Highest test accuracy				20/21		19/21

experiments, we use a laptop with i7 processor and 16GB of RAM. In order to tune the learning rate of SGD/GD, we first perform training with a bunch of learning different learning rates and select the one with the best performance on the validation datasets, which is 0:005 in these experiments.

For comparatively larger scale image classification experiments in Figure 5, we utilize a cluster GPU with 50GB of memory. However since the equivalent convex program in (8) has constraint which are challenging to handle for these datasets, we propose the following unconstrained convex problem which has the same global minima with the constrained version as discussed in [44]

$$\min_{\mathbf{z};\mathbf{z}^0 2 \mathrm{R}^{d m_1 M P_1 P_2}} L\left(\mathbf{X}\left(\mathbf{z}^0 \quad \mathbf{z}\right); \mathbf{y}\right) + \quad (k\mathbf{z}k_{G;1} + k\mathbf{z}^0 k_{G;1}) + \quad (h_C(\mathbf{z}) + h_C(\mathbf{z}^0)) \tag{11}$$

where > 0 coefficient to penalize the violated constraints and $h_C(\mathbf{z})$ is a function to sum the absolute value of all constraint violations defined as

$$h_C(\mathbf{z}) := \mathbf{1}^T \sum_{i;j_1;l;s} \left(\left(\quad (2\mathbf{D}_{1ij_1} \quad \mathbf{I}_n)\mathbf{X}\mathbf{z}^s_{ij_1l} \right)_+ \right) + \mathbf{1}^T \sum_{i;l;s} \left(\quad \sum_{j_1} I^s_{ij_1l} (2\mathbf{D}_{2l} \quad \mathbf{I}_n)\mathbf{D}_{1ij_1}\mathbf{X}\mathbf{z}^s_{ij_1l} \right)_+ :$$

Thus, we obtain an unconstrained version of convex optimization problem (11), where one can use commonly employed first-order gradient based optimizers, e.g., SGD and Adam, available in deep learning libraries such PyTorch and Tensorflow. For both CIFAR-10 and Fashion-MNIST, we use the same training and test splits in the original datasets. We again perform a grid search to tune the learning rate, where the best performance is achieved by the following choices

$$\left({}_{Convex}; \ {}_{SGD}; \ {}_{Adam}; \ {}_{Adagrad}; \ {}_{Adadelta}; \ {}^{WD}_{SGD} \right) = (5e \quad 7; 5e \quad 3; 2e \quad 5; 2e \quad 3; 3e \quad 1; 1)$$

and

$$\left({}_{Convex}; \ {}_{SGD}; \ {}_{Adam}; \ {}_{Adagrad}; \ {}_{Adadelta}; \ {}^{WD}_{SGD} \right) = (1e \quad 5; 2e \quad 1; 2e \quad 3; 1e \quad 2; 3; 1)$$

as the learning rates for CIFAR-10 and Fashion-MNIST, respectively. We choose the momentum coefficient of the SGD optimizer as 0:9. In addition to this, we set $P_1 P_2 = K$ and $= 1e \quad 5$.

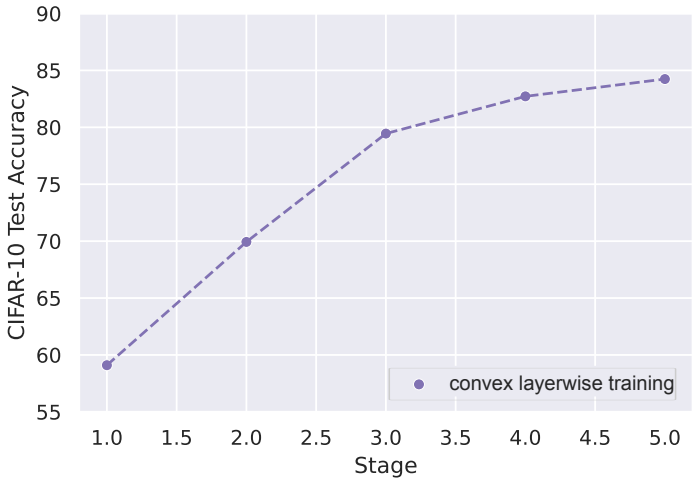


Figure 6: Test accuracy for convex layerwise training, where each stage is our three-layer convex formulation in Theorem 1. Here, we train three-layer neural networks sequentially using convex optimization to build deep networks.

More importantly, we remark that these experiments are performed by using a small sampled subset of hyperplane arrangements rather than sampling all possible arrangements as detailed in Remark 3.3 of [29]. In particular, we first generate random weight matrices from a multivariate standard normal distribution and then solve the convex program using only the arrangements of the sampled weight matrices.

Details for the experiments in Figure 6: Layerwise training with shallow neural networks was proven to work remarkably well. Particularly, [38] shows that one can train three-layer neural networks sequentially to build deep networks that outperform end-to-end training with SOTA architectures. In Figure 6, we apply this layerwise training procedure with our convex training approach. In particular, each stage in this figure is our three-layer convex formulation in Theorem 1. Here, we use the same experimental setting in the previous section. We observe that making the network deeper by stacking convex layers resulted in significant performance improvements. Specificaly, at the fifth stage, we achieved almost 85% accuracy for CIFAR-10 unlike below 60% accuracies in Figure 5.

A.2 Parallel ReLU networks

The parallel networks $f(\mathbf{X})$ models a wide range of NNs in practice. As an example, standard NNs and ResNets [30] are special cases of this network architecture. To illustrate this, let us consider a parallel ReLU with two sub-networks and four layers, i.e., $K = 2$ and $L = 4$. If we set $\mathbf{W}_{lk} = \mathbf{W}_l$ $8k\ 2\ [2]; l\ 2\ [4]$ then our architecture reduces to a standard four-layer network

$$\sum_{k=1}^{2} \left(\left((\mathbf{X}\mathbf{W}_{1k})_+ \mathbf{W}_{2k} \right)_+ \mathbf{W}_{3k} \right)_+ \mathbf{w}_{4k} = \left(\left((\mathbf{X}\mathbf{W}_1)_+ \mathbf{W}_2 \right)_+ \mathbf{W}_3 \right)_+ \mathbf{w}_4;$$

where $\mathbf{W}_1\ 2\ \mathrm{R}^{d\ \ m_1}; \mathbf{W}_2\ 2\ \mathrm{R}^{m_1\ \ m_2}; \mathbf{W}_3\ 2\ \mathrm{R}^{m_2\ \ m_3}; \mathbf{w}_4\ 2\ \mathrm{R}^{m_3}$. For ResNets, we first remark that since residual blocks are usually used after a ReLU activation function, which is positively homogeneous of degree one, in practice, each residual block takes only nonnegative entries as its inputs. Thus, we can assume $\mathbf{X}\ 2\ \mathrm{R}_+^{n\ \ d}$ without loss of generality. We also assume that weights obey the following form: $\mathbf{W}_{11} = \mathbf{W}_1, \mathbf{W}_{21} = \mathbf{W}_2, \mathbf{W}_{12} = \mathbf{W}_{22} = \mathbf{I}_d, \mathbf{W}_{31} = \mathbf{W}_{32} = \mathbf{W}_3$, and $\mathbf{w}_{41} = \mathbf{w}_{42} = \mathbf{w}_4$ then

$$f(\mathbf{X}) = \sum_{k=1}^{2} \left(\left((\mathbf{X}\mathbf{W}_{1k})_+ \mathbf{W}_{2k} \right)_+ \mathbf{W}_{3k} \right)_+ \mathbf{w}_{4k} = \left(\left((\mathbf{X}\mathbf{W}_1)_+ \mathbf{W}_2 \right)_+ \mathbf{W}_3 \right)_+ \mathbf{w}_4 + (\mathbf{X}\mathbf{W}_3)_+ \mathbf{w}_4$$

which is a shallow ResNet as demonstrated in Figure 1 of [21].

