

Black Boxes and Looking Glasses: Multi-level Symmetries, Reflection Planes, and Convex Optimization in Deep Networks

Emi Zeger, *Student Member, IEEE*, and Mert Pilanci, *Member, IEEE*,

Abstract

We consider a class of *deep narrow networks*, which have piecewise linear activations and enable analysis on the role of depth in representational power. We show that training deep narrow neural networks with ℓ_1 regularization and an arbitrary input dimension can be formulated as equivalent convex Lasso problems with novel features, expressed using geometric algebra, that encode multi-level symmetries. While the number of features is exponential in the data dimension, the Lasso formulation provides convexity and interpretability of certain neural networks as feature selectors. We describe the full Lasso dictionary for 3-layer deep narrow networks, and a subset of the dictionary for deeper networks. The multi-level symmetries are characterized for absolute value activation. Using the equivalent Lasso form, we prove that deep narrow networks with non-monotone activations inherently learn symmetric structures representing reflections of training data. In particular, 3-layer deep narrow networks learn distances to hyperplanes that are reflected across training points. Greater depth enables higher levels of symmetries that become more apparent in deep narrow networks with absolute value activation, which have features corresponding to higher levels of Cantor sets for certain training data. Numerical experiments, including training on embeddings generated by Large Language Models (LLMs), support the theory.

I. INTRODUCTION

Recent advancements have demonstrated that deep neural networks are powerful models for data processing, classification, regression, and generation. While [1] and other universal approximation theorems show that in theory, 2-layer neural networks can approximate any reasonable function given a sufficient number of neurons, it remains an open problem what kinds of functions a neural network *tends* to converge to during training in practice, and how this changes with depth. We are interested in the following question:

Is there a fundamental difference in the nature of functions learned by deep networks, as opposed to shallow networks?

We establish a foundation to answer this question by transforming non-convex training problems for a class of depth-focused networks with absolute value and leaky ReLU activations into convex formulations and analyzing their resulting structure. **We show that deep narrow nets favor symmetric structures, with greater depth**

E. Zeger and M. Pilanci are with the Electrical Engineering Department, Stanford, CA, 94305

representing multi-level symmetries encoded via reflections of data points. The convex formulations involve an exponential number of features with respect to the data dimension, but offers increased transparency into analyzing deep narrow networks.

This work is organized as follows. Section I-A defines the deep narrow network architecture and corresponding Lasso problems, and Section I-B summarizes our results connecting them. Section II describes prior work related to the contributions listed in Section I-B. Section III defines tools from geometric algebra and illustrates their use in simpler prior results that are of similar nature as this work. Using geometric algebra, Section IV-A discusses our main results for deep narrow networks of depth 3, and Section IV-B extends these results to arbitrary depth. Section VI gives numerical experiments. For space and clarity, proofs, certain details and results are deferred to the Appendix. In particular, Section A addresses the choices of the deep narrow network architecture, activation and weight regularization during training, as well as the limitations and extendability of results to more general architectures.

A. Background: deep narrow networks and Lasso models

A neural network is a parameterized map $\mathbb{R}^{1 \times d} \rightarrow \mathbb{R}$. A 2-layer network with m neurons is

$$f_2(\mathbf{x}; \theta) = \sigma(\mathbf{x}\mathbf{W} + \mathbf{b})\boldsymbol{\alpha} + \xi, \quad \text{where } \theta = \{\mathbf{W} \in \mathbb{R}^{d \times m}, \mathbf{b} \in \mathbb{R}^{1 \times m}, \boldsymbol{\alpha} \in \mathbb{R}^m, \xi \in \mathbb{R}\} \quad (1)$$

is a set of trainable parameters. Assume σ is a piecewise linear activation function. Equivalently,

$$f_2(\mathbf{x}; \theta) = \sum_{i=1}^m \sigma(\mathbf{x}\mathbf{w}^{(i)} + b^{(i)})\alpha_i + \xi \quad (2)$$

where $\mathbf{w}^{(i)} \in \mathbb{R}^d$ are the columns of $\mathbf{W}$ and $b^{(i)} \in \mathbb{R}$ are the elements of $\mathbf{b}$. The parameters $\mathbf{w}^{(i)}, b^{(i)}$ are the weight and bias for the i^{th} neuron $\sigma(\mathbf{x}\mathbf{w}^{(i)} + b^{(i)})$. To study deeper networks, we begin by composing each neuron in (2) with another to make a 3-layer network with m_3 neurons in each layer as

$$f_3(\mathbf{x}; \theta) = \sum_{i=1}^{m_3} \sigma(\sigma(\mathbf{x}\mathbf{w}^{(i,1)} + b^{(i,1)})w^{(i,2)} + b^{(i,2)})\alpha_i + \xi. \quad (3)$$

We continue this pattern for deeper layers (as formally defined by (21) in Section E) and call networks of this form *deep narrow networks*¹. While more general networks typically employ matrix weights, deep narrow networks assume $w^{(i,l)}, b^{(i,l)} \in \mathbb{R}$ for all layers l except for $\mathbf{w}^{(i,1)} \in \mathbb{R}^d$. Our goal with the deep narrow network architecture is to understand the role of depth in the simplest non-trivial class of networks (see Section A-A for discussion). Figure 1 illustrates the deep narrow network architecture. Let N be the number of training samples and $\mathbf{X} \in \mathbb{R}^{N \times d}$ be the *training data matrix*, where each row of $\mathbf{X}$ is a training sample $\mathbf{x}_n \in \mathbb{R}^{1 \times d}$. The neural network extends to matrix inputs row-wise. We consider regression problems, although the results can also be applied to classification problems. The *target* or *label* vector is $\mathbf{y} \in \mathbb{R}^N$. The neural network *training problem* is

$$\min_{\theta \in \Theta} \frac{1}{2} \|\mathbf{f}_L(\mathbf{X}; \theta) - \mathbf{y}\|_2^2 + \frac{\beta}{L} \sum_{i=1}^{m_L} \left(|\alpha_i|^L + \sum_{l=1}^{L-1} \|\mathbf{w}^{(i,l)}\|_1^L \right) \quad (4)$$

¹An alternative name that emphasizes the parallel nature of these networks might be "deep parallel network"; however, we avoid this term as it might overstate the architecture by implying that each parallel unit has more than one neuron per layer.

where $\theta = \{\mathbf{W}^{(i,l)}, \mathbf{b}^{(i,l)}, \alpha_i, \xi : i \in [m_L], l \in [L-1]\}$ denotes an ordered set of trainable parameters, $[n] = \{1, \dots, n\}$, and Θ is the parameter space. Neural network inputs, parameters, and outputs are all real-valued. The *regularization parameter* is $\beta > 0$, which is fixed. The training problem is non-convex, which can make neural networks difficult to train and interpret. In contrast, a *Lasso problem* is a convex optimization problem of the form

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

where $\mathbf{A}$ is a *dictionary matrix*, and its columns $\mathbf{A}_i \in \mathbb{R}^N$ are *feature vectors*, and $\beta > 0$. We will show that the Lasso problem and the deep narrow network training problem are equivalent (Theorem IV.8). The Lasso problem and its solution set are well-studied [2], [3], and the Lasso problem has efficient solvers using proximal gradient methods and Least Angle Regression [4], [5]. For training, the convexity of the Lasso problem implies that all of its stationary points are optimal, which prevents convergence to sub-optimal local minima that can occur with the non-convex training problem (4).

The ℓ_1 regularization in the Lasso problem (5) favors a sparse solution for $\mathbf{z}$ [2], making the Lasso problem interpretable as a feature selector. The dictionary columns $\mathbf{A}_i$ where $z_i \neq 0$ constitute a sparse set of feature vectors that we will show correspond to *feature* functions defining an optimal neural network. For $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$, the *reflection of a about b* is $R(\mathbf{a}, \mathbf{b}) = 2\mathbf{b} - \mathbf{a}$. With respect to $\mathbf{b}$, the points $\mathbf{a}$ and $R(\mathbf{a}, \mathbf{b})$ exhibit a *single-level symmetry*. Multi-level symmetries can be created through nested reflections. For example, a double reflection can be of the form $R(R(\mathbf{a}, \mathbf{b}), \mathbf{c})$. Section IV-B formally defines higher order reflections.

B. Contributions

We show the following:

- 1) Training deep narrow networks of arbitrary depth is equivalent to solving convex Lasso problems with a finite set of features (Theorem IV.8).
- 2) For 3-layer deep narrow networks with non-monotone activations (such as leaky ReLU and absolute value), these features represent normalized reflections (Theorem IV.5). In particular, for absolute value activation, the features measure distances to reflection planes spanned by subsets of training data (Theorem IV.6)
- 3) Lasso features for absolute value activation represent multi-level symmetries of increasing complexity (Theorem IV.9). Features for certain training data can represent Cantor sets, with network depth corresponding to Cantor set level (Theorem V.1).
- 4) 3-layer deep narrow networks admit explicit Lasso dictionaries described using geometric algebra that represent volumes of parallelotopes spanned by training data, and explicit reconstructions of optimal networks from corresponding Lasso models. Reconstructed, optimal first-layer weights are orthogonal to subsets of augmented training data (Theorem IV.2).
- 5) In experiments (Section VI), networks trained using Large Language Model (LLM) text embeddings learn Lasso-like features and networks trained on Cantor set data demonstrate advantages of depth as consistent with theory.

C. Architecture and training problem selection

We summarize the architectural and regularization choices used in our network and training problem (4), in particular their connections to prior work and role in enabling tractable convex reformulations.

1) *First-layer ℓ_1 regularization $\|\mathbf{W}^{(i,1)}\|_1$ and intermediate-layer scalar weights $\mathbf{W}^{(i,l)} \in \mathbb{R}$* : These assumptions are used to obtain a finite set of discrete Lasso features. Constraints on $\|\mathbf{W}^{(i,1)}\|_1$ comprise one of the settings in which [6] formulates convex models of infinitely wide, 2-layer networks. As a special case when $d = 1$, we have $\mathbf{W}^{(i,1)} \in \mathbb{R}$, so the ℓ_1 regularization for all $\mathbf{W}^{(i,l)}$ is equivalent to any ℓ_p norm. Finite feature sets for Lasso are also obtained in 3-layer ReLU networks with 1-D data and equal-magnitude rather than scalar intermediate weights [7]; and (up to an approximation) in 2-layer ReLU networks with ℓ_2 instead of ℓ_1 regularization on data satisfying certain dispersion properties [8]. This indicates our assumptions are not the only way to admit similar (possibly approximate) features, but provide a tractable setting for exactly characterizing them. Extending such results to deeper, leaky ReLU networks is a direction for future work.

2) *Final-layer regularization $\sum_i |\alpha_i|^L$ and raising regularization to L* (4): These choices are used in a rescaling step (Lemma E.1) that simplifies the training regularization to a form of $\|\alpha\|_1$. This $\|\alpha\|_1$ regularization is directly assumed in [9], and in [10] with first-layer weight norm constraints. Additionally [6] considers 2-layer networks with $\|\alpha\|_1$ or total variation norm constraints. When $L = 2$, our regularization reduces to $\sum_i |\alpha_i|^L = \|\alpha\|_2^2$. In a similar form to $\sum_i |\alpha_i|^L$, [10] uses $\sum_i |\alpha_i|^p$ regularization for $p = 1, \frac{3}{2}, 2, 3$ for 2-layer networks. Raising the regularization to L reflects the multiplicative properties of a deep network. This is similar in spirit to path regularization, which [11] uses in an analogous rescaling step for deep ReLU networks.

3) *$\frac{\beta}{L}$ regularization coefficient* (4): This is a normalization chosen so that the resulting Lasso problem has regularization parameter β without additional scaling; it can otherwise in practice be absorbed into β . Our results also apply when $\beta \rightarrow 0$ by using the minimum norm problem (20) in Section D-A, and generalize to other convex loss functions.

II. PRIOR WORK

This work combines convex optimization theory and geometric algebra to study neural network learning. Convex analysis of neural networks has been studied in [6], [9], [12]. However, these works assume that networks have infinite width, limiting their practical application. Recently, the training problem for finite-width, shallow ReLU networks has been shown to be equivalent to a convex reformulation with group ℓ_1 -norm regularization [13]. Performing analysis in a similar spirit to [13], the work in [7] is most related to our convexification approach and is described next in Section II-A. Using geometric algebra has been explored for neural networks in the context of having neurons perform geometric algebra operations or representing data using geometric algebra [14], [15]. Our work is of a different nature, using conventional, real-valued neuron weights and operations but interpreting the learned features using geometric algebra. This is explored in [8], which is the most relevant to our work with geometric algebra and is discussed next in Section II-A.

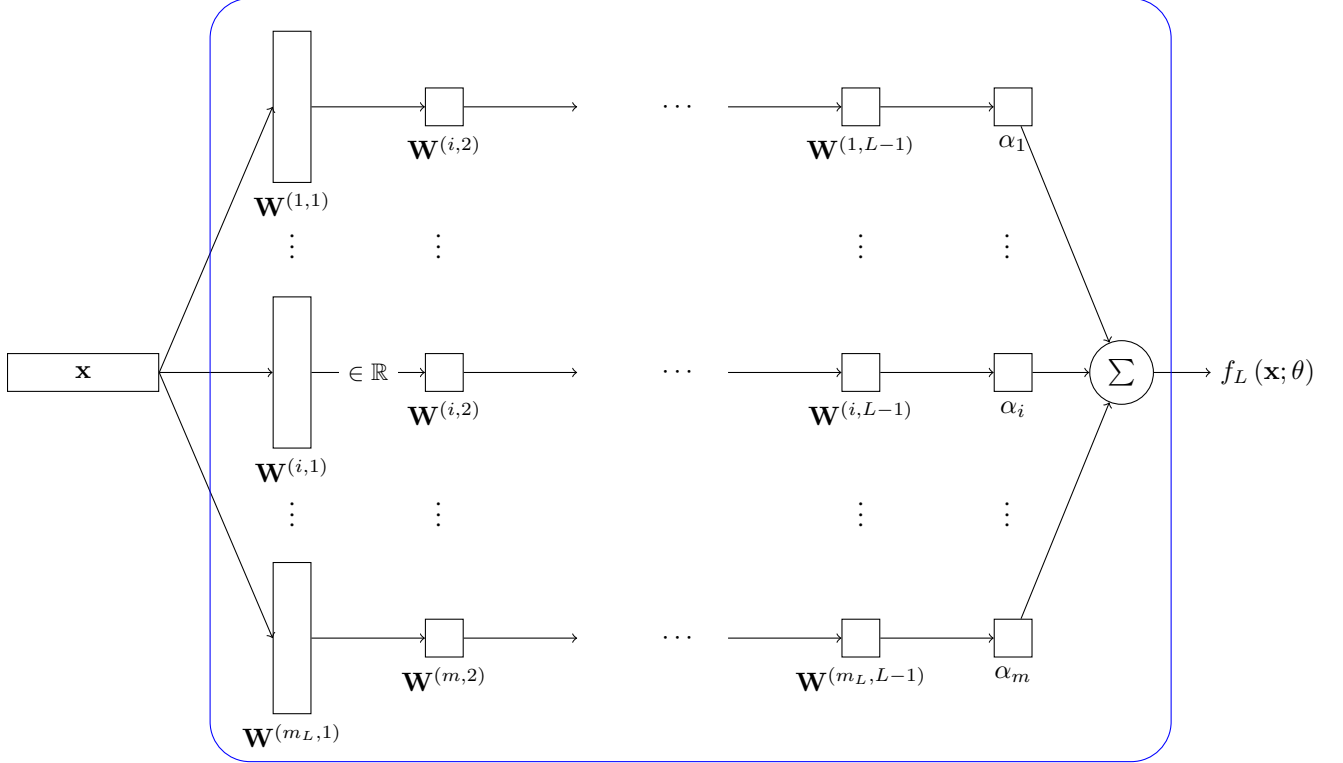


Fig. 1: Diagram of a deep narrow network (enclosed by the blue rectangle) with $\mathbf{W}^{(i,1)} \in \mathbb{R}^d$ and all other weights as scalars. The first layer of each unit i compresses input $\mathbf{x} \in \mathbb{R}^{1 \times d}$ to a scalar $\mathbf{x}\mathbf{W}^{(i,1)} \in \mathbb{R}$.

A. Specific relation of Contributions in Section I-B to prior work in [7] and [8]:

[7] shows equivalences between deep narrow networks with 1-D input data and Lasso models representing reflections, and [8] uses geometric algebra to formulate 2 and 3-layer ReLU networks as Lasso models. [7] shows Item 1 in the above Contributions for 1-D inputs and our work extends this to arbitrary input dimension. [7] finds reflection features for absolute value activation as a special case of Item 2. We generalize this to a broad class of activations by introducing normalized reflections, which Theorem IV.5 shows are learned by 3-layer leaky ReLU networks.

This work also introduces a novel interpretation of features as distances between reflection planes. Related to Item 3, [7] proves that 3-layer and 4-layer deep narrow networks with 1-D inputs and absolute value activation learn reflections and double reflections, respectively, and empirically observes more complex reflections in 5-layer networks. Section IV-B rigorously formalizes these observations by introducing the concept of a reflection order that generalizes single and double reflections to higher levels of reflections. Theorem IV.9 proves quantitative results for the reflection orders represented by features in arbitrarily deep but finite networks, showing that the reflection complexity grows linearly with the number of layers in a deep narrow network. The connection to Cantor sets in Item 3 is not covered even for the univariate case in [7]. [8] shows Item 4 for ReLU activations, and Theorem IV.2 extends it to general classes of leaky ReLU and absolute value activations.

Unlike [7] and [8], this work also shows that a network’s equivalent Lasso model contains a *sparsity factor* $r(\mathbf{w})$ defined as the ratio of l_2 and l_1 norms of a vector $\mathbf{w}$, that encourages parsimonious solutions (Theorem IV.6). This sparsity factor has also been studied in [16], [17]. Although our work focuses on proving that reflection planes and features arise through convex equivalences in even simple architectures, as discussed in Item 5, experiments in Section VI show that LLMs may learn similar structures, an application that is not addressed in [7].

B. Linear deep narrow networks

A version of the deep narrow network but with linear activations, termed “diagonal linear neural networks”, has been studied in [18]–[24] to show that during training (gradient descent or flow, or their stochastic counterparts), networks are implicitly biased to learn regularized or minimal-norm and sparse models, depending on initialization and depth [18]–[20], step size [21]–[23], momentum [23], and stochasticity [24]. Our deep narrow networks add standard nonlinear activations on the diagonal structure, making them more realistic in practice. These prior works study the gradient descent and gradient flow training of a non-regularized problem, while our work studies a more realistic regularized training problem and focuses on showing its equivalence to a convex training problem, which enables direct insight into the globally optimal solution regardless of initialization, without requiring assumptions such as infinitesimal step size for gradient flow, and providing well-studied convex optimization theory addressing the effect of hyperparameters such as step size and momentum coefficients. While the diagonal linear networks studied above exhibit implicit training regularization and our Lasso formulations also reveal a form of regularization as a ℓ_1 penalty on features, these two regularization effects are fundamentally different. Implicit training regularization is tied to training dynamics which depend on training hyperparameters as described above, while our Lasso regularization is a property of globally optimal solutions, independent of the training process.

C. Symmetries in deeper networks

Prior works [25]–[29] show that deeper networks can more effectively represent certain hierarchical or symmetric functions. [25] shows that composing a function $f : \mathbb{R} \rightarrow \mathbb{R}$ with a class of triangular-like functions will repeat and scale f , and uses this property to construct exponentially oscillatory functions that can be expressed by deep 1-D-input networks but not shallow ones unless they have an exponentially large width - a “depth separation” property [26], [30]. [29] shows that deeper networks better represent high-frequency functions using Fourier theory and representing cosines with ReLUs. [30] shows that when deep ReLU networks are trained with gradient descent to fit Cantor-like fractal data, the fraction of the training data that lies on the “coarse” details of the fractal compared to the finer details determines how well the networks can approximate the data distribution.

These prior works generally focus on expressivity, i.e., what functions can be represented by neural networks and to what accuracy depending on the network size, without considering weight regularization. While Theorem IV.9 also presents a class of increasingly complex functions that deeper networks may learn more effectively, Theorem IV.9 considers a regularized training problem, which more closely models training networks in practice. In addition, unlike these prior works, we derive a relationship between depth and symmetry from convex optimization theory. For example, like [30], Theorem V.1 analyzes how absolute value networks learn Cantor data, but while [30] focuses

on gradient algorithms to fit non-convex neural networks to the data and providing estimation bounds, Theorem V.1 finds features that are defined by exact convex equivalences to the training problem. Moreover, while [25], [29] focus on depth separation properties for evenly spaced oscillations, Theorem IV.9 shows that deeper networks with absolute value activations can learn reflections forming multi-level symmetries. The prior works [25]–[29] suggest the network symmetries shown in this paper for deep narrow networks with leaky ReLU activations may extend to more general, wider ReLU networks.

The geometric structure of LLMs has also been explored through other lenses such as the linear representation hypothesis, which posits that LLMs represent basic semantic concepts as linear directions [31], [32], or vectors [33] where independent concepts are orthogonal to each other. [33] extends this hypothesis to show that categorical concepts can be represented as polytopes, which are arranged orthogonally for hierarchical concepts. While [32], [33] focus on geometric structures induced by the final softmax layer of LLMs without considering the internal architecture, and do not guarantee that trained models in practice achieve precisely these structures, our work proves exact results with geometric interpretations considering the entire architecture of much simpler models, namely deep narrow networks.

III. PRIOR WORK ON SHALLOW NETWORKS, AND GEOMETRIC ALGEBRA BACKGROUND

This section introduces the equivalences established in prior work between simple 2-layer networks and Lasso problems. The Lasso problems have features representing volumes of parallelotopes spanned by training data. The convex Lasso problem is *equivalent*² to the non-convex training problem in that they have the same optimal value and a network that is optimal in the training problem can be reconstructed from an optimal Lasso solution. In all of the Lasso equivalence results, assume that $N > d$ and $\mathbf{X}$ has full column rank [8] (there are enough samples relative to their dimension) [8] and the number of neurons in the final layer satisfies $m_L \geq \|\mathbf{z}^*\|_0$, where $\mathbf{z}^*$ is a solution to the Lasso problem and $\|\mathbf{z}^*\|_0$ denotes the number of non-zero elements in $\mathbf{z}^*$. [7] shows that

Theorem III.1 ([7]). *The training problem for a 2-layer network with a piecewise linear activation σ and 1-D data is equivalent to a Lasso problem with dictionary elements*

$$\mathbf{A}_{i,j} = \sigma(x_i - x_j). \quad (6)$$

For $\sigma(x) = (x)_+$, the dictionary elements $(x_i - x_j)_+$ in Theorem III.1 measure the positive distance between training samples x_i and x_j , or equivalently, their 1-D volume. To extend this volume interpretation to higher dimensions, [8] uses the *generalized cross product* from geometric algebra, defined for vectors $\mathbf{v}_1, \dots, \mathbf{v}_{d-1} \in \mathbb{R}^d$ as

$$\times_{i=1}^{d-1} \mathbf{v}_i = \sum_i^{d-1} (-1)^{i-1} |V_i| e_i \quad (7)$$

where $e_i \in \mathbb{R}^d$ is the i^{th} canonical basis vector, $V_i = [\mathbf{v}_1, \dots, \mathbf{v}_{i-1}, \mathbf{v}_{i+1}, \dots, \mathbf{v}_{d-1}]$ is a $(d-1) \times (d-1)$ square matrix, and $|V_i|$ is the determinant of V_i [34], [8]. It can be shown that the generalized cross product is a vector that is orthogonal to $\mathbf{v}_1, \dots, \mathbf{v}_{d-1}$. It can also be shown that the l_2 norm of the generalized cross product of $\mathbf{v}_1, \dots, \mathbf{v}_{d-1}$

²this use of the term "equivalent" is taken from its typical use in optimization theory

gives the $d-1$ volume of the parallelotope spanned by $\mathbf{v}_1, \dots, \mathbf{v}_{d-1}$, while the inner product of $\times_{i=1}^{d-1} \mathbf{v}_i$ with a vector $\mathbf{v}_d$ gives the volume of the parallelotope spanned by $\mathbf{v}_1, \dots, \mathbf{v}_d$ [34]. Using the generalized cross product, [8] generalizes Theorem III.1 to higher input dimensions as

Theorem III.2 ([8]). *The training problem for a 2-layer ReLU network with zero bias parameters and data of any dimension is equivalent to a Lasso problem with dictionary elements*

$$\mathbf{A}_{i,j} = \frac{\text{Vol}_+(\mathbf{x}_i, \mathbf{x}_{j_1}, \dots, \mathbf{x}_{j_{d-1}})}{\|\mathbf{x}_{j_1} \times \dots \times \mathbf{x}_{j_{d-1}}\|_1} = \frac{(\mathbf{x}_i \times \mathbf{x}_{j_1} \times \dots \times \mathbf{x}_{j_{d-1}})_+}{\|\mathbf{x}_{j_1} \times \dots \times \mathbf{x}_{j_{d-1}}\|_1} \quad (8)$$

where the multi-index $j=(j_1, \dots, j_{d-1})$ indexes over all combinations of $d-1$ samples of $\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{e}_1, \dots, \mathbf{e}_d$.

In Theorem III.2, Vol_+ refers to the positive part of the signed volume of the parallelotope spanned by its input vectors. The next section extends these results to deeper layers via deep narrow networks.

Remark III.3. *In geometric algebra, the generalized cross product is related to the wedge product $\wedge$, which represents taking signed areas of polytopes spanned by vectors. It can be shown that the generalized cross product (7) can be written as*

$$\times_{i=1}^{d-1} \mathbf{v}_i = \star \wedge_{i=1}^{d-1} \mathbf{v}_i \quad (9)$$

where $\star$ is an operator representing the orthogonal complement of the span of its input vectors. Instead of relying on determinants (7), the wedge product for higher-dimensional objects can be defined systematically from their actions on basis vectors using associative and distributive laws. Section B provides more background on geometric algebra.

IV. MAIN RESULTS

In this section, we state our main results on the equivalence between training deep narrow networks and Lasso problems with geometric features. In contrast to shallow networks (Section III), the deeper layers allow features to represent reflections of training data for non-monotone activations.

A. 3-layer neural networks

We show that 3-layer deep narrow networks are equivalent to Lasso problems with discrete (finite) dictionaries. Proofs are deferred to Section F and Section G. We consider the leaky ReLU activation $\sigma(x)=a(x)_+ + b(-x)_+$, where $(x)_+ = \max\{x, 0\}$ and $a>0$ (analogous results hold if $a<0$.) Given $a > 0$, the leaky ReLU is (strictly) non-monotone if $b > 0$. Examples include absolute value ($a=b=1$) but not ReLU ($a=1, b=0$). We associate midpoints and reflections to the slopes of non-monotone leaky ReLU in the following definition.

Definition IV.1. *Let $a, b > 0, \sigma(x) = a(x)_+ + b(-x)_+$. With respect to σ , the normalized midpoint $m_{\alpha,\beta}^\sigma$ of α and β , and normalized reflection $R^\sigma(\alpha, \beta)$ of α about β , are*

$$m_{\alpha,\beta}^\sigma = -\frac{a \max\{\alpha, \beta\} + b \min\{\alpha, \beta\}}{a + b}, \quad R^\sigma(\alpha, \beta) = \begin{cases} \beta + \frac{b}{a}(\beta - \alpha) = \frac{(a+b)}{a}\beta - \frac{b}{a}\alpha & \text{if } \beta \leq \alpha \\ \beta + \frac{a}{b}(\beta - \alpha) = \frac{(a+b)}{b}\beta - \frac{a}{b}\alpha & \text{else.} \end{cases} \quad (10)$$

For $\sigma(x)=|x|$, the normalized midpoint is the usual midpoint $\frac{\alpha+\beta}{2}$ and the normalized reflection reduces to the usual reflection $R(\alpha, \beta)=\beta+(\beta-\alpha)=2\beta-\alpha$. The ratio of a to b determines how far the reflection of α overshoots or undershoots β . The coefficients in front of β and α sum to 1. The next result gives the equivalence between the training problem (4) for 3-layer deep narrow networks (3) and Lasso models (5), using normalized reflections and midpoints (Definition IV.1). Given data subsequence indices $k \in [2(d-1)]$, data indices $j_{-1}, j_0, j_{2k} \in [N]$ and augmented data $\mathbf{x}'_{j_1} \in \{m_{\mathbf{x}_{j_{-1}}, \mathbf{x}_{j_0}, \mathbf{x}_{j_{-1}}}\}$, $\mathbf{x}'_{j_{2k+1}} \in \{\mathbf{x}'_{j_1}, R^\sigma(\mathbf{x}_{j_0}, \mathbf{x}'_{j_1})\} \cup \{\mathbf{x}_{j_{2k}} - e_l\}_{l \in [d]}$, define the ordered set of augmented data $\mathcal{W} = \{\mathbf{x}_{j_i} - \mathbf{x}'_{j_{i+1}} : 2 \leq i \leq 2(d-1)\}$. The next result applies to networks with leaky ReLU activations that are not monotone. This includes absolute value activations, but not ReLU. Analogous results for ReLU are given in [8]. Optimal Lasso solutions are denoted as z_i^* and ξ^* . The result uses the generalized cross product (7).

