Unveiling Hidden Convexity in Deep Learning: A Sparse Signal Processing Perspective

Emi Zeger and Mert Pilanci
Department of Electrical Engineering, Stanford University

INTRODUCTION

Deep neural networks, particularly those with the Rectified Linear Unit (ReLU) activation functions, have achieved remarkable success across diverse machine learning tasks, including image recognition, audio processing and language modeling [1]. Despite this success, the non-convex nature of their loss functions complicates optimization and limits our theoretical understanding. These challenges are especially pertinent for signal processing applications where stability, robustness, and interpretability are crucial.

Various theoretical approaches have been developed for analyzing neural networks. In this paper, we highlight addressing these issues through the recently developed convex equivalence of ReLU neural networks and their connection to sparse signal processing models. Recent research has uncovered hidden convexity in the loss landscapes of certain neural network architectures, notably two-layer ReLU networks followed by deeper networks and variations of network architectures [2–24]. By reframing the training process as a convex optimization task, it becomes possible to efficiently find globally optimal solutions, offering new perspectives on the network's generalization and robustness characteristics while facilitating interpretability. Leveraging Lasso, group Lasso and structure-inducing regularization frameworks, which are fundamental tools in sparse signal processing and compressed sensing, neural network training can be approached as a convex optimization problem, enabling the interpretation of both globally and locally optimal solutions. Moreover, these concepts expand to accommodate other activations, advanced network architectures and higher-dimensional data by incorporating geometric algebra, which provides a unified geometric framework for interpreting inner workings of neural networks.

This paper is intended to provide an accessible and educational overview that bridges recent advances in the mathematics of deep learning with traditional signal processing, inviting the signal processing community to consider these insights for broader applications.

The paper is organized as follows. We first give a brief background on neural networks and approaches to analyze them using convex optimization. We then give an equivalence theorem between a 2-layer ReLU network and a convex group Lasso problem. We describe how deeper networks and alternative architectures can also be formulated as convex problems, and give some experimental results that demonstrate the performance benefit of training the network as a convex model. Finally, we discuss remaining challenges and research directions for convex analysis of neural networks.

Notation: We denote the vector of ones by $\mathbf{1}$. The boolean function $1\{x\}$ returns 1 if x is true, and 0 otherwise. All functions and operations including $1\{x\}$ and $\geq$ extend to vector inputs elementwise. Denote $[n] = \{1, \cdots, n\}$. There are n training samples $\mathbf{x}_i$, each of dimension d, which are stacked into the data matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$ of rank r. The $d \times d$ identity matrix is $\mathbf{I}_d$. We use m to denote the number of hidden neurons.

BACKGROUND: NEURAL NETWORKS

Neural networks are a class of parameterized functions used to fit data to labels or targets, a task known as supervised learning. Neural networks are characterized by composing functions called neurons. A *neuron* is a parameterized function $f_{\text{neuron}} : \mathbb{R}^d \to \mathbb{R}$,

$$f_{\text{neuron}}(\mathbf{x}) = \sigma(\mathbf{x}^T\mathbf{w} + b) \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are *weight* and *bias* parameters, respectively, and $\sigma : \mathbb{R} \to \mathbb{R}$ is some nonlinear *activation* function. A common activation function is the Rectified Linear Unit (ReLU): $\sigma(x) = \max\{x, 0\}$. An activation function is *active* at x if $\sigma(x) \neq 0$. For example, non-negative inputs activate a ReLU. The nonlinearity of an activation function distinguishes a neuron from a traditional linear model. The neuron (1) is inspired by a biological neuron in the brain, which receives synaptic inputs (which can be viewed as $\mathbf{x}$) whose intensity is modulated by the number of receptors ($\mathbf{w}$) and then fires an action potential as output. In the brain, neurons can operate in a variety of series, feedback, and in parallel pathways [25]. Motivated by this biology, a *hidden layer* is a stack of m parallel neurons: $f_{\text{layer}} : \mathbb{R}^d \to \mathbb{R}^m$,

$$f_{\text{layer}}(\mathbf{x}) = \sigma(\mathbf{x}^T\mathbf{W} + \mathbf{b}), \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{d \times m}$, $\mathbf{b} \in \mathbb{R}^{1 \times m}$ and σ extends to vector inputs element-wise. An L-layer network is generally constructed from $L - 1$ nonlinear hidden layers (2) composed with each other, followed by an outer linear layer that combines all of the neuron paths. A L-layer network has L layers and depth L, and the width of a layer is the number of neurons in that layer. A standard 2-layer neural network is $f : \mathbb{R}^d \to \mathbb{R}$,

$$f(\mathbf{x}) = \sigma(\mathbf{x}^T\mathbf{W} + \mathbf{b})\boldsymbol{\alpha} + \xi \tag{3}$$

where (2) is the first hidden layer of (3), and $\boldsymbol{\alpha} \in \mathbb{R}^p, \xi \in \mathbb{R}$ are the weight and external bias of the outer linear layer. The bias parameters are often omitted for simplicity or made implicit. A network is said to be shallow if it has only 2 layers, and deep if it has more. The neural network (3) is a 2-layer, *fully connected*, *feed-forward* network. There are many types of neural networks consisting of variations on (3), with a variety of architectures for deeper networks (with more layers).

Neural networks are models used to fit data to labels/targets based on known pairs $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)$ that are given, where y_i is the *target* of *training data* $\mathbf{x}_i$. Given training data, a neural network is *trained* by finding parameters (weights and biases) so that $f(\mathbf{x}_i) \approx y_i$, which is formulated as a *training problem*

$$\min_{\theta} \sum_{i=1}^{n} \ell\left(f\left(\mathbf{x}_i\right), y_i\right) + R(\theta) \tag{4}$$

where $\ell(\cdot)$ is a loss function quantifying the error between $f(\mathbf{x}_i)$ and y_i, θ is the set of parameters, and R is a regularization function that penalizes large parameter magnitudes to favor simpler solutions.

BACKGROUND: PRIOR WORK ON CONVEX NEURAL NETWORKS

While impressive advancements in engineering have been made in designing neural networks to perform advanced tasks, rigorously and intuitively understanding neural networks from a theoretical perspective remains a challenging open problem, which is the focus of this paper. A major obstacle to analyzing neural networks is that they optimize a training problem (4) that is non-convex due to the product of inner and outer weights, for example $\mathbf{W}$ and $\boldsymbol{\alpha}$ in a 2-layer network (2). Traditional approaches to training a network involve performing gradient descent on the training problem, but this can converge to a suboptimal local minimum due to the non-convexity of the training problem, in contrast to training a convex function where all stationary points are globally optimal. Since convex functions are much better understood, one major approach to study neural networks has been to reformulate them via convex optimization. To facilitate analysis, simplifying assumptions are often made, such as focusing on networks with shallow depth or infinite width.

It was shown in [26] that neural network training can be formulated as an infinite dimensional convex optimization problem. They provide an incremental non-convex algorithm to train a network, in particular a 2-layer network with sign or tanh activation and l_1 regularization, to global optimality. This is achieved by successively adding neurons, under the assumption that the corresponding non-convex subproblem is solved to global optimality at each step. The generalization properties of 2-layer networks with homogeneous activations (such as ReLU) and with infinitely many neurons are studied in [27]. Importantly, [27] provides bounds on the accuracy of using infinite width and shows that convex relaxations of the training problem can achieve the same bounds under certain assumptions.

In [28], a convex formulation is presented for deep neural networks with infinite width and infinite-dimensional features, trained with regularization. A finite, discrete neural network is treated as a random sampling of neurons [28]. In [29], a "Neural Balance Theorem" is demonstrated, which states that the magnitude of the input and output weights of any neuron with homogeneous activation must be equal. This weight scaling is used an important first step in the convexification approach of [3].

Building on prior works, a recent line of work has taken a new approach to convexifying networks that relates them to Lasso problems. This Lasso approach uncovers certain variable selection properties of weight-decay regularization [27], and tackles more realistic versions of neural networks with finite width. Specifically, in contrast to [26–28], the Lasso strategy analyzes neural networks with a finite number of neurons, finite number of training data, a variety of architectures, and an explicit convex reformulation of the training problem, giving both practical approaches for training and theoretical insights into neural network representation power through ideas from signal processing, discussed next.

BACKGROUND ON SPARSE SIGNAL PROCESSING, COMPRESSED SENSING, AND GEOMETRIC ALGEBRA

Sparse signal processing is a fundamental area in signal processing that deals with signals which can be represented using a small number of non-zero coefficients in some basis or dictionary. This sparsity leads to efficient storage, transmission, and processing of signals. A key application of sparse signal processing is *compressed sensing* [30, 31], which asserts that sparse signals can be recovered from far fewer measurements than traditionally required by Nyquist sampling theory.

Lasso

The Least Absolute Shrinkage and Selection Operator (Lasso) [32] is a convex model that has become a cornerstone in sparse signal processing. Lasso performs variable selection and regularization simultaneously, making it an effective tool for recovering sparse signals in compressed sensing applications. The Lasso optimization problem is defined as:

$$\min_{\mathbf{z},\xi} \frac{1}{2}\|\mathbf{A}\mathbf{z} + \xi\mathbf{1} - \mathbf{y}\|_2^2 + \beta\|\mathbf{z}\|_1 \tag{5}$$

where $\mathbf{A}$ is the measurement matrix with columns $\mathbf{A}_i \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^n$ is the observation vector, $\mathbf{z} \in \mathbb{R}^d$ is the sparse signal to be recovered, ξ is a bias signal to be recovered, and $\beta > 0$ is a regularization parameter controlling the sparsity level. The ℓ_1-norm $\|\mathbf{z}\|_1 = \sum_{i=1}^{d} |z_i|$ promotes sparsity by penalizing the absolute sum of the coefficients. $\mathbf{A}$ is also called a *dictionary matrix*, and its columns $\mathbf{A}_i \in \mathbb{R}^n$ are *feature vectors*. In compressed sensing, Lasso serves as a convex relaxation of the NP-hard ℓ_0-norm minimization problem, providing an efficient computational method for sparse recovery with theoretical guarantees under certain conditions on the measurement matrix $\mathbf{A}$.

Group Lasso

Group Lasso extends the concept of Lasso to promote sparsity at the group level [33]. In many signal processing applications, signals exhibit group sparsity, where non-zero coefficients occur in clusters or groups. Group Lasso accounts for this structure by grouping variables and applying an ℓ_1/ℓ_2-norm regularization:

$$\min_{\mathbf{z},\xi} \frac{1}{2}\|\mathbf{A}\mathbf{z} + \xi\mathbf{1} - \mathbf{y}\|_2^2 + \beta \sum_{g=1}^{G} \|\mathbf{z}^{(g)}\|_2 \tag{6}$$

where $\mathbf{z}^{(g)}$ denotes the coefficients in group g, and $\|\mathbf{z}^{(g)}\|_2$ is the ℓ_2-norm of the group coefficients. This encourages entire groups of coefficients to be zero, promoting structured sparsity that aligns with the underlying signal characteristics. Figure 1b illustrates an example of group variable selection in group Lasso. When the groups consist of individual entries of β, we have $\sum_{g=1}^{G} \|\mathbf{z}^{(g)}\|_2 = \sum_j |z_j|$, which reduces to the ℓ_1-norm.