A.3 Proof of Lemma 1

Let us first define $r_{kj_{L-1}} := \sqrt{\sum_{j_1,\dots,j_{L-2}} \left(k\mathbf{w}_{1kj_1} k_2^2 \prod_{l=2}^{L-1} w_{lkj_{l-1}j_l}^2 \right)} > 0$. Notice that if $r_{kj_{L-1}} = 0$, this means that k^{th} sub-network does not contribute the output of the parallel network in (1). Therefore, we can remove the paths with $r_{kj_{L-1}} = 0$ without loss of generality. Now, we use the following change of variable

$$\mathbf{W}_{lk}^0 = \mathbf{W}_{lk}, \forall l \in [L-2]; \; \mathbf{w}_{(L-1)kj_{L-1}}^0 = \frac{\mathbf{w}_{(L-1)kj_{L-1}}}{r_{kj_{L-1}}}, \forall l \in [L-1]; \; w_{Lkj_{L-1}}^0 = r_{kj_{L-1}} w_{Lkj_{L-1}}.$$

We now note that

$$\sum_{j_1,\dots,j_{L-2}} \left(k\mathbf{w}_{1kj_1}^0 k_2^2 \prod_{l=2}^{L-1} w_{lkj_{l-1}j_l}^{0^2} \right) = \frac{1}{r_{kj_{L-1}}^2} \sum_{j_1,\dots,j_{L-2}} \left(k\mathbf{w}_{1kj_1} k_2^2 \prod_{l=2}^{L-1} w_{lkj_{l-1}j_l}^2 \right) = 1.$$

Then, (2) can be restated as follows

$$\begin{aligned}
p_L &= \min_{\{\{\mathbf{W}_{lk}\}_{l=1}^{L-1}\}_{k=1}^K} L\left(\sum_{k=1}^K (\mathbf{X}\mathbf{W}_{1k})_+ \cdots \mathbf{W}_{(L-1)k})_+ \mathbf{w}_{Lk}, \mathbf{y} \right) + \beta \sum_{k=1}^K \sqrt{\sum_{j_1,\dots,j_{L-1}} k\mathbf{w}_{1kj_1} k_2^2 \prod_{l=2}^{L-1} w_{lkj_{l-1}j_l}^2} \\
&= \min_{\substack{\{\{\mathbf{W}_{lk}\}_{l=1}^{L-1}\}_{k=1}^K \\ \{\mathbf{w}_{Lk}^0\}_{k=1}^K}} L\left(\sum_{k=1}^K \sum_{j_{L-1}} \frac{(\mathbf{X}\mathbf{W}_{1k})_+ \cdots \mathbf{w}_{(L-1)kj_{L-1}})_+}{r_{kj_{L-1}}} w_{Lkj_{L-1}}^0, \mathbf{y} \right) \\
&\quad + \beta \sum_{k=1}^K \sqrt{\sum_{j_{L-1}} w_{Lkj_{L-1}}^{0^2} \frac{\sum_{j_1,\dots,j_{L-2}} k\mathbf{w}_{1kj_1} k_2^2 \prod_{l=2}^{L-1} w_{lkj_{l-1}j_l}^2}{r_{kj_{L-1}}^2}} \\
&= \min_{\{\{\mathbf{W}_{lk}\}_{l=1}^{L-1}\}_{k=1}^K} L\left(\sum_{k=1}^K \sum_{j_{L-1}} (\mathbf{X}\mathbf{W}_{1k})_+ \cdots r_{kj_{L-1}}^{-1} \mathbf{w}_{(L-1)kj_{L-1}})_+ w_{Lkj_{L-1}}^0, \mathbf{y} \right) + \beta \sum_{k=1}^K k\mathbf{w}_{Lk}^0 k_2 \\
&= \min_{\{\{\mathbf{W}_{lk}^0\}_{l=1}^{L-1}\}_{k=1}^K} L\left(\sum_{k=1}^K \mathbf{X}\mathbf{W}_{1k}^0)_+ \cdots \mathbf{W}_{(L-1)k}^0)_+ \mathbf{w}_{Lk}^0, \mathbf{y} \right) + \beta \sum_{k=1}^K k\mathbf{w}_{Lk}^0 k_2,
\end{aligned}$$

where $\mathcal{S} := \left\{ (\mathbf{W}_1, \dots, \mathbf{W}_{L-1}) : \sum_{j_1,\dots,j_{L-2}} \left(k\mathbf{w}_{1j_1} k_2^2 \prod_{l=2}^{L-1} w_{lj_{l-1}j_l}^2 \right) = 1, \forall j_{L-1} \in [m_{L-1}] \right\}$.

We also note that one can relax the equality constraint as an inequality constraint without loss of generality. This is basically due to the fact that if a constraint is not tight, i.e., strictly less than one, at the optimum then we can remove that constraint and make the corresponding output layer weight arbitrarily small via a simple scaling to make the objective value smaller. However, this would lead to a contradiction since this scaling further reduces the objective, which means that the initial set of layer weights (that yields a strict inequality in the constraints) are not optimal.

□

A.4 Proof of Theorem 1

To obtain the bidual problem of (4), we first utilize semi-infinite duality theory as follows. We first compute the dual of (6) with respect to the dual parameter $\mathbf{v}$ to get

$$p_1 := \min_{k\mathbf{w}_3 k_2 \leq 1} \min_{\mu} L\left(\int_{\mathcal{S}} \left((\mathbf{X}\mathbf{W}_1)_+ \mathbf{W}_2 \right)_+ \mathbf{w}_3 d\mu(\cdot), \mathbf{y} \right) + \beta \, k\mu\|_{TV}, \tag{12}$$

where $k\mu\|_{TV}$ denotes the total variation norm of the signed measure μ. Notice that (12) is an infinite-dimensional neural network training problem similar to the one studied in [35]. More importantly, this problem is convex since the model is linear with respect to the measure μ and the loss and regularization functions are convex [35]. Thus, we have no duality gap, i.e., $d_3 = p_1$. In addition to this, even though (12) is an infinite-dimensional convex optimization problem, it reduces to a problem

with at most $n+1$ neurons at the optimum due to Caratheodory's theorem [32]. Therefore, (12) can be equivalently stated as the following finite-size convex optimization problem

$$p_1 = \min_{\mathbf{W}_1, \mathbf{W}_2, \mathbf{w}_3} L\left(\sum_{k=1}^{K} \left((\mathbf{X}\mathbf{W}_{1k})_+ \mathbf{W}_{2k}\right)_+ \mathbf{w}_{3k}; \mathbf{y}\right) + \sum_{k=1}^{K} \|\mathbf{w}_{3k}\|_2$$

$$= \min_{\mathbf{W}_1, \mathbf{W}_2, \mathbf{w}_3} L\left(f(\mathbf{X}); \mathbf{y}\right) + \sum_{k=1}^{K} \|\mathbf{w}_{3k}\|_2, \tag{13}$$

where $K \geq n+1$. We further remark that given $K \geq K^*$, (13) and (5) are the same problems, which also proves strong duality as $p_3 = p_1 = d_3$. In the remainder of the proof, we show that using an alternative representation for the ReLU activation, we can achieve a finite-dimensional convex bidual formulation.