Theorem IV.2. *The training problem for a 3-layer deep narrow network with non-monotone leaky ReLU activation and d -dimensional data is equivalent to a Lasso problem with dictionary elements $\mathbf{A}_{i,j} = f_j(\mathbf{x}_i)$ where $\mathbf{x}_i$ is the i^{th} training sample and*

$$f_j(\mathbf{x}) = \frac{\sigma(s(V(\mathbf{x}) - V(\mathbf{x}_{j_0})))}{\|\times_{\tilde{\mathbf{x}} \in \mathcal{W}} \tilde{\mathbf{x}}\|_1} \quad \text{where } V(\mathbf{x}) = \sigma(\text{Vol}_s(\{\mathbf{x} - \mathbf{x}'_{j_1}\} \cup \mathcal{W})) \quad (11)$$

The multi-index $j = (j_{-1}, j_0, j_1, j_2, \dots, j_{2d-1})$ indexes over all possible $\mathcal{W}$ and $s \in \{-1, 1\}$. An optimal neural network can be reconstructed as $f_3(\mathbf{x}; \theta) = \sum_i z_i^ f_j(\mathbf{x}) + \xi^*$, and the corresponding optimal $\mathbf{W}^{(1)}$ is orthogonal to all $\tilde{\mathbf{x}} \in \mathcal{W}$.*

In Theorem IV.2, Vol_s denotes the signed volume of the parallelotope spanned by the argument vectors. Each feature $f_j(\mathbf{x})$ measures a signed difference of volumes defined by training data and their reflections or averages. A similar version of Theorem IV.2 for monotone activations is given in Theorem G.1 in Section G: deep narrow networks with monotone activations are also Lasso problems with volumetric features. The monotone features are simpler than for non-monotone activations, but they do not induce reflections.

Remark IV.3. *For $\sigma(x)=|x|$, the numerator of $f_j(\mathbf{x})$ in (11) simplifies to $|V(\mathbf{x}) - V(\mathbf{x}_{j_0})|$ where $V(\mathbf{x}) = \text{Vol}(\{\mathbf{x} - \mathbf{x}'_{j_1}\} \cup \mathcal{W})$ is the unsigned volume.*

Theorem IV.2 gives a novel characterization of optimal networks as Lasso solutions. Theorem IV.2 states that we can train a globally optimal network by constructing a dictionary matrix, solving the Lasso problem (5) using well-known techniques [5], and then reconstructing an optimal network. The dictionary is constructed from differences of points in an *augmented* training data set $\mathcal{W}$ composed of normalized reflections and averages of training data. For shorthand, we may refer to these augmented data points as the training data. Augmenting data with averages has been shown to improve the performance of training [35], and the Lasso equivalence may suggest theoretical justification for this phenomenon. We now discuss several advantages of using the Lasso problem.

Training benefits: The convexity of the Lasso problem ensures global optimality. In non-convex problems, training can get stuck in local, suboptimal optima depending on the initialization. The convex Lasso model avoids this problem. Additionally, using the Lasso model reduces the need to tune hyperparameters such as the number of neurons m , since $m = \|\mathbf{z}^*\|_0$ suffices.

Uniqueness: The training problem and Lasso problems may not have unique solutions. [7] discusses the relationship between the solution set of the training problem with 1-D data and the set of neural networks reconstructed from all Lasso solutions. Analysis of the Lasso solution sets and stationary points in the training problem for higher-dimensional data is an area for future work.

Complexity: The complexity of finding a Lasso solution with a dictionary of size $N \times F$ is $O(N^2 F)$ [5], which is linear in the number of feature vectors F . F is finite and can be bounded. Note the training problem itself is computationally complex: training the non-convex problem can be viewed as optimizing only a heuristic for the true exponentially hard training problem [36], while the Lasso problem is equivalent to it.

Lemma IV.4. *In Theorem IV.2, the Lasso dictionary $\mathbf{A}$ has $O((Nd)^d)$ feature vectors. The exponential complexity in d cannot be avoided for global optimization of ReLU networks unless $\mathbf{P}=\mathbf{NP}$.*

Lemma IV.4 is an overestimate of the dictionary size, since the Lasso dictionary contains repeated columns and the presence of linearly dependent vectors $\mathbf{x}_{j_k} - \mathbf{x}'_{j_{k+1}}$ (11) spanning a parallelotope will make its volume 0, resulting in a $\mathbf{0}$ vector that does not contribute to the Lasso model. To reduce the computational overhead of creating the dictionary, the features can be subsampled [37]. [36] suggests that subsampling hyperplanes corresponds to achieving local optima. The Lasso model can also be used in a polishing step instead of using it to train the entire model from scratch. Polishing consists of partially training a network with the non-convex training problem, extracting the breakplanes from the neurons to estimate a Lasso dictionary, and then solving Lasso to fine-tune the network [8]. The Lasso problem allows theoretical insight into the solution set and solution path of optimal neural networks reconstructed from them, discussed in Section C.

Interpretability: The ℓ_1 regularization in the Lasso problem selects a minimal number of feature vectors [2] which are discrete samples of features $f_j(\mathbf{x})$. Intuitively, the network *learns* these features, as a linear combination of feature functions corresponds to a parsimonious and optimal network. The features interpolate subsets of the training data. The mapping between the features and optimal weights is in the Appendix (Definition F.4). Theorem IV.2 gives a volumetric interpretation of features. The features also have geometric algebraic and a distance-based interpretations, discussed next. A piecewise linear function $f : \mathbb{R} \rightarrow \mathbb{R}$ has a *breakpoint* at $\mathbf{x}$ if f changes slope at $\mathbf{x}$. A function $g : \mathbb{R}^{1 \times d} \rightarrow \mathbb{R}$ defined by $g(\mathbf{x}) = f(\mathbf{x}\mathbf{w})$ where $\mathbf{w} \in \mathbb{R}^d$ has a *breakplane* along the plane $\{\mathbf{x} \in \mathbb{R}^{1 \times d} : \mathbf{x}\mathbf{w} + b = 0\}$ if f has a breakpoint at $\mathbf{x}\mathbf{w}$. A breakplane is a "kink" in a function's graph. The next result gives a geometric interpretation of the features in terms of normalized reflections (Definition IV.1).

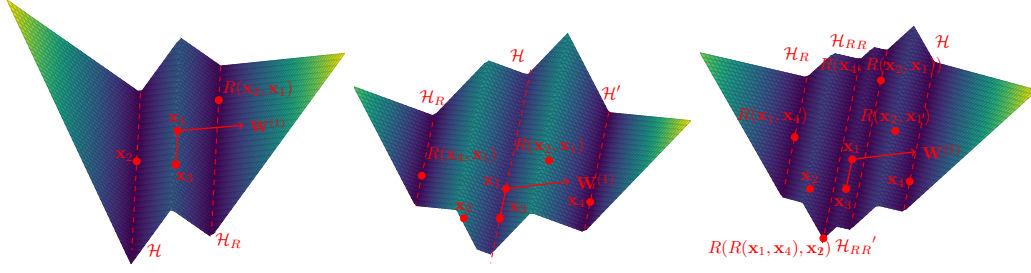


Fig. 2: Examples of 3-layer (left) and 4-layer (middle, right) deep narrow network features for 2-D data. Red dashed lines indicate reflection planes (lines in $\mathbb{R}^2$).

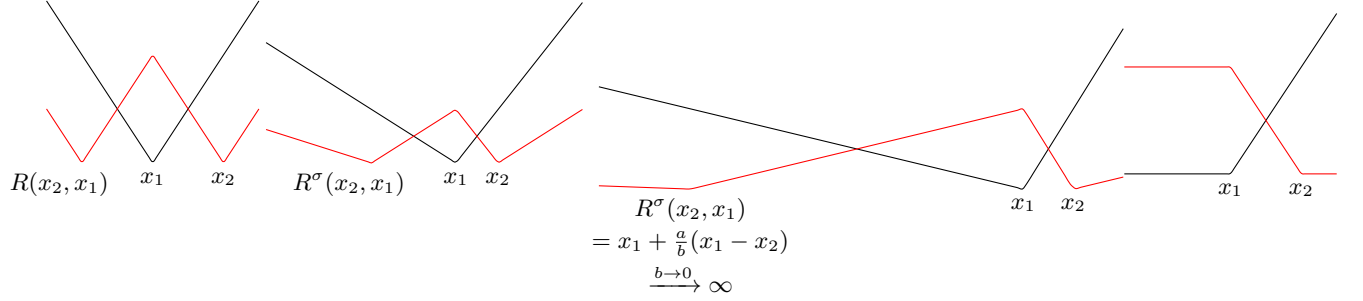


Fig. 3: Example illustrations of 2-layer (black) and 3-layer (red) features in deep narrow networks with absolute value (left), leaky ReLU (middle two), and ReLU (right) activations. As leaky ReLU approaches ReLU ($b \rightarrow \infty$), so do their features, with the normalized reflections $R^\sigma(x_2, x_1)$ moving to infinity for ReLU, meaning no reflections.

Theorem IV.5. *The 3-layer non-monotone leaky ReLU dictionary contains features with breakplanes based at training samples and normalized reflections of training data: $(\mathbf{x} - R^\sigma(\mathbf{x}_i, \mathbf{x}_j)) \mathbf{w}^{(1)} = 0$ for some training samples $\mathbf{x}_i, \mathbf{x}_j$.*

Theorem IV.5 shows that the features of a non-monotone leaky ReLU network are piecewise linear functions whose linear pieces fold along "creases" determined by the normalized reflection. The left plot of Figure 2 illustrates these folds for $\sigma(x) = |x|$. To investigate this idea, we focus on absolute value activation for tractability. A vector $\mathbf{x}$ is *sparse* if it has few non-zero elements. The *sparsity factor* of $\mathbf{x} \neq \mathbf{0}$ is $r(\mathbf{x}) = \|\mathbf{x}\|_2 / \|\mathbf{x}\|_1 \in [1/\sqrt{N}, 1]$. The sparsity factor measures a vector's sparseness: if $\mathbf{x}$ has one non-zero element, then $r(\mathbf{x}) = 1$ and if $\mathbf{x}$ is parallel to $\mathbf{1}$, then $r(\mathbf{x}) = 1/\sqrt{N}$. The more sparse $\mathbf{x}$ is, the larger its sparsity factor.

Next, given $j_{-1}, j_0 \in [N]$, $\mathbf{w} \in \mathbb{R}^d$, define the hyperplanes $\mathcal{H} = \{\mathbf{x} \in \mathbb{R}^d : (\mathbf{x} - \mathbf{x}_{j_{-1}}) \mathbf{w} = 0\}$ containing sample $\mathbf{x}_{j_{-1}}$ and $\mathcal{H}_0 = \{\mathbf{x} \in \mathbb{R}^d : (\mathbf{x} - \mathbf{x}_{j_0}) \mathbf{w} = 0\}$ containing $\mathbf{x}_{j_0}$. The *average* of the hyperplanes $\mathcal{H}$ and $\mathcal{H}_0$ is the hyperplane $\mathcal{H}_A = \{\mathbf{x} \in \mathbb{R}^d : (\mathbf{x} - (\mathbf{x}_{j_{-1}} + \mathbf{x}_{j_0})/2) \mathbf{w} = 0\}$. The *reflection* of the hyperplanes $\mathcal{H}$ across $\mathcal{H}_0$ is $\mathcal{H}_R = \{\mathbf{x} \in \mathbb{R}^d : (\mathbf{x} - R(\mathcal{H}_{j_{-1}}, \mathcal{H}_{j_0})) \mathbf{w} = 0\}$. The next result (Theorem IV.6) states that 3-layer absolute value features represent distances scaled by the sparsity factor to parallel planes $\mathcal{H}_A$ containing averages and $\mathcal{H}_R$ containing reflections of training data. These features are linked to geometric algebra through an explicit formulation using

wedge products, generalized cross products, and Hodge star operations from (7) and Remark III.3. Denote the distance from $\mathbf{x}$ to $\mathcal{S} \subset \mathbb{R}^d$ as $\text{Dist}(\mathbf{x}, \mathcal{S})$.

Theorem IV.6. *The 3-layer features for $\sigma(x)=|x|$ in Theorem IV.2 can be written as*

$$f_j(\mathbf{x}) = r(\mathbf{w}) \begin{cases} \text{Dist}(\mathbf{x}, \mathcal{H} \cup \mathcal{H}_R) & \text{if } \mathbf{x}'_{j_1} = \mathbf{x}_{j-1} \\ \text{Dist}(\mathbf{x}, \mathcal{H} \cup \mathcal{H}_A) & \text{else} \end{cases} = \frac{|\star W(\mathbf{x})| - |\star W(\mathbf{x}_{j_0})|}{\|\bigwedge_{\tilde{\mathbf{x}} \in \mathcal{W}} \tilde{\mathbf{x}}\|_1} \quad (12)$$

where $\mathbf{w} = \left(\bigotimes_{\tilde{\mathbf{x}} \in \mathcal{W}} \right)^T$, and $W(\mathbf{x}) = (\mathbf{x} - \mathbf{x}'_{j_1}) \wedge \left(\bigwedge_{\tilde{\mathbf{x}} \in \mathcal{W}} \tilde{\mathbf{x}}\right)$. An optimal $\mathbf{W}^{(1)}$ is a scalar multiple of $\mathbf{w}$.

The proof is in Section F. Equation (12) states the 3-layer dictionary for $\sigma(x)=|x|$ includes *reflection features* that measure the minimum distance to two parallel *reflection planes*: $\mathcal{H}$, which contains $\mathbf{x}_{j-1}$, and $\mathcal{H}_R$, which is the reflection of $\mathcal{H}$ across the hyperplane $\mathcal{H}_0$ containing $\mathbf{x}_{j_0}$. Since distance is an unsigned value, the network induces a symmetry about the two planes. These planes are spanned by a subset of training data and are orthogonal to $\mathbf{W}^{(1)}$. Figure 9 illustrates an example of reflection planes $\mathcal{H}, \mathcal{H}_R$ in $\mathbb{R}^3$. The right plot of Figure 2 graphs a feature $f_j(\mathbf{x})$ in $\mathbb{R}^2$. The feature outputs the distance between the reflection "planes" $\mathcal{H}, \mathcal{H}_R$, which are lines in $\mathbb{R}^2$ and are plotted as red, dashed lines. These are axes of symmetry encoded in 3-layer features. In the notation used in Theorem IV.2, Figure 9 and the right plot of Figure 2 both graph the case where $\mathbf{x}'_{j_1} = \mathbf{x}_1, \mathbf{x}_{j_0} = \mathbf{x}_2, \mathbf{x}_{j_2} = \mathbf{x}_3, \mathbf{x}'_{j_3} = \mathbf{x}_1$, (and additionally $\mathbf{x}_{j_4} = \mathbf{x}_4, \mathbf{x}'_5 = \mathbf{x}_1$ for Figure 9). We conclude this section on 3-layer networks by giving some intuition for ReLU features (Section IV-A1) and how to interpret reflection planes (Section IV-A2). These two discussions are informal, and rigorous analysis will be explored in future work.

1) *ReLU features*: As Theorem IV.5 and Theorem IV.6 show that leaky ReLU and especially absolute value features can encode normalized/reflections, a natural question is, what about ReLU features? Loosely speaking, the sequence of non-monotone leaky ReLUs $\sigma_n(x) = (x)_+ + \frac{1}{n}(-x)_+$ approaches ReLU as $n \rightarrow \infty$. In this process, leaky ReLU features may approach ReLU features.

First, consider 3-layer, 1-D deep narrow networks. [7] shows that such ReLU features (of arbitrary depth) do not have reflections. Specifically, 3-layer deep narrow network networks with ReLU activations have features $f(x) = ((x - x_i)_+ - (x_j - x_i)_+)_+$ that are shaped as "capped ramps" with only 2 breakpoints, as illustrated in the right-most graph of Figure 3. As shown in Figure 3, for deep narrow networks, leaky ReLU features with normalized reflections interpolate between absolute value features with reflections and ReLU features with no reflections. For a leaky ReLU activation with slopes $a = 1$ and $b > 0$, as $b \rightarrow 0$, (10) shows that the normalized reflection point $R^\sigma(\alpha, \beta) \rightarrow \infty$ for $\beta \geq \alpha$ and $R^\sigma(\alpha, \beta) \rightarrow \beta$ for $\beta \leq \alpha$. The location of the normalized reflection (10) in a leaky ReLU feature tends to infinity or the point being reflected about, so that the leaky ReLU feature approaches a ReLU feature with no reflections. The capped ramps can be viewed as limits of the features for leaky ReLU.

This transition from leaky ReLU to ReLU intuitively suggests why ReLU features in deep narrow networks don't contain reflections. Nonetheless, any approximating (non-monotone) leaky ReLU that is used instead of ReLU in practice will have reflections (albeit at potentially far locations). A wider network can induce ReLU reflections as shown in [7]. Intuitively speaking, the above observations would extend to higher dimensions such that as leaky

ReLU approaches ReLU activation, normalized reflection breakplanes would slide farther away from their parallel breakplanes at the training data, until they are at distance infinity away and never occur.

2) *Reflection plane interpretation:* Informally, we can interpret reflection planes in Theorem IV.6 as *concepts* [33] capturing symmetries or analogies in the data. Each plane is spanned by a subset of training data ($\mathcal{W}$ in Theorem IV.6) which are part of the concept. The feature outputs the same value for inputs that are translated in parallel to the concept planes. Thus the features can be viewed as measuring the similarity in orientation between an input and the concept. A feature will also output the same value for data reflected across $\mathcal{H}_0$, which can be interpreted as recognizing structural similarities or analogies between concepts, with $\mathcal{H}_0$ forming an axis of symmetry. With different values of $\mathbf{x}_{j_0}, \mathbf{x}_{j_{-1}}$, the network searches for symmetries across various pairs of training points. Intuitively, the concept plane $\mathcal{H}$ forms a local "universe" or coordinate system centered at an origin located at $\mathbf{x}_{j_{-1}}$, and relative to this new origin, the feature measures the similarity between an input and the concept, via a dot product with the vector $\mathbf{W}^{(1)}$ representing the orientation of $\mathcal{H}$. There is an analogous "parallel universe" centered at the reflection of $\mathbf{x}_{j_{-1}}$ across $\mathbf{x}_{j_0}$, where the feature measures the distance between the input and the reflected concept. The feature outputs the closer of the two similarities between the universes. Theorem IV.6 shows that the reflection hyperplanes represent breakplanes of 3-layer networks and create axes of symmetry. The reflection breakplanes generalize to deeper networks, discussed next.

B. Deeper networks

Here, we extend a Lasso equivalence to deep narrow networks of arbitrary depth, which are similarly defined as their 3-layer version (3), formally shown in Section E.

Definition IV.7. A feature for an L -layer deep narrow network with any activation is a parameterized function $f : \mathbb{R}^{1 \times d} \rightarrow \mathbb{R}$ defined as $f(\mathbf{x}) = \mathbf{X}^{(L)}(\mathbf{x})$ where $\mathbf{X}^{(l)}$ is defined as in (15) and $\mathbf{W}^{(l)} \in \{-1, 1\}$ for $l > 1$. A feature with bias parameters of the form $\mathbf{b}^{(l)} = -\mathbf{X}^{(l)}(\mathbf{x}_{j_l})\mathbf{W}^{(l)}$ with $j_l \in [N]$ is denoted as f_j where $j = (j_1, \dots, j_{L-1})$, and its bias parameters are called data feature biases.

In Definition IV.7, the multi-index $j = (j_1, \dots, j_{L-1})$ defines the bias parameters of the feature f_j by specifying the training data $\mathbf{x}_{j_l}$ used in the data feature biases $\mathbf{b}^{(l)} = -\mathbf{X}^{(l)}(\mathbf{x}_{j_l})\mathbf{W}^{(l)}$. The *data feature sub-library* is a set of feature functions with data feature biases. As an example, for $\sigma(x) = |x|$ and 1-D input data $x_1, \dots, x_N$, the features in the data feature sub-library for an L -layer network take the form $f^{(L)}(x)$ with the recursive structure

$$f^{(l+1)}(x) = |f^{(l)}(x) - f^{(l)}(x'_l)| \text{ for } l > 2 \text{ and } f^{(2)}(x) = |x - b_1|, \quad x'_l \in \{x_1, \dots, x_N\}. \quad (13)$$

Section V will provide an explicit example where the features (13) encode well-defined multi-level symmetries for certain exponentially distributed datasets. Features for $\sigma(x) = |x|$, arbitrary input dimension and general non-data feature biases are detailed in (F-A) and Remark F.3. The next result states that deep narrow networks of *any* depth and input dimension are implicit Lasso models whose dictionary contains sampled versions of feature functions defined in Definition IV.7.

Theorem IV.8. *The training problems for deep narrow networks with leaky ReLU activation, arbitrary depth and input dimension are equivalent to Lasso problems with finite, discrete dictionaries. The set of feature vectors contains a data feature sub-library and consists of feature functions (58) sampled at training data points. The full dictionary for deep narrow networks of depth $L > 3$ may contain additional features beyond this sub-library. An optimal network can be reconstructed as $\sum_j z_j^* f_j(\mathbf{x}) + \xi^*$.*

The proof is in Section G. Definition F.4 describes a mapping between features and optimal weights. Theorem IV.8 holds for both monotone and non-monotone activations. The data feature sub-library is a set of novel features that can represent multi-level structures, especially for non-monotone activations. The most symmetric structures are for $\sigma(x) = |x|$, which generates reflections of increasing complexity, or *order*. The *standard reflections* $R(\mathbf{x}_0, \mathbf{x}_1)$ and $R(\mathbf{x}_1, \mathbf{x}_0)$ create a single-level symmetry of the point being reflected and its reflection. An *order-2* reflection is a reflection of a reflection, which induces a *second-level symmetry* around the reflection. Formally, an *order-0* reflection of a point $\mathbf{x}_0 \in \mathbb{R}^d$ is simply $R_0(\mathbf{x}_0) = \mathbf{x}_0$. Given points $\mathbf{x}_0, \mathbf{x}_1$, the *standard reflections* $R(\mathbf{x}_0, \mathbf{x}_1)$ and $R(\mathbf{x}_1, \mathbf{x}_0)$ are *order-1* reflections. In general for $k > 0$, the *order- k reflection* of $\mathbf{x}_0, \dots, \mathbf{x}_k \in \mathbb{R}^d$ is of the form $R_k(\mathbf{x}_0, \dots, \mathbf{x}_k) \in \{R(R_{k-1}(\mathbf{x}_0, \dots, \mathbf{x}_{k-1}), \mathbf{x}_k), R(\mathbf{x}_k, R_{k-1}(\mathbf{x}_0, \dots, \mathbf{x}_{k-1}))\}$. Figure 10 in the Appendix plots examples of reflections of order 0, 1 and 2 in $\mathbb{R}^2$. As seen in Figure 10, in $\mathbb{R}^2$, $R(\mathbf{a}, \mathbf{b})$ is the point on the line between $\mathbf{a}$ and $\mathbf{b}$, whose distance from $\mathbf{b}$ is the same as the distance from $\mathbf{b}$ to $\mathbf{a}$, creating a symmetry.

Theorem IV.6 states that the features for a 3-layer network measure distances to reflection planes, which include breakplanes at first-order reflections. The left plot of Figure 2 illustrates an example of a 3-layer feature. The weight $\mathbf{W}^{(1)}$ is orthogonal to $\mathbf{x}_3 - \mathbf{x}_1$ and the breakplanes occur along $\mathcal{H}_1 = \{(\mathbf{x} - \mathbf{x}_1) \mathbf{W}^{(1)} = 0\}$, $\mathcal{H} = \{(\mathbf{x} - \mathbf{x}_2) \mathbf{W}^{(1)} = 0\}$ and $\mathcal{H}_R = \{(\mathbf{x} - R(\mathbf{x}_2, \mathbf{x}_1)) \mathbf{W}^{(1)} = 0\}$. The height of the graph is the minimum of the distance from the reflection planes $\mathcal{H}$ and $\mathcal{H}_R$. Similarly, 4-layer features can have breakplanes that represent first or second-order reflections, as shown respectively in the middle and right plots, which plots examples of data features. The peaks and troughs of the features plotted in Figure 2 correspond to breakplanes. The troughs are indicated by dashed red lines and depict higher-level symmetries and reflection planes. Deeper networks with non-monotone activations have the capacity for learning increasingly complex reflection features and symmetries.

Theorem IV.9 (Multi-level symmetries). *A deep narrow network of any input dimension, $\sigma(x) = |x|$, and depth L is equivalent to a Lasso problem with a dictionary containing features with no reflections for $L=2$, standard reflections for $L=3$, and up to order-2($L-3$) reflections for $L>3$.*

Section H has the proof. Theorem IV.9 shows that each layer induces higher order reflections and deeper levels of symmetry. Theorem IV.9 also describes the data feature sub-library in Theorem IV.8 and shows that the maximum order of reflections in features grows linearly with depth. A full analysis of the entire dictionary for networks with more than 3 layers is an area for future work. Nonetheless, our approaches and proof techniques demonstrate the principle of how to find all of the dictionary elements for deeper networks, as discussed in the proof of Theorem IV.8.

Multi-scale structures with other activations: Extending Theorem IV.9 to other leaky ReLU functions is an area of future work. Theorem IV.5 states that 3-layer networks with non-monotone leaky ReLU have normalized reflection

features, and a similar proof to Theorem IV.9 would likely apply to their deeper versions, with normalized reflections of higher order. The features for leaky ReLU are not necessarily symmetric, but they can exhibit multi-scale structures as seen in an example in Figure 11.

V. ADVANTAGES OF MULTI-LEVEL SYMMETRIES FOR DATA FITTING: CANTOR FEATURES

We give an example demonstrating the expressivity of deep networks with multi-level symmetries. The Cantor set is the set of points formed by starting with the interval $[0, 1]$ and recursively removing $\frac{1}{3}$ of the middle of each remaining segment [38], [39]. We show an example of a 1-D dataset that induces a feature with similar structure to the Cantor set. We present numerical examples in Section VI-C to show the advantage in expressivity. The Cantor set has deep applications in mathematics and is an example of a fractal, a pattern with self-similar structures. Fractals have applications in areas as diverse as physics [40], image compression [41], computer graphics, and civil engineering [42].

1-D triadic data: Suppose we have 1-D data $x_1 > \dots > x_N$ where $x_n = \frac{1}{2} - \sum_{i=1}^n r^{i-1}$. In other words, $x_1 = \frac{1}{2} > x_2 = x_1 - r > x_3 = x_2 - r^2 > \dots > x_N = x_{N-1} - r^{N-1}$ where $r \in (0, \frac{1}{3}]$. We call this a *r-adic data set*. In particular assume that $r = \frac{1}{3}$: we call the data set *triadic*. Consider the features (13) in the data feature sub-library for absolute value activation and the particular case when $x'_l = x_l$. So $f^{(2)}(x) = |x - x_1|$ and the $(l+1)^{\text{th}}$ feature is $f^{(l+1)}(x) = |f^{(l)}(x) - f^{(l)}(x_l)|$. We call these particular features (13) with $x'_l = x_l$ and triadic data as *Cantor features*, for reasons shown below. Let $\mathcal{G}^{(l)} = \{(x, y) : x \in [0, 1], y = f^{(l)}(x)\}$ be the graph of the Cantor feature $f^{(l)}(x)$ over $x \in [0, 1]$. Define the function $T^{(l)} : [r, 1-r] \rightarrow \mathbb{R}$ parameterized by l as

$$T^{(l)}(x) = \begin{cases} x - r + \frac{1}{2}r^{l-2} & \text{if } x \in [r, \frac{1}{2}] \\ 1 - x - r + \frac{1}{2}r^{l-2} & \text{if } x \in [\frac{1}{2}, 1-r] \end{cases} \quad (14)$$

For $x \in \mathbb{R}$, let $T_L(x) = \frac{x}{3}$ and $T_R(x) = 1 - T_L(x) = \frac{2+x}{3}$. The functions T_L and T_R represent the left and right recursive transformations generating the Cantor set, respectively. In 1-D, the n^{th} Cantor set is defined as $C_n = T_L(C_{n-1}) \cup T_R(C_{n-1})$, where $C_0 = [0, 1]$. The graph of the Cantor feature has a similar recursive structure as the Cantor set, as shown next. With a slight abuse of notation, for $(x, y) \in \mathbb{R}$, let $T_L(x, y) = (\frac{x}{3}, \frac{y}{3})$ and $T_R(x, y) = (1 - \frac{x}{3}, 1 - \frac{y}{3})$. The next result shows that each L -layer deep library for $\sigma(x)=|x|$ contains a *Cantor feature* with the structure of a level $L-2$ Cantor set.

Theorem V.1. *For $l \geq 2$, Cantor features for triadic data can be recursively expressed as $\mathcal{G}^{(l+1)} = T_L(\mathcal{G}^{(l)}) \cup T^{(l)}([r, 1-r]) \cup T_R(\mathcal{G}^{(l)})$.*

The proof is in Section I. Each deeper Cantor feature scales the previous layer's Cantor feature by $\frac{1}{3}$, copies and places the miniaturized feature over $[0, r]$ and $[1-r, 1]$, and bridges the two copies with the function $T^{(l)}$ over $[r, 1-r]$. There is a self-similarity between the Cantor feature at layer l , and the first and last thirds of the Cantor feature at layer $l+1$. The Cantor feature is an example of multi-level symmetry in neural network features.

