Journal of Artificial Intelligence Research 74 (2022) 1775-1790 Submitted 12/2021; published 08/2022

The Computational Complexity of ReLU Network Training Parameterized by Data Dimensionality

Vincent Froese VINCENT.FROESE@TU-BERLIN.DE
Technische Universität Berlin,
Algorithmics and Computational Complexity,
Ernst-Reuter-Platz 7,
D-10587 Berlin, Germany

Christoph Hertrich C.HERTRICH@LSE.AC.UK
London School of Economics and Political Science,
Department of Mathematics,
Houghton Street,
London WC2A 2AE, United Kingdom

Rolf Niedermeier ROLF.NIEDERMEIER@TU-BERLIN.DE
Technische Universität Berlin,
Algorithmics and Computational Complexity,
Ernst-Reuter-Platz 7,
D-10587 Berlin, Germany

Abstract

Understanding the computational complexity of training simple neural networks with rectified linear units (ReLUs) has recently been a subject of intensive research. Closing gaps and complementing results from the literature, we present several results on the parameterized complexity of training two-layer ReLU networks with respect to various loss functions. After a brief discussion of other parameters, we focus on analyzing the influence of the dimension d of the training data on the computational complexity. We provide running time lower bounds in terms of W[1]-hardness for parameter d and prove that known brute-force strategies are essentially optimal (assuming the Exponential Time Hypothesis). In comparison with previous work, our results hold for a broad(er) range of loss functions, including ℓ^p-loss for all $p \in [0, \infty]$. In particular, we improve a known polynomial-time algorithm for constant d and convex loss functions to a more general class of loss functions, matching our running time lower bounds also in these cases.

1. Introduction

Dimensionality reduction of data is a central issue in many machine learning scenarios (Bartal, Fandina, & Neiman, 2019; van der Maaten, Postma, & van der Herik, 2009). In this paper, our focus is on addressing a natural follow-up question: To what extent can "low-dimensionality" of data points help in lowering the worst-case computational complexity of the task of training neural networks? This question is particularly relevant from a practical point of view because real-life data, even if high-dimensional, is often assumed to be contained in a low-dimensional submanifold of the input space. To answer this question, we will employ tools and concepts from parameterized complexity analysis. Doing so, we focus on the very basic case of two-layer ReLU (rectified linear units) neural network training.

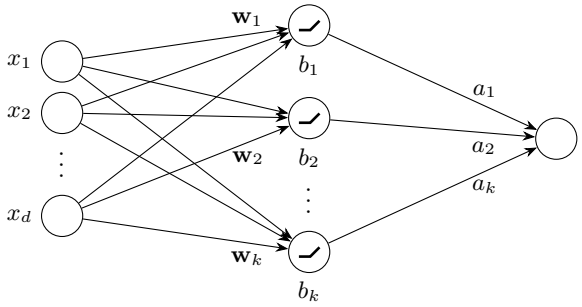

Figure 1: The neural network architecture we study in this paper: After the input layer (left) with d input neurons, we have one hidden layer with k ReLU neurons and a single output neuron without additional activation function.

We believe that the studied problem, though very simple and practically irrelevant, is a basic building block in the grand challenge of gaining a fundamental understanding of the power and limitations of ReLU networks. As already pointed out in the literature (Bakshi, Jayaram, & Woodruff, 2019; Goel, Klivans, Manurangsi, & Reichman, 2021), understanding shallow networks is a natural first step which is already quite involved and challenging. To the best of our knowledge, we are the first to apply principles from parameterized complexity theory to empirical risk minimization of ReLU networks. Before proceeding with a discussion of related work and our new contributions, we first provide some formal definitions concerning the problems which are central to our work.

We study empirical risk minimization for neural networks with ReLU activation. A *rectifier* is the function $[x]_+ \coloneqq \max(0, x)$. Given a loss function $\ell\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$, mapping the predicted and the true label to a loss value, the problem of training a two-layer ReLU neural network with k hidden neurons and a single output neuron (see Figure 1) is defined as follows:

k-ReLU(ℓ)

Input: Data points $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$.

Task: Find weight vectors $\mathbf{w}_1, \dots, \mathbf{w}_k \in \mathbb{R}^d$, biases $b_1, \dots, b_k \in \mathbb{R}$, and coefficients $a_1, \dots, a_k \in \{-1, 1\}$ that minimize

$$\sum_{i=1}^{n} \ell\left(\sum_{j=1}^{k} a_j [\langle \mathbf{w}_j, \mathbf{x}_i \rangle + b_j]_+,\ y_i \right).$$

As usual in the context of computational complexity analysis, we consider the decision instead of the optimization version. The corresponding decision problem is to decide whether a target training error of at most $\gamma \in \mathbb{R}$ can be achieved. We believe it is of fundamental interest to know where the borderlines of exact solvability of ReLU training are. One of our main contributions is to prove strong computational hardness results that already hold for only a single hidden ReLU neuron, that is, $k = 1$.

In this work, we focus on the case of ℓ^p-loss functions, that is, $\ell(\hat{y}, y) = |\hat{y} - y|^p$, for $p \in]0, \infty[$. Note that this includes both convex and concave loss functions (with respect

to the absolute error $|\hat{y} - y|$). In addition, we also investigate the limit cases $p = 0$ and $p = \infty$. The ℓ^0-loss (which is completely insensitive to outliers) simply counts the number of points that are not perfectly fitted, while the ℓ^∞-loss only cares about outliers, that is, it measures only the largest error on any data point. The parameter p can be used to smoothly interpolate between these two extreme options in practice. Notably, concave loss functions ($p < 1$) are explicitly used to obtain very outlier-robust methods (Wang, Nie, Cai, & Huang, 2013; Jiang, Nie, & Huang, 2015; de Araujo, Hirata, & Rakotomamonjy, 2018). See also (Janocha & Czarnecki, 2016) for an analysis of the impact of different loss functions on neural network training.

Without loss of generality, we always assume that $n \geq d + 1$ because otherwise the problem can be solved in the lower-dimensional affine hull of the input data points.[1]

The core theme of our work is to better understand how the dimension parameter d influences the computational complexity of ReLU network training as defined above. To this end, we conduct a parameterized complexity analysis (Downey & Fellows, 2013). Before moving on to study the key parameter d, let us briefly discuss other parameters occurring in our setting. The most natural other parameters are: the number k of ReLU neurons in the hidden layer, the number n of input points, and the maximum target error γ.