Compressed Sensing (CS)

CS is a signal processing technique that compresses a signal $\mathbf{z} \in \mathbb{R}^d$ by projecting it onto $\tilde{d} < d$ measurement vectors $\mathbf{a}_i \in \mathbb{R}^d$ via

$$y_i = \mathbf{z}^T \mathbf{a}_i \tag{7}$$

for $i \in [\tilde{d}]$. The measurement vectors can be randomly sampled, for example $\mathbf{a}_i \sim \mathcal{N}(0, \mathbf{I}_d)$ [31, 34]. Given $\mathbf{y}_i$ and $\mathbf{a}_i$, the compressed sensing technique recovers the data $\mathbf{x}$ by finding the sparsest solution that is consistent with the observations (7). In 1-bit compressive sensing [35], the measurements are further quantized via $y_i = Q(\mathbf{z}^T \mathbf{a}_i)$, where $Q(x) = \text{sign}(x) \in \{-1, 1\}$ so that each measurement is compressed to only one bit of information. Then the signal $\mathbf{x}$ is recovered in general CS by solving (5) or (6) (depending on the sparsity structure of the signals), and in 1-bit CS by solving

$$\begin{aligned} &\min_{\mathbf{z}} \ \|\mathbf{z}\|_1 \\ &\text{s.t. } \text{Diag}(\mathbf{y})(\mathbf{A}\mathbf{z}) \geq 0, \ \|\mathbf{z}\|_2 = 1 \end{aligned}$$

where $\mathbf{y}$ is a vector of measurements y_i and $\mathbf{A}$ is a matrix whose rows are $\mathbf{a}_i$, which are given. The $\|\mathbf{z}\|_2 = 1$ constraint is used to resolve ambiguity in the solution.

Nuclear Norm Regularization

The *nuclear norm* of a matrix $\mathbf{W}$ is the ℓ_1-norm of its singular values: $\|\mathbf{W}\|_* = \sum_{k=1}^m \sigma_k(\mathbf{W}) = \|\sigma(\mathbf{W})\|_1$. As a special case, the nuclear norm of a positive semidefinite matrix is its trace. The nuclear norm is often used in optimization problems to search for low-rank matrices. We discuss two examples, robust PCA and matrix completion.

1) Robust PCA: The robust Principal Component Analysis (PCA) problem [36] for a matrix $\mathbf{X}$ is

$$\min_{\mathbf{W},\mathbf{S}:\mathbf{X}=\mathbf{S}+\mathbf{W}} \|\mathbf{W}\|_* + \|\mathbf{S}\|_1. \tag{8}$$

The robust PCA problem is a convex heuristic to decompose $\mathbf{X}$ into the sum of a low rank matrix $\mathbf{W}$ and a sparse matrix $\mathbf{S}$ by penalizing the singular values of $\mathbf{W}$ and all elements of $\mathbf{S}$. The low-rank matrix $\mathbf{W}$ can represent the underlying low-dimensional subspace and the sparse matrix $\mathbf{S}$ represents outliers.

2) Matrix completion: The matrix completion problem [37] is as follows. Given a $n \times n$ matrix where only some of the values are known, fill in the rest of the matrix so that is has the lowest rank possible, consistent with the given elements. This problem can approximately solved by filling in the unknown values that give the lowest nuclear norm [37].

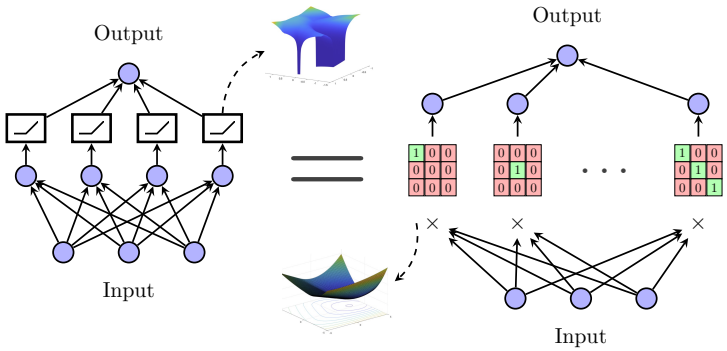
Geometric Algebra

Geometric algebra is a mathematical framework that generalizes complex numbers to higher-dimensional vector spaces, including and quaternions and hypercomplex numbers. It provides an elegant way to represent and perform operations on geometric objects [38]. There has been recent work on approaching signal processing through geometric algebra to develop new algorithms for image, audio, and video

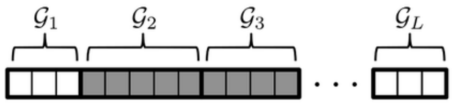
processing [39]. Representing signals with geometric algebra can improve signal processing and deep learning by leveraging geometric representations, enabling more compact and expressive models [39].

CONVEXIFYING NEURAL NETWORKS

A recent line of work [2–24] has introduced and developed a framework of exact, Lasso-like convex formulations of non-convex neural network optimization problems. These results show that neural networks with Rectified Linear Unit (ReLU), threshold or polynomial activation functions can be trained to global optimality using convex optimization. Figure 1a from [40] illustrates this point on the equivalence of neural networks and convex models. Furthermore, algorithmic tricks such as Batch Normalization which are essential for the success of local search heuristics can be demystified and enhanced [12]. These convex reformulations extend to adversarial networks [22], polynomial activation networks [23], networks with quantized weights [13], deep networks, transformers and diffusion models [9, 10]. A key observation in these results is the *hidden convexity* arising from the *sum of non-convex functions* which satisfy certain assumptions such as piecewise linear/polynomial structure. The mathematical proof techniques used in the proofs of these results are convex analytic in nature, and include convex geometry, polar duality and analysis of extreme points. There is an open-source convex optimization library *Scalable Convex Neural Networks* which outperforms other baselines in training shallow ReLU neural networks [41]. Furthermore, the proposed methods offer rigorous optimality guarantees, assured stopping conditions, and numerical stability and reliability which are essential in mission critical problems. The convexification ensures that training neural networks is agnostic to hyperparameters such as initialization, mini-batching, and step sizes, which typically exert a significant influence on the performance of local optimization methods. While an equivalent training problem, even if convex, must have the same worst case computational complexity than the original problem, the convex versions offer more intuition and insight by trading function complexity for data complexity and uncover special but useful cases in which solutions are known to be easily found, including solutions in closed form [4, 16, 42]. The rest of this paper explores the key elements of this convex equivalence, starting with 2-layer networks.



(a) Non-convex ReLU network representation [40].



(b) Group Lasso penalizes entire groups of variables (shown as $g_1, g_2, g_3, \cdots, g_L$) and selects only a few groups (depicted by the white g_1, g_L boxes in this example) to contain any non-zero variables.

Fig. 1: Equivalence (Figure 1a) of a 2-layer ReLU network to a Lasso model with a group norm penalty (Figure 1b).

2-LAYER NETWORKS

ReLU activation scalar output networks

A two-layer neural network (2) with ReLU activation is

$$f(\mathbf{x}) = \left(\mathbf{x}^T\mathbf{W}\right)_+ \boldsymbol{\alpha}. \tag{9}$$

The external bias ξ in (2) is omitted for simplicity, and the internal bias $\mathbf{b}$ can be implicitly added by appending a 1 to $\mathbf{x}$ and an extra row to $\mathbf{W}$. Let $\mathbf{X}$ be a $d \times n$ data matrix consisting of training samples $\mathbf{x}_1, \cdots, \mathbf{x}_n$ as columns. The training problem (4) for this network (9) using l_2 loss and l_2 regularization is

$$\min_{\mathbf{W},\boldsymbol{\alpha}} \frac{1}{2} \left\|f(\mathbf{X}) - \mathbf{y}\right\|^2 + \beta \left(\|\mathbf{W}\|_F^2 + \|\boldsymbol{\alpha}\|_2^2\right), \tag{10}$$

$\beta > 0$ is a fixed regularization parameter. The term $\|\mathbf{W}\|_F^2 + \|\boldsymbol{\alpha}\|_2^2$ represents *weight decay* regularization, which helps prevent overfitting by controlling the complexity of the function class. While the training problem (10) uses l_2 loss, most results discussed extend to general convex loss functions.

The training problem is *equivalent* to another optimization problem P if they share the same optimal objective values and if an optimal network for the training problem can be reconstructed from an optimal solution of P. It was shown in [3] that the non-convex training problem (10) for a 2-layer ReLU neural network and the convex group Lasso problem (6) with certain linear constraints are equivalent, for a Lasso dictionary matrix $\mathbf{A} = [\mathbf{D}_1\mathbf{X}^T, ..., \mathbf{D}_G\mathbf{X}^T]$, where $\mathbf{D}_1, ..., \mathbf{D}_G$ are fixed hyperplane encoding matrices that represent separation patterns of the dataset, formally defined next.

Hyperplane Arrangement Patterns. The set of *hyperplane arrangement patterns* of a data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ is

$$\mathcal{H} = \left\{1\{\mathbf{X}^T\mathbf{w} \geq 0\} : \mathbf{w} \in \mathbf{R}^d\right\}. \tag{11}$$

Training data on the positive side of the hyperplane $\mathbf{X^T w} = 0$ satisfy $\mathbf{x}^T\mathbf{w} \geq 0$ and thus activate the neuron $(\mathbf{x}^T\mathbf{w})_+$, while those on the negative side do not. The number of patterns is $|\mathcal{H}| \leq 2r\left(\frac{e(n-1)}{r}\right)^r$, where $r = \text{rank}(\mathbf{X}) \leq \min\{n, d\}$ [3]. Let $G = |\mathcal{H}|$ and enumerate the patterns as $\mathcal{H} = \{\mathbf{h}_1, \cdots, \mathbf{h}_G\}$. The *activation chamber* for a pattern $\mathbf{h}_g \in \mathcal{H}$ is $\mathcal{K}^{(g)} = \{\mathbf{w} \in \mathbb{R}^d : 1\{\mathbf{X}\mathbf{w} \geq \mathbf{0}\} = \mathbf{h}_g\}$, which is the cone of all weights that induce an activation pattern $\mathbf{h}_g$. The *hyperplane encoding matrix* is $\mathbf{D}_g = \text{Diag}(\mathbf{h}_g)$ for $g = 1, \cdots, G$. For fixed $\mathbf{X}$, a neuron's output $(\mathbf{X}^T\mathbf{w})_+$ as a function of $\mathbf{w}$ is linear over $\mathbf{w} \in \mathcal{K}^{(g)}$, as $\left(\mathbf{X}^T\mathbf{w}\right)_+ = \mathbf{D}_g\mathbf{X}^T\mathbf{w}$. The activation chambers partition $\mathbb{R}^d = \bigcup_{g=1}^G \mathcal{K}^{(g)}$, and so the matrices $\mathbf{D}_1, \cdots, \mathbf{D}_G$ completely characterize the piecewise linearity of the neuron output $(\mathbf{X}^T\mathbf{w})_+$.

Theorem 1 ([3])**.** *The non-convex training problem* (10) *for a* 2*-layer ReLU network is equivalent to the convex group Lasso problem*

$$\min_{\mathbf{u}^{(g)},\mathbf{v}^{(g)}\in\mathcal{K}^{(g)}} \frac{1}{2}\|\sum_{g=1}^G \mathbf{D}_g\mathbf{X}^T\left(\mathbf{u}^{(g)} - \mathbf{v}^{(g)}\right) - \mathbf{y}\|_2^2 + \beta\sum_{g=1}^G \left(\|\mathbf{u}^{(g)}\|_2 + \|\mathbf{v}^{(g)}\|_2\right) \tag{12}$$

where $\mathcal{K}^{(g)} = \{\mathbf{z}^{(g)} : (2\mathbf{D}_g - \mathbf{I})\mathbf{X}^T\mathbf{z}^{(g)} \geq 0\}$*, provided* $m \geq m^*$ *where* m^* *is the number of nonzero* $\mathbf{u}^{(g)}, \mathbf{v}^{(g)}$.