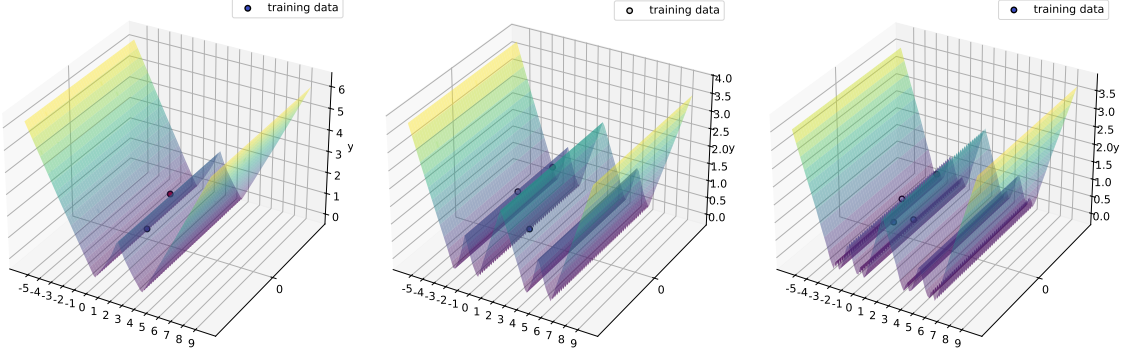


Fig. 4: Adam-trained neural net with absolute value activation and 2-D training data, of depth 3 (left), 4 (middle), and 5 (right). The vertical axis is the target/label. There are breakplanes of order up to $L-2$ at reflections of data points. The 3- and 4-layer networks match the corresponding features shown in the left and right-most plots of Figure 2, respectively.

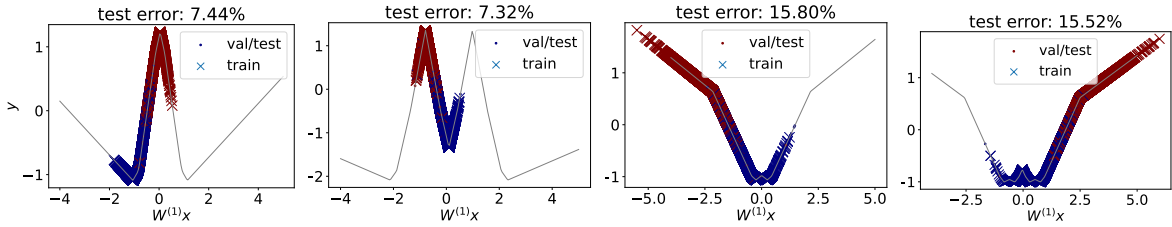


Fig. 5: Projections of 3 (left) and 4-layer (right) networks trained on embeddings of IMDB reviews exhibit multi-level symmetry. True labels of 1 and -1 are red and blue. Vertical axis is the target y .

VI. NUMERICAL RESULTS

We perform experiments on *standard* networks (15), as well as wider networks to show the applicability of insights from Lasso. To show the applicability of our results to more traditional architectures, we use the *standard network* defined recursively as

$$f_L(\mathbf{x}; \theta) = \mathbf{X}^{(L)}\alpha + \xi \text{ where } \mathbf{X}^{(l+1)} = \sigma(\mathbf{X}^{(l)}\mathbf{W}^{(l)} + \mathbf{b}^{(l)}), \quad \mathbf{X}^{(1)} = \mathbf{x} \quad (15)$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times 1}$, $\mathbf{b}^{(1)} \in \mathbb{R}$, $\dots$, $\mathbf{W}^{(l)} \in \mathbb{R}^{m \times m}$, $\mathbf{b}^{(l)} \in \mathbb{R}$, $\mathbf{W}^{(L-1)} \in \mathbb{R}^{1 \times m}$, $\mathbf{b}^{(L-1)} \in \mathbb{R}^{1 \times m}$, $\xi \in \mathbb{R}$, $\alpha \in \mathbb{R}^m$. In contrast to the deep narrow network, the inner weights of the standard network are matrices instead of scalars.

A. Toy data (small d, N): Deep narrow networks with $\sigma(x) = |x|$

We start with training $m = 100$ -neuron standard neural networks on hand-picked toy datasets whose 1-D projections have known, tractable minimum-norm Lasso solutions that consist of individual Lasso features [7]. The networks are shown in Figure 4. The networks resemble the theoretical features drawn in Figure 2. A similar experiment is performed with a ReLU network, which also demonstrates breakplanes at reflections of training data (Figure 12, Section D).

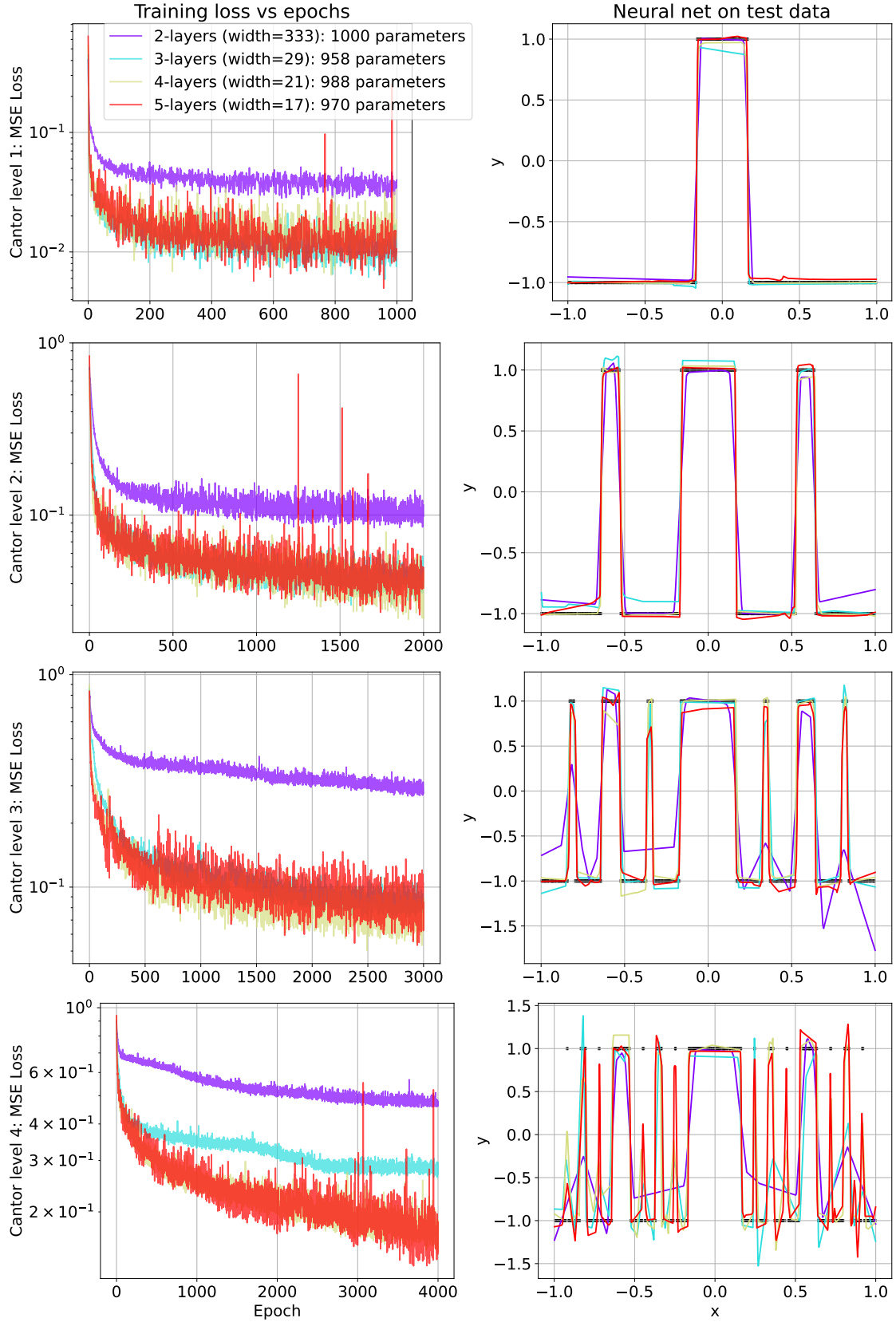


Fig. 6: Examples of neural networks in Figure 7 trained on 1-D Cantor sets of levels 1 – 4. Deeper networks fit the Cantor sets more accurately.

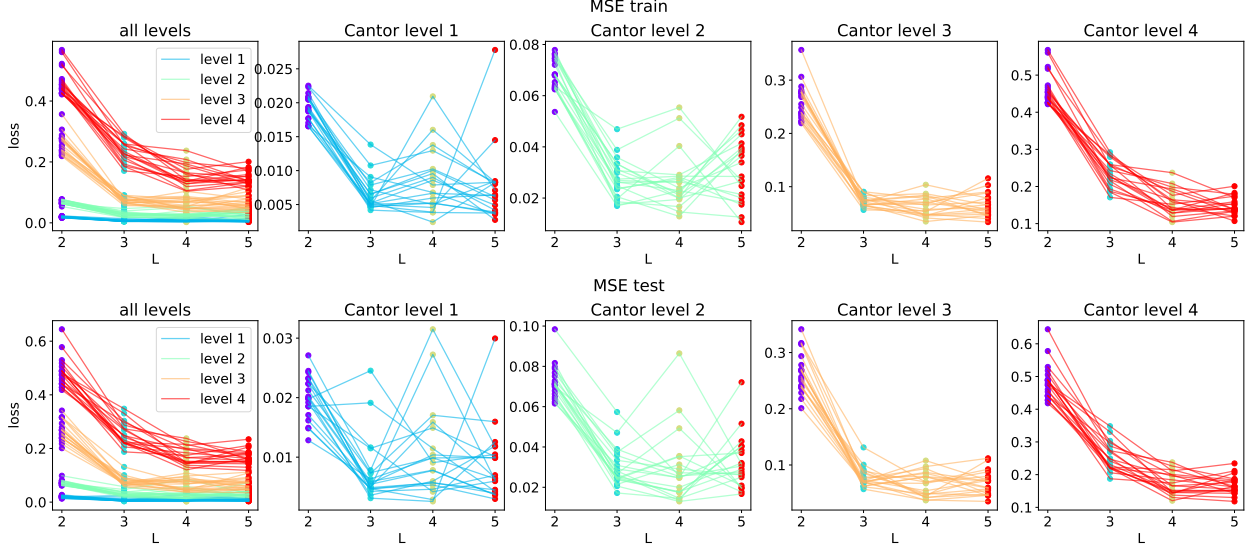


Fig. 7: MSE of neural nets of depths $L \in \{2, 3, 4, 5\}$ trained on Cantor sets of levels 1 – 4. Each curve (a sequence of four connected points) corresponds to the MSE resulting from a different initialization. Examples of networks for each depth and Cantor level are plotted in Figure 6. For larger-level Cantor sets, deeper networks tend to have higher performance (lower MSE).

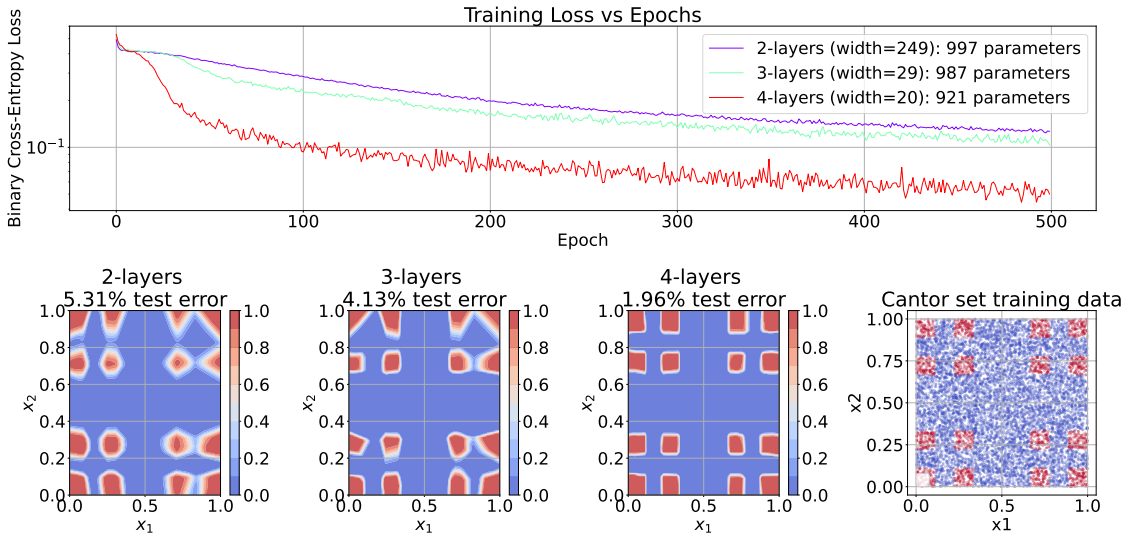


Fig. 8: Neural networks trained on 2-D Cantor sets of level 2. Deeper networks more cleanly fit the Cantor sets. Additional experiments varying the initialization are in Figure 7.

The known minimum-norm Lasso solutions are used to initialize 3, 4 and 5-layer standard neural networks (15) with $m = 100$ neurons and trained with Adam, shown in Figure 4. The second coordinate is 0 in all samples. These toy datasets are designed in an attempt to give the simplest examples where neural networks specifically produce individual, theoretically predicted features out of the wide range of other possible functions (including non-symmetric ones) that the neural networks could have generated. Indeed, the networks appear as 2-D versions of the optimal 1-D Lasso solutions found in [7], verifying our intuition. As shown in Figure 4, the networks exhibit breakplanes at data points and their reflections which are not among the training data. The neural networks exhibit features and reflection lines matching those shown in the right and left plots of Figure 2 for 3 and 4 layers, respectively, demonstrating Theorem IV.6 and Theorem IV.9. The networks plotted in Figure 4 show that features gain more complex reflections with depth, as predicted by Theorem IV.9. For space, details are in Section D-A. While useful for intuition, this simulation is limited in consisting of a small hand-picked training dataset. The next section presents experiments on a larger and higher-dimensional dataset.

B. LLM embeddings of IMDB data (large d, N): Standard networks with $\sigma(x)=|x|$

Next, we train neural networks using Adam to perform binary ($y_n = \pm 1$) classification of text and observe multi-level symmetries. The neural networks have standard architectures (15), where the last hidden layer and the final outer layer each have $m > 1$ neurons. The intermediate layers have one neuron each. While this architecture is limited, it enables a straightforward and meaningful extraction of a first-layer weight vector in order to visualize symmetric features hidden in the network as described below.

Models from OpenAI GPT-4 and Bert are trained to embed text as vectors, which standard networks are then trained on as input to classify the corresponding text. These text embeddings generated by the LLMs simply serve as high-dimensional input features to subsequent classifiers. Our goal is to study the structure of these subsequent networks. The standard networks are trained using the non-convex regression problem (4). Positive network outputs are interpreted as $y_n=1$ and negative outputs as $y_n=-1$. Figure 5 plots predictions of neural nets that are trained to perform sentiment analysis, rating IMDB reviews as having positive or negative sentiments [43]. Figure 5 plots the points $(\mathbf{W}^{(1)}\mathbf{x}_n, \hat{y}_n)$ in red and blue, where $\hat{y}_n = f_L(\mathbf{x}_n; \theta)$ is the network's prediction on the training sample $\mathbf{x}_n$ and the color corresponds to the training labels as red for $y_n=1$ and blue for $y_n=-1$. The network prediction on all points $\mathbf{x}$ is plotted in gray. For any constant $c \in \mathbb{R}$, a Lasso feature (58) has constant value along $\{\mathbf{x} : \mathbf{W}^{(1)}\mathbf{x}=c\}$, so projecting the input along $\mathbf{W}^{(1)}$ can be viewed as a cross-section of the feature. The network is trained on high dimensional embedding vectors, and the projection along $\mathbf{W}^{(1)}$ is only for visualization in Figure 5.

As discussed in Section A-D, an absolute value activation does not imply that a neural network is a symmetric function, but it does imply that the network is a linear combination of symmetric features (Definition IV.7). The overall symmetric shapes of Figure 5 are consistent with the network choosing a single symmetric feature (or a sparse set), which supports the Lasso equivalence in Theorem IV.2. More importantly, however, Figure 5 demonstrates *multi-level*, fractal-like symmetries with breakpoints that are reflections. This is particularly apparent in the blue data near 0 on the bottom row: the 4-layer network (left) generates 2 smaller "hills" that appear self-similar to the small hill near 0 of the 3-layer network. These properties are characteristic of the features predicted by Theorem IV.9 and illustrated in

Figure 2. Reflections are a consequence of function composition rather than symmetric activations, as Theorem IV.2 shows that even leaky ReLU can induce normalized reflection features. In particular, the 3-layer networks in Figure 5 resemble the 3-layer network in the left plot of Figure 2, while the 4-layer network Figure 5 resemble the 4-layer network shown in the middle and right plots, respectively, of Figure 2. This supports Theorem IV.9.

As discussed in Section IV-A1, as leaky ReLU approaches ReLU, so do their corresponding features. Accordingly, we expect it to be difficult to observe reflection features in the deep narrow networks trained on IMDB data in Section D-B if standard leaky ReLU is used. Hence absolute value activations are used instead to illustrate network affinities for symmetries. To help address the challenge of investigating symmetry features with leaky ReLU, the next section simulates a data set with known multi-level symmetries that show standard leaky ReLU networks can have higher representation power with depth for more complex symmetries. Section D-B contains training details and additional results due to space. We note this experiment is intended to explore the presence of features described by the theory in real data. The exact theoretical results of this paper are not intended as analysis nor practical training approaches for the GPT-4 or BERT models themselves.

Architecture: Note that the 2-layer standard networks used in these experiments have the same form as those commonly used in practice.” $f(\mathbf{x}; \theta) = \sigma(\mathbf{x}\mathbf{W})\alpha$ where $\mathbf{x} \in \mathbb{R}^{1 \times d}$, $\mathbf{W} \in \mathbb{R}^{d \times m}$, $\alpha \in \mathbb{R}^m$. (Biases are omitted here for simplicity). Deeper architectures used in these experiments have some differences from general, more typical networks. For example, 3-layer networks are of the form $f(\mathbf{x}; \theta) = \sigma(\sigma(\mathbf{x}\mathbf{W}^{(1)})\mathbf{W}^{(2)})\alpha$ where $\mathbf{x} \in \mathbb{R}^{1 \times d}$, $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times 1}$, $\mathbf{W}^{(2)} \in \mathbb{R}^{1 \times m}$, $\alpha \in \mathbb{R}^m$. The only structural difference between this network and a more general, fully-connected 3-layer network is that the second dimension of $\mathbf{W}^{(1)}$ (and thus the first dimension of $\mathbf{W}^{(2)}$) is fixed to 1 rather than being arbitrary. These deeper standard networks can be viewed as sub-networks within typical wider networks, and the symmetric patterns that they learn may suggest implicit symmetries in wider networks as well. To elucidate symmetric behaviors in wider networks and demonstrate experiments with non-symmetric activations, we experiment with Cantor sets, discussed next.

C. Cantor set (small d , large N): Wider networks with leaky ReLU

Theorem V.1 states that layer- L Cantor features have structural similarities to $(L - 2)^{\text{th}}$ -level Cantor sets. Since the feature libraries of deeper networks contain those of shallower networks, this suggests that for a Cantor set of some fixed level, network performance improves with depth, and that deeper networks with absolute value activation can better learn higher-level Cantor sets. We train 2, 3, 4 and 5-layer neural networks with leaky ReLU (standard Pytorch default slopes of $b = -0.01$ and $a = 1$) on 1-D Cantor sets in Figure 6. Training is repeated for 20 random initializations and the results across different trials are collected in Figure 7. Figure 6 plots the network predictions for one example trial at each Cantor level and network depth.

For convenience, the 1-D Cantor set’s full initial interval $C_0 = [0, 1]$ was scaled to $[-1, 1]$. The training dataset for the level $n \in [4]$ Cantor set is simulated as follows: $N = 1000$ points from $[-1, 1]$ are sampled uniformly at random as training data x_i , and each x_i is assigned label $y_i = 1$ if x_i is in the level- n Cantor set, and $y_i = -1$ otherwise. The testing data is formed in the same manner. The neural networks were trained as a regression task with MSE loss to fit the 1-D data x_i to $y_i \in \{-1, 1\}$. The neural networks were wider, rectangular versions of

the standard networks defined in (15) with the modification that $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times m}$, $\mathbf{b}^{(l)} \in \mathbb{R}^{1 \times m}$ for all $l \in [L - 1]$, and $\mathbf{W}^{(l)} \in \mathbb{R}^{m \times m}$ for all $l = 2, \dots, L - 1$. As before, $\xi \in \mathbb{R}$, $\alpha \in \mathbb{R}^m$. All of the neural networks of different depths are designed to have a similar total number of parameters by varying the width m of each network, e.g. making shallow networks wider. Section D-C describes the width calculation.

In Figure 6 and Figure 7, typically 3-layer networks learn Cantor sets more accurately than 2-layer networks given the same number of epochs and learning rate, and for larger-level Cantor sets, deeper networks show higher performance, even with approximately the same total number of parameters. This is consistent with deeper features learning more complex multi-level symmetries. On smaller-level Cantor sets however, the performance of deeper networks can plateau or worsen compared to shallow networks. This indicates that deep networks may be fitting deep network features not present in shallower networks, which are too complex for simpler data, causing overfitting. Shallower networks may better fit data with simpler structures.

Figure 8 illustrates classification experiments on 2-D Cantor sets with binary labels corresponding to set inclusion or exclusion in the Cantor set. A 2-D Cantor set is the Cartesian product of a 1-D Cantor set with itself. $N = 10^8$ random points in $[0, 1]^2$ were used as training data x_i with corresponding label $y_i \in \{0, 1\}$ indicating if x_i was in the Cantor set or not. The testing data is simulated in the same manner. The neural network was trained as a binary classification problem with binary cross-entropy loss. We use the same rectangular network architecture as used for the 1-D Cantor set experiment, with an approximately constant number of total parameters across networks of different depths. A Cantor set of level $n = 2$ was used and the experiment was repeated over 5 different initializations. The results for each initialization are shown in Figure 7 in the Supplementary Material due to space. Figure 7 plots the training loss, training data, and contour plots of the trained network’s output, which represents the probability of being in the Cantor set, across a grid of 100×100 points in $[0, 1]^2$. In Figure 7, deeper networks tend to more quickly learn the data and given the same number of epochs and learning rate, more cleanly fit the Cantor set, with improved training and testing loss. Section D-C contains further experiments and details.

VII. DISCUSSION

Our results show that deep narrow networks with non-monotone activations learn sparse combinations of symmetry-based features whose complexity increase with depth. This suggests that depth enables such networks to more efficiently represent classes of functions with higher intricate geometric or reflectional structures, possibly improving generalization. This is seen in the Lasso feature-like properties and network fitting of Cantor sets (Section V, Section VI-C). While the extent of symmetry in a Cantor set is defined by the Cantor set level, quantifying the level of symmetry in general data is an area to explore in future work, as well as providing formal statistical bounds on test performance.

Intuition from the Lasso model can guide selection of network depth based on the data and interpretation of the solutions obtained in practice. Our LLM experiments (Section VI-B) illustrate that network predictions may be interpreted with Lasso reflection features. Model designers may further tune the “leakiness” of ReLU: closer to absolute value for data with symmetric reflections, or closer to ReLU for flatter normalized reflections. We can interpret ReLU networks as learning features that are an extreme limit of leaky ReLU features (Figure 3).

The Lasso formulation also reveals a depth–width tradeoff: deeper networks expand the feature dictionary, while limited width restricts the number of active features. The Lasso equivalence shows that regularization acts at the level of features rather than individual weights, inducing sparsity in the set of active parallel units. This implies that in practice large regularization coefficients may encourage networks to select fewer, but not necessarily simpler, features. Since deeper networks include the feature dictionaries of shallower ones, they can achieve the same training error but may introduce more complex features, potentially leading to overfitting or artifacts. This may be related to model hallucinations, a direction for future investigation.

The convex reformulation may be computationally challenging in high dimensions. For example in Section VI-B, the dimension and size of the IMDB dataset may be too large for direct training of Lasso models to be practical. However, understanding the geometric features encoded in the Lasso model provides theoretical insight into the structure of the network’s predictions, what the network is capable of learning in principle, and the role of depth in shaping representation. Our results do not guarantee that SGD reaches optimal Lasso solutions. The goal in this paper is not to prove convergence to this optimum or the behavior of any specific training algorithm, but rather to characterize the structure of ideal solutions and loss landscape. For convex formulations of 2-layer ReLU networks, [36] suggests that local optima that may be found by gradient descent correspond to subsampled Lasso problems. Extending such work to deeper networks, applying our specific Lasso results to gradient-based training analysis and developing training algorithms that more reliably reach optimal solutions are left as open problems.

VIII. LIMITATIONS

Limitations of this work and areas addressing them include the deep narrow network architecture (Section A-A), ℓ_1 regularization in training (Section I-C), and focus on leaky ReLU activation, especially absolute value activation (Section A-D, Section A-E). Other limitations include computational complexity (Section IV-A) of the convex Lasso problem and experimental limitations in demonstrating symmetries in neural networks (Section D).

IX. CONCLUSION AND FUTURE DIRECTIONS

We prove an equivalence between deep narrow networks and Lasso problems with geometric features. We believe that multi-level symmetries hold for standard ReLU networks, which remains a question left for future work. An interesting direction is applying our work to data consisting of symmetrical and hierarchical patterns that appear in nature, for example Turing patterns that appear in diverse phenomena from embryogenesis in developmental biology [44] to skin patterning of animals and shell species [45]. Insight into the connection between depth and learning power can also help design compact models that are compute and memory-efficient, which are relevant in edge and green computing.

ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation (NSF) under Grant DMS-2134248; in part by the NSF CAREER Award under Grant CCF-2236829; in part by the U.S. Army Research Office Early Career Award under Grant W911NF-21-1-0242; in part by the Office of Naval Research under Grant N00014-24-1-2164, in part by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1656518.

APPENDIX A

MODELING: WHY STUDY THIS ARCHITECTURE AND ACTIVATION?

A. Architecture: deep narrow networks

Although the simplicity of deep narrow networks is a limitation of this work, we believe fully understanding general networks first requires having a thorough understanding of simpler architectures. Even shallow networks can be challenging to study and train due to neural networks being non-convex. Being equivalent to standard networks (15) for 2 layers, deep narrow networks are a natural extension to deeper networks with more expressive capacity. The scalar weights are a simplification that enables tractable analysis and isolates the role of depth in function composition, a key characteristic of neural networks. We show that increasing the number of layers in a deep narrow network even with the constant width $m_1 = \dots = m_{L-1} = m$ enables the network to learn richer features. While each layer of a deep narrow network has the same number m_L of neurons, this m_L can be arbitrarily large so that the network's expressive power is not too limited. Moreover, a deep narrow network of any depth can be converted into a standard architecture [7]. [7] uses a deep narrow network to develop equivalent Lasso models for 1-D data. Section II-B discusses other works that used a simpler version of the deep narrow network.

B. Do results generalize to deep standard networks?

[46] shows that a parallel structure is necessary for a convexification approach similar to the one in this paper to produce a network that is equivalent, and for standard networks, even a linear activation function such as $\sigma(x) = x$ results in a duality gap, meaning the transformed network is suboptimal. With a standard network, it is still possible to create a convex program that is a lower bound on the training problem, but the convex equivalence proof fails at the final step of reconstructing a network that establishes equality between the training and the convex problems. Therefore for worst-case data, a parallel network is necessary. However, a standard network can suffice in certain conditions. When β is sufficiently large, or if the data is sufficiently structured, the equivalent Lasso problem can select a single feature, which corresponds to a parallel network with one unit ($m_L = 1$). Such a network is both parallel and standard, and therefore the deep narrow results apply to standard networks. This is seen in the experiment shown in Figure 12 for ReLU and Figure 4. We note that a deep narrow network is a special case of a *parallel network* described in Section E, which [7] shows can be converted to a standard architecture (15).

C. Activation: Leaky ReLU

The leaky ReLU is used in practice to avoid “dead neurons” [47]. We show that networks with non-monotone leaky ReLU ($a, b > 0$) exhibit features that represent normalized reflections. While in practice networks are more likely to use monotone ($b \leq 0$) leaky ReLU than non-monotone, a non-monotonic activation generates reflection features in the network because it flips the direction of the neural network at zero crossings, creating breakpoints corresponding to reflections. Section A-E and Section IV-A1 discuss application to ReLU. We pay particular attention to absolute value activation because its corresponding features are the most symmetric and tractable to analyze, discussed next.

D. Why study absolute value activation?

The absolute value activation is a symmetric function and this allows for tractable, simpler formulations of neural network features. For example, [7] shows that for 1-D data, a 2-layer network is equivalent to a Lasso model with a $N \times 2N$ dictionary matrix for ReLU, while this dictionary reduces to a $N \times N$ symmetric and invertible dictionary matrix for absolute value activation, where N is the number of training samples. In higher dimensions, the absolute value activation is also simpler to analyze for gaining insight into their geometric structures as it suffices to consider only the magnitude of inner weight parameters rather than doing an exponential case analysis on their signs.