It turns out that the parameterized complexity for these three parameters is already settled due to the known literature and simple observations. First, the case $k = 1$ (that is, 1-ReLU(ℓ)) is known to be NP-hard for ℓ^2-loss (Dey, Wang, & Xie, 2020; Goel et al., 2021) and we even extend the NP-hardness to ℓ^p-loss for arbitrary $p \in [0, \infty[$ (see Theorem 1); this renders the parameter k hopeless in terms of getting efficient parameterized algorithms. For the parameter n, fixed-parameter tractability was already observed by Goel et al. (2021) (see also related work). Finally, for $\gamma = 0$, the case $k = 1$ is polynomial-time solvable (Goel et al., 2021) and for $k \geq 2$ NP-hardness is known (Bakshi et al., 2019; Goel et al., 2021). Thus, the dimension d clearly is the most interesting parameter also from a theoretical point of view. We close some knowledge gaps from the literature with respect to d by proving strong computational hardness results as well as matching upper bounds. Before discussing our results in more detail, let us first review the closely related literature.

Related Work. The NP-hardness of empirical risk minimization with ℓ^2-loss for a single ReLU was shown independently by Dey et al. (2020) and Goel et al. (2021). The work of Goel et al. (2021) is probably the one closest to our work. They provided an in-depth study concerning tight hardness results for depth-2 ReLU networks such as NP-hardness, conditional running time lower bounds, and hardness of approximation. Arora, Basu, Mianjy, and Mukherjee (2018) provided a polynomial-time algorithm for k-ReLU(ℓ) for $d \in O(1)$ and convex loss ℓ; in terms of parameterized algorithmics, this is an XP-algorithm for parameter d: the degree of the polynomial of the running time (only) depends on d. The underlying idea of their algorithm is to basically iterate over all $O(n^d)$ hyperplane partitions of the n data points. Indeed, as pointed out by Goel et al. (2021), since there are at most 2^n partitions, the same algorithm implies fixed-parameter tractability for the pa-

1. To see this, assume that all $\mathbf{x}_i$ are contained in an affine subspace A with dimension strictly less than d. Using a bijective affine transformation $T\colon A \to \mathbb{R}^{\dim A}$, we now can solve the equivalent k-ReLU(ℓ) problem with the lower-dimensional data points $(T(x_i), y_i)$, $i \in [n]$. The solution weights for the original problem can then be obtained by composing T with the affine transformation given by the weights of the modified problem.

rameter n. Moreover, Goel et al. (2021) remarked that the well-known Exponential Time Hypothesis (ETH) implies that no $2^{o(n)}$-time algorithm exists. Deciding whether zero error ($\gamma = 0$) is possible (that is, *realizable* data) is polynomial-time solvable for a single ReLU by linear programming (Goel et al., 2021) and NP-hard for two ReLUs (Goel et al., 2021). Approximation has been subject to further works (Dey et al., 2020; Diakonikolas, Goel, Karmalkar, Klivans, & Soltanolkotabi, 2020; Goel et al., 2021). Furthermore, Bakshi et al. (2019) and Chen, Klivans, and Meka (2021) showed fixed-parameter tractability results for related but different learning concepts of ReLU networks and Boob, Dey, and Lan (2022) studied the computational complexity of ReLU networks where the output neuron is also a ReLU. Pilanci and Ergen (2020) show that training a 2-layer neural network can be reformulated as a convex program. However, the implications on the computational complexity are limited since their result requires the number of hidden neurons to be very large. Bertschinger, Hertrich, Jungeblut, Miltzow, and Weber (2022) show that training 2-layer neural networks is complete for the complexity class $\exists\mathbb{R}$ (existential theory of the reals), implying that the problem is presumably not contained in NP. They generalize a previous result by Abrahamsen, Kleist, and Miltzow (2021), who showed the same fact for specifically designed, more complex architectures.

Finally, note that the number of dimensions appears naturally in parameterized complexity studies for geometric problems (Giannopoulos, Knauer, & Rote, 2009; Knauer, König, & Werner, 2015); moreover, it occurs also in recent studies for principal component analysis (PCA) (Fomin, Golovach, & Simonov, 2020; Simonov, Fomin, Golovach, & Panolan, 2019) and in computer vision (Chin, Cai, & Neumann, 2020).

Our Contributions. Essentially focusing on the influence of the dimension parameter d (which so far has been neglected in the literature), we provide two main contributions in terms of worst-case complexity analysis: First, training a two-layer ReLU neural network is already computationally intractable even for a single hidden neuron and small d (Theorem 1), that is, we show W[1]-hardness for parameter d and provide an ETH-based lower bound of $n^{\Omega(d)}$ matching the running time upper bound of $n^{O(d)}$ of the brute-force algorithm due to Arora et al. (2018). Hence, our result shows that the combinatorial search among all $O(n^d)$ possible hyperplane partitions is essentially the best one can do. Notably, our hardness results even hold for very sparse data points with binary labels. It is particularly surprising that (parameterized) hardness already appears in the case of a single ReLU neuron because this model is almost a linear model. Linear models (like support vector machines) can be easily trained in polynomial time. Hence, the presence of a single nonlinearity, in the form of a single hyperplane of breakpoints, increases the computational complexity a lot. This also indicates that learning more complicated network architectures is expected to be even more difficult because the set of representable piecewise linear functions becomes much more complex (Hertrich, Basu, Di Summa, & Skutella, 2021).

Second, on the positive side, for any $k \geq 1$, we extend the XP-result for convex loss functions by Arora et al. (2018) to concave loss functions (Theorem 4). Note that for W[1]-hard problems, an XP-algorithm is the best one can hope for. Hence, we completely settle the computational complexity parameterized by dimension d of training a two-layer ReLU neural network for any ℓ^p-loss with $p \in [0, \infty[$.

Table 1: (Parameterized) computational complexity of training a single ReLU neuron with respect to parameter d (input dimension) for ℓ^p-loss functions.

	Hardness	Algorithm
$p \in [0, 1[$	W[1]-h. + no $n^{o(d)}$-time alg. (Theorem 1)	$n^{O(d)}\,\mathrm{poly}(n, d)$ (Theorem 4)
$p \in [1, \infty[$	W[1]-h. + no $n^{o(d)}$-time alg. (Theorem 1)	$n^{d}\,\mathrm{poly}(n, d)$ (Arora et al., 2018)
$p = \infty$	-	$\mathrm{poly}(n, d)$ (Proposition 3)

Besides these two main findings filling gaps in the literature, we also contribute a polynomial-time algorithm (for arbitrary dimension) for training a single-neuron ReLU network when using the ℓ^∞-loss function (Proposition 3). This generalizes the polynomial-time result due to Goel et al. (2021) for the zero-error case. Table 1 provides an overview of our results for the special case of a single hidden neuron ($k = 1$).

While hardness of approximation was already shown by Goel et al. (2021), we complement this by parameterized hardness of exact solutions. Thus, our work points the way ahead towards a need of combining both algorithmic approaches and to look for approximation algorithms in FPT time.

