Tight Hardness Results for Training Depth-2 ReLU Networks*

Surbhi Goel† Adam Klivans‡ Pasin Manurangsi§ Daniel Reichman¶

November 30, 2020

Abstract

We prove several hardness results for training depth-2 neural networks with the ReLU activation function; these networks are simply weighted sums (that may include negative coefficients) of ReLUs. Our goal is to output a depth-2 neural network that minimizes the square loss with respect to a given training set. We prove that this problem is NP-hard already for a network with a single ReLU. We also prove NP-hardness for outputting a weighted sum of k ReLUs minimizing the squared error (for $k > 1$) even in the realizable setting (i.e., when the labels are consistent with an unknown depth-2 ReLU network). We are also able to obtain lower bounds on the running time in terms of the desired additive error ϵ. To obtain our lower bounds, we use the Gap Exponential Time Hypothesis (Gap-ETH) as well as a new hypothesis regarding the hardness of approximating the well known Densest κ-Subgraph problem in subexponential time (these hypotheses are used separately in proving different lower bounds). For example, we prove that under reasonable hardness assumptions, any *proper* learning algorithm for finding the best fitting ReLU must run in time exponential in $1/\epsilon^2$. Together with a previous work regarding improperly learning a ReLU [GKKT17], this implies the first separation between proper and improper algorithms for learning a ReLU. We also study the problem of properly learning a depth-2 network of ReLUs with bounded weights giving new (worst-case) upper bounds on the running time needed to learn such networks both in the realizable and agnostic settings. Our upper bounds on the running time essentially matches our lower bounds in terms of the dependency on ϵ.

1 Introduction

Neural networks have become popular in machine learning tasks arising in multiple applications such as computer vision, natural language processing, game playing and robotics [LBH15]. One attractive feature of neural networks is being universal approximations: a network with a single hidden layer[1] with sufficiently many neurons can approximate arbitrary well any measurable real-valued function [HSW89, Cyb89]. These networks are typically trained on labeled data by setting

*This work subsumes our earlier manuscript [MR18].

†Microsoft Research NYC. Email: goel.surbhi@microsoft.com. Work was done while the author was a PhD student at UT Austin and was supported by the JP Morgan AI Research PhD Fellowship.

‡UT Austin. Email: klivans@cs.utexas.edu. Supported by NSF awards AF-1909204, AF-1717896, and the NSF AI Institute for Foundations of Machine Learning (IFML). Work done while visiting the Institute for Advanced Study, Princeton, NJ.

§Google Research. Email: pasin@google.com. Part of this work was done while the author was at UC Berkeley and was partially supported by NSF under Grants No. CCF 1655215 and CCF 1815434.

¶WPI. Email: daniel.reichman@gmail.com.

[1]We also refer to such networks as depth-2 networks or shallow networks.

the weights of the units to minimize the loss function (often the squared loss is used) over the training data. The challenge is to find a computationally efficient way to set the weights to achieve low error. While heuristics such as stochastic gradient descent (SGD) have been successful in practice, our theoretical understand about the amount of running-time needed to train neural networks is still lacking.

It has been known for decades [BR89, Meg88, Jud88] that finding a set of weights that minimizes the loss of the training set is NP-hard. These hardness results, however, only apply to classification problems and to settings where the neural networks involved use discrete, Boolean activations. Our focus here is on neural networks with real inputs whose neurons have the real-valued ReLU activation function. Specifically, we consider depth-2 networks of ReLUs, namely either a single ReLU or a weighted sum of ReLUs [2], and the optimization problem of training them giving labelled data points, which are defined below.

Definition 1. *A* rectifier *is the real function* $[x]_+ := \max(0, x)$. *A* rectified linear unit (ReLU) *is a function* $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ *of the form* $f(\mathbf{x}) = [\langle \mathbf{w}, \mathbf{x} \rangle]_+$ *where* $\mathbf{w} \in \mathbb{R}^n$ *is fixed. A* depth-2 neural network with k ReLUs *(abbreviated as* k-ReLU*) is a function from* $\mathbb{R}^n$ *to* $\mathbb{R}$ *defined by*

$$\text{RELU}_{\mathbf{w}^1,\ldots,\mathbf{w}^k,\mathbf{a}}(\mathbf{z}) = \sum_{j=1}^{k} a_j [\langle \mathbf{w}^j, \mathbf{z} \rangle]_+.$$

Here $\mathbf{x} \in \mathbb{R}^n$ *is the input,* $\mathbf{a} = (a_1, \ldots, a_k) \in \{-1, 1\}^k$ *is a vector of "coefficients",* $\mathbf{w}^j = (w_1^j, \ldots, w_n^j) \in \mathbb{R}^n$ *is a weight vector associated with the* j*-th unit. When* $a_1 = \cdots = a_k = 1$*, we refer to* $\text{RELU}_{\mathbf{w}^1,\ldots,\mathbf{w}^k,\mathbf{a}}(\mathbf{z})$ *as the sum of* k *ReLUs, and we may omit* $\mathbf{a}$ *from the subscript.*

We note that the assumption that $a_1, \ldots, a_k \in \{+1, -1\}$ is without loss of generality (e.g., [PS16]): for any non-zero $a_1, \ldots, a_k \in \mathbb{R} \setminus \{0\}$ and $\mathbf{w}^1, \ldots, \mathbf{w}^k$, we may consider $\hat{a}_1 = \frac{a_1}{|a_1|}, \ldots, \hat{a}_k = \frac{a_k}{|a_k|}$ and $\hat{\mathbf{w}}^1 = |a_1|\mathbf{w}^1, \ldots, \hat{\mathbf{w}}^k = |a_k|\mathbf{w}^k$ instead, which represent the same depth-2 network of k ReLUs.

