A Library of Mirrors: Deep Neural Nets in Low Dimensions are Convex Lasso Models with Reflection Features

Emi Zeger*, Yifei Wang*, Aaron Mishkin†, Tolga Ergen*
Emmanuel Candès‡, and Mert Pilanci*
Stanford University

Key words. deep neural networks, convex optimization, LASSO, sparsity

MSC codes. 62M45, 46N10

Abstract. We prove that training certain classes of neural networks on 1-D data is equivalent to solving convex Lasso problems with discrete, explicitly defined dictionary matrices. We consider neural networks with piecewise linear activations and depths ranging from 2 to an arbitrary but finite number of layers. We first show that two-layer networks with piecewise linear activations are equivalent to Lasso models using a discrete dictionary of ramp functions, with breakpoints corresponding to the training data points. In certain general architectures with absolute value or ReLU activations, a third layer surprisingly creates features that reflect the training data about themselves. Additional layers in a *deep narrow network* progressively generate reflections of these reflections. The Lasso representation provides valuable insights into the analysis of globally optimal networks, elucidating their solution landscapes and enabling closed-form solutions in certain minimal regularization cases. Numerical results show that reflections also occur when optimizing standard deep networks using standard non-convex optimizers. Additionally, we demonstrate our theory with autoregressive time series models.

1. Introduction.

Training deep neural networks is an important optimization problem. However, the non-convexity of neural nets makes their training challenging. In this paper, we show that for low-dimensional data, e.g., 1-D or 2-D, training a deep neural network can be simplified to solving a convex Lasso problem with an easily constructed dictionary matrix.

Neural networks are used as predictive models for low-dimensional data in applications such as processing time-series data, especially acoustic signals, as well as in "coordinate-based" multi-layer perceptrons (MLP)s (23). They can also be used to predict time series financial data (Section 4). In acoustic signal processing, raw audio data from a single microphone represents a 1-D time series (34; 36) (although it can also be represented as higher-dimensional spectrograms). Coordinate-based MLPs are a class of models used in computer vision and graphics that input low-dimensional positional data, typically 3-D coordinates representing a single location, and output representations of properties such as shape or color at that location (18; 37; 23). Low-dimensional techniques used in coordinate-based MLPs such as Fourier representation features (38) are also useful in physics-informed machine learning problems (23). Neural networks are also used on data assumed to have a low-dimensional structure such as using Bayesian inference to learn low-dimensional representations (9) and uncertainty quantification (42; 43). There has been recent study on the problem of two-layer ReLU neural

*Department of Electrical Engineering
†Department of Computer Science
‡Departments of Statistics and Mathematics

networks on 1-D data (21; 22; 26). It has been proved that an optimal two-layer ReLU neural network precisely interpolates 1-D training data as a piecewise linear function for which the breakpoints are at the data points (35; 13; 14).

However, even for low-dimensional data, the current literature still lacks analysis on the expressive power and learning capabilities of deeper neural networks with generic activations. This motivates us to study the optimization of 2 and 3-layer networks with piecewise linear activations and deeper neural networks with sign and ReLU activations. For 1-D data, we simplify the training problem by recasting it as a convex Lasso problem, which has been extensively studied (10; 39; 40).

Convex analysis of neural networks was developed in several prior works. As an example, infinite-width neural networks enable the convexification of the overall model (3; 4; 16). However, due to the infinite-width assumption, these results do not reflect finite-width neural networks in practice. Recently, a series of papers (12; 13; 33) developed a convex analytic framework for the training problem of two-layer neural networks with ReLU activation. As a follow-up work, a similar approach is used to formulate the training problem for threshold activations with data in general d-dimensions as a Lasso problem (11). However, the dictionary matrix is described implicitly and requires high computational complexity to create (11). By focusing on 1-D data, we can provide simple, explicit Lasso dictionaries and consider additional activations. For example, we analyze networks with sign activation, which is useful in contexts such as saving memory to meet hardware constraints (7; 24).

Throughout the paper, all scalar functions extend to vector and matrix inputs component-wise. We write vectors as $\mathbf{v} = (v_1, \cdots, v_n)$ and denote the set of n-dimensional, real-valued column and row vectors by $\mathbb{R}^n$ and $\mathbb{R}^{1\times n}$, respectively. For $L \geq 2$, an L-layer *neural network* for d-dimensional data is denoted by $f_L(\mathbf{x}; \theta) : \mathbb{R}^{1\times d} \to \mathbb{R}$, where $\mathbf{x} \in \mathbb{R}^{1\times d}$ is an input row vector and θ is the *parameter set*. The set θ may contain matrices, vectors, and scalars representing weights and biases. We let $\theta \in \Theta$, where Θ is the *parameter space*. Let $\mathbf{X} \in \mathbb{R}^{N\times d}$ be a *data matrix* consisting of N *training samples* $\mathbf{x}_1, \cdots \mathbf{x}_N \in \mathbb{R}^{1\times d}$. We consider regression tasks and call $\mathbf{y} \in \mathbb{R}^N$ the *target vector*. The (non-convex) neural net *training problem* is

$$\min_{\theta\in\Theta} \frac{1}{2}\|f_L(\mathbf{X}; \theta) - \mathbf{y}\|_2^2 + \frac{\beta}{\tilde{L}}\|\theta_w\|^{\tilde{L}} \tag{1.1}$$

where $\beta > 0$ is a regularization coefficient for a subset of parameters $\theta_w \subset \theta$ that incur a weight penalty when training. We denote $\|\theta_w\|^{\tilde{L}} = \sum_{q\in\theta_w} \|q\|_2^{\tilde{L}}$, which penalizes the total network weight. The results can be generalized to other p-norm regularizations as well. $\tilde{L}$ is the *effective regularized depth*, defined to be the usual depth L for ReLU, leaky ReLU, and absolute value activations, but defined to be just 2 for threshold and sign activations. This is motivated by the property that neurons with sign or threshold activations are invariant to the magnitude of their neuron weights, so it does not make sense to penalize the inner weights (Remark D.1). A central element of this paper is the *Lasso problem*

$$\min_{\mathbf{z},\xi} \frac{1}{2}\|\mathbf{A}\mathbf{z} + \xi\mathbf{1} - \mathbf{y}\|_2^2 + \tilde{\beta}\|\mathbf{z}\|_1 \tag{1.2}$$

where $\mathbf{z}$ is a vector, $\xi \in \mathbb{R}$, $\mathbf{1}$ is a vector of ones, and $\tilde{\beta} > 0$ (whose relation to β in (1.1) is

defined in Section 3). $\mathbf{A}$ is called the *dictionary matrix*, with columns $\mathbf{A}_i \in \mathbb{R}^N$.

A neural net is *trained* by searching for θ that optimizes (1.1). A neural net $f_L(\mathbf{x};\theta)$ is called *optimal* if θ is a global minimizer of (1.1). Unfortunately, training is complicated by the non-convexity of the optimization problem. However, for data of dimension d=1, we reformulate the training problem (1.1) into the equivalent but simpler Lasso problem (1.2), where $\mathbf{A}$ is a fixed matrix that is constructed based on the training data $\mathbf{X}$ and neural net architecture. Moreover, the dictionary matrix columns $\mathbf{A}_i$ correspond to piecewise linear functions $f_i : \mathbb{R} \to \mathbb{R}$ we call *features* that satisfy the following:

1. For all $n = 1, \cdots, N$, $f_i(x_n) = \mathbf{A}_{n,i}$.
2. If $\mathbf{z}, \xi$ are Lasso solutions, then $f(x) = \sum_{i=1}^{n} z_i f_i(x) + \xi$ is equal to an optimal network.

The set of features for a given network is a *dictionary*. We call a collection of dictionaries a *library* when referring to the features from multiple depths. We explicitly provide the elements of $\mathbf{A}$, making it straightforward to build and solve the convex Lasso problem instead of solving the non-convex training problem. This reformulation allows for exploiting fast Lasso solvers based on the proximal gradient method and Least Angle Regression (LARS) (10), (31). The Lasso representation also elucidates the solution path of neural networks. The *solution path* for the Lasso or training problem is the map from $\beta \in (0, \infty)$ to the solution set. The Lasso solution path is well understood (39; 40; 10; 31), providing insight into the solution path of the ReLU training problem (28).

Whereas in the training problem (1.1), the quality of the neural net fit to the data is measured by the l_2 loss as $\frac{1}{2}||f_L(\mathbf{X};\theta) - \mathbf{y}||_2^2$, our results generalize to a wide class of convex loss functions $\mathcal{L}_{\mathbf{y}} : \mathbb{R}^N \to \mathbb{R}$. With a general loss function, (1.1) becomes $\min_{\theta\in\Theta} \mathcal{L}_{\mathbf{y}}(f_L(\mathbf{X};\theta)) + \frac{\beta}{\tilde{L}}\|\theta_w\|^{\tilde{L}}$. This is shown to be equivalent to the generalization of (1.2), namely $\min_{\mathbf{z},\xi} \mathcal{L}_{\mathbf{y}}(\mathbf{A}\mathbf{z} + \xi\mathbf{1} - \mathbf{y}) + \tilde{\beta}\|\mathbf{z}\|_1$. The Lasso problem selects solutions $\mathbf{z}$ that generalize well by penalizing their total weight in l_1 norm (39). The l_1 norm typically selects a small number of elements in $\mathbf{z}$ to be nonzero. The Lasso equivalence demonstrates that neural networks can learn a sparse representation of the data by selecting certain features to fit $\mathbf{y}$.

This paper is organized as follows. Section 2 defines various neural network architectures. Section 3 describes our main theoretical result: neural networks are solutions to Lasso problems. One important consequence of the Lasso equivalence is that deep networks with ReLU and absolute value activations learn geometric reflections in the data, where the *reflection* of scalars a about b is $R_{(a,b)} = 2b - a$. The next sections describe applications of the Lasso equivalence. Section 4 uses the Lasso formulation to improve training of neural networks that predict financial time-series. Appendix E examines the relationship between the entire set of optimal neural nets given by the training problem versus the Lasso problem, while Appendix C applies the Lasso model to examine neural net behavior under minimum regularization, finding closed-form solutions. Appendix D presents experiments that support our theoretical results, and shows examples where neural networks trained with Adam naturally exhibit Lasso features.

1.1. Contributions.

Our contributions can be summarized as follows:

- Training neural networks with symmetrized or deep narrow architectures on 1-D data is equivalent to solving Lasso problems with finite, explicit and fixed dictionaries of basis signals that grow richer with depth (Theorems 3.11, 3.6). We identify dictionaries for

various architectures in closed form (Lemma B.2).

- Features with reflections of training data appear in libraries for certain 3-layer architectures with ReLU (Theorem 3.5) or deep narrow networks with absolute value activation (Theorem 3.2). Experimentally, training these networks using the Adam optimizer leads to the same reflection features that we prove and matches our theoretical results on the global optima (Appendix D).
- After depth 3, the libraries freeze for deep narrow networks with ReLU activation that have the same number of biased neurons in the middle and final layers (Lemma 3.7). But, the libraries grow when the width expands to twice as many neurons in the middle layer as the final layer for networks with ReLU activation, enabling greater representation power (Theorem 3.11).

1.2. Notation. Assume 1-D training data is ordered as $x_1 > x_2 > \cdots > x_N$. The indicator function of a logical statement z is $\mathbf{1}\{z\}$. Let $[n] = \{1, 2, \ldots, n\}$. The number of nonzero elements in a vector $\mathbf{z}$ is $||\mathbf{z}||_0$. Let $\mathbf{1}, \mathbf{0} \in \mathbb{R}^N$ be the all-ones and all-zeros vectors, respectively. A network that uses ReLU activation is a "ReLU network" or "ReLU-activated network," and a feature in a Lasso problem that is equivalent to a ReLU network is a "ReLU feature." Similar terminology holds for other activations and architectures.

2. Neural net architectures. This section is devoted to defining neural net terminology and notation to be used throughout the rest of the paper. Let $L \geq 2$ be the depth of a neural network (which has $L-1$ hidden layers). We assume the activation $\sigma : \mathbb{R} \to \mathbb{R}$ is *piecewise linear around* 0, i.e., of the form

$$\sigma(x) = \begin{cases} a^- x + b^- & \text{if } x < 0 \\ a^+ x + b^+ & \text{otherwise} \end{cases} \tag{2.1}$$

for some $a^-, a^+, b^-, b^+ \in \mathbb{R}$. As shorthand, "piecewise linear" will mean "piecewise linear around 0." We focus on the piecewise linear activations of ReLU, leaky ReLU, absolute value, sign, and threshold functions. The ReLU activation is $\sigma(x) = (x)_+ := \max\{x, 0\}$, and absolute value activation is $\sigma(x) = |x|$. The leaky ReLU generalizes ReLU and absolute value as $\sigma(x) = (a^+\mathbf{1}\{x > 0\} + a^-\mathbf{1}\{x < 0\})x$ where $a^+ \neq a^-$. ReLU, leaky ReLU and absolute value activations will be referred to as "continuous piecewise linear." The threshold activation is $\sigma(x) = \mathbf{1}\{x \geq 0\}$, and the sign activation is $\sigma(x) = \text{sign}(x)$, where $\text{sign}(x)$ is -1 if $x < 0$, and 1 if $x \geq 0$. Note $\text{sign}(0) = 1$.

For $\mathbf{Z} \in \mathbb{R}^{n \times m}, \mathbf{s} \in \mathbb{R}^m$, let $\sigma_{\mathbf{s}}(\mathbf{Z}) = \sigma(\mathbf{Z})\text{Diag}(\mathbf{s})$. When the columns of $\sigma(\mathbf{Z})$ are neuron outputs, the i^{th} column of $\sigma_{\mathbf{s}}(\mathbf{Z}) \in \mathbb{R}^{N \times m}$ represents a neuron scaled by an *amplitude parameter* $s_i \in \mathbb{R}$. Amplitude parameters are (trainable) parameters for only the sign and threshold activations, and are to be ignored by interpreting them as 1 for ReLU, leaky ReLU, and absolute value activations.

Next, we define some neural net architectures. The parameter set is partitioned into $\theta = \theta_w \cup \theta_b \cup \{\xi\}$, where θ_b is a set of *internal bias* terms, and ξ is an *external* bias term. We define the elements of each parameter set below. We define neural nets by their output on row vectors $\mathbf{x} \in \mathbb{R}^{1 \times d}$. Their outputs then extend to matrix inputs $\mathbf{X} \in \mathbb{R}^{N \times d}$ row-wise.

2.1. Standard networks. The following is a commonly studied neural net architecture. Let $L \geq 2$, the number of layers. Let $m_1 = d$ and $m_l \in \mathbb{N}$ for $l \in [L] - \{1\}$ which are the number of neurons in each layer. For $l \in [L-1]$, let $\mathbf{W}^{(l)} \in \mathbb{R}^{m_l \times m_{l+1}}, \mathbf{s}^{(l)} \in \mathbb{R}^{m_{l+1}}, \mathbf{b}^{(l)} \in \mathbb{R}^{1 \times m_{l+1}}, \xi \in \mathbb{R}$, which denote weights, amplitude parameters, internal biases, and external bias, respectively. Let $\mathbf{X}^{(1)} = \mathbf{x} \in \mathbb{R}^{1 \times d}$ be the input to the neural net and $\mathbf{X}^{(l+1)} \in \mathbb{R}^{1 \times m_{l+1}}$ be viewed as the inputs to layer $l+1$, defined by

$$\mathbf{X}^{(l+1)} = \sigma_{\mathbf{s}^{(l)}}\left(\mathbf{X}^{(l)}\mathbf{W}^{(l)} + \mathbf{b}^{(l)}\right). \tag{2.2}$$

Let $\boldsymbol{\alpha} \in \mathbb{R}^{m_L}$, which is the vector of final layer coefficients. A *standard neural network* is $f_L(\mathbf{x};\theta) = \xi + \mathbf{X}^{(L)}\boldsymbol{\alpha}$. The regularized and bias parameter sets are $\theta_w = \left\{\boldsymbol{\alpha}, \mathbf{W}^{(l)}, \mathbf{s}^{(l)} : l \in [L-1]\right\}$ and $\theta_b = \left\{\mathbf{b}^{(l)} : l \in [L-1]\right\}$, respectively.

Analyzing the training problem for standard networks has traditionally been challenging due to provably non-zero duality gaps (46). However, parallel architectures can be examined through the lens of convex analysis, and they can also be converted into standard architectures, which significantly streamlines the analytical process. Furthermore, employing parallel branches of neural networks has proven especially effective in enhancing modern mixture-of-experts structures. By changing the architecture to a parallel structure defined next, we show that the training problem simplifies to the Lasso problem. These alternative architectures allow neural nets to be reconstructed more tractably from a Lasso solution than with a standard network.

2.2. Parallel networks. A parallel network is a linear combination of certain standard networks in parallel, as we now define. Each standard network is called a *parallel unit*. Let $L \geq 2, m_0 = d, m_{L-1} = 1$ and $m_l \in \mathbb{N}$ for $l \in [L] - \{L-1\}$. For $i \in [m_L], l \in [L-1]$, let $\mathbf{W}^{(i,l)} \in \mathbb{R}^{m_{l-1} \times m_l}$, $\mathbf{s}^{(i,l)} \in \mathbb{R}^{m_l}$, $\mathbf{b}^{(i,l)} \in \mathbb{R}^{1 \times m_l}, \xi \in \mathbb{R}$, which are the weights, amplitude parameters, and biases of the i^{th} parallel unit.[1] Let $\hat{\mathbf{X}}^{(i,1)} = \mathbf{x} \in \mathbb{R}^{1 \times d}$ be the input to the neural net and $\hat{\mathbf{X}}^{(i,l+1)} \in \mathbb{R}^{1 \times m_l}$ be viewed as the input to layer $l+1$ in unit i, defined by

$$\hat{\mathbf{X}}^{(i,l+1)} = \sigma_{\mathbf{s}^{(i,l)}}\left(\hat{\mathbf{X}}^{(i,l)}\mathbf{W}^{(i,l)} + \mathbf{b}^{(i,l)}\right). \tag{2.3}$$

Let $\boldsymbol{\alpha} \in \mathbb{R}^{m_L}$. A *parallel neural network* is $f_L(\mathbf{x};\theta) = \xi + \sum_{i=1}^{m_L} \hat{\mathbf{X}}^{(i,L)}\alpha_i$. In the parallel architecture, m_L is the number of neurons in the final layer and for $i \in [m_L]$, we define the disjoint unions $\theta_w = \bigcup_{i \in [m_L]} \theta_w^{(i)}$ and $\theta_b = \bigcup_{i \in [m_L]} \theta_b^{(i)}$. The regularized and bias parameter sets are $\theta_w^{(i)} = \left\{\alpha_i, \mathbf{s}^{(i,l)}, \mathbf{W}^{(i,l)} : l \in [L-1]\right\}, \theta_b^{(i)} = \left\{\mathbf{b}^{(i,l)} : l \in [L-1]\right\}$, for $i \in [m_L]$. We view parallel units as functions $\hat{\mathbf{X}}^{(i,L)} : \mathbb{R}^{1 \times d} \to \mathbb{R}$ and with abuse of notation write $\hat{\mathbf{X}}^{(i,L)}(\mathbf{x}) \in \mathbb{R}$ as the output of $\hat{\mathbf{X}}^{(i,L)}$ evaluated on a sample $\mathbf{x}$. For a training dataset $\mathbf{X} \in \mathbb{R}^{N \times d}$, we denote the evaluations of the functions $\hat{\mathbf{X}}^{(i,L)}$ on the training data as $\hat{\mathbf{X}}^{(i,L)}(\mathbf{X})$. The condition that $m_{L-1} = 1$ is required for parallel networks to output a scalar. Unlike parallel networks,

[1] While we define the first dimension of $\mathbf{W}^{(1)}$ to be $m_1 = d$ for standard networks, it is $m_0 = d$ for parallel networks. This offset is due to the requirement that the second dimension of $\mathbf{W}^{(i,L-1)}$ must be 1 rather than m_L as for standard networks.

standard networks do not require $m_{L-1} = 1$. Each parallel unit of a parallel network (with $m_L \geq 1$) is a special case of a standard network (with $m_L = 1$). Therefore a parallel network with just one parallel unit is a special case of a standard network (with $m_L = 1$.) This paper primarily focuses on parallel architectures. However, a parallel network can be converted into a standard network, and a 2-layer network is both a standard and parallel network, as shown in Appendix A.

3. Main results. In this section, we show that for 1-D data and certain architectures, non-convex deep neural net training problems are *equivalent* to Lasso problems, that is, their optimal values are the same, and given a Lasso solution, we can reconstruct a neural net that is optimal in the training problem. We say that a neural network *model* is equivalent to a Lasso model to mean that the optimal models are convertible to each other.

Unless otherwise stated, for the rest of this paper assume the data is 1-D. Define $\tilde{\beta} = \frac{\beta}{2}$ in (1.1),(1.2) for 3-layer symmetrized networks (defined below). For all other networks, let $\tilde{\beta} = \beta$.

We focus on 2-layer networks with piecewise linear activations and deep networks with continuous piecewise linear activations. A *deep narrow network* is a parallel network where the number of neurons in each parallel path is equal to 1, i.e., $m_1{=}\cdots{=}m_{L-1}{=}1$. For a 2-layer network, the parallel and deep narrow networks are the same as the standard network, as shown in Appendix A.2. A deep narrow network has m_L parallel units. Each l^{th} parallel unit has $L-1$ non-linear hidden layers: $l = 1, \cdots, L-1$. Each l^{th} hidden layer has a single neuron with a weight $\mathbf{W}^{(i,l)} \in \mathbb{R}$ and bias $\mathbf{b}^{(i,l)} \in \mathbb{R}$. Therefore the total number of neurons is m_L units $\times\, (L-1)$ layers per unit $\times\, (1)$ neuron per layer $= m_L(L-1)$. For example, a 2-layer network has m_L neurons.

A *symmetrized 3-layer network* is a parallel network with continuous piecewise linear activation where $m_1 = 2$ (which means each i^{th} unit out of m_3 parallel units has 2-D weight vectors $\mathbf{W}^{(i,1)}$ and $\mathbf{W}^{(i,2)}$ for the first and second layer, respectively) and the parameter space Θ enforces the constraint $\left|\mathbf{W}_1^{(i,l)}\right| = \left|\mathbf{W}_2^{(i,l)}\right|$ for $l \in [2], i \in [m_3]$ where $\mathbf{W}_j^{(i,l)}$ is the j^{th} element of $\mathbf{W}^{(i,l)}$, which is a column/row vector. Therefore a symmetrized 3-layer network has $2m_3$ neurons in the "middle" layer. A symmetrized network extends the expressibility of a deep narrow network by expanding its width. For other architectures, see the supplementary material (47).

In this section, we focus on the architectures discussed above to derive explicit, simple features that provide intuition towards understanding the representation power of neural networks. We reformulate the training problem for 2-layer networks and for deeper depths with absolute value, ReLU and leaky ReLU activations into a convex Lasso problem. In general, our convexification approach and proofs provide a framework to analyze neural networks with more arbitrary widths. We begin for with convex formulations for networks with absolute value activation, as their convex models are relatively simple. Then we extend these formulations for the traditional ReLU activation. Finally, we generalize to other various piecewise linear activations that have more complex convex formulations by formally defining a library of features. Proofs are deferred to Appendix D.3.

3.1. Absolute value activation. We first discuss networks with absolute value activation, as the symmetry of $|x|$ significantly simplifies the features. However, absolute value activation

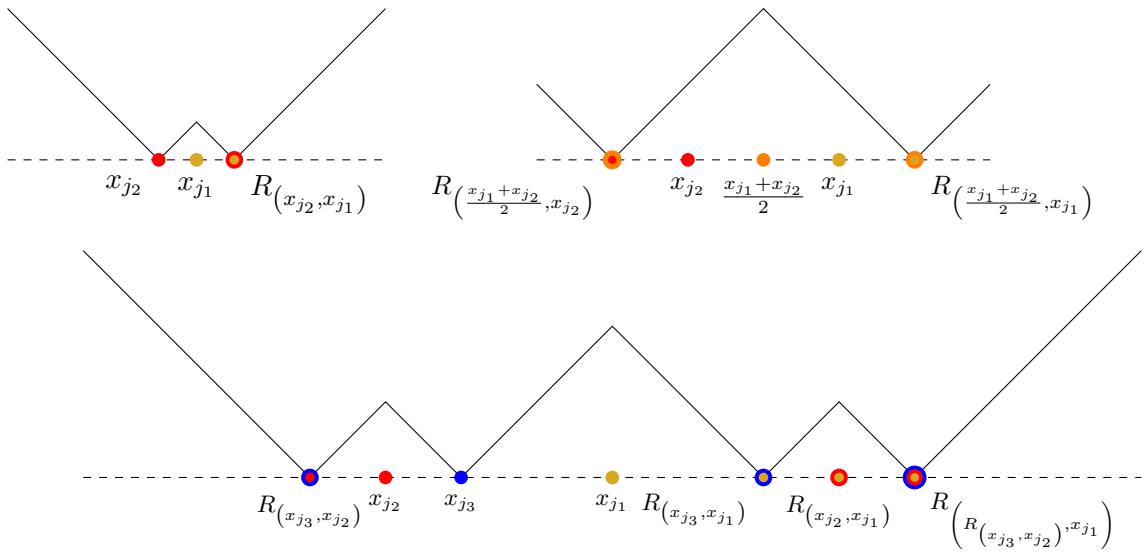


Figure 3.1: Example features are drawn as defined in Definition 3.10, not including reversed directions, for deep narrow networks with absolute value activation. The features drawn are $\hat{\mathbf{X}}^{(3)}(x)=\big||x-x_{j_1}|-|x_{j_2}-x_{j_1}|\big|$ where $x_{j_1}=3, x_{j_2}=2$ (top left) and $x_{j_1}=8, x_{j_2}=4$ (top right) for 3 layers and $\hat{\mathbf{X}}^{(3)}(x)=\Big|\big||x-x_{j_1}|-|x_{j_2}-x_{j_1}|\big|-\big||x_{j_3}-x_{j_1}|-|x_{j_2}-x_{j_1}|\big|\Big|$ where $x_{j_1}=7, x_{j_2}=4, x_{j_3}=5$, for 4 layers. Top row: 3-layer features. The top left feature contains a breakpoint at the reflection of x_{j_2} (red) across x_{j_1} (gold), which is denoted as $R_{(x_{j_2},x_{j_1})}$ (red encircling gold). Other breakpoints are colored similarly. The top right feature has breakpoints at averages of training data and their reflections about other training data. Bottom row: an example of a 4-layer feature, which contains a double reflection of x_{j_3} (blue) reflected across x_{j_2} (red), then reflected across x_{j_1} (gold), which is denoted as $R_{\left(R_{(x_{j_3},x_{j_2})},x_{j_1}\right)}$ (blue encircling red, encircling gold). All lines have slopes ± 1, and $x_{j_1}, x_{j_2}, x_{j_3}$ are training data.

models for two-layer networks are equivalent to ReLU activation models, as long as a skip connection is present. A 2-layer network with a *skip connection* is $f_L^{\text{skip}}(x;\theta) = f_L(x;\theta) + \omega x$ if $x \in \mathbb{R}$ (or more generally, $f_L^{\text{skip}}(\mathbf{x};\theta) = f_L(\mathbf{x};\theta) + \mathbf{x}\omega$) where $\omega \in \mathbb{R}^d$ is a trainable parameter in θ.