Now we restate the dual problem (6) as

$$d_3 = \max_{\mathbf{v}} \; -L^*(\mathbf{v}) \text{ s.t. } \max_{\mathbf{W}_1, \mathbf{W}_2} \left\| \mathbf{v}^T \left((\mathbf{X}\mathbf{W}_1)_+ \mathbf{W}_2\right)_+ \right\|_2 \leq \beta. \tag{14}$$

We first note that using the representation in (7), the dual constraint in (14) can be written as

$$\begin{aligned}
&\max_{\mathbf{W}_1, \mathbf{W}_2} \left\| \mathbf{v}^T (\mathbf{X}\mathbf{W}_1)_+ \mathbf{W}_2 \right\|_+ \|_2 \leq \beta \\
&\Leftrightarrow \max_{\mathbf{W}_1, \mathbf{W}_2} \sqrt{\sum_{j_2=1}^{m_2} \left(\mathbf{v}^T (\mathbf{X}\mathbf{W}_1)_+ \mathbf{w}_{2j_2}\right)_+^2} \leq \beta \\
&\Leftrightarrow \max_{\mathbf{W}_1, \mathbf{W}_2} m_2 \left(\mathbf{v}^T (\mathbf{X}\mathbf{W}_1)_+ \mathbf{w}_2\right)_+^2 \leq \beta^2 \\
&\Leftrightarrow \max_{I_{j_1} \in \{0,1\}^n} \max_{\mathbf{W}_2} m_2 \left(\mathbf{v}^T \sum_{j_1=1}^{m_1} I_{j_1} (\mathbf{X}\mathbf{w}_{1j_1} j \mathbf{w}_{2j_1 j})_+\right)_+^2 \leq \beta^2 \\
&\Leftrightarrow \max_{\substack{i_1 \in [P_1] \\ i_2 \in [P_2]}} \max_{I_{j_1} \in \{0,1\}^n} \max_{\mathbf{w}_{j_1} g_{j_1} \in C_{il}} \sqrt{m_2}\, \mathbf{v}^T \mathbf{D}_{2l} \sum_{j_1=1}^{m_1} I_{j_1} \mathbf{D}_{1ij_1} \mathbf{X}\mathbf{w}_{j_1} \leq \beta
\end{aligned} \tag{15}$$

where $g_s = \{(\mathbf{W}_1, \mathbf{W}_2) : \sum_{j_1=1}^{m_1} \|\mathbf{w}_{1j_1}\|_2^2 \|\mathbf{w}_{2j_1 j_2}\|_2^2 \leq 1, \forall j_2 \in [m_2]\}$, we apply a variable change as $\mathbf{w}_{j_1} = j\mathbf{w}_{2j_1} j \mathbf{w}_{1j_1}$ and define the set C_{il} as

$$C_{il} := \left\{ \mathbf{w}_{j_1} g_{j_1} : (2\mathbf{D}_{2l} - \mathbf{I}_n) \sum_{j_1=1}^{m_1} I_{j_1} \mathbf{D}_{1ij_1} \mathbf{X}\mathbf{w}_{j_1} \geq 0, (2\mathbf{D}_{1ij_1} - \mathbf{I}_n)\mathbf{X}\mathbf{w}_{j_1} \geq 0, \forall j_1 \in [m_1], \sum_{j_1=1}^{m_1} \|\mathbf{w}_{j_1}\|_2^2 \leq 1 \right\}.$$

We also note that P_1 and P_2 denote the number of possible hyperplane arrangement for the first and second ReLU layer (see Appendix A.10 for details).

Then we have

$$\max_{\mathbf{W}_1, \mathbf{W}_2} \left\| \mathbf{v}^T (\mathbf{X}\mathbf{W}_1)_+ \mathbf{W}_2 \right\|_+ \|_2 \leq \beta \quad \Leftrightarrow \quad \max_{\substack{i_1 \in [P_1] \\ i_2 \in [P_2]}} \max_{I_{j_1} \in \{0,1\}^n} \max_{\mathbf{w}_{j_1} g_{j_1} \in C_{il}} \sqrt{m_2}\, \mathbf{v}^T \mathbf{D}_{2l} \sum_{j_1=1}^{m_1} I_{j_1} \mathbf{D}_{1ij_1} \mathbf{X}\mathbf{w}_{j_1} \leq \beta,$$

$$\Leftrightarrow \begin{cases} \max_{\mathbf{w}^s_{ij_1 l} g_{j_1} \in C^s_{il}} \sqrt{m_2}\, \mathbf{v}^T \mathbf{D}_{2l} \sum_{j_1=1}^{m_1} I^s_{ij_1 l} \mathbf{D}_{1ij_1} \mathbf{X}\mathbf{w}^s_{ij_1 l} \leq \beta, \\ \max_{\mathbf{w}^{s^0}_{ij_1 l} g_{j_1} \in C^s_{il}} \sqrt{m_2}\, \mathbf{v}^T \mathbf{D}_{2l} \sum_{j_1=1}^{m_1} I^{s^0}_{ij_1 l} \mathbf{D}_{1ij_1} \mathbf{X}\mathbf{w}^{s^0}_{ij_1 l} \leq \beta, \end{cases} \forall i \in [P_1], l \in [P_2], s \in [M],$$

where we use $M := \{I_{j_1}\}_{j_1=1}^{m_1} = 2^{m_1}$ to enumerate all possible sign patterns $\{I_{j_1}\}_{j_1=1}^{m_1}$ of size m_1. Using the equivalent representation above, we rewrite the dual (14) as

$$\max_{\mathbf{v}} -L^*(\mathbf{v}) \text{ s.t. } \max_{\mathbf{w}^s_{ij_1 l} g_{j_1} \in C^s_{il}} \sqrt{m_2}\, \mathbf{v}^T \mathbf{D}_{2l} \sum_{j_1=1}^{m_1} I^s_{ij_1 l} \mathbf{D}_{1ij_1} \mathbf{X}\mathbf{w}^s_{ij_1 l} \leq \beta, \quad \forall i, l, s \tag{16}$$

$$\max_{\mathbf{w}^{s^0}_{ij_1 l} g_{j_1} \in C^s_{il}} \sqrt{m_2}\, \mathbf{v}^T \mathbf{D}_{2l} \sum_{j_1=1}^{m_1} I^{s^0}_{ij_1 l} \mathbf{D}_{1ij_1} \mathbf{X}\mathbf{w}^{s^0}_{ij_1 l} \leq \beta, \quad \forall i, l, s. \tag{17}$$

Since the problem above is convex and satisfies the Slater's condition when all the parameters are set to zero, we have strong duality [33], and thus we can state (16) as

$$\min_{\substack{\xi_{il}^s \geq 0 \\ \xi_{il}^{s^0} \geq 0}} \max_{\mathbf{v}} \min_{\substack{\{\mathbf{w}_{ij_1l}^s, g_{j_1}\} \in C_{il}^s \\ \{\mathbf{w}_{ij_1l}^{s^0}, g_{j_1}\} \in C_{il}^s}} L(\mathbf{v}) + \sum_{s=1}^{M}\sum_{i=1}^{P_1}\sum_{l=1}^{P_2} \xi_{il}^s \left(\rho \sqrt{m_2}\mathbf{v}^T \mathbf{D}_{2l} \sum_{j_1=1}^{m_1} I_{ij_1l}^s \mathbf{D}_{1ij_1} \mathbf{X} \mathbf{w}_{ij_1l}^s \right) \tag{18}$$

$$+ \sum_{s=1}^{M}\sum_{i=1}^{P_1}\sum_{l=1}^{P_2} \xi_{il}^{s^0} \left(+ \rho \sqrt{m_2}\mathbf{v}^T \mathbf{D}_{2l} \sum_{j_1=1}^{m_1} I_{ij_1l}^{s^0} \mathbf{D}_{1ij_1} \mathbf{X} \mathbf{w}_{ij_1l}^{s^0} \right): \tag{19}$$