Moreover, absolute value networks are useful in their own right. Neural networks with absolute value activation have been shown to perform comparably to ReLU and have advantages such as no exploding or vanishing gradients, making them suitable for deep networks [48] and outperforming other activations in computer vision [49]. Our results suggest that absolute value activations are especially primed to capture reflections and symmetries in the data, leading to their representation power.

In addition, absolute value activation in 2-layer networks is equivalent to ReLU with skip connection [7]. For a 2-layer neural network $f_2(\mathbf{x}; \theta)$, a *skip connection* adds a linear term ωx as $f'_2(x; \theta) = f_2(x; \theta) + \omega x$ if $x \in \mathbb{R}$ (or for row vectors $\mathbf{x}$, $f'_L(x; \theta) = f_L(\mathbf{x}; \theta) + \mathbf{x}\omega$) where $\omega \in \mathbb{R}^d$ is a trainable parameter of θ . [7] shows that

Lemma A.1 ([7]). *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 their corresponding solutions.*

E. Do results generalize to ReLU?

Extensions to deep, general ReLU networks are left to future work, but the reflection features observed with absolute value activation lay the groundwork for searching for similar geometric features in ReLU networks. Lemma A.1 in [7] shows that 2-layer networks with absolute value activation are equivalent to those with ReLU when the network uses a skip connection. In both cases, ReLU and absolute value networks are piecewise linear functions.

[7] also shows that unlike absolute value, the features and expressive power of networks with ReLU trained on 1-D data with a deep narrow network architecture stagnate after 3 layers and never produce reflections. However, when the width is doubled compared to a deep narrow network, certain networks with ReLU demonstrate reflection features [7]. Intuitively, this is because 2 ReLU activations are needed to capture the decreasing and increasing segments of the absolute value activation to have comparable expressivity. Therefore, increasing the width of ReLU networks with 1-D input data introduces reflection features. While this result was only for 1-D data in [7], a similar phenomenon is shown for 2-D data in this work, as seen in Figure 12 which illustrates a reflection feature occurring in a network trained on 2-D data. This suggests that wider ReLU architectures with higher-dimensional data may also produce reflection features. As discussed in Section IV-A1, 3-layer ReLU features in deep narrow networks can be viewed as limits of features for leaky ReLU.

While having symmetric activations, deep narrow networks can also express non-symmetric functions. However, they are primed to express multi-level symmetric functions due to the ℓ_1 feature selection revealed by their Lasso

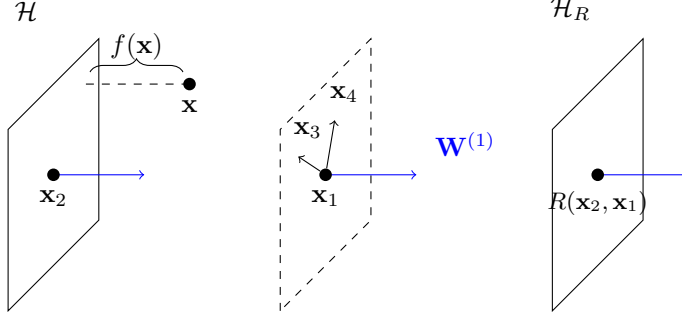


Fig. 9: Example of reflection planes for 3-layer neural networks with data in $\mathbb{R}^3$.

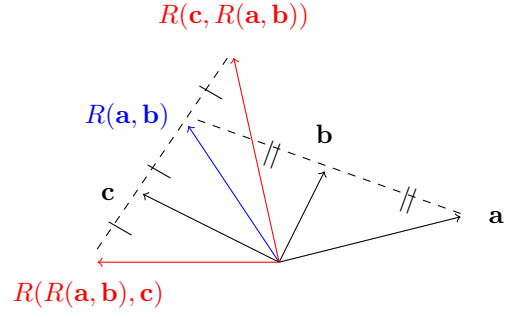


Fig. 10: Reflections in $\mathbb{R}^2$ of different orders. Order 0: $\mathbf{a}, \mathbf{b}, \mathbf{c}$. Order 1: $R(\mathbf{a}, \mathbf{b})$. Order 2: $R(R(\mathbf{a}, \mathbf{b}), \mathbf{c}), R(\mathbf{c}, R(\mathbf{a}, \mathbf{b}))$.

equivalence. While the exact features will depend on the activation function and may incorporate more general reflections, our analysis in the proofs of Lasso equivalence and the presence of normalized reflection features in networks with general, non-monotonic activations suggest that rather than being a simple consequence of the symmetry of the absolute value activation, the *multi-level, fractal-like structures arise in deep narrow networks because of the composition of activation functions, which is a fundamental characteristic of neural networks*. Therefore similar multi-level features may exist in more general networks as well.

APPENDIX B GEOMETRIC ALGEBRA

Recently, [8] showed that geometric algebra, also known as Clifford algebra, is an elucidating framework for describing neural network features. Geometric algebra is an extension of vector algebra that centers around geometric operations such as translations and reflections [34], [50], [51], and generalizes vector operations such as cross products, inner products, and outer products from a geometric perspective [34]. Geometric algebra provides a unified framework for expressing laws of physics [52], [53] and has applications in computer graphics [54].

Consider the Euclidean vector space $\mathbb{R}^d$ with a scalar product (also called a inner or dot product for vector inputs). A *geometric algebra over $\mathbb{R}^d$* is a vector space $\mathbb{G}^d$ (over the scalar field $\mathbb{R}$) of elements called *multivectors*, equipped with an associative binary operation called a *geometric product* that satisfies the following: (i) $\mathbb{G}^d$ contains $\mathbb{R}^d$ and is closed under the geometric product, (ii) geometric product is bilinear and distributes over addition, (iii) scalar multiplication is associative and commutative with the geometric product, (iv) the geometric product of a vector with itself is equal to the scalar product of the vector with itself [50].

The geometric product of $A, B \in \mathbb{G}^d$ is denoted as AB . It can be shown that the geometric product of the basis vectors e_i, e_j of $\mathbb{R}^d \subset \mathbb{G}^d$ satisfy $e_i e_i = 1$ and $e_i e_j = -e_j e_i$ [50]. Every multivector is a linear combination of *basis blades*, which are multivectors that are geometric products of the form $E = e_{\mathcal{G}[1]} \cdots e_{\mathcal{G}[|\mathcal{G}|]} = \prod_{i \in \mathcal{G}} e_i$, where $\mathcal{G} \subset [d]$ is an ordered set with no repeated elements and $\mathcal{G}[i]$ is its i^{th} element. A basis blade E has *grade* $g(E) = |\mathcal{G}|$. Scalars are defined to have grade 0. A dot product $A \cdot B$ and wedge product $A \wedge B$ are defined for multivectors

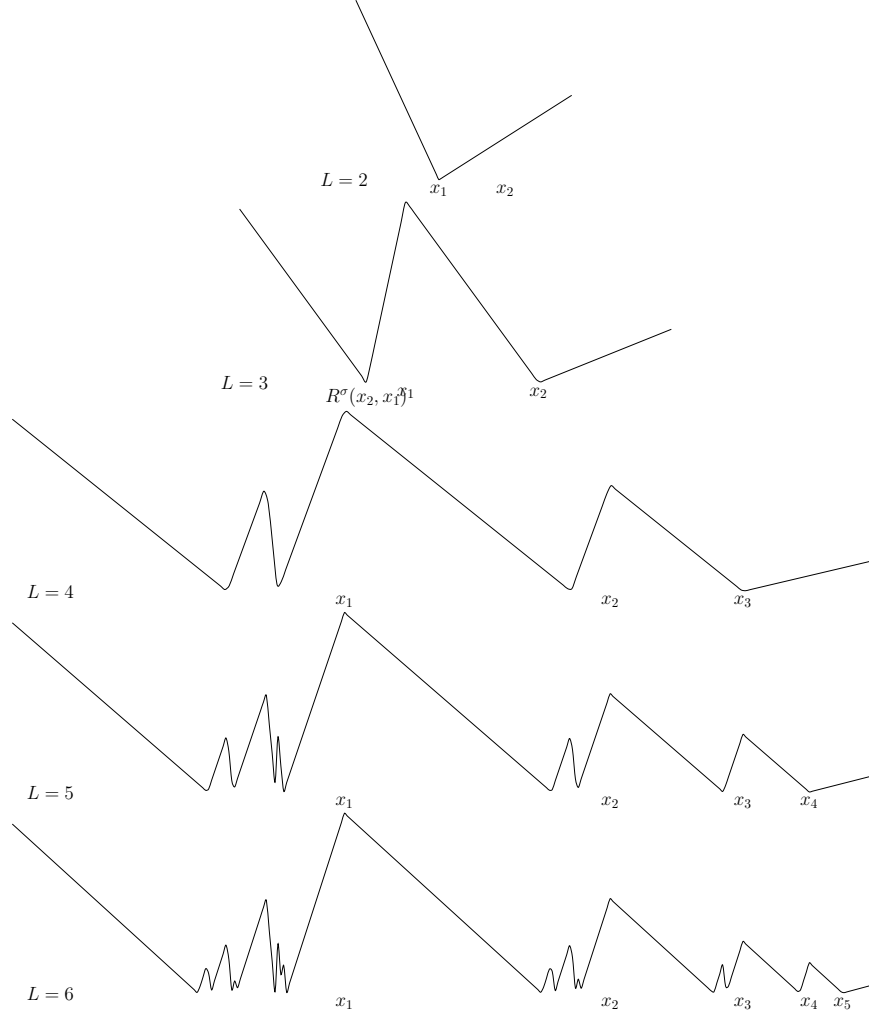


Fig. 11: Examples of features (Definition IV.7) for networks with leaky relu activation $\sigma(x) = a(x)_+ + b(-x)_+$ where $a = 0.5, b = 1.7$. The input data is 1-D so that the features can be more clearly compared visually. Each layer- L feature uses training data $x_1, \dots, x_{L-1}$ for $L = 2, \dots, 6$. In these features, $\mathbf{W}^{(l)} = 1$ and $\mathbf{b}^{(l)} = x_l$. The data is $x_1 = 0, x_2 = 4, x_3 = 6, x_4 = 7$. The successively halved spacing between the data contributes to a feature with a multi-level structure that is more clear, with structures that have self-similar qualities appearing with each layer. The 3-layer feature exhibits a breakpoint at a normalized reflection $R^\sigma(x_2, x_1)$ of x_2 about x_1 .

in a way that generalizes the vector inner product and cross product, respectively. For vectors $A, B \in \mathbb{R}^d \subset \mathbb{G}^d$, $AB = A \cdot B + A \wedge B$ where the dot product represents the projection of A onto B , and the wedge product represents the signed (oriented) area of the triangle spanned by A and B .

Since $e_i e_i = 1$ and $e_i e_j = -e_j e_i$, the grade of all multivectors in $\mathbb{G}^d$ is at most d . A *pseudoscalar* is a multivector of grade d . The *unit pseudoscalar* is $\mathbf{I} = e_1 \cdots e_d$. The *unit pseudoscalar inverse* is $\mathbf{I}^{-1} = e_d \cdots e_1$ which satisfies $\mathbf{I}^{-1} \mathbf{I} = 1 = \mathbf{I} \mathbf{I}^{-1}$. The *Hodge star operator* $\star : \mathbb{G}^d \rightarrow \mathbb{G}^d$ is defined as $\star(\mathbf{v}) = \star \mathbf{v} = \mathbf{v} \mathbf{I}^{-1}$. If $\mathbf{v}$ is a wedge product of vectors, $\star \mathbf{v}$ represents the orthogonal complement of their span.

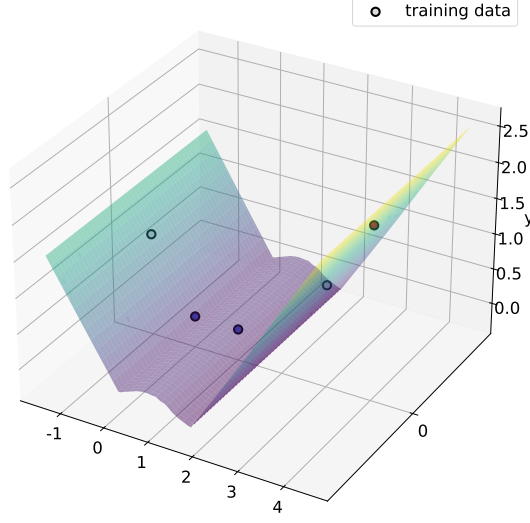


Fig. 12: 3-layer ReLU neural net. A breakplane occurs at a reflection of training data, which is not in the training set.

Wedge and inner product: A dot product of multivectors, unlike a dot product of vectors, can have multivector outputs. The *dot* or *inner product* of arbitrary basis blades $E_1 = \prod_{i \in \mathcal{G}^{(1)}} e_i$ and $E_2 = \prod_{i \in \mathcal{G}^{(2)}} e_i$ is $E_1 \cdot E_2 = E_1 E_2$ if $g(E_1 E_2) = |g(E_1) - g(E_2)|$, and 0 otherwise. In other words, if $G^{(1)} \subset G^{(2)}$, then $E_1 E_2$ is composed of basis blades in $\mathcal{G}^{(2)} - \mathcal{G}^{(1)}$, and $E_1 E_2 = 0$ otherwise. (Similarly if $G^{(2)} \subset G^{(1)}$). The *wedge* or *outer product* is defined by $E_1 \wedge E_2 = E_1 E_2$ if $E_1 E_2$ has grade $g(E_1 E_2) = g(E_1) + g(E_2)$, and 0 otherwise. So if $\mathcal{G}^{(i)} \cap \mathcal{G}^{(j)} = \emptyset$, then $E_1 E_2$ is composed of basis blades in $\mathcal{G}^{(1)} \cup \mathcal{G}^{(2)}$, and otherwise, $E_1 E_2 = 0$. The dot and wedge product extend to multivectors linearly. A *vector* is a special case of a multivector $\mathbf{v} \in \mathbb{R}^d \subset \mathbb{G}^d$ of grade 1. For more detailed background, see [50].

Note that linearly distributing terms and applying $e_i e_i = 1$ shows that $\wedge_{i=1}^{d-1} \mathbf{v}_i = \wedge_{i=1}^{d-1} \sum_{j=1}^d v_{ij} e_j$ consists of a sum of $d-1$ -vectors, so their product with $e_d \cdots e_1$, as implied by the Hodge star operator in (9) is a vector, which is consistent with (7).

Remark B.1. It holds that $\star(\mathbf{u} \wedge (\wedge_{i=1}^{d-1} \mathbf{v}_i)) = \mathbf{u}^T (\times_{i=1}^{d-1} \mathbf{v}_i) \in \mathbb{R}$ and $\|\wedge_{i=1}^{d-1} \mathbf{v}_i\|_2 = \|\times_{i=1}^{d-1} \mathbf{v}_i\|_2$. This is because

$$\begin{aligned} \wedge_{i=1}^{d-1} \mathbf{v}_i &= \sum_{k=1}^d \left(\left(\wedge_{i=1}^{d-1} \mathbf{v}_i \right)^T (e_1 \cdots e_{k-1} e_{k+1} \cdots e_d) \right) e_1 \cdots e_{k-1} e_{k+1} \cdots e_d \\ &= \sum_{k=1}^d \left(\times_{i=1}^{d-1} \mathbf{v}_i \right)_k e_1 \cdots e_{k-1} e_{k+1} \cdots e_d \end{aligned} \quad (16)$$

and $e_j \wedge e_k = 0$ if $j \neq k$, we have $\mathbf{u} \wedge (\wedge_{i=1}^{d-1} \mathbf{v}_i) = \sum_{j=1}^d u_j e_j \wedge (\wedge_{i=1}^{d-1} \mathbf{v}_i) = \sum_{j=1}^d u_j e_j \wedge \left(\left(\times_{i=1}^{d-1} \mathbf{v}_i \right)_k \right) e_1 \cdots e_{j-1} e_{j+1} \cdots e_d$. So $\star(\mathbf{u} \wedge (\wedge_{i=1}^{d-1} \mathbf{v}_i)) = \mathbf{u} \wedge (\wedge_{i=1}^{d-1} \mathbf{v}_i) e_d \cdots e_1 = \mathbf{u}^T (\times_{i=1}^{d-1} \mathbf{v}_i) \in \mathbb{R}$. From (16) it also follows that $\|\wedge_{i=1}^{d-1} \mathbf{v}_i\|_2 = \|\times_{i=1}^{d-1} \mathbf{v}_i\|_2$.

Remark B.2. *It holds that*

$$|\star(\mathbf{v}_1 \wedge \cdots \wedge \mathbf{v}_d)| = \text{Vol}(\mathbf{v}_1, \dots, \mathbf{v}_d) \quad (17)$$

and

$$\text{Dist}(\mathbf{v}_d, \text{Span}(\mathbf{v}_1, \dots, \mathbf{v}_{d-1})) = \frac{\star(\mathbf{v}_1 \wedge \cdots \wedge \mathbf{v}_{d-1})}{\|\mathbf{v}_1 \wedge \cdots \wedge \mathbf{v}_{d-1}\|_2}. \quad (18)$$

Here, $\mathbf{v}_1, \mathbf{v}_2$, let $\text{Dist}(\mathbf{v}_1, \mathbf{v}_2)$ denote the unsigned Euclidean distance $\|\mathbf{v}_1 - \mathbf{v}_2\|_2$ between $\mathbf{v}_1$ and $\mathbf{v}_2$. The distance between a vector $\mathbf{v}$ and a set S is $\min\{\text{Dist}(\mathbf{v}, \mathbf{x}) : \mathbf{x} \in S\}$. In other words, $\mathbf{v}_1 \wedge \cdots \wedge \mathbf{v}_{d-1}$ is a vector whose magnitude is the unsigned volume of the $d-1$ -dimensional parallelotope spanned by $\mathbf{v}_1, \dots, \mathbf{v}_{d-1}$, which represents the "base" of the d -dimensional parallelotope $\mathcal{P}(\mathbf{v}_1, \dots, \mathbf{v}_d)$ and (18) is the signed "height" of $\mathcal{P}(\mathbf{v}_1, \dots, \mathbf{v}_d)$, or the distance of $\mathbf{v}_d$ to the span of $\mathbf{v}_1, \dots, \mathbf{v}_{d-1}$. Note the Hodge star operator $\star$ converts pseudoscalars into a scalars.

APPENDIX C

LASSO SOLUTION SET AND PATH

The Lasso equivalence offers theoretical insight into a subset of optimal neural networks, namely those reconstructed from Lasso solutions, as the Lasso solution set is well-studied [2], [3]. Using [3], [7] shows the following.

Lemma C.1 ([7]). *The optimal model fit $\hat{\mathbf{y}} = \mathbf{A}\mathbf{z}^* + \xi^*\mathbf{1}$, the linear fit $\mathbf{A}\mathbf{z}^*$, and optimal bias ξ^* are the same for all optimal solutions $(\mathbf{z}^*, \xi^*)$ to the Lasso problem.*

Therefore there is a well-defined optimal fit $\hat{\mathbf{y}}$, and we define the equicorrelation set $\mathcal{E}_\beta$ as

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

The equicorrelation set contains the indices of the features maximally correlated with the residual $\hat{\mathbf{y}} - \mathbf{y}$, which determine solution set.

Proposition C.2 ([7]). *Then the set of global optima of the Lasso problem is*

$$\begin{aligned} \Phi^*(\beta) = \{(\mathbf{z}, \xi^*) : z_i \neq 0 \Rightarrow \text{sign}(z_i) = \text{sign}(\mathbf{A}_i^\top(\hat{\mathbf{y}} - \mathbf{y}))\}, \\ z_i = 0 \forall i \notin \mathcal{E}_\beta, \mathbf{A}\mathbf{z} = \hat{\mathbf{y}} - \xi^*\mathbf{1}\}. \end{aligned} \quad (19)$$

The solution set $\Phi^*(\beta)$ is a polyhedron with vertices that are solutions with the sparsest optimal $\mathbf{z}$ [55], [7].

It can be shown from Equation (19) that all neural nets reconstructed from the Lasso share the same model fit and have features corresponding to indices contained in most the equicorrelation set [7]. One optimal neural net corresponding to a Lasso solution can give all others by applying a reconstruction map to Φ^β [7].

The solution path is the Lasso solution set as the regularization parameter β changes, which is known to be continuous [3]. Since the reconstruction map is continuous, so is the corresponding path of reconstructed neural net solutions as the regularization sweeps, and the LARS algorithm [5] can efficiently compute the solution path [7].

Analysis of neural networks that are optimal or stationary in the training problem but do not correspond to reconstructed Lasso solutions are an area for future work.

APPENDIX D

NUMERICAL RESULTS

The standard neural net in (15) can be expanded as

$$f_L(\mathbf{x}; \theta) = \sigma \left(\cdots \left(\sigma \left(\sigma \left(\mathbf{x} \mathbf{W}^{(1)} + \mathbf{b}^{(1)} \right) \mathbf{W}^{(2)} + \mathbf{b}^{(2)} \right) \cdots \right) \mathbf{W}^{(L-1)} + \mathbf{b}^{(L-1)} \right) \boldsymbol{\alpha} + \xi.$$

A. *Toy data (small d, N): Deep narrow networks with $\sigma(x) = |x|$*

We start with training $m = 100$ -neuron standard neural networks on hand-picked toy datasets whose 1-D projection have known, tractable minimum-norm Lasso solutions that consist of individual Lasso features [7]. The networks are shown in Figure 4. The networks resemble the theoretical features drawn in Figure 2. A similar experiment is performed with a ReLU network, which also demonstrates breakplanes at reflections of training data, shown in Figure 12.

The known minimum-norm Lasso solutions are used to initialize 3, 4 and 5-layer standard neural networks (15) with $m = 100$ neurons and trained with Adam, shown in Figure 4. The second coordinate is 0 in all samples. These toy datasets are designed in an attempt to give the simplest examples where neural networks specifically produce individual, theoretically predicted features out of the wide range of other possible functions (including non-symmetric ones) that the neural networks could have generated. The networks appear as 2-D versions of the optimal 1-D Lasso solutions found in [7], and therefore this simple dataset gives an example where we can verify our intuition. As shown in Figure 4, the networks exhibit breakplanes at data points and their reflections which are not among the training data. The neural networks exhibit features and reflection lines matching those shown in the right and left plots of Figure 2 for 3 and 4 layers, respectively, demonstrating Theorem IV.6 and Theorem IV.9. The networks shown in Figure 4 show that features gain more complex reflections with depth, as predicted by Theorem IV.9. While useful for intuition, this simulation is limited in consisting of a small hand-picked training dataset. Section D-B presents experiments on a larger and higher-dimensional dataset.

Details: We first project the 2-D data to 1-D along the first coordinate and solve the Lasso problem for the 1-D data as $\beta \rightarrow 0$. The *minimum (ℓ_1) norm subject to interpolation* version of the Lasso problem is

$$\min_{\mathbf{z}, \xi} \|\mathbf{z}\|_1 \text{ s.t. } \mathbf{A}\mathbf{z} + \xi \mathbf{1} = \mathbf{y}. \quad (20)$$

Loosely speaking, as $\beta \rightarrow 0$, if $\mathbf{A}$ has full column rank, the Lasso problem approaches the minimum norm problem (20). For 1-D data, an optimal solution to (20) for certain simple sets of training data is known [7]. After solving the Lasso problem, the non-convex model (4) with the original 2-D data is trained with $\beta = 10^{-7} \approx 0$, where a minimum norm problem solution is used to pre-initialize a subset of the neurons. All other weights (we use $m=100$) are initialized randomly according to Pytorch defaults. Only one neuron per layer is pre-initialized, and for the neuron weight in the first layer, the second coordinate is left to random initialization. A learning rate of $5(10^{-3})$ and weight decay of 10^{-4} are used. The data is trained over 10^3 epochs.

In Figure 4, the training data is $(\mathbf{x}_1, y_1=2), (\mathbf{x}_2, y_2=0)$ for 3 layers, $(\mathbf{x}_1, y_1=2), (\mathbf{x}_2, y_2=0), (\mathbf{x}_3, y_3=-1)$ for 4 layers, and $(\mathbf{x}_1, y_1=1.75), (\mathbf{x}_2, y_2=0.25), (\mathbf{x}_3, y_3=0.75), (\mathbf{x}_4, y_4=-1.75)$ for 5 layers, where $\mathbf{x}_1=(2, 0), \mathbf{x}_2=(0, 0), \mathbf{x}_3=(-1, 0), \mathbf{x}_4=(-1.75, 0)$.

In Figure 12, the same experiment setup as used for Figure 4 is used for a 3-layer network, except $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times 2}$, $\mathbf{b}^{(2)} \in \mathbb{R}$, $\mathbf{W}^{(2)} \in \mathbb{R}^{2 \times m}$, $\mathbf{b}^{(L-1)} \in \mathbb{R}^{2 \times m}$ and we require all neuron weights in each layer have the same magnitude. The data is $\mathbf{x}_1 = (4, 0)$, $\mathbf{x}_2 = (3, 0)$, $\mathbf{x}_3 = (1, 0)$, $\mathbf{x}_4 = (0, 0)$, $\mathbf{x}_5 = (-1, 0)$, $y_1 = 2$, $y_2 = 1$, $y_3 = 0$, $y_4 = 0$, $y_5 = 1$. More general architectures are an area of exploration in future work.

B. Large Language Model data

We perform the same experiments as in Section VI-B, but with the GLUE data set, specifically GLUE-CoLA and GLUE-QQP [56]–[67]. GLUE (General Language Understanding Evaluation) is a benchmark for testing the performance of models learning language processing. GLUE-CoLA (Corpus of Linguistic Acceptability) contains text that is to be binary classified whether it is grammatically correct or not. GLUE-QQP (Quora Question Pairs) contains pairs of questions that are to be binary classified as having the same semantic meaning or not. Figure 1, Figure 2, Figure 3, Figure 4, Figure 5, Figure 6 in the Supplementary Material plot the results. Vector embeddings from both GPT-4 and Bert [68] LLMs are used. As in Figure 5, points with $y_n=1$ are plotted in red, and points with $y_n=-1$ are plotted in blue. The overall network is plotted in gray. The training data is plotted with 'x' and the validation and test data are plotted with '.' markers.

A standard architecture (15) was trained using Adam with a learning rate of $5(10^{-3})$, weight decay of 10^{-4} , and $\beta=10^{-7}$. There were $N=10^4$ data points, the network had $m=10$ neurons, and the network was trained over 100 epochs. For OpenAI's GLUE-CoLA input embeddings, 175 epochs and $m = 10$ neurons were used. For Bert GLUE-CoLA, 125 epochs and $m = 5$ were used. For OpenAI and Bert GLUE-QQP, 150 epochs and $m = 10$ were used. For OpenAI and Bert IMDB, 100 epochs and $m = 10$ were used. These parameters were chosen based on validation set results.

A linear support vector machine (SVM) classifier was also trained on the data and its percentage errors for each data set and embedding model were as follows: for IMDB: Bert - 15.560, OpenAI - 7.140; for GLUE-CoLA: Bert - 23.932, OpenAI - 29.810; for GLUE-QQP: Bert - 26.680, OpenAI- 23.460. While the linear SVM performed better than some of the instances of the neural networks, posing possible limitations on network usefulness, the best performing neural networks (out of 5 instances of initial points) performed better than the linear SVM. The neural networks had the highest improvement over the SVM for the GLUE-CoLA and GLUE-QQP data. Moreover, the performance of the neural networks performance is comparable to results in [37]. The networks are trained on the non-convex training problem. What distinguishes this experiment from [37] is demonstrating that networks learn novel features consistent with Lasso models.

Networks trained on both OpenAI and Bert embeddings exhibit similarities to Lasso features Figure 4. However, the networks trained on OpenAI embeddings (Figure 1) have higher accuracy and closer matches to Lasso features than those trained using Bert. The higher-performing networks trained using GPT-4, such as Figure 1, have striking similarities to Lasso features Figure 4. This is consistent with the network's equivalence to the Lasso model, as a network that is optimal in the Lasso problem is globally optimal. The figures show that when networks get deeper, they change by increasing their breakplanes, with more breakpoints appearing in the plots. Code from the Github repository for [37] was used to generate the embeddings.

C. Cantor set

In Figure 8, the networks are trained on more epochs. While the difference in performance between the shallow and deeper networks decreases when they are trained longer, the deeper networks still have better performance, and we see that the deeper networks achieve higher performance faster than the shallower networks, suggesting symmetric structures are implicitly favored by deeper networks over shallower networks. Figure 9 shows a similar experiment but with a different learning rate. In Figure 9, deeper networks learn the symmetries faster than shallow networks even with a similar number of total parameters. The 1-D experiment was designed to visualize the geometric shape of the neural networks when regressing to ± 1 labels corresponding to Cantor sets, while the 2-D experiment was designed to illustrate binary classification on Cantor sets.

The equalization of total number of neurons across networks is done as follows. The number of real-valued, trainable parameters in an L -layer network as a function of the width m is $p(m) = (L-2)m^2 + (d+L)m + 1$, where the input data is dimension d . This follows from $\mathbf{W}^{(1)}$ contributing dm parameters, $\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(L-1)}$ contributing $(L-1)m$ parameters, $\mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L-1)}$ contributing $(L-2)m^2$ parameters, α contributing m parameters, and ξ contributing 1 parameter. We set a target number of parameters $P = 1000$ and solve for the largest integer width m such that $p(m) \leq P$.