Parameterized Complexity. We assume the reader to be familiar with basic concepts of computational complexity theory. Parameterized complexity is a multivariate approach to study the time complexity of computational problems (Downey & Fellows, 2013; Cygan, Fomin, Kowalik, Lokshtanov, Marx, Pilipczuk, Pilipczuk, & Saurabh, 2015). An instance (x, k) of a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ consists of a classical problem instance $x \in \Sigma^*$ and a *parameter* value $k \in \mathbb{N}$. A parameterized problem is *fixed-parameter tractable (fpt)* (contained in the class FPT) if there exists an algorithm solving any instance (x, k) in $f(k) \cdot |x|^{O(1)}$ time, where f is a function solely depending on k. Note that fixed-parameter tractability implies polynomial time for constant parameter values where, importantly, the degree of the polynomial is independent from the parameter value. The class W[1] contains parameterized problems which are presumably not in FPT (e.g. Clique parameterized by the size of the requested clique). Parameterized intractability can be shown in terms of W[1]-hardness which is defined via *parameterized reductions*. A parameterized reduction from L to L' is an algorithm that maps an instance (x, k) in $f(k) \cdot |x|^{O(1)}$ time to an instance (x', k') such that $k' \leq g(k)$ for some function g and $(x, k) \in L$ if and only if $(x', k') \in L'$. The class XP contains all parameterized problems which can be solved in polynomial time if the parameter is a constant, that is, in time $f(k) \cdot |x|^{g(k)}$. It is known that FPT $\subseteq$ W[1] $\subseteq$ XP and that FPT $\subsetneq$ XP.

Exponential Time Hypothesis. The Exponential Time Hypothesis (ETH) (Impagliazzo & Paturi, 2001) states that 3-SAT cannot be solved in subexponential time in the number n of variables of the Boolean input formula. That is, there exists a constant $c > 0$ such that 3-SAT cannot be solved in $O(2^{cn})$ time. The ETH implies that FPT $\neq$ W[1] (and hence P $\neq$ NP) (Cygan et al., 2015). It also implies running time lower bounds, for example, that Clique cannot be solved in $\rho(k) \cdot n^{o(k)}$ time for any function ρ, where k is the size of the sought clique and n is the number of graph vertices (Cygan et al., 2015).

Notation. For $n \in \mathbb{N}$, let $[n] := \{1, \dots, n\}$.

2. Hardness of Training a Single ReLU with ℓ^p-Loss in Small Dimension

In this section, we show that there is no hope to obtain an FPT algorithm with respect to parameter d for training even the simplest possible architecture consisting of a single neuron with respect to the ℓ^p-loss for any $p \in [0, \infty[$. To this end, we show intractability of the problem 1-ReLU(ℓ^p). For $p = 0$, the problem is to minimize the number of data points that are not perfectly fitted.

Theorem 1. *For $p \in [0, \infty[$, 1-ReLU(ℓ^p) is NP-hard, W[1]-hard with respect to dimension d, and it cannot be solved in $\rho(d) \cdot n^{o(d)}$ time for any computable function ρ unless the Exponential Time Hypothesis fails.*

Note that Goel et al. (2021) proved NP-hardness and conditional running time lower bounds for additive approximation of k-ReLU(ℓ^2) for $k \geq 1$. Their running time lower bound for $k = 1$ is based on a newly introduced assumption of inapproximability of finding dense subgraphs and the lower bound for $k > 1$ assumes the *Gap Exponential Time Hypothesis* (Dinur, 2016) (which implies the ETH). Their results are achieved via gap reductions from the problems of finding dense subgraphs and coloring hypergraphs. The reductions are focused on providing a gap which guarantees the approximation hardness. They achieve this by using a "large" number d of dimensions (typically equal to the number of vertices of the input (hyper)graph). For our purpose, however, we need a more fine-grained parameterized reduction where d is "small". To this end, we reduce from a colored variant of Clique such that d is linearly upper-bounded in the size of the clique.

Proof of Theorem 1. We reduce from the following problem.

Multicolored Clique

Input:	An undirected graph $G = (V, E)$ where the vertices are colored with k colors.
Question:	Does G contain a k-clique (a complete subgraph with k vertices) with exactly one vertex from each color?

Multicolored Clique is NP-hard, W[1]-hard with respect to k, and not solvable in $\rho(k) \cdot |V|^{o(k)}$ time for any computable function ρ unless the Exponential Time Hypothesis fails (Cygan et al., 2015). We give a parameterized reduction (which is also polynomial-time) from Multicolored Clique to 1-ReLU(ℓ^p) where the dimension of the data points is $d = 2k$. Hence, the theorem follows.

Let $G = (V, E)$ be an undirected graph with $N := |V|$ vertices and let $c \colon V \to [k]$ be a vertex coloring. We denote by $V_j = \{v_{j,1}, \dots, v_{j,N_j}\}$ the set of vertices with color j, where $N_j := |V_j|$. In the following, we construct a set of data points from $\mathbb{R}^{2k}$ with labels in $\{0, 1\}$, as well as a target error $\gamma \in \mathbb{R}$, such that these data points can be fitted by a ReLU function with ℓ^p-error at most γ if and only if a multicolored k-clique exists in G.

We set the target error to $\gamma := N - k$. Next, we define a small value $0 < \delta < 1$ such that making an absolute error of value $1 - \delta$ on $N - k + 1$ different input points already exceeds

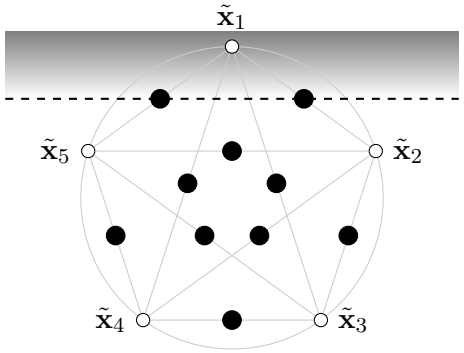

Figure 2: Schematic illustration of the reduction from Multicolored Clique. Shown are two dimensions corresponding to one of the k colors. The white points $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_5$ correspond to vertices of that color and have label 1. Black points indicate M copies of the corresponding middle point with label 0. The dashed line indicates the hyperplane defined by the weight vector $\mathbf{w}$ of the ReLU neuron and the shaded area indicates the predictions of the neuron (darker means larger values). The idea is that exactly one white point can be predicted correctly (which selects the corresponding vertex to be in the clique) without predicting a black point incorrectly and thereby exceeding the error.

the threshold γ. For $p = 0$, we simply choose $\delta \coloneqq 0.5$. For $p > 0$, let $\tilde{p} \coloneqq \max\{p, 1\}$ and $\delta \coloneqq 1/(2\tilde{p}(N - k + 1))$. This yields

$$\begin{aligned}(1-\delta)^p(N-k+1) &\geq (1-\tilde{p}\delta)(N-k+1)\\ &> (1-2\tilde{p}\delta)(N-k+1)\\ &= N-k=\gamma,\end{aligned} \tag{1}$$

where, in the case $p > 1$, the first inequality follows from Bernoulli's inequality, and in the case $p \leq 1$, it follows from $x^p \geq x$ for all $x \in [0, 1]$.