Due to Sion's minimax theorem [51], we can change the order the inner minimization and maximization to obtain closed-form solutions for the maximization over the variable $\mathbf{v}$. This yields the following problem

$$\min_{\xi_{il}^s \geq 0} \min_{\substack{\{\mathbf{w}_{ij_1l}^s, g_{j_1}\} \in C_{il}^s \\ \{\mathbf{w}_{ij_1l}^{s^0}, g_{j_1}\} \in C_{il}^s}} L\left(\sum_{s=1}^{M}\sum_{l=1}^{P_2}\sum_{i=1}^{P_1}\sum_{j_1=1}^{m_1} \rho \sqrt{m_2} \mathbf{D}_{2l}\mathbf{D}_{1ij_1}\mathbf{X}(I_{ij_1l}^s \ \xi_{il}^s \mathbf{w}_{ij_1l}^s \quad I_{ij_1l}^{s^0} \ \xi_{il}^{s^0} \mathbf{w}_{ij_1l}^{s^0}); \mathbf{y} \right)$$

$$+ \sum_{s=1}^{M}\sum_{i=1}^{P_1}\sum_{l=1}^{P_2} (\xi_{il}^s + \xi_{il}^{s^0}): \tag{20}$$

Finally, we introduce a set of variable changes changes as $\mathbf{z}_{ij_1l}^s = \rho \sqrt{m_2} \ \xi_{il}^s I_{ij_1l}^s \mathbf{w}_{ij_1l}^s$ and $\mathbf{z}_{ij_1l}^{s^0} = \rho \sqrt{m_2} \ \xi_{il}^{s^0} I_{ij_1l}^{s^0} \mathbf{w}_{ij_1l}^{s^0}$ such that (20) can be cast as the following convex problem

$$\min_{\substack{\{\mathbf{z}_{ij_1l}^s, g_{j_1}\} \in C_{il}^{s^0} \\ \{\mathbf{z}_{ij_1l}^{s^0}, g_{j_1}\} \in C_{il}^{s^0}}} L\left(\sum_{s=1}^{M}\sum_{l=1}^{P_2}\sum_{i=1}^{P_1}\sum_{j_1=1}^{m_1} \mathbf{D}_{2l}\mathbf{D}_{1ij_1}\mathbf{X}(\mathbf{z}_{ij_1l}^s \quad \mathbf{z}_{ij_1l}^{s^0}); \mathbf{y} \right)$$

$$+ \rho \sqrt{m_2} \sum_{s=1}^{M}\sum_{i=1}^{P_1}\sum_{l=1}^{P_2} \left(\sqrt{\sum_{j_1=1}^{m_1} k\mathbf{z}_{ij_1l}^s k_2^2} + \sqrt{\sum_{j_1=1}^{m_1} k\mathbf{z}_{ij_1l}^{s^0} k_2^2} \right); \tag{21}$$

where the constraint set C_{il}^s are defined as

$$C_{il}^{s^0} := \left\{ \{\mathbf{z}_{j_1}, g_{j_1}\} : (2\mathbf{D}_{2l} \quad \mathbf{I}_n) \sum_{j_1=1}^{m_1} \mathbf{D}_{1ij_1}\mathbf{X}\mathbf{z}_{j_1} \quad 0; \ (2\mathbf{D}_{1ij_1} \quad \mathbf{I}_n)\mathbf{X} I_{ij_1l}^s \mathbf{z}_{j_1} \quad 0; \forall j_1 \in [m_1] \right\}:$$

Notice that (21) is a constrained convex optimization problem with $2dm_1MP_1P_2$ variables and $2n(m_1+1)MP_1P_2$ constraints in the set C_{il}^s.

□

A.5 Proof of Proposition 1

In this section, we prove that once the convex program in (21) is globally optimized to obtain a set of optimal solutions $\{\mathbf{z}_{ij_1l}^s; \mathbf{z}_{ij_1l}^{s^0}\}_{i;j_1;l;s}$, one can recover an optimal solution to the non-convex training

problem (4) via a simple closed-form mapping as detailed below

$$\mathbf{W}_{1k} = \begin{cases} \frac{1}{m_2}\begin{bmatrix} I^s_{i1l}\mathbf{z}^s_{i1l} & I^s_{i2l}\mathbf{z}^s_{i2l} & \cdots & I^s_{im_1l}\mathbf{z}^s_{im_1l} \end{bmatrix} & \text{if } 1 \quad k \quad MP_1P_2 \\ \frac{1}{m_2}\begin{bmatrix} I^{s^0}_{i1l}\mathbf{z}^{s^0}_{i1l} & I^{s^0}_{i2l}\mathbf{z}^{s^0}_{i2l} & \cdots & I^{s^0}_{im_1l}\mathbf{z}^{s^0}_{im_1l} \end{bmatrix} & \text{if } MP_1P_2 + 1 \quad k \quad 2MP_1P_2 \end{cases}$$

$$\mathbf{W}_{2k} = \begin{cases} \begin{bmatrix} I^s_{i1l} & I^s_{i1l} & \cdots & I^s_{11l} \\ \vdots & \vdots & \cdots & \vdots \\ I^s_{im_1l} & I^s_{im_1l} & \cdots & I^s_{1m_1l} \end{bmatrix} & \text{if } 1 \quad k \quad MP_1P_2 \\ \begin{bmatrix} I^{s^0}_{i1l} & I^{s^0}_{i1l} & \cdots & I^{s^0}_{11l} \\ \vdots & \vdots & \cdots & \vdots \\ I^{s^0}_{im_1l} & I^{s^0}_{im_1l} & \cdots & I^{s^0}_{1m_1l} \end{bmatrix} & \text{if } MP_1P_2 + 1 \quad k \quad 2MP_1P_2 \end{cases}$$

$$\mathbf{w}_{3k} = \begin{cases} \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T & \text{if } 1 \quad k \quad MP_1P_2 \\ \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T & \text{if } MP_1P_2 + 1 \quad k \quad 2MP_1P_2 \end{cases};$$

where

$$(s; l; i) = \begin{cases} \left\lfloor \frac{k \quad 1}{P_1P_2} \right\rfloor + 1; \left\lfloor \frac{k \quad 1 \quad (s \quad 1)P_1P_2}{P_1} \right\rfloor + 1; k \quad (s \quad 1)P_1P_2 \quad (l \quad 1)P_1 & \text{if } 1 \quad k \quad MP_1P_2 \\ \left\lfloor \frac{k^0 \quad 1}{P_1P_2} \right\rfloor + 1; \left\lfloor \frac{k^0 \quad 1 \quad (s \quad 1)P_1P_2}{P_1} \right\rfloor + 1; k^0 \quad (s \quad 1)P_1P_2 \quad (l \quad 1)P_1 & \text{else} \end{cases}$$

with $k^0 = k \quad MP_1P_2$. Hence, we achieve an optimal solution to the original non-convex training problem (4) as $f\mathbf{W}_{1k}; \mathbf{W}_{2k}; \mathbf{w}_{3k} g_{k=1}^{2MP_1P_2}$, where $\mathbf{W}_{1k} \; 2 \; \mathsf{R}^{d \quad m_1}$, $\mathbf{W}_{2k} \; 2 \; \mathsf{R}^{m_1 \quad m_2}$, and $\mathbf{w}_{3k} \; 2 \; \mathsf{R}^{m_2}$ respectively. Next, we confirm that the proposed set of layer weights are indeed optimal by plugging them back to both the convex and non-convex objectives.