Theorem 1 facilitates the global optimization of ReLU neural networks through convex optimization and allows for the interpretation of the network as a sparse Lasso model. The number m^* is a critical threshold on the number of neurons (number of columns of $\mathbf{W}$) necessary to allow the network to be sufficiently expressive to model the data. The variable $\mathbf{z} \in \mathbb{R}^{Gd}$ is partitioned into G consecutive subvectors $\mathbf{z}^{(g)} = \mathbf{u}^{(g)} - \mathbf{v}^{(g)} \in \mathbb{R}^d$, which represent the g^{th} neuron, in the group Lasso (6). An optimal neural network is reconstructed as

$$\begin{aligned} \mathbf{w}^{(g)} &= \frac{\mathbf{u}^{(g)}}{\sqrt{\|\mathbf{u}^{(g)}\|_2}}, \alpha_g = \sqrt{\|\mathbf{u}^{(g)}\|_2} \\ \text{or } \mathbf{w}^{(g)} &= \frac{\mathbf{v}^{(g)}}{\sqrt{\|\mathbf{v}^{(g)}\|_2}}, \alpha_g = -\sqrt{|\mathbf{v}^{(g)}|} \end{aligned} \tag{13}$$

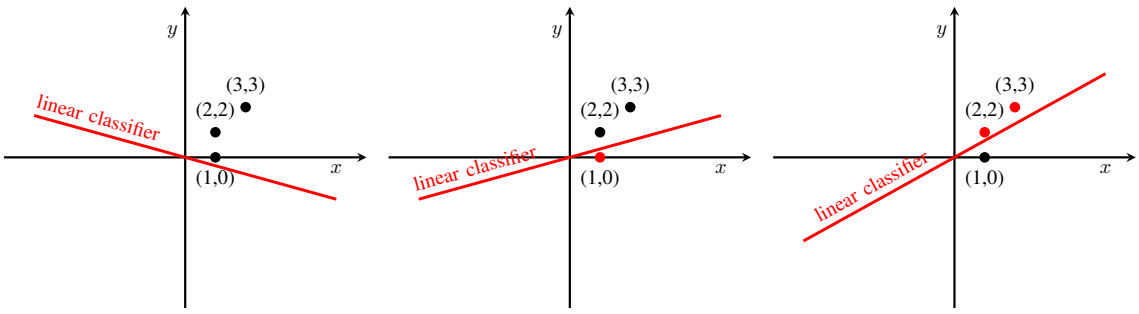
for all non-zero $\mathbf{u}^{(g)}, \mathbf{v}^{(g)}$. The Lasso variables $\mathbf{u}^{(g)}$ and $\mathbf{v}^{(g)}$ represent weights corresponding to positive versus negative final-layer weights α_g. An optimal solution defines one optimal network, however, all optimal solutions can be found via the optimal set of 12 up to permutation and splitting [43]. The reconstruction (13) obeys an optimal scaling property of neural networks, described next [29].

Optimal Neural Scaling [29]. The ReLU neural network $f(\mathbf{x}) = \sum_{j=1}^m \sigma(\mathbf{x}^T\mathbf{w}^{(j)})_+\alpha_j$ is invariant to multiplying $\mathbf{w}^{(j)}$ and dividing α_j by any positive scalar γ_j. On the other hand, for fixed $\mathbf{w}^{(j)}, \alpha_j$, their training regularization $\|\gamma_j\mathbf{w}^{(j)}\|_2^2 + |\frac{1}{\gamma_j}\alpha_j|^2$ is minimized over γ_j when $\|\gamma_j\mathbf{w}^{(j)}\|_2 = |\frac{1}{\gamma_j}\alpha_j|$, i.e., $\gamma_j^* = \frac{|\alpha_j|}{\|\mathbf{w}^{(j)}\|_2}$. Therefore an optimal neural network that minimizes the regularized training problem will have equal magnitude inner and outer weights $|\alpha_j| = \|\mathbf{w}^{(j)}\|_2$ for all j. Intuitively, this means that an optimal network balances weights evenly between layers. Similar scaling properties hold for deeper networks and other activations. Next we give an example of the convex formulation of a neural network.

Example [40]: Consider the training data matrix

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 \\ 2 & 3 & 0 \end{bmatrix}. \tag{14}$$

Although there are $2^3 = 8$ distinct binary sequences of length 3, $\mathbf{X}$ has $G = 3$ hyperplane arrangements in this case excluding the 0 matrix, as illustrated below from [40].



$$D_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad D_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad D_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Consider training a 2-layer ReLU network on the data matrix in (14). For an arbitrary label vector $\mathbf{y} \in \mathbb{R}^3$ and the squared loss, the network in the equivalent convex program (12) is written piecewise linearly as

$$f(\mathbf{X}) = \mathbf{D}_1\mathbf{X}^T(\mathbf{u}^{(1)} - \mathbf{v}^{(1)}) + \mathbf{D}_2\mathbf{X}^T(\mathbf{u}^{(2)} - \mathbf{v}^{(2)}) + \mathbf{D}^{(3)}\mathbf{X}^T(\mathbf{u}^{(3)} - \mathbf{v}^{(3)}). \tag{15}$$

This neural network (15) in the convex group Lasso (12) is interpretable using a sparse signal processing perspective [33]: it looks for a group sparse model to explain the response $\mathbf{y}$ via a mixture of linear models. The linear term $\mathbf{u}^{(2)} - \mathbf{v}^{(2)}$ predicts on $\{\mathbf{x}_1, \mathbf{x}_2\}$, and $\mathbf{u}^{(3)} - \mathbf{v}^{(3)}$ on $\{\mathbf{x}^{(3)}\}$, etc. Due to the regularization term $\sum_{g=1}^{3} \|\mathbf{u}^{(g)}\|_2 + \|\mathbf{v}^{(g)}\|_2$ in (12), only a few of these linear terms will be non-zero at the optimum, showing a bias towards simple solutions among all piecewise linear models. The convex formulation therefore offers insights into the role of regularization in preventing overfitting. While the hyperplanes can be enumerated by hand for the 2-D example data (14), for data in high dimensions, enumerating all hyperplanes becomes impractical. The next section addresses the complexity of enumerating hyperplane arrangements and how to reduce the complexity.

Computational Complexity of Global Optimization and Randomized Sampling for Guaranteed Approximation: The complexity of solving the convex neural network program is proportional to $(\frac{n}{d})^d$, where n is the number of samples and d is the dimension of the input. Although the exponential dependence with respect to d is unavoidable, this is significantly lower than brute-force search over the linear regions of ReLUs ($O\left(2^{md}\right)$) [3]. In [44], it was proven that the patterns can be randomly subsampled to lower the complexity to $\frac{n}{d}\log n$ with a guaranteed $\sqrt{\log n}$ relative approximation of the objective. This enables fully polynomial-time approximation schemes for the convex neural network program. "Sampling Arrangement Patterns" discusses hyperplane sampling and its relation to signal processing techniques and geometry.

Hyperplane Arrangements and Zonotopes

Hyperplane arrangements (11) can be described geometrically. Specifically, the hyperplane arrangements of $\mathbf{X}$ correspond to vertices of the *zonotope* (Figure 2) of $\mathbf{X}$, defined as

$$\mathcal{Z} = \text{Conv}\left\{\sum_{i=1}^{n} h_i \mathbf{x}_i : h_i \in \{0,1\}\right\} = \{\mathbf{X}^T\mathbf{h} : \mathbf{h} \in [0,1]^n\}. \tag{16}$$

The correspondence is as follows [40]. Since $\mathcal{Z}(\mathbf{X})$ is a polytope, for every $\mathbf{w} \in \mathbb{R}^d$, there is a vertex $\mathbf{z}^*$ of $\mathcal{Z}$ such that

$$(\mathbf{z}^*)^T \mathbf{w} = S(\mathbf{w}) = \max_{\mathbf{z}\in\mathcal{Z}} \mathbf{z}^T\mathbf{w}. \tag{17}$$

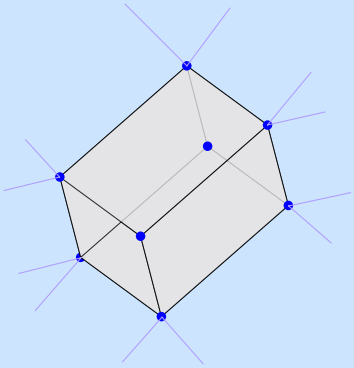


Fig. 2: Zonotope example. Lines indicate normal cones.

The line $\mathbf{z}^T\mathbf{w} = (\mathbf{z}^*)^T\mathbf{w}$ is a supporting hyperplane of $\mathcal{Z}$, shown in Figure 3, and $S(\mathbf{w})$ is the support function of $\mathcal{Z}$. We have shown that vertices maximize the support function of $\mathcal{Z}$. Conversely, for every vertex $\mathbf{z}^*$, there is a $\mathbf{w}$ such that that $\mathbf{z}^* = \arg\max_{\mathbf{z}\in\mathcal{Z}} \mathbf{z}^T\mathbf{w}$. Now, (17) is equivalent to

$$\max_{h_i\in[0,1]} \sum_{i=1}^{n} h_i \mathbf{x}_i^T \mathbf{w}$$

whose solution is a hyperplane arrangement $\mathbf{h} = \mathbf{1}\{\mathbf{X}^T\mathbf{w} \geq 0\}$. Therefore each vertex $\mathbf{z}^*$ corresponds to an activation chamber $\{\mathbf{w} : \text{sign}\left(\mathbf{X}^T\mathbf{w}\right) = \mathbf{h}\}$.

The proof of Theorem 1 involves showing that the bidual of the non-convex training problem is equivalent to the Lasso problem and reconstructing a network from the Lasso problem that achieves the same objective in the training problem as the Lasso problem, thus closing the duality gap and proving equivalence. The next section describes the key convex duality elements of the proof.