Lemma 3.1. *The training problem for a 2-layer network with skip connection and ReLU activation remains equivalent if the activation is changed to absolute value, and there is a map between the solutions for either activation.*

By Lemma 3.1, the absolute value activation is of interest to analyze as it can map to ReLU when skip connections are incorporated. Next we define some terms used to state our results.

For a piecewise linear function $f : \mathbb{R} \to \mathbb{R}$, x *is a breakpoint* of f if f changes slope or is discontinuous at x. A breakpoint is a "kink" in the graph of f. For $a, b, c \in \mathbb{R}$, *double reflection* is a reflection of a reflection, i.e., is of the form $R_{(R_{(a,c)},b)}$ or $R_{(b,R_{(a,c)})}$. The *generalized*

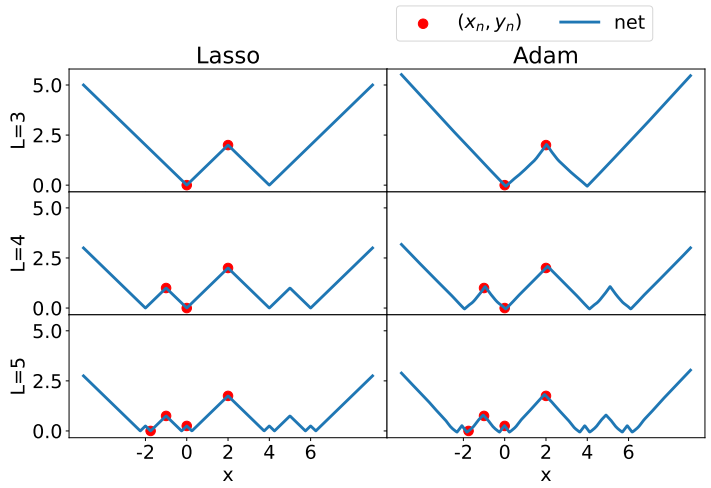


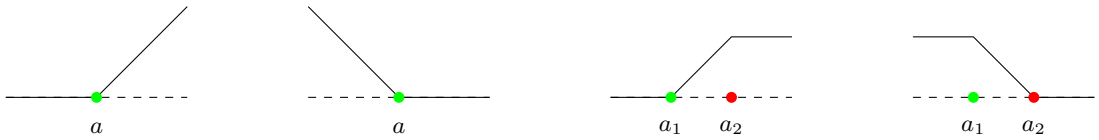
Figure 3.2: Lasso and Adam-trained deep narrow networks with absolute value activation. For L=3, the breakpoint at 4 is not a training point; it is the reflection of x_2=0 across x_1=2. For L=4, the breakpoint at 6 is not a training point; it is x_2=0 reflected about x_3=−1 to −2 (which not a training point) and then reflected across x_1=2. Similarly the 5-layer network contains more complex reflections.

reflection of a and c about b is $a + c - b = R_{(b,\frac{a+c}{2})}$, the reflection of b about the average of a and c. When $a = c$, the generalized reflection of a and c about b is the reflection of a about b. A breakpoint that is of the form $R_{(x_{j_1},x_{j_2})}$ for training data x_{j_1}, x_{j_2} is called a *reflection breakpoint* and a feature with a reflection breakpoint is called a *reflection feature*; and similarly for double reflections. Networks with absolute value activation can be modeled as Lasso problems with reflection features, as stated next.

Theorem 3.2 (Lasso equivalent of deep absolute value networks). *A deep narrow network of arbitrary depth with $\sigma(x) = |x|$ is equivalent[2] to a Lasso model with a finite set of features. Its dictionary matrix for 2 layers is $\mathbf{A}_{i,j} = |x_i - x_j|$. For 3 and 4 layers, its library includes features whose i^{th} element is $\big||x_i - x_{j_1}| - |x_{j_2} - x_{j_1}|\big|$ and $\Big|\big||x_i - x_{j_1}| - |x_{j_2} - x_{j_1}|\big| - \big||x_{j_3} - x_{j_1}| - |x_{j_2} - x_{j_1}|\big|\Big|$, respectively, over all training samples $x_i, x_{j_1}, x_{j_2}, x_{j_3}$. A similar pattern holds for deeper networks. The 2-layer features have breakpoints exactly at training data. The libraries for 3 and 4 layers additionally include reflection and double reflection features, respectively.*

Theorem 3.2 implies that an absolute value network learns to model data with a discrete and fixed dictionary of features. Figure 3.1 plots these features for L=3 and L=4. It illustrates that the breakpoints of the L=3 features occur at training data reflections. The bottom row plots a possible feature for 4 layers, which shows breakpoints at reflections and double reflections. Even with just one neuron per layer (per path for parallel networks), adding a third layer in neural networks with absolute value activation adds features to the dictionary, and adds new

[2]See Section 3 for the definition of model equivalence.



$\mathrm{ReLU}_a^+(x)$ $\mathrm{ReLU}_a^-(x)$ $\mathrm{Ramp}_{a_1,a_2}^+(x)$ $\mathrm{Ramp}_{a_1,a_2}^-(x)$

Figure 3.3: Examples of capped ramp functions in Definition 3.3. If $a, a_1, a_2 \in \{x_1, \cdots, x_N\}$, these functions are Lasso features for 3-layer deep narrow networks with ReLU activation.

locations of breakpoint to those features, namely at reflections $R_{(x_i,x_j)}$ of data points about themselves. Adding a fourth layer creates double reflections. In Figure 3.2, standard 3, 4, and 5-layer networks are trained with Adam. We make $\beta = 10^{-7}$ close to zero, and also solve the Lasso problem as $\beta \to 0$ (Appendix C). The Lasso solutions are used to initialize a small subset of the neurons in the non-convex training. The Adam-trained networks closely match the Lasso solutions. Moreover, the 5-layer networks suggest that features continue to gain more complex reflections with depth. Simulation details are given in Appendix D.

Our approach lays a foundation which future work can use to enumerate complete feature libraries for $L \geq 4$ layers. Reflection features allow neural networks to fit functions with breakpoints at locations in between data points. The reflection breakpoints for absolute value networks suggest that they can learn geometric structures or symmetries from the data. Moreover, as the depth increases, this dictionary expands, which deepens its representation power.

3.2. ReLU activation. With intuition from absolute value networks, we next discuss ReLU networks. Since reflection features appear in 3-layer networks with absolute value activation and $|x| = \frac{(x)_+ + (-x)_+}{2}$, we might expect reflections to also appear in 3-layer ReLU networks with twice as many neurons in the middle layer as the absolute value network. This is indeed the case, as shown below. First, we define parameterized families of scalar input, real-valued functions that will be used to describe features for ReLU networks.

Definition 3.3. *Let* $a_1 \in [-\infty, \infty), a_2 \in (-\infty, \infty]$. *The* capped ramp *functions are*

$$Ramp_{a_1,a_2}^+(x) = \begin{cases} 0 & \text{if } x \leq a_1 \\ x - a_1 & \text{if } a_1 \leq x \leq a_2 \\ a_2 - a_1 & \text{if } x \geq a_2 \end{cases}, \quad Ramp_{a_1,a_2}^-(x) = \begin{cases} a_2 - a_1 & \text{if } x \leq a_1 \\ a_2 - x & \text{if } a_1 \leq x \leq a_2 \\ 0 & \text{if } x \geq a_2 \end{cases}$$

provided that $a_1 \leq a_2$, *and otherwise* $Ramp_{a_1,a_2}^+ = Ramp_{a_1,a_2}^- = 0$. *In particular, the* ramp *functions are* $ReLU_a^+(x) = Ramp_{a,\infty}^+ = (x-a)_+$ *and* $ReLU_a^-(x) = Ramp_{-\infty,a}^- = (a-x)_+$.

In Definition 3.3, the parameters a, a_1, a_2 are the breakpoints of ramp and capped ramp functions. This is illustrated in Figure 3.3. Using these features, we state our results on Lasso equivalence for ReLU networks.

Theorem 3.4 (deep narrow ReLU network representation capability stagnates). *A deep narrow network of arbitrary depth with ReLU activation is equivalent to a Lasso model with a finite set of features. Its library contains only ramps and capped ramps, and beyond* 3 *layers, ReLU features do not change as the network deepens.*

In contrast to absolute value activation, for deep narrow networks, the ReLU library never gains reflection features. However, a symmetrized ReLU architecture creates reflections.

Theorem 3.5 (wider ReLU networks do not stagnate and generate reflections). *A three-layer symmetrized network with ReLU activation is equivalent to a Lasso model with a finite set of features, including those with breakpoints at reflections.*

Figure 3.4 plots ReLU features. A 2-layer ReLU network learns ramp features. Extending the depth to one more layer without changing the width creates a deep narrow 3-layer network, which additionally learns capped ramp features. Extending both the depth and width creates a 3-layer symmetrized network, which learns an additional host of features, including generalized reflection features shown in the bottom row of the figure. The generalized reflections $x_j+x_k-x_i$ are reflections when $j=k$. Reflection features appear in Adam-trained ReLU networks as predicted by the Lasso formulation, as shown in Figure 3.5. Details of this simulation are given in Appendix D.

In addition to ReLU and absolute value networks, leaky ReLU networks are also equivalent to Lasso models. Moreover, we can upper bound the size of the libraries.

Theorem 3.6. *A deep narrow network of any depth $L \geq 2$ and piecewise linear activation is equivalent to a Lasso problem with a finite set of features. The number of features is $O(N^2)$ for ReLU activation and $O(N^{L-1}2^L L!)$ for leaky ReLU and absolute value activations.*

In particular, the number of features is finite and at most polynomial in the number of training points and exponential in the depth. However, the number of features is an overestimate and in fact saturates for certain activations and architectures, as seen in the next result.

Lemma 3.7. *Training a deep narrow ReLU network with an arbitrary number of layers ($L \geq 2$) is a Lasso problem where features are ReLU or capped ramp functions with breakpoints at data points. The number of features is $O(N^2)$.*

Lemma 3.7 is a consequence of Theorem 3.6 and Theorem 3.4 but is stated to emphasize the property that the number of features is capped for deep narrow ReLU networks. Having only one neuron per layer limits the expressibility of ReLU networks. In contrast to absolute value, ReLU networks rely more heavily on wider layers for expressibility.

3.3. General piecewise linear activations.

In the next definition, recall a^+, a^- (2.1) are the positive and negative slopes for an activation σ.

Definition 3.8. *For $\alpha, \beta, a^+, a^- \in \mathbb{R}$ such that $a^+ \neq a^-$, let the* normalized midpoint *be*

$$m_{\alpha,\beta} = -\frac{a^+ \max\{\alpha,\beta\} - a^- \min\{\alpha,\beta\}}{a^+ - a^-}. \tag{3.1}$$

Note $m_{\alpha,\beta}$ is the midpoint $\frac{\alpha+\beta}{2}$ between α and β if $\sigma(x)=|x|$. If $\sigma(x)=(x)_+$, or if $\sigma(x)=|x|$ and $\alpha=\beta$, then $m_{\alpha,\beta}=\alpha$. We use the normalized midpoint to define bias parameters for features.

depth L	activation σ	features	reflections?	ξ=0?	number of features	results
2	ReLU, leaky ReLU	$\sigma(x-x_n)$, $\sigma(x_n-x)$	no	yes	$2N$	Theorem 3.11 Corollary 3.15
2	absolute value	$\sigma(x-x_n)$	no	yes	N	Theorem 3.11 Corollary 3.15
2	threshold	$\sigma(x-x_n)$, $\sigma(x_n-x)$	no	no	$2N$	Theorem 3.11 Corollary 3.15
2	sign	$\sigma(x-x_n)$	no	no	N	Theorem 3.11 Corollary 3.15
3+	ReLU	deep library	no	yes	$O(N^2)$	Lemma 3.7
3	ReLU, symmetrized	includes deep library	yes, generalized	yes	$O(2^{2(L-1)}N^3)$ for deep library	Theorem 3.5 Theorem 3.13
3	absolute value	deep library	yes	yes	$O(N^2)$	Theorem 3.2, Theorem 3.11
4+	absolute value	includes deep library	yes, multilevel	yes	$O(N^{L-1}L!)$ for deep library	Theorem 3.2, Theorem 3.6

Table 3.1: Properties of equivalent Lasso problems for different architectures. Multilevel reflections include reflections of reflections. By Definition 3.10, the size of the deep library is $O(2^L N^{L-1})$. For absolute value, there is no 2^L since all weights can be 1.

Recall that $\hat{\mathbf{X}}^{(1,L)}$ (2.3) denotes a parallel unit, which is a special case of a standard network (Subsection 2.2). Let $\hat{\mathbf{X}}^{(l)}{=}\hat{\mathbf{X}}^{(1,l)}$, $\mathbf{W}^{(l)}{=}\mathbf{W}^{(1,l)}$, and $\mathbf{b}^{(l)}{=}\mathbf{b}^{(1,l)}$. The output $\hat{\mathbf{X}}^{(l)}$ of each layer can be interpreted as a feature extracted by that layer. This motivates the following definition.

Definition 3.9. *A bias parameter* $\mathbf{b}^{(l)}$ *is a* data feature bias *if* $\mathbf{b}^{(l)}{=}{-}\hat{\mathbf{X}}^{(l)}(\mathbf{x}_0)\mathbf{W}^{(l)}$ *for some column vector* $\mathbf{x}_0{\in}\{x_1,\cdots,x_N\}^{m_l}$. *The bias parameter* $\mathbf{b}^{(1)}$ *is a first-layer* midpoint feature bias *if* $\mathbf{b}^{(1)}{=}-m_{x_{j_1},x_{j_1'}}\mathbf{W}^{(1)}$ *for some* $j_1, j_1' \in [N]$ *and it is in a* 3*-layer deep narrow network with a non-monotone, continuous piecewise linear activation (such as* $\sigma(x)=|x|$*.)*

These bias parameters are used to define a deep library.

Definition 3.10 (Deep Library). *Consider a* 3*-layer symmetrized or* L*-layer deep narrow network. A* feature *is a function of the form* $\hat{\mathbf{X}}^{(L)}(x)$ *with data or midpoint feature biases and all elements of* $\mathbf{W}^{(1)}$ *being* ±1. *The* deep library *is the set of output vectors* $\hat{\mathbf{X}}^{(L)}(\mathbf{X}) \in \mathbb{R}^N$ *of all possible features on the training data.*

The deep library contains a finite number of standard networks evaluated on the training data. The next result shows that a neural network learns features in the deep library.

Theorem 3.11 (complete Lasso libraries for general activations). *Consider a deep narrow* L*-layer network where* $L{\in}\{2,3\}$ *and with activation* σ *which is ReLU, leaky ReLU, absolute value, sign, or threshold if* $L{=}2$*, and ReLU, leaky ReLU or absolute value if* $L{=}3$*. Consider*

a Lasso problem whose dictionary is the deep library and where ξ=0 if σ is sign or threshold. Suppose $(\mathbf{z}^*, \xi^*)$ is a solution, and let m^*=$\|\mathbf{z}^*\|_0$. This Lasso problem is equivalent to the training problem for the network, provided $m_L \geq m^*$.

The critical number of neurons satisfies $m^* \leq 2N$ for 2-layer networks and in general is at most the number of features. The notion of equivalence between optimization problems is that they have the same optimal value and their solutions can be found from each other. Definition B.3 defines a map to reconstruct a neural network from a Lasso solution, and Lemma B.4 shows that this is optimal. The general reconstructed network is equivalent to the function $f(x) = \sum_i z_i^* f_i(x) + \xi^*$ where $f_i(x) = \hat{\mathbf{X}}^{(i,L)}(x)$ is the feature corresponding to the i^{th} column $\mathbf{A}_i$ of the dictionary. In the simplest case when L=2, a feature is of the form $f_i(x) = \sigma\left(x\mathbf{W}^{(i,1)} + \mathbf{b}^{(i,1)}\right) = \sigma\left((x - x_n)\,\mathbf{W}^{(i,1)}\right)$ with a data feature bias $\mathbf{b}^{(i,1)} = -\mathbf{W}^{(i,1)}x_n$ where $\mathbf{W}^{(1)} = \pm 1$, and $f_i(\mathbf{X})$ is a vector in the deep library. Lemma B.2 explicitly describes features for various networks. While Theorem 3.2 describes a subset of the library for $L \geq 3$ layers, Theorem 3.11 defines the full library for 3 layers.

Remark 3.12. *The dictionary for an architecture discussed in Theorem 3.11 is a superset of any dictionary with the same architecture but shallower depth.*

The next theorem generalizes Theorem 3.5 to leaky ReLU activations.

Theorem 3.13. *The training problem for a 3-layer symmetrized network with monotone activations such as ReLU is equivalent to a Lasso problem with solution $(\mathbf{z}^*, \xi^*)$ and whose dictionary contains a deep library with $O(2^{2(L-1)}N^3)$ features, provided $m_L \geq m^*$, where $m^* = \|\mathbf{z}^*\|_0$.*

Theorem 3.13 states that the deep library is a sub-dictionary for symmetrized networks. Finding the full dictionary for a symmetrized network is an area of future work. Theorem 3.11 shows that instead of training a neural network with a non-convex problem and reaching a possibly local optimum, we can simply solve a straightforward Lasso problem whose convexity guarantees that gradient descent approaches global optimality. In previous work (11), a similar Lasso formulation is developed for networks with threshold activation but requires up to 2^N features of length N in the dictionary for a 2-layer network. In contrast, with 1-D data, Theorem 3.11 shows that at most $2N^2$ features are needed for a 2-layer network.

Remark 3.14. *Note that when the network is 2 layers, the equivalent Lasso dictionary only contains features with breakpoints at training data, leading to a prediction with breakpoints only at data locations. In contrast, when the network has 3 layers there can be breakpoints at* ***reflections*** *of data points with respect to other data points due to the reflection features. As a result, for activations such as absolute value and ReLU with symmetrized networks, the sequence of dictionaries as the network gets deeper converges to a richer library that includes reflections.*

Our approach lays the foundation to further analyze the evolution of feature libraries over expanding depth and widths as an area of future work. For 2-layer networks, the dictionary (Theorem 3.11) is simple, as described next.

Corollary 3.15 (2-layer libraries). *Let $\mathbf{A}_+, \mathbf{A}_- \in \mathbb{R}^{N\times N}$ with $(\mathbf{A}_+)_{i,n}$=$\sigma(x_i - x_n)$, $(\mathbf{A}_-)_{i,n}$=$\sigma(x_n - x_i)$. We can write the dictionary matrix for 2-layer networks as $\mathbf{A}$=$\mathbf{A}_+$ for absolute value and sign activations, and $\mathbf{A}$=$[\mathbf{A}_+, \mathbf{A}_-] \in \mathbb{R}^{N\times 2N}$ for ReLU, leaky ReLU, and threshold*

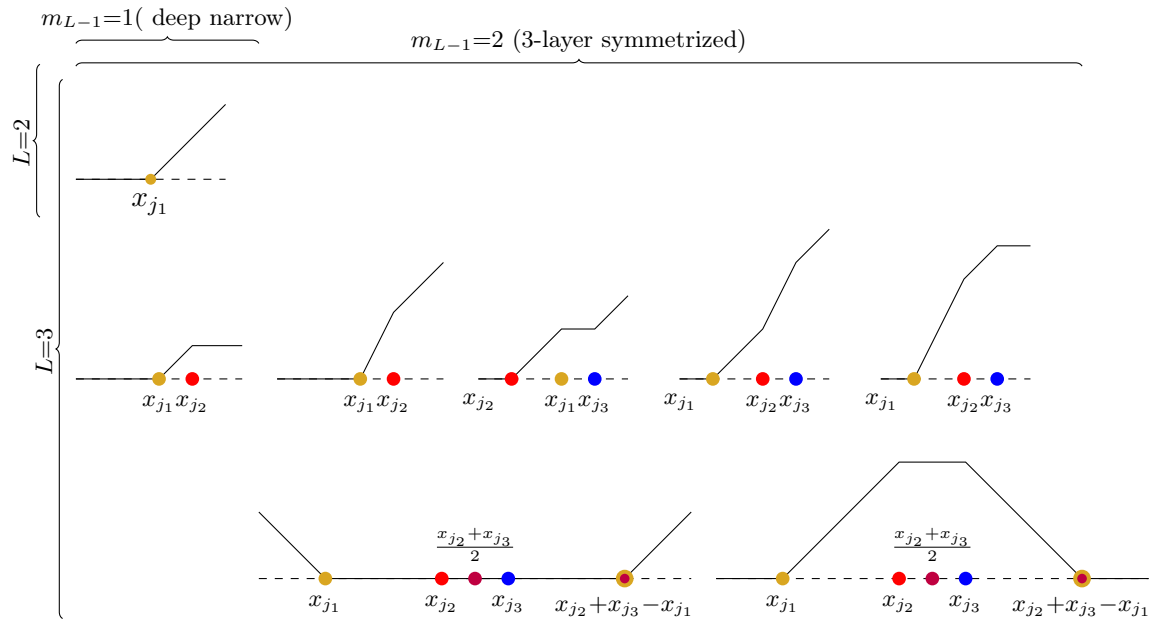


Figure 3.4: Example ReLU features, excluding reverse directions. Bottom row: 3-layer symmetrized ReLU features with breakpoints at generalized reflections of x_{j_1} (gold) across x_{j_2} (red) and x_{j_3} (blue) $x_{j_2}+x_{j_3}-x_{j_1}$ (gold encircling purple). Lines have slopes $\pm 2, \pm 1$ or 0.

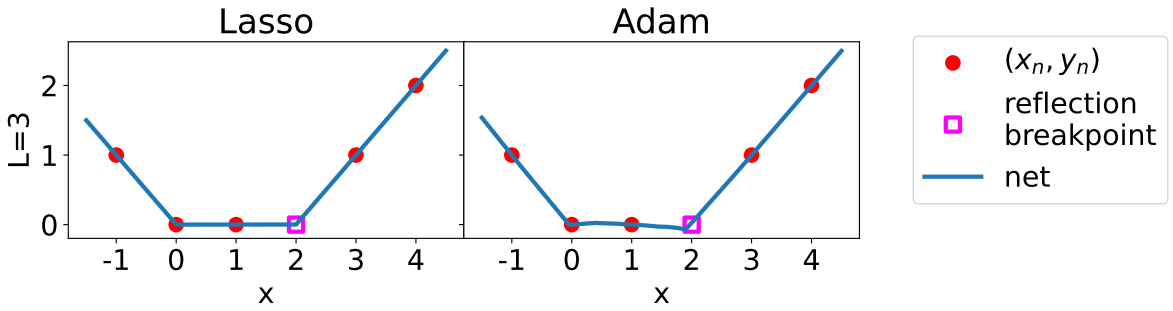


Figure 3.5: Lasso and Adam-trained 3-layer symmetrized ReLU networks. Most crucially, the breakpoint at 2 is not a training point; it is the reflection of $x_4 = 0$ across $x_3 = 1$.

activations.