We first verify that both the optimal convex and non-convex layer weights give the same network output as follows (8), i.e.,

$$\sum_{k=1}^{2MP_1P_2} \left((\mathbf{X}\mathbf{W}_{1k})_+ \mathbf{W}_{2k} \right)_+ \mathbf{w}_{3k} = \sum_{s=1}^{M} \sum_{l=1}^{P_2} \sum_{i=1}^{P_1} \sum_{j=1}^{m_1} \mathbf{D}_{2l}\mathbf{D}_{1ij_1}\mathbf{X} \left(\mathbf{z}^s_{ij_1l} \quad \mathbf{z}^{s^0}_{ij_1l} \right):$$

Now, we show that the proposed set of weight matrices for the non-convex problem achieves the same regularization cost with (21), i.e.,

$$\sum_{k=1}^{2MP_1P_2} \sqrt{\sum_{j_2=1}^{m_2} \sum_{j_1=1}^{m_1} k\mathbf{w}_{1kj_1} k_2^2 w^2_{2kj_1j_2} w^2_{3kj_2}} = \cancel{p}\frac{1}{m_2} \sum_{s=1}^{M} \sum_{i=1}^{P_1} \sum_{l=1}^{P_2} \left(\sqrt{\sum_{j_1=1}^{m_1} k\mathbf{z}^s_{ij_1l} k_2} + \sqrt{\sum_{j=1}^{m_1} k\mathbf{z}^{s^0}_{ij_1l} k_2} \right):$$

Since $f\mathbf{W}_{1k}; \mathbf{W}_{2k}; \mathbf{w}_{3k} g_{k=1}^{2MP_1P_2}$ yields the same network output and regularization cost with the optimal parameters of the convex program in (21), we conclude that the proposed set parameters for the non-convex problem also achieves the optimal objective value p_3, i.e.,

$$p_3 = L \left(\sum_{k=1}^{2MP_1P_2} \left((\mathbf{X}\mathbf{W}_{1k})_+ \mathbf{W}_{2k} \right)_+ \mathbf{w}_{3k}; \mathbf{y} \right) + \sum_{k=1}^{2MP_1P_2} \sqrt{\sum_{j_2=1}^{m_2} \sum_{j_1=1}^{m_1} k\mathbf{w}_{kj_1} k_2^2 w^2_{2kj_1j_2} w^2_{3kj_2}}:$$

□

A.6 Proof of Theorem 2

We start with defining the optimal parameters for the original and rank-k approximation of the rescaled problem in (5) as

$$f(\mathbf{W}_{1k}; \mathbf{W}_{2k}; \mathbf{w}_{3k}) g_{k=1}^{K} := \underset{2 \quad s}{\operatorname{argmin}} \; L \left(\sum_{k=1}^{K} \left((\mathbf{X}\mathbf{W}_{1k})_+ \mathbf{W}_{2k} \right)_+ \mathbf{w}_{3k}; \mathbf{y} \right) + \sum_{k=1}^{K} k\mathbf{w}_{3k} k_2$$

$$f(\hat{\mathbf{W}}_{1k}; \hat{\mathbf{W}}_{2k}; \hat{\mathbf{w}}_{3k}) g_{k=1}^{K} := \underset{2 \quad s}{\operatorname{argmin}} \; L \left(\sum_{k=1}^{K} \left(\left(\hat{\mathbf{X}}_r \mathbf{W}_{1k} \right)_+ \mathbf{W}_{2k} \right)_+ \mathbf{w}_{3k}; \mathbf{y} \right) + \sum_{k=1}^{K} k\mathbf{w}_{3k} k_2 \tag{22}$$

and the objective value achieved by the parameters trained using $\hat{\mathbf{X}}_r$ as

$$p_r := L\left(\sum_{k=1}^{K}\left(\left(\mathbf{X}\hat{\mathbf{W}}_{1k}\right)_+ \hat{\mathbf{W}}_{2k}\right)_+ \hat{\mathbf{w}}_{3k}; \mathbf{y}\right) + \sum_{k=1}^{K} k\hat{\mathbf{w}}_{3k}k_2.$$

Then, we have

$$\begin{aligned}
p_3 &= L\left(\sum_{k=1}^{K}\left(\left(\mathbf{X}\mathbf{W}_{1k}\right)_+ \mathbf{W}_{2k}\right)_+ \mathbf{w}_{3k}; \mathbf{y}\right) + \sum_{k=1}^{K} k\mathbf{w}_{3k}k_2 \\
&\overset{(i)}{\leq} L\left(\sum_{k=1}^{K}\left(\left(\mathbf{X}\hat{\mathbf{W}}_{1k}\right)_+ \hat{\mathbf{W}}_{2k}\right)_+ \hat{\mathbf{w}}_{3k}; \mathbf{y}\right) + \sum_{k=1}^{K} k\hat{\mathbf{w}}_{3k}k_2 = p_r \\
&\overset{(ii)}{\leq} L\left(\sum_{k=1}^{K}\left(\left(\hat{\mathbf{X}}_r\hat{\mathbf{W}}_{1k}\right)_+ \hat{\mathbf{W}}_{2k}\right)_+ \hat{\mathbf{w}}_{3k}; \mathbf{y}\right) + \left(+ p_{\overline{m_1 m_2}R_{r+1}}\right)\sum_{k=1}^{K} k\hat{\mathbf{w}}_{3k}k_2 \\
&\quad \left(L\left(\sum_{k=1}^{K}\left(\left(\hat{\mathbf{X}}_r\hat{\mathbf{W}}_{1k}\right)_+ \hat{\mathbf{W}}_{2k}\right)_+ \hat{\mathbf{w}}_{3k}; \mathbf{y}\right) + \sum_{k=1}^{K} k\hat{\mathbf{w}}_{3k}k_2\right)\left(1 + \frac{p_{\overline{m_1 m_2}R_{r+1}}}{}\right) \\
&\overset{(iii)}{\leq} \left(L\left(\sum_{k=1}^{K}\left(\left(\hat{\mathbf{X}}_r\mathbf{W}_{1k}\right)_+ \mathbf{W}_{2k}\right)_+ \mathbf{w}_{3k}; \mathbf{y}\right) + \sum_{k=1}^{K} k\mathbf{w}_{3k}k_2\right)\left(1 + \frac{p_{\overline{m_1 m_2}R_{r+1}}}{}\right) \\
&\overset{(iv)}{\leq} \left(L\left(\sum_{k=1}^{K}\left(\left(\mathbf{X}\mathbf{W}_{1k}\right)_+ \mathbf{W}_{2k}\right)_+ \mathbf{w}_{3k}; \mathbf{y}\right) + \sum_{k=1}^{K} k\mathbf{w}_{3k}k_2\right)\left(1 + \frac{p_{\overline{m_1 m_2}R_{r+1}}}{}\right)^2 \\
&= p_3\left(1 + \frac{p_{\overline{m_1 m_2}R_{r+1}}}{}\right)^2;
\end{aligned}$$

where (i) and (iii) follow from the optimality definitions of the original and approximated problems in (22). In addition, (ii) and (iv) follow from the relations below