Convex duality of 2-layer ReLU networks. "Optimal Neural Scaling" shows that the optimal regularization in the training problem (10) is $\beta \sum_{j=1}^{m} \|\mathbf{w}^{(j)}\|_2 |\alpha_j|$ and the network can be written as $f(\mathbf{x}) = \sum_{j=1}^{m} \sigma(\mathbf{x}^T \frac{\mathbf{w}^{(j)}}{\|\mathbf{w}^{(j)}\|_2})_+$. Subsume $\|\mathbf{w}^{(j)}\|_2 \alpha_j \to \alpha_j$ and rewrite the training problem (10) as

$$\begin{aligned} \min_{\|\mathbf{w}^{(j)}\|_2=1, \boldsymbol{\alpha}} & \frac{1}{2}\|\mathbf{u} - \mathbf{y}\|^2 + \beta\|\boldsymbol{\alpha}\|_1 \\ \text{s.t. } & \mathbf{u} = \sum_{j=1}^{m} (\mathbf{X}^T\mathbf{w}^{(j)})_+ \alpha_j. \end{aligned} \tag{18}$$

The Lagrangian of (18) has linear and l_1 norm terms of α, so minimizing it over α gives the dual of (18):

$$\begin{aligned} \max & -\frac{1}{2}\|\mathbf{v} - \mathbf{y}\|_2^2 \\ \text{s.t. } & \max_{\|\mathbf{w}\|_2\leq 1} |\lambda^T (\mathbf{X}^T\mathbf{w})_+| \leq \beta. \end{aligned} \tag{19}$$

While (19) has a semi-infinite constraint over all $\|\mathbf{w}\|_2 \leq 1$, there are in fact a finite number of unique possible vectors $\left(\mathbf{X}^T\mathbf{w}\right)_+$, corresponding to activation chambers $\mathcal{K}^{(g)}$, as shown in "Hyperplane Arrangement Patterns." Maximizing over each $\mathcal{K}^{(g)}$ makes the constraint in (18) finite and linear:

$$\begin{aligned} &\max \frac{1}{2}\|\mathbf{v}-\mathbf{y}\|_2^2 \\ &\text{s.t.} \quad \max_{\|\mathbf{w}\|_2 \leq 1, \mathbf{w} \in \mathcal{K}^{(g)}} |\lambda^T \mathbf{D}_g \mathbf{X}^T \mathbf{w}| \leq \beta \text{ for all } g = 1, \cdots, G \end{aligned} \tag{20}$$

Taking the dual of the (20), assigning $\mathbf{u}^{(g)}, \mathbf{v}^{(g)}$ to correspond to positive and negative signs inside the absolute value, and simplifying gives the Lasso problem. The Lasso problem is therefore a lower bound on the training problem. However, the reconstruction (13) gives a network that achieves the same objective, and thus the Lasso problem and training problem are equivalent. **Hyperplane enumeration gives the key equivalence of** (19) **and** (20)**, and the key Lasso duality is the equivalence of** (20) **and** (12). Similar approaches are adapted to derive the convex equivalents for other network architectures, discussed next.

Sampling Arrangement Patterns

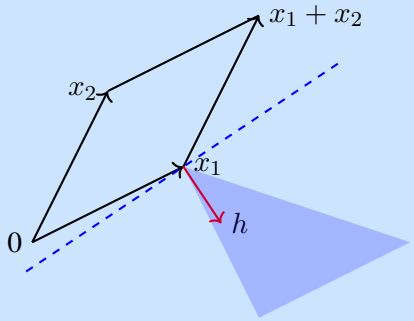


Fig. 3: Zonotope normal cones

In practice, one can use a smaller subset of hyperplane arrangements (11) in the Lasso problem (12) to reduce the exponential complexity of enumerating all arrangements. This approximation yields a subsampled form of the Lasso problem, and is proven to correspond to stationary points of the non-convex training problem. [40].

Suppose we sample hyperplane arrangements by generating a random $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I}_d)$ and evaluating $1\{\mathbf{X}^T\mathbf{w} \geq 0\}$. This implicitly samples vertices of the data's zonotope (16), whose normal cone solid angles (Figure 3) are proportional to the probability of sampling each vertex [40].

The hyperplane sampling approach resembles 1-bit compressive sensing [35], which randomly samples a signal $\mathbf{x} \in \mathbb{R}^d$ as $\text{sign}(\mathbf{x}^T\mathbf{w}_i)$ where $\mathbf{w}_i = \mathbf{a}_i, \mathbf{x}_i = \mathbf{z}_i$ in (7). Multiple signals stacked into a data matrix $\mathbf{X}$ can be randomly sampled at once as $\text{sign}\left(\mathbf{X}^T\mathbf{w}\right)$ where $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I}_d)$, which is precisely sampling activation chambers, or equivalently, the vertices of a zonotope.

Sampling patterns is also related to Locality-Sensitive Hashing (LSH), a method for efficiently finding nearest neighbors (in Euclidean distance) of an entry $\mathbf{x} \in \mathbb{R}^d$ in a database [45]. LHS places $\mathbf{x}$ in a database bin according to the sign of $\mathbf{x}^T\mathbf{w}$ where $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I}_d)$, similar to 1-bit CS.

Vector output ReLU networks and nuclear norm extension of Lasso

The convexification result in Theorem 1 extends to vector-output networks. A 2-layer vector-output network has the same architecture as (2) but the outer layer weight vector $\boldsymbol{\alpha}$ becomes a matrix $\bar{\boldsymbol{\alpha}} \in \mathbb{R}^{m \times c}$, making the output a vector:

$$f(\mathbf{x}) = \sigma(\mathbf{x}^T \mathbf{W}) \bar{\boldsymbol{\alpha}} \in \mathbb{R}^c \tag{21}$$

and the labels are now vectors $\mathbf{y}_1, \cdots, \mathbf{y}_n \in \mathbb{R}^c$. Vector-output networks are used in multi-class classification and multidimensional regression. Shallow vector-output networks are useful in layer-wise training, where a network is trained one layer at a time by freezing the rest of the network. Let $\mathbf{Y}$ be the matrix with columns $\mathbf{y}_i$. We still have $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{W} \in \mathbb{R}^{d \times m}$. The training problem for a 2-layer vector-output network (21) is analogous to that for scalar networks (10):

$$\min_{\mathbf{W}, \bar{\boldsymbol{\alpha}}} \frac{1}{2} \left\| f(\mathbf{X}) - \mathbf{Y} \right\|_F^2 + \beta \left(\|\mathbf{W}\|_F^2 + \|\boldsymbol{\alpha}\|_F^2 \right) \tag{22}$$

Suppose the activation is ReLU. The convex equivalence in Theorem 1 applies to the vector output case (22) with the following modifications: the vector variables $\mathbf{u}^g, \mathbf{v}^g$ become matrices, their group l_1 regularizations become nuclear norms, and their activation chambers change from vector sets $\mathcal{K}^{(g)}$ to matrix sets $\mathcal{K}^{(g)'} = \{\mathbf{u}\mathbf{v}^T : \mathbf{u} \in \mathcal{K}^{(g)}, \mathbf{v} \in \mathbb{R}^d\} \cap \mathcal{B}_*$ where $\mathcal{B}_* = \{\mathbf{Z} : \|\mathbf{Z}\|_* \leq 1\}$ is the nuclear norm ball of radius 1 [4]. Formally, there exists $m^* \leq nc + 1$ such that if $m \geq m^*$, training (22) is equivalent to

$$\min_{\mathbf{U}^{(g)}, \mathbf{V}^{(g)} \in \mathcal{K}^{(g)'}} \frac{1}{2} \| \sum_{g=1}^{G} \mathbf{D}_g \mathbf{X}^T \left(\mathbf{U}^{(g)} - \mathbf{V}^{(g)} \right) - \mathbf{y} \|_2^2 + \beta \sum_{g=1}^{G} \left(\|\mathbf{U}^{(g)}\|_* + \|\mathbf{V}^{(g)}\|_* \right) \tag{23}$$

where the diagonal matrices $\mathbf{D}_g$ in (23) are defined similarly as in the scalar output ("Hyperplane Arrangement Patterns") [4]. The above problem is regularized via a group nuclear norm regularization, which encourages low-rank solutions [36]. Under the hood, the above result shows that vector output ReLU networks are piecewise low-rank models that select neurons sparingly. Training vector-valued networks is also equivalent to copositive programs, which are optimization problems whose variables are copositve matrices [4].

So far, we have focused on networks with the traditionally used ReLU activation. The convex equivalences extend to other activations as well, discussed next.

Activations beyond ReLU

In this section, we discuss convex reformulations of networks with more general activations, which are useful for different types of data and tasks.

Polynomial Activations and Semidefinite Programming: Polynomial activations have historically offered an attractive alternative to ReLU for theoretical analysis due to their smoothness, which can also help in training. Recently, there has been renewed interest in polynomial activations, as they enable training on encrypted data [46]. A 2-layer network with quadratic activation $\sigma(x) = x^2$ is

$$f(\mathbf{x}) = \sum_{j=1}^{m} \left(\mathbf{x}^T \mathbf{w}^{(j)} \right)^2 \alpha_j \tag{24}$$

Fig. 4: (Left) The Neural Cone $\mathcal{C}_2^1$ described by $(u^2\alpha, u\alpha, \alpha) \in \mathbb{R}^3$ where $u, \alpha \in \mathbb{R}, |u| \leq 1$. (Right) Neural Spectrahedron $\mathcal{M}(1)$ described by $(Z_{11}, Z_{12}, Z_{22}) \in \mathbb{R}^3$ where $Z = \begin{bmatrix} Z_{11} & Z_{12} & Z_{13} \\ Z_{12} & Z_{22} & Z_{23} \\ Z_{13} & Z_{23} & Z_{33} \end{bmatrix} \succeq 0$, $Z_{11} + Z_{22} = Z_{33} \leq 1$ constrained to a slice of the spectrahedron, which is higher dimensional [23].

where $\mathbf{w}^{(j)}$ are the columns of $\mathbf{W}$ and α_j are the elements of $\boldsymbol{\alpha}$ in (3). Consider a training problem for a quadratic activation network (24) with cubic instead of quadratic regularization:

$$\min_{\mathbf{W},\boldsymbol{\alpha}} \frac{1}{2} \|f(\mathbf{X}) - \mathbf{y}\|^2 + \frac{\beta}{c} \sum_{j=1}^{m} \left(\|\mathbf{w}^{(j)}\|^3 + |\alpha_j|^3 \right) \tag{25}$$

where $c = 2^{\frac{1}{3}} + 2^{-\frac{1}{3}} \approx 1.89$. The non-convex training problem (25) is equivalent to the following convex semidefinite program (SDP) with nuclear norm regularization [23]:

$$\begin{aligned} \min_{\mathbf{U} \in \mathbb{S}^n, \mathbf{z} \in \mathbb{R}^n} & \frac{1}{2} \|\mathbf{z} - \mathbf{y}\|^2 + \beta \|\mathbf{U}\|_* \\ \text{s.t. } & z_i = \mathbf{x}_i^T \mathbf{U} \mathbf{x}_i, i = 1, \cdots, n \end{aligned} \tag{26}$$

provided that the number of neurons is $m \geq m^* = \text{rank}(\mathbf{U}^*)$. The complexity of solving the SDP (26) is polynomial in n, d and m [23]. There also exists an equivalent polynomial-time solvable SDP if the cubic regularization in the training problem (25) is replaced with an l_1 penalty $\beta \|\boldsymbol{\alpha}\|_1$ and the activation is a general quadratic function $\sigma(x) = ax^2 + bx + c$, under a normalization constraint on the inner weights [23]. An optimal neural network can be found through the eigenvalue decomposition of $\mathbf{U}$ [23].