In Corollary 3.15, $\mathbf{A}_+$ and $\mathbf{A}_-$ contain features $\hat{\mathbf{X}}^{(2)}(\mathbf{X})$ where $\mathbf{W}^{(1)} = 1$ and $\mathbf{W}^{(1)} = -1$, respectively (Definition 3.10). Figure 3.6 illustrates the features for $\mathbf{A}_+$ for the ReLU and threshold activations. Using the notation of Definition 3.10, the 3-layer deep narrow absolute value features are of the form $\hat{\mathbf{X}}^{(3)}(x)=\big||x-x_{j_1}|-|x_{j_2}-x_{j_1}|\big|$, and for 4 layers, include features of the form $\hat{\mathbf{X}}^{(4)}(x)=\Big|\big||x-x_{j_1}|-|x_{j_2}-x_{j_1}|\big|-\big||x_{j_3}-x_{j_1}|-|x_{j_2}-x_{j_1}|\big|\Big|$. These features are plotted

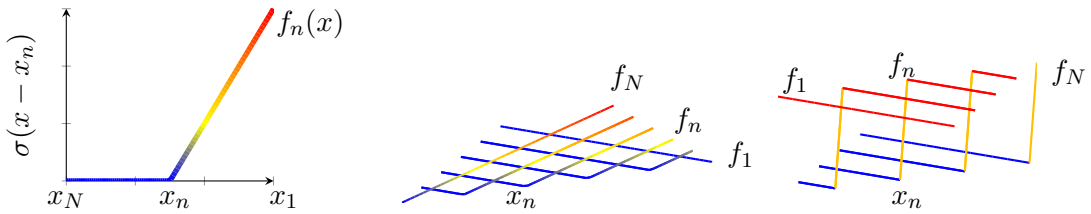


Figure 3.6: The features $f_n(x) = \sigma(x - x_n)$ define the 2-layer dictionary $\mathbf{A}_+ \in \mathbb{R}^{N \times N}$ as $\mathbf{A}_{+i,n} = f_n(x_i)$. An individual feature $f_n(x)$ for ReLU activation is plotted in the left figure, and it is plotted alongside all other $f_1(x), \cdots, f_N(x)$ in the middle figure. The right figure plots $f_1(x), \cdots, f_N(x)$ for threshold activation.

in Figure 3.1, which highlights their reflection and double reflection breakpoints. Figure 3.1 also illustrates breakpoints at averages and reflections of averages of training data, which correspond to midpoint feature biases in Definition 3.9. Figure 3.4 shows a subset of the ReLU library, with generalized reflection features for 3-layer symmetrized networks. Figure 3.4 illustrates general feature shapes not including mirrored directions. In Figure B.1, we choose distinct training samples $x_i, x_j, x_k \in \{-1, 0, 2\}$ and numerically compute all possible deep library features for 3-layer, symmetrized ReLU networks using Definition 3.10. Since the features $\hat{\mathbf{X}}^{(L)}(x)$ are continuous with respect to x_i, x_j, x_k, the features for non-distinct x_i, x_j, x_k can be extrapolated by merging adjacent training points. Figure B.1 shows that when exhaustively enumerating over all possible values of data feature biases and ± 1 weights, there are redundant features, and so the upper bound on features in Theorem 3.13 can be an overestimate. The numerically plotted features are consistent with Figure 3.4. In addition to graphical representations, Lemma B.2 in Appendix B gives examples of simple, explicit expressions for features defined in Definition 3.10.

3.4. Discussion: extensions, limitations and open problems. Appendix B.1 gives explicit examples of reconstructed networks and discusses the equivalence of the Lasso model and reconstructed network on test points. We note that the optimal solutions may not be unique. Appendix E discusses the relation between the solution sets of the Lasso and non-convex training problems with respect to a specified straightforward reconstruction. Lemma E.2 shows that the optimal Lasso fit $\hat{\mathbf{y}} = \mathbf{A}\mathbf{z}^* + \xi^* \mathbf{1}$, the linear fit $\mathbf{A}\mathbf{z}^*$ and the optimal bias ξ^* are unique, and the set of all optimal Lasso solutions can be characterized by an *equicorrelation set*, as described in Proposition E.3. Figure E.1 analyzes the generalization properties of neural networks reconstructed from different Lasso solutions, and shows that their performance can vary widely. Further analysis of all possible reconstruction maps between the models and the span of all stationary solutions of the non-convex model is an area of future work.

There is an analogous result to the convex equivalence in Theorem 3.11 for deep and wide sign and threshold networks which is provided in the supplementary material (47). Further extending the Lasso formulations for wider, more general neural network architectures is an area of future work. A limitation of this work is the computationally large number of features as the network grows larger and deeper. In practice, given a formulation of Lasso features, one can subsample features to reduce the computational complexity. Subsampling Lasso problem

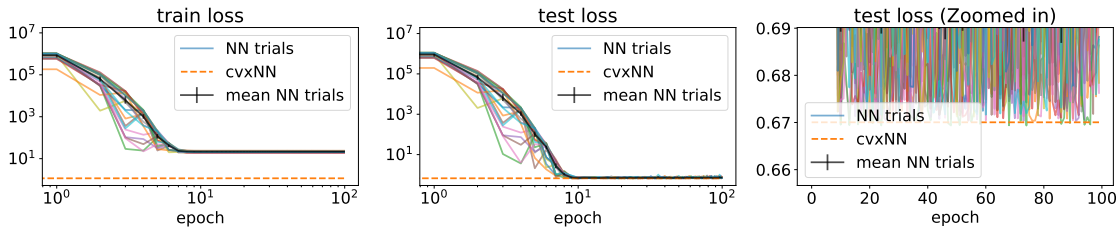


Figure 4.1: Comparison of autoregressive models $x_t = f(x_{t-1}; \theta) + \epsilon_t$ using convex (cvxNN) and non-convex (NN) training with 2-layer neural networks and the linear model AR(1) to forecast minutely 2017 Bitcoin prices (1). The non-linear models outperform the linear AR(1) model. Moreover, SGD generally underperforms in training and test loss compared to the convex model which is guaranteed to find a global optimum of the NN objective. The non-black, solid colored curves plot each NN trial, and the black curve plots their mean, with the vertical lines indicating one standard deviation of the data above and below the mean.

representations and their approximation guarantees has been studied in (15; 25; 44; 45) and is shown to maintain improved performance over training the non-convex network. Future work can adapt these approaches to the networks discussed in this paper. Regardless of the number of features, Lasso problems are fully transparent and intuitive as convex models that perform feature selection, in contrast to non-convex, black-box neural networks. There is extensive literature on the training and generalization behavior of Lasso models.

4. Application: Time-series modeling. In this section, we apply the Lasso problem for neural networks to an autoregression problem. Suppose at times $1, \cdots, T+1$ we observe data points $x_1, \cdots, x_{T+1} \in \mathbb{R}$ that follow the time-series model

$$x_t = f(x_{t-1}; \theta) + \epsilon_t, \tag{4.1}$$

where $f : \mathbb{R} \to \mathbb{R}$ is parameterized by some parameter θ and $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ represents observation noise. The parameter θ is unknown, and the goal is find θ that best fits the model (4.1) to the data $x_1, \cdots, x_{T+1}$. For example, the *auto-regressive model with lag* 1 (AR(1)) is a linear model $f(x; \theta) = ax$ where $\theta = \{a\}$ is chosen as a solution to $\min_{\theta \in \Theta} \sum_{t=1}^{T} (f(x_t; \theta) - x_{t+1})^2$. For a more expressive model, instead of $f(x; \theta) = ax$ suppose we use a 2-layer neural network

$$f_2^{\mathrm{NN}}(x; \theta) = \sum_{i=1}^{m} |xw_i + b_i| \alpha_i. \tag{4.2}$$

The parameter set is $\theta = \{w_i, b_i, \alpha_i\}_{i=1}^m$. Consider a training problem of the form

$$\min_{\theta \in \Theta} \frac{1}{T} \sum_{t=1}^{T} \mathcal{L}\left(f_2^{\mathrm{NN}}(x_t; \theta) - x_{t+1}\right) + \frac{\beta}{2} \|\theta_w\|_2^2. \tag{4.3}$$

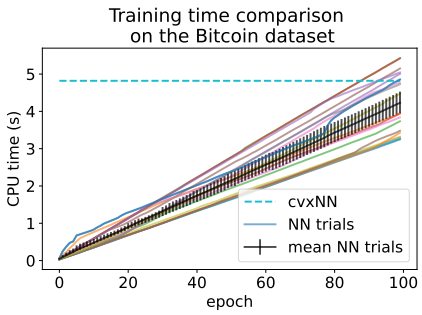


Figure 4.2: Training time for 2-layer neural networks trained on 2017 minutely Bitcoin data in Figure 4.1. The non-black, solid colored curves plot the time for each NN trial, and the black curve plots their mean, with the vertical lines indicating one standard deviation of the data above and below the mean.

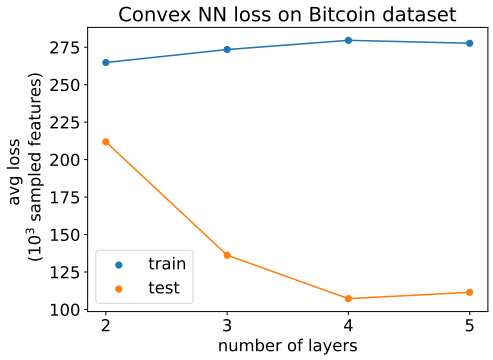


Figure 4.3: Mean squared loss over $T = 10^3$ time samples of Bitcoin data (BTC-2017min) for autoregressive deep narrow networks (one for each depth) trained on subsampled Lasso problems with 10^3 features.

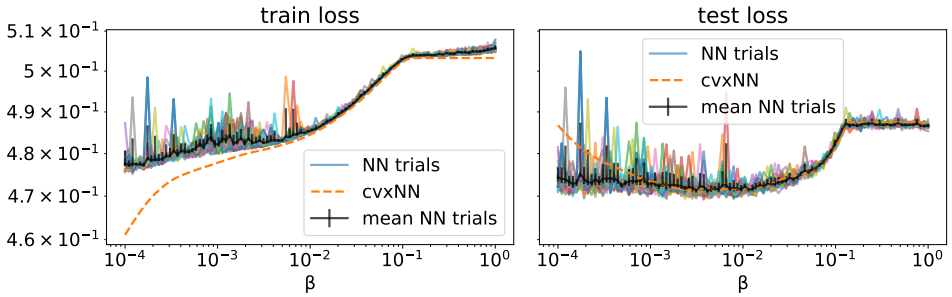


Figure 4.4: The regularization path for planted data with $\sigma^2 = 1$, $p = 5$ planted neurons. The non-black, solid colored curves plot each NN trial, and the black curve plots their mean, with the vertical lines indicating one standard deviation of the data above and below the mean.

If $\mathcal{L}(x)=x^2$ then (4.3) is an *autoregression* problem, where the network represents predictors of x_{t+1} from x_t. Problem (4.3) can be solved by converting it to an equivalent Lasso problem (1.2) where $A_{i,j}=|x_i-x_j|$ and $y_i = x_{i+1}$. Figure 4.1 shows that using the Lasso model to predict Bitcoin price reaches a lower loss than training with the non-convex model. In our experiments, we define the training loss as the objective of the training problem (4.3), which is the sum of the performance loss and the regularization term $\frac{\beta}{2}\|\theta_w\|_2^2$. We define the test loss as just the performance loss $\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}\left(f_2^{\mathrm{NN}}\left(x_t';\theta\right)-x_{t+1}'\right)$ evaluated on test data x_t', without the regularization term. Therefore we note that the training loss can be higher than the test loss,

as seen in Figure 4.1. We plot the training loss defined in this way to test our theory that the training loss is globally optimized using the convex training. We plot the test loss without the regularization term to show the practical performance of the network on unseen data and how well the regularization improved generalization. In Figure 4.2, for each trial of training the non-convex network, we measure the time that each epoch takes and plot the cumulative time. As shown in Figure 4.2, the non-convex model takes less time to train than the convex model until it converges to a solution, but the non-convex model converges to a suboptimal network.

We also investigate training on planted data. We build a network $f_2^{\text{NN}}(x;\theta)$ (4.2) with p known, or *planted* neurons. We use this network to generate training samples $x_1,\ldots,x_{T+1}$ based on (4.1) with $f(x;\theta)=f_2^{\text{NN}}(x;\theta)$ where $x_1\sim\mathcal{N}(0,\sigma^2)$. Using the same model $f_2^{\text{NN}}(x;\theta)$, we also generate test samples $x_1^{\text{test}},\ldots,x_{T+1}^{\text{test}}$ in an analogous way, independently of the training data. This allows us to generate independent testing and training sets with the same distributions, whereas in the real Bitcoin data, the testing samples subsequently follow the training samples and may still have time-based correlations with the training data. We use $T=1000$ time samples. Then, we try to recover the planted neurons based on only the training samples by solving the NN AR/QR training problems. The *regularization path* is the optimal neural net's performance loss as a function of the regularization coefficient β. Figure 4.4 plots the regularization path for σ^2=1 in (4.1) and p=10 neurons in the planted model. The regularization path is the optimal loss as a function of the regularization parameter β. Training the convex model finds an optimal training loss, while training the non-convex model may find a suboptimal loss. The loss versus β curves taken by the non-convex models, which are the jagged curves in Figure 4.4, are therefore noisy estimates of the true regularization path taken by the convex model, which is the smoother curve in Figure 4.4. More details are found in Appendix E.2.

In Figure 4.3 and Figure 4.5, deep narrow neural networks of various depths with $\sigma(x)=|x|$ are trained using the convex Lasso model on Bitcoin price data (minutely price in 2017). To allow a comparison where the training complexity is the same for all networks, each network uses 10^3 Lasso features that are randomly sampled from the deep library as follows. Since data feature biases enable reflection features, for simplicity and efficiency, the experiment uses deep library features with only data feature biases, and omits midpoint features. The experiment randomly and uniformly samples a data feature bias $\mathbf{b}^{(1)}=-x_{j_1}$ out of $j_1\in[N]$, which is used to create a 2-layer Lasso feature. Then another training sample x_{j_2} is randomly selected, and a 3-layer feature is created with $\mathbf{b}^{(1)}=-x_{j_1},\mathbf{b}^{(2)}=-\hat{\mathbf{X}}^{(2)}(x_{j_2})$ using the same x_{j_1} as for the 2-layer feature. Similarly another sample x_{j_3} is randomly selected and a 4-layer feature is constructed from x_{j_3} and the prior samples with $\mathbf{b}^{(1)}=x_{j_1},\mathbf{b}^{(2)}=\hat{\mathbf{X}}^{(2)}(x_{j_2}),\mathbf{b}^{(3)}=\hat{\mathbf{X}}^{(3)}(x_{j_3})$. A similar procedure creates a 5^{th}-layer feature. In this process, the feature selected for layer L in this process will either be the same feature as for layer $L-1$ if $j_{L-1}=j_{L-2}$, or be the feature for $L-1$ with an additional breakpoint. The experiment repeats this process 10^3 times to create a single subsampled Lasso training problem with 10^3 sampled features for each network depth L. Figure 4.3 and Figure 4.5 show the results for a single example network trained from such a subsampled Lasso. The experiment repeats this sampling procedure 200 times to create 200 subsampled Lasso problems for each depth. Figure 4.6 and Figure 4.7 show results from the 200 networks trained for each depth.

Note that an alternative sampling procedure could be first enumerating all features and

then randomly sampling from the entire feature set, which would create independent samples from different depths. However, since enumerating all features can be computationally intensive, the sampling is done in the sequential procedure described above. The dictionaries of the networks are not subsets of each other, and so the Lasso training loss does not necessarily decrease with depth, as seen in Figure 4.3.

Figure 4.3 and Figure 4.5 are consistent with Figure 4.6 and Figure 4.7, and show that while deeper networks can have higher training error as shown in Figure 4.6, they generally have better test error, at least up to a certain depth. The test performance of the networks in Figure 4.7 improves until $L = 4$ but slightly worsens for $L = 5$ (not shown in Figure 4.5). Figure 4.6 also shows that while 5-layer networks can perform well on test data depending on the subsampled features, they generally have high variance in test loss and large test errors. The improved test performance with limited depth suggests that certain deeper features, which have richer levels of reflections, tend to generalize data better.

This performance loss when $L = 5$ may be due to the network overfitting to overly complex features after the network is already deep enough that its features sufficiently capture the relationships in the data. It may also occur if the 10^3 subsampled features are no longer representative of the entire library, whose size increases with depth: the small fraction of subsampled features may be missing out on key features for the data. The number of subsampled features was set to 10^3 to reduce computational complexity, which is a practical limitation in this experiment. Increasing the number of sampled features may increase the performance of the 5-layer networks.

In our experiments, the Lasso-trained deep narrow networks of depth greater than 2 do not outperform deep networks that are trained with the non-convex training problem, unlike the experiment in Figure 4.1 where 2-layer convex networks outperformed non-convex networks. This may be due to insufficient Lasso feature sampling and hyperparameter choices in the non-convex program. A practical limitation affecting these results is insufficient sampling; addressing this issue will be the subject of future work.

5. Conclusion.

Our results show that certain deep neural networks with a variety of activation functions trained on 1-D data with weight regularization can be recast as convex Lasso models with simple dictionary matrices. This provides critical insight into their solution path as the weight regularization changes. The Lasso problem also provides a fast way to train neural networks for 1-D data. Moreover, the understanding of the neural networks through Lasso models could also be used to explore designing better neural network architectures.

We proved that reflection features can emerge in the Lasso dictionary for certain absolute value and ReLU networks when the depth is 3 or deeper. This leads to predictions that have breakpoints at reflections of data points about other data points. In contrast, for networks of depth 2, the breakpoints are only located at a subset of training data. We believe that this mechanism enables deep neural networks to generalize to the unseen by encoding a geometric regularity prior.

Our analysis of various architectures provides a foundation for studying more complex network topologies. The 1-D results can extend to sufficiently structured or low rank data in higher dimensions. Generalizing to higher dimensions is also an area of future work. Building on a similar theme, (32) showed that the structure of hidden neurons can be expressed through

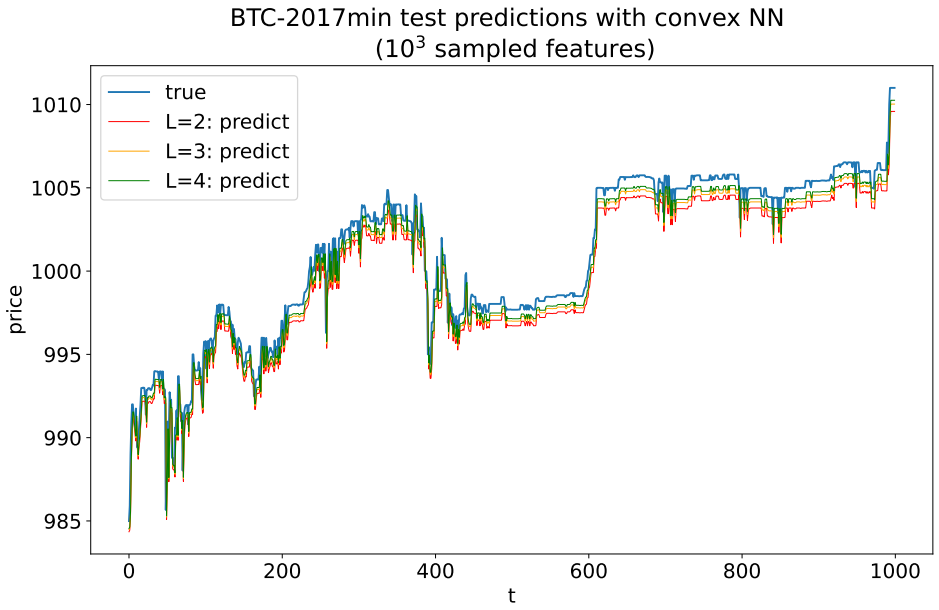


Figure 4.5: Predictions of example autoregressive neural networks (one for each depth) trained using the convex Lasso problem with 10^3 sampled features on predicting $T = 10^3$ BTC-2017min samples. The network predictions improve as the depth increases from $L = 2$ to $L = 4$. For each network of depth L, the regularization is $\beta = \frac{1}{T}$ in the training problem (4.3) for 2 layers and (1.1) for more general L layers.

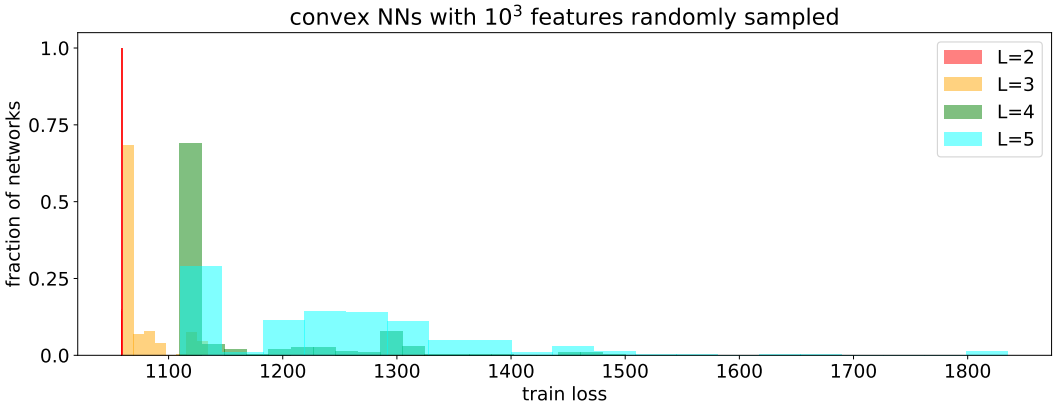


Figure 4.6: Histogram of training loss of deep narrow networks (200 networks for each depth L) trained on the Lasso problem with randomly sampled 10^3 features for autoregression on BTC-2017min with $T = 10^3$ samples. trained on the Lasso problem with randomly sampled 10^3 features for autoregression on BTC-2017min with $T = 10^3$ samples.

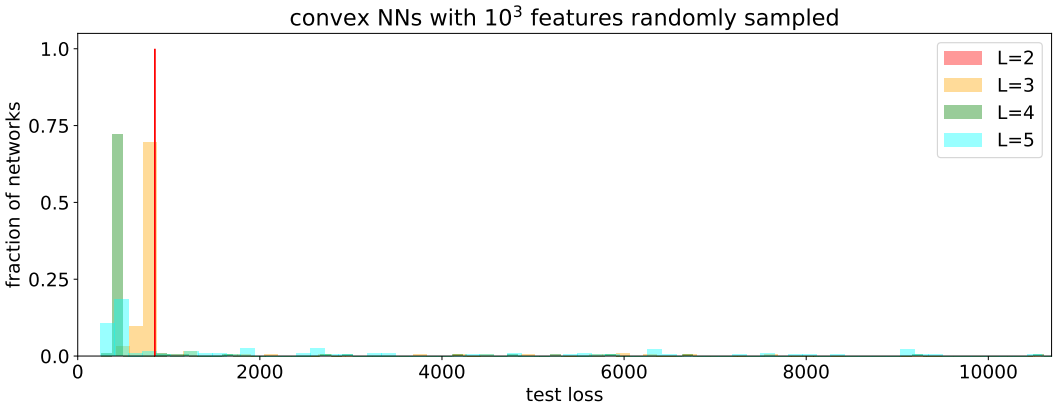


Figure 4.7: Histogram of training loss of deep narrow networks (200 networks for each depth L) trained on the Lasso problem with randomly sampled 10^3 features for autoregression on BTC-2017min with $T = 10^3$ samples. The test loss appears to improves with depth. For visual clarity, the loss is only shown up to a value that contains $\geq 80\%$ of the 4-layer samples, $\geq 90\%$ of the 3-layer samples, and all of the 2-layer samples (using the full dictionary.)

convex optimization and Clifford's Geometric Algebra. The techniques developed in this paper can be combined with the Clifford Algebra to develop higher-dimensional analogues of the results.

References.

[1] *Kaggle*, https://www.kaggle.com/datasets/prasoonkottarathil/btcinusd.

[2] S. Arora, N. Cohen, W. Hu, and Y. Luo, *Implicit regularization in deep matrix factorization*, Advances in neural information processing systems, 32 (2019).

[3] F. Bach, *Breaking the curse of dimensionality with convex neural networks*, JMLR, 18 (2017), pp. 629–681.

[4] 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.

[5] J. M. Borwein and A. S. Lewis, *Convex Analysis and Nonlinear Optimization: Theory and Examples*, Springer, 2000.

[6] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[7] A. Bulat and G. Tzimiropoulos, *XNOR-Net++: Improved binary neural networks*, ArXiv, abs/1909.13863 (2019).

[8] M.-H. Chen and B. W. Schmeiser, *General hit-and-run monte carlo sampling for evaluating multidimensional integrals*, Operations Research Letters, 19 (1996), pp. 161–169.

[9] P. Chen, K. Wu, J. Chen, T. O'Leary-Roseberry, and O. Ghattas, *Projected Stein variational newton: A fast and scalable Bayesian inference method in high dimensions*, Advances in Neural Information Processing Systems, 32 (2019).

[10] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, *Least angle regression*, The Annals of statistics, 32 (2004), pp. 407–499.

[11] T. Ergen, H. I. Gulluk, J. Lacotte, and M. Pilanci, *Globally optimal training of neural networks with threshold activation functions*, arXiv:2303.03382, (2023).

[12] T. Ergen and M. Pilanci, *Convex geometry of two-layer ReLU networks: Implicit autoencoding and interpretable models*, PMLR, 26–28 Aug. 2020, pp. 4024–4033.

[13] T. Ergen and M. Pilanci, *Convex geometry and duality of over-parameterized neural networks*, The Journal of Machine Learning Research, 22 (2021), pp. 9646–9708.

[14] T. Ergen and M. Pilanci, *Revealing the structure of deep neural networks via convex duality*, in ICML, PMLR, 2021, pp. 3004–3014.

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

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

[17] S. Feizi, H. Javadi, J. M. Zhang, and D. Tse, *Porcupine neural networks: (almost) all local optima are global*, ArXiv, abs/1710.02196 (2017).

[18] K. Genova, F. Cole, D. Vlasic, A. Sarna, W. T. Freeman, and T. A. Funkhouser, *Learning shape templates with structured implicit functions*, 2019 IEEE/CVF International Conference on Computer Vision (ICCV), (2019), pp. 7153–7163, https://api.semanticscholar.org/CorpusID:118642996.

[19] G. Gidel, F. Bach, and S. Lacoste-Julien, *Implicit regularization of discrete gradient dynamics in linear neural networks*, Advances in Neural Information Processing Systems, 32 (2019).