$$\begin{aligned}
&L\left(\sum_{k=1}^{K}\left(\mathbf{X}\hat{\mathbf{W}}_{1k} + \hat{\mathbf{W}}_{2k}\right)_+ \hat{\mathbf{w}}_{3k}; \mathbf{y}\right) \\
&= L\left(\sum_{k=1}^{K}\sum_{j_2=1}^{m_2}\left(\mathbf{X}\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \hat{\mathbf{w}}_{3kj_2} - \left(\hat{\mathbf{X}}_r\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \hat{\mathbf{w}}_{3kj_2} + \left(\hat{\mathbf{X}}_r\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \hat{\mathbf{w}}_{3kj_2}; \mathbf{y}\right) \\
&\overset{(1)}{\leq} L\left(\sum_{k=1}^{K}\sum_{j_2=1}^{m_2}\left(\mathbf{X}\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \hat{\mathbf{w}}_{3kj_2} - \left(\hat{\mathbf{X}}_r\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \hat{\mathbf{w}}_{3kj_2}; \mathbf{y}\right) + L\left(\sum_{k=1}^{K}\sum_{j_2=1}^{m_2}\left(\hat{\mathbf{X}}_r\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \hat{\mathbf{w}}_{3kj_2}; \mathbf{y}\right) \\
&\overset{(2)}{\leq} R\left(\sum_{k=1}^{K}\sum_{j_2=1}^{m_2}\left(\mathbf{X}\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \hat{\mathbf{w}}_{3kj_2} - \left(\hat{\mathbf{X}}_r\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \hat{\mathbf{w}}_{3kj_2}\right) + L\left(\sum_{k=1}^{K}\sum_{j_2=1}^{m_2}\left(\hat{\mathbf{X}}_r\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \hat{\mathbf{w}}_{3kj_2}; \mathbf{y}\right) \\
&= R\left(\sum_{k=1}^{K}\sum_{j_2=1}^{m_2}\left(\mathbf{X}\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \hat{\mathbf{w}}_{3kj_2} - \left(\hat{\mathbf{X}}_r\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \hat{\mathbf{w}}_{3kj_2}\right) + L\left(\sum_{k=1}^{K}\sum_{j_2=1}^{m_2}\left(\hat{\mathbf{X}}_r\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \hat{\mathbf{w}}_{3kj_2}; \mathbf{y}\right) \\
&\overset{(3)}{\leq} R\left(\sum_{k=1}^{K}\sum_{j_2=1}^{m_2}\left(\mathbf{X}\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ - \left(\hat{\mathbf{X}}_r\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \right) + L\left(\sum_{k=1}^{K}\sum_{j_2=1}^{m_2}\left(\hat{\mathbf{X}}_r\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \hat{\mathbf{w}}_{3kj_2}; \mathbf{y}\right) \\
&\leq R\max_{k\in[K];\, j_2\in[m_2]}\left(\mathbf{X}\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ - \left(\hat{\mathbf{X}}_r\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \Big\|_2 \sum_{k=1}^{K} k\hat{\mathbf{w}}_{3k}k_1 + L\left(\sum_{k=1}^{K}\sum_{j_2=1}^{m_2}\left(\hat{\mathbf{X}}_r\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \hat{\mathbf{w}}_{3kj_2}; \mathbf{y}\right) \\
&\overset{(4)}{\leq} R\max_{k\in[K]}\left\|\mathbf{X} - \hat{\mathbf{X}}_r\right\|_2 \sum_{j_1=1}^{m_1} k\hat{\mathbf{w}}_{1kj_1}k_2 j\hat{\mathbf{w}}_{2kj_1j_2}j \sum_{k=1}^{K} k\hat{\mathbf{w}}_{3k}k_1 + L\left(\sum_{k=1}^{K}\sum_{j_2=1}^{m_2}\left(\hat{\mathbf{X}}_r\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \hat{\mathbf{w}}_{3kj_2}; \mathbf{y}\right) \\
&\overset{(5)}{\leq} p_{\overline{m_1}R}\max_{k\in[K]}\left\|\mathbf{X} - \hat{\mathbf{X}}_r\right\|_2 \sqrt{\sum_{j_1=1}^{m_1} k\hat{\mathbf{w}}_{1kj_1}k_2^2 j\hat{\mathbf{w}}_{2kj_1j_2}j^2} \sum_{k=1}^{K} k\hat{\mathbf{w}}_{3k}k_1 + L\left(\sum_{k=1}^{K}\sum_{j_2=1}^{m_2}\left(\hat{\mathbf{X}}_r\hat{\mathbf{W}}_{1k} + \hat{\mathbf{w}}_{2kj_2}\right)_+ \hat{\mathbf{w}}_{3kj_2}; \mathbf{y}\right)
\end{aligned}$$

$$
\begin{aligned}
&\overset{(6)}{=} \sqrt{m_1} R\, \epsilon_{r+1} \sum_{k=1}^{K} \|\hat{\mathbf{w}}_{3k}\|_1 + \mathcal{L}\left(\sum_{k=1}^{K} \sum_{j_2=1}^{m_2} \left(\left(\hat{\mathbf{X}}_r \hat{\mathbf{W}}_{1k}\right)_+ \hat{\mathbf{w}}_{2kj_2} \right)_+ \hat{w}_{3kj_2}, \mathbf{y} \right) \\
&\le \sqrt{m_1 m_2} R\, \epsilon_{r+1} \sum_{k=1}^{K} \|\hat{\mathbf{w}}_{3k}\|_2 + \mathcal{L}\left(\sum_{k=1}^{K} \sum_{j_2=1}^{m_2} \left(\left(\hat{\mathbf{X}}_r \hat{\mathbf{W}}_{1k}\right)_+ \hat{\mathbf{w}}_{2kj_2} \right)_+ \hat{w}_{3kj_2}, \mathbf{y} \right),
\end{aligned}
$$

where we use the convexity and R-Lipschtiz property of the loss function, convexity of ℓ_2-norm, 1-Lipschitz property of the ReLU activation, $\|\mathbf{x}\|_1 \le \sqrt{m}\|\mathbf{x}\|_2$ for $\mathbf{x} \in \mathbb{R}^m$, and $\max_k \sum_{j_1=1}^{m_1} \|\hat{\mathbf{w}}_{1kj_1}\|_2^2 \hat{w}_{2kj_1j_2}^2 \le 1$ from the rescaling in Lemma 1 for $(1), (2), (3), (4), (5),$ and (6) respectively. □

A.7 Proof for the dual problem in (3)

In order to prove the dual problem, we directly utilize Fenchel duality [33]. Let us first rewrite the primal regularized training problem after the application of the rescaling in Lemma 1 as follows

$$
p_L^* = \min_{\hat{\mathbf{y}} \in \mathbb{R}^n, \theta \in \Theta_s} \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) + \beta \sum_{k=1}^{K} \|\mathbf{w}_{Lk}\|_2 \text{ s.t. } \hat{\mathbf{y}} = \sum_{k=1}^{K} \left((\mathbf{X}\mathbf{W}_{1k})_+ \ldots \mathbf{W}_{(L-1)k}\right)_+ \mathbf{w}_{Lk}. \quad (23)
$$

The corresponding Lagrangian can be computed by incorporating the constraints into objective via a dual veriable as follows

$$
L(\mathbf{v}, \hat{\mathbf{y}}, \mathbf{w}_{Lk}) = \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) - \mathbf{v}^T \hat{\mathbf{y}} + \mathbf{v}^T \sum_{k=1}^{K} \left((\mathbf{X}\mathbf{W}_{1k})_+ \ldots \mathbf{W}_{(L-1)k}\right)_+ \mathbf{w}_{Lk} + \beta \sum_{k=1}^{K} \|\mathbf{w}_{Lk}\|_2.
$$