In addition, we define a large integer M such that making an absolute error of δ on M different input points also exceeds the threshold γ. For $p = 0$, we choose $M \coloneqq N - k + 1$. For $p > 0$, we set $M \coloneqq \lceil \gamma/\delta^p \rceil + 1$, which implies

$$M\delta^p > \gamma. \tag{2}$$

Note that $\gamma \in O(N)$ and $1/\delta \in O(N)$. Thus, M is polynomially bounded in the size of G.

Let $N_{\max} \coloneqq \max_{j \in [k]} N_j$ be the maximum number of vertices of one color. For our reduction we need $N_{\max}$ distinct rational points on the unit circle centered at the origin. For example, one can choose

$$\tilde{\mathbf{x}}_i \coloneqq \left(\frac{1-i^2}{1+i^2}, \frac{2i}{1+i^2}\right) \in \mathbb{Q}^2$$

for each $i = 1, 2, \ldots, N_{\max}$ (Silverman & Tate, 1994). For each vertex $v_{j,i} \in V$, $j \in [k]$, $i \in [N_j]$, let $\mathbf{x}_{j,i} = (\mathbf{0}_{2j-2}, \tilde{\mathbf{x}}_i, \mathbf{0}_{2k-2j}) \in \mathbb{R}^{2k}$ be the point $\tilde{\mathbf{x}}_i$ lifted to $2k$ dimensions, where each color corresponds to two dimensions. Here, we use the notation $\mathbf{0}_r$ for the r-dimensional

zero-vector. We add two types of data points to our instance (see Figure 2 for an example). First, for each vertex $v_{j,i} \in V$, add the point $(\mathbf{x}_{j,i}, 1) \in \mathbb{R}^{2k} \times \mathbb{R}$. Second, for each pair of distinct vertices $v_{j,i} \neq v_{j',i'} \in V$, if they cannot both be part of a multicolored clique because they are non-adjacent or have the same color, then add M copies of the point $\big((\mathbf{x}_{j,i} + \mathbf{x}_{j',i'})/2, 0\big) \in \mathbb{R}^{2k} \times \mathbb{R}$. This finishes the construction.

We now show that there is a multicolored clique of size k in G if and only if these data points can be fitted by a ReLU with ℓ^p-error at most γ. This then completes our reduction from Multicolored Clique to 1-ReLU(ℓ^p) and hence implies the theorem.

For the first direction, assume that the vertices $v_{1,i_1}, \dots, v_{k,i_k}$ form a multicolored clique of size k in G. We define $\varepsilon \coloneqq 1 - \max_{i \neq i' \in [N_{\max}]} \langle \tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_{i'} \rangle$. Observe that $\varepsilon > 0$, since the points $\tilde{\mathbf{x}}_i$, $i \in [N_{\max}]$, are distinct points on the unit circle. Let $\mathbf{w} \coloneqq 2/\varepsilon \cdot (\tilde{\mathbf{x}}_{i_1}, \tilde{\mathbf{x}}_{i_2}, \dots, \tilde{\mathbf{x}}_{i_k}) \in \mathbb{R}^{2k}$ and $b \coloneqq 1 - 2/\varepsilon$. We claim that the ReLU function $f(\mathbf{x}) = [\langle \mathbf{w}, \mathbf{x} \rangle + b]_+$ achieves an ℓ^p-error of exactly $\gamma = N - k$. To see this, first note that for each $j \in [k]$, we have

$$\langle \mathbf{w}, \mathbf{x}_{j,i_j} \rangle + b = 2/\varepsilon \cdot \langle \tilde{\mathbf{x}}_{i_j}, \tilde{\mathbf{x}}_{i_j} \rangle + 1 - 2/\varepsilon = 1,$$

where we used that $\tilde{\mathbf{x}}_{i_j}$ lies on the unit circle. Hence, the k points $\mathbf{x}_{1,i_1}, \dots, \mathbf{x}_{k,i_k}$ are perfectly fitted. Second, for each $v_{j,i} \in V \setminus \{v_{1,i_1}, \dots, v_{k,i_k}\}$, we have

$$\langle \mathbf{w}, \mathbf{x}_{j,i} \rangle + b = 2/\varepsilon \cdot \langle \tilde{\mathbf{x}}_{i_j}, \tilde{\mathbf{x}}_i \rangle + 1 - 2/\varepsilon \leq 2/\varepsilon \cdot (1 - \varepsilon) + 1 - 2/\varepsilon = -1,$$

where the inequality follows from our choice of ε. Hence, for each of these $N - k$ points, we have $f(\mathbf{x}_{j,i}) = 0$, that is, we incur an error of 1. Finally, for each pair of distinct vertices $v_{j,i} \neq v_{j',i'} \in V$ that are either non-adjacent or have the same color, note that at most one of the two vertices can belong to the clique. Thus, making use of our two calculations above, we obtain

$$\langle \mathbf{w}, (\mathbf{x}_{j,i} + \mathbf{x}_{j',i'})/2 \rangle + b = ((\langle \mathbf{w}, \mathbf{x}_{j,i} \rangle + b) + (\langle \mathbf{w}, \mathbf{x}_{j',i'} \rangle + b))/2 \leq (1 - 1)/2 = 0.$$

Hence, all points with label 0 are fitted exactly and the total ℓ^p-error is equal to $\gamma = N - k$.

For the reverse direction, suppose that there exist $\mathbf{w} \in \mathbb{R}^{2k}$ and $b \in \mathbb{R}$ such that the ReLU function $f(\mathbf{x}) = [\langle \mathbf{w}, \mathbf{x} \rangle + b]_+$ achieves an ℓ^p-error of at most $\gamma = N - k$. We show that the set

$$C \coloneqq \{v_{j,i} \in V \mid f(\mathbf{x}_{j,i}) > \delta\}$$

forms a multicolored clique in G. First, observe that $|C| \geq k$, because otherwise all data points associated with vertices in $V \setminus C$ would incur a total ℓ^p-error of at least $(1 - \delta)^p (N - k + 1)$, which is larger than γ by (1). Hence, it remains to show for each pair of vertices $v_{j,i} \neq v_{j',i'} \in C$ that they belong to different color classes and are adjacent. Suppose the contrary. Then, by construction, the 1-ReLU(ℓ^p) instance also contains M copies of the point $\big((\mathbf{x}_{j,i} + \mathbf{x}_{j',i'})/2, 0\big) \in \mathbb{R}^{2k} \times \mathbb{R}$. From $\langle \mathbf{w}, \mathbf{x}_{j,i} \rangle + b > \delta$ and $\langle \mathbf{w}, \mathbf{x}_{j',i'} \rangle + b > \delta$, it follows by linearity that

$$f((\mathbf{x}_{j,i} + \mathbf{x}_{j',i'})/2) \geq \langle \mathbf{w}, (\mathbf{x}_{j,i} + \mathbf{x}_{j',i'})/2 \rangle + b > \delta.$$

Thus, we incur an ℓ^p-error of at least $M\delta^p$, which is larger than γ by (2). Hence, C is indeed a multicolored k-clique. □

A closer inspection of the above proof reveals that hardness even holds for a more restricted problem.