Threshold Activation: The threshold activation is $\sigma(x) = s\mathbf{1}\{x \geq 0\}$, where the scalar s is a trainable amplitude parameter. Since threshold activations output only one of two values, they are useful in hardware implementations with memory, power, and computational complexity constraints, as well as quantizing neural network parameters. Threshold activations can also model biological neurons which communicate through binary signals of firing action potentials. A 2-layer neural network with threshold activation is

$$f(\mathbf{x}) = \sum_{j=1}^{m} s_j \mathbf{1}\{\mathbf{x}^T \mathbf{w}^{(j)} \geq 0\} \alpha_j. \tag{27}$$

Application	Architecture	Lasso Features	Lasso Regularization	Complexity	Details
regression	Scalar output, ReLU 2-layer	$\mathbf{D}_i\mathbf{X}$	l_1 group	$O(\frac{n}{r})^r$	[3]
multi-class classification, layer-wise learning	Vector output, ReLU 2-layer	$\mathbf{D}_i\mathbf{X}$	nuclear norm	$O(n^r)$	[4]
more expressive networks for complex data	Scalar output, ReLU 3-layer	$\mathbf{D}_i\mathbf{D}_j\mathbf{X}$	l_1 group	poly(n,d) exp(r)	[47]
cryptography	polynomial activation, 2-layer	$\mathbf{x}_i(\cdot)\mathbf{x}_i^T$ SDP	nuclear norm	poly(n,d,m)	[23]
memory/energy efficiency, quantization	threshold activation 2-layer	diag$(\mathbf{D}_i)$	l_1	$O(n^{3r})$	[16]
time-series data	scalar output, 1-D, 2-layer	$\sigma(\mathbf{x}-x_j\mathbf{1})$	l_1	$O(n^3)$	[42]

TABLE I: Neural networks and properties of their equivalent convex programs. $\mathbf{D}_i$ is a hyperplane encoding matrix and diag$(\mathbf{D}_i)$ is the diagonal vector of $\mathbf{D}_i$. The number of training samples and neurons are n and m, respectively. The dimension of the training data is d and the rank of the data matrix is r.

Let $\mathbf{s}$ be the vector of s_j's. The non-convex training problem for a 2-layer threshold network (27) is

$$\min_{\mathbf{W},\boldsymbol{\alpha},\mathbf{s}} \frac{1}{2}\left\|f(\mathbf{X})-\mathbf{y}\right\|^2+\beta\left(\|\mathbf{W}\|_F^2+\|\boldsymbol{\alpha}\|_2^2+\|\mathbf{s}\|_2^2\right). \tag{28}$$

The training problem (28) regularizes the amplitude parameters s_j along with the weights, as they are all trainable. Let $\mathbf{A}$ be a matrix whose columns are hyperplane arrangement patterns $\mathbf{h}=\mathbf{1}\{\mathbf{X}^T\mathbf{w}\geq 0\}\in\{0,1\}^n$ (11). The nonconvex training problem (28) is equivalent to the unconstrained Lasso problem

$$\min_{\mathbf{z}} \frac{1}{2}\|\mathbf{A}\mathbf{z}-\mathbf{y}\|_2^2+\beta\|\mathbf{z}\|_1\,, \tag{29}$$

provided $m\geq m^*$, which is the number of nonzero z_i^* [16].

In contrast to the networks with ReLU activation, the convex equivalent is a standard Lasso problem instead of a group Lasso, and has no constraints. The hyperplane arrangement patterns themselves are the features of the Lasso.

The complexity of solving the convex Lasso problem (29) is $O(n^{3r})$ [16]. An optimal neural network can be reconstructed by finding weights corresponding to the activation patterns in the dictionary [16]. Deeper networks with threshold activation can be similarly formulated as convex, unconstrained Lasso problems, but with an expanded dictionary whose columns correspond to multilevel hyperplane arrangement patterns [16]. The next sections discuss deeper networks with other activations.

DEEPER RELU NETWORKS

While shallow networks suffice in some situations, in practice, deeper networks are used to capture more complex relationships between data. Is it possible to equate deeper networks to convex models? It will be shown that it depends on the architecture.

Importance of Parallel Architecture

We consider two architectures for a deep neural network. First, a L-layer *standard* network extends the 2-layer network (3) by simply composing more layers as $f : \mathbb{R}^d \to \mathbb{R}$,

$$f(\mathbf{x}) = \left(f^{(L-1)} \circ f^{(L-2)} \circ \cdots \circ f^{(1)}(\mathbf{x}) \right) \boldsymbol{\alpha} + \xi \tag{30}$$

where $f^{(l)}(\mathbf{x}) = \sigma\left(\mathbf{x}^T \mathbf{W}^{(l)} + \mathbf{b}^{(l)}\right)$ is the l^{th} layer (2) and the trainable parameters are weight matrices $\mathbf{W}^{(l)}$ and bias vectors $\mathbf{b}^{(l)}$ for $l \in [L-1]$, and final layer parameters $\boldsymbol{\alpha}$ and ξ. An alternative architecture is a L-layer *parallel network* which linearly combines multiple units of composed layers as $f : \mathbb{R}^d \to \mathbb{R}$,

$$f(\mathbf{x}) = \sum_{i=1}^{m_L} \left(f^{(i,L-1)} \circ f^{(i,L-2)} \circ \cdots \circ f^{(i,1)}(\mathbf{x}) \right) \alpha_i + \xi \tag{31}$$

where $f^{(i,l)}(\mathbf{x})(\mathbf{x}) = \sigma\left(\mathbf{x}^T \mathbf{W}^{(i,l)} + \mathbf{b}^{(i,l)}\right)$ is the l^{th} layer (2) of the i^{th} unit, which consists of m_l neurons. The trainable parameters are the outermost weights $\boldsymbol{\alpha} \in \mathbb{R}^{m_L}$, an external bias $\xi \in \mathbb{R}$; and the inner weights $\mathbf{W}^{(i,l)} \in \mathbb{R}^{m_{l-1} \times m_l}$ and inner biases $\mathbf{b}^{(i,l)} \in \mathbb{R}^{m_l}$ for each layer $l \in [L-1]$ and unit $i \in [m_L]$, where $m_0 = d$. The standard (30) and parallel (31) architectures have both been studied in literature [48], [27], and are equivalent for a 2-layer network.

As shown in "Convex Duality of 2-layer ReLU Networks", the convex formulation of 2-layer networks is found as the bidual of the non-convex training problem, which gives a lower bound on the training problem. This lower bound is met with equality for 2-layer networks; in other words, strong duality is achieved and there is no duality gap, so the convex formulation is equivalent to the training problem [48]. Applying the same duality approach to deeper networks with a parallel architecture results in an equivalent convex problem with no duality gap [48]. However, applying this approach for standard networks, even with a linear activation function such as $\sigma(x) = x$, results in a nonzero duality gap [48]. Therefore, a parallel architecture is necessary for extending this convex analysis to more layers. The next section describes the equivalent convex formulation for a 3-layer ReLU parallel network.

3-layer ReLU Networks with Path Regularization

The group Lasso convex formulation (12) for 2 layers extends to deeper networks [10]. In these formulations, the diagonal matrices $\{\mathbf{D}_g\}_{g=1}^G$ that encode linear separation patterns are replaced with a product of recursive linear separation patterns of the form $\{\mathbf{D}_i \mathbf{D}_j\}_{i=1,j=1}^{G_1,G_2}$. This results in a convex program with more complex hyperplane patterns, which are fitted to the data via a sparsity inducing regularization term. Consider the training problem for a 3-layer parallel ReLU network $f(\mathbf{x}) = \sum_{i=1}^{m_3} \left(\left(\mathbf{X}^T \mathbf{W}^{(i,1)} \right)_+ \mathbf{W}^{(i,2)} \right)_+ \alpha_i$ with *path regularization*:

$$\min_{\mathbf{W},\boldsymbol{\alpha}} \frac{1}{2} \sum_{i=1}^{n} \left(f(\mathbf{x}_i) - y_i \right)^2 + \beta \sum_{i=1}^{m} \sqrt{\sum_{j,k} \left\| \mathbf{w}_j^{(i,1)} \right\|_2^2 \left(w_{j,k}^{(i,2)} \right)^2 \alpha_k^2} \tag{32}$$

Fig. 5: Generic shape of $\mathbf{A}^+ \in \mathbb{R}^{N\times N}$ defined by ${\mathbf{A}^+}_{i,n} = \sigma(x_i - x_n)$, where σ is ReLU (left) and sign activation (right). Each i^{th} curve represents a feature. The points $(i, n, {\mathbf{A}^+}_{i,n})$ are plotted in 3-D, with ${\mathbf{A}^+}_{i,n}$ represented by the curve height and color. Here, $n \in [N]$ but each curve interpolates between integer values of n [42].

where $\mathbf{w}_j^{(i,1)}$ is the j^{th} column of $\mathbf{W}^{(i,1)} \in \mathbb{R}^{d\times m_1}$ and $w_{j,k}^{(i,2)}$ is the $(j,k)^{\text{th}}$ element of $\mathbf{W}^{(i,2)} \in \mathbb{R}^{m_1\times m_2}$. The regularization penalizes all of the paths from the input through the m_3 parallel networks to the output. The 3-layer ReLU training problem (32) is equivalent to the convex group Lasso problem

$$\min_{\mathbf{u}^{(i,j,k)},\mathbf{v}^{(i,j,k)}\in\mathcal{K}^{(i,j)}} \frac{1}{2}\|\sum_{i=1}^{G_1}\sum_{j,k=1}^{G_2} \mathbf{D}_i\mathbf{D}'_{j,k}\mathbf{X}^T(\mathbf{u}^{(i,j,k)} - \mathbf{v}^{(i,j,k)}) - \mathbf{y}\|_2^2 + \beta\sum_{i,j}\left(\|\mathbf{u}^{(i,j,k)}\|_2 + \|\mathbf{v}^{(i,j,k)}\|_2\right) \tag{33}$$

where $\mathbf{D}_i, \mathbf{D}'_{j,k}$ are diagonal matrices encoding hyperplane arrangement patterns in the first and second layer, and $K^{(i,j)}$ are multilayer activation chambers [47]. The multilayer hyperplanes divide the 2-layer activation chambers $\mathcal{K}^{(i)}$ into subchambers $\mathcal{K}^{(i,j)}$. This partitions the data into finer, more granular regions over which the network acts as local linear models. The convex formulation shows that a deeper network has higher representation power because it can learn the structure of a diversely populated data set more deeply through its activation patterns and can tailor its implicit linear models more locally in each activation chamber. The convex formulation provides a geometric interpretation of the network operations, revealing the underlying sparsity structure. A network reconstruction formula is given in [47].

Thus far, the dimension of the input data to a neural network has been arbitrary. Next, we momentarily focus on the special case of 1-D data, which offers concrete insights on neural network structure. Then we will return to arbitrary dimensional data and explore how insights from 1-D data can elucidate geometric structures in networks trained on more general data.

SPECIAL CASE: 1-D DATA REVEALS STRUCTURED FEATURES IN NEURAL NETWORKS

If the input data is 1-D, convexifying neural network training simplifies greatly, and the convex formulation gives new insights into neural networks. 1-D data occurs often in time series regression, for example predicting financial data such as stock prices [42]. Networks with 1-D data are equivalent to Lasso models without constraints and with explicit dictionaries, which removes the hyperplane enumeration task required for convexifying networks with high dimensional data (12). Akin to Lasso models [32] used for sparse signal processing, neural networks learn to represent data using as few neurons as possible.