We then define the following dual function

$$
\begin{aligned}
g(\mathbf{v}) &= \min_{\hat{\mathbf{y}}, \mathbf{w}_{Lk}} L(\mathbf{v}, \hat{\mathbf{y}}, \mathbf{w}_{Lk}) \\
&= \min_{\hat{\mathbf{y}}, \mathbf{w}_{Lk}} \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) - \mathbf{v}^T \hat{\mathbf{y}} + \mathbf{v}^T \sum_{k=1}^{K} \left((\mathbf{X}\mathbf{W}_{1k})_+ \ldots \mathbf{W}_{(L-1)k}\right)_+ \mathbf{w}_{Lk} + \beta \sum_{k=1}^{K} \|\mathbf{w}_{Lk}\|_2 \\
&= -\mathcal{L}^*(\mathbf{v}) \text{ s.t. } \left\| \mathbf{v}^T \left((\mathbf{X}\mathbf{W}_{1k})_+ \ldots \mathbf{W}_{(L-1)k}\right)_+ \right\|_2 \le \beta, \ \forall k \in [K],
\end{aligned}
$$

where $\mathcal{L}^*$ is the Fenchel conjugate function defined as [33]

$$
\mathcal{L}^*(\mathbf{v}) := \max_{\mathbf{z}} \mathbf{z}^T \mathbf{v} - \mathcal{L}(\mathbf{z}, \mathbf{y}).
$$

Hence, we write the dual problem of (23) as

$$
p_L^* = \min_{\theta \in \Theta_s} \max_{\mathbf{v}} g(\mathbf{v}) = \min_{\theta \in \Theta_s} \max_{\mathbf{v}} -\mathcal{L}^*(\mathbf{v}) \text{ s.t. } \left\| \mathbf{v}^T \left((\mathbf{X}\mathbf{W}_{1k})_+ \ldots \mathbf{W}_{(L-1)k}\right)_+ \right\|_2 \le \beta, \ \forall k \in [K].
$$

However, since the hidden layer weights are the variables of the outer minimization, we cannot directly characterize the optimal hidden layer weight in the form above. Thus, as the last step of the derivation, we change the order of the minimization over θ and the maximization over $\mathbf{v}$ to obtain the following lower bound

$$
\begin{aligned}
p_L^* \ge d_L^* &= \max_{\mathbf{v}} \min_{\theta \in \Theta_s} -\mathcal{L}^*(\mathbf{v}) \text{ s.t. } \left\| \mathbf{v}^T \left((\mathbf{X}\mathbf{W}_{1k})_+ \ldots \mathbf{W}_{(L-1)k}\right)_+ \right\|_2 \le \beta, \ \forall k \in [K] \\
&= \max_{\mathbf{v}} -\mathcal{L}^*(\mathbf{v}) \text{ s.t. } \max_{\theta \in \Theta_s} \left\| \mathbf{v}^T \left((\mathbf{X}\mathbf{W}_{1})_+ \ldots \mathbf{W}_{(L-1)}\right)_+ \right\|_2 \le \beta.
\end{aligned}
$$

□

A.8 Hyperplane arrangements

Here, we review the notion of hyperplane arrangements detailed in [29].

We first define the set of all hyperplane arrangements for the data matrix $\mathbf{X}$ as

$$
\mathcal{H} := \bigcup \left\{ \{\operatorname{sign}(\mathbf{X}\mathbf{w})\} : \mathbf{w} \in \mathbb{R}^d \right\},
$$

where $|H| \leq 2^n$. The set H all possible $\{+1, -1\}$ labelings of the data samples $\{\mathbf{x}_i\}_{i=1}^n$ via a linear classifier $\mathbf{w} \in \mathbb{R}^d$. We now define a new set to denote the indices with positive signs for each element in the set H as $S := \left\{ \{ h_i = 1 \} : \mathbf{h} \in H \right\}$. With this definition, we note that given an element $S \in \mathcal{S}$, one can introduce a diagonal matrix $\mathbf{D}(S) \in \mathbb{R}^{n \times n}$ defined as

$$\mathbf{D}(S)_{ii} := \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases}.$$

$\mathbf{D}(S)$ can be also viewed as a diagonal matrix of indicators, where each diagonal entry is one if the the corresponding sample labeled as $+1$ by the linear classifier $\mathbf{w}$, zero otherwise. Therefore, the output of ReLU activation can be equivalently written as $(\mathbf{X}\mathbf{w})_+ = \mathbf{D}(S)\mathbf{X}\mathbf{w}$ provided that $\mathbf{D}(S)\mathbf{X}\mathbf{w} \geq 0$ and $(\mathbf{I}_n - \mathbf{D}(S))\mathbf{X}\mathbf{w} \leq 0$ are satisfied. One can define more compactly these two constraints as $(2\mathbf{D}(S) - \mathbf{I}_n)\mathbf{X}\mathbf{w} \geq 0$. We now denote the cardinality of $\mathcal{S}$ as P, and obtain the following upperbound

$$P \leq 2 \sum_{k=0}^{r-1} \binom{n-1}{k} \leq 2r \left(\frac{e(n-1)}{r} \right)^r$$

where $r := \text{rank}(\mathbf{X}) \leq \min(n, d)$ [52–55].

A.9 Low rank model in Theorem 2

In Section 3.1, we propose an ϵ-approximate training approach that has polynomial-time complexity even when the data matrix is full rank. Here, you can select the rank r by plugging in the desired approximation error and network structure in equation 10. We show that the approximation error proved in Theorem 2 can be arbitrarily small for practically relevant problems. As an example, consider a parallel architecture training problem with ℓ_2 loss function, then the upperbound becomes $(1 + \frac{\sqrt{m_1 m_2}\,\sigma_{r+1}}{\epsilon})^2$, which can be arbitrarily close to one due to presence of noise component (with small σ_{r+1}) in most datasets in practice (see Figure 4 for an empirical verification). This observation is also valid for several benchmark datasets, including MNIST, CIFAR-10, and CIFAR-100, which exhibit exponentially decaying singular values (see Figure 7) and therefore effectively has a low rank structure. In addition, singular values can be computed to set the target rank and the value of the regularization coefficient to obtain any desired approximation ratio using Theorem 2.

A.10 Proof of Proposition 2 and Corollary 1

We first review the multi-layer hyperplane arrangements concept introduced in Section 2.1 of [17]. Based on this concept, we then calculate the training complexity to globally solve the convex program in Theorem 1.

If we denote the number of hyperplane arrangements for the first ReLU layer as P_1, then from Appendix A.8 we know that