Corollary 2. *For $p \in [0, \infty[$, 1-ReLU(ℓ^p) is NP-hard, W[1]-hard with respect to d, and cannot be solved in $\rho(d) \cdot n^{o(d)}$ time for any computable function ρ (assuming the Exponential Time Hypothesis) even if all input data points contain at most four non-zero entries and have binary labels.*

We further remark that the basic idea of the reduction in the proof of Theorem 1 also works for more general loss functions. Essentially, the only necessary condition is that the value M can be chosen such that it is polynomially bounded in the size of the graph G and satisfies an inequality analogous to (2) where $(\cdot)^p$ is replaced by the corresponding loss function. We refrain from giving a precise formalization here. Moreover, it is natural to expect that k-ReLU with $k > 1$ is computationally even more difficult than the one-neuron case. Hence, our hardness results should also hold there as well. Indeed, we expect this to be also true for deeper neural networks; however, a formal proof would require a more profound understanding of the complicated functions expressible with deeper networks and is left for future work.

Our findings tell us that in order to achieve fixed-parameter tractability, one has to consider other parameters to combine with the dimension d. A natural parameter is the target loss γ. However, this is not a promising parameter since it can be made arbitrarily small by scaling all values. If we consider the number σ of different coordinate values of the $\mathbf{x}_i$, then we trivially obtain fixed-parameter tractability in combination with d since the overall number of different data points is at most σ^d. Hence, the algorithm by Arora et al. (2018) runs in $\sigma^{d^2} \cdot \text{poly}(nd)$ time.

To sum up, identifying promising parameters (or parameter combinations) to obtain tractable cases remains a challenge worthwhile further investigation.

3. Polynomial-time Algorithm for a Single ReLU with Maximum Norm

As pointed out by Goel et al. (2021), deciding whether given data points are realizable by a single ReLU neuron (that is, whether $\gamma = 0$) can be done in polynomial time via linear programming. In other words, it is possible to check whether the input points can be perfectly fitted by a single ReLU neuron and, in case of a positive answer, to find the corresponding weights in polynomial time. Recall that the same problem is NP-hard in the case of two (or more) neurons by Goel et al. (2021).

In this section, we extend this result to minimizing the ℓ^∞-loss, that is, minimizing the maximum prediction error. In fact, we provide a polynomial-time optimization algorithm (not only decision) for a problem variant that generalizes ℓ^∞-loss minimization. In this variant, the real labels y_i for the data points $\mathbf{x}_i$ are replaced by target intervals $[\alpha_i, \beta_i]$ with $\alpha_i \leq \beta_i$ and we aim to minimize the maximum deviation of a prediction from its corresponding target interval. To this end, we define $\text{dist}_{\alpha,\beta}(t) \coloneqq \max\{\alpha - t, 0, t - \beta\}$ to be the *distance* of $t \in \mathbb{R}$ to the interval $[\alpha, \beta]$.

ReLU(ℓ^∞-Interval)

Input: Data points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ and interval boundaries $\alpha_1 \leq \beta_1, \ldots, \alpha_n \leq \beta_n \in \mathbb{R}$.

Task: Find $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ that minimize $\max_{i \in [n]} \text{dist}_{\alpha_i,\beta_i}([\langle \mathbf{w}, \mathbf{x}_i \rangle + b]_+)$.

Note that we obtain ℓ^∞-loss minimization by setting $\alpha_i = \beta_i = y_i$ for all $i \in [n]$.

Proposition 3. ReLU(ℓ^∞-Interval) *can be solved in polynomial time.*

Proof. We show that an optimal solution can be found via solving a series of linear programs. For each $i \in [n]$ with $\alpha_i > 0$, our algorithm finds out whether the optimal objective value γ^* is larger or smaller than α_i. In the first case, the prediction $\langle \mathbf{w}, \mathbf{x}_i \rangle + b$ is allowed to be arbitrarily small, while in the second case we need to ensure the lower bound $\langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq \alpha_i - \gamma^*$. Therefore, we implement a binary search to find an interval in which γ^* is contained as follows. Let $\{\tilde{\alpha}_1, \tilde{\alpha}_2, \dots, \tilde{\alpha}_r\}$ be the set of all distinct positive α_i-values, $i \in [n]$, sorted by index such that $0 =: \tilde{\alpha}_0 < \tilde{\alpha}_1 < \cdots < \tilde{\alpha}_r < \tilde{\alpha}_{r+1} := \infty$. Let $s^* \in [r+1]$ denote the (unknown) index with $\gamma^* \in [\tilde{\alpha}_{s^*-1}, \tilde{\alpha}_{s^*}[$. For each $s \in [r+1]$, we define a linear program denoted by LP(s) which minimizes the maximum deviation under the assumption that only the predictions for data points $\mathbf{x}_i$ with $\alpha_i \geq \tilde{\alpha}_s$ are bounded from below, while all other predictions can be arbitrarily small.

$$
\begin{aligned}
\min_{\mathbf{w},b,\gamma} \quad & \gamma \\
\text{s.t.} \quad & \langle \mathbf{w}, \mathbf{x}_i \rangle + b \in [\alpha_i - \gamma, \beta_i + \gamma], && i \in [n] \text{ with } \alpha_i \geq \tilde{\alpha}_s, \\
& \langle \mathbf{w}, \mathbf{x}_i \rangle + b \leq \beta_i + \gamma, && i \in [n] \text{ with } \alpha_i < \tilde{\alpha}_s, \\
& \gamma \geq -\beta_i, && i \in [n], \\
& \gamma \geq 0.
\end{aligned}
\tag{LP(s)}
$$

Here, the constraint $\gamma \geq -\beta_i$ is only relevant if $\beta_i < 0$. In this case, it is needed to ensure that the error is at least $-\beta_i$ because a ReLU unit can only output nonnegative values.