[20] S. Gunasekar, B. E. Woodworth, S. Bhojanapalli, B. Neyshabur, and N. Srebro, *Implicit regularization in matrix factorization*, Advances in neural information processing systems, 30 (2017).
[21] N. Joshi, G. Vardi, and N. Srebro, *Noisy interpolation learning with shallow univariate relu networks*, arXiv.2307.15396, (2023).
[22] K. Karhadkar, M. Murray, H. Tseran, and G. Montúfar, *Mildly overparameterized relu networks have a favorable loss landscape*, arXiv:2305.19510, (2023), https://arxiv.org/abs/2305.19510.
[23] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, *Physics-informed machine learning*, Nature Reviews Physics, 3 (2021), pp. 422–440, https://doi.org/10.1038/s42254-021-00314-5, https://doi.org/10.1038/s42254-021-00314-5.
[24] M. Kim and P. Smaragdis, *Bitwise neural networks for efficient single-channel source separation*, in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 701–705.
[25] S. Kim and M. Pilanci, *Convex relaxations of relu neural networks approximate global optima in polynomial time*, ICML, (2024).
[26] G. Kornowski, G. Yehudai, and O. Shamir, *From tempered to benign overfitting in ReLU neural networks*, arXiv:2305.15141, (2023), https://arxiv.org/abs/2305.15141.
[27] A. Mishkin and M. Pilanci, *The solution path of the group lasso*, 2022, https://api.semanticscholar.org/CorpusID:259504228.
[28] A. Mishkin and M. Pilanci, *Optimal sets and solution paths of ReLU networks*, in International Conference on Machine Learning, ICML 2023, PMLR, 2023.
[29] A. Mishkin and M. Pilanci, *Optimal sets and solution paths of relu networks*, in Proceedings of the 40th International Conference on Machine Learning, ICML'23, JMLR.org, 2023.
[30] B. Neyshabur, *Implicit regularization in deep learning*, arXiv preprint arXiv:1709.01953, (2017).
[31] M. R. Osborne, B. Presnell, and B. A. Turlach, *A new approach to variable selection in least squares problems*, IMA journal of numerical analysis, 20 (2000), pp. 389–403.
[32] M. Pilanci, *From complexity to clarity: Analytical expressions of deep neural network weights via Clifford's geometric algebra and convexity*, arXiv:2309.16512, (2023).
[33] M. Pilanci and T. Ergen, *Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks*, in Proceedings of the 37th International Conference on Machine Learning, vol. 119, 13–18 July 2020, pp. 7695–7705.
[34] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S.-Y. Chang, and T. Sainath, *Deep learning for audio signal processing*, IEEE Journal of Selected Topics in Signal Processing, 13 (2019), pp. 206–219.
[35] P. Savarese, I. Evron, D. Soudry, and N. Srebro, *How do infinite width bounded norm networks look in function space?*, Annual Conference on Learning Theory, (2019), pp. 2667–2690.
[36] J. Serrà, S. Pascual, and C. S. Perales, *Blow: a single-scale hyperconditioned flow for non-parallel raw-audio voice conversion*, Advances in Neural Information Processing

Systems, 32 (2019).
[37] K. O. Stanley, *Compositional pattern producing networks: A novel abstraction of development*, Genetic Programming and Evolvable Machines, 8 (2007), p. 131–162, https://doi.org/10.1007/s10710-007-9028-8, https://doi.org/10.1007/s10710-007-9028-8.
[38] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, *Fourier features let networks learn high frequency functions in low dimensional domains*, in Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Red Hook, NY, USA, 2020, Curran Associates Inc.
[39] R. Tibshirani, *Regression shrinkage and selection via the Lasso*, Journal of the Royal Statistical Society: Series B (Methodological), 58 (1996), pp. 267–288, https://arxiv.org/abs/https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1996.tb02080.x.
[40] R. J. Tibshirani, *The lasso problem and uniqueness*, Electronic Journal of Statistics, 7 (2013), pp. 1456–1490.
[41] S. Vaiter, C. Deledalle, G. Peyré, J. Fadili, and C. Dossal, *The degrees of freedom of the group lasso for a general design*, CoRR, abs/1212.6478 (2012).
[42] Y. Wang, P. Chen, and W. Li, *Projected Wasserstein gradient descent for high-dimensional Bayesian inference*, SIAM/ASA Journal on Uncertainty Quantification, 10 (2022), pp. 1513–1532.
[43] Y. Wang, P. Chen, M. Pilanci, and W. Li, *Optimal neural network approximation of Wasserstein gradient direction via convex optimization*, arXiv:2205.13098, (2022).
[44] Y. Wang, S. Kim, P. Chu, I. Subramaniam, and M. Pilanci, *Randomized geometric algebra methods for convex neural networks*, ArXiv, abs/2406.02806 (2024), https://api.semanticscholar.org/CorpusID:270258553.
[45] Y. Wang, J. Lacotte, and M. Pilanci, *The hidden convex optimization landscape of regularized two-layer relu networks: an exact characterization of optimal solutions*, in International Conference on Learning Representations, 2021.
[46] T. E. Yifei Wang and M. Pilanci, *Parallel deep neural networks have zero duality gap*, ICLR, (2023s).
[47] E. Zeger, Y. Wang, T. E. Aaron Mishkin, E. Cand'es, and M. Pilanci, *A library of mirrors (technical report)*, arXiv preprint:2403.01046, (2024).

Appendix A. Detailed results for Section 2.

A.1. Parallel to standard architecture conversion. Given a parallel network, a standard network can be constructed as follows. Let $\mathbf{W}^{(1)} = \left[\mathbf{W}^{(1,1)} \cdots \mathbf{W}^{(m_1,1)}\right]$. For $l \geq 1$, let $\mathbf{b}^{(l)} = \left(\mathbf{b}^{(1,l)} \cdots \mathbf{b}^{(m_l,l)}\right)$. For $l > 1$, let $\mathbf{W}^{(l)} = \text{blockdiag}\left(\mathbf{W}^{(1,l)} \cdots \mathbf{W}^{(m_l,l)}\right)$. And let $\boldsymbol{\alpha}, \xi$ be the same in the standard network as the parallel one.

A.2. 2-layer network equivalences. A 2-layer network is a standard network $f_2(\mathbf{x};\theta) = \xi + \mathbf{X}^{(2)}\boldsymbol{\alpha}$ by setting $L = 2, m_1 = d$ in (2.2) as $\mathbf{X}^{(2)} = \sigma\left(\mathbf{X}^{(1)}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}\right)$ where $\mathbf{W}^{(1)} \in \mathbb{R}^{m_1 \times m_2}, \mathbf{b}^{(1)} \in \mathbb{R}^{1 \times m_2}$. In other words, $\mathbf{W}^{(1)}$ contains $\mathbf{W}^{(i,1)}$ as columns, and $\mathbf{b}^{(1)}$ contains $\mathbf{b}^{(i,1)}$ as elements.

A 2-layer network is a special case of a parallel network $f_2(\mathbf{x};\theta) = \xi + \sum_{i=1}^{m_2} \hat{\mathbf{X}}^{(i,2)}\alpha_i$ by setting $L = 2, m_0 = d, m_1 = m_{L-1} = 1$ in (2.3) where $\hat{\mathbf{X}}^{(i,2)} = \sigma\left(\hat{\mathbf{X}}^{(i,1)}\mathbf{W}^{(i,1)} + \mathbf{b}^{(i,1)}\right)$ for $\mathbf{W}^{(i,1)} \in \mathbb{R}^{m_0 \times m_1}, \mathbf{b}^{(i,1)} \in \mathbb{R}^{1 \times m_1}$. A 2-layer network is also a special case of a deep narrow network where $m_1{=}m_{L-1}{=}1$.

Appendix B. Detailed results for Section 3 .

Proofs are defered to Appendix D.3.

Definition B.1. *Define the function $\mathcal{W}_{\alpha,\beta} : \mathbb{R} \to \mathbb{R}$ parameterized by $\alpha, \beta \in \mathbb{R}$ as*

$$\mathcal{W}_{\alpha,\beta}(x) = \begin{cases} \alpha - x & \text{if } x \leq \alpha \\ x - \alpha & \text{if } \alpha \leq x \leq \beta \\ R_{(\alpha,\beta)} - x & \text{if } \beta \leq x \leq R_{(\alpha,\beta)} \\ x - R_{(\alpha,\beta)} & \text{if } x \geq R_{(\alpha,\beta)}. \end{cases}$$

Figure 3.4 plots all ReLU features $\hat{\mathbf{X}}^{(L)}(x)$ for an example dataset. As shown in the figure, symmetrized ReLU network features contain generalized reflections of the form $x_i{+}x_j{-}x_k$. We now describe features for other activations.

Lemma B.2 (Examples of the Deep Library). *Consider a deep narrow network of depth $L \in \{2, 3\}$.*
If $\mathbf{L = 2}$*: for* $\mathbf{b}^{(1)} = -x_j\mathbf{W}^{(1)}$,

$$\hat{\mathbf{X}}^{(L)}(x) = \begin{cases} \sigma(x_{j_1} - x) & \text{if } \mathbf{W}^{(1)} = -1 \\ \sigma(x - x_{j_1}) & \text{if } \mathbf{W}^{(1)} = 1 \end{cases}$$

If $\mathbf{L = 3}$***:***
if $\sigma(x) = ReLU(x)$ and $m_1 = 1$: for $\mathbf{b}^{(1)} = -x_{j_1}\mathbf{W}^{(1)}$,
if $\mathbf{W}^{(2)} = 1$*:*

$$\hat{\mathbf{X}}^{(2)}(x) = \begin{cases} ReLU^{-}_{\min\{x_{j_1}, x_{j_2}\}}(x) & \text{if } \mathbf{W}^{(1)} = -1 \\ ReLU^{+}_{\max\{x_{j_1}, x_{j_2}\}}(x) & \text{if } \mathbf{W}^{(1)} = 1 \end{cases}$$

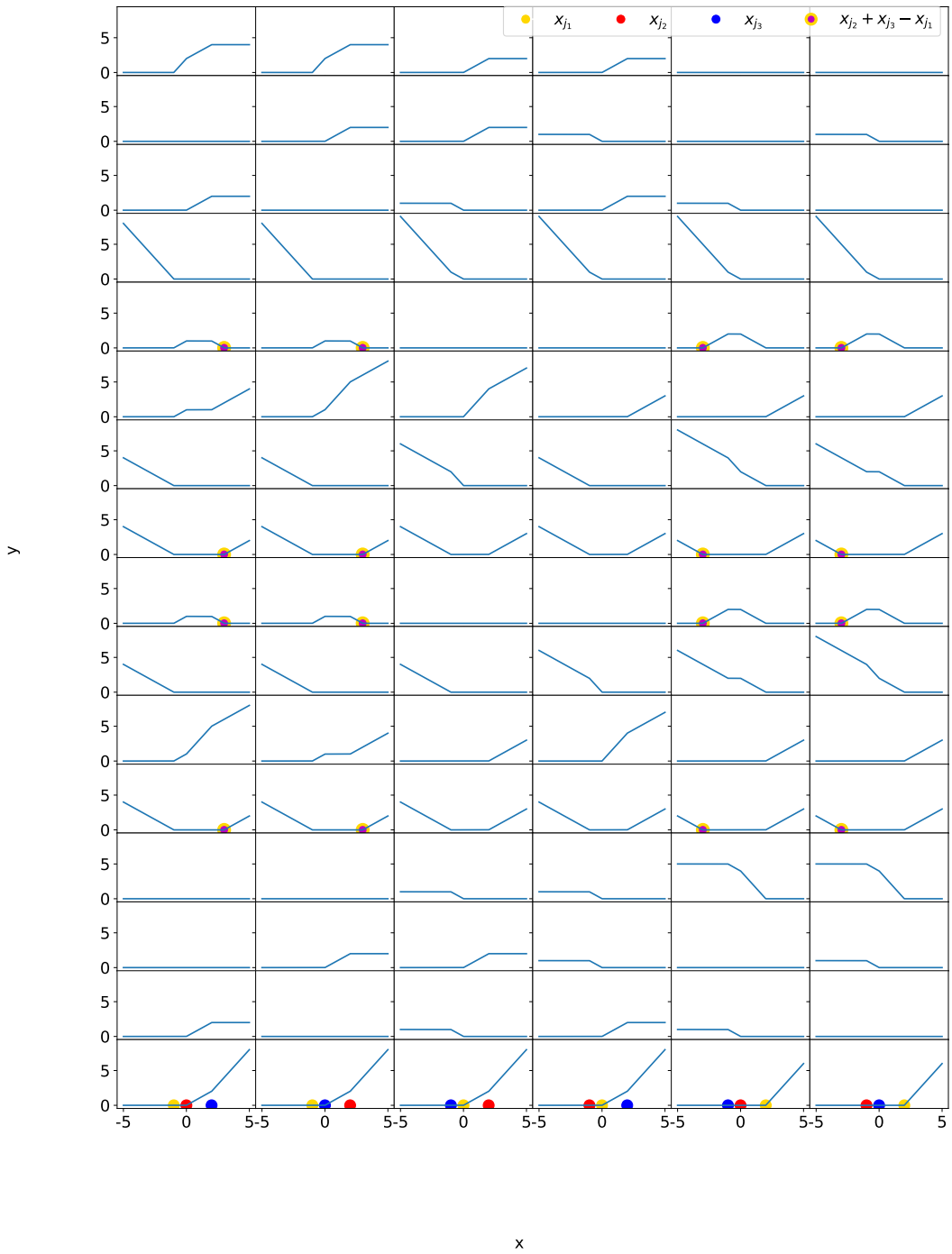


Figure B.1: Figure for Appendix B. Deep library features for a 3-layer symmetrized ReLU network. Each row corresponds to a different set of weights $\mathbf{W}^{(1)} \in \{-1, 1\}^{1\times 2}, \mathbf{W}^{(2)} \in \{-1, 1\}^{2\times 1}$. Each column corresponds to a different ordering of x_i, x_j, x_k. The generalized reflections of x_{j_1} (gold) across x_{j_2} (red) and x_{j_3} (blue) are depicted by gold encircling purple.

if $\mathbf{W}^{(2)} = -1$*:*

$$\hat{\mathbf{X}}^{(L)}(x) = \begin{cases} Ramp^{+}_{x_{j_2},x_{j_1}}(x) & \text{if } \mathbf{W}^{(1)} = -1 \\ Ramp^{-}_{x_{j_1},x_{j_2}}(x) & \text{if } \mathbf{W}^{(1)} = 1. \end{cases}$$

if $\sigma(x) = |x|$ *and* $m_1 = 1$*: for* $a \in \left\{x_{j_1}, \frac{x_{j_1}+x_{j_2}}{2}\right\}$,

$$\hat{\mathbf{X}}^{(L)}(x) = \begin{cases} \mathcal{W}_{\min\left\{x_{j_2}, R_{\left(x_{j_2},a\right)}\right\},a}(x) & \text{if } \mathbf{b}^{(1)} = -a\mathbf{W}^{(1)} \\ \mathcal{W}_{\min\left\{R_{\left(a,x_{j_1}\right)}, R_{\left(a,x_{j_2}\right)}\right\},a}(x) & \text{if } \mathbf{b}^{(1)} = -a\mathbf{W}^{(1)}. \end{cases}$$

B.1. Reconstruction results. In this section, let $(\mathbf{z}^*, \xi^*)$ be a solution to the Lasso problem. We give a map to efficiently and explicitly reconstruct an optimal neural net from $(\mathbf{z}^*, \xi^*)$ by leveraging the structure of the deep library (Definition 3.10). Proofs are deferred to Appendix D.3.

Lasso model on future test points: The Lasso problem can be interpreted as finding a linear model $f : \mathbb{R}^m \to \mathbb{R}, f(\mathbf{a}) = \mathbf{z}^T\mathbf{a} + \xi$ such that for given samples $\mathbf{a}_1, \cdots, \mathbf{a}_N \in \mathbb{R}^m$ and labels $y_1, \cdots, y_N \in \mathbb{R}$, we have $f(\mathbf{a}_i) \sim y_i$. Now suppose the future test point is $x \in \mathbb{R}$. For simplicity, consider a 2-layer network with $\sigma(x) = x$. Since $\sigma(x) = x$, the Lasso dictionary is a simple $n \times n$ dictionary with $A_{i,j} = \sigma(x_i - x_j)$. For $\mathbf{a} = (\sigma(x - x_1), \cdots, \sigma(x - x_N))$, the Lasso model and reconstructed network have the same value as $f(\mathbf{a}) = \mathbf{z}^T\mathbf{a} + \xi = \sum_{i=1}^N z_i\sigma(x - x_i) + \xi$.

Definition B.3 (Reconstructed parameters). *The* reconstructed parameters *for a parallel network are constructed as follows. For each* i^{th} *column* $\mathbf{A}_i$ *of the dictionary matrix such that* $z_i^* \neq 0$*, let* $\hat{\mathbf{X}}^{(i,L)}$ *be the parallel unit* $\hat{\mathbf{X}}^{(L)}$ *corresponding to that column in the deep library. Let* $\boldsymbol{\alpha}$=$\mathbf{z}^*$ *and* ξ=ξ^**. For sign and threshold activation, let all amplitude parameters be* 1*. Finally, unscale parameters (Definition D.17).*

Definition B.3 reconstructs parameters using the columns of the deep library, which are defined in Definition 3.9 and used to construct the Lasso problem. A *reconstructed network* is a network with reconstructed parameters. Some examples of optimal reconstructed parameters and networks are given below.

Reconstruction Examples. Let $\mathbf{z}^*, \xi^*$ be a solution to the Lasso problem. Let $\mathcal{J} = \{j : z_j^* \neq 0\}$. The following are some examples of optimal reconstructed networks using Definition B.3.
For simplicity, let $\sigma(x) = |x|$. For a 2-layer network, a set of optimal parameters is

$$\alpha_j = \text{sign}(z_j^*)\sqrt{|z_j^*|}, \mathbf{W}^{(j,1)} = \sqrt{|z_j^*|}, \mathbf{b}^{(j,1)} = -x_j\sqrt{|z_j^*|}, \xi = \xi^*. \tag{B.1}$$

The corresponding optimal reconstructed network is

$$f(x) = \sum_{j\in\mathcal{J}} \sigma\left(x\mathbf{W}^{(j,1)} + \mathbf{b}^{(j,1)}\right)\alpha_j + \xi \tag{B.2}$$

$$= \sum_{j\in\mathcal{J}} \sigma\left(x\sqrt{|z_j^*|} - x_j\sqrt{|z_j^*|}\right)\operatorname{sign}(z_j^*)\sqrt{|z_j^*|} + \xi^* \tag{B.3}$$

$$= \sum_{j\in\mathcal{J}} z_j^*\sigma(x - x_j) + \xi^*. \tag{B.4}$$

By optimal parameter scaling (Definition D.17), the optimal weights and biases have a factor of $\sqrt{|z_j^*|}$ in (B.1), which are plugged in to the neural network form (B.2) to yield (B.3), which simplifies to (B.4). The reconstruction for general activations is similar.

A 3-layer deep narrow network has dictionary elements $A_{i,j} = \sigma(\sigma(x_i - x_{j_1}) - \sigma(x_{j_2} - x_{j_1}))$ where the columns are indexed by $j = (j_1, j_2)$. A set of optimal parameters is $\alpha_j = \operatorname{sign}(z_j^*)|z_j^*|^{\frac{1}{3}}, \mathbf{W}^{(j,2)} = |z_j^*|^{\frac{1}{3}}, \mathbf{W}^{(j,1)} = |z_j^*|^{\frac{1}{3}}, \mathbf{b}^{(j,2)} = -\sigma(x_{j_2} - x_{j_1})|z_j^*|^{\frac{2}{3}}, \mathbf{b}^{(j,1)} = -x_{j_1}|z_j^*|^{\frac{1}{3}}, \xi = \xi^*$. The corresponding optimal reconstructed network simplifies to

$$f(x) = \sum_{j\in\mathcal{J}} z_j^*\sigma\left(\sigma(x - x_{j_1}) - \sigma(x_{j_2} - x_{j_1})\right) + \xi^*.$$

For a 3-layer symmetrized network, a set of optimal parameters is $\alpha_j = \operatorname{sign}(z_j^*)|2z_j^*|^{\frac{1}{3}}$, $\mathbf{W}_1^{(j,2)} = s_{j_4}\frac{1}{\sqrt{2}}|2z_j^*|^{\frac{1}{3}}, \mathbf{W}_2^{(j,2)} = s_{j_5}\frac{1}{\sqrt{2}}|2z_j^*|^{\frac{1}{3}}, \mathbf{W}_1^{(j,1)} = s_{j_6}\frac{1}{\sqrt{2}}|2z_j^*|^{\frac{1}{3}}, \mathbf{W}_2^{(j,1)} = s_{j_7}\frac{1}{\sqrt{2}}|2z_j^*|^{\frac{1}{3}}$, $\mathbf{b}_1^{(j,1)} = -x_{j_1}\mathbf{W}_1^{(j,1)}\frac{|2z_j^*|^{\frac{1}{3}}}{\sqrt{2}}, \mathbf{b}_2^{(j,1)} = -x_{j_2}\mathbf{W}_2^{(j,1)}\frac{|2z_j^*|^{\frac{1}{3}}}{\sqrt{2}}$, $\mathbf{b}^{(j,2)} = -\left(-\left(\sigma\left((x_{j_3} - x_{j_1})W_1^{(j,1)}\right)W_1^{(j,2)} - \sigma\left((x_{j_3} - x_{j_2})W_2^{(j,1)}\right)W_2^{(j,2)}\right)\right)\frac{|2z_j^*|^{\frac{2}{3}}}{2}, \xi = \xi^*$, where $s_{j_4}, \cdots, s_{j_7} \in \{-1, 1\}$ and the multi-index is $j = (j_1, \cdots, j_7)$. The corresponding optimal reconstructed network simplifies to

$$f(x) = \sum_{j\in\mathcal{J}} z_j^*\sigma\Bigg(\left(\left(\sigma\left((x - x_{j_1})W_1^{(j,1)}\right)W_1^{(j,2)} + \sigma\left((x - x_{j_2})W_2^{(j,1)}\right)W_2^{(j,2)}\right)\right)$$
$$- \left(\sigma\left((x_{j_3} - x_{j_1})W_1^{(j,1)}\right)W_1^{(j,2)} - \sigma\left((x_{j_3} - x_{j_2})W_2^{(j,1)}\right)W_2^{(j,2)}\right)\Bigg) + \xi^*.$$

General 3-layer, wide networks are out of the scope of the paper.

Lemma B.4. *A reconstructed parallel network is optimal in the training problem.*

Appendix C. Solution sets of Lasso under minimal regularization. One of the insights that the Lasso formulation provides is that under minimal regularization, certain neural nets perfectly interpolate the data. Proofs in this section are deferred to Appendix D.4.

Corollary C.1. *For the ReLU, absolute value, sign, and threshold networks with $L = 2$ layers, and sign-activated deeper networks, if $m_L \geq m^*$, then $f_L(\mathbf{X}; \theta) \to \mathbf{y}$ as $\beta \to 0$.*

In Corollary C.1, m^* depends on L and the activation and is defined in Theorem 3.11. The Lasso equivalence and reconstruction also shed light on optimal neural network structure as regularization decreases. The *minimum (l_1) norm subject to interpolation* version of the Lasso problem is

$$\min_{\mathbf{z},\xi} \|\mathbf{z}\|_1 \text{ s.t. } \mathbf{A}\mathbf{z} + \xi\mathbf{1} = \mathbf{y}. \tag{C.1}$$

Assume $\mathbf{A}$ has at least as many columns as rows. Loosely speaking, as $\beta \to 0$, if $\mathbf{A}$ has full rank, the Lasso problem (1.2) "approaches" the minimum norm problem (C.1), where $\xi = 0$ for sign and threshold activations (27; 29). The rest of this section describes the solution sets of (C.1) for certain networks. Recall that the training samples are ordered. For 2-layer networks with absolute value activation, the dictionary $\mathbf{A}$ is $N \times N$ where $A_{i,j} = |x_i - x_j|$, and the next result analyzes the first and last elements z_1^* and z_N^* of a solution to the Lasso problem.

Proposition C.2. *Let $L = 2$. Suppose σ is the absolute value activation. Let $\mathbf{z}^*$ be a solution to (C.1). Then, we have $z_1^* z_N^* \leq 0$. Moreover, the entire solution set of (C.1) for $\mathbf{z}^*$ is*

$$\left\{ \mathbf{z}^* + t \text{ sign}(z_1^*)(1, 0, \cdots, 0, 1)^T \middle| -|z_1^*| \leq t \leq |z_N^*| \right\}. \tag{C.2}$$

Proposition C.3. *For a 2-layer network with sign activation and $\beta \geq 0$, the Lasso problem (1.2) has a unique solution. Furthermore, the minimum norm solution in (C.1) is $\mathbf{z}^* = \mathbf{A}^{-1}\mathbf{y}$.*

Given an optimal bias term ξ^*, if $\mathbf{A}$ is invertible, then $\mathbf{z}^* = \mathbf{A}^{-1}(\mathbf{y} - \xi^*\mathbf{1})$ is optimal in (C.1). Appendix D.5 explicitly finds $\mathbf{A}^{-1}$ for some activation functions. The structure of $\mathbf{A}^{-1}$ suggests the behavior of neural networks under minimal regularization: sign-activated neural networks act as difference detectors, while neural networks with absolute value activation, whose subgradient is the sign activation, act as a second-order difference detectors (see Remark D.23). The next result shows that threshold-activated neural networks are also difference detectors, but for the special case of positive, nonincreasing y_n. An example of such data is cumulative revenue, e.g. $y_n = \sum_{i=1}^n r_i$ where r_i is the revenue in dollars earned on day i. Since the data is assumed to be ordered, $i < j$ implies $x_i \geq x_j$, and hence for nonincreasing y_n, we have $y_i \geq y_j$.

Proposition C.4. *Let $L = 2$. Suppose σ is threshold activation and $y_1 \geq \cdots \geq y_N \geq 0$. Then*

$$z_n^* = \begin{cases} y_n - y_{n-1} & \text{if } n \leq N-1 \\ y_N & \text{if } n = N \\ 0 & \text{else} \end{cases}$$

is the unique solution to the minimum norm problem (C.1).