We note that the Cantor dataset is piecewise nonlinear with multiple linear pieces that would be difficult for a single linear model to fit. In contrast, a neural network can be viewed as a piecewise linear model over its activation regions [13]. Although the Cantor dataset is a mathematical rather than real-world object, it is an example of a fractal, which appears in many natural phenomena. Moreover, this simulation provides a concrete example that suggests that deeper networks have higher representation power for geometric, multi-level symmetries. The experiment also indicates that the theoretical results can extend beyond deep narrow networks with absolute value activation to standard wider networks with leaky ReLU. Extending the experimental work to training on other symmetrical patterns in nature is an area of future research. Note that while in the IMDB experiments in Section D-B, a symmetric (absolute value) activation is used to train on real (IMDB) data, in these Cantor experiments, a non-symmetric (leaky ReLU) activation is used to train on symmetric (Cantor) simulated data.

While our formulation of the symmetric features holds for $\beta > 0$, the networks are trained without regularization ($\beta = 0$) to show the relationship between depth and symmetry even with minimal regularization. For the 1-D Cantor set experiment ($d = 1$), Adam with a learning rate of 0.005 and batch size of 32 were used. The number of epochs is shown in Figure 6 and was chosen so that the training loss for at least the 2-layer networks were observed to plateau by the end of training. The number of epochs for the 2-D experiment are shown in Figure 7, Figure 8, Figure 9 in the Supplementary Material. For the 2-D Cantor set experiment ($d = 2$), Adam was used with a learning rate of 0.005 and the batch size was 128.

APPENDIX E

NEURAL NET ARCHITECTURE

Note: The notation and some techniques for convexification are similar to [7] and [8]. For finding the Lasso features, we can assume that the data matrix is full column rank [8]. All of the Lasso equivalences hold provided

that $m^* \geq \|\mathbf{z}\|_0$.

Let $L \geq 2$, $m_0 = d$, $m_{L-1} = 1$ and $m_l \in \mathbb{N}$ for $l \in [L] - \{L-1\}$. A *neural network* of depth L is a parameterized function $f_L(\cdot; \theta) : \mathbb{R}^{1 \times d} \rightarrow \mathbb{R}$ defined recursively as

$$\begin{aligned} f_L(\mathbf{x}; \theta) &= \sum_{i=1}^{m_L} \mathbf{X}^{(i,L)} \alpha_i + \xi \\ \mathbf{X}^{(i,l+1)} &= \sigma \left(\mathbf{X}^{(i,l)} \mathbf{W}^{(i,l)} + \mathbf{b}^{(i,l)} \right) \\ \mathbf{X}^{(i,1)} &= \mathbf{x}. \end{aligned} \quad (21)$$

We can view $\mathbf{X}^{(i,l)}$ as an input to the i^{th} unit in layer l . The trainable parameters, which the set θ contains, are the outermost weights $\alpha_i \in \mathbb{R}$, an external bias $\xi \in \mathbb{R}$; and the inner weights $\mathbf{W}^{(i,l)} \in \mathbb{R}^{m'_{l-1} \times m'_l}$ and inner biases $\mathbf{b}^{(i,l)} \in \mathbb{R}^{1 \times m'_l}$ for each layer l . There are $m_l = m_L m'_l$ neurons in each layer. The *activation* function is $\sigma : \mathbb{R} \rightarrow \mathbb{R}$. Note $m'_0 = d$, which is the input data dimension, and $m'_{L-1} = 1$, corresponding to the output being a scalar.

The neural network defined recursively with (21) can be expanded as

$$f_L(\mathbf{x}; \theta) = \sum_{i=1}^{m_L} \sigma \left(\dots \left(\sigma \left(\mathbf{x} \mathbf{W}^{(i,1)} + \mathbf{b}^{(i,1)} \right) \mathbf{W}^{(i,2)} + \mathbf{b}^{(i,2)} \right) \dots \right) \mathbf{W}^{(i,L-1)} + \mathbf{b}^{(i,L-1)} \alpha_i + \xi. \quad (22)$$

Note that for $i \in [m_L], l \in [L]$, we have $\mathbf{X}^{(i,l)} \in \mathbb{R}^{1 \times m_{l-1}}$. The set of regularized parameters is $\theta_w = \{\mathbf{W}^{(i,l)}, \alpha_i : i \in [m_L], l \in [L-1]\} \subset \theta$. All functions extend to vector and matrix-valued inputs elementwise.

The deep narrow network is a network (21) with absolute value activation $\sigma(x) = |x|$, arbitrary depth L , and a width $m_1 = \dots = m_L = m$ that is arbitrary but constant across layers. For the first layer, $\mathbf{X}^{(i,2)} = |\mathbf{x} \mathbf{W}^{(i,1)} + b^{(i,1)}|$ where $\mathbf{W}^{(i,1)} \in \mathbb{R}^d$ and in all other layers, $\mathbf{X}^{(i,l+1)} = |\mathbf{x} w^{(i,l)} + b^{(i,l)}|$ in (21), where $w^{(i,l)}, b^{(i,l)} \in \mathbb{R}$.

A deep narrow network network can be expanded as

$$f_L(\mathbf{x}; \theta) = \sum_{i=1}^m \left| \dots \left| \mathbf{x} \mathbf{W}^{(i,1)} + b^{(i,1)} \right| w^{(i,2)} + b^{(i,2)} \right| \dots \left| w^{(i,L-1)} + b^{(i,L-1)} \right| \alpha_i + \xi. \quad (23)$$

The training problem (4) can be stated generally as

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

where $\mathcal{L}_{\mathbf{y}} : \mathbb{R}^N \rightarrow \mathbb{R}$ is called the *loss* function that is convex and parameterized by $\mathbf{y}$, and $r(\theta)$ is a regularization term that penalizes large weight magnitudes. In (4), the loss function is $\mathcal{L}_{\mathbf{y}}(\cdot) = \frac{1}{2} \|\cdot - \mathbf{y}\|_2^2$. We can omit the external bias ξ from the network $f_L(\mathbf{X}; \theta)$ by absorbing it into the loss $\mathcal{L}_{\mathbf{y}}$. We assume $r(\theta_w)$ is of the form

$$r(\theta_w) = \sum_{i=1}^{m_L} \left(\left(r^{(i,L)}(\alpha_i) \right)^L + \sum_{l=1}^{L-1} \left(r^{(i,l)}(\mathbf{W}^{(i,l)}) \right)^L \right) \quad (24)$$

where $r^{(i,l)}$ is a nonnegative function that is positively homogeneous, i.e., for any positive scalar a , $r^{(i,l)}(a\mathbf{W}) = ar^{(i,l)}(\mathbf{W})$. In (4), the regularization is $r^{(i,l)}(x) = \|x\|_1$.

Lemma E.1. *The training problem is equivalent to the rescaled problem*

$$\min_{\theta \in \Theta: r^{(i,l)}(\mathbf{W}^{(i,l)})=1} \mathcal{L}_{\mathbf{y}}(f_L(\mathbf{X}; \theta)) + \beta \sum_{i=1}^{m_L} r^{(i,L)}(\alpha_i). \quad (25)$$

Proof. By the AM-GM inequality on (24), a lower bound on the training problem is

$$\min_{\theta \in \Theta} \mathcal{L}_{\mathbf{y}}(f_L(\mathbf{X}; \theta)) + \beta \sum_{i=1}^{m_L} r^{(i,L)}(\alpha_i) \prod_{l=1}^{L-1} r_{(i,l)}(\mathbf{W}^{(i,l)}). \quad (26)$$

Consider the minimization problem

$$\min_{\theta \in \Theta: r^{(i,l)}(\mathbf{W}^{(i,l)})=1} \mathcal{L}_{\mathbf{y}}(f_L(\mathbf{X}; \theta)) + \beta \sum_{i=1}^{m_L} r^{(i,L)}(\alpha_i) \prod_{l=1}^{L-1} r_{(i,l)}(\mathbf{W}^{(i,l)}). \quad (27)$$

Problem (27) is an upper bound on (26). Given optimal $\{\mathbf{W}^{(i,l)}, \alpha_i\}$ in (26), the rescaled parameters $\mathbf{W}^{(i,l)'} = \mathbf{W}^{(i,l)} / r^{(i,l)}(\mathbf{W}^{(i,l)})$ and $\alpha'_i = \alpha_i \prod_{l=1}^{L-1} r_{(i,l)}(\mathbf{W}^{(i,l)})$ (and rescaled bias parameters) achieve the same objective in (27). Hence (27) and (26) are equivalent. Given an optimal parameter $q \in \{\mathbf{W}^{(i,l)}, \alpha_i\}$ in (27), the rescaled parameters $q' = |\alpha_i|^{\frac{1}{L}} q$ (and rescaled bias parameters) achieve the same objective in the training problem, which is therefore equivalent to (27). Simplifying (27) gives (25). $\square$

Let $\mathbf{X}^{(L)} = \mathbf{X}^{(1,L)}$. Assume $r^{(i,l)}(x) = \|x\|_1$.

Lemma E.2. *A lower bound on the rescaled training problem is*

$$\max_{\lambda \in \mathbb{R}^N} -\mathcal{L}_{\mathbf{y}}^*(\lambda) \quad \text{s.t.} \quad \max_{\theta \in \Theta: \|\mathbf{W}^{(1,l)}\|_1=1} \left| \lambda^T \mathbf{X}^{(L)} \right| \leq \beta, \quad (28)$$

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

Proof. Find the dual of the rescaled training problem by rewriting it as

$$\min_{\theta \in \Theta: \|\mathbf{W}^{(i,l)}\|_1=1} \mathcal{L}_{\mathbf{y}}(\mathbf{z}) + \beta \|\alpha\|_1, \quad \text{s.t.} \quad \mathbf{z} = \sum_{i=1}^{m_L} \alpha_i \mathbf{X}^{(i,L)}. \quad (29)$$

The Lagrangian of problem (29) is $L(\lambda, \theta) = \mathcal{L}_{\mathbf{y}}(\mathbf{z}) + \beta \|\alpha\|_1 - \lambda^T \mathbf{z} + \sum_{i=1}^{m_L} \lambda^T \mathbf{X}^{(i,L)} \alpha_i$. Minimize the Lagrangian over $\mathbf{z}$ and α and use Fenchel duality [69]. The dual of (29) is

$$\max_{\lambda \in \mathbb{R}^N} -\mathcal{L}_{\mathbf{y}}^*(\lambda) \quad \text{s.t.} \quad \max_{\theta \in \Theta: \|\mathbf{W}^{(i,l)}\|_1=1} \left| \lambda^T \mathbf{X}^{(i,L)} \right| \leq \beta, i \in [m_L]. \quad (30)$$

Observe $\mathbf{X}^{(i,L)}$ is of the same form for all $i \in [m_L]$. So the m_L constraints in (30) collapse to just one constraint. Then we can write (30) as (28). $\square$

APPENDIX F

DEEP NARROW NETWORK WITH ABSOLUTE VALUE ACTIVATION

Remark F.1 (Relation to proofs for 1-D case in [7]). *A new challenge in extending the Lasso equivalence such as Theorem IV.2 from the 1-D case in [7] to the multidimensional input case is making the Lasso dictionary finite, as follows. The rescaling step in Lemma E.1 induces the constraint that each intermediate layer weight has ℓ_1 norm 1. In the univariate setting, each intermediate layer weight is a scalar, so the rescaling step constrains them to ± 1 . Thus in the univariate case, the intermediate weights belong to a finite set, which after taking the bidual of the training problem results in a Lasso problem with a finite number of features. When the input has dimension d , the*

first weights also must have dimension d , so the rescaling step constrains the first-layer weights to lie in an ℓ_1 ball, which is an infinite set of values. This results in a semi-infinite dual problem (having a potentially infinite number of constraints) that would lead to a Lasso problem with an infinite dictionary. We resolve this by showing that it suffices to consider only a finite set of constraints corresponding to vertices of the ℓ_1 ball in the dual.

Theorem F.2. *The training problem for a 3-layer deep narrow network and d -dimensional data is equivalent to a Lasso problem with dictionary elements $\mathbf{A}_{i,j} = f_j(\mathbf{x}_i)$ where $\mathbf{x}_i$ is the i^{th} training sample and*

$$f_j(\mathbf{x}) = \frac{|V(\mathbf{x}) - V(\mathbf{x}_{j_0})|}{\left\| (\mathbf{x}_{j_2} - \mathbf{x}'_{j_3}) \times \cdots \times (\mathbf{x}_{j_{2(d-1)}} - \mathbf{x}'_{j_{2d-1}}) \right\|_1} \quad (31)$$

where

$$V(\mathbf{x}) = \text{Vol} \left(\mathbf{x} - \mathbf{x}'_{j_1}, \mathbf{x}_{j_2} - \mathbf{x}'_{j_3}, \dots, \mathbf{x}_{j_{2(d-1)}} - \mathbf{x}'_{j_{2d-1}} \right)$$

provided that $m \geq \|\mathbf{z}^*\|_0$, where $\mathbf{z}^*, \xi^*$ is a Lasso solution. The multi-index $j = (j_{-1}, j_0, j_1, j_2, \dots, j_{2d-1})$ indexes over all $j_{-1}, j_{2k} \in [N]$, $\mathbf{x}'_{j_1} \in \left\{ \frac{\mathbf{x}_{j_{-1}} + \mathbf{x}_{j_0}}{2}, \mathbf{x}_{j_{-1}} \right\}$ and $\mathbf{x}'_{j_{2k+1}} \in \left\{ \mathbf{x}'_{j_1}, R(\mathbf{x}_{j_0}, \mathbf{x}'_{j_1}) \right\} \cup \{ \mathbf{x}_{j_{2k}} - e_l \}_{l \in [d]}$ for $k \geq 1$. An optimal neural network can be reconstructed as $f_3(\mathbf{x}; \theta) = \sum_i z_i^* f_j(\mathbf{x}) + \xi^*$, and the corresponding optimal $\mathbf{W}^{(1)}$ is orthogonal to $\mathbf{x}_{j_2} - \mathbf{x}'_{j_3}, \dots, \mathbf{x}_{j_{2(d-1)}} - \mathbf{x}'_{j_{2d-1}}$.

In this section, we assume $\sigma = |x|$.

Let $n^{(1)}, \dots, n^{(L-1)} \in [N]$. Define the breakplane sets $\mathcal{K}^{(L)} = \{ \mathbf{x}_{n^{(L-1)}} \}$ if $L=2$ and

$$\mathcal{K}^{(L)} = \left\{ \mathbf{X}_{n^{(L-2)}}^{(L-2)}, \frac{\mathbf{X}_{n^{(L-1)}}^{(L-2)} + \mathbf{X}_{n^{(L-2)}}^{(L-2)}}{2} \right\} \quad (32)$$

otherwise. For $\mathbf{x}'_{j_1} \in \mathcal{K}^{(3)}$, let $\mathbf{M}_1 = \mathbf{X} - \mathbf{1}\mathbf{x}'_{j_1}$ and

$$\begin{aligned} f_{\mathbf{x}'_{j_1}} : \mathbb{R}^d &\rightarrow \mathbb{R}^N, & f_{\mathbf{x}'_{j_1}}(\mathbf{W}) &= \mathbf{M}_1 \mathbf{W} \\ & & &= (\mathbf{X} - \mathbf{1}\mathbf{x}'_{j_1}) \mathbf{W} \\ S_{\mathbf{x}'_{j_1}} : \mathbb{R}^d &\rightarrow \{-1, 1\}^N, & S_{\mathbf{x}'_{j_1}}(\mathbf{W}) &= \text{sign} \left(f_{\mathbf{x}'_{j_1}}(\mathbf{W}) \right). \end{aligned} \quad (33)$$

For $\mathbf{d}^{(1)} \in S_{\mathbf{x}'_{j_1}}(\mathbb{R}^d)$, let

$$\mathbf{M}_2 = \left(\text{Diag}(\mathbf{d}^{(1)}) - d_{n^{(2)}}^{(1)} \mathbf{E}_{n^{(2)}} \right) \mathbf{M}_1 \quad (34)$$

and

$$\begin{aligned} f_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}} : \mathbb{R}^d &\rightarrow \mathbb{R}^N, & \left(f_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}}(\mathbf{W}) \right)_n &= (\mathbf{M}_2 \mathbf{W})_n \\ & & &= \left| \left(\mathbf{X}_n^{(1)} - \mathbf{x}'_{j_1} \right) \mathbf{W} \right| - \left| \left(\mathbf{X}_{n^{(2)}}^{(1)} - \mathbf{x}'_{j_1} \right) \mathbf{W} \right| \text{ for } \mathbf{W} \in S_{\mathbf{x}'_{j_1}}^{-1}(\mathbb{R}^d) \\ S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}} : \mathbb{R}^d &\rightarrow \{-1, 1\}^{N \times N}, & S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}}(\mathbf{W}) &= \text{sign} \left(f_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}}(\mathbf{W}) \right) \end{aligned} \quad (35)$$

where $\mathbf{E}_{n^{(2)}}$ denotes a $N \times N$ matrix whose $(n^{(2)})^{\text{th}}$ column is $\mathbf{1}$ and all other elements are 0. These functions (33), (35) represent the sign patterns of a neuron in the first and second layers. Given $n^{(1)}, n^{(2)} \in [N]$, $\mathbf{x}'_{j_1} \in \mathcal{K}^{(3)}$, a fixed sign pattern $\mathbf{d}^{(1)} \in S_{\mathbf{x}'_{j_1}}(\mathbb{R}^d)$, let $\mathbf{W}^{(1)} \in S_{\mathbf{x}'_{j_1}}^{-1}(\mathbf{d}^{(1)})$ and

$$\begin{aligned}\mathcal{X}^{(3)} &= \{\mathbf{x}_n - \mathbf{x}'_{j_1}\}_n \cup \{\mathbf{x}_n - \mathbf{x}_{n(2)}\}_n \cup \{\mathbf{x}_n - R(\mathbf{x}_{n(2)}, \mathbf{a})\}_n \cup \{e_1, \dots, e_d\} \\ \Delta\mathcal{X} &= \left\{ \pm \frac{\times_{i=1}^{d-1} \Delta\mathbf{x}_i}{\|\times_{i=1}^{d-1} \Delta\mathbf{x}_i\|_1} : \Delta\mathbf{x}_i \in \mathcal{X}^{(3)} \right\}.\end{aligned}\quad (36)$$

Let $\delta B = \{\mathbf{W} \in \mathbb{R}^d : \|\mathbf{W}\|_1 = 1\}$ be the boundary of the ℓ_1 ball of radius 1.

Proof of Theorem F.2. In a deep narrow network (23), $\mathbf{W}^{(i,2)}, \dots, \mathbf{W}^{(i,L-1)}, \mathbf{b}^{(i)}, \dots, \mathbf{b}^{(L-1)}$ are scalar-valued. Suppose $\sigma(x) = |x|$. For fixed $\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(L-1)}, \mathbf{b}^{(i)} = \mathbf{b}^{(i)*}$ is optimal if $\mathbf{b}^{(i)*} \in \arg \max_{\mathbf{b}^{(i)}} \max_{\mathbf{b}^{(i+1)}} \dots \max_{\mathbf{b}^{(L-1)}} \left| \sum_{n=1}^N \lambda_n \mathbf{X}_n^{(L)} \right|$, and analogously for $\mathbf{W}^{(i)}$. We will analyze the constraint of the dual (28):

$$\max_{\theta \in \Theta: \|\mathbf{W}^{(1,i)}\|_1 = 1} \left| \lambda^T \mathbf{X}^{(L)} \right| \leq \beta. \quad (37)$$

Satisfying (37) requires $\mathbf{1}^T \lambda = 0$, otherwise $\max_{\theta \in \Theta} |\lambda^T \mathbf{X}^{(L)}| = \infty$. So assume $\mathbf{1}^T \lambda = 0$. We will use the following property: if $f : \mathbb{R} \rightarrow \mathbb{R}$ is a bounded piecewise linear function, $\arg \max_x |f(x)|$ contains a breakplane (or a breakpoint in $\mathbb{R}$) of f . The objective in the dual constraint (37) is $|\lambda^T \mathbf{X}^{(L)}| = \left| \sum_{n=1}^N \lambda_n \mathbf{X}_n^{(L)} \right|$ where

$$\mathbf{X}_n^{(L)} = \left| \mathbf{X}_n^{(L-1)} \mathbf{W}^{(L-1)} + \mathbf{b}^{(L-1)} \right|. \quad (38)$$

The breakplanes of $\mathbf{X}_n^{(L)}$ as a function of $\mathbf{b}^{(L-1)}$ occur at $\mathbf{b}^{(L-1)} = -\mathbf{X}_n^{(L-1)} \mathbf{W}^{(L-1)}$. Therefore for some $n^{(L-1)} \in [N]$, $\mathbf{b}^{(L-1)*} = -\mathbf{X}_{n^{(L-1)}}^{(L-1)} \mathbf{W}^{(L-1)}$ is optimal. Now suppose $L > 2$. Plugging $\mathbf{b}^{(L-1)*}$ into $\mathbf{X}_n^{(L)}$ (38) gives

$$\begin{aligned}\mathbf{X}_n^{(L)} &= \left| \left(\mathbf{X}_n^{(L-1)} - \mathbf{X}_{n^{(L-1)}}^{(L-1)} \right) \mathbf{W}^{(L-1)} \right| \\ &= \left| \left| \mathbf{X}_n^{(L-2)} \mathbf{W}^{(L-2)} + \mathbf{b}^{(L-2)} \right| - \left| \mathbf{X}_{n^{(L-1)}}^{(L-2)} \mathbf{W}^{(L-2)} + \mathbf{b}^{(L-2)} \right| \right|.\end{aligned}\quad (39)$$

By (39), for fixed $n^{(L-1)}$, the set of breakplanes of $\sum_n \lambda_n \mathbf{X}_n^{(L)}$ as a function of $\mathbf{b}^{(L-2)}$ is $\{ -\mathbf{x}'_{n^{(L-2)}} \mathbf{W}^{(L-2)} : \mathbf{x}'_{n^{(L-2)}} \in \mathcal{K}^{(L)}, n^{(L-2)} \in [N] \}$. So there exists $n^{(L-2)} \in [N]$ and $\mathbf{x}'_{n^{(L-2)}} \in \mathcal{K}^{(L)}$ such that

$$\mathbf{b}^{(L-2)*} = -\mathbf{x}'_{n^{(L-2)}} \mathbf{W}^{(L-2)} \quad (40)$$

is optimal.

Now suppose $L=3$. Let $\mathbf{x}'_{j_1} \in \mathcal{K}_{n^{(1)}, n^{(2)}}$. By (40), $\mathbf{b}^{(1)} = -\mathbf{x}'_{j_1} \mathbf{W}^{(1)}$ and plugging this into (39) gives

$$\mathbf{X}_n^{(3)} = \left| \left| \left(\mathbf{X}_n^{(1)} - \mathbf{x}'_{j_1} \right) \mathbf{W}^{(1)} \right| - \left| \left(\mathbf{X}_{n^{(2)}}^{(1)} - \mathbf{x}'_{j_1} \right) \mathbf{W}^{(1)} \right| \right| \quad (41)$$

By (41), the dual constraint (37) is equivalent to the following:

$$\max_{\mathbf{W}^{(2)} \in \{-1, 1\}} \max_{n^{(1)}, n^{(2)} \in [N]} \max_{\mathbf{x}'_{j_1} \in \mathcal{K}^{(3)}} \max_{\mathbf{d}^{(1)} \in S_{\mathbf{x}'_{j_1}}(\mathbb{R}^d)} \max_{\mathbf{d}^{(2)} \in S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}}(\mathbb{R}^d)} \max_{\mathbf{W}^{(1)} \in S_{\mathbf{x}'_{j_1}}^{-1}(\mathbf{d}^{(1)}) \cap S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}}^{-1}(\mathbf{d}^{(2)}) \cap \delta B} \left| \sum_{n=1}^N \lambda_n \mathbf{X}_n^{(L)} \right| \leq \beta, \quad (42)$$

$$\mathbf{1}^T \lambda = 0.$$

For each fixed $\mathbf{d}^{(1)} \in S_{\mathbf{x}'_{j_1}}(\mathbb{R}^d)$ and $\mathbf{d}^{(2)} \in S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}}(\mathbb{R}^d)$, the term

$$\mathbf{X}_n^{(L)} = \left(\left((\mathbf{X}_n - \mathbf{x}'_{j_1}) \mathbf{W}^{(1)} \right) d_n^{(1)} - \left((\mathbf{X}_{n^{(2)}} - \mathbf{x}'_{j_1}) \mathbf{W}^{(1)} \right) d_{n^{(2)}}^{(1)} \right) d_n^{(2)} \quad (43)$$

in the objective is linear in $\mathbf{W}^{(1)}$. Let $B = \{\mathbf{W} \in \mathbb{R}^d : \|\mathbf{W}\|_1 \leq 1\}$ be the filled ℓ_1 ball of radius 1. If we replace δB by B in (42), the constraint (42) remains equivalent since the $\mathbf{W}^{(1)}$ that maximizes the left hand side of (42) will have the largest feasible magnitude. Then for each fixed $n^{(1)}, n^{(2)}, \mathbf{x}'_{j_1}, \mathbf{b}^{(1)}, \mathbf{d}^{(1)}$, the left hand side of the dual constraint (37) is the maximization of a linear function of $\mathbf{W}^{(1)}$ over the polytope

$$\mathcal{P} = S_{\mathbf{x}'_{j_1}}^{-1} \left(\mathbf{d}^{(1)} \right) \cap S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}}^{-1} \left(\mathbf{d}^{(2)} \right) \cap B \quad (44)$$

which is equivalent to the same maximization over the finite set of extremal points of $\mathcal{P}$. Let $\mathbf{E}_{n^{(2)}}$ be a $N \times N$ matrix whose $(n^{(2)})^{\text{th}}$ column is $\mathbf{1}$ and all other elements are 0. The first-layer sign pattern function is the sign of a linear function of $\mathbf{W} : \text{sign}(\mathbf{X} - \mathbf{1}\mathbf{x}'_{j_1}) \mathbf{W}$. The second-layer sign pattern function can also be written as the sign of a linear function of $\mathbf{W}$, as

$$S_{\mathbf{d}, \mathbf{x}'_{j_1}, n}(\mathbf{W}) = \text{sign} \left((\text{Diag}(\mathbf{d}) - d_n \mathbf{E}_n) (\mathbf{X} - \mathbf{1}\mathbf{x}'_{j_1}) \mathbf{W} \right) \quad (45)$$

Therefore we can apply Lemma 19 in [8] as follows. Let $\mathbf{I}$ denote an identity matrix, and let

$$\mathbf{M} = \begin{pmatrix} \mathbf{M}_1 \\ \mathbf{M}_2 \\ \mathbf{I} \end{pmatrix}.$$

The size of $\mathbf{M}$ is $(2N+d) \times d$. For $S \subset [N]$, let $\mathbf{M}_S$ denote a submatrix of $\mathbf{M}$ consisting of the rows indexed by S . The matrix $\mathbf{M}$ depends on $\mathbf{x}'_{j_1}, \mathbf{d}^{(1)}$ and $n^{(2)}$. Let

$$\mathcal{E}_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}} = \left\{ \mathbf{W}^{(1)} \in \mathbb{R}^d : \exists S \subset [N], |S| = d-1, \text{ s.t. } \mathbf{M}_S \mathbf{W}^{(1)} = \mathbf{0}, \text{ rank}(\mathbf{M}_S) = d-1 \right\}. \quad (46)$$

Let $\mathbf{M}_{s_i}$ denote the i^{th} row of the submatrix $\mathbf{M}_S$. By the orthogonality property of generalized cross products (7),

$$\mathcal{E}_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}} \cap \delta B = \left\{ \pm \frac{\times_{i=1}^{d-1} \mathbf{M}_{s_i}}{\left\| \times_{i=1}^{d-1} \mathbf{M}_{s_i} \right\|_1} : \text{rank}(\mathbf{M}_S) = d-1 \right\} \subset \Delta \mathcal{X} \quad (47)$$

where $\Delta \mathcal{X}$ is defined in (36). By a similar argument as Lemma 19 in [8], the set of extremal points of $\mathcal{P}$ is

$$\mathcal{E}_{\mathbf{d}^{(2)}, \mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}} = S_{\mathbf{x}'_{j_1}}^{-1} \left(\mathbf{d}^{(1)} \right) \cap S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}}^{-1} \left(\mathbf{d}^{(2)} \right) \cap \mathcal{E}_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}} \cap \delta B. \quad (48)$$

Therefore for fixed $n^{(1)}, n^{(2)}, \mathbf{x}'_{j_1}$, plugging (47) into (48) gives

$$\bigcup_{\mathbf{d}^{(1)} \in S_{\mathbf{x}'_{j_1}}(\mathbb{R}^d), \mathbf{d}^{(2)} \in S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}}(\mathbb{R}^d)} \mathcal{E}_{\mathbf{d}^{(2)}, \mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}} \subset \Delta \mathcal{X}. \quad (49)$$

Plugging in (49) as a superset of extremal points of $\mathcal{P}$ into (42) shows that the dual constraint (37) is equivalent to

$$\max_{\mathbf{W}^{(2)} \in \{-1, 1\}} \max_{n^{(1)}, n^{(2)} \in [N]} \max_{\mathbf{x}'_{j_1} \in \mathcal{K}_{n^{(1)}, n^{(2)}}} \max_{\mathbf{W}^{(1)} \in \Delta \mathcal{X}} \left| \sum_{n=1}^N \lambda_n \mathbf{X}_n^{(L)} \right| \leq \beta, \mathbf{1}^T \boldsymbol{\lambda} = 0. \quad (50)$$

Using (21), we can write (59) as $\mathbf{b}^{(l)} = -\mathbf{X}_{n^{(l)}}^{(l)}$ for $l > 1$, where $\mathbf{W}^{(l')} = 1$ for $l' > 1$.