Suppose we already knew the optimal index s^*. Observe that, by construction of LP(s^*), a triplet $(\mathbf{w}, b, \gamma)$ is an optimal solution for LP(s^*) if and only if $(\mathbf{w}, b)$ is optimal for the problem ReLU(ℓ^∞-Interval) with objective value γ. Hence, it only remains to show how s^* can be found. To this end, let $\gamma(s)$ be the objective value of LP(s) for each $s \in [r+1]$. Note that $\gamma(s_1) \geq \gamma(s_2)$ for $s_1 < s_2$ because the set of constraints of LP(s_1) is a superset of the constraints of LP(s_2). Hence, for $s > s^*$, it follows that

$$\gamma(s) \leq \gamma(s^*) = \gamma^* < \tilde{\alpha}_{s^*} \leq \tilde{\alpha}_{s-1}.$$

Similarly, for $s < s^*$, we obtain

$$\gamma(s) \geq \gamma(s^*) = \gamma^* \geq \tilde{\alpha}_{s^*-1} \geq \tilde{\alpha}_s.$$

As a consequence, we can determine whether $s < s^*$, $s = s^*$, or $s \geq s^*$ by solving LP(s) and comparing $\gamma(s)$ with $\tilde{\alpha}_s$ and $\tilde{\alpha}_{s-1}$. Thus, using binary search and solving $\mathcal{O}(\log n)$ linear programs, we can determine s^* and the optimal solution γ^* together with the corresponding weights $\mathbf{w}$ and b. □

We remark that, analogously to the original problem with labels y_i, the zero-error case for the variant with intervals $[\alpha_i, \beta_i]$ can be solved with a single linear program instead of a binary search: It suffices to run LP(1) once. This results in objective value 0 if and only if all data points can be fitted precisely within their intervals.

4. Polynomial-time Algorithm for Concave Loss in Fixed Dimension

In this section, we prove that, for any loss function of the form $\ell(\hat{y}, y) = \tilde{\ell}(|\hat{y} - y|)$ where $\tilde{\ell}\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is concave, the problem k-ReLU(ℓ) is polynomial-time solvable for constant d (that is, it is in XP with respect to d). In particular, this covers the case of ℓ^p-loss for $p \in [0, 1[$. Notably, concave loss functions can yield increased robustness by mitigating the influence of outliers. For convex loss functions, in particular for the ℓ^p-loss with $p \geq 1$, an analogous result has already been shown by Arora et al. (2018, Theorem 4.1). More precisely, they showed that, if ℓ is convex, then k-ReLU(ℓ) can be solved in $O(2^k n^{dk} \operatorname{poly}(n, d, k))$ time. The idea of their algorithm is essentially to try out all $O(n^d)$ hyperplane partitions of the n input points for each of the k ReLU neurons and solve a corresponding convex program.

For the concave case, we follow a similar approach. The only but decisive difference is that the occurring subproblems are not convex programs. Instead, we show that they can be written as optimization problems over polyhedra with an objective function that is piecewise concave. It is well-known that global optima of concave problems always occur at a vertex of the feasible polyhedron (Benson, 1995) and that it is possible to enumerate all vertices of the polyhedron in XP-time (Kaibel & Pfetsch, 2003). However, since in our case the objective function is only piecewise concave, it is possible that no vertex is a global optimum. Instead, we need to enumerate all vertices of all concave pieces of the feasible region. We show that this can still be done in XP-time, completing the parameterized complexity classification picture.

Theorem 4. *For every loss function being of the form $\ell(\hat{y}, y) = \tilde{\ell}(|\hat{y} - y|)$ with a concave function $\tilde{\ell}\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, the problem k-ReLU(ℓ) is solvable in time $2^k (nk)^{O(dk)} \operatorname{poly}(n, d, k)$.*

Proof. Following the approach by Arora et al. (2018, Algorithm 1), for each neuron $j \in [k]$, we consider each coefficient $a_j \in \{-1, 1\}$ and each hyperplane partition $P_+^j \cup P_-^j = [n]$, $P_+^j \cap P_-^j = \emptyset$, of the n (indices of the) data points (that is, there exists a $(d-1)$-dimensional hyperplane, defined by a vector $\mathbf{w}_j$ and a bias b_j, separating P_+^j and P_-^j, compare Figure 3). Here, P_+^j is the *active* set, where $\langle \mathbf{w}_j, \mathbf{x}_i \rangle + b_j \geq 0$ shall hold for each $i \in P_+^j$ and $\langle \mathbf{w}_j, \mathbf{x}_i \rangle + b_j \leq 0$ for each $i \in P_-^j$. As in the algorithm by Arora et al. (2018), this results in a total of (at most) $2^k n^{dk}$ subproblems. For fixed coefficients a_j and fixed partitions (P_+^j, P_-^j), $j \in [k]$, the corresponding subproblem (compare Line 8 in Algorithm 1 of Arora et al. (2018)) is the following:

$$\begin{aligned} \min_{\mathbf{w}_j, b_j} \quad & \sum_{i=1}^{n} \tilde{\ell}\left(\left| y_i - \sum_{j\colon i \in P_+^j} a_j (\langle \mathbf{w}_j, \mathbf{x}_i \rangle + b_j) \right| \right) \\ \text{s.t.} \quad & \langle \mathbf{w}_j, \mathbf{x}_i \rangle + b_j \leq 0, \qquad j \in [k], i \in P_-^j, \\ & \langle \mathbf{w}_j, \mathbf{x}_i \rangle + b_j \geq 0, \qquad j \in [k], i \in P_+^j. \end{aligned} \tag{3}$$

In the following, we show that this problem can be solved in XP-time with respect to d.

As argued in the introduction, we may assume without loss of generality that the affine hull of the data points $\mathbf{x}_i$, $i \in [n]$, is the whole space $\mathbb{R}^d$ because, otherwise, we could solve

the problem within a lower-dimensional affine subspace. We first show that this implies that the feasible region $P \subseteq \mathbb{R}^{kd+k}$ of (3) is *pointed*, that is, it has at least one vertex. More precisely, we show that the zero vector $\mathbf{0}_{kd+k}$ is a vertex of P. To do so, we need to show that it satisfies $kd + k$ linearly independent constraints of (3) with equality. Since $\mathbf{0}_{kd+k}$ satisfies every constraint of (3) with equality, we only need to show that there exist $kd + k$ linearly independent rows. We write $\mathbf{r}_{ij} \coloneqq (\mathbf{0}_{d(j-1)}, \mathbf{x}_i, \mathbf{0}_{d(k-j)}, \mathbf{e}_j) \in \mathbb{R}^{kd+k}$, $i \in [n]$, $j \in [k]$, for the kn rows of the constraint matrix, where $\mathbf{e}_j \in \{0,1\}^k$ is the j-th unit vector. By our assumption that the affine hull of the data points is the whole space $\mathbb{R}^d$, there exists a subset $S \subseteq [n]$ of $d+1$ indices such that the $d+1$ vectors $\mathbf{x}_i$, $i \in S$, are affinely independent. This implies that, for each fixed j, the $d + 1$ rows $\mathbf{r}_{ij}$ are linearly independent. Moreover, since, for $j_1 \neq j_2$ and arbitrary $i_1, i_2 \in [n]$, two rows $\mathbf{r}_{i_1 j_1}$ and $\mathbf{r}_{i_2 j_2}$ have non-zero entries only in distinct columns, it follows that the $kd + k$ rows $\mathbf{r}_{ij}$, $i \in S$, $j \in [k]$, are linearly independent. Hence, P is pointed.