The next result gives a lower bound on the optimal value of the minimum weight problem for ReLU networks. If we can find $\mathbf{z}$ with a l_1-norm that meets the lower bound and a ξ such that $\mathbf{A}\mathbf{z} + \xi\mathbf{1} = \mathbf{y}$, then we know $\mathbf{z}, \xi$ is optimal. In this section, for $n \in [N-1]$, let $\mu_n = \frac{y_n - y_{n+1}}{x_n - x_{n+1}}$ be the slope between the n^{th} and $n+1^{th}$ data points. Let $\mu_N = 0$.

Lemma C.5. *The optimal value $\|\mathbf{z}^*\|_1$ of the minimum norm problem (C.1) for deep narrow networks with ReLU or absolute value activation is at least* $\max_{n \in [N-1]} |\mu_n|$.

When $L{=}2$, the next result gives a solution to the min-norm problem. For $i{\in}[N]$, let $(z_+)_i$ and $(z_-)_i$ be the Lasso variable corresponding to the features $\mathrm{ReLU}^+_{x_i}$ and $\mathrm{ReLU}^-_{x_i}$, respectively. In other words, $\mathbf{z}_+$ corresponds to $\mathbf{A}_+$, and $\mathbf{z}_-$ corresponds to $\mathbf{A}_-$ as defined in Corollary 3.15.

Lemma C.6. *The min-norm problem* (C.1) *for* $L{=}2, \sigma{=}ReLU$ *has optimal value* $\|\mathbf{z}^*\|_1{=}\sum_{n=1}^{N-1}|\mu_n{-}\mu_{n+1}|$. *An optimal solution is* $(z_+)_{n+1}{=}\mu_n{-}\mu_{n+1}$ *for* $n{\in}[N{-}1]$, $\mathbf{z}_-{=}\mathbf{0}$, *and* $\xi{=}y_N$.

Lemma C.7. *For a 3-layer symmetrized ReLU network and training data as shown in Figure 3.2, the optimal value* $\|\mathbf{z}^*\|_1$ *of the minimum norm problem* (C.1) *is at least* 1.

Appendix D. Numerical results. The following simulations support our theoretical results.

In Figure 3.2, a deep narrow, absolute value network is trained. In Figure 3.5, a 3-layer symmetrized ReLU network is trained. Both neural nets have standard architecture, $m_L{=}100$, and are trained with Adam with the non-convex problem. The learning rate is $5(10^{-3})$, weight decay is 10^{-4}, and $\beta{=}10^{-7}{\approx}0$. When $\beta{\to}0$, the Lasso problem approaches the minimum norm problem (C.1). For each $L{\in}\{3,4,5\}$, the neural net reconstructed from the single feature in the left plot with corresponding Lasso parameter $z_i^* = 1$ and $\xi^* = 0$ is optimal in the minimum norm problem (C.1) by Lemma C.5 and Lemma C.7. This network is used to initialize a subset of the neurons in the non-convex training. All other weights are initialized randomly according to Pytorch defaults. The figures show that the networks trained with the non-convex problem closely match the Lasso solutions. The networks exhibit breakpoints at data points and their reflections not in the training data. The standard architecture in the non-convex model shows the applicability of the Lasso formulation. SGD gives similar results as Adam.

Supplementary Material.

Definitions and preliminaries.

D.1. Activation function. A function f is *bounded* if there is $M \geq 0$ with $|f(x)| \leq M$ for all x. If $\sigma(x)$ is piecewise linear around zero, $\sigma(x)$ is bounded if and only if $a^- = a^+ = 0$, e.g. $\sigma(x)$ is a threshold or sign activation. We call f *symmetric* if it is an even or odd function, for example absolute value. The activation $\sigma(x)$ is defined to be *homogeneous* if for any $a \geq 0$, $\sigma(ax) = a\sigma(x)$. Homogeneous activations include ReLU, leaky ReLU, and absolute value, and don't have amplitude parameters. We say $\sigma(x)$ is *sign-determined* if its value depends only on the sign of its input and not its magnitude. Threshold and sign activations are sign-determined.

D.2. Effective depth.

Remark D.1. *Plugging* (2.3) *into itself shows that in parallel network, for* $l \in [L-2]$,

$$\hat{\mathbf{X}}^{(i,l+2)} = \sigma_{\mathbf{s}^{(i,l+1)}}\left(\sigma\left(\hat{\mathbf{X}}^{(i,l)}\mathbf{W}^{(i,l+1)} + \mathbf{b}^{(i,l)}\right)\mathbf{s}^{(i,l)}\mathbf{W}^{(i,l+1)} + \mathbf{b}^{(i,l+1)}\right).$$

The inner parameters *of a parallel network are* $\mathbf{s}^{(i,l)}$ *for* $l \leq L-2$ *and* $\mathbf{W}^{(i,l)}$ *for* $l \leq L-1$. *Suppose* σ *is sign-determined. Then* $f_L(\mathbf{X};\theta)$ *is invariant to the value of the inner parameters, so they would be driven to* 0 *by weight regularization. We define the minimum value in* (1.1) *as an infimum which is approached as the norms of the inner parameters approach* 0. *Therefore the inner parameters are not regularized (the effective depth is* 2*), and we optimize for their directions rather than their magnitudes.*

Parallel networks with data dimension $d \geq 1$.

For convenience, we will omit ξ when writing $f_L(\mathbf{X};\theta)$. This does not change the training problem because we can write (1.1) as $\min_{\theta - \{\xi\}} \frac{\tilde{\beta}}{\tilde{L}}\|\theta_w\|_2^2 + \min_{\xi}\{\mathcal{L}_{\mathbf{y}}(f_L(X) - \xi\mathbf{1} + \xi\mathbf{1})\}$ and apply the change of variables/functions $\theta' = \theta - \{\xi\}$, $f_L(\mathbf{X};\theta)' = f_L(\mathbf{X};\theta) - \xi\mathbf{1}$ and $\mathcal{L}_{\mathbf{y}}(\mathbf{z})' = \min_{\xi} L_y(\mathbf{z} + \xi\mathbf{1})$. The loss function absorbs ξ while preserving its convexity. We begin by assuming the data is d-dimensional and consider the general *training problem*

$$\min_{\theta \in \Theta} \mathcal{L}_{\mathbf{y}}(f_L(\mathbf{X};\theta)) + \frac{\beta}{\tilde{L}} r(\theta)$$

with a general regularization of the form $r(\theta) = \sum_{i=1}^{m_L} \sum_{\theta^{(i,l)} \in \theta_w^{(i)}} \left(r^{(i,l)}\left(\theta^{(i,l)}\right)\right)^{\tilde{L}}$, where $\theta^{(i,l)}$ is a parameter such as $\mathbf{W}^{(i,l)}$ and $r^{(i,l)}$ is a regularization function such as $r^{(i,l)}\left(\mathbf{W}^{(i,l)}\right) = \left\|\mathbf{W}^{(i,l)}\right\|_p$. Assume $r^{(i,l)}$ is nonnegative and positively homogeneous, i.e., for any $\alpha \geq 0$, $r^{(i,l)}(\alpha\mathbf{W}) = \alpha r^{(i,l)}(\mathbf{W}) \geq 0$. The training problem (1.1) is a special case of the general training problem.

Definition D.2. *A* simplified *neural network with sign-determined activation is*

$$
\begin{aligned}
f_L\left(\mathbf{x};\theta\right) &= \xi + \sum_{i=1}^{m_L} \hat{\mathbf{X}}^{(i,L)} \alpha_i s^{(i,L-1)} \\
\hat{\mathbf{X}}^{(i,l+1)} &= \sigma\left(\hat{\mathbf{X}}^{(i,l)} \mathbf{W}^{(i,l)} + \mathbf{b}^{(i,l)}\right), l \in [L-1] \\
\theta_w^{(i)} &= \left\{\alpha_i, s^{(i,L-1)}\right\}, \theta_b^{(i)} = \left\{\mathbf{b}^{(i,l)}, \mathbf{W}^{(i,l)} : l \in [L-1]\right\} \text{ for } i \in [m_L]
\end{aligned} \tag{D.1}
$$

for a parallel network.

In other words, a simplified network has amplitude parameters only in the last layer.

Lemma D.3. *The simplified neural network is equivalent to the original neural network.*

Proof. By Remark D.1, it suffices to only regularize the outermost $\tilde{L}$=2 layers. For parallel networks, apply a change of variables $\mathbf{W}^{(i,l)'} = \mathbf{s}^{(i,l-1)}\mathbf{W}^{(i,l)}$ for $2 \leq l \leq L-1$. This removes $\mathbf{s}^{(i,l)}$ from θ for $l \leq L-2$. ∎

Let $\mathcal{D} = \{L-\tilde{L}+1, \cdots, L\}$. By Lemma D.3, the training problem's regularization is

$$
r(\theta) = \sum_{i=1}^{m_L} \sum_{l \in \mathcal{D}} \left(r^{(i,l)}\left(\theta_w^{(i,l)}\right)\right)^{\tilde{L}} \tag{D.2}
$$

where $\theta^{(i,L)} = \alpha_i$ and $\theta^{(i,l)} = \mathbf{W}^{(i,l)}$ for $l < L$ in parallel networks with homogeneous σ, $\theta^{(i,L)} = \alpha_i$ and $\theta^{(i,L-1)} = s^{(i,L-1)}$ in parallel networks with sign-determined σ. Similar to the parallel network in Subsection 2.2, extend the standard (2.2) network definitions row-wise to the cases where the input is $\mathbf{X} \in \mathbb{R}^{N \times d}$.

Lemma D.4. *Let $\tilde{\mathbf{X}}^{(i)} \in \mathbb{R}^N$ be $\hat{\mathbf{X}}^{(i,L)}$ for a parallel network, where the input is $\mathbf{X} \in \mathbb{R}^{N \times d}$. The training problem is equivalent to*

$$
\min_{\theta \in \Theta : r_{(i,l)}\left(\theta^{(i,l)}\right) = 1, l \in \mathcal{D} - \{L\}} \mathcal{L}_{\mathbf{y}}\left(\sum_{i=1}^{m_L} \alpha_i \tilde{\mathbf{X}}^{(i)}\right) + \beta \sum_{i=1}^{m_L} r_{(i,L)}\left(\alpha_i\right) \tag{D.3}
$$

Proof. By the AM-GM inequality on (D.2), a lower bound on the training problem is

$$
\min_{\theta \in \Theta} \mathcal{L}_y\left(f_L\left(\mathbf{X};\theta\right)\right) + \beta \sum_{i=1}^{m_L} \prod_{l \in \mathcal{D}} r_{(i,l)}\left(\theta^{(i,l)}\right). \tag{D.4}
$$

Consider the minimization problem

$$
\min_{\theta \in \Theta : r_{(i,l)}\left(\theta^{(i,l)}\right) = 1, l \in \mathcal{D} - \{L\}} \mathcal{L}_y\left(f_L\left(\mathbf{X};\theta\right)\right) + \beta \sum_{i=1}^{m_L} \prod_{l \in \mathcal{D}} r_{(i,l)}\left(\theta^{(i,l)}\right) \tag{D.5}
$$

Problem (D.5) is an upper bound on (D.4). Given optimal $\left\{\theta^{(i,l)}\right\}$ in (D.4), the rescaled parameters $\theta^{(i,l)'} = \theta^{(i,l)} / r_{(i,l)}\left(\theta^{(i,l)}\right)$ for $l \in \mathcal{D} - \{L\}$ and $\theta^{(i,L)'} = \theta^{(i,L)} \prod_{l \in \mathcal{D}} r^{(i,l)}\left(\theta^{(i,l)}\right)$ (and rescaled

bias parameters) achieve the same objective in (D.5). Hence (D.5) and (D.4) are equivalent. Given optimal $\left\{\theta^{(i,l)}\right\}$ in (D.5), the rescaled parameters $\theta^{(i,l)'}=|\theta^{(i,L)}|^{\frac{1}{L}}\theta^{(i,l)}$ (and rescaled bias parameters) achieve the same objective in the training problem, which is therefore equivalent to (D.5). Simplifying (D.5) gives (D.3). ∎

Lemma D.4 applies to networks without any weight constraints, i.e., it excludes 3-layer symmetrized networks. A L-layer *symmetrized network* is a parallel network with homogeneous activation such that $m_{L-3}=1$ and the elements of $\mathbf{W}^{(i,l)}$ have the same magnitude for $l\in\{L-2,L-1\}$. In a symmetrized network, the last two layer's weights are vectors: $\mathbf{W}^{(i,L-2)}\in\mathbb{R}^{1\times m_{L-2}}$ and $\mathbf{W}^{(i,L-1)}\in\mathbb{R}^{m_{L-2}}$. The constraint on the weight magnitudes is encoded in Θ. A 3-layer symmetrized network and a deep narrow network are special cases of a symmetrized network.

Lemma D.5. *Let* $r^{(i,l)}\left(\mathbf{W}^{(i,l)}\right)=\left\|\mathbf{W}^{(i,l)}\right\|_2$ *for* $l\in\{L-2,L-1\}$. *The rescaled problem for a symmetrized network is equivalent to*

(D.6)

$$\min_{\theta\in\Theta:r^{(i,l)}\left(\mathbf{W}^{(i,l)}\right)=1 \text{ for } l\in[L-3];\left|\mathbf{W}_j^{(i,l)}\right|=1 \text{ for } l\in\{L-2,L-1\},j\in[m_{L-3}]} \mathcal{L}_{\mathbf{y}}\left(f_L\left(\mathbf{X};\theta\right)\right)+\tilde{\beta}\sum_{i=1}^{m_L} r_{(i,L)}\left(\alpha_i\right)$$

where $\tilde{\beta}=\frac{\beta}{m_{L-2}}$.

Proof. Since $m_{L-3}=1$, we have $\mathbf{W}^{(i,L-2)}\in\mathbb{R}^{1\times m_{L-2}}$ and $\mathbf{W}^{(i,L-1)}\in\mathbb{R}^{m_{L-2}}$. The constraint states that for $l\in\{L-2,L-1\}$, $\left|\mathbf{W}_1^{(i,l)}\right|=\cdots=\left|\mathbf{W}_{m_{L-2}}^{(i,l)}\right|$. For $l\in\{L-2,L-1\}$, given $\mathbf{W}^{(i,l)},\mathbf{b}^{(i,l)}$ and α_i apply a change of variables $\mathbf{W}^{(i,l)'}=\sqrt{m_{L-2}}\mathbf{W}^{(i,l)}$ and $\mathbf{b}^{(i,l)'}=\sqrt{m_{L-2}}^{l+3-L}\mathbf{b}^{(i,l)}$ and $\alpha_i'=\frac{1}{m_{L-2}}\alpha_i$ to the parameters in (D.3) to arrive at (D.6). ∎

Henceforth, assume $r_{(i,L)}\left(\alpha_i\right)=|\alpha_i|$.

Definition D.6. *Define the* rescaled training problem *as*

$$\min_{\theta\in\Theta}\mathcal{L}_{\mathbf{y}}\left(\sum_{i=1}^{m_L}\alpha_i\tilde{\mathbf{X}}^{(i)}\right)+\tilde{\beta}\sum_{i=1}^{m_L}|\alpha_i|. \tag{D.7}$$

If the network is symmetrized, $\tilde{\beta}=\frac{\beta}{m_{L-2}}$ *and* Θ *includes the constraints that* $r^{(i,l)}\left(\mathbf{W}^{(i,l)}\right)=1$ *for* $l\in[L-3]$ *and* $\left|\mathbf{W}_j^{(i,l)}\right|=1$ *for* $l\in\{L-2,L-1\}$. *Otherwise,* $\tilde{\beta}=\beta$ *and* $r_{(i,l)}\left(\theta^{(i,l)}\right)=1$ *for* $l\in\mathcal{D}-\{L\}$. $\tilde{\mathbf{X}}^{(i)}$ *is defined as in Lemma D.4.*

For 3-layer networks with l_2 regularization ($r^{(i,l)}\left(\mathbf{W}^{(i,l)}\right)=\|\mathbf{W}^{(i,l)}\|_2$), Θ constrains the absolute value of all elements of all inner layer weights to be 1 in the rescaled training problem. The rescaled training problem is equivalent to both symmetrized and non-symmetrized networks.

Lemma D.7. *The rescaled training problem* (D.7) *is equivalent to the training problem for all the architectures and activations discussed above.*

Proof. Follows from Lemma D.4 and Lemma D.5. ∎

Lemma D.8. *A lower bound on the rescaled training problem is the dual problem*

$$\max_{\lambda\in\mathbb{R}^N} -\mathcal{L}_{\mathbf{y}}^*(\lambda) \quad s.t. \quad \max_{\theta\in\Theta}\left|\lambda^T\tilde{\mathbf{X}}\right| \leq \tilde{\beta}, \tag{D.8}$$

where $\tilde{\mathbf{X}}=\tilde{\mathbf{X}}^{(1)}$ *and* $f^*(\mathbf{x}):=\max_{\mathbf{x}}\left\{\mathbf{z}^T\mathbf{x}-f(\mathbf{x})\right\}$ *is the convex conjugate of* f.

Proof. Find the dual of (D.3), by rewriting (D.3) as

$$\min_{\theta\in\Theta}\mathcal{L}_{\mathbf{y}}(\mathbf{z})+\tilde{\beta}||\boldsymbol{\alpha}||_1, \quad \text{s.t.} \quad \mathbf{z}=\sum_{i=1}^{m_L}\alpha_i\tilde{\mathbf{X}}^{(i)}. \tag{D.9}$$

The Lagrangian of problem (D.9) is $L\left(\lambda,\theta\right)=\mathcal{L}_{\mathbf{y}}(\mathbf{z})+\tilde{\beta}||\boldsymbol{\alpha}||_1-\lambda^T\mathbf{z}+\sum_{i=1}^{m_L}\lambda^T\tilde{\mathbf{X}}^{(i)}\alpha_i$. Minimize the Lagrangian over $\mathbf{z}$ and $\boldsymbol{\alpha}$ and use Fenchel duality (6). The dual of (D.9) is

$$\max_{\lambda\in\mathbb{R}^N} -\mathcal{L}_{\mathbf{y}}^*(\lambda) \quad \text{s.t.} \quad \max_{\theta\in\Theta}\left|\lambda^T\tilde{\mathbf{X}}^{(i)}\right| \leq \tilde{\beta}, i\in[m_L]. \tag{D.10}$$

In the tree and parallel nets, $\tilde{\mathbf{X}}^{(i)}$ is of the same form for all $i\in[m_L]$. So the m_L constraints in (D.10) collapse to a single constraint. Then we can write (D.10) as (D.8). ∎

For a parallel network, the dual problem (D.8) is

$$\max_{\lambda\in\mathbb{R}^N} -\mathcal{L}_{\mathbf{y}}^*(\lambda) \quad \text{s.t.} \quad \max_{\theta\in\Theta}\left|\lambda^T\hat{\mathbf{X}}^{(L)}\right| \leq \tilde{\beta}, \tag{D.11}$$

where $\hat{\mathbf{X}}^{(1)}$=$\mathbf{X}$. Henceforth, all regularizations are l_2-norm: e.g., for a parallel network, $r^{(i,l)}\left(\mathbf{W}^{(i,l)}\right)=\left\|\mathbf{W}^{(i,l)}\right\|_2$, denoting the square root of sum of squares of $\mathbf{W}^{(i,l)}$'s elements.

Deep narrow and symmetrized networks with d=1. In this section, we assume the data is 1-D and find the maximizers of $\left|\lambda^T\hat{\mathbf{X}}^{(L)}\right|$ in the dual constraint (D.11). To this end, assume the elements of $\mathbf{W}^{(l)}$ are ±1. Note that $\mathbf{b}^{(L-1)}$ is a scalar and $\mathbf{X}^{(l+1)}$=$\sigma\left(\mathbf{X}^{(l)}\mathbf{W}^{(l)}+\mathbf{b}^{(l)}\mathbf{1}\right)\in\mathbb{R}^N$. The next remark refers to the leaky ReLU slopes a^+ and a^- defined in Section 2. Let $\mathcal{K}(f)$ and $\mathcal{Z}(f)$ be the sets of breakpoints and zeros of a function f, respectively.

Remark D.9. *Let* b=$\mathbf{b}^{(L-1)}$, a_n=$\hat{\mathbf{X}}_n^{(L-1)}\mathbf{W}^{(L-1)}$, $g_n(b)$=$\sigma\left(a_n+b\right)$, *and* $g(b)$=$\sum_{n=1}^N\lambda_n g_n(b)$ =$\lambda^T\mathbf{X}^{(L)}$. *Let* $\mathcal{I}$=$\bigcup_{n=1}^N\mathcal{K}\left(g_n\right)\supset\mathcal{K}(g)$. *Then* $g(b)$=$\sum_{n=1}^N\lambda_n a(a_n+b)$=$ab\sum_{n=1}^N\lambda_n$+ $\sum_{n=1}^N\lambda_n a_n a$ *for* b *large enough (with* a=a^+*) and for* b *small enough (with* a=a^-*). So* a^-=a^+=0 *or* $\sum_{n=1}^N\lambda_n$=0 *if and only if* g *is bounded, if and only if* g *has a (finite) maximizer and minimizer. In this case, assuming* g *is not a constant function,* $\mathcal{I}$ *contains a maximizer and minimizer of* g.

Henceforth, assume $\lambda^T\mathbf{1}$=0 if $a^-\neq0$ or $a^+\neq0$. Suppose $\mathbf{b}^{(l)},\cdots,\mathbf{b}^{(L-1)}$ are scalar-valued. We call $\mathbf{b}^{(l)}$=$\mathbf{b}^{(l)^*}$ *optimal* if $\mathbf{b}^{(l)^*}\in\arg\max_{\mathbf{b}^{(l)}}\max_{\mathbf{b}^{(l+1)}}\cdots\max_{\mathbf{b}^{(L-1)}}\left|\sum_{n=1}^N\lambda_n\hat{\mathbf{X}}_n^{(L)}\right|$. The next result refers to the normalized midpoint defined in (3.1).

Lemma D.10. *Let* σ *be leaky ReLU. Let* $\alpha,\beta\in\mathbb{R}$ *with* $\alpha\neq\beta$. *Let* $f(x)$=$\sigma\left(x+\alpha\right)-\sigma\left(x+\beta\right)$. *For* $s\in\{-,+\}$, *let* $g_s(x)$=$\sigma(sx)$. *Then, if* σ *is monotone,*

$$\mathcal{K}\left(g_s(f)\right)\subset\{-\alpha,-\beta\}. \tag{D.12}$$

Otherwise if σ is not monotone,

$$\mathcal{K}(g_s(f)) \subset \{-\alpha, -\beta, m_{\alpha,\beta}\}. \tag{D.13}$$

Proof. Since $f_{\alpha,\beta}$ is piecewise linear,

$$\mathcal{K}\left(g_s(f)\right) \subset \mathcal{K}(f) \cup \mathcal{Z}(f). \tag{D.14}$$

Moreover if $f(x)$ has the same sign for all x, then observe that

$$\mathcal{K}\left(g_s(f)\right) = \mathcal{K}(f). \tag{D.15}$$

Let $\mathcal{G}(f) = \{(x, f(x)) : x \in \mathcal{K}(f)\}$. We find that

$$\mathcal{G}(f) = \left\{\left(-\max\{\alpha,\beta\}, a^-(\alpha-\beta)\right), \left(-\min\{\alpha,\beta\}, a^+(\alpha-\beta)\right)\right\}. \tag{D.16}$$

In particular, $\mathcal{K}(f)=\{-\alpha,-\beta\}$. Observe that f has constant value (an in particular, sign) beyond its breakpoints. If σ is monotone, then $\text{sign}(a^+)=\text{sign}(a^-)$, so (D.16) implies $\text{sign}\left(f(x)\right)=\text{sign}\left(a^+(\alpha-\beta)\right)$ for all x, so (D.15) implies (D.12). If σ is not monotone, linearly interpolating between the points in $\mathcal{G}(f)$ (D.16) shows that f changes sign exactly when $x=m_{\alpha,\beta}$, so (D.14) implies (D.13). ∎

Lemma D.11. *Let σ be piecewise linear if $L=2$ and leaky ReLU otherwise. Let $l\in\{L-1, L-2\}$. Consider the bias $\mathbf{b}^{(l)}$. Let $m_{L-2}=1$. If σ is monotone there is either a data feature bias that is optimal. If σ is not monotone there is either a data feature bias or a midpoint feature bias $b^{(L-2)}=m_{\mathbf{X}^{(L-2)}_{n^{(L-2)}},\mathbf{X}^{(L-2)}_{n^{(L-1)}}}\mathbf{W}^{(L-2)}$ (if $l=L-2$) that is optimal.*

Proof. Remark D.9 implies that $\bigcup_{n=1}^N \mathcal{K}\left(\hat{\mathbf{X}}^{(L)}_n\right)$ contains an optimal $\mathbf{b}^{(L-1)}$. The breakpoint of $\hat{\mathbf{X}}^{(L)}_n=\sigma\left(\mathbf{X}^{(L-1)}_n\mathbf{W}^{(L-1)}+\mathbf{b}^{(L-1)}\right)$ as a function of $\mathbf{b}^{(L-1)}$ is $\mathbf{b}^{(L-1)}=-\mathbf{X}^{(L-1)}_n\mathbf{W}^{(L-1)}$. Therefore for some $n^{(L-1)}\in[N]$, the data feature bias $\mathbf{b}^{(L-1)}=-\mathbf{X}^{(L-1)}_{n^{(L-1)}}\mathbf{W}^{(L-1)}$ is optimal. Plugging in this optimal $\mathbf{b}^{(L-1)}$ back into $\hat{\mathbf{X}}^{(L)}_n$ gives

$$\lambda^T\mathbf{X}^{(L)}=\sum_{n=1}^N \lambda_n\sigma\left(\left(\hat{\mathbf{X}}^{(L-1)}_n-\hat{\mathbf{X}}^{(L-1)}_{n^{(L-1)}}\right)\mathbf{W}^{(L-1)}\right).$$

For $L>2$, expanding $\hat{\mathbf{X}}^{(L-1)}$ gives

$$\lambda^T\mathbf{X}^{(L)}=\sum_{n=1}^N \lambda_n\sigma\left(\left(\sigma\left(\hat{\mathbf{X}}^{(L-2)}_n\mathbf{W}^{(L-2)}+\mathbf{b}^{(L-2)}\right)-\sigma\left(\hat{\mathbf{X}}^{(L-2)}_{n^{(L-1)}}\mathbf{W}^{(L-2)}+\mathbf{b}^{(L-2)}\right)\right)\mathbf{W}^{(L-1)}\right). \tag{D.17}$$

Since $m_{L-2}=1$, we have $\mathbf{b}^{(L-2)}\in\mathbb{R}$. Applying Lemma D.10 with $x=\mathbf{b}^{(L-2)}$, $\alpha=\hat{\mathbf{X}}^{(L-2)}_n\mathbf{W}^{(L-2)}$, $\beta=\hat{\mathbf{X}}^{(L-1)}_n\mathbf{W}^{(L-2)}$, $s=\mathbf{W}^{(L-1)}$ and a similar argument as Remark D.9 gives the result. ∎

Lemma D.12. *Let $L\geq 2$. Consider a deep narrow ReLU network with L layers. For every $l\in\{2,\cdots,L\}$, a data feature bias is optimal for layer $L-l+1$, and with this optimal bias, if $l<L$, there exists $N_1, N_2\in[N]$ such that for all $n\in[N]$, either $\hat{\mathbf{X}}^{(L)}_n=0$ or*

(D.18) $$\hat{\mathbf{X}}_n^{(L)} = \pm\left(\sigma\left(\hat{\mathbf{X}}_{N_1}^{(L-l)}\mathbf{W}^{(L-l)}+\mathbf{b}^{(L-l)}\right)-\sigma\left(\hat{\mathbf{X}}_{N_2}^{(L-l)}+\mathbf{b}^{(L-l)}\right)\right).$$

Proof. We prove by induction on l.