The dual of (50) is the Lasso problem (5) where $\mathbf{A}_{i,j}$ is determined by (41), or

$$\mathbf{A}_i = \mathbf{X}^{(3)}(\mathbf{X}). \quad (51)$$

In (51), $\mathbf{X}^{(3)}(\mathbf{X})$ is the output of the network $\mathbf{X}^{(3)}$ with data matrix $\mathbf{X}$ as input, where $\mathbf{W}^{(2)} = 1$ and $\mathbf{b}^{(l)}$ determined by (59) (with $\mathbf{x}'_{j_1} = \mathbf{x}'_{j_1}$), i.e. which have i^{th} element

$$\mathbf{A}_{i,j} = \left| \left| (\mathbf{x}_i - \mathbf{x}'_{j_1}) \mathbf{W}^{(1)} \right| - \left| (\mathbf{x}_{n^{(2)}} - \mathbf{x}'_{j_1}) \mathbf{W}^{(1)} \right| \right| \quad (52)$$

The Lasso problem is the bidual of the rescaled problem, and therefore a lower bound on it, which is equivalent to the original training problem. In fact, Remark F.5 shows that the Lasso problem is equivalent to the original training problem, and the reconstruction holds. Rename $j_0 = n^{(2)}, j_{-1} = n^{(1)}$.

Note that $\mathbf{M}_2 = \text{Diag}(\mathbf{d}) (\mathbf{X} - \mathbf{1}\mathbf{x}'_{j_1}) - d_{n^{(2)}} \mathbf{E}_n (\mathbf{X} - \mathbf{1}\mathbf{x}'_{j_1})$ and so its n^{th} row is $d_n (\mathbf{x}_n - \mathbf{x}'_{j_1}) - d_{n^{(2)}} (\mathbf{x}_{n^{(2)}} - \mathbf{x}'_{j_1}) \in \{\pm (\mathbf{x}_n - \mathbf{x}_{n^{(2)}}), \pm (\mathbf{x}_n - R(\mathbf{x}_{n^{(2)}}, \mathbf{x}'_{j_1}))\}$. By (47), this gives $\mathbf{W}^{(1)}$. Remark B.1 and (17) then give the volume formulation. Then we plug into (52). The orthogonality property of $\mathbf{W}^{(1)}$ holds by the orthogonality property of cross products. $\square$

Proof of Lemma IV.4. Since $j_{-1}, j_0, j_2, j_4, \dots, j_{2(d-1)} \in [N]$, there are N^{d+1} choices for determining $\mathbf{x}_{j_{-1}}, \mathbf{x}_{j_0}, \mathbf{x}_{j_2}, \mathbf{x}_{j_4}, \dots, \mathbf{x}_{j_{2(d-1)}}$. For each j_{-1}, j_0 there are $\left| \left\{ \frac{\mathbf{x}_{j_{-1}} + \mathbf{x}_{j_0}}{2}, \mathbf{x}_{j_{-1}} \right\} \right| = 2$ options for $\mathbf{x}'_{j_1}$. For each $j_{-1}, j_0, \mathbf{x}_{j_{2k}}$, there are $\left| \left\{ \mathbf{x}'_{j_1}, R(\mathbf{x}_{j_0}, \mathbf{x}'_{j_1}) \right\} \cup \{\mathbf{x}_{j_{2k}} - e_l\}_{l \in [d]} \right| = 2 + d$ options for $\mathbf{x}'_{j_{2k+1}}$. And there are 2 options for $s \in \{-1, 1\}$. So the total number of options for j is at most $O(2^2(N^{d+1})(2+d)^{d-1}) = O((Nd)^d)$. The complexity for training a 2-layer network must be exponential in d unless $\mathbf{P} = \mathbf{NP}$ [13]. $\square$

Remark F.3. The 2-layer feature function is

$$f(\mathbf{x}) = \left| (\mathbf{x} - \mathbf{x}_{n^{(1)}}) \frac{\overbrace{\prod_{i=1}^{d-1} \Delta \mathbf{x}_i}^{\mathbf{W}^{(1)} \perp \Delta \mathbf{x}_1, \dots, \Delta \mathbf{x}_{d-1}}}{\left\| \prod_{i=1}^{d-1} \Delta \mathbf{x}_i \right\|_1} \right|. \quad (53)$$

Similarly the 3-layer feature function is

$$f(\mathbf{x}) = \left| \left| (\mathbf{x} - \mathbf{x}'_{j_1}) \frac{\overbrace{\prod_{i=1}^{d-1} \Delta \mathbf{x}_i}^{\mathbf{W}^{(1)} \perp \Delta \mathbf{x}_1, \dots, \Delta \mathbf{x}_{d-1}}}{\left\| \prod_{i=1}^{d-1} \Delta \mathbf{x}_i \right\|_1} \right| - \left| (\mathbf{x}_{n^{(2)}} - \mathbf{x}'_{j_1}) \frac{\overbrace{\prod_{i=1}^{d-1} \Delta \mathbf{x}_i}^{-b^{(2)}}}{\left\| \prod_{i=1}^{d-1} \Delta \mathbf{x}_i \right\|_1} \right| \right|. \quad (54)$$

Definition F.4. Given an optimal solution to the Lasso problem whose features are (58), the L -layer reconstruction is as follows. Set $\alpha = \mathbf{z}^*, \gamma = \gamma^*, \mathbf{W}^{(i,l)} = 1$ for $l > 1$, and for each dictionary column $\mathbf{A}_i$, consider the corresponding $\mathbf{W}^{(1)}, \mathbf{b}^{(l)}$ and let $\mathbf{W}^{(i,1)} = \mathbf{W}^{(1)}$ and $\mathbf{b}^{(i,l)} = \mathbf{b}^{(l)}$. For data feature biases, use Definition IV.7 (in particular, (59) for absolute value) for $\mathbf{b}^{(l)}$. For $L=3$, $\mathbf{W}^{(1)} \in \Delta \mathcal{X}^{(3)}$ (36). This gives a rescaled network. Then, let $\gamma_i = |\alpha_i|^{\frac{1}{L}}$. Finally change variables as $q' = \text{sign}(q)\gamma_i$ for $q \in \{\alpha_i, \mathbf{W}^{(i,l)}\}$ and $\mathbf{b}^{(i,l)'} = \mathbf{b}^{(i,l)} (\gamma_i)^l$ for the reconstructed network. For example, a reconstructed 3-layer neural network with absolute value activation from a Lasso solution $\mathbf{z}^*, \xi^*$ is

$$f_3(\mathbf{x}; \theta) = \sum_j (z_j^*)^{\frac{1}{3}} \left| (z_j^*)^{\frac{1}{3}} \left| (z_j^*)^{\frac{1}{3}} (\mathbf{x} - \mathbf{x}'_{j_1}) \frac{\prod_{i=1}^{d-1} \Delta \mathbf{x}_i^{(j)}}{\left\| \prod_{i=1}^{d-1} \Delta \mathbf{x}_i^{(j)} \right\|_1} \right| - (z_j^*)^{\frac{1}{3}} \left| (z_j^*)^{\frac{1}{3}} (\mathbf{x}_{n^{(2)}} - \mathbf{x}'_{j_1}) \frac{\prod_{i=1}^{d-1} \Delta \mathbf{x}_i^{(j)}}{\left\| \prod_{i=1}^{d-1} \Delta \mathbf{x}_i^{(j)} \right\|_1} \right| \right| + \xi^*. \quad (55)$$

where the index (j) indexes over all possible $\mathbf{W}^{(1)}, \mathbf{x}'_{j_1}, \mathbf{x}_{n(2)}$ corresponding to the j^{th} feature vector.

Remark F.5. The rescaled network in Definition F.4 achieves the same value in the rescaled problem as the Lasso optimal value. Definition F.4 then "un-scales" the weights in the rescaled network to give a reconstructed network that achieves the same objective in the original training problem as the optimal Lasso value. The rescaled network, reconstructed network and the function $\sum_i z_i^* f_j(\mathbf{x}) + \xi^*$ are all equivalent as functions, but have different interpretations of neuron weights.

Proof of Theorem IV.6. The wedge product formulation follows from Theorem IV.6. To show the distance formulation, observe that the features are given by (52) as

$$f_j(\mathbf{x}) = \left| \left| (\mathbf{x} - \mathbf{x}'_{j_1}) \mathbf{W}^{(1)} \right| - \left| (\mathbf{x}_{n(2)} - \mathbf{x}'_{j_1}) \mathbf{W}^{(1)} \right| \right| \quad (56)$$

with $\mathbf{W}^{(1)} = \frac{\mathbf{w}}{\|\mathbf{w}\|_1} \in \Delta \mathcal{X}^{(3)}$ (36), where $\mathbf{w} = \times_{i=1}^{d-1} \Delta \mathbf{x}_i^{(j)}$. Let ℓ_1 and ℓ_2 be the hyperplanes defined by $(\mathbf{x} - R(\mathbf{x}_{n(2)}, \mathbf{x}'_{j_1})) \mathbf{w} = 0$ and $(\mathbf{x} - R(\mathbf{x}_{n(2)}, \mathbf{x}'_{j_1})) \mathbf{w} = 0$, respectively. Let $l = \ell_1 \cup \ell_2$. By Remark B.2,

$$\begin{aligned} f_j(\mathbf{x}) &= \min \left\{ \left| (\mathbf{x} - \mathbf{x}_{n(2)}) \mathbf{W}^{(1)} \right|, \left| (\mathbf{x} - R(\mathbf{x}_{n(2)}, \mathbf{x}'_{j_1})) \mathbf{W}^{(1)} \right| \right\} \\ &= r(\mathbf{w}) \min \left\{ \left| (\mathbf{x} - \mathbf{x}_{n(2)}) \frac{\mathbf{w}}{\|\mathbf{w}\|_2} \right|, \left| (\mathbf{x} - R(\mathbf{x}_{n(2)}, \mathbf{x}'_{j_1})) \frac{\mathbf{w}}{\|\mathbf{w}\|_2} \right| \right\} \\ &= r(\mathbf{w}) \min \{d(\mathbf{x}, \ell_1), d(\mathbf{x}, \ell_2)\} \\ &= r(\mathbf{w}) d(\mathbf{x}, l). \end{aligned} \quad (57)$$

Now rename $j_0 = n^{(2)}, j_{-1} = j_1$. Observe that if $\mathbf{x}'_{j_1} = \frac{\mathbf{x}_{j_{-1}} + \mathbf{x}_{j_0}}{2}$ then $R(\mathbf{x}_{j_0}, \mathbf{x}'_{j_1}) = \mathbf{x}_{x_{j_{-1}}}$. A case analysis on $\mathbf{x}'_{j_1} \in \left\{ \frac{\mathbf{x}_{j_{-1}} + \mathbf{x}_{j_0}}{2}, \mathbf{x}_{j_{-1}} \right\}$ gives (12). $\square$

A. Features for absolute value

For $\sigma(x) = |x|$, the feature function Definition IV.7 reduces to

$$f_j(\mathbf{x}) = \left| \dots \left| \left| \mathbf{x} \mathbf{W}^{(1)} + b^{(1)} \right| + b^{(2)} \right| \dots \left| + b^{(L-1)} \right| \right|. \quad (58)$$

A feature function (58) consists of only adding a bias and taking an absolute value after the first layer and can be viewed a deep narrow network where $m=1, \alpha=1, \xi=0$ and $w^{(i,l)}=1$ for $l>1$ (23). A feature function has *data feature biases* if for some sequence of training samples $\mathbf{x}_{j_1}, \dots, \mathbf{x}_{j_{L-1}}$, the feature function's bias parameters are

$$\begin{aligned} b^{(1)} &= -\mathbf{x}_{j_1} \mathbf{W}^{(1)} \\ b^{(l)} &= - \left| \dots \left| \left| \mathbf{x}_{j_l} \mathbf{W}^{(1)} + b^{(1)} \right| + b^{(2)} \right| \dots \left| + b^{(l-1)} \right| \right|, \end{aligned} \quad (59)$$

for $l \in [L-1]$.

Theorem F.6. The training problems for deep narrow networks with arbitrary depth and input dimension are equivalent to Lasso problems with finite, discrete dictionaries. The set of feature vectors contains a data feature

sub-library and consists of feature functions (58) sampled at training data points. An optimal network is reconstructed as $\sum_i z_i^* f_j(\mathbf{x}) + \xi^*$.

Proof of Theorem F.6. Recall the proof of Theorem F.2 until (40). Continuing this argument of finding a finite set of possible breakplanes of $\lambda^T \mathbf{X}^{(L)}$ as a function of $\mathbf{b}^{(L-l)}$, plugging them in to $\mathbf{X}^{(L)}$ and further expanding $\lambda^T \mathbf{X}^{(L)}$ as a function of $\mathbf{b}^{(L-l-1)}$ until $l=L-2$ shows that for each fixed $\mathbf{W}^{(1)}$, there is a finite set of optimal $\mathbf{b}^{(1)}$, which determines a finite set of optimal $\mathbf{b}^{(2)}$, and so on determining a finite set of all other possible $\mathbf{b}^{(l)}$. We also see that for all $l>1$, $\mathbf{W}^{(l)}=1$ and $\mathbf{W}^{(l)}=-1$ are both optimal. Furthermore, the optimal values of $\mathbf{b}^{(l)}$ include

$$\mathbf{b}^{(l)} = -\mathbf{X}_{n^{(L-l)}}^{(l)} \mathbf{W}^{(l)} \quad (60)$$

for $n^{(L-l)} \in [N]$, consistent with Definition IV.7. We can express $\mathbf{X}_n^{(L)}$ as a function of $\mathbf{W}^{(1)}$, generalizing (41). Next, partition $\mathbb{R}^d$ into a finite set of polytopes $\mathcal{P}_1, \dots, \mathcal{P}_p$ corresponding to all possible sign patterns of activation arguments (generalizing (44)). The objective linearizes for $\mathbf{W}^{(1)}$ within each polytope, so maximizing $|\lambda^T \mathbf{X}^{(L)}|$ is equivalent to maximizing over the finite set of extreme points of the polytope, generalizing (47). So the dual constraint (37) consists of a finite set of linear constraints. As in the proof of Theorem F.2, the bidual of the training problem is a Lasso problem whose dictionary columns are $\mathbf{X}^{(L)}(\mathbf{X})$ (generalizing (51)) over all possible optimal $\mathbf{W}^{(1)}, \mathbf{b}^{(l)}$. A similar reconstruction as Definition F.4 of an optimal neural network from a Lasso solution holds and shows that the Lasso problem and the training problem are equivalent. Note that the biases are those breakpoints that were found in forming the maximizing the dual constraints. And (60) shows that the dictionary contains a data feature sub-library. Note that the 4-layer features plotted in the middle and right plots of Figure 2 depict $\mathbf{W}^{(1)}$ as orthogonal to the difference of training samples. By a similar argument as the proof of Theorem IV.2, this property holds for deeper layers as well. $\square$

APPENDIX G

LEAKY ReLU NETWORKS

[7] shows that the training problem for 2-layer general leaky ReLU networks with 1-D data are equivalent to the Lasso problem (5) where $A_{i,j} = \sigma(x_i - x_j)$. This corresponds to features $f_j(x) = \sigma(x - x_j)$, which are leaky ReLU functions with breakpoints based at data samples x_j [7]. [7] shows that for non-monotone leaky ReLU, 3-layer networks with 1-D data are equivalent to Lasso models with features whose first-layer biases are data points or normalized midpoints.

Proof of Theorem IV.5. By Definition F.4 and (82), the features are

$$f_j(\mathbf{x}) = \sigma \left(\left(\sigma \left((\mathbf{x} - \mathbf{x}'_{j_1}) \mathbf{w}^{(1)} \right) - \sigma \left((\mathbf{x}_{n^{(2)}} - \mathbf{x}'_{j_1}) \mathbf{w}^{(1)} \right) \right) w^{(2)} \right). \quad (61)$$

The breakplanes occur along the zeros of the arguments to σ that depend on $\mathbf{x}$. The breakplane $(\mathbf{x} - \mathbf{x}'_{j_1}) \mathbf{w}^{(1)} = 0$ zeros out the inner activation $\sigma((\mathbf{x} - \mathbf{x}'_{j_1}) \mathbf{w}^{(1)})$. There are two breakplanes that zero out the outer activation, corresponding to two cases: when $(\mathbf{x} - \mathbf{x}'_{j_1}) \mathbf{w}^{(1)}$ and $(\mathbf{x}_{n^{(2)}} - \mathbf{x}'_{j_1}) \mathbf{w}^{(1)}$ have the same sign or not. If they have the same sign, the breakplane is $(\mathbf{x} - \mathbf{x}_{n^{(2)}}) \mathbf{w}^{(1)} = 0$.

Now suppose they don't have the same sign. Then the breakplane is along $a(\mathbf{x} - \mathbf{x}'_{j_1})\mathbf{w}^{(1)} = -b(\mathbf{x}_{n^{(2)}} - \mathbf{x}'_{j_1})\mathbf{w}^{(1)}$ if $(\mathbf{x}_{n^{(2)}} - \mathbf{x}'_{j_1})\mathbf{w}^{(1)} \leq 0$ (and $(\mathbf{x} - \mathbf{x}'_{j_1})\mathbf{w}^{(1)} \geq 0$) or $b(\mathbf{x} - \mathbf{x}'_{j_1})\mathbf{w}^{(1)} = -a(\mathbf{x}_{n^{(2)}} - \mathbf{x}'_{j_1})\mathbf{w}^{(1)}$ (and $(\mathbf{x} - \mathbf{x}'_{j_1})\mathbf{w}^{(1)} \leq 0$) otherwise. Note that each systems is consistent if and only if a and b have the same sign, ie σ is not monotone. This gives the breakplane $(\mathbf{x} - R^\sigma(\mathbf{x}_{n^{(2)}}, \mathbf{x}'_{j_1}))\mathbf{w}^{(1)} = 0$.

For features where $\mathbf{x}'_{j_1}$ is a training sample, Theorem IV.5 is proven. $\square$

Proof sketch of Theorem IV.2: Let $\bar{\sigma}(x) = a(x)_+ - b(-x)_+$ be the *conjugate* of σ . Then $\sigma(x) = \bar{\sigma}(\text{sign}(x))x$. A similar argument as the proof in Theorem F.2 holds for leaky ReLU if we replace absolute value by σ , explicitly write $w^{(2)}$, and include $\bar{\sigma}(x)$ to linearize the activation. Specifically, Theorem F.2 holds for leaky ReLU with the following changes.

In (32), replace $\frac{\mathbf{X}_{n^{(L-1)}}^{(L-2)} + \mathbf{X}_{n^{(L-2)}}^{(L-2)}}{2}$ by the normalized midpoint $m_{\mathbf{X}_{n^{(L-1)}}^{(L-2)}, \mathbf{X}_{n^{(L-2)}}^{(L-2)}}^\sigma$.

In (34) and (43), modify the definition of $\mathbf{M}_2$ by adding $\bar{\sigma}(\cdot)$ terms: for $\mathbf{d}^{(1)} \in S_{\mathbf{x}'_{j_1}}(\mathbb{R}^d)$, $w^{(2)} \in \{-1, 1\}$, $\mathbf{M}_2 = \bar{\sigma}(w^{(2)}) \left(\text{Diag}(\bar{\sigma}(\mathbf{d}^{(1)})) - \bar{\sigma}(d_{n^{(2)}}^{(1)}) \mathbf{E}_{n^{(2)}} \right) \mathbf{M}_1$ in (64) and $\mathbf{X}_n^{(L)} = \left(((\mathbf{X}_n - \mathbf{x}'_{j_1}) \mathbf{W}^{(1)}) \bar{\sigma}(d_n^{(1)}) - ((\mathbf{X}_{n^{(2)}} - \mathbf{x}'_{j_1}) \mathbf{W}^{(1)}) \bar{\sigma}(d_{n^{(2)}}^{(1)}) \right) \bar{\sigma}(d_n^{(2)})$ in (73).

Rename by adding $w^{(2)}$ to the subscripts in the functions in (35) as $f_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}}, S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}}$ and to the sets in (46), (48) as $\mathcal{E}_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}}, \mathcal{E}_{\mathbf{d}^{(2)}, \mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}}$.

Replace absolute value functions with σ in (35), (38), (39), (41), and (52). Multiply the argument of the outer absolute value by $w^{(2)}$ in (39), (41), (52). Note that $w^{(2)} \in \{-1, 1\}$.

(44) now holds for each fixed $n^{(1)}, n^{(2)}, \mathbf{x}'_{j_1}, \mathbf{b}^{(1)}, \mathbf{d}^{(1)}$, and $w^{(2)}$. Modify (45) as $S_{\mathbf{d}, \mathbf{x}'_{j_1}, w^{(2)}, n}(\mathbf{W}) = \text{sign}(w^{(2)} (\text{Diag}(\bar{\sigma}(\mathbf{d})) - \bar{\sigma}(d_n) \mathbf{E}_n) (\mathbf{X} - \mathbf{1}\mathbf{x}'_{j_1}) \mathbf{W})$. Note that the n^{th} row of $\mathbf{M}_2$ is $\bar{\sigma}(w^{(2)}) (\bar{\sigma}(d_n) (\mathbf{x}_n - \mathbf{x}'_{j_1}) - \bar{\sigma}(d_{n^{(2)}}) (\mathbf{x}_{n^{(2)}} - \mathbf{x}'_{j_1})) \in \{\pm c \bar{\sigma}(w^{(2)}) (\mathbf{x}_n - \mathbf{x}_{n^{(2)}}), \pm c \bar{\sigma}(w^{(2)}) (\mathbf{x}_n - R^\sigma(\mathbf{x}_{n^{(2)}}, \mathbf{x}'_{j_1}))\}$ where $c \in \{a, b\}$. However, the $c \bar{\sigma}(w^{(2)})$ terms cancel in the numerator and denominator cross product terms (77) for $\mathbf{W}$.

The proof of Theorem IV.2 holds for both non-monotone and monotone leaky ReLU, except for the last step when writing out the breakplane set and simplifying the rows of $\mathbf{M}_2$.

Proof of Theorem IV.2. Let $n^{(1)}, \dots, n^{(L-1)} \in [N]$. Define the *breakplane sets* $\mathcal{K}^{(L)} = \{\mathbf{x}_{n^{(L-1)}}\}$ if $L=2$, $\mathcal{K}^{(L)} = \{\mathbf{x}_{n^{(L-2)}}\}$ if $L>2$ and σ is monotone, and

$$\mathcal{K}^{(L)} = \left\{ \mathbf{X}_{n^{(L-2)}}^{(L-2)}, m_{\mathbf{X}_{n^{(L-1)}}^{(L-2)}, \mathbf{X}_{n^{(L-2)}}^{(L-2)}}^\sigma \right\} \quad (62)$$

if $L>2$ and σ is not monotone. For $\mathbf{x}'_{j_1} \in \mathcal{K}^{(3)}$, let $\mathbf{M}_1 = \mathbf{X} - \mathbf{1}\mathbf{x}'_{j_1}$ and

$$\begin{aligned} f_{\mathbf{x}'_{j_1}} : \mathbb{R}^d &\rightarrow \mathbb{R}^N, & f_{\mathbf{x}'_{j_1}}(\mathbf{W}) &= \mathbf{M}_1 \mathbf{W} \\ & & &= (\mathbf{X} - \mathbf{1}\mathbf{x}'_{j_1}) \mathbf{W} \end{aligned} \quad (63)$$

$$S_{\mathbf{x}'_{j_1}} : \mathbb{R}^d \rightarrow \{-1, 1\}^N, \quad S_{\mathbf{x}'_{j_1}}(\mathbf{W}) = \text{sign}(f_{\mathbf{x}'_{j_1}}(\mathbf{W})).$$

Note that $\mathbf{W}^{(2)} = w^{(2)} \in \{-1, 1\}$ by Lemma E.1 and $\mathbf{W}^{(2)}$ being a scalar in a deep narrow network. For $\mathbf{d}^{(1)} \in S_{\mathbf{x}'_{j_1}}(\mathbb{R}^d)$, $w^{(2)} \in \{-1, 1\}$, let

$$\mathbf{M}_2 = \bar{\sigma}\left(w^{(2)}\right) \left(\text{Diag}(\bar{\sigma}(\mathbf{d}^{(1)})) - \bar{\sigma}\left(d_{n^{(2)}}^{(1)}\right) \mathbf{E}_{n^{(2)}} \right) \mathbf{M}_1 \quad (64)$$

and

$$\begin{aligned} f_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}} : \mathbb{R}^d \rightarrow \mathbb{R}^N, \quad & \left(f_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}}(\mathbf{W}) \right)_n = (\mathbf{M}_2 \mathbf{W})_n \\ & = \sigma\left(\left(\mathbf{X}_n^{(1)} - \mathbf{x}'_{j_1}\right) \mathbf{W}\right) - \sigma\left(\left(\mathbf{X}_{n^{(2)}}^{(1)} - \mathbf{x}'_{j_1}\right) \mathbf{W}\right) \\ & \text{for } \mathbf{W} \in S_{\mathbf{x}'_{j_1}}^{-1}(\mathbb{R}^d) \\ S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}} : \mathbb{R}^d \rightarrow \{-1, 1\}^{N \times N}, \quad & S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}}(\mathbf{W}) = \text{sign}\left(f_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}}(\mathbf{W})\right) \end{aligned} \quad (65)$$

where $\mathbf{E}_{n^{(2)}}$ denotes a $N \times N$ matrix whose $(n^{(2)})^{\text{th}}$ column is $\mathbf{1}$ and all other elements are 0. These functions (63), (65) represent the sign patterns of a neuron in the first and second layers. Given $n^{(1)}, n^{(2)} \in [N]$, $\mathbf{x}'_{j_1} \in \mathcal{K}^{(3)}$, a fixed sign pattern $\mathbf{d}^{(1)} \in S_{\mathbf{x}'_{j_1}}(\mathbb{R}^d)$, let $\mathbf{W}^{(1)} \in S_{\mathbf{x}'_{j_1}}^{-1}(\mathbf{d}^{(1)})$ and

$$\begin{aligned} \mathcal{X}^{(3)} &= \{\mathbf{x}_n - \mathbf{x}'_{j_1}\}_n \cup \{\mathbf{x}_n - \mathbf{x}_{n^{(2)}}\}_n \cup \{\mathbf{x}_n - R(\mathbf{x}_{n^{(2)}}, \mathbf{a})\}_n \cup \{e_1, \dots, e_d\} \\ \Delta \mathcal{X} &= \left\{ \pm \frac{\times_{i=1}^{d-1} \Delta \mathbf{x}_i}{\|\times_{i=1}^{d-1} \Delta \mathbf{x}_i\|_1} : \Delta \mathbf{x}_i \in \mathcal{X}^{(3)} \right\}. \end{aligned} \quad (66)$$

Let $\delta B = \{\mathbf{W} \in \mathbb{R}^d : \|\mathbf{W}\|_1 = 1\}$ be the boundary of the ℓ_1 ball of radius 1.

In a deep narrow network (23), $\mathbf{W}^{(i,2)}, \dots, \mathbf{W}^{(i,L-1)}, \mathbf{b}^{(l)}, \dots, \mathbf{b}^{(L-1)}$ are scalar-valued. Suppose $\sigma(x)$ is leaky ReLU. For fixed $\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(L-1)}, \mathbf{b}^{(l)} = \mathbf{b}^{(l)*}$ is *optimal* if $\mathbf{b}^{(l)*} \in \arg \max_{\mathbf{b}^{(l)}} \max_{\mathbf{b}^{(l+1)}} \dots \max_{\mathbf{b}^{(L-1)}} \left| \sum_{n=1}^N \lambda_n \mathbf{X}_n^{(L)} \right|$, and analogously for $\mathbf{W}^{(l)}$. We will analyze the constraint of the dual (28):

$$\max_{\theta \in \Theta} \left| \lambda^T \mathbf{X}^{(L)} \right| \leq \beta. \quad (67)$$

Satisfying (67) requires $\mathbf{1}^T \lambda = 0$, otherwise $\max_{\theta \in \Theta} \left| \lambda^T \mathbf{X}^{(L)} \right| = \infty$. So assume $\mathbf{1}^T \lambda = 0$. We will use the following property: if $f : \mathbb{R} \rightarrow \mathbb{R}$ is a bounded piecewise linear function, $\arg \max_x |f(x)|$ contains a breakplane (or a *breakpoint* in $\mathbb{R}$) of f . The objective in the dual constraint (67) is $\left| \lambda^T \mathbf{X}^{(L)} \right| = \left| \sum_{n=1}^N \lambda_n \mathbf{X}_n^{(L)} \right|$ where

$$\mathbf{X}_n^{(L)} = \sigma\left(\mathbf{X}_n^{(L-1)} \mathbf{W}^{(L-1)} + \mathbf{b}^{(L-1)}\right). \quad (68)$$

The breakplanes of $\mathbf{X}_n^{(L)}$ as a function of $\mathbf{b}^{(L-1)}$ occur at $\mathbf{b}^{(L-1)} = -\mathbf{X}_n^{(L-1)} \mathbf{W}^{(L-1)}$. Therefore for some $n^{(L-1)} \in [N]$, $\mathbf{b}^{(L-1)*} = -\mathbf{X}_{n^{(L-1)}}^{(L-1)} \mathbf{W}^{(L-1)}$ is optimal. Now suppose $L > 2$. Plugging $\mathbf{b}^{(L-1)*}$ into $\mathbf{X}_n^{(L)}$ (68) gives

$$\begin{aligned} \mathbf{X}_n^{(L)} &= \sigma\left(\left(\mathbf{X}_n^{(L-1)} - \mathbf{X}_{n^{(L-1)}}^{(L-1)}\right) \mathbf{W}^{(L-1)}\right) \\ &= \sigma\left(\left(\sigma\left(\mathbf{X}_n^{(L-2)} \mathbf{W}^{(L-2)} + \mathbf{b}^{(L-2)}\right) - \sigma\left(\mathbf{X}_{n^{(L-1)}}^{(L-2)} \mathbf{W}^{(L-2)} + \mathbf{b}^{(L-2)}\right)\right) \mathbf{W}^{(L-1)}\right). \end{aligned} \quad (69)$$

By (69) and Lemma D.10 in [7], for fixed $n^{(L-1)}$, the set of breakplanes of $\sum_n \lambda_n \mathbf{X}_n^{(L)}$ as a function of $\mathbf{b}^{(L-2)}$ is $\{-\mathbf{x}'_{n^{(L-2)}} \mathbf{W}^{(L-2)} : \mathbf{x}'_{n^{(L-2)}} \in \mathcal{K}^{(L)}, n^{(L-2)} \in [N]\}$. So there exists $n^{(L-2)} \in [N]$ and $\mathbf{x}'_{n^{(L-2)}} \in \mathcal{K}^{(L)}$ such that

$$\mathbf{b}^{(L-2)*} = -\mathbf{x}'_{n^{(L-2)}} \mathbf{W}^{(L-2)} \quad (70)$$

is optimal.