1-D data and 2-layer networks: a simple Lasso model

Consider a 2-layer network (3) $f : \mathbb{R} \to \mathbb{R}$ with 1-D data input $x \in \mathbb{R}$:

$$f(x) = \sum_{j=1}^{m} \sigma(xw^{(j)} + b^{(j)})\alpha_j + \xi, \tag{34}$$

where σ is a general piecewise linear activation: ReLU, absolute value $\sigma(x) = |x|$, leaky ReLU $\sigma(x) = a(x)_+ + b(-x)_+$, sign $\sigma(x) = s\text{sign}(x)$ (where s is a trainable amplitude), or threshold activation. The network in (34) explicitly includes the bias parameters, as they perform a key role in the derivation of the convex problem. The training problem is (27) for sign and threshold activations to account for amplitude parameter regularization, and (10) otherwise. This non-convex training problem is equivalent to the unconstrained convex Lasso problem (5) where we evenly partition the dictionary matrix $\mathbf{A}$ and variable $\mathbf{z}$ as $\mathbf{A} = [\mathbf{A}^+ \mathbf{A}^-] \in \mathbb{R}^{n \times 2n}, \mathbf{z} = \begin{pmatrix} \mathbf{z}^+ \\ \mathbf{z}^- \end{pmatrix} \in \mathbb{R}^{2n}$. Specifically, the equivalent Lasso problem is

$$\min_{\mathbf{z},\xi} \frac{1}{2} \|\mathbf{A}^+ \mathbf{z}^+ + \mathbf{A}^- \mathbf{z}^- + \xi \mathbf{1} - \mathbf{y}\|_2^2 + \beta \|\mathbf{z}\|_1 \tag{35}$$

provided that $m \geq g^*$, the cardinality of $\mathbf{z}^*$ [42]. The submatrices of the dictionary are $\mathbf{A}^+, \mathbf{A}^- \in \mathbb{R}^{n \times n}$ with $A^+_{i,j} = \sigma(x_i - x_j), A^-_{i,j} = \sigma(x_j - x_i)$. For symmetric activations such as $\sigma(x) = |x|$, we can replace $\mathbf{A}$ by $\mathbf{A}^+$ and $\mathbf{z}$ by $\mathbf{z}^+$. Let $\mathcal{J}^+ = \{j : z_j^{+*} \neq 0\}, \mathcal{J}^- = \{j : z_j^{-*} \neq 0\} \subset [n]$ be the non-zero indices of the Lasso (35) solution. Partitioning parameters into two subsets designated by superscripts $+$ and $-$, an optimal network (34) is reconstructed from an optimal Lasso solution $\mathbf{z}^*, \xi^*$ as

$$f(x) = \sum_{j \in J^+} \sigma(x \underbrace{-x_j}_{b^{(j,+)}}) \underbrace{z_j^{+*}}_{\alpha_j^+} + \sum_{j \in J^-} \sigma(\underbrace{x_j}_{b^{(j,-)}} - x) \underbrace{z_j^{-*}}_{\alpha_j^-} + \underbrace{\xi^*}_{\xi} \tag{36}$$

where the weights are $w^{(j,+)} = 1, w^{(j,-)} = -1$ and then rescaled according to "Optimal Neural Scaling". The reconstructed network (36) has at most $2n$ neurons. The reconstructed optimal network (36) is piecewise linear, with breakpoints between the linear pieces at training samples x_j. When the training problem imposes minimal regularization, i.e., $\beta \to 0$, the network linearly interpolates the training data. The optimal network in (36) is a linear combination of *feature functions* $f_j^+, f_j^- : \mathbb{R} \to \mathbb{R}$ of the form

$$f_j^+(x) = \sigma(x - x_j), \quad f_j^-(x) = \sigma(x_j - x) \tag{37}$$

which are linear functions with breakpoints at x_j. The columns of the dictionary matrix are called features. Each j^{th} feature of the dictionary submatrices $\mathbf{A}^+$ and $\mathbf{A}^-$ consists of a feature function (37) sampled at all of the training data: $A^+_{i,j} = f_j^+(x_i) = \sigma(x_i - x_j)$, and similarly $A^-_{i,j} = f_j^-(x_i)$. Figure 5 visualizes examples of features for ReLU and sign activation. The convex formulation (35) implies that an optimal neural network can be represented as a sparse linear combination of basis functions (37) centered at the training data points. The network can be interpreted as learning the training samples and approximately interpolating between them. The next section extends the 1-D, 2-layer analysis to deeper networks.

Deeper networks with 1-D data: Emergence of reflection features

As previously shown, networks with 2 layers and 1-D data are composed of feature functions (37) that are piecewise linear functions with breakpoints at training samples and which interpolate dictionary columns. Deeper networks also are composed of feature functions that are similar to the 2-layer case, but with more complex breakpoints representing geometric properties such as reflections. The *reflection* [42] of a point $\mathbf{a} \in \mathbb{R}^d$ about a point $\mathbf{b} \in \mathbb{R}^d$ is

$$R_{(\mathbf{a},\mathbf{b})} = \mathbf{b} + (\mathbf{b} - \mathbf{a}) = 2\mathbf{b} - \mathbf{a}. \tag{38}$$

Consider a 3-layer ReLU network with 1-D input and 2 neurons in the middle layer of each parallel unit:

$$f(x) = \sum_{j=1}^{m_3} \left(\left(x w_1^{(j,1)} + b_1^{(j,1)} \right)_+ w_1^{(j,2)} + \left(x w_2^{(j,1)} + b_2^{(j,1)} \right)_+ w_2^{(j,2)} \right)_+ \alpha_j + \xi. \tag{39}$$

The number of parallel units m_3 in (39) is arbitrary. Surprisingly, the simple ReLU architecture (39), with certain weight normalization constraints, is equivalent to the above Lasso formulation (35), where the columns of the dictionary matrix $\mathbf{A}$ are replaced by vectors called *reflection features* [42]. As similar to the 2-layer case, the reflection features represent the sampled values of piecewise linear feature functions with breakpoints at locations including training samples x_i. However, the 3-layer features can have additional breakpoints at reflections of data, of the form $R_{(x_i,x_j)}$ (38) [42]. Figure 7a from [42] illustrates features with breakpoints at reflections for 3-layer ReLU networks (39) trained on 1-D data. Next, we discuss extending these geometric observations to higher dimensional data.

HIGH DIMENSIONAL DATA: GEOMETRIC ALGEBRA AND FEATURES IN NEURAL NETWORKS

The Lasso formulation in the previous section can be extended to higher-dimensional data using Clifford Algebra, a mathematical framework that has been recently explored in signal processing [39]. The dictionary matrices $\mathbf{A}^+$ and $\mathbf{A}^-$ in the Lasso problem (35) are identified as encoding volumes of oriented simplices formed by the data points [38]. Namely, $(x_i - x_j)_+$ is the positive part of the signed volume of the interval $[x_i, x_j]$. The generalization is given by wedge products of the data points. The *wedge product* of vectors $\mathbf{a}$ and $\mathbf{b}$ is denoted as $\mathbf{a} \wedge \mathbf{b}$, which measures the signed area of the parallelogram spanned by $\mathbf{a}$ and $\mathbf{b}$. Figure 6 from [38] illustrates a wedge product in $\mathbf{R}^3$. The higher-dimensional dictionary elements are $A_{ij} = \frac{\left(\mathbf{x}_i \wedge \mathbf{x}_{j_1} \wedge \cdots \wedge \mathbf{x}_{j_{d-1}}\right)_+}{\|\mathbf{x}_{j_1} \wedge \cdots \wedge \mathbf{x}_{j_{d-1}}\|_2}$, leading to a convex program that captures the geometric relationships in the data [38]. Notably, A_{ij} is a ratio of volumes of parallelograms, where the positive part of the signed volume encoding the order of the indices.

As an example, consider a 2-layer network $f(\mathbf{x}) = \sigma(\mathbf{x}^T\mathbf{W})\boldsymbol{\alpha}$ trained on $\mathbf{x}_i \in \mathbb{R}^2, y_i \in \mathbb{R}$. The non-convex training problem (4) with l_2-norm loss and l_p-norm weight decay regularization [38] is equivalent to

$$\min_{\mathbf{z}} \|\mathbf{A}\mathbf{z} - \mathbf{y}\|_2^2 + \beta\|\mathbf{z}\|_1$$

where $A_{ij} = \frac{2}{\|\mathbf{x}_j\|_p}\mathbf{Vol}(\triangle(0, \mathbf{x}_i, \mathbf{x}_j))_+$ where $\triangle(a, b, c)$ is the triangle with vertices a, b, c and $\mathbf{Vol}$ is the 2-volume (area). [38]. If we add a bias term to the neurons then $A_{ij} = \frac{2}{\|\mathbf{x}_{j_1} - \mathbf{x}_{j_2}\|_p}\mathbf{Vol}(\triangle(\mathbf{x}_i, \mathbf{x}_{j_1}, \mathbf{x}_{j_2}))_+$

where $j = (j_1, j_2)$. Here, $\mathbf{Vol}(\cdot)_+$ distinguishes triangles based on the orientation, i.e., whether their vertices form clockwise or counter-clockwise loops. The convex problem gives an exact optimal network when the regularization is l_1-norm and an ϵ-optimal network when the regularization is l_2-norm [38].

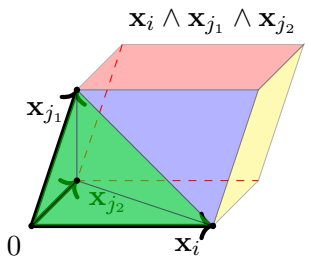
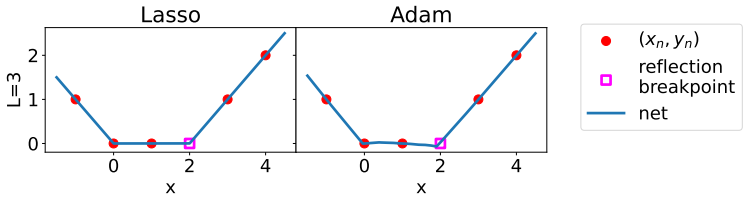


Fig. 6: Wedge product in geometric algebra [38].

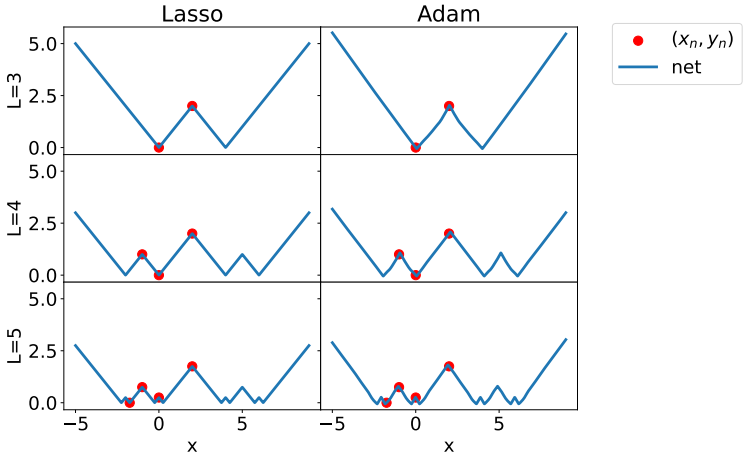
The geometric algebra approach provides a unified framework for interpreting deep neural networks as a dictionary of wedge product features, revealing the underlying geometric structures inherited from the training data.

Geometric algebra can be similarly used to describe equivalent convex Lasso problems for networks with other activations such as leaky ReLU activation and absolute value activation with arbitrary dimensional data [49]. In fact, parallel networks with absolute value activation demonstrate similar reflection features as ReLU networks (discussed previously), but reflections appear even with only one neuron per unit per layer. Specifically, a L-layer *deep narrow network* is

$$f(\mathbf{x}) = \sum_{i=1}^{m_L} \sigma\left(\cdots\left(\sigma\left(\sigma\left(\mathbf{x}^T\mathbf{W}^{(i,1)} + b^{(i,1)}\right) w^{(i,2)} + b^{(i,2)}\right)\cdots\right) w^{(i,L-1)} + b^{(i,L-1)}\right) \alpha_i + \xi. \tag{40}$$



(a) 3-layer ReLU network (39). The breakpoint at 2 is not a training point; it is the reflection of training points 0 across 1 [42].