Base case: suppose $l=2$. Then the claim holds by (D.17) in the proof of Lemma D.11.

Now, suppose the claim holds for $l{=}k{\in}\{2,\cdots,L{-}1\}$. Applying argument similar to the proof of Lemma D.11 to (D.18) shows that a data feature bias $\mathbf{b}^{(L-k)}{=}{-}\hat{\mathbf{X}}_{n^{(L-k)}}^{(L-k)}\mathbf{W}^{(L-k)}$ is optimal for $\mathbf{b}^{(L-l+1)}$ when $l{=}k{+}1$. Plugging in this optimal $\mathbf{b}^{(L-k)}$ into $\hat{\mathbf{X}}^{(L)}$ in (D.18) for $l{=}k$ and expanding $\hat{\mathbf{X}}^{(L-k)}{=}\sigma\left(\hat{\mathbf{X}}^{(L-k-1)}\mathbf{W}^{(L-k-1)}{+}\mathbf{b}^{(L-k-1)}\right)$ shows that for some $s^{(1)},s^{(2)}{\in}\{-1,1\}$, either $\hat{\mathbf{X}}_n^{(L)}{=}0$ or

$$\begin{aligned}\hat{\mathbf{X}}_n^{(L)} =&\sigma\Big(s^{(1)}\Big(\sigma\Big(s^{(2)}\Big(\sigma\Big(\hat{\mathbf{X}}_{n'}^{(L-k-1)}\mathbf{W}^{(L-k-1)}{+}\mathbf{b}^{(L-k-1)}\Big)-\sigma\Big(\hat{\mathbf{X}}_{n^{(L-2)}}^{(L-k-1)}\mathbf{W}^{(L-3)}{+}\mathbf{b}^{(L-k-1)}\Big)\Big)\Big)\\ &-\sigma\Big(s^{(2)}\Big(\sigma\Big(\hat{\mathbf{X}}_{n^{(L-1)}}^{(L-k-1)}\mathbf{W}^{(L-k-1)}{+}\mathbf{b}^{(L-k-1)}\Big)-\sigma\Big(\hat{\mathbf{X}}_{n^{(L-2)}}^{(L-k-1)}\mathbf{W}^{(L-k-1)}{+}\mathbf{b}^{(L-k-1)}\Big)\Big)\Big)\Big)\Big)\\ \in&\Big\{s^{(2)}\Big(\sigma\Big(\hat{\mathbf{X}}_{n'}^{(L-k-1)}\mathbf{W}^{(L-k-1)}+\mathbf{b}^{(L-k-1)}\Big)-\sigma\Big(\hat{\mathbf{X}}_{n^{(L-2)}}^{(L-k-1)}+\mathbf{b}^{(L-k-1)}\Big)\Big),\\ &-s^{(2)}\Big(\sigma\Big(\hat{\mathbf{X}}_{n^{(L-1)}}^{(L-k-1)}\mathbf{W}^{(L-k-1)}+\mathbf{b}^{(L-k-1)}\Big)-\sigma\Big(\hat{\mathbf{X}}_{n^{(L-2)}}^{(L-k-1)}+\mathbf{b}^{(L-k-1)}\Big)\Big),\\ &s^{(2)}\Big(\sigma\Big(\hat{\mathbf{X}}_{n'}^{(L-k-1)}\mathbf{W}^{(L-k-1)}+\mathbf{b}^{(L-k-1)}\Big)-\sigma\Big(\hat{\mathbf{X}}_{n^{(L-1)}}^{(L-k-1)}+\mathbf{b}^{(L-k-1)}\Big)\Big),0\Big\}.\end{aligned}$$

So (D.18) holds for $l{=}k{+}1$. Finally if (D.18) holds for $l{=}L{-}1$ then by a similar argument as the proof of Lemma D.11, a data feature bias for $\mathbf{b}^{(L-l+1)}$ is optimal when $l{=}k{+}1{=}L$. By induction, the result holds. ∎

Lemma D.13. *Let $L\geq 2$. Consider a deep narrow ReLU network with L layers. For $l\in[L-1]$, suppose $\mathbf{b}^{(l)}$ is a data feature. For every $l{\in}\{2,\cdots,L{-}1\}$, there exist $N_1,N_2{\in}[N]$ such that for all x,*

(D.19) $$\hat{\mathbf{X}}^{(l)}(x)\in\left\{ReLU^{\pm}_{x_{N_1}}(x),Ramp^{\pm}_{x_{N_1},x_{N_2}}(x)\right\}.$$

Proof. We prove by induction on l. Base case: for $l=2$, $\hat{\mathbf{X}}^{(2)}(x){=}\sigma\left(x\mathbf{W}^{(1)}{+}\mathbf{b}^{(1)}\right){=}\sigma\left((x{-}x_{n^{(1)}})\,\mathbf{W}^{(1)}\right){=}\mathrm{ReLU}^{\mathbf{W}^{(1)}}_{x_{n^{(1)}}}(x)$. Now, suppose the claim holds for $l=1,\cdots,k{<}L{-}1$. Then $\hat{\mathbf{X}}^{(k+1)}(x){=}\sigma\left(\hat{\mathbf{X}}^{(k)}\mathbf{W}^{(k)}{+}\mathbf{b}^{(k)}\right){=}\sigma\left(\left(\hat{\mathbf{X}}_n^{(k)}{-}\hat{\mathbf{X}}_{n^{(k)}}^{(k)}\right)\mathbf{W}^{(k)}\right)$ so either there exist $N_1',N_2'{\in}\left\{N_1,n^{(k)}\right\},s{\in}\{+,-\}$ such that for all x, $\hat{\mathbf{X}}^{(k+1)}(x){=}\sigma\left(\left(\mathrm{ReLU}^{\pm}_{x_{N_1}}(x){-}\mathrm{ReLU}^{\pm}_{x_{N_1}}(x_{n^{(k)}})\right)\mathbf{W}^{(k)}\right){\in}\left\{\mathrm{ReLU}^{s}_{x_{N_1'}}(x),\mathrm{Ramp}^{s}_{x_{N_1'},x_{N_1'}}(x)\right\}$, or there exist $N_1',N_2'\in\left\{N_1,N_2,n^{(k)}\right\},s\in\{+,-\}$ such that for all x, $\hat{\mathbf{X}}^{(k+1)}(x){=}\sigma\left(\left(\mathrm{Ramp}^{\pm}_{x_{N_1},x_{N_2}}(x){-}\mathrm{Ramp}^{\pm}_{x_{N_1},x_{N_2}}(x_{n^{(k)}})\right)\mathbf{W}^{(k)}\right){=}\mathrm{Ramp}^{s}_{x_{N_1'},x_{N_2'}}(x)$. By induction, the result holds. ∎

Lemma D.14. *Consider a deep narrow network. Let $L\geq 2$. For all $l\in[L-1]$, there exist $n^{(L-l)}\in[N]$ and a l-tuple $(a_0,a_1,\cdots,a_l)$ such that for all $l\in[L-1]$,*

(D.20) $$\mathbf{b}^{(L-l)^*} = -\sum_{i=1}^{l} a_i \hat{\mathbf{X}}^{(L-l)}_{n^{(L-i)}} \mathbf{W}^{(L-l)}$$

is optimal. Note that optimal $\mathbf{b}^{(1)}, \cdots, \mathbf{b}^{(L-1)}$ *can be found sequentially, by computing* $\mathbf{b}^{(l)^*}$ *as a function of* $\mathbf{b}^{(l-1)^*}$ *and so on. Moreover, there are* $O\left(2^L L!\right)$ *options for* $\mathbf{b}^{(1)^*}$*. Additionally, for all* $l \in [L-1]$*, under such optimal* $\mathbf{b}^{(1)}, \cdots, \mathbf{b}^{(l-1)}$*, for all* $n \in [N]$*,*

(D.21) $$\hat{\mathbf{X}}^{(L-l+1)}_n = \sigma\left(\hat{\mathbf{X}}^{(L-l)'}_n \mathbf{W}^{(L-l)}\right)$$

where

(D.22) $$\hat{\mathbf{X}}^{(L-l)'}_n = a_0 \hat{\mathbf{X}}^{(L-l)}_n + \sum_{i=1}^{l} a_i \hat{\mathbf{X}}^{(L-l)}_{n^{(L-i)}}.$$

Proof. We prove (D.20), (D.21) and (D.22) by strong induction on l.

Base case: suppose $l = 1$. Then (D.20), (D.21), (D.22) hold by Lemma D.11 and its proof.

Now, suppose (D.20), (D.21) and (D.22) hold for $l{=}1, \cdots, k{<}L{-}1$. By Equation (D.21) for $l{=}k$, $\hat{\mathbf{X}}^{(L-k+1)}_n(x)$ is locally a linear combination of the $k{+}1$ terms $\hat{\mathbf{X}}^{(L-k+1)}_{n^{(L-k-1)}}, \hat{\mathbf{X}}^{(L-k+1)}_{n^{(L-1)}}, \cdots, \hat{\mathbf{X}}^{(L-k+1)}_{n^{(L-k)}}$, and hence so is $\hat{\mathbf{X}}^{(L)}$. So the breakpoints of $\hat{\mathbf{X}}^{(L)}$ as a function of $\mathbf{b}^{(L-(k+1))}$ are linear combinations of the $k{+}1$ terms, which proves (D.20) for $l{=}k{+}1$. Plugging in this breakpoint $\mathbf{b}^{(L-k-1)}$ into $\hat{\mathbf{X}}^{(L-k)}_n$ proves (D.21) and (D.22) for $l{=}k{+}1$. Therefore (D.20), (D.21) and (D.22) hold for all $l \in [L{-}1]$.

Now we prove that the number of options for $\mathbf{b}^{(1)}$ is $O\left(2^L L!\right)$. By Equation (D.22) for $l{=}L{-}1$, as a function of $\mathbf{b}^{(1)}$, $\hat{\mathbf{X}}^{(L-1)'}_j$ has breakpoints at $-\hat{\mathbf{X}}^{(L-l-1)}_i \mathbf{W}^{(L-l-1)}$ for $i \in \{n, n^{(L-1)}, \cdots, n^{(L-l)}\}$, which totals to $l{+}1{=}L$ breakpoints. A piecewise linear function f has up to plus or minus one as many zeros as breakpoints, and the zeros become breakpoints in $\sigma(f)$. So by Equation (D.21), $\hat{\mathbf{X}}^{(L-l+1)}_n$ as a function of $\mathbf{b}^{(1)}$ has breakpoints at $O(2L)$ places. Then by Equation (D.22) for $l{=}L{-}2$, $\hat{\mathbf{X}}^{(L-l+1)'}_n$ as a function of $\mathbf{b}^{(1)}$ has breakpoints at $O(2(L{-}1)(L))$ places. And by Equation (D.21) for $l{=}L{-}2$, $\hat{\mathbf{X}}^{(L-k+2)}_n$ as a function of $\mathbf{b}^{(1)}$ has breakpoints at $O\left(2^2 L(L-1)\right)$ places. By repeating this argument for $l{=}L{-}1, \cdots, 1$, $\hat{\mathbf{X}}^{(L)}_n$ as a function of $\mathbf{b}^{(1)}$ has breakpoints at $O\left(2^L L!\right)$ places. ∎

Recall the deep library defined in Definition 3.10. Note that any $\mathbf{a}$ in the deep library is of the form $\mathbf{a} = \hat{\mathbf{X}}^{(L)}(\mathbf{X})$, with $\mathbf{a}_n = \hat{\mathbf{X}}^{(L)}(x_n)$.

Lemma D.15. *For* $L \in \{2, 3\}$ *the maximization constraint in* (D.11) *is equivalent to: for all vectors* $\mathbf{a}$ *in the deep library,*

(D.23) $$\left|\lambda^T \mathbf{a}\right| \le \tilde{\beta}$$
$$\mathbf{1}^T \lambda = 0 \text{ if } a^+ \neq 0 \text{ or } a^- \neq 0.$$

Proof. Lemma D.11 gives (D.23). Now, if the activation is symmetric, then $\hat{\mathbf{X}}^{(L)}$ is invariant under the sign of the components of $\mathbf{W}^{(1)}$. Next, recall $x_1 > \cdots > x_N$ and $\text{sign}(0) = 1$. If the activation is sign, then for all $n \in [N-1]$, with $\mathbf{W}^{(1)} = 1$ and $\mathbf{b}^{(1)} = -x_n$ we have $\hat{\mathbf{X}}^{(L=2)}(\mathbf{X}) = \sigma(\mathbf{X} - x_n) = \left(\mathbf{1}_{1:n}^T, -\mathbf{1}_{n+1:N}^T\right) = -\sigma(x_{n+1} - \mathbf{X})$; while for $\mathbf{W}^{(1)} = -1$ and $\mathbf{b}^{(1)} = x_{n+1}$ we have $\hat{\mathbf{X}}^{(L=2)}(\mathbf{X}) = \sigma(x_{n+1} - \mathbf{X})$. And for $\mathbf{W}^{(1)} = 1$ and $\mathbf{b}^{(1)} = -x_N$, we have $\hat{\mathbf{X}}^{(2)}(\mathbf{X}) = \mathbf{1}$ while for $\mathbf{W}^{(1)} = -1$ and $\mathbf{b}^{(1)} = x_1$ we have $\hat{\mathbf{X}}^{(2)}(\mathbf{X}) = \mathbf{1}$. So for $L = 2$ with symmetric or sign activation, (D.23) is unchanged if $\mathbf{W}^{(1)} \in \{-1, 1\}$ is restricted to be 1. ∎

Lemma D.16. *Let* $\mathbf{A}$ *be a matrix whose set of columns is the deep library. Replace the maximization constraint in* (D.11) *with* (D.23). *The dual of* (D.11) *then is*

$$\min_{\mathbf{z},\xi\in\mathbb{R}} \mathcal{L}_{\mathbf{y}}(\mathbf{A}\mathbf{z} + \xi\mathbf{1}) + \tilde{\beta}\|\mathbf{z}\|_1, \quad \textit{where} \quad \xi = 0 \textit{ if } c_1 = c_2 = 0. \tag{D.24}$$

Proof. Problem (D.11) can be written as

$$-\min_{\lambda\in\mathbb{R}^N} \mathcal{L}_{\mathbf{y}}^*(\lambda) \quad \text{s.t.} \quad \lambda^T\mathbf{1} = 0 \text{ if } a^- \neq 0 \text{ or } a^+ \neq 0, \text{ and } |\lambda^T\mathbf{A}| \leq \tilde{\beta}\mathbf{1}^T. \tag{D.25}$$

The Lagrangian of the negation of (D.25) with bidual variables $\mathbf{z}, \xi$ is

$$L(\lambda, \mathbf{z}, \xi) = \mathcal{L}_{\mathbf{y}}^*(\lambda) - \lambda^T(\mathbf{A}\mathbf{z} + \xi\mathbf{1}) - \tilde{\beta}\|\mathbf{z}\|_1, \text{ where } \xi = 0 \text{ if } a^- = a^+ = 0. \tag{D.26}$$

Equation (D.26) holds because the constraint $|\lambda^T\mathbf{A}| \leq \tilde{\beta}\mathbf{1}^T$ i.e., $\lambda^T\mathbf{A} - \tilde{\beta}\mathbf{1}^T \leq \mathbf{0}^T, -\lambda^T\mathbf{A} - \tilde{\beta}\mathbf{1}^T \leq \mathbf{0}^T$, appears in the Lagrangian as $\lambda^T\mathbf{A}\left(\mathbf{z^{(1)}} - \mathbf{z^{(2)}}\right) - \tilde{\beta}\mathbf{1}^T\left(\mathbf{z^{(1)}} + \mathbf{z^{(2)}}\right)$ with bidual variables $\mathbf{z^{(1)}}, \mathbf{z^{(2)}}$, which are combined into one bidual variable $\mathbf{z} = \mathbf{z^{(1)}} - \mathbf{z^{(2)}}$. This makes $\mathbf{z^{(1)}} + \mathbf{z^{(2)}} = \|\mathbf{z}\|_1$. Changing variables $\mathbf{z}' = -\mathbf{z}, \xi' = -\xi$ gives (D.26). Since $\mathcal{L}^{**} = \mathcal{L}$ (5), $\inf_\lambda L(\lambda, \mathbf{z}, \xi) = -\mathcal{L}_y - \tilde{\beta}\|\mathbf{z}\|_1$ and negating its maximization gives (D.24). ∎

Definition D.17 (Parameter unscaling). *For symmetrized networks, change* $\alpha_i' = 2\alpha_i$. *Let* $\gamma_i = |\alpha_i|^{\frac{1}{L}}$. *For ReLU, absolute value, or leaky ReLU, parameter unscaling is the following transformation. For symmetrized networks, first change variables as* $\mathbf{W}^{(i,l)'} = \frac{1}{\sqrt{2}}\mathbf{W}^{(i,l)}$ *and* $\mathbf{b}^{(i,l)'} = \frac{1}{\sqrt{2}^l}\mathbf{b}^{(i,l)}$ *for* $l \in [2]$. *Then, for all architectures, change variables as* $q' = \text{sign}(q)\gamma_i$ *for* $q \in \theta_w^{(i)}$. *For parallel networks, change variables as* $\mathbf{b}^{(i,l)'} = \mathbf{b}^{(i,l)}(\gamma_i)^l$. *For sign and threshold activations, parameter unscaling is the transformation* $\alpha_i' = \text{sign}(\alpha_i)\sqrt{|\alpha_i|}$ *and* $s^{(i,L-1)'} = \sqrt{|\alpha_i|}$ *for parallel nets.*

D.3. Proofs of results in Section 3 .

Proof of Lemma 3.7. The Lasso equivalence follows from a similar argument as the proof of Theorem 3.11. The Lasso features follow from Lemma D.12 and Lemma D.13 with $l = L$. ∎

Proof of Theorem 3.6. By the proof of Lemma D.14 and a similar argument as Theorem 3.11, the training problem is equivalent to a Lasso problem. Moreover Lemma D.14 and choice of $n^{(1)}, \cdots, n^{(L-1)} \in [N]$ give the number of possible $\mathbf{b}^{(1)}$ for given $\mathbf{W}^{(1)}, \cdots, \mathbf{W}^{(L-1)}$ as $O\left(N^{L-1}2^L L!\right)$. Note that by (D.21) and (D.22), $\mathbf{b}^{(1)}$ determines all other $\mathbf{b}^{(l)}$. Finally, there are 2^L possibilities of $\mathbf{W}^{(l)} \in \{-1, 1\}$. The ReLU result follows from Lemma 3.7. ∎

Proof of Theorem 3.11. By Lemma D.16, problem (D.24) is a lower bound on the training problem (1.1), where the Lasso features are all vectors in the deep library. Let $(\mathbf{z}^*, \xi^*)$ be a Lasso solution. From Definition 3.10, it can be seen that $\mathbf{A}\mathbf{z}^*+\xi^*=\sum_i z_i^*\mathbf{A}_i+\xi^*=\sum_i \alpha_i\hat{\mathbf{X}}^{(i,L)}(\mathbf{X})+\xi^* =f_L(\mathbf{X};\theta)$ and $\|\mathbf{z}^*\|_1=\|\boldsymbol{\alpha}\|_1$. Therefore a reconstructed neural net achieves the same objective in the rescaled training problem, as $(\mathbf{z}^*, \xi^*)$ does in the Lasso objective. Parameter unscaling (Definition D.17) makes them achieve the same objective in the training problem (see Remark D.20). Therefore the Lasso problem in Theorem 3.11 is equivalent to the training problem. ∎

Proof of Lemma B.4. By Theorem 3.11 and its proof, the reconstructed neural net achieves the same objective as the Lasso problem, which is equivalent to the training problem. So the reconstructed neural net is optimal. ∎

Proof of Theorem 3.13. By considering each component of $\mathbf{b}^{(l)}$ separately in the proof of Lemma D.11, we see that optimal bias parameters for a symmetrized network can include data features. The Lasso equivalence follows from a similar argument as the proof of Theorem 3.11. In the deep library, each $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}$ consists of 2 elements that are ± 1, and given $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}$, each of the 2 elements of $\mathbf{b}^{(1)}$ as well as the scalar $\mathbf{b}^{(2)}$ have N choices. So the number of possible features in the deep library is at most $2^{2(L-1)}N^3 = 2^4N^3$. ∎

Proof of Theorem 3.5. Follows from Theorem 3.13 and evaluating the features. See proof of Figure 3.4 below. ∎

Proof of Theorem 3.2. For $L\in\{2,3\}$, apply Theorem 3.6, Theorem 3.11 where $\sigma(x) = |x|$ and Lemma B.2. For $L=4$, continue the same line of argument as Lemma D.11 by plugging in $\mathbf{b}^{(L-2)}=-\hat{\mathbf{X}}^{(L-2)}_{n^{(L-2)}}\mathbf{W}^{(L-2)}$ into (D.17), expanding $\hat{\mathbf{X}}^{(L-2)}=\sigma\left(\hat{\mathbf{X}}^{(L-3)}\mathbf{W}^{(L-3)}+\mathbf{b}^{(L-3)}\right)$, and observing that the breakpoints of $\lambda^T\hat{\mathbf{X}}^{(L)}$ as a function of $\mathbf{b}^{(L-3)}$ include data features, and hence these can be optimal bias parameters. Plugging in these data feature biases in to $\hat{\mathbf{X}}^{(4)}$ gives the features and reflections. Continue repeating a similar argument for $L > 4$. The reconstruction and Lasso equivalence follows from a similar argument as the proof of Theorem 3.11. ∎

Remark D.18. *In the proof of Theorem* 3.2 *for $L \geq 4$, we only used one of the possible forms of an optimal $\mathbf{b}^{(L-2)}$ specified by Lemma* D.11 *for absolute value activation, and we only specified one of the possible forms of a breakpoint in $\lambda^T\hat{\mathbf{X}}^{(L)}$ as a function of $\mathbf{b}^{(L-3)}$. So a 4-layer network may have additional features not specified.*

Proof of Theorem 3.4. Follows from Lemma 3.7 and Theorem 3.11 where $\sigma(x) = (x)_+$ is monotone. ∎

Proof of Corollary 3.15. Follows from Theorem 3.11 for $L = 2$. ∎

Remark D.19. *For $b_1, b_2 \geq 0$,*

$$||x - b_1| - b_2| = \begin{cases} b_1 - b_2 - x & \text{if } x \leq b_1 - b_2 \\ x - (b_1 - b_2) & \text{if } b_1 - b_2 \leq x \leq b_1 \\ b_1 + b_2 - x & \text{if } b_1 \leq x \leq b_1 + b_2 \\ x - (b_1 + b_2) & \text{if } x \geq s_1 b_1 + b_2 \end{cases}$$
$$= \mathcal{W}_{b_1 - b_2, b_1}.$$

Proof of Lemma B.2, *Figure* 3.1, *Figure* 3.4. Follows from direct computation of Equation (2.2) and application of Remark D.19. In particular, for ReLU symmetrized networks, it can be verified that for $\mathbf{W}^{(1)} = (-1, 1), \mathbf{W}^{(2)} = (-1, -1)$; $\mathbf{W}^{(1)} = (-1, 1), \mathbf{W}^{(2)} = (1, 1)$; $\mathbf{W}^{(1)} = (1, -1), \mathbf{W}^{(2)} = (-1, -1)$; and $\mathbf{W}^{(1)} = (1, -1), \mathbf{W}^{(2)} = (1, 1)$, the deep library features can contain reflections. Note ReLU symmetrized features are consistent with Figure B.1. ∎

Remark D.20. *For sign-determined activations, by Remark* D.1*, the inner weights can be unregularized. So, reconstructed parameters (as defined in Definition* B.3*) that are unscaled according to Definition* D.17 *achieve the same objective in the training problem as the optimal value of the rescaled problem.*

Proof of Lemma 3.1. Since $|x| = 2(x)_+ + x$, we have $\sum_{i=1}^{m_2} \left|\mathbf{X}\mathbf{W}^{(i,1)} + \mathbf{b}^{(i,1)}\right| \alpha_i + \mathbf{X}\omega + \xi$ $= \sum_{i=1}^{m_2} 2\left(\mathbf{X}\mathbf{W}^{(i,1)} + \mathbf{b}^{(i,1)}\right)_+ \alpha_i + \mathbf{X}\left(\omega - \sum_{i=1}^{m_2} \mathbf{W}^{(i,1)}\alpha_i\right) - \sum_{i=1}^{m_2} \mathbf{b}^{(i,1)}\alpha_i + \xi$. Given a solution $\mathbf{W}^{(i,l)^*}, \mathbf{b}^{(i,l)^*}, \alpha_i{}^*, \xi^*, \omega^*$ to the training problem for absolute value activation with skip connection, the parameters $\mathbf{W}^{(i,l)} = \mathbf{W}^{(i,l)^*}, \mathbf{b}^{(i,l)^*} = \mathbf{b}^{(i,l)^*}, \alpha_i = 2\alpha_i{}^*, \xi = \xi^* - \sum_{i=1}^{m_2} \mathbf{b}^{(i,l)^*}\alpha_i{}^*, \omega = \omega^*$ $- \sum_{i=1}^{m_2} \mathbf{W}^{(i,1)^*}\alpha_i{}^*$ achieve the same objective in the training problem for ReLU activation with skip connection. Conversely, since $(x)_+ = \frac{|x|+x}{2}$, we have $\sum_{i=1}^{m_2} \left(\mathbf{X}\mathbf{W}^{(i,1)} + \mathbf{b}^{(i,1)}\right)_+ \alpha_i + \mathbf{X}\omega + \xi$ $= \frac{1}{2}\sum_{i=1}^{m_2} \left|\mathbf{X}\mathbf{W}^{(i,1)} + \mathbf{b}^{(i,1)}\right| \alpha_i + \mathbf{X}\left(\omega + \frac{1}{2}\sum_{i=1}^{m_2} \mathbf{W}^{(i,1)}\alpha_i\right) + \frac{1}{2}\sum_{i=1}^{m_2} \mathbf{b}^{(i,1)}\alpha_i + \xi$. Given a solution $\mathbf{W}^{(i,l)^*}, \mathbf{b}^{(i,l)^*}, \alpha_i{}^*, \xi^*, \omega^*$ to the training problem for ReLU activation with skip connection, the parameters $\mathbf{W}^{(i,l)} = \mathbf{W}^{(i,l)^*}, \mathbf{b}^{(i,l)^*} = \mathbf{b}^{(i,l)^*}, \alpha_i = \frac{1}{2}\alpha_i{}^*, \xi = \xi^* + \frac{1}{2}\sum_{i=1}^{m_2} \mathbf{b}^{(i,l)^*}\alpha_i{}^*$, $\omega = \omega^* + \frac{1}{2}\sum_{i=1}^{m_2} \mathbf{W}^{(i,1)^*}\alpha_i{}^*$ achieve the same objective in the training problem for absolute value activation with skip connection. Therefore the problems achieve the same optimal value and we have given a map between the solutions. ∎

Solution sets of Lasso under minimal regularization.