Now suppose $L=3$. Let $\mathbf{x}'_{j_1} \in \mathcal{K}_{n^{(1)}, n^{(2)}}$. By (70), $\mathbf{b}^{(1)} = -\mathbf{x}'_{j_1} \mathbf{W}^{(1)}$ and plugging this into (69) gives

$$\mathbf{X}_n^{(3)} = \sigma \left(\left(\left(\left(\mathbf{X}_n^{(1)} - \mathbf{x}'_{j_1} \right) \mathbf{W}^{(1)} \right) - \sigma \left(\sigma \left(\mathbf{X}_{n^{(2)}}^{(1)} - \mathbf{x}'_{j_1} \right) \mathbf{W}^{(1)} \right) \right) w^{(2)} \right) \quad (71)$$

By (71), the dual constraint (67) is equivalent to the following:

$$\max_{\mathbf{W}^{(2)} \in \{-1, 1\}} \max_{n^{(1)}, n^{(2)} \in [N]} \max_{\mathbf{x}'_{j_1} \in \mathcal{K}^{(3)}} \max_{\mathbf{d}^{(1)} \in S_{\mathbf{x}'_{j_1}}(\mathbb{R}^d)} \max_{\mathbf{d}^{(2)} \in S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}}(\mathbb{R}^d)} \max_{\mathbf{W}^{(1)} \in S_{\mathbf{x}'_{j_1}}^{-1}(\mathbf{d}^{(1)}) \cap S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}}^{-1}(\mathbf{d}^{(2)}) \cap \delta B} \left| \sum_{n=1}^N \lambda_n \mathbf{X}_n^{(L)} \right| \leq \beta, \quad (72)$$

$$\mathbf{1}^T \lambda = 0.$$

For each fixed $\mathbf{d}^{(1)} \in S_{\mathbf{x}'_{j_1}}(\mathbb{R}^d)$ and $\mathbf{d}^{(2)} \in S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}}(\mathbb{R}^d)$, the term

$$\mathbf{X}_n^{(L)} = \left(\left(\left(\mathbf{X}_n - \mathbf{x}'_{j_1} \right) \mathbf{W}^{(1)} \right) \bar{\sigma} \left(d_n^{(1)} \right) - \left(\left(\mathbf{X}_{n^{(2)}} - \mathbf{x}'_{j_1} \right) \mathbf{W}^{(1)} \right) \bar{\sigma} \left(d_{n^{(2)}}^{(1)} \right) \right) \bar{\sigma} \left(d_n^{(2)} \right) \quad (73)$$

in the objective is linear in $\mathbf{W}^{(1)}$. Let $B = \{\mathbf{W} \in \mathbb{R}^d : \|\mathbf{W}\|_1 \leq 1\}$ be the filled ℓ_1 ball of radius 1. If we replace δB by B in (72), the constraint (72) remains equivalent since the $\mathbf{W}^{(1)}$ that maximizes the left hand side of (72) will have the largest feasible magnitude. Then for each fixed $n^{(1)}, n^{(2)}, \mathbf{x}'_{j_1}, \mathbf{b}^{(1)}, \mathbf{d}^{(1)}, w^{(2)}$, the left hand side of the dual constraint (37) is the maximization of a linear function of $\mathbf{W}^{(1)}$ over the polytope

$$\mathcal{P} = S_{\mathbf{x}'_{j_1}}^{-1}(\mathbf{d}^{(1)}) \cap S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}}^{-1}(\mathbf{d}^{(2)}) \cap B \quad (74)$$

which is equivalent to the same maximization over the finite set of extremal points of $\mathcal{P}$. Let $\mathbf{E}_{n^{(2)}}$ be a $N \times N$ matrix whose $(n^{(2)})^{\text{th}}$ column is $\mathbf{1}$ and all other elements are 0. The first-layer sign pattern function is the sign of a linear function of $\mathbf{W} : \text{sign}(\mathbf{X} - \mathbf{1}\mathbf{x}'_{j_1}) \mathbf{W}$. The second-layer sign pattern function can also be written as the sign of a linear function of $\mathbf{W}$, as

$$S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}}(\mathbf{W}) = \text{sign} \left(w^{(2)} (\text{Diag}(\bar{\sigma}(\mathbf{d})) - \bar{\sigma}(d_n) \mathbf{E}_n) (\mathbf{X} - \mathbf{1}\mathbf{x}'_{j_1}) \mathbf{W} \right) \quad (75)$$

. Therefore we can apply Lemma 19 in [8] as follows. Let $\mathbf{I}$ denote an identity matrix, and let

$$\mathbf{M} = \begin{pmatrix} \mathbf{M}_1 \\ \mathbf{M}_2 \\ \mathbf{I} \end{pmatrix}.$$

The size of $\mathbf{M}$ is $(2N+d) \times d$. For $S \subset [N]$, let $\mathbf{M}_S$ denote a submatrix of $\mathbf{M}$ consisting of the rows indexed by S . The matrix $\mathbf{M}$ depends on $\mathbf{x}'_{j_1}, \mathbf{d}^{(1)}$ and $n^{(2)}$. Let

$$\mathcal{E}_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}} = \left\{ \mathbf{W}^{(1)} \in \mathbb{R}^d : \exists S \subset [N], |S| = d-1, \text{ s.t. } \mathbf{M}_S \mathbf{W}^{(1)} = \mathbf{0}, \text{ rank}(\mathbf{M}_S) = d-1 \right\}. \quad (76)$$

Let $\mathbf{M}_{s_i}$ denote the i^{th} row of the submatrix $\mathbf{M}_S$. By the orthogonality property of generalized cross products (7),

$$\mathcal{E}_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}} \cap \delta B = \left\{ \pm \frac{\times_{i=1}^{d-1} \mathbf{M}_{s_i}}{\|\times_{i=1}^{d-1} \mathbf{M}_{s_i}\|_1} : \text{rank}(\mathbf{M}_S) = d-1 \right\} \subset \Delta \mathcal{X} \quad (77)$$

where $\Delta\mathcal{X}$ is defined in (66). By a similar argument as Lemma 19 in [8], the set of extremal points of $\mathcal{P}$ is

$$\mathcal{E}_{\mathbf{d}^{(2)}, \mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}} = S_{\mathbf{x}'_{j_1}}^{-1} \left(\mathbf{d}^{(1)} \right) \cap S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}}^{-1} \left(\mathbf{d}^{(2)} \right) \cap \mathcal{E}_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}} \cap \delta B. \quad (78)$$

Therefore for fixed $n^{(1)}, n^{(2)}, \mathbf{x}'_{j_1}$, plugging (77) into (78) gives

$$\bigcup_{\mathbf{d}^{(1)} \in S_{\mathbf{x}'_{j_1}}(\mathbb{R}^d), \mathbf{d}^{(2)} \in S_{\mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}}(\mathbb{R}^d)} \mathcal{E}_{\mathbf{d}^{(2)}, \mathbf{d}^{(1)}, \mathbf{x}'_{j_1}, n^{(2)}, w^{(2)}} \subset \Delta\mathcal{X}. \quad (79)$$

Plugging in (79) as a superset of extremal points of $\mathcal{P}$ into (72) shows that the dual constraint (67) is equivalent to

$$\max_{\mathbf{W}^{(2)} \in \{-1, 1\}} \max_{n^{(1)}, n^{(2)} \in [N]} \max_{\mathbf{x}'_{j_1} \in \mathcal{K}_{n^{(1)}, n^{(2)}}} \max_{\mathbf{W}^{(1)} \in \Delta\mathcal{X}} \left| \sum_{n=1}^N \lambda_n \mathbf{X}_n^{(L)} \right| \leq \beta, \mathbf{1}^T \lambda = 0. \quad (80)$$

The dual of (80) is the Lasso problem (5) where $\mathbf{A}_{i,j}$ is determined by (71), or

$$\mathbf{A}_i = \mathbf{X}^{(3)}(\mathbf{X}). \quad (81)$$

In (81), $\mathbf{X}^{(3)}(\mathbf{X})$ is the output of the network $\mathbf{X}^{(3)}$ with data matrix $\mathbf{X}$ as input, where $\mathbf{W}^{(2)} = \mathbf{1}$ and $\mathbf{b}^{(l)}$ determined by $\mathbf{b}^{(l)} = -\mathbf{X}_{n^{(l)}}^{(l)}$, i.e. which have i^{th} element

$$\mathbf{A}_{i,j} = \sigma \left(\left(\sigma \left((\mathbf{x}_i - \mathbf{x}'_{j_1}) \mathbf{W}^{(1)} \right) - \sigma \left((\mathbf{x}_{n^{(2)}} - \mathbf{x}'_{j_1}) \mathbf{W}^{(1)} \right) \right) w^{(2)} \right). \quad (82)$$

The Lasso problem is the bidual of the rescaled problem, and therefore a lower bound on it, which is equivalent to the original training problem. In fact, Remark F.5 shows that the Lasso problem is equivalent to the original training problem, and the reconstruction holds.

Rename $j_0 = n^{(2)}, j_{-1} = n^{(1)}, s = w^{(2)}$. Note that by a similar argument as the proof of Theorem IV.5, the n^{th} row of $\mathbf{M}_2$ is $\bar{\sigma}(w^{(2)}) \left(\bar{\sigma}(d_n) (\mathbf{x}_n - \mathbf{x}'_{j_1}) - \bar{\sigma}(d_{n^{(2)}}) (\mathbf{x}_{n^{(2)}} - \mathbf{x}'_{j_1}) \right) \in \left\{ \pm c \bar{\sigma}(w^{(2)}) (\mathbf{x}_n - \mathbf{x}_{n^{(2)}}), \pm c \bar{\sigma}(w^{(2)}) (\mathbf{x}_n - R^\sigma(\mathbf{x}_{n^{(2)}}, \mathbf{x}'_{j_1})) \right\}$ where $c \in \{a, b\}$ if σ is not monotone, and otherwise the n^{th} row is in $\{\pm c \bar{\sigma}(w^{(2)}) (\mathbf{x}_n - \mathbf{x}_{n^{(2)}})\}$. However, the $c \bar{\sigma}(w^{(2)})$ terms cancel in the numerator and denominator cross product terms (77) for $\mathbf{W}^{(1)}$. Remark B.1 and (17) then give the volume formulation. Then we plug into (82). The orthogonality property of $\mathbf{W}^{(1)}$ holds by the orthogonality property of cross products. $\square$

Proof of Theorem IV.8. Recall the proof of Theorem IV.2 until (70). Continuing this argument of finding a finite set of possible breakplanes of $\lambda^T \mathbf{X}^{(L)}$ as a function of $\mathbf{b}^{(L-l)}$, plugging them in to $\mathbf{X}^{(L)}$ and further expanding $\lambda^T \mathbf{X}^{(L)}$ as a function of $\mathbf{b}^{(L-l-1)}$ until $l=L-2$ shows that for each fixed $\mathbf{W}^{(1)}$, there is a finite set of optimal $\mathbf{b}^{(1)}$, which determines a finite set of optimal $\mathbf{b}^{(2)}$, and so on determining a finite set of all other possible $\mathbf{b}^{(l)}$. Furthermore, the optimal values of $\mathbf{b}^{(l)}$ include

$$\mathbf{b}^{(l)} = -\mathbf{X}_{n^{(L-l)}}^{(l)} \mathbf{W}^{(l)} \quad (83)$$

for $n^{(L-l)} \in [N]$. We can express $\mathbf{X}_n^{(L)}$ as a function of $\mathbf{W}^{(1)}$, generalizing (71). Next, partition $\mathbb{R}^d$ into a finite set of polytopes $\mathcal{P}_1, \dots, \mathcal{P}_p$ corresponding to all possible sign patterns of activation arguments (generalizing (74)). The objective linearizes for $\mathbf{W}^{(1)}$ within each polytope, so maximizing $|\lambda^T \mathbf{X}^{(L)}|$ is equivalent to maximizing over the

finite set of extreme points of the polytope, generalizing (77). So the dual constraint (67) consists of a finite set of linear constraints. As in the proof of Theorem IV.2, the bidual of the training problem is a Lasso problem whose dictionary columns are $\mathbf{X}^{(L)}(\mathbf{X})$ (generalizing (81)) over all possible optimal $\mathbf{W}^{(1)}, \mathbf{b}^{(l)}$. A similar reconstruction as Definition F.4 of an optimal neural network from a Lasso solution holds and shows that the Lasso problem and the training problem are equivalent. Note that the biases are those breakpoints that were found in forming the maximizing the dual constraints. And (83) shows that the dictionary contains a data feature sub-library. Note that the 4-layer features plotted in the middle and right plots of Figure 2 depict $\mathbf{W}^{(1)}$ as orthogonal to the difference of training samples. By a similar argument as the proof of Theorem IV.2, this property holds for deeper layers as well. $\square$

Theorem G.1. *The training problem for a 3-layer deep narrow network with monotone leaky ReLU activation and d -dimensional data is equivalent to a Lasso problem with dictionary elements $\mathbf{A}_{i,j} = f_j(\mathbf{x}_i)$ where $\mathbf{x}_i$ is the i^{th} training sample and*

$$f_j(\mathbf{x}) = \frac{\sigma(s(V(\mathbf{x}) - V(\mathbf{x}_{j_0})))}{\left\| (\mathbf{x}_{j_2} - \mathbf{x}'_{j_3}) \times \cdots \times (\mathbf{x}_{j_{2(d-1)}} - \mathbf{x}'_{j_{2d-1}}) \right\|_1} \quad (84)$$

where

$$V(\mathbf{x}) = \sigma \left(\text{Vol}_s \left(\mathbf{x} - \mathbf{x}_{j_1}, \mathbf{x}_{j_2} - \mathbf{x}'_{j_3}, \cdots, \mathbf{x}_{j_{2(d-1)}} - \mathbf{x}'_{j_{2d-1}} \right) \right)$$

provided that $m \geq \|\mathbf{z}^*\|_0$, where $\mathbf{z}^*, \xi^*$ is a Lasso solution. The multi-index $j = (j_1, j_2, \cdots, j_{2d-1})$ indexes over all $j_1, j_{2k} \in [N]$ and $\mathbf{x}'_{j_{2k+1}} \in \{\mathbf{x}_{j_1}\} \cup \{\mathbf{x}_{j_{2k}} - e_l\}_{l \in [d]}$ for $k \geq 1$, and $s \in \{-1, 1\}$. An optimal neural network can be reconstructed as $f_3(\mathbf{x}; \theta) = \sum_i z_i^* f_j(\mathbf{x}) + \xi^*$, and the corresponding optimal $\mathbf{W}^{(1)}$ is orthogonal to $\mathbf{x}_{j_2} - \mathbf{x}'_{j_3}, \cdots, \mathbf{x}_{j_{2(d-1)}} - \mathbf{x}'_{j_{2d-1}}$.

Proof. The same proof as for Theorem G.1 (in Section G) applies except for the last step, as the breakpoint set is different for monotone activations and the rows of $\mathbf{M}_2$ do not induce reflections. The indexing of terms in (11) simplify to: the multi-index $j = (j_{-1}, j_0, j_1, j_2, \cdots, j_{2d-1})$ indexes over all $j_{-1}, j_{2k} \in [N], \mathbf{x}'_{j_1} \in \{\mathbf{x}_{j_{-1}}\}$ and $\mathbf{x}'_{j_{2k+1}} \in \{\mathbf{x}'_{j_1}\} \cup \{\mathbf{x}_{j_{2k}} - e_l\}_{l \in [d]}$ for $k \geq 1$, and $s \in \{-1, 1\}$. Therefore we can replace the index j_{-1} by j_1 . $\square$

APPENDIX H

DEEP FEATURES WITH HIGHER ORDER REFLECTIONS

We define $\mathbf{x}$ to be an order- l reflection of a set $\mathcal{S}^{(l)} = \{\mathbf{x}_0, \cdots, \mathbf{x}_k\}$ if $\mathbf{x}$ can be written recursively in the form $R_l = R_l(\mathcal{S}^{(l)}) \in \{R(R_{l-1}(\mathcal{S}^{(l-1)}), \mathbf{x}'), R(\mathbf{x}', R_{l-1}(\mathcal{S}^{(l)}))\}$ where $\mathcal{S}^{(l-1)} \subset \mathcal{S}^{(l)}$ with possibly repeating elements and $|\mathcal{S}^{(l)}| = l+1$. Note that an order- k reflection is also an order- j reflection for any $j \geq k$. In the following lemma, let $a^{(L)}, b^{(L)} \in \mathbb{R}, \mathbf{c}^{(L)} \in \mathbb{R}^N, j_1, \cdots, j_L \in [N]$. For $l \in [L-1]$, recursively define $a^{(l)} = |a^{(l+1)} - c_{j_{l+1}}^{(l+1)}|, b^{(l)} = |b^{(l+1)} - c_{j_{l+1}}^{(l+1)}|, c_{j_l}^{(l)} = |c_{j_l}^{(l+1)} - c_{j_{l+1}}^{(l+1)}|$. Let $\mathcal{S}^{(l)} = \{a^{(l)}, b^{(l)}\} \cup \{c_{l'}^{(l)} : l' \leq l\}$. We will see that this recursively models the structure of a feature function in a data feature sub-library.

Lemma H.1. *Let $l \in [L]$. Let $R_0^{(l,+)} = R_0^{(l,-)} = R(a^{(l)}, b^{(l)})$. For $l' \in \{0, \cdots, L-l-1\}$, there exist $l+l'+1$ -order reflections $R_{2(l'+1)}^{(l+l'+1,+)}, R_{2(l'+1)}^{(l+l'+1,-)}$ of $\mathcal{S}^{(l+l'+1)}$ such that*

$$R_{2l'}^{(l+l', \pm)} \in \left\{ R_{2(l'+1)}^{(l+l'+1,+)} - c_{l+l'+1}^{(l+l'+1)}, c_{l+l'+1}^{(l+l'+1)} - R_{2(l'+1)}^{(l+l'+1,-)} \right\}. \quad (85)$$

for $R_{2l'}^{(l+l', \pm)} \in \{R_{2l'}^{(l+l', +)}, R_{2l'}^{(l+l', -)}\}$.

Proof of Lemma H.1. Observe that for any l ,

$$\begin{aligned}
R(a^{(l)}, b^{(l)}) &= R\left(\left|a^{(l+1)} - c_{j_{l+1}}^{(l+1)}\right|, \left|b^{(l+1)} - c_{j_{l+1}}^{(l+1)}\right|\right) \\
&\in \pm \left\{R\left(a^{(l+1)} - c_{j_{l+1}}^{(l+1)}, b^{(l+1)} - c_{j_{l+1}}^{(l+1)}\right), R\left(a^{(l+1)} - c_{j_{l+1}}^{(l+1)}, c_{j_{l+1}}^{(l+1)} - b^{(l+1)}\right)\right\} \\
&= \pm \left\{-a^{(l+1)} + 2b^{(l+1)} - c_{j_{l+1}}^{(l+1)}, -a^{(l+1)} - 2b^{(l+1)} + 3c_{j_{l+1}}^{(l+1)}\right\} \\
&= \pm \left\{R\left(a^{(l+1)}, b^{(l+1)}\right) - c_{j_{l+1}}^{(l+1)} = c_{j_{l+1}}^{(l+1)} - R\left(a^{(l+1)}, b^{(l+1)}\right), \right. \\
&\quad \left. R\left(a^{(l+1)}, R\left(b^{(l+1)}, c_{j_{l+1}}^{(l+1)}\right)\right) - c_{j_{l+1}}^{(l+1)} = c_{j_{l+1}}^{(l+1)} - R\left(a^{(l+1)}, c_{j_{l+1}}^{(l+1)}\right), b^{(l+1)}\right\}.
\end{aligned} \tag{86}$$

Therefore for some $a^{(l+1)'}, b^{(l+1)'}, c^{(l+1)'} \in \{a^{(l+1)}, b^{(l+1)}, c_{j_{l+1}}^{(l+1)}\}$ and

$$R_2^{(l+1, +)}, R_2^{(l+1, -)} \in \left\{R\left(R\left(a^{(l+1)'}, b^{(l+1)'}\right), c^{(l+1)'}\right), R\left(c^{(l+1)'}, R\left(a^{(l+1)'}, b^{(l+1)'}\right)\right)\right\} \tag{87}$$

which are order-2 reflections of $\{a^{(l+1)}, b^{(l+1)}, c_{j_{l+1}}^{(l+1)}\}$, we have

$$R(a^{(l)}, b^{(l)}) = \left\{c_{j_{l+1}}^{(l+1)} - R_2^{(l+1, -)} = R_2^{(l+1, +)} - c_{j_{l+1}}^{(l+1)}\right\}. \tag{88}$$

Applying (88) for $l' = l + 1$ gives

$$R(a^{(l+1)}, b^{(l+1)}) \in \left\{c_{j_{l+2}}^{(l+2)} - R_2^{(l+2, -)} = R_2^{(l+2, +)} - c_{j_{l+2}}^{(l+2)}\right\} \tag{89}$$

for some $R_2^{(l+2, +)}, R_2^{(l+2, -)}$ that are order-2 reflections of $\{a^{(l+2)}, b^{(l+2)}, c_{j_{l+2}}^{(l+2)}\}$. Plugging (89) into (87) shows that for $R_2^{(l+1, \pm)} \in \{R_2^{(l+1, +)}, R_2^{(l+1, -)}\}$ we have

$$\begin{aligned}
R_2^{(l+1, \pm)} &= R\left(R\left(a^{(l+1)'}, b^{(l+1)'}\right), c^{(l+1)'}\right) \\
&= 2 \left|c^{(l+2)'} - c_{j_{l+2}}^{(l+2)}\right| - \left\{c_{j_{l+2}}^{(l+2)} - R_2^{(l+2, -)} = R_2^{(l+2, +)} - c_{j_{l+2}}^{(l+2)}\right\} \\
&\in \left\{-c_{j_{l+2}}^{(l+2)} + 2c^{(l+2)'} - R_2^{(l+2, +)}, c_{j_{l+2}}^{(l+2)} - 2c^{(l+2)'} + R_2^{(l+2, -)}\right\} \\
&= \left\{R\left(R_2^{(l+2, +)}, c^{(l+2)'}\right) - c_{j_{l+2}}^{(l+2)} = c_{j_{l+2}}^{(l+2)} - R\left(R_2^{(l+2, +)}, c^{(l+2)'}\right), \right. \\
&\quad \left. = c_{j_{l+2}}^{(l+2)} - R\left(R_2^{(l+2, -)}, c^{(l+2)'}\right) = R\left(R_2^{(l+2, -)}, c^{(l+2)'}\right) - c_{j_{l+2}}^{(l+2)}\right\}
\end{aligned} \tag{90}$$

or alternatively,

$$\begin{aligned}
R_2^{(l+1, \pm)} &= R\left(c^{(l+1)'}, R\left(a^{(l+1)'}, b^{(l+1)'}\right)\right) \\
&= 2 \left\{c_{j_{l+2}}^{(l+2)} - R_2^{(l+2, -)} = R_2^{(l+2, +)} - c_{j_{l+2}}^{(l+2)}\right\} - \left|c^{(l+2)'} - c_{j_{l+2}}^{(l+2)}\right| \\
&\in \left\{-c_{j_{l+2}}^{(l+2)} + 2R_2^{(l+2, +)} - c^{(l+2)'}, c_{j_{l+2}}^{(l+2)} - 2R_2^{(l+2, -)} + c^{(l+2)'}\right\} \\
&= \left\{R\left(c^{(l+2)'}, R_2^{(l+2, +)}\right) - c_{j_{l+2}}^{(l+2)} = c_{j_{l+2}}^{(l+2)} - R\left(c^{(l+2)'}, R_2^{(l+2, +)}\right), \right. \\
&\quad \left. c_{j_{l+2}}^{(l+2)} - R\left(c^{(l+2)'}, R_2^{(l+2, -)}\right) = R\left(c^{(l+2)'}, R_2^{(l+2, -)}\right) - c_{j_{l+2}}^{(l+2)}\right\}
\end{aligned} \tag{91}$$

for some $c^{(l+2)'} \in \{a^{(l+2)}, b^{(l+2)}, c_{j_{l+1}}^{(l+2)}\}$. Therefore (90) and (91) show that in all cases,

$$R_2^{(l+1,\pm)} \in \left\{ R_4^{(l+2,+)} - c_{j_{l+2}}^{(l+2)}, c_{j_{l+2}}^{(l+2)} - R_4^{(l+2,-)} \right\} \quad (92)$$

for some order-4 reflections $R_4^{(l+2,+)}$, $R_4^{(l+2,-)}$ of $\{a^{(l+2)}, b^{(l+2)}, c_{j_{l+1}}^{(l+2)}, c_{j_{l+2}}^{(l+2)}\}$.

Applying a similar argument used to reach (92) for $l' = l + 2$ shows that

$$R_2^{(l+2,\pm)} \in \left\{ R_4^{(l+3,+)} - c_{j_{l+3}}^{(l+3)}, c_{j_{l+3}}^{(l+3)} - R_4^{(l+3,-)} \right\} \quad (93)$$

for some order-4 reflections $R_4^{(l+3,+)}$, $R_4^{(l+3,-)}$ of $a^{(l+3)}, b^{(l+3)}, c_{j_{l+2}}^{(l+3)}, c_{j_{l+3}}^{(l+3)}$. Now plug (93) into the last lines of (90) and (91). Applying the same argument as (90) and (91) but changing the reflection order subscripts from 1 to 2 and from 2 to 4, and adding 1 to the layer superscripts $l + 1, l + 2$ shows that

$$R_4^{(l+2,\pm)} \in \left\{ R_6^{(l+3,+)} - c_{j_{l+3}}^{(l+3)}, c_{j_{l+3}}^{(l+3)} - R_6^{(l+3,-)} \right\} \quad (94)$$

for some order-6 reflections $R_6^{(l+3,+)}$, $R_6^{(l+3,-)}$ of $\{a^{(l+3)}, b^{(l+3)}, c_{j_{l+1}}^{(l+3)}, c_{j_{l+2}}^{(l+3)}, c_{j_{l+3}}^{(l+3)}\}$.

Comparing (88), (92), (94) and repeating the same argument shows that

$$\begin{aligned} R_1^l &= R(a^{(l)}, b^{(l)}) \\ &\in \left\{ R_2^{(l+1,+)} - c^{(l+1)} = c^{(l+1)} - R_2^{(l+1,-)} \right\} \\ R_2^{(l+1,\pm)} &\in \left\{ R_4^{(l+2,+)} - c^{(l+2)}, c^{(l+2)} - R_4^{(l+2,-)} \right\} \\ R_4^{(l+2,\pm)} &\in \left\{ R_6^{(l+3,+)} - c^{(l+3)}, c^{(l+3)} - R_6^{(l+3,-)} \right\} \\ &\vdots \\ R_{2^{l'}}^{(l+l',\pm)} &\in \left\{ R_{2^{(l'+1)}}^{(l+l'+1,+)} - c^{(l+l'+1)}, c^{(l+l'+1)} - R_{2^{(l'+1)}}^{(l+l'+1,-)} \right\}. \end{aligned} \quad (95)$$

This completes the proof. $\square$

Proof sketch of Theorem IV.9: We can easily show that 2 and 3-layer networks can be written using reflections. We extend this to deeper layers by first showing that reflections of $\mathbf{X}^{(l)}$ can be expressed as higher order reflections of $\mathbf{X}^{(k)}$ for $k < l$ (Lemma H.1). Then, we recursively plug the higher-order reflections into the deeper networks (99). This will show that for $l > 2$, the neural net is $\mathbf{X}^{(L)} = \left| \mathbf{X}^{(L-l)} - R_{2^{(l-3)}}^{(L-l)} \right|$ for some order- $2(l-3)$ reflection of the parallel units (features) at layer $L-l$. Finally, plug in $l=L-1$.