Next, we divide the feasible region P of (3) into several polyhedral pieces, depending on the sign of the prediction error at each data point. Let $\mathbf{s} = (s_i)_{i\in[n]} \in \{-1,1\}^n$ be a sign vector and let

$$P(\mathbf{s}) \coloneqq \left\{ (\mathbf{w}_1, \dots, \mathbf{w}_k, b_1, \dots, b_k) \in P \;\middle|\; \forall i \in [n] \colon s_i \left(y_i - \sum_{j \colon i \in P_+^j} a_j (\langle \mathbf{w}_j, \mathbf{x}_i \rangle + b_j) \right) \geq 0 \right\}$$

be the subset of the feasible region P for which the sign of the prediction error for each data point $\mathbf{x}_i$ coincides with s_i. Since P is pointed, $P(\mathbf{s})$ must be pointed as well. Moreover, by definition, the prediction error of every data point has a fixed sign within $P(\mathbf{s})$, implying that the objective function of (3) (as a sum of concave functions) is concave within $P(\mathbf{s})$ (compare Figure 4). In addition, the objective value is trivially bounded from below by 0. Since the minimum of a bounded (from below), concave function over a pointed, nonempty polyhedral set is always attained by a vertex (Benson, 1995), it follows that $P(\mathbf{s})$ is either empty or must have a vertex minimizing the loss within $P(\mathbf{s})$. Since $P = \bigcup_{\mathbf{s}\in\{-1,1\}^n} P(\mathbf{s})$, it follows that the optimal solution of (3) must be a vertex of one of the polyhedral sets $P(\mathbf{s})$. Hence, it suffices to enumerate all these vertices. Compare Figure 5 for a schematic illustration of this idea. Each vertex of one of the polyhedra $P(\mathbf{s})$ is given by $kd + k$ linearly independent inequalities that hold with equality. For selecting these $kd+k$ equations, we have the choice between a total of $kn + n$ equations: the kn constraints of (3) as well as the n equations corresponding to the sign constraints defined by $\mathbf{s}$. Note that these n equations are the same for each $\mathbf{s}$ although the inequalities are different.

We conclude that it suffices to check all $\binom{kn+n}{kd+k} \leq (nk)^{O(dk)}$ possible subsets of $kd + k$ equations. If the chosen equations are linearly independent, then we can determine the corresponding unique solution and check whether it is a feasible solution to (3). For each chosen set of equations, these steps can be done in $\operatorname{poly}(n, d, k)$ time. Among all feasible solutions found that way, we take the best one. Consequently, each of the (at most) $2^k n^{dk}$ subproblems can be solved in $(nk)^{O(dk)} \operatorname{poly}(n, d, k)$ time, resulting in the claimed overall running time. □

In comparison to the algorithm for convex loss functions (Arora et al., 2018), our algorithm for concave loss functions requires more time to solve the $O(2^k n^{dk})$ many subproblems,

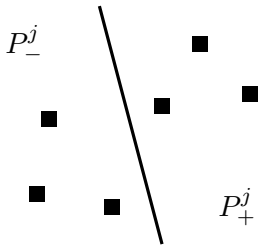

Figure 3: Two-dimensional illustration of the hyperplane partitions of the data points into $P_+^j = \{i \in [n] \mid \langle \mathbf{w}_j, \mathbf{x}_i \rangle + b_j \geq 0\}$ and $P_-^j = \{i \in [n] \mid \langle \mathbf{w}_j, \mathbf{x}_i \rangle + b_j < 0\}$.

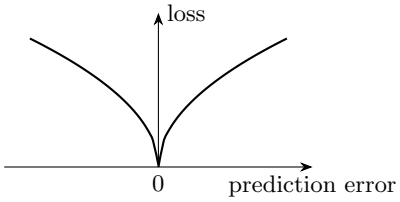

Figure 4: The contribution of a data point $\mathbf{x}_i$ to the objective function is not globally concave. However, it is concave if the sign of the prediction error $y_i - \sum_{j\colon i \in P_+^j} a_j(\langle \mathbf{w}_j, \mathbf{x}_i \rangle + b_j)$ is fixed.

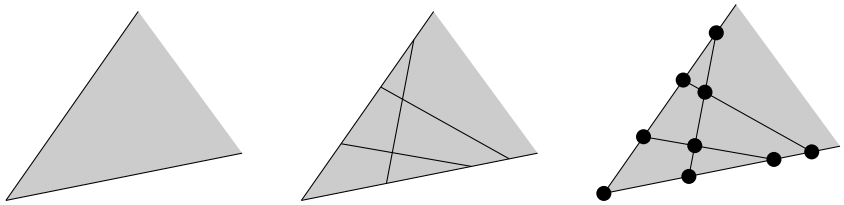

Figure 5: Schematic illustration of how an optimal solution to the subproblem (3) can be found. The feasible region is a pointed polyhedral cone (left). The hyperplanes where the prediction error at a certain data point equals zero subdivide P into the regions $P(\mathbf{s})$ (middle). Since the objective function is concave in each of these regions, it suffices to check the vertices of all regions (right).

namely $(nk)^{O(dk)}\,\mathrm{poly}(n, d, k)$ instead of $\mathrm{poly}(n, d, k)$ time each. This confirms the general theme in optimization that convex problems are easier to solve than non-convex problems. However, due to the combinatorial search, both cases result in an XP overall running time.

5. Conclusion

We closed some gaps regarding the computational complexity of training ReLU networks by proving tight parameterized hardness results and essentially optimal algorithms, thus settling the parameterized complexity. Notably, as Goel et al. (2021) point out, every *proper learning* algorithm also solves the training problem. Hence, our results also imply parameterized hardness of proper learning.

As our results confirm computational intractability also from a parameterized perspective, this motivates the challenging task to identify suitable parameters to achieve fixed-parameter tractability results. For example, parameterizing by some "distance from triviality" measure (e.g. assuming specially structured input data) might be an interesting approach (Niedermeier, 2006). We conclude with some specific open questions:

- What is the parameterized complexity of k-ReLU(ℓ) with respect to d for $k \geq 2$ in the zero-error case? While polynomial-time algorithms for $k = 1$ are available (compare Goel et al. (2021) and Section 3), the zero-error case is NP-hard for $k \geq 2$ (Bakshi

et al., 2019; Goel et al., 2021). Notably, Boob et al. (2022, Section 4) showed NP-hardness of the zero-error case for $k = 2$ if the output neuron is also a ReLU. They give a polynomial-time reduction from the 2-Hyperplane Separability problem, which is known to be W[1]-hard with respect to dimension and to have an ETH-based running time lower bound (Giannopoulos et al., 2009). In fact, the reduction of Boob et al. (2022) is a parameterized reduction with respect to d (the reduction uses two additional dimensions). Thus, as a corollary, we obtain that, for $k = 2$, the zero-error case with ReLU output is W[1]-hard with respect to d and not solvable in $n^{o(d)}$ time assuming ETH.