Remark D.21. *For each optimal $\mathbf{z}^*$ of the Lasso problem, minimizing the objective over ξ gives the optimal bias term as $\xi^* = \left(\mathbf{y} - \mathbf{1}\mathbf{1}^T\mathbf{y}\right) - \left(\mathbf{A} - \mathbf{1}\mathbf{1}^T\mathbf{A}\right)\mathbf{z}^*$.*

Remark D.22. *For a neural net with $L = 2$ layers and sign activation, by Theorem* 3.11 *the Lasso problem has an objective function $f(\mathbf{z}) = \frac{1}{2}\|\mathbf{A}\mathbf{z} - \mathbf{y}\|_2^2 + \beta\|\mathbf{z}\|_1$ where $\mathbf{A} \in \mathbb{R}^{N \times N}$. By Lemma* D.25*, $\mathbf{A}$ is full rank, which makes f strongly convex. Therefore the Lasso problem has a unique solution $\mathbf{z}^*$ (6). Moreover, for any Lasso problem, $\mathbf{z}^*$ satisfies the subgradient condition $\mathbf{0} \in \delta f(\mathbf{z}) = \mathbf{A}^T(\mathbf{A}\mathbf{z}^* - \mathbf{y}) + \beta\partial\|\mathbf{z}^*\|_1$. Equivalently,*

$$\frac{1}{\beta}\mathbf{A}_n^T(\mathbf{A}\mathbf{z}^* - \mathbf{y}) \in \begin{cases} \{-\text{sign}(\mathbf{z}_n^*)\} & \text{if } \mathbf{z}_n^* \neq 0 \\ [-1, 1] & \text{if } \mathbf{z}_n^* = 0 \end{cases}, \quad n \in [N].$$

D.4. Proofs of results in Appendix C. Let $\mathbf{e}^{(n)} \in \mathbb{R}^N$ be the n^{th} canonical basis vector, that is $\mathbf{e}_i^{(n)} = \mathbf{1}\{i = n\}$.

Proof of Proposition C.2. We analyze the solution set of $\mathbf{Az} + \xi\mathbf{1} = \mathbf{y}$. We note that $\left(I - \mathbf{1}\mathbf{1}^T/N\right)\mathbf{Az} = \left(I - \mathbf{1}\mathbf{1}^T/N\right)\mathbf{y}$.. As $\mathbf{z}^*$ is optimal in (C.1) , this implies that $(I - \mathbf{1}\mathbf{1}^T/N)\mathbf{Az}^* = (I - \mathbf{1}\mathbf{1}^T/N)\mathbf{y}$. This implies that $(I - \mathbf{1}\mathbf{1}^T/N)\mathbf{A}(\mathbf{z} - \mathbf{z}^*) = 0$. As $x_1 > x_2 > \cdots > x_N$, we have $\mathbf{A}\left(\mathbf{e}^{(1)} + \mathbf{e}^{(N)}\right) \propto \mathbf{1}$. As $\mathbf{A}$ is invertible by Lemma D.26 in Appendix D.5, this implies that there exists $t \in \mathbb{R}$ such that $\mathbf{z} - \mathbf{z}^* = t\left(\mathbf{e}^{(1)} + \mathbf{e}^{(N)}\right)$. It is impossible to have $z_1^* z_N^* > 0$ from the optimality of $\mathbf{z}^*$. Otherwise, by taking $t = -\text{sign}(z_1^*)\min\{|z_1^*|, |z_n^*|\}$, we have $\|\mathbf{z}\|_1 = \|\mathbf{z}^*\|_1 - 2\min\{|z_1^*|, |z_n^*|\} < \|\mathbf{z}^*\|_1$. Therefore, we have $z_1^* z_n^* \leq 0$. We can reparameterize $\mathbf{z} = \mathbf{z}^* + t\text{sign}(z_1^*)(\mathbf{e}^{(1)} + \mathbf{e}^{(N)})$. It is easy to verify that for t such that $-|z_1^*| \leq t \leq |z_n^*|$, we have $\|\mathbf{z}\|_1 = \|\mathbf{z}_n\|_1$, while for other choice of t, we have $\|\mathbf{z}\|_1 > \|\mathbf{z}_n\|_1$. Therefore, the solution set of (C.1) is given by (C.2). ∎

Proof of Proposition C.3. Follows from Remark D.22 describing the Lasso objective. ∎

Proof of Proposition C.4. By Lemma D.27, for $n \in [N-1]$, $z_{+n}^* = y_n - y_{n-1} \geq 0$ and $z_{+N}^* = y_N \geq 0$. So $\mathbf{z}^*$ achieves an objective value of $\|\mathbf{z}^*\|_1 = y_1$ in (C.1). Now let $\mathbf{z}$ be any solution to (C.1). Then $\mathbf{Az} = \mathbf{y}$. Since the first row of $\mathbf{A}$ is $[\mathbf{1}^T, \mathbf{0}^T]$, we have $y_1 = (\mathbf{Az})_1 = \mathbf{1}^T\mathbf{z}_+ \leq \|\mathbf{z}_+\| \leq \|\mathbf{z}\|_1 \leq \|\mathbf{z}^*\| = y_1$. So $\|\mathbf{z}_+\|_1 = \|\mathbf{z}\|_1 = y_1$, leaving $\mathbf{z}_- = \mathbf{0} = \mathbf{z}_-^*$. Therefore $\mathbf{z}_+ = \mathbf{A}^{-1}\mathbf{y} = \mathbf{z}_+^*$. Applying Lemma D.27 gives the result. ∎

Proof of Corollary C.1 . By Lemma D.25, Lemma D.26, Lemma D.27 and Lemma D.28, the dictionary matrix for the 2-layer net is full rank for sign, absolute value, threshold and ReLU activations. The dictionary matrices for deeper nets with sign activation are also full rank by Remark 3.12. Let $\mathbf{u} = \mathbf{A}^T(\mathbf{Az}^* - \mathbf{y})$. By Remark D.22, as $\beta \to 0$, we have $\mathbf{u} \to \mathbf{0}$, so $\mathbf{Az} - \mathbf{y} = (\mathbf{AA}^T)^{-1}\mathbf{Au} \to \mathbf{0}$. So as $\beta \to 0$, the optimal Lasso objective approaches 0, and by and Theorem 3.11, so does the training problem. So $f_L(\mathbf{X}; \theta) \xrightarrow{\beta \to 0} \mathbf{y}$. ∎

Proof of Lemma C.7. It can be verified that as shown in Figure 3.4, the Lasso features for a symmetrized network all have slope magnitude $0, 1$, or 2. However, only monotone features contain a segment with slope magnitude 2, and the training data (x_n, y_n) in Figure 3.5 is not monotone. There is a "left branch" consisting of $\{(x_5, y_5) = (-1, 1), (x_4, y_4) = (0, 0)\}$ and a "right branch" consisting of $\{(x_2, y_2) = (3, 1), (x_1, y_1) = (4, 2)\}$. Let $\mathbf{z}^*, \xi^*$ be a Lasso solution that fits the data exactly: $\mathbf{Az}^* + \xi^* = \mathbf{y}$. Let $\mathcal{I}^-$ ($\mathcal{I}^+$) be the set of indices i where the i^{th} feature is monotone and has negative (positive) slope over the left (right) branch. Let $\mathcal{I}^0$ be the set of indices corresponding to features that are not monotone. Let m^- and m^+ be the magnitude of the slopes of the left and right branch, respectively. Then $2\sum_{i\in\mathcal{I}^-}|z_i^*| + \sum_{i\in\mathcal{I}^0}|z_i^*| \geq m^-$ and $2\sum_{i\in\mathcal{I}^+}|z_i^*| + \sum_{i\in\mathcal{I}^0}|z_i^*| \geq m^+$. Note $\mathcal{I}^-, \mathcal{I}^+, \mathcal{I}^0$ are all pairwise disjoint. Therefore $\|\mathbf{z}^*\|_1 \geq \frac{m^+ + m^-}{2} = 1$. ∎

As seen from the proof, Lemma C.7 can be generalized to other training data examples.

Proof of Lemma C.5. Since ReLU and absolute value activations have slopes ± 1 or 0, and the weights of deep narrow networks are ± 1, the features $\hat{\mathbf{X}}^{(L)}(x)$ have slopes ± 1 or 0. Observe $\mathbf{y} = f_L(\mathbf{X}; \theta) = \xi^*\mathbf{1} + \mathbf{Az}^* = \xi^*\mathbf{1} + \sum_i z_i^*\mathbf{A}_i$. For any $n \in [N-1]$, $|\mu_n| = \left|\frac{f_L(\mathbf{X};\theta)_{n+1} - f_L(\mathbf{X};\theta)_n}{x_{n+1} - x_n}\right| = \left|\frac{\sum_i z_i^*(\mathbf{A}_{n+1,i} - \mathbf{A}_{n,i})}{x_{n+1} - x_n}\right| = \left|\sum_i z_i^* \frac{\hat{\mathbf{X}}^{(L)}(x_{n+1}) - \hat{\mathbf{X}}^{(L)}(x_n)}{x_{n+1} - x_n}\right|$

$\leq \sum_i |z_i^*| \left| \frac{\hat{\mathbf{X}}^{(L)}(x_{n+1}) - \hat{\mathbf{X}}^{(L)}(x_n)}{x_{n+1} - x_n} \right| \leq \sum_i |z_i^*| = \|\mathbf{z}^*\|$. ∎

Proof of Lemma C.6. Let $n \in [N-1]$. Let $\mathcal{S}_n^+ = \{i \in [N] : i > n, (z_+)_i \neq 0\}, \mathcal{S}_n^- = \{i \in [N] : i \leq n, (z_-)_i \neq 0\}$. Observe $\mathcal{S}_{n+1}^+ = \mathcal{S}_n^+ - \{n+1\}$ if $(z_+)_{n+1} \neq 0$ and $\mathcal{S}_{n+1}^+ = \mathcal{S}_n^+$ otherwise. Similarly, $\mathcal{S}_{n+1}^- = \mathcal{S}_n^+ \cup \{n+1\}$ if $(z_-)_{n+1} \neq 0$ and $\mathcal{S}_{n+1}^- = \mathcal{S}_n^-$ otherwise. Now, $\mathrm{ReLU}_{x_i}^+$ has slope 1 after x_i and $\mathrm{ReLU}_{x_i}^-$ has slope 1 before x_i, so $\mu_n = \sum_{i \in \mathcal{S}_n^+} (z_+)_i + \sum_{i \in \mathcal{S}_n^-} (z_-)_i$. Therefore, $|\mu_n - \mu_{n+1}| = \left|-(z_+)_{n+1} + (z_-)_{n+1}\right| \leq \left|(z_+)_{n+1}\right| + \left|(z_-)_{n+1}\right|$. So $\sum_{n=1}^{N-1} |\mu_n - \mu_{n+1}| \leq \|\mathbf{z}^*\|_1 - |(z_+)_1| - |(z_-)_1| \leq \|\mathbf{z}^*\|_1$.

Now, for any $n \in [N-1]$, $(\mathbf{A}\mathbf{z}+\xi\mathbf{1})_n - (\mathbf{A}\mathbf{z}+\xi\mathbf{1})_{n+1} = \sum_{i=1}^N (z_+)_i \left(\mathbf{A}_{+n,i} - \mathbf{A}_{+n+1,i}\right) = \sum_{i=1}^N (z_+)_i \left((x_n - x_i)_+ - (x_{n+1} - x_i)_+\right) = \sum_{i=n+1}^N (\mu_{i-1} - \mu_i)(x_n - x_{n+1}) = (x_n - x_{n+1})\,\mu_n = y_n - y_{n+1}$. And $(\mathbf{A}\mathbf{z} + \xi\mathbf{1})_N = \xi + \sum_{i=1}^N (z_+)_i (x_N - x_i)_+ = y_N + \sum_{i=1}^N (z_+)_i (0) = y_N$. So $\mathbf{A}\mathbf{z}+\xi\mathbf{1} = \mathbf{y}$. And $\|\mathbf{z}\|_1 = \sum_{n=1}^{N-1} |\mu_n - \mu_{n+1}|$, which exactly hits the lower bound on $\|\mathbf{z}^*\|_1$. Therefore $\mathbf{z}, \xi$ is optimal. ∎

Remark D.23. *By Lemma D.26, the absolute value network "de-biases" the target vector, normalizes is by the interval lengths $x_i - x_{i+1}$, and applies $\mathbf{E}$ (which contains the difference matrix Δ) twice, acting as a second-order difference detector. By Lemma D.25, the sign network's dictionary inverse contains Δ just once, acting as a first-order difference detector.*

Inverses of 2-layer dictionary matrices.

In this section, we consider the 2-layer dictionary matrix $\mathbf{A}$ as defined in Corollary 3.15. Define the *finite difference matrix*

$$\Delta = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -1 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \in \mathbb{R}^{N-1 \times N-1}. \tag{D.27}$$

Multiplying a matrix on its right by Δ subtracts its consecutive rows. Define the diagonal matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$ by $\mathbf{D}_{i,i} = \frac{1}{x_i - x_{i+1}}$ for $i \in [N-1]$ and $\mathbf{D}_{N,N} = \frac{1}{x_1 - x_N}$. For $n \in \mathbb{N}$, let

$$\mathbf{A}_n^{(s)} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ -1 & 1 & 1 & \cdots & 1 \\ -1 & -1 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & -1 & \cdots & 1 \end{pmatrix}, \mathbf{A}_n^{(t)} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & & \ddots \ \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbb{R}^{n \times n}. \tag{D.28}$$

Remark D.24. *The dictionary matrices for sign and threshold activation satisfy $\mathbf{A} = \mathbf{A}_N^{(s)}$ and $\mathbf{A}_+ = \mathbf{A}_N^{(t)}$, respectively.*

D.5. Results about 2-layer dictionary matrices.

Lemma D.25. *The dictionary matrix for $\sigma(x) = \mathrm{sign}(x)$ has inverse* $\mathbf{A}^{-1} = \frac{1}{2} \left(\begin{array}{ccc|c} & & & 0 \\ & \Delta & & \vdots \\ & & & 0 \\ & & & -1 \\ \hline 1 & 0 & \cdots\ 0 & 1 \end{array}\right)$.

Proof. Multiplying the two matrices (see Remark D.24) gives the identity. ∎

Lemma D.26. *The dictionary matrix* $\mathbf{A}$ *for absolute value activation has inverse* $\mathbf{A}^{-1} = \frac{1}{2}\mathbf{PEDE}$, *where* $\mathbf{P} = \left(\begin{array}{cccc|c} 0 & 0 & \cdots & 0 & 1 \\ \hline & & & & 0 \\ & & I_{N-1} & & \vdots \\ & & & & 0 \\ & & & & 0 \end{array}\right), \mathbf{E} = \left(\begin{array}{ccc|c} & & & 0 \\ & \Delta & & \vdots \\ & & & 0 \\ & & & -1 \\ \hline -1 & 0 & \cdots & 0 \quad -1 \end{array}\right)$.

Proof. For $i \in [N-1], j \in [N]$, $\mathbf{A}_{i,j} - \mathbf{A}_{i+1,j} = \begin{cases} (x_j - x_i) - (x_j - x_{i+1}) = x_{i+1} - x_i & \text{if } i > j \\ (x_i - x_j) - (x_{i+1} - x_j) = x_i - x_{i+1} & \text{if } i \leq j. \end{cases}$
And for all $j \in [N]$, $\mathbf{A}_{1,j} + \mathbf{A}_{N,j} = (x_1 - x_j) + (x_j - x_N) = x_1 + x_N$. Therefore,

$$\mathbf{DEA} = \left(\begin{array}{c|ccc} -1 & & & \\ \vdots & & \mathbf{A}^{(s)}_{N-1} & \\ -1 & & & \\ \hline -1 & -1 & \cdots & -1 \end{array}\right), \quad \frac{1}{2}\mathbf{EDEA} = \left(\begin{array}{c|ccc} 0 & & & \\ \vdots & & I_{N-1} & \\ 0 & & & \\ \hline 1 & 0 & \cdots & 0 \end{array}\right).$$

Applying the permutation $\mathbf{P}$ makes $\frac{1}{2}\mathbf{PEDEA} = \mathbf{I}$, so $\mathbf{A^{-1}} = \frac{1}{2}\mathbf{PEDE}$. ∎

Lemma D.27. *The inverse of* $\mathbf{A}_+$ *for threshold activation is* $\mathbf{A}_+^{-1} = \left(\begin{array}{ccc|c} & & & 0 \\ & \Delta & & \vdots \\ & & & 0 \\ & & & -1 \\ \hline 0 & 0 & \cdots & 0 \quad 1 \end{array}\right)$.

Proof. Multiplying the two matrices (see Remark D.24) gives the identity. ∎

Lemma D.28. *The submatrix* $[(\mathbf{A}_+)_{1:N,2:N}, (\mathbf{A}_-)_{1:N,1}] \in \mathbb{R}^{N \times N}$ *of the dictionary matrix for ReLU activation has inverse* $\mathbf{E}_+\mathbf{DE}_-$, *where*

$$\mathbf{E}_+ = \left(\begin{array}{ccc|c} & & & 0 \\ & \Delta & & \vdots \\ & & & 0 \\ & & & 1 \\ \hline 0 & 0 & \cdots & 0 \quad 1 \end{array}\right), \quad \mathbf{E}_- = \left(\begin{array}{ccc|c} & & & 0 \\ & \Delta & & \vdots \\ & & & 0 \\ & & & -1 \\ \hline 0 & 0 & \cdots & 0 \quad -1 \end{array}\right).$$

Proof. For $i \in [N-1], j \in [N]$,
$(\mathbf{A}_+)_{i,j} - (\mathbf{A}_+)_{i+1,j} = \begin{cases} 0 & \text{if } i \geq j \\ (x_i - x_j) - (x_{i+1} - x_j) = x_i - x_{i+1} & \text{if } i < j \end{cases}$
and $(\mathbf{A}_-)_{i,1} - (\mathbf{A}_-)_{i+1,1} = (x_1 - x_i) - (x_1 - x_{i+1}) = x_{i+1} - x_i$. Observe that $\mathbf{DE}_-\mathbf{A} = \left(\begin{array}{ccc|c} & & & -1 \\ & \mathbf{A}^{(t)}_{N-1} & & \vdots \\ & & & -1 \\ & & & -1 \\ \hline 0 & 0 & \cdots & 0 \quad 1 \end{array}\right)$, and applying $\mathbf{E}_+$ gives $\mathbf{I}$. ∎

Appendix E. The solution sets of Lasso and the training problem.

Recall that each 2-layer feature corresponds to $\mathbf{W}^{(1)} \in \{-1, 1\}, \mathbf{b}^{(1)} = -\mathbf{W}^{(1)} x_n$ for some $n \in [N]$. We next define functions $R^{z \to \alpha}, R^{\alpha, w, b \to \theta}, R^{w,b}, R$ that are composed with each other to reconstruct 2-layer networks. Each feature in a 2-layer network is of the form $f(x) = \sigma\left(x\mathbf{W}^{(1)} + \mathbf{b}^{(1)}\right)$ where $\mathbf{W}^{(1)} \in \{-1, 1\}$ and $\mathbf{b}^{(1)} = -\mathbf{W}^{(1)} x_n$ for some $n \in [N]$.

Definition E.1. *Consider a 2-layer ReLU, absolute value, or leaky ReLU network. Let* $R^{z\to\alpha}(z)=\text{sign}(z)\sqrt{|z|}$. *For* $w\in\{-1,1\}, b\in\{-x_n : n\in[N]\}$, *let* $R^{\alpha,w,b\to\theta}(\alpha)=(\alpha,\alpha w,-\alpha w b)$. *Let* $R^{w,b}(z)=R^{\alpha,w,b\to\theta}\left(R^{z\to\alpha}(z)\right)$. *Define the* 2*-layer* reconstruction function $R(\mathbf{z})$ *which outputs a vector whose* i^{th} *element is* $R^{w,b}(z_i)$, *where* $(w,b)=\left(\mathbf{W}^{(1)},\mathbf{b}^{(1)}\right)$ *corresponds to the* i^{th} *feature.*

In Definition E.1, we have abused notation where α, z, w in the superscripts of functions are only used to identify the function, and are distinct from the arguments to the functions. The function $R^{\alpha,w,b\to\theta}$ is parameterized by w, b. For a 2-layer network, let $w_i=\mathbf{W}^{(i,1)}, b_i=\mathbf{b}^{(i,1)}\in\mathbb{R}$ and $\mathbf{w}, \mathbf{b}, \boldsymbol{\alpha}$ be vectors stacking together all w_i, b_i, α_i respectively. The reconstructed parameters are $(\boldsymbol{\alpha}, \mathbf{w}, \mathbf{b}) = R(\mathbf{z}^*)$ and $\xi = \xi^*$.

For a 2-layer network, let $w_i=\mathbf{W}^{(i,1)}, b_i=\mathbf{b}^{(i,1)}\in\mathbb{R}$ and $\mathbf{w}, \mathbf{b}, \boldsymbol{\alpha}$ be vectors stacking together all w_i, b_i, α_i respectively. The reconstructed parameters are $(\boldsymbol{\alpha}, \mathbf{w}, \mathbf{b}) = R(\mathbf{z}^*)$ and $\xi = \xi^*$. We have shown that training neural networks on 1-D data is equivalent to fitting a Lasso model. Now we develop analytical expressions for all minima of the Lasso problem and its relationship to the set of all minima of the training problem. These results, which build on the existing literature for convex reformulations (28) as well as characterizations of the Lasso (10), illustrate that the Lasso model provides insight into non-convex networks. We focus on 2-layer models with ReLU, leaky ReLU and absolute value activations, although our results can be extended to other architectures by considering the corresponding neural net reconstruction. Proofs are deferred to Appendix E.1.

We start by characterizing the set of global optima to the Lasso problem (1.2). Suppose $(\mathbf{z}^*, \xi^*)$ is a solution to the convex training problem. We first extend classical uniqueness results (41; 40) for the model fit to the Lasso with unregularized bias.

Lemma E.2. *Fix* $\beta > 0$. *Then the optimal model fit* $\hat{\mathbf{y}} = \mathbf{A}\mathbf{z}^* + \xi^*\mathbf{1}$ *and linear fit* $\mathbf{A}\mathbf{z}^*$ *are the same for all optimal solutions* $(\mathbf{z}^*, \xi^*)$ *to Lasso problem* (1.2). *As a result,* ξ^* *is also unique.*

As a result, we can uniquely identify the equicorrelation set $\mathcal{E}_\beta$ using $\hat{\mathbf{y}}$,

$$\hat{\mathbf{y}} = \mathbf{A}\mathbf{z}^* + \xi^*\mathbf{1}, \qquad \mathcal{E}_\beta = \left\{ i : |\mathbf{A}_i^\top(\hat{\mathbf{y}} - \mathbf{y})| = \beta \right\}.$$

The equicorrelation set contains the features maximally correlated with the residual $\hat{\mathbf{y}} - \mathbf{y}$ and plays a critical role in the solution set.