Proof of Theorem IV.9. The features $f_j(\mathbf{x})$ (58) of the data feature sub-library have bias parameters (59). In other words, $f_j(\mathbf{x}) = \mathbf{X}^{(L)}(\mathbf{x})$ where for all $l \in [L-1]$ there exists $j_l \in [N]$ such that for all $\mathbf{x} \in \mathbb{R}^d$,

$$\mathbf{X}^{(l+1)}(\mathbf{x}) = \left| \mathbf{X}^{(l)}(\mathbf{x}) - \mathbf{X}^{(l)}(\mathbf{x}_{j_l}) \right|. \quad (96)$$

For shorthand, denote $\mathbf{X}^{(l)} = \mathbf{X}^{(l)}(\mathbf{x})$, $\mathbf{X}_j^{(l)} = \mathbf{X}^{(l)}(\mathbf{x}_{j_l})$. Then

$$\begin{aligned} \mathbf{X}^{(L)} &= \left| \mathbf{X}^{(L-1)} - \mathbf{X}_{j_{L-1}}^{(L-1)} \right| \\ &= \left| \left| \mathbf{X}^{(L-2)} - \mathbf{X}_{j_{L-2}}^{(L-2)} \right| - \left| \mathbf{X}_{j_{L-1}}^{(L-2)} - \mathbf{X}_{j_{L-2}}^{(L-2)} \right| \right| \\ &\in \left\{ \left| \mathbf{X}^{(L-2)} - \mathbf{X}_{j_{L-2}}^{(L-2)} \right|, \left| \mathbf{X}^{(L-2)} - R\left(\mathbf{X}_{j_{L-1}}^{(L-2)}, \mathbf{X}_{j_{L-2}}^{(L-2)}\right) \right| \right\}. \end{aligned} \quad (97)$$

Therefore

$$\mathbf{X}^{(L)} = \left| \mathbf{X}^{(L-2)} - R_1^{(L-2)} \right| \quad (98)$$

for some order-1 reflection $R_1^{(L-2)} \in \left\{ R\left(\mathbf{X}_{j_{L-2}}^{(L-2)}, \mathbf{X}_{j_{L-2}}^{(L-2)}\right), R\left(\mathbf{X}_{j_{L-1}}^{(L-2)}, \mathbf{X}_{j_{L-2}}^{(L-2)}\right) \right\}$. Expand $\mathbf{X}^{(L-2)}$ in (98) with (96) and apply Lemma H.1 to $R_1^{(L-2)}$. Specifically, we set $a^{(l)}, b^{(l)} \in \left\{ \mathbf{X}_{j_{L-1}}^{(L-2)}, \mathbf{X}_{j_{L-2}}^{(L-2)} \right\}$, let j_a, j_b be the subscripts (j_{L-1} or j_{L-2}) of $a^{(l)}, b^{(l)}$, and let $a^{(l+1)} = \mathbf{X}_{j_a}^{(L-3)}, b^{(l+1)} = \mathbf{X}_{j_b}^{(L-3)}, c^{(l+1)} = \mathbf{X}_{j_{L-3}}^{(L-3)}$ in (88). This yields

$$\begin{aligned} \mathbf{X}^{(L)} &= \left| \left| \mathbf{X}^{(L-3)} - \mathbf{X}_{j_{L-3}}^{(L-3)} \right| - \left\{ R_2^{(L-3,+)} - \mathbf{X}_{j_{L-3}}^{(L-3)} = \mathbf{X}_{j_{L-3}}^{(L-3)} - R_2^{(L-3,-)} \right\} \right| \\ &= \left| \mathbf{X}^{(L-3)} - R_2^{(L-3,\pm)} \right|. \end{aligned} \quad (99)$$

Next, repeat the same argument (99) but use (93) instead of (88) to get $\mathbf{X}^{(L)} = \left| \mathbf{X}^{(L-4)} - R_2^{(L-4,\pm)} \right|$. Continue repeating the same argument by using (85) for general l . Specifically, for fixed $j_a, j_b \in [N]$, for all $l \in [L]$, let $a^{(l)} = \mathbf{X}_{j_a}^{(L-l)}, b^{(l)} = \mathbf{X}_{j_b}^{(L-l)}$ and $c_{j_l}^{(l)} = \mathbf{X}_{j_{L-l}}^{(L-l)}$ in (93). We arrive at $\mathbf{X}^{(L)} = \left| \mathbf{X} - R_{2(L-3)} \right|$. Thus for $L > 3$, the neural net has breakplanes at reflections of training data of up to order-2($L-3$). The network has breakplanes at reflections of order 0 for $L = 2$ and order 1 for $L = 3$ (98), respectively. $\square$

APPENDIX I

PROOFS FOR SECTION V

As in Section V, assume the data is 1-D labeled in descending order: $x_1 \geq \dots \geq x_N$, and $f^{(l)}$ are Cantor features. We first focus on showing self-symmetry of an l -layer feature over $[0, 1]$ with a $(l+1)$ -layer feature over the first third of $[0, 1]$, and then extend this to the second third. The proofs are mostly based on induction using the recursive form of the features (13) and the r -adic nature of the data.

For convenience, let $q = \frac{1}{r}$. The next lemma shows that scaling a feature by q is equivalent to scaling the argument of a modified feature by q . In other words, we can “bring q inside” the absolute value functions.

Lemma I.1. For $l \geq 2$, $qf^{(l)}(x) = \tilde{f}^{(l)}(qx)$ where $\tilde{f}^{(2)}(x) = |x - qx_1|$, $\tilde{f}^{(l+1)}(x) = \left| \tilde{f}^{(l)}(x) - \tilde{f}^{(l)}(qx_l) \right|$.

Proof of Lemma I.1. By induction on l . Base case: for $l = 2$, we have $qf^{(2)}(x) = q|x - x_1| = |qx - qx_1| = \tilde{f}^{(2)}(qx)$. Now let $l \geq 2$. Suppose $qf^{(l)}(x) = \tilde{f}^{(l)}(qx)$. Then $qf^{(l+1)}(x) = q|f^{(l)}(x) - f^{(l)}(x_l)| = |qf^{(l)}(x) - qf^{(l)}(x_l)| = |\tilde{f}^{(l)}(qx) - \tilde{f}^{(l)}(qx_l)| = \tilde{f}^{(l+1)}(qx)$. By induction, the claim holds for all $l \geq 2$. $\square$

Next, we show that when the data are ordered and $x'_l = x_l$, the features evaluated at training data measure distances between them.

Lemma I.2. Suppose $x_1 \geq \dots \geq x_N$. For $l \geq 2$ for all $n \geq l-1$, $f^{(l)}(x_n) = x_{l-1} - x_n$.

Proof of Lemma I.2. By induction on l . Recall that $x_1 > \dots > x_N$.

Base case: For $l = 2$, for all $n \geq l - 1 = 1$, $f^{(2)}(x_n) = |x_n - x_1| = x_1 - x_n$.

Now let $l \geq 2$ and suppose the claim holds for $f^{(l)}$. Then for all $n \geq (l + 1) - 1 = l$, we have $f^{(l+1)}(x_n) = |f^{(l)}(x_n) - f^{(l)}(x_l)| = |(x_{l-1} - x_n) - (x_{l-1} - x_l)| = |x_l - x_n| = x_l - x_n$. By induction, the claim holds for all $l \geq 2$. $\square$

The next lemma shows that scaling down an l -layer feature by q gives the $(l + 1)$ -layer feature, at the points x_l and x_{l+1} , respectively.

Lemma I.3. *Suppose the data is r -adic. For all $l \geq 2$, $qf^{(l+1)}(x_{l+1}) = f^{(l)}(x_l)$.*

Proof of Lemma I.3. By Lemma I.2 and x_n being r -adic, we have $qf^{(l+1)}(x_{l+1}) = q(x_l - x_{l+1}) = qr^l = r^{l-1} = (x_{l-1} - x_l) = f^{(l)}(x_l)$. $\square$

We extend the previous Lemma I.1 and Lemma I.3 to show that for an l -layer feature, scaling down its output by q and scaling up its argument by q gives the $(l + 1)$ -layer feature, for all $x \in [0, r]$.

Lemma I.4. *Suppose the data is triadic. Then $f^{(l)}(qx) = qf^{(l+1)}(x)$ for $x \in [0, \frac{1}{q}]$.*

Proof of Lemma I.4. We know $qf^{(l+1)}(x) = \tilde{f}^{(l+1)}(qx)$. Let $z = qx$. It suffices to show $f^{(l)}(z) = \tilde{f}^{(l+1)}(z)$. We show this by induction on l .

Base case: for $l = 2$, we have $\tilde{f}^{(l+1)}(z) = \tilde{f}^{(3)}(z) = |\tilde{f}^{(2)}(z) - \tilde{f}^{(2)}(qx_2)| = ||z - qx_1| - |qx_2 - qx_1|| = |qx_1 - z - q(x_1 - x_2)| = |z - qx_2| = |z - x_1| = f^{(2)}(z)$, where we used that $qx_1 = \frac{q}{2} \geq \frac{3}{2} \geq 1 \geq z$ and $x_1 \geq x_2$ as well as $qx_2 = q(x_1 - r) = qx_1 - 1 = \frac{q}{2} - 1$, which is equal to $x_1 = \frac{1}{2}$ for $q = 3$.

Now let $l \geq 2$. Suppose $f^{(l)}(z) = \tilde{f}^{(l+1)}(z)$. Then $\tilde{f}^{(l+2)}(z) = |\tilde{f}^{(l+1)}(z) - \tilde{f}^{(l+1)}(qx_{l+1})| = |f^{(l)}(z) - qf^{(l+1)}(x_{l+1})| = |f^{(l)}(z) - f^{(l)}(x_l)| = f^{(l+1)}(z)$ where we used Lemma I.3 for the second to last equality. By induction the claim holds for all $l \geq 2$. $\square$

Let $\mathcal{G}^{(l)}|_{[a,b]} = \{(x, y) \in \mathcal{G}^{(l)} : x \in [a, b]\}$ be the restriction of the graph $\mathcal{G}^{(l)}$ to domain $[a, b]$. The next lemma states that there is a self-similarity between the graphs of a layer- l feature and a layer $l + 1$ -feature restricted to $[0, r]$. This establishes a Cantor-like self-similarity for the first third of the interval $[0, 1]$.

Lemma I.5. *Suppose the data is triadic. Then $(x, y) \in \mathcal{G}^{(l)}$ if and only if $(rx, ry) \in \mathcal{G}^{(l+1)}|_{[0,r]}$.*

Proof of Lemma I.5. By Lemma I.4, $f^{(l)}(qx) = qf^{(l+1)}(x)$ for $x \in [0, r]$, ie $rf^{(l)}(x) = f^{(l+1)}(rx)$.

Therefore $(rx, ry) \in \mathcal{G}^{(l+1)}|_{[0,r]}$ iff $rx \in [0, r]$ and $ry = f^{(l+1)}(rx)$, iff $x \in [0, 1]$ and $ry = rf^{(l)}(x)$, iff $x \in [0, 1]$ and $y = f^{(l)}(x)$, iff $(x, y) \in \mathcal{G}^{(l)}$. $\square$

The next lemma states that the features are symmetric about x_1 , which will be used in extending self-similarity from the first third of $[0, 1]$ to the second third.

Lemma I.6. *Suppose the data is 1-D and $b_1 = -x_1$. For all $l \geq 2$, $f^{(l)}(x) = f^{(l)}(1 - x)$.*

Proof. By induction on l .

Base case: for $l = 2$, $f^{(2)}(x) = |x - x_1| = |x_1 - x| = |(2x_1 - x) - x_1| = f^{(2)}(2x_1 - x) = f^{(2)}(1 - x)$.

Now, let $l \geq 2$. Suppose $f^{(l)}(x) = f^{(l)}(1 - x)$. Then $f^{(l+1)}(x) = |f^{(l)}(x) - f^{(l)}(x_l)| = |f^{(l)}(1 - x) - f^{(l)}(x_l)| = f^{(l+1)}(1 - x)$. By induction, the claim holds for all $l \geq 2$. $\square$

The next lemma shows that self-similarities of the feature graph on the first third of $[0, 1]$ can be reflected to yield self-similarities on the last third.

Lemma I.7. *Suppose the data is triadic. Then $(x, y) \in \mathcal{G}^{(l)}$ if and only if $(1 - rx, ry) \in \mathcal{G}^{(l+1)}|_{[1-r, 1]}$.*

Proof of Lemma I.7. By Lemma I.4, $rf^{(l)}(x') = f^{(l+1)}(rx')$. And by Lemma I.6, $f^{(l+1)}(1 - rx) = f^{(l+1)}(rx)$.

We have $(1 - rx, ry) \in \mathcal{G}^{(l+1)}|_{[1-r, 1]}$ iff $1 - rx \in [1 - r, r]$ and $ry = f^{(l+1)}(1 - rx)$, iff $x \in [0, 1]$ and $ry = f^{(l+1)}(rx)$, iff $x \in [0, 1]$ and $y = f^{(l+1)}(x)$, iff $(x, y) \in \mathcal{G}^{(l)}$. $\square$

The next lemma expresses the reflective nature of the features in a way that matches the Cantor set formulation.

Lemma I.8. *Suppose the data is triadic. Then $(x, y) \in \mathcal{G}^{(l)}$ if and only if $(1 - (1 - x)r, ry) \in \mathcal{G}^{(l+1)}|_{[1-r, 1]}$.*

Proof of Lemma I.8. By Lemma I.4, $rf^{(l)}(x') = f^{(l+1)}(rx')$. And by Lemma I.6, $f^{(l+1)}(1 - (1 - x)r) = f^{(l+1)}((1 - x)r)$

We have $(1 - r(1 - x), ry) \in \mathcal{G}^{(l+1)}|_{[1-r, 1]}$ iff $1 - r(1 - x) \in [1 - r, r]$ and $ry = f^{(l+1)}(1 - r(1 - x))$, iff $x \in [0, 1]$ and $ry = f^{(l+1)}(r(1 - x))$, iff $x \in [0, 1]$ and $y = f^{(l+1)}(1 - x)$, iff $x \in [0, 1]$ and $y = f^{(l+1)}(x)$, iff $(x, y) \in \mathcal{G}^{(l)}$. $\square$

The next two lemmas describes the feature between the first and last third of $[0, 1]$.

Lemma I.9. *Suppose the data is triadic. For $l \geq 3$ and $x \in [r, \frac{1}{2}]$, $f^{(l)}(x) = x - \left(r - \frac{r^{l-2}}{2}\right)$.*

Proof. By induction on l . Let $x \in [r, \frac{1}{2}]$. Base case: for $l = 3$ and $r = \frac{1}{3}$, $f^{(3)}(x) = ||x - x_1| - |x_2 - x_1|| = |(x_1 - x) - r| = x_1 - x - r = x - (r)$ using that $x \leq \frac{1}{2} = x_1$ and $r \leq x \leq x_1 = \frac{1}{2}$. Now let $l \geq 3$. Suppose $f^{(l)}(x) = x - \left(r - \frac{r^{l-2}}{2}\right)$. By Lemma I.2, $f^{(l)}(x_l) = x_{l-1} - x_l = r^{l-1}$. So $f^{(l+1)}(x) = |f^{(l)}(x) - f^{(l)}(x_l)| = \left|x - \left(r - \frac{r^{l-2}}{2}\right) - r^{l-1}\right| = \left|x - \left(r - r^{(l-1)}\left(\frac{r^{-1}}{2} - 1\right)\right)\right|$. Now, since $r = \frac{1}{3}$ we have $r^{(l-1)}\left(\frac{r^{-1}}{2} - 1\right) = r^{(l-1)}\left(\frac{3}{2} - 1\right) = \frac{r^{l-1}}{2}$, so $f^{(l+1)}(x) = \left|x - \frac{r^{(l+1)-2}}{2}\right|$. By induction the claim holds for all $l \geq 3$. $\square$

Lemma I.10. *Suppose the data is triadic. For $l \geq 3$ and $x \in [\frac{1}{2}, 1 - r]$, $f^{(l)}(x) = \left(1 - r + \frac{r^{l-2}}{2}\right) - x$.*

Proof. Apply Lemma I.9 for $x' = 1 - x$. $\square$

Proof of Theorem V.1. By Lemma I.5, $T_L(\mathcal{G}^{(l)}) = \mathcal{G}^{(l+1)}|_{[0, r]}$. Note $T_R(x) = \frac{2+x}{3} = 1 - \frac{1-x}{3} = 1 - (1-x)r$. By Lemma I.8, $T_R(\mathcal{G}^{(l)}) = \mathcal{G}^{(l+1)}|_{[1-r, 1]}$. By Lemma I.9 and Lemma I.10, $T^{(l)}(\mathcal{G}^{(l)}) = \mathcal{G}^{(l+1)}|_{[r, 1-r]}$. $\square$

ACKNOWLEDGMENT

ChatGPT was used for Python starter code to help produce the following figures: Figure 6 and Figure 8 in Section VI-C; Figure 7, Figure 8, Figure 9 in the Supplementary Material.

REFERENCES

- [1] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0893608089900208>
- [2] R. Tibshirani, “Regression shrinkage and selection via the Lasso,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.
- [3] R. J. Tibshirani, “The lasso problem and uniqueness,” *Electronic Journal of Statistics*, vol. 7, pp. 1456–1490, 2013.
- [4] M. R. Osborne, B. Presnell, and B. A. Turlach, “A new approach to variable selection in least squares problems,” *IMA journal of numerical analysis*, vol. 20, no. 3, pp. 389–403, 2000.
- [5] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, “Least angle regression,” *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.
- [6] F. Bach, “Breaking the curse of dimensionality with convex neural networks,” *JMLR*, vol. 18, no. 1, pp. 629–681, 2017.
- [7] E. Zeger, Y. Wang, T. E. Aaron Mishkin, E. Candès, and M. Pilanci, “A library of mirrors: Deep neural nets in low dimensions are convex lasso models with reflection features,” *arXiv preprint:2403.01046*, 2024.
- [8] M. Pilanci, “From complexity to clarity: Analytical expressions of deep neural network weights via clifford algebra and convexity,” *Transactions on Machine Learning Research*, 2024. [Online]. Available: <https://openreview.net/forum?id=6XPwSwEWsV>
- [9] 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.
- [10] B. Bartan and M. Pilanci, “Neural spectrahedra and semidefinite lifts: global convex optimization of degree-two polynomial activation neural networks in polynomial-time,” *Math. Program.*, vol. 213, no. 1–2, p. 737–769, Nov. 2024. [Online]. Available: <https://doi.org/10.1007/s10107-024-02153-5>
- [11] T. Ergen and M. Pilanci, “Path regularization: A convexity and sparsity inducing regularization for parallel relu networks,” in *NeurIPS*, 2023.
- [12] C. Fang, Y. Gu, W. Zhang, and T. Zhang, “Convex formulation of overparameterized deep neural networks,” *arXiv:1911.07626*, 2019.
- [13] 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, 2020, pp. 7695–7705.
- [14] J. Brehmer, P. D. Haan, S. Behrends, and T. Cohen, “Geometric algebra transformer,” in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. [Online]. Available: <https://openreview.net/forum?id=M7r2CO4tJC>
- [15] D. Ruhe, J. K. Gupta, S. De Keninck, M. Welling, and J. Brandstetter, “Geometric clifford algebra networks,” in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML’23. JMLR.org, 2023.
- [16] P. Yin, E. Esser, and J. Xin, “Ratio and difference of l_1 and l_2 norms and sparse representation with coherent dictionaries,” 2014. [Online]. Available: <https://api.semanticscholar.org/CorpusID:62886972>
- [17] Y. Xu, A. Narayan, H. Tran, and C. G. Webster, “Analysis of the ratio of l_1 and l_2 norms in compressed sensing,” *Applied and Computational Harmonic Analysis*, vol. 55, 2021.
- [18] B. Woodworth, S. Gunasekar, J. D. Lee, E. Moroshko, P. Savarese, I. Golan, D. Soudry, and N. Srebro, “Kernel and rich regimes in overparametrized models,” in *Conference on Learning Theory*. PMLR, 2020, pp. 3635–3673.
- [19] E. Moroshko, B. E. Woodworth, S. Gunasekar, J. D. Lee, N. Srebro, and D. Soudry, “Implicit bias in deep linear classification: Initialization scale vs training accuracy,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 182–22 193, 2020.
- [20] H. Labarrière, C. Molinari, L. Rosasco, C. Vega, and S. Villa, “Optimization insights into deep diagonal linear networks,” *arXiv preprint*, 2024, arXiv:2412.16765.
- [21] M. S. Nacson, K. Ravichandran, N. Srebro, and D. Soudry, “Implicit bias of the step size in linear diagonal neural networks,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 16 270–16 295.
- [22] M. Even, S. Pesme, S. Gunasekar, and N. Flammarion, “(s)gd over diagonal linear networks: Implicit bias, large stepsizes and edge of stability,” in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 29 406–29 448.
- [23] H. Papazov, S. Pesme, and N. Flammarion, “Leveraging continuous time to understand momentum when training diagonal linear networks,” in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2024, pp. 3556–3564.
- [24] S. Pesme, L. Pillaud-Vivien, and N. Flammarion, “Implicit bias of sgd for diagonal linear networks: A provable benefit of stochasticity,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 218–29 230, 2021.

- [25] M. Telgarsky, “Benefits of depth in neural networks,” *JMLR: Workshop and Conference Proceedings*, vol. 49, pp. 1–23, 2016. [Online]. Available: <https://arxiv.org/abs/1602.04485>
- [26] R. Eldan and O. Shamir, “The power of depth for feedforward neural networks,” *In Conference on Learning Theory, 2016.*, 2016. [Online]. Available: <https://arxiv.org/pdf/1512.03965>
- [27] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, “Why and when can deep – but not shallow – networks avoid the curse of dimensionality: a review,” 2016, cBMM Memo No. 058, February 7, 2017. [Online]. Available: <https://arxiv.org/abs/1611.00740>
- [28] H. W. Lin, M. Tegmark, and D. Rolnick, “Why does deep and cheap learning work so well?” *Journal of Statistical Physics*, vol. 168, no. 6, pp. 1223–1247, 2017. [Online]. Available: <https://link.springer.com/article/10.1007/s10955-017-1836-5>
- [29] G. Bresler and D. Nagaraj, “Sharp representation theorems for relu networks with precise dependence on depth,” in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020.
- [30] E. Malach and S. Shalev-Shwartz, “Is deeper better only when shallow is good?” *NeurIPS*, vol. abs/1903.03488, 2019. [Online]. Available: <https://api.semanticscholar.org/CorpusID:72940881>
- [31] T. Mikolov, W. tau Yih, and G. Zweig, “Linguistic regularities in continuous space word representations,” in *North American Chapter of the Association for Computational Linguistics*, 2013. [Online]. Available: <https://api.semanticscholar.org/CorpusID:7478738>
- [32] K. Park, Y. J. Choe, and V. Veitch, “The linear representation hypothesis and the geometry of large language models,” *ArXiv*, vol. abs/2311.03658, 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:265042984>
- [33] K. Park, Y. J. Choe, Y. Jiang, and V. Veitch, “The geometry of categorical and hierarchical concepts in large language models,” *ArXiv*, vol. abs/2406.01506, 2024. [Online]. Available: <https://api.semanticscholar.org/CorpusID:270216615>
- [34] M. Berger, *Geometry I*. Springer Science & Business Media, 2009.
- [35] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond empirical risk minimization,” in *International Conference on Learning Representations*, 2018. [Online]. Available: <https://openreview.net/forum?id=r1Ddp1-Rb>
- [36] 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.
- [37] Y. Wang, S. Kim, P. Chu, I. Subramaniam, and M. Pilanci, “Randomized geometric algebra methods for convex neural networks,” *ArXiv*, vol. abs/2406.02806, 2024. [Online]. Available: <https://api.semanticscholar.org/CorpusID:270258553>
- [38] H. J. S. Smith, “On the integration of discontinuous functions,” *Proceedings of the London Mathematical Society*, vol. 6, pp. 140–153, 1874.
- [39] G. Cantor, “Über unendliche, lineare punktmannigfaltigkeiten v [on infinite, linear point-manifolds (sets), part 5],” *Mathematische Annalen*, vol. 21, pp. 545–591, 1883, archived from the original on 2015-09-24. Retrieved 2011-01-10.
- [40] J. Lévy-Véhel, E. Lutton, and C. Tricot, *Fractals in engineering*. Springer, 2005.
- [41] D. Saupe and R. Hamzaoui, “A review of the fractal image compression literature,” *SIGGRAPH Comput. Graph.*, vol. 28, no. 4, p. 268–276, Nov. 1994. [Online]. Available: <https://doi.org/10.1145/193234.193246>
- [42] H. Diab and N. Abboud, “The engineering applications of fractals,” *SIMULATION*, vol. 57, no. 2, pp. 89–97, 1991.
- [43] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning word vectors for sentiment analysis,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. [Online]. Available: <http://www.aclweb.org/anthology/P11-1015>
- [44] A. M. Turing, “The chemical basis of morphogenesis,” *Philosophical Transactions of the Royal Society of London B*, vol. 237, no. 641, pp. 37–72, 1952.
- [45] S. Kondo and T. Miura, “Reaction-diffusion model as a framework for understanding biological pattern formation,” *Science*, vol. 329, no. 5999, pp. 1616–1620, 2010. [Online]. Available: <https://www.science.org/doi/abs/10.1126/science.1179047>
- [46] T. E. Yifei Wang and M. Pilanci, “Parallel deep neural networks have zero duality gap,” *ICLR*, 2023s.
- [47] T. Szandała, *Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks*, 01 2021, pp. 203–224.
- [48] O. I. Berngardt, “Improving classification neural networks by using absolute activation function (mnist/lenet-5 example),” *arXiv preprint:2304.11758*, 2023.
- [49] I. Vallés-Pérez, E. Soria-Olivas, M. Martínez-Sober, A. J. Serrano-López, J. Vila-Francés, and J. Gómez-Sanchís, “Empirical study of the modulus as activation function in computer vision applications,” *Engineering Applications of Artificial Intelligence*, vol. 120, 2023.
- [50] C. Perwass, H. Edelsbrunner, L. Kobbelt, and K. Polthier, *Geometric algebra with applications in engineering*. Springer, 2009, vol. 4.
- [51] E. Chisolm, “Geometric algebra,” *arXiv:1205.5935*, 05 2012.
- [52] C. Doran and A. Lasenby, *Geometric algebra for physicists*. Cambridge University Press, 2003.
- [53] D. Hestenes, “Spacetime physics with geometric algebra,” *American Journal of Physics*, vol. 71, no. 7, pp. 691–714, 2003.

- [54] J. Vince, *Geometric algebra for computer graphics*. Springer Science & Business Media, 2008.
- [55] A. Mishkin and M. Pilanci, “Optimal sets and solution paths of ReLU networks,” in *International Conference on Machine Learning, ICML 2023*. PMLR, 2023.
- [56] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, “GLUE: A multi-task benchmark and analysis platform for natural language understanding,” 2019, in the Proceedings of ICLR.
- [57] A. Warstadt, A. Singh, and S. R. Bowman, “Neural network acceptability judgments,” *arXiv preprint 1805.12471*, 2018.
- [58] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proceedings of EMNLP*, 2013, pp. 1631–1642.
- [59] W. B. Dolan and C. Brockett, “Automatically constructing a corpus of sentential paraphrases,” in *Proceedings of the International Workshop on Paraphrasing*, 2005.
- [60] E. Agirre, L. M^arquez, and R. Wicentowski, Eds., *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*. Prague, Czech Republic: Association for Computational Linguistics, June 2007.
- [61] A. Williams, N. Nangia, and S. R. Bowman, “A broad-coverage challenge corpus for sentence understanding through inference,” in *Proceedings of NAACL-HLT*, 2018.
- [62] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “SQuAD: 100,000+ questions for machine comprehension of text,” in *Proceedings of EMNLP*. Association for Computational Linguistics, 2016, pp. 2383–2392.
- [63] I. Dagan, O. Glickman, and B. Magnini, “The PASCAL recognising textual entailment challenge,” in *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising textual entailment*. Springer, 2006, pp. 177–190.
- [64] R. Bar Haim, I. Dagan, B. Dolan, L. Ferro, D. Giampiccolo, B. Magnini, and I. Szpektor, “The second PASCAL recognising textual entailment challenge,” 2006.
- [65] D. Giampiccolo, B. Magnini, I. Dagan, and B. Dolan, “The third PASCAL recognizing textual entailment challenge,” in *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*. Association for Computational Linguistics, 2007, pp. 1–9.
- [66] H. J. Levesque, E. Davis, and L. Morgenstern, “The Winograd schema challenge,” in *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, vol. 46, 2011, p. 47.
- [67] L. Bentivogli, I. Dagan, H. T. Dang, D. Giampiccolo, and B. Magnini, “The fifth PASCAL recognizing textual entailment challenge,” 2009.
- [68] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2019. [Online]. Available: <https://api.semanticscholar.org/CorpusID:52967399>
- [69] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.