- Is 1-ReLU(ℓ) fixed-parameter tractable with respect to d if all input points contain at most three non-zero entries? For at most four non-zero entries, we showed W[1]-hardness in Corollary 2.
- Confronting inapproximability results for polynomial-time algorithms (see, e.g., Goel et al. (2021)) and our W[1]-hardness result for exact algorithms (Theorem 1), a natural follow-up question is: Can acceptable worst-case approximation ratios be obtained in FPT-time?
- What is the (parameterized) complexity of training deeper neural networks (with at least three layers)?

Acknowledgments

This work was done while Christoph Hertrich was part of the Combinatorial Optimization and Graph Algorithms Group at Technische Universität Berlin and received funding from DFG-GRK 2434 "Facets of Complexity". Christoph Hertrich would like to thank Amitabh Basu, Marco Di Summa, and Martin Skutella for many valuable discussions about ReLU Neural Networks.

References

Abrahamsen, M., Kleist, L., & Miltzow, T. (2021). Training neural networks is $\exists\mathbb{R}$-complete. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems (NeurIPS '21)*.

Arora, R., Basu, A., Mianjy, P., & Mukherjee, A. (2018). Understanding deep neural networks with rectified linear units. In *Proceedings of the 6th International Conference on Learning Representations (ICLR '18)*.

Bakshi, A., Jayaram, R., & Woodruff, D. P. (2019). Learning two layer rectified neural networks in polynomial time. In *Conference on Learning Theory (COLT '19)*, Vol. 99 of *Proceedings of Machine Learning Research*, pp. 195–268. PMLR.

Bartal, Y., Fandina, N., & Neiman, O. (2019). Dimensionality reduction: theoretical perspective on practical measures. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS '19)*, pp. 10576–10588.

Benson, H. P. (1995). Concave minimization: Theory, applications and algorithms. In *Handbook of Global Optimization*, pp. 43–148. Springer.

Bertschinger, D., Hertrich, C., Jungeblut, P., Miltzow, T., & Weber, S. (2022). Training fully connected neural networks is $\exists\mathbb{R}$-complete. *arXiv preprint*, *arXiv:2204.01368*.

Boob, D., Dey, S. S., & Lan, G. (2022). Complexity of training relu neural network. *Discrete Optimization*, *44*.

Chen, S., Klivans, A. R., & Meka, R. (2021). Learning deep ReLU networks is fixed-parameter tractable. In *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS '21)*.

Chin, T., Cai, Z., & Neumann, F. (2020). Robust fitting in computer vision: Easy or hard?. *International Journal of Computer Vision*, *128*(3), 575–587.

Cygan, M., Fomin, F. V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., & Saurabh, S. (2015). *Parameterized Algorithms*. Springer.

de Araujo, R. W. M., Hirata, R., & Rakotomamonjy, A. (2018). Concave losses for robust dictionary learning. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2176–2180. IEEE.

Dey, S. S., Wang, G., & Xie, Y. (2020). Approximation algorithms for training one-node ReLU neural networks. *IEEE Transactions on Signal Processing*, *68*, 6696–6706.

Diakonikolas, I., Goel, S., Karmalkar, S., Klivans, A. R., & Soltanolkotabi, M. (2020). Approximation schemes for ReLU regression. In *Conference on Learning Theory (COLT '20)*, Vol. 125 of *Proceedings of Machine Learning Research*, pp. 1452–1485. PMLR.

Dinur, I. (2016). Mildly exponential reduction from gap 3SAT to polynomial-gap label-cover. *Electronic Colloquium on Computational Complexity*, *128*.

Downey, R. G., & Fellows, M. R. (2013). *Fundamentals of Parameterized Complexity*. Springer.

Fomin, F. V., Golovach, P. A., & Simonov, K. (2020). Parameterized Complexity of PCA. In *17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2020)*, Vol. 162 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 1:1–1:5.

Giannopoulos, P., Knauer, C., & Rote, G. (2009). The parameterized complexity of some geometric problems in unbounded dimension. In *Parameterized and Exact Computation, 4th International Workshop (IWPEC '09)*, Vol. 5917 of *LNCS*, pp. 198–209. Springer.

Goel, S., Klivans, A. R., Manurangsi, P., & Reichman, D. (2021). Tight hardness results for training depth-2 ReLU networks. In *12th Innovations in Theoretical Computer Science Conference (ITCS '21)*, Vol. 185 of *LIPIcs*, pp. 22:1–22:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Hertrich, C., Basu, A., Di Summa, M., & Skutella, M. (2021). Towards lower bounds on the depth of ReLU neural networks. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems (NeurIPS '21)*.

Impagliazzo, R., & Paturi, R. (2001). On the complexity of k-SAT. *Journal of Computer and System Sciences*, *62*(2), 367–375.

Janocha, K., & Czarnecki, W. M. (2016). On loss functions for deep neural networks in classification. *Schedae Informaticae*, *25*, 49–59.

Jiang, W., Nie, F., & Huang, H. (2015). Robust dictionary learning with capped l1-norm. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Kaibel, V., & Pfetsch, M. E. (2003). Some algorithmic problems in polytope theory. In Joswig, M., & Takayama, N. (Eds.), *Algebra, Geometry and Software Systems*, pp. 23–47. Springer.

Knauer, C., König, S., & Werner, D. (2015). Fixed-parameter complexity and approximability of norm maximization. *Discrete & Computational Geometry*, *53*(2), 276–295.

Niedermeier, R. (2006). *Invitation to Fixed-Parameter Algorithms*. Oxford University Press.

Pilanci, M., & Ergen, T. (2020). Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks. In *International Conference on Machine Learning*, pp. 7695–7705. PMLR.

Silverman, J. H., & Tate, J. (1994). *Rational Points on Elliptic Curves*. Springer.

Simonov, K., Fomin, F. V., Golovach, P. A., & Panolan, F. (2019). Refined complexity of PCA with outliers. In *Proceedings of the 36th International Conference on Machine Learning (ICML '19)*, Vol. 97 of *Proceedings of Machine Learning Research*, pp. 5818–5826. PMLR.

van der Maaten, L., Postma, E., & van der Herik, J. (2009). Dimensionality reduction: A comparative review. *Tilburg University, TiCC TR 2009-005*.

Wang, H., Nie, F., Cai, W., & Huang, H. (2013). Semi-supervised robust dictionary learning via efficient l-norms minimization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1145–1152.