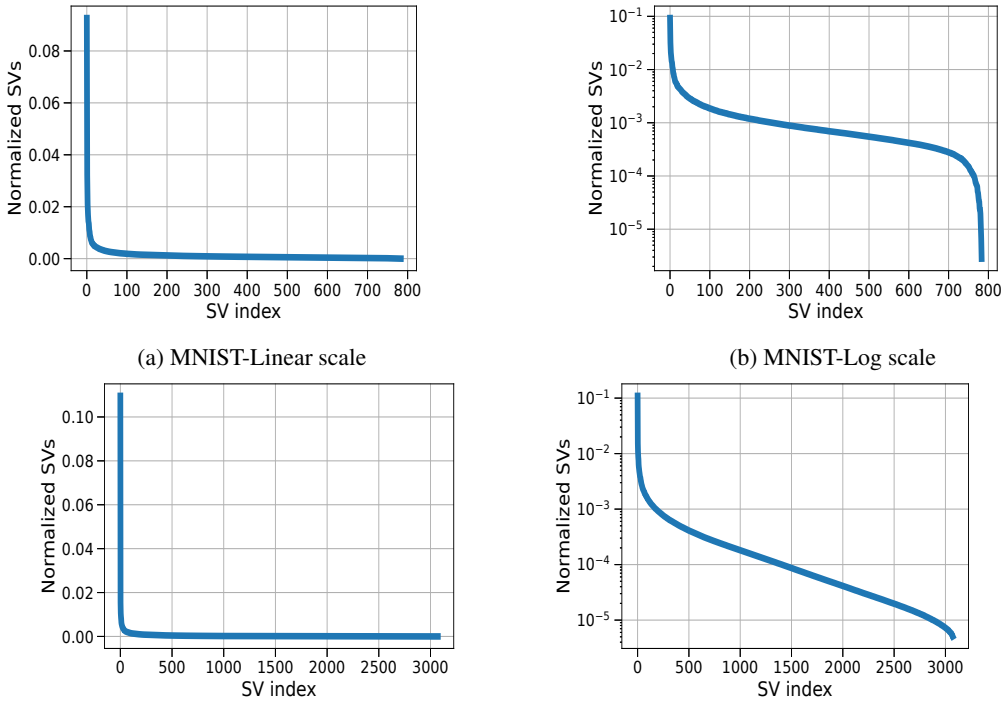
$$P_1 \leq 2r \left(\frac{e(n-1)}{r} \right)^r \leq O(n^r). \tag{24}$$

In order to obtain a bound for the number of hyperplane arrangements in the second ReLU layer we first note that preactivations of the second ReLU layer, i.e., $(\mathbf{X}\mathbf{W}_1)_+ \mathbf{w}_2$ can be equivalently represented as a matrix-vector product form by using the effective data matrix $\mathbf{X} := [\mathbf{I}_1 \mathbf{D}_{11}\mathbf{X} \;\; \mathbf{I}_2 \mathbf{D}_{12}\mathbf{X} \;\; \cdots \;\; \mathbf{I}_{m_1}\mathbf{D}_{1m_1}\mathbf{X}]$ due to the equivalent representation in (7). Therefore, given $\text{rank}(\mathbf{X}) = r_2 \leq m_1 r$

$$P_2 \leq 2 \sum_{k=0}^{r_2-1} \binom{n-1}{k} \leq 2r_2 \left(\frac{e(n-1)}{r_2} \right)^{r_2} \leq 2m_1 r \left(\frac{e(n-1)}{m_1 r} \right)^{m_1 r}.$$

However, notice that $\mathbf{X}$ is not a fixed data matrix since we can choose each diagonal $\mathbf{D}_{1j_1}$ among a set $\{\mathbf{D}_i\}_{i=1}^{P_1}$ of size P_1 due to (24) and the sign pattern $\mathbf{I}_{j_1}$ among the set $\{+1, -1\}$ of size 2. Thus, in the worst-case, we have the following upper-bound

$$P_2 \leq P_2 (2P_1)^{m_1} \leq \frac{2^{2m_1+1}(e(n-1))^{2m_1 r}}{m_1^{m_1 r - 1} r^{2m_1 r - m_1 - 1}} \leq O(n^{m_1 r}). \tag{25}$$



(a) MNIST-Linear scale

(b) MNIST-Log scale

(c) CIFAR-10-Linear scale

(d) CIFAR-10-Log scale

Figure 7: The values of normalized singular values of the data matrix for the MNIST and CIFAR-10 datasets. As illustrated in both figures, the singular values follow an exponentially decaying trends indicating an effective low rank structure.

Notice that given fixed scalars m_1 and r, both P_1 and P_2 are polynomial terms with respect to the number of data samples n and the feature dimension d.

Remark 3. *Notice that Convolutional Neural Networks (CNNs) operate on the patch matrices $\{\mathbf{X}_b\}_{b=1}^{B}$ instead of the full data matrix $\mathbf{X}$, where $\mathbf{X}_b \in \mathbb{R}^{n \times h}$ and h denotes the filter size. Hence, even when the data matrix is full rank, i.e., $r = \min\{n, d\}$, the number of hyperplane arrangements P_1 is upperbounded as $P_1 \leq O(n^{r_c})$, where $r_c := \max_b \mathrm{rank}(\mathbf{X}_b) \leq h \ll \min\{n, d\}$ (see [41] for details). For instance, let us consider a CNN with $m_1 = 512$ filters of size 3×3, then $r_c \leq 9$ independent of data dimension n, d. As a consequence, weight sharing structure in CNNs dramatically limits the number of possible hyperplane arrangements. This also explains efficiency and remarkable generalization performance of CNNs in practice.*

Training complexity analysis: Here, we calculate the computational complexity to globally solve the convex program in (8). Note that (8) is a convex optimization problem with $2dm_1MP_1P_2$ variables and $2n(m_1+1)MP_1P_2$ constraints. Therefore, due to the upperbounds in (24) and (25) of Appendix A.8, the convex program (8) can be globally optimized by a standard interior-point solver with the computational complexity $O(d^3 m_1^3 2^{3(m_1+1)} n^{3(m_1+1)r})$, which is a polynomial-time complexity in terms of n, d.

The analysis in this section can be recursively extended to arbitrarily deep parallel networks. First notice that if we apply the same approach to obtain an upperbound on P_3, then due to the multiplicative pattern in (25), we obtain $P_3 \leq P_3^{\prime}(2P_2)^{m_2} \leq O(n^{m_2 m_1 r})$. In a similar manner, the number of hyperplane arrangements in the l^{th} layer is upperbounded as $P_l \leq O(n^{r \prod_{j=1}^{l-1} m_j})$, which is polynomial in both n and d for fixed data rank r and fixed layer widths $\{m_j\}_{j=1}^{l-1}$.

□

A.11 Extension to vector outputs

In this section, we extend the analysis to parallel networks with multiple outputs where the label matrix is defined as $\mathbf{Y} \; 2 \; \mathsf{R}^{n \;\; C}$ provided that there exist C classes/outputs. Then the primal non-convex training problem is as follows

$$p_v := \min_{2} L \left(\sum_{k=1}^{K} f_{;k}(\mathbf{X}); \mathbf{Y} \right) + \sum_{k=1}^{K} \sqrt{\sum_{j_1; j_2; \cdots; j_L} \left(k\mathbf{w}_{1kj_1} k_2^2 \prod_{l=2}^{L} \; {}^{1} w^2_{lkj_l \;\; 1j_l} k\mathbf{w}^2_{Lkj_L \;\; 1j_L} k^2 \right)} :$$

Applying Lemma 1 and each step in the proof of Theorem 1 yield

$$d_v := \max_{\mathbf{V}} \min_{2 \;\; s} \;\; L \;\; (\mathbf{V}) \text{ s.t. } \left\| \mathbf{V}^T \left((\mathbf{X}\mathbf{W}_{1k})_+ ::: \mathbf{W}_{(L \;\; 1)k} \right)_+ \right\|_F \qquad ; 8k \; 2 \; [K];$$

where the corresponding Fenchel conjugate function is

$$L \;\; (\mathbf{V}) := \max_{\mathbf{Z}} \text{trace} \left(\mathbf{Z}^T \mathbf{V} \right) \quad L \; (\mathbf{Z}; \mathbf{Y}) :$$

Notice that above we have a dual matrix $\mathbf{V}$ instead of the dual vector in the scalar output case. More importantly, here, we have $`_2$ norm in the dual constraint unlike the scalar output case with absolute value. Therefore, the vector-output case is slightly more challenging than the scalar output case and yields a different regularization function in the equivalent convex program.

The rest of the derivations directly follows the steps in Section A.4 and [56].