(b) Deep narrow networks (40) with $\sigma(x) = |x|$. For L=3, the third breakpoint at 2 is not a training point; it is the reflection of training points 0 across 1. Deeper network predictions exhibit multilevel reflections.

Fig. 7: Lasso and Adam-trained deep narrow networks [42].

While 3-layer networks with ReLU activation need at least 2 neurons in each layer of each unit to exhibit reflection features [42], a 3-layer deep narrow network with absolute value activation and just one neuron per layer and unit has features with breakpoints at reflections of training data [49]. Deep narrow networks with 4 layers and absolute value activation have features with breakpoints at reflections of training data reflections. Figure 7b from [42] illustrates the breakpoints at deeper reflections for a neural network trained on sample 1-D data. Absolute value deep narrow networks exhibit increasingly multi-level symmetries and reflections with increasing depth, capturing more complex data relations [49]. For the exact convex Lasso formulations of deep

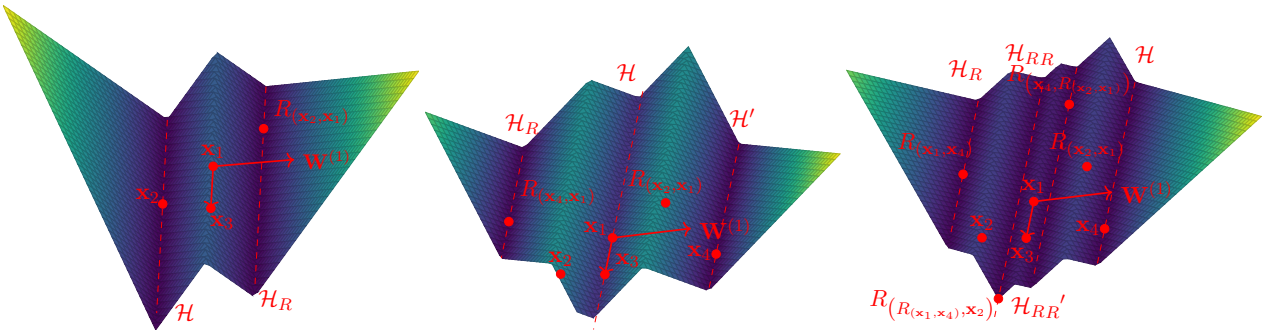


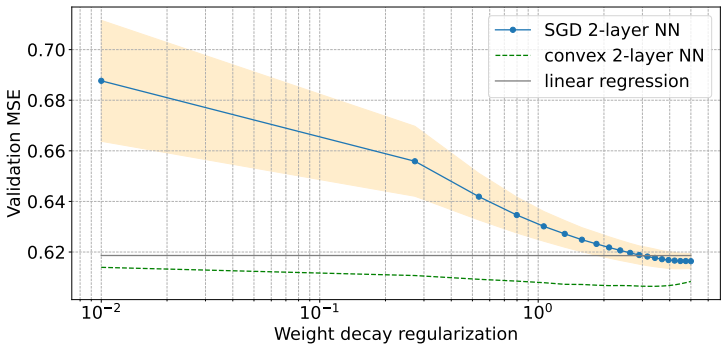
Fig. 8: Examples of 3-layer (left) and 4-layer (middle, right) deep narrow features for 2-D data. Red dashed lines indicate reflection planes (lines in $\mathbb{R}^2$) [49].

networks with higher dimensional data, feature breakpoints become breaklines and breakplanes. Figure 8 from [49] illustrates the Lasso features for 2-D data. The features have breaklines along data points ($\mathcal{H}$) as well as reflected breaklines ($\mathcal{H}_R$) and double reflected breaklines ($\mathcal{H}_{RR}$).

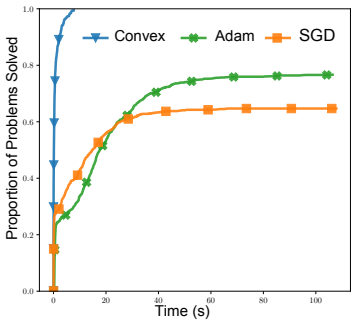
While the simplicity of absolute value activation leads to tractable and insightful convex formulations, similar convex formulations and features are observed in other piecewise linear activations such as leaky ReLU, with generalized versions of reflections [49]. This result establishes a theoretical foundation for the success of deep ReLU and absolute value networks by linking them to equivalent Lasso formulations whose dictionaries capture geometric and symmetry properties, providing novel insights into the impact of depth on neural network expressivity and generalization.

Extensions to Other Neural Network Architectures

The convex representation approach extends to modern architectures revealing new forms of convex regularizers. The following contains a brief overview. Convolutional Neural Networks (CNNs) are popular for image and audio problems and can be written as Lasso models by incorporating convolutional structures into the convex formulation [50]. Transformers and attention mechanisms are used in Large Language Models for understanding human language. They are convexified by modeling attention and utilizing nuclear norm regularization [15]. Generative Adversarial Networks (GANs) are used to generate new samples from a distribution, such as to produce synthetic images or audio. Training GANs can be represented as convex-concave saddle point problems [22]. Diffusion models are used to generate new images through a sequence of adding noise and denoising steps. Training diffusion models can be formulated as a convex problem by expressing the score-matching objective as a convex Lasso problem [51]. Convex reformulation techniques also reveal that batch normalization corresponds to applying a whitening matrix to the data [12]. Next, we discuss numerical experiments that demonstrate performance advantages of training neural networks by using their convex models.



(a) Time series forecasting of NY Stock Exchange.



(b) Performance profile.

Fig. 9: Experimental results comparing neural network training with non-convex and convex formulations.

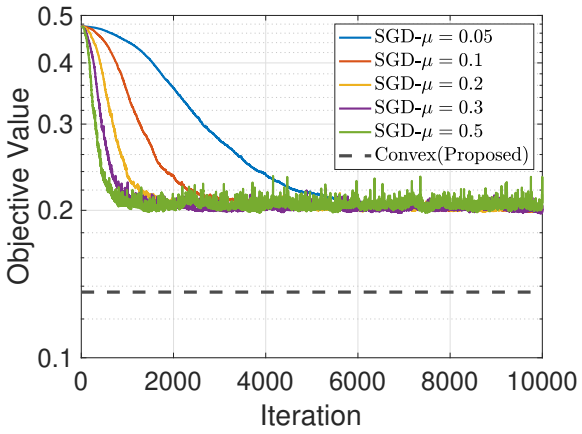
SIMULATIONS

Autoregressive Signal Prediction

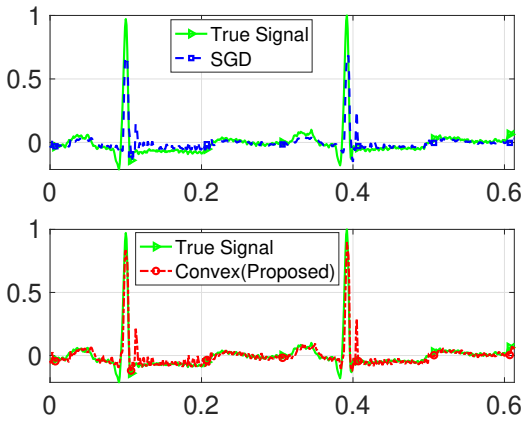
In autoregression, a signal $\mathbf{x} \in \mathbb{R}^d$ at time t is modeled as a function of its past T samples $\mathbf{x}_t = f_\theta(\mathbf{x}_{t-1}, \cdots, \mathbf{x}_{t-T})$ where θ is a parameter set for f. A linear autoregressive model is $f_\theta(\mathbf{x}_{t-1}, \cdots, \mathbf{x}_{t-T}) = a_1\mathbf{x}_{t-1} + \cdots + a_T\mathbf{x}_{t-T}$ where the parameter coefficients $\theta = \{a_1, \cdots, a_T\}$ are estimated from the data. Alternatively, the autoregressive model f_θ can be modeled by a neural network for a more expressive, non-linear model. The performance improvement of using the convex model to train neural networks on autoregression is tested in the following two experiments on financial data and ECG data.

The first experiment performs time series forecasting on the New York stock exchange dataset [52]. The log volume of exchange is predicted as the target from the log volume, Dow Jones return, and log volatility in the past 5 time steps. Linear and non-linear autoregressive models are analyzed. Figure 9a illustrates the results. The networks trained with a convex program (green dashed line) have better performance than those trained with the non-convex training program (blue). And both convex and non-convex trained neural networks perform better than a linear model (black line). Also note the significant variability in SGD predictions indicated by the shaded area representing one standard deviation across 10 random initializations. In contrast, the convex solver produces consistent results regardless of initialization.

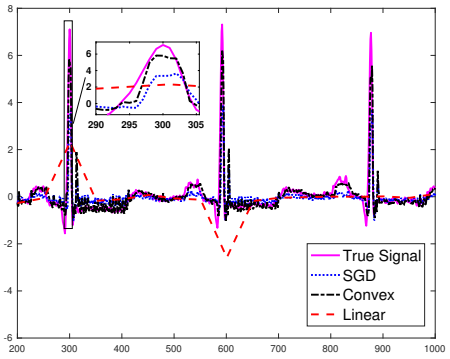
The second experiment, from [40], measures the performance of neural network training algorithms in a time series prediction task using ECG data. Results are illustrated in Figure 10. Each sample consists of three consecutive voltage values as features, aiming to predict the next voltage value. The dataset contains $n = 2393$ observations with $d = 3$ features. A two-layer ReLU network is trained using both stochastic gradient descent (SGD) and the convex Lasso method. The SGD was performed with a batch size of 100 and a grid of learning rates μ. As shown in Fig. 10, the convex optimization method outperforms SGD in both training loss and test prediction accuracy. The SGD fails to achieve the optimal training objective value obtained by the convex method, leading to poorer generalization. This demonstrates the practical advantages of the convex approach in signal processing applications. The Lasso formulation also reveals why neural networks perform better than linear methods. In contrast to a linear classifier which treats all



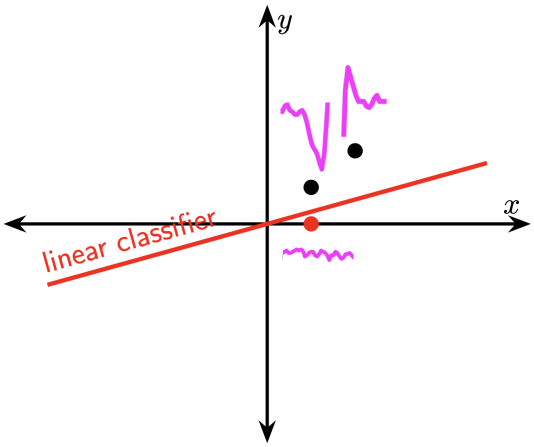
(a) Training objective value



(b) Test predictions



(c) Linear versus network predictions



(d) Lasso shows that neural networks are implicitly local linear classifiers.

Fig. 10: Comparison of a two-layer ReLU network trained with SGD and the convex program on ECG data [40]. The convex method achieves lower training loss (10a) and better test performance (10b) compared to SGD with various learning rates μ. The convex program shows that neural networks are local linear models can therefore adapt to different regions of data better than standard linear models (10c), (10d).

signals the same, whether they are spiking or in a rest phase, the Lasso problem acts as a linear classifier locally, treating different types of signals with different classifiers, as visualized in Figure 10 (d). The equivalence of the Lasso model and neural networks therefore shows that neural networks are implicitly adapting to the data locally. The Lasso formulation gives a globally optimal neural network and hence performs better than the network trained with SGD, which only reaches a local optimum.