Proposition E.3. *Suppose* $\beta>0$. *Then the set of global optima of the Lasso problem* (1.2) *is*

(E.1)
$$\Phi^*(\beta) = \left\{ (\mathbf{z}, \xi^*) : z_i \neq 0 \Rightarrow \text{sign}(z_i) = \text{sign}\left(\mathbf{A}_i^\top(\hat{\mathbf{y}} - \mathbf{y})\right),\ z_i = 0\,\forall i \notin \mathcal{E}_\beta,\ \mathbf{A}\mathbf{z} = \hat{\mathbf{y}} - \xi^*\mathbf{1}. \right\}$$

The solution set $\Phi^*(\beta)$ is polyhedral and its vertices correspond exactly to minimal models, i.e. models with the fewest non-zero elements of $\mathbf{z}$ (28). Let R be the reconstruction function described in Definition E.1. All networks generated from applying R to a Lasso solution are globally optimal in the training problem. The next result gives a full description of such networks.

Proposition E.4. *Suppose* $\beta > 0$ *and the activation is ReLU, leaky ReLU or absolute value.*

The set of all 2-layer Lasso-reconstructed networks is

$$R(\Phi(\beta)) = \Bigg\{ (\mathbf{w}, \mathbf{b}, \alpha, \xi^*) : \alpha_i \neq 0 \Rightarrow \operatorname{sign}(\alpha_i) = \operatorname{sign}\left(\mathbf{A}_i^\top (\mathbf{y} - \hat{\mathbf{y}})\right), b_i = -x_i \frac{\tilde{\alpha}_i}{\sqrt{|\alpha_i|}}, \\ w_i = \frac{\tilde{\alpha}_i}{\sqrt{|\alpha_i|}}; \; \alpha_i = 0 \, \forall i \notin \mathcal{E}_\beta, \; f_2(\mathbf{X}; \theta) = \hat{\mathbf{y}} \Bigg\}. \tag{E.2}$$

Proposition E.4 shows that all neural nets trained using our Lasso and reconstruction share the same model fit whose set of active neurons is at most the equicorrelation set. By finding just *one* optimal neural net that solves Lasso, we can form $R(\Phi^\beta)$ and compute all others.

The min-norm solution path is continuous for the Lasso problem (40). Since the solution mapping in Definition B.3, Appendix D.3 is continuous, the corresponding reconstructed neural net path is also continuous as long as the network is sufficiently wide. Moreover, we can compute this path efficiently using the LARS algorithm (10). This is in contrast to the under-parameterized setting, where the regularization path is discontinuous (28).

What subset of optimal, or more generally, stationary, points of the non-convex training problem (1.1) consist of Lasso-generated networks $R(\Phi(\beta))$? First, $R(\Phi(\beta))$ can generate additional optimal networks through neuron splitting, described as follows. Consider a single neuron $\alpha\sigma_s(wx + b)$ (where $\alpha, w, b \in \mathbb{R}$), and let $\{\gamma_i\}_{i=1}^n \subset [0,1]^n$ be such that $\sum_{i=1}^n \gamma_i = 1$. The neuron can be *split* into n neurons $\left\{\sqrt{\gamma_i}\alpha\sigma\left(\sqrt{\gamma_i}wx + \sqrt{\gamma_i}b\right)\right\}_{i=1}^n$ (45). For any collection Θ of parameter sets θ, let $P(\Theta)$ be the collection of parameter sets generated by all possible neuron splits and permutations of each $\theta \in \Theta$. Next, let $\mathcal{C}(\beta)$ and $\tilde{C}(\beta)$ be the sets of Clarke stationary points and solutions to the non-convex training problem (1.1), respectively.

Proposition E.5. *Suppose $L = 2, \beta > 0$, the activation is ReLU, leaky ReLU or absolute value and $m^* \leq m \leq 2N$. Let $\Theta^P = \{\theta : \forall i \in [m], \exists j \in [N] \text{ s.t. } b_i = -x_j w_i\}$. Then*

$$P(R(\Phi(\beta))) = \tilde{\mathcal{C}}(\beta) \cap \Theta^P = \mathcal{C}(\beta) \cap \Theta^P. \tag{E.3}$$

Proposition E.5 states that up to neuron splitting and permutation, our Lasso method gives all stationary points in the training problem satisfying $b_i = -x_i w_i$. Moreover, all such points are optimal in the training problem, similar to (17).

Since optimal solutions are stationary, a neural net reconstructed from the Lasso model is in $\tilde{C}(\beta) \subset \mathcal{C}(\beta)$. However, $C(\beta) \not\subset \Theta^P$. This is because there may be other neural nets with the same output on $\mathbf{X}$ as the reconstructed net so that they are all in $\mathcal{C}(\beta)$, but that differ in the the unregularized parameters $\mathbf{b}$ and ξ, so that they are not in Θ^P. For example, if β is large enough, the Lasso solution is $\mathbf{z} = \mathbf{0}$ (10), so the reconstructed net will have $\boldsymbol{\alpha} = \mathbf{0}$, which makes the neural net invariant to $\mathbf{b}$. In this section, we analyzed the general structure of the Lasso solution set when $\beta > 0$. Next, we show how this can be leveraged to explore the generalization performance of different optimal neural networks.

Generalization properties of networks constructed from different Lasso solutions. While all two-layer Lasso-reconstructed networks must have the same model fit for a fixed training set (Proposition E.4), they may have very different predictions on an independent test set. In particular, some models may generalize better at test time due to the "implicit regularization" (20) of the rule used to choose a model from the optimal set. Studying implicit regularization for

different problems and selection rules has been a topic of significant interest in the optimization community in recent years (30; 2; 19). However, our characterization of the optimal set introduces a completely new perspective that we can (at least in principle) compute the generalization of all optimal Lasso solutions simultaneously.

We approximate the distribution of test mean-square error (MSE) over the set of optimal three-layer, deep narrow ReLU networks reconstructed from Lasso solutions by sampling optimal solutions uniformly from the Lasso solution set. Since the Lasso optimal set is polyhedral, it is possible to sample optimal Lasso models uniformly at random using a hit-and-run algorithm (8). These samples are then reconstructed into non-convex neural network models and evaluated on a test set. In this way, we can approximate not just the implicit regularization of specific solutions, but also approximate the entire generalization range of the optimal Lasso set.

Figure E.1 shows the distribution of test MSEs for a planted affine model and a planted three-layer narrow ReLU network. We estimate each distribution using 5000 hit-and-run samples as described above. For each synthetic problem, we generate a set of 5 training examples and 100 test examples by sampling $x_i \sim \mathcal{N}(0,1)$ and then computing $y_i = ax_i + b + \epsilon_i$ or $y_i = f_3(x_i;\theta) + \epsilon_i$ for the planted affine and planted ReLU models, respectively. Here, $\epsilon_i \sim \mathcal{N}(0, 0.1^2)$ and θ is chosen so that $f_3(x_i;\theta)$ depends on only five features from the deep library.

We find that test performance of different optimal neural networks varies widely despite the models having exactly the same training MSE and model norm. For the planted ReLU problem, the variation over the optimal set is even greater than the variation due to the regularization strength, implying that implicit regularization may play an even more important role than hyper-parameter selection for some problems. We also find that the min-norm solution (marked with a red star) tends to the median generalization performance. This surprising observation contradicts the common wisdom that min-norm solutions have superior generalization, at least for networks reconstructed from Lasso solutions.

The solution sets of Lasso and the training problem.

E.1. Proofs for results in Appendix E.

Lemma E.2. Uniqueness of $\hat{\mathbf{y}}$ is a consequence of strict convexity of the squared-error loss. We refer to (40, Lemma 1) for a proof of this result. To see that $\hat{\mathbf{q}} = \mathbf{A}\mathbf{z}^*$ is also unique, consider minimizing over ξ in (1.2). Since ξ is unregularized, this is a quadratic minimization problem with solution

$$\begin{aligned}\xi^* &= \bar{\mathbf{y}} - \frac{1}{n}\langle \mathbf{1}, \mathbf{A}\mathbf{z}\rangle \\ &= \bar{y} - \bar{q},\end{aligned}$$

where we have defined $\bar{y} = \frac{1}{n}\sum_{i=1}^n \mathbf{y}_i$ and $\bar{q} = \frac{1}{n}\sum_{i=1}^n \hat{\mathbf{q}}_i$. Plugging this back into the lasso gives an optimization problem purely in $\mathbf{z}$,

$$\min_{\mathbf{z}} \frac{1}{2}\left\|\mathbf{A}\mathbf{z} - \frac{1}{n}\langle \mathbf{1}, \mathbf{A}\mathbf{z}\rangle \mathbf{1} - (\mathbf{y} - \bar{y}\mathbf{1})\right\|_2^2 + \tilde{\beta}\|\mathbf{z}\|_1.$$

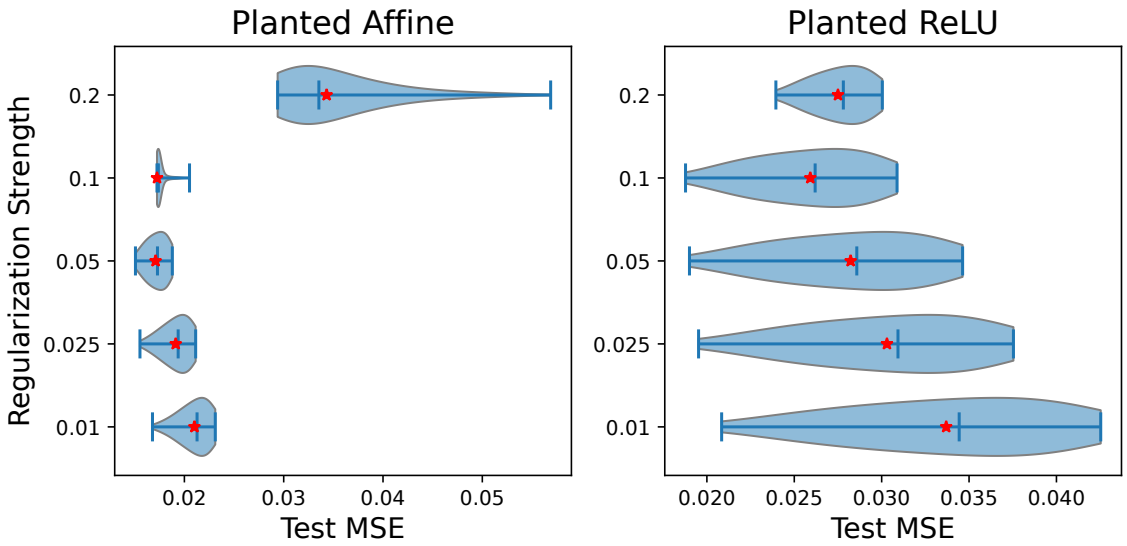


Figure E.1: Distribution of 5000 test MSEs obtained by sampling uniformly from the optimal set for three-layer narrow ReLU networks. We experiment with two synthetic 1-D datasets, one with a planted noisy affine model (left) and one with a planted noisy three-layer narrow ReLU network (right). The red star marks the test MSE of the minimum Euclidean-norm solution to the Lasso reformulation, while the bars mark the minimum, median, and maximum sample test MSEs, respectively.

The squared-error term of the objective can be parameterized in terms of $\hat{\mathbf{q}}$ as

$$\frac{1}{2}\left\|\hat{\mathbf{q}} - \frac{1}{n}\langle \mathbf{1}, \hat{\mathbf{q}}\rangle \mathbf{1} - (\mathbf{y} - \bar{y}\mathbf{1})\right\|_2^2.$$

Clearly this objective is convex in $\hat{\mathbf{q}}$, but it's not yet obvious if it is strictly convex. The Hessian of the squared-error is given by,

$$\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top.$$

For any $\mathbf{w} \in \mathbb{R}^n$, we have,

$$\begin{aligned}
\mathbf{w}^\top \mathbf{H}\mathbf{w} &= \|\mathbf{w}\|_2^2 - \frac{1}{n}(\langle \mathbf{1}, \mathbf{w}\rangle)^2 \\
&\geq \|\mathbf{w}\|_2^2 - \frac{1}{n}\|\mathbf{1}\|_2^2\|\mathbf{w}\|_2^2 \\
&= 0,
\end{aligned}$$

by Cauchy-Schwarz. Moreover, equality holds if and only if $\mathbf{w} \propto \mathbf{1}$. This implies that the squared-loss is strictly convex along the cord between $\hat{\mathbf{q}}_1$ and $\hat{\mathbf{q}}_2$ if and only if $\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2 \not\propto \mathbf{1}$.

Suppose an optimal fit satisfies $\hat{\mathbf{q}} \propto \mathbf{1}$ with $\mathbf{z}^* \neq 0$. Then by setting $\xi' = \xi^* + \hat{\mathbf{q}}_0$ and $\mathbf{z}' = \mathbf{0}$ we can strictly decrease the lasso objective (since $\mathbf{z}$ is regularized and ξ is not). But this

contradicts optimality of $\mathbf{z}^*$. We conclude that no optimal linear fit $\hat{\mathbf{q}}$ is proportional to the one vector. Combining this with the preceding argument shows that the least-squares objective is strictly convex in $\hat{\mathbf{q}}$. It is now straightforward to show that the optimal $\hat{\mathbf{q}}$ is unique following a similar argument as (40, Lemma 1). ∎

Proposition E.3. The result is almost a sub-case of that given by Mishkin and Pilanci (28) with the exception that the bias parameter ξ, is not regularized. Therefore optimality conditions do not impose a sign constraint and it is sufficient that $\mathbf{1}^\top(\mathbf{A}\mathbf{z}+\xi\mathbf{1}-\mathbf{y})=0$ for ξ to be optimal. This stationarity condition is guaranteed by $\mathbf{A}\mathbf{z}+\xi\mathbf{1}=\hat{\mathbf{y}}$. In Lemma E.2, we showed that these optimality conditions admit a unique solution in $\mathbf{A}\mathbf{z}$ and ξ. Thus, we can conclude ξ^* is the unique optimal output bias and $\hat{\mathbf{y}}-\xi^*\mathbf{1}$ is unique. Now let us look at the parameters z_i. If $i\notin\mathcal{E}(\beta)$, then $z_i=0$ is necessary and sufficient from standard results on the Lasso (40). If $i\in\mathcal{E}(\beta)$ and $z_i\neq 0$, then $\mathbf{A}_i^\top(\hat{\mathbf{y}}-\mathbf{y})=\beta\text{sign}(z_i)$, which shows that $\mathbf{z}_i$ satisfies first-order conditions. If $z_i=0$, then first-order optimality is immediate since $|\mathbf{A}_i^\top(\hat{\mathbf{y}}-\mathbf{y})|\leq\beta$, holds. Putting these cases together completes the proof. ∎

Proposition E.4. This result follows from applying the reconstruction in Definition B.3 to each optimal point in Φ. The reconstuction sets $\alpha_i=\text{sign}(z_i)\sqrt{|z_i|}$. From this we deduce $\text{sign}(\alpha_i)=\text{sign}\left(\mathbf{A}_i^\top(\mathbf{y}-\hat{\mathbf{y}})\right)$. The solution mapping determines the values of w_i and b_i and in terms of α_i. Finally, the constraint $f_2\left(\mathbf{X};\theta\right)=\hat{\mathbf{y}}$ follows immediately by equality of the convex and non-convex prediction functions on the training set. ∎

Lemma E.6. *Suppose $L=2$, and the activation is ReLU, leaky ReLU or absolute value. Suppose $m^*\leq m\leq 2N$. Since $m\leq 2N$, we can let $\Theta^{\text{Lasso,stat}}=\{\theta:\exists j\in[N]\ s.t.\ b_i=-x_jw_i\}\subset\Theta$. Let $\theta^*\in\Theta^{\text{Lasso,stat}}\cap C(\beta)$. Then θ^* is a minima of the Lasso problem.*

Proof. We can ignore ξ since its derivative and reconstruction are straightforward, and it does not interact with any other parameters. We denote vector-vector operations as being performed elementwise.

Since $m^*\leq m$, a neural net reconstructed from Lasso is optimal in the training problem. Let $F(\theta)$ and $F^{\text{Lasso}}(\mathbf{z})$ be the objectives of the non-convex training problem (1.1) and Lasso (1.2), respectively. The parameters θ are stationary if $\theta\in C(\beta)$, i.e., $0\in\partial F(\theta)$.

Let $\Theta^{\text{Lasso}}=\{\theta:\exists j\in[N]\text{ s.t. }|w_i|=|\alpha_i|,b_i=-x_jw_i\}\subset\Theta^{\text{Lasso,stat}}$. By a similar argument as the proof of Theorem 3 in (45), since $0\in\partial F(\theta^*)$, we have $|w_i|=|\alpha_i|$ for all neurons i. Therefore $\theta^*\in\Theta^{\text{Lasso}}$. Thus for $\alpha^*\in\theta^*$, observe that $\theta^*=R^{\alpha,w,b\to\theta}(\boldsymbol{\alpha}^*)$. Let $\tilde{F}(\boldsymbol{\alpha})=F\left(R^{\alpha,w,b\to\theta}(\boldsymbol{\alpha})\right)$.

Since $0\in\partial F(\theta^*)$ at $(\boldsymbol{\alpha}^*)=\left(R^{\alpha,w,b\to\theta}\right)^{-1}(\theta^*)$, we have $\tilde{F}(\boldsymbol{\alpha}^*)=F(\theta^*)$. Perform the following operations elementwise. The chain rule gives $\partial\tilde{F}(\boldsymbol{\alpha}^*)=\partial F(\theta^*)\partial R^{\alpha,w,b\to\theta}(\boldsymbol{\alpha}^*)\ni\mathbf{0}$. Let $R^{\alpha toz}(\boldsymbol{\alpha})=\text{sign}(\boldsymbol{\alpha})\boldsymbol{\alpha}^2$. At $\mathbf{z}^*=R^{\alpha\to z}(\boldsymbol{\alpha}^*)$, we have $F^{\text{Lasso}}(\mathbf{z}^*)=\tilde{F}(\boldsymbol{\alpha}^*)$. The chain rule gives $\partial F^{\text{Lasso}}(\mathbf{z}^*)=\partial\tilde{F}(\boldsymbol{\alpha}^*)\partial R(\mathbf{z}^*)\ni\mathbf{0}$. Since the Lasso problem (1.2) is convex, the result holds. ∎

Proof of Proposition E.5. Observe that $R(\Phi(\beta))\subseteq\tilde{\mathcal{C}}(\beta)\cap\Theta^{\text{Lasso,stat}}\subseteq\mathcal{C}(\beta)\cap\Theta^{\text{Lasso,stat}}\subseteq R(\Phi(\beta))$, where the first and last subset inequality follow from Theorem 3.11 and Lemma E.6, respectively. Therefore all subsets in the above expression are equal. Observe that $P\left(\Theta^{\text{Lasso,stat}}\right)=\Theta^P$ and so $P\left(C(\beta)\cap\Theta^{\text{Lasso,stat}}\right)=C(\beta)\cap P\left(\Theta^{\text{Lasso,stat}}\right)=C(\beta)\cap\Theta^P$ and similarly $P\left(\tilde{C}(\beta)\cap\Theta^{\text{Lasso,stat}}\right)=\tilde{C}(\beta)\cap\Theta^P$. Now apply P to all subsets above. ∎

E.2. Detailed results of Section 4. Unless otherwise stated, all experiments train for 100 epochs. In Figure 4.3, Figure 4.5, Figure 4.6 and Figure 4.7, the number of time steps is $T = 1001$, the deep narrow architecture is specified by $m_L = 10$ parallel units, and $\beta = \frac{1}{T}$ in the training problem (4.3) and (1.1) for more general L layers.

In Figure 4.1, Figure 4.2 and Figure 4.4, we solve the autoregression problems with 20 trials of stochastic gradient descent (SGD) initializations versus using the Lasso problem (1.2). We compare against the baseline linear method $f(x;\theta) = ax + b$ (AR1+bias), where we include an additional bias term b. For each dataset, training and test sets with T data points each are chosen. In Figure 4.1 and Figure 4.2, the learning rate is 0.05 and $T = 2000$. In Figure 4.4, the learning rate is 0.1. Figure 4.1 shows that using Lasso(cvxNN) yields a consistent lower bound on the training loss and demonstrates strong generalization properties, compared to large fluctuation in the loss curves of the non-convex method (NN). Additional results are in the supplementary material (47).

Appendix E.2 discusses planted data, for example shown in the regularization path plotted in Figure 4.4. Figure E.3 shows training curves for networks for planted data with $\sigma^2 = 0.01$ in (4.1) and learning rate 0.1. In Figure 4.4 and Figure E.3, $T = 1000$ time steps is used for generating each of the planted training and testing data. Figure E.3 shows that using Lasso gives lower training loss when using planted data. This appears to occur because different trials of NN (neural net trained directly without Lasso) get stuck into local minima. The global optimum that cvxNN reaches also enjoys effective generalization properties, as seen by the test loss. In Figure 4.1, Figure 4.2, Figure 4.4, and Figure E.3, $m = 200$ neurons are trained over 100 epochs and $\beta = 1$ in the training problem (4.3).

Figure E.2 shows that even for the same number of features, the training time increases with depth. Autoregressive deep narrow neural networks are trained on $T = 1000$ samples of BTC-2017min data with the Lasso program, with 10^3 features subsampled. The random subsampling is repeated 40 times according to the procedure described in Section 4.

A *quantile regression* problem is the training problem (4.3) with the quantile loss $\mathcal{L} = L_\tau(z) = 0.5|z| + (\tau - 0.5)z$, where the network $f_2^{\mathrm{NN}}(x_t;\theta)$ models the τ-quantile of the prediction for x_{t+1} based on the observation x_t. Additional results and comparisons including applying the same experiments to quantile regression and hourly BTC price (BTC-hourly) are in the supplementary material (47).

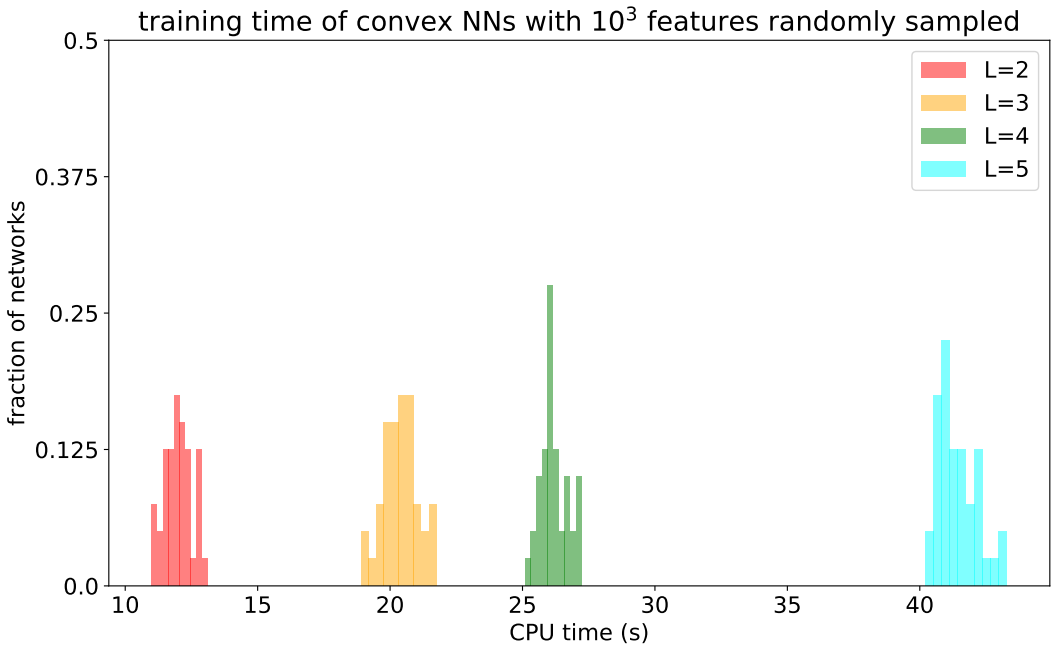


Figure E.2: Histogram of training time of deep narrow networks (40 networks for each depth L) trained on the Lasso problem with randomly sampled 10^3 features for autoregression on BTC-2017min with $T = 10^3$ samples. The training time increases with depth.

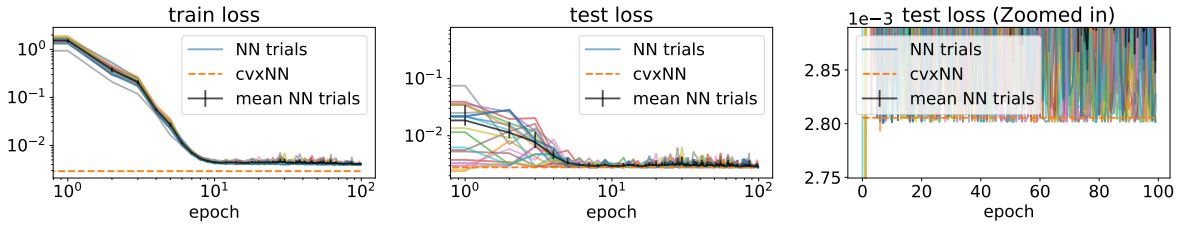


Figure E.3: Loss curves for planted data where $\sigma^2 = 0.01$ in (4.1) and there are $p = 10$ planted neurons. The non-black, solid colored curves plot each NN trial, and the black curve plots their mean, with the vertical lines indicating one standard deviation of the data above and below the mean.