Optimization Algorithms

Using specialized convex optimization solvers, such as proximal gradient methods, we can achieve faster convergence and robustness compared to traditional methods [14]. Figure 9b from [14] compares the

proportion of problems solved to 10^{-3} relative training accuracy over 400 UC Irvine datasets. The convex version achieves a global optima and performs better than training with the non-convex problem.

SUMMARY

The training of two-layer ReLU neural networks can be equivalently formulated as a convex optimization problem, specifically within the frameworks of Lasso and Group Lasso from sparse signal processing and compressed sensing. By leveraging geometric algebra, this convex equivalence extends to higher-dimensional data, providing a geometric interpretation of neural network operations. This approach offers theoretical insights and practical advantages, including stability, interpretability, efficient optimization and improved generalization. A limitation of this method is the restriction on network architecture. In deeper models, the number of features increases significantly, making it more difficult to use the convex formulations. Future work includes extending these methods to deeper networks with modern activations and exploring their implications in various signal processing and machine learning applications.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[2] T. Ergen and M. Pilanci, "Convex geometry and duality of over-parameterized neural networks," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 9646–9708, 2021.

[3] M. Pilanci and T. Ergen, "Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for 2-layer networks," in *International Conference on Machine Learning*, 2020.

[4] A. Sahiner, T. Ergen, J. Pauly, and M. Pilanci, "Vector-output relu neural network problems are copositive programs: Convex analysis of two layer networks and polynomial-time algorithms," *International Conference on Learning Representations (ICLR)*, 2021.

[5] A. Sahiner, M. Mardani, B. Ozturkler, M. Pilanci, and J. Pauly, "Convex regularization behind neural reconstruction," *International Conference on Learning Representations (ICLR)*, 2021.

[6] T. Ergen and M. Pilanci, "Implicit convex regularizers of cnn architectures: Convex optimization of 2- and 3-layer networks in polytime," *International Conference on Learning Representations*, 2021.

[7] B. Bartan and M. Pilanci, "Convex relaxations of convolutional neural nets," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 4928–4932.

[8] T. Ergen and M. Pilanci, "Convex geometry of two-layer relu networks: Implicit autoencoding and interpretable models," in *International Conference on Artificial Intelligence and Statistics*, 2020, pp. 4024–4033.

[9] ——, "Revealing the structure of deep neural networks via convex duality," *International Conference on Machine Learning*, 2021.

[10] ——, "Global optimality beyond two layers: Training deep relu networks via convex programs," in *International Conference on Machine Learning*. PMLR, 2021, pp. 2993–3003.

[11] Y. Wang, J. Lacotte, and M. Pilanci, "The hidden convex optimization landscape of two-layer relu neural networks," *International Conference on Learning Representations*, 2022.

[12] T. Ergen, A. Sahiner, B. Ozturkler, J. Pauly, M. Mardani, and M. Pilanci, "Demystifying batch normalization in relu networks: Equivalent convex optimization models and implicit regularization," *International Conference on Learning Representations*, 2021.

[13] B. Bartan and M. Pilanci, "Training quantized neural networks to global optimality via semidefinite programming," *International Conference on Machine Learning*, 2021.

[14] A. Mishkin, A. Sahiner, and M. Pilanci, "Fast convex optimization for two-layer relu networks: Equivalent model classes and cone decompositions," *International Conference on Machine Learning*, 2022.

[15] A. Sahiner, T. Ergen, B. Ozturkler, J. Pauly, M. Mardani, and M. Pilanci, "Unraveling attention via convex duality: Analysis and interpretations of vision transformers," in *International Conference on Machine Learning*, 2022, pp. 19 050–19 088.

[16] T. Ergen, H. I. Gulluk, J. Lacotte, and M. Pilanci, "Globally optimal training of neural networks with threshold activation functions," *International Conference on Learning Representations*, 2023.

[17] A. Mishkin and M. Pilanci, "Optimal sets and solution paths of relu networks," *International Conference on Machine Learning*, 2023.

[18] B. Bartan and M. Pilanci, "Training quantized neural networks to global optimality via semidefinite programming," in *International Conference on Machine Learning*. PMLR, 2021, pp. 694–704.

[19] Y. Wang, Y. Hua, E. Candés, and M. Pilanci, "Overparameterized relu neural networks learn the simplest models: Neural isometry and exact recovery," *accepted to the IEEE Transactions on Information Theory*, 2025.

[20] T. Ergen and M. Pilanci, "Path regularization: A convexity and sparsity inducing regularization for parallel relu networks," *International Conference on Machine Learning (ICML)*, 2021.

[21] Y. Wang, P. Chen, M. Pilanci, and W. Li, "Optimal neural network approximation of wasserstein gradient direction via convex optimization," *SIAM Journal on Mathematics of Data Science*, 2024.

[22] A. Sahiner, T. Ergen, B. Ozturkler, B. Bartan, J. Pauly, M. Mardani, and M. Pilanci, "Hidden convexity of wasserstein gans: Interpretable generative models with closed-form solutions," *International Conference on Learning Representations*, 2021.

[23] B. Bartan and M. Pilanci, "Neural spectrahedra and semidefinite lifts: Global convex optimization of polynomial activation neural networks in fully polynomial-time," *Mathematical Programming*, 2024.

[24] T. Ergen and M. Pilanci, "Convex geometry and duality of over-parameterized neural networks," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 9646–9708, 2021.

[25] L. Luo, "Architectures of neuronal circuits," *Science*, vol. 373, no. 6559, p. eabg7285, 2021. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.abg7285

[26] Y. Bengio, N. Roux, P. Vincent, O. Delalleau, and P. Marcotte, "Convex neural networks," in *Advances in Neural Information Processing Systems*, vol. 18. MIT Press, 2005.

[27] F. Bach, "Breaking the curse of dimensionality with convex neural networks," *JMLR*, vol. 18, no. 1, pp. 629–681, 2017.

[28] C. Fang, Y. Gu, W. Zhang, and T. Zhang, "Convex formulation of overparameterized deep neural networks," *arXiv:1911.07626*, 2019.

[29] L. Yang, J. Zhang, J. Shenouda, D. Papailiopoulos, K. Lee, and R. D. Nowak, "A better way to decay: Proximal gradient training algorithms for neural nets," *OPT2022: 14th Annual Workshop on Optimization for Machine Learning*, vol. abs/2210.03069, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252735239

[30] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on information theory*, vol. 52, no. 2, pp. 489–509, 2006.

[31] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[32] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 58, no. 1, pp. 267–288, 1996.

[33] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

[34] E. J. Candès *et al.*, "Compressive sampling," in *Proceedings of the international congress of mathematicians*, vol. 3. Madrid, Spain, 2006, pp. 1433–1452.

[35] P. T. Boufounos and R. G. Baraniuk, "1-bit compressive sensing," in *2008 42nd Annual Conference on Information Sciences and Systems*, 2008, pp. 16–21.

[36] E. J. Candes and T. Tao, "The power of convex relaxation: Near-optimal matrix completion," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2053–2080, 2010.

[37] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *J. ACM*, vol. 58, no. 3, Jun. 2011. [Online]. Available: https://doi.org/10.1145/1970392.1970395

[38] M. Pilanci, "From complexity to clarity: Analytical expressions of deep neural network weights via clifford algebra and convexity," *Transactions on Machine Learning Research, 2024*.

[39] N. A. Valous, E. Hitzer, S. Vitabile, S. Bernstein, C. Lavor, D. Abbott, M. E. Luna-Elizarrarás, and W. Lopes, "Hypercomplex

signal and image processing: Part 1," *IEEE Signal Processing Magazine*, vol. 41, no. 2, pp. 11–13, 2024.

[40] T. Ergen and M. Pilanci, "The convex landscape of neural networks: Characterizing global optima and stationary points via lasso models," *ArXiv*, vol. abs/2312.12657, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:266374487

[41] A. Mishkin, A. Sahiner, and M. Pilanci, "Fast convex optimization for two-layer relu networks: Equivalent model classes and cone decompositions," *CoRR*, vol. abs/2202.01331, 2022. [Online]. Available: https://arxiv.org/abs/2202.01331

[42] E. Zeger, Y. Wang, A. Mishkin, T. Ergen, E. Candès, and M. Pilanci, "A library of mirrors: Deep neural nets in low dimensions are convex lasso models with reflection features," *arXiv preprint arXiv:2403.01046*, 2024.

[43] Y. Wang, J. Lacotte, and M. Pilanci, "The hidden convex optimization landscape of regularized two-layer relu networks: an exact characterization of optimal solutions," *International Conference on Learning Representations*, 2022.

[44] S. Kim and M. Pilanci, "Convex relaxations of relu neural networks approximate global optima in polynomial time," *ICML*, 2024.

[45] M. Slaney and M. Casey, "Locality-sensitive hashing for finding nearest neighbors [lecture notes]," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 128–131, 2008.

[46] S. Obla, X. Gong, A. Aloufi, P. Hu, and D. Takabi, "Effective activation functions for homomorphic evaluation of deep neural networks," *IEEE Access*, vol. 8, pp. 153 098–153 112, 2020.

[47] T. Ergen and M. Pilanci, "Path regularization: A convexity and sparsity inducing regularization for parallel relu networks," 2023.

[48] T. E. Yifei Wang and M. Pilanci, "Parallel deep neural networks have zero duality gap," *ICLR*, 2023s.

[49] E. Zeger and M. Pilanci, "Black boxes and looking glasses: Multilevel symmetries, reflection planes, and convex optimization in deep networks," *arXiv preprint arXiv:2410.04279*, 2024.

[50] T. Ergen and M. Pilanci, "Implicit convex regularizers of cnn architectures: Convex optimization of 2- and 3-layer networks in polynomial time," *International Conference on Learning Representations*, 2021.

[51] F. Zhang and M. Pilanci, "Analyzing neural network-based generative diffusion models through convex optimization," *arXiv preprint arXiv:2402.01965*, 2024.

[52] G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor, *An Introduction to Statistical Learning with Applications in Python*, ser. Springer Texts in Statistics. Cham: Springer, 2023. [Online]. Available: https://link.springer.com/book/10.1007/978-3-031-38747-0

BIOGRAPHIES

Emi Zeger (*Student Member, IEEE*) received the B.S. degree in mathematics, the B.S. degree in electrical engineering (EE), and the M.S. degree in EE from UCLA, CA, USA, in 2021. She is pursuing the Ph.D. degree in EE at Stanford University, CA. Her research interests are in optimization, neural networks and sustainable computing. She is the recipient of a best paper award at the IEEE International Conference on Communications (ICC) in 2024.

Mert Pilanci (*Member, IEEE*) is an Assistant Professor in the Department of Electrical Engineering at Stanford University. He received his Ph.D. in Electrical Engineering and Computer Sciences from the University of California, Berkeley in 2016. In 2017, he was a postdoctoral fellow working with Emmanuel Candès at Stanford University. His research interests include optimization, machine learning, and signal processing, with a focus on developing theory and efficient algorithms for neural networks. He is a recipient of the NSF CAREER Award and the Army Research Office Early Career Award. He is a Member of the IEEE and a Member of SIAM.