When training neural networks composed of ReLUs, a popular method is to find, given training data, a set of coefficients and weights for each gate minimizing the squared loss.

Definition 2. *Given a set of* m *samples* $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^n$ *along with* m *labels* $y_1, \ldots, y_m \in \mathbb{R}$*, our goal is to find* $\mathbf{w}^1, \ldots \mathbf{w}^k, \mathbf{a}$ *which minimize the average squared training error of the sample, i.e.,*

$$\min_{\mathbf{w}^1,\ldots,\mathbf{w}^k,\mathbf{a}} \frac{1}{m} \sum_{i=1}^{m} \left(\text{RELU}_{\mathbf{w}^1,\ldots,\mathbf{w}^k,\mathbf{a}}(\mathbf{x}_i) - y_i\right)^2 \tag{1}$$

We refer to the optimization problem (1) as the k-ReLU training problem *(aka* k-ReLU regression*).*

When $\mathbf{w}^j = (w_1^j, \ldots, w_n^j)$ *are assumed to have Euclidean norm at most* 1 *and* y_i *are assumed to be in* $[-k, k]$*, we refer to the optimization problem above as the* bounded k-ReLU training problem*.*

Sometimes we assume that the "coefficient" vector $\mathbf{a}$ is fixed in advance (and known to the optimizer) and not part of the input to the training problem. We mention this explicitly when relevant. Also observe that in the optimization problem above we are looking for a **global minimum** rather than a local minimum. A multiset of samples $\{(\mathbf{x}_i, y_i)\}_{i \in [m]}$ is said to be *realizable* if there exist $\mathbf{w}^1, \cdots, \mathbf{w}^k, \mathbf{a}$ which result in zero training error.

[2] We also assume all the biases of the units are 0.

Our goal is to pin down the computational complexity of the training problem for depth-2 networks of ReLUs, by answering the following question:

Question 1. *What is the worst-case running time of training a k-ReLU?*

We focus on depth-2 networks which are rather involved and give rise to nontrivial algorithmic challenges [VW19, BJW19]. Understanding shallow networks seems to be a prerequisite for understanding the complexity of training networks of depth greater than 2.

1.1 Our results

We first consider arguably the simplest possible network: a single ReLU. We show that, already for such a network, the training problem is NP-hard. In fact, our result even rules out a large factor *multiplicative* approximation of the minimum squared error, as stated below.

Theorem 1 (Hardness of Training a single ReLU)**.** *The 1-ReLU training problem is NP-hard. Furthermore, given a sample of m data points of dimension n it is NP-hard to approximate the optimal squared error within a multiplicative factor of $(nm)^{1/poly\log\log(nm)}$.*

Given such a strong multiplicative inapproximability result, a natural question is whether one can get a good algorithm for *additive* approximation guarantee. Notice that we cannot hope for additive approximation in general, because scaling the samples and their labels can make the additive approximation gap arbitrarily large. Hence, we must consider the bounded 1-ReLU Training problem. For this, we give a simple $2^{O(1/\epsilon^2)}\text{poly}(n,m)$ time algorithm with additive approximation ϵ. Furthermore, it easily generalizes to the case of the bounded k-ReLU Training problem for $k > 1$, but we have to pay a factor of k^5 in the exponent:

Theorem 2 (Training Algorithm)**.** *There is a (randomized) algorithm that can solve the bounded k-ReLU training problem to within any additive error $\epsilon > 0$ in time $2^{O(k^5/\epsilon^2)}poly(n,m)$.*

Perhaps more surprisingly, we can prove a tight running time lower bound for the bounded 1-ReLU training problem, which shows that the term $1/\epsilon^2$ in the exponent is necessary. Our running time lower bound relies on the assumption that there is no subexponential time algorithm for approximating the *Densest κ-Subgraph* problem within any constant (multiplicative) factor. Recall that, in the Densest κ-Subgraph (DκS) problem, we are given a graph $G = (V, E)$ and a positive integer κ. The goal is to select a subset $T \subseteq V$ of κ vertices that induces as many edges as possible. We use $\text{den}_\kappa(G)$ to denote this optimum[3] and N to denote the number of vertices, $|V|$. Our hypothesis can be stated formally as follows.

Hypothesis 1. *For every constant $C \geq 1$, there exist[4] $\delta = \delta(C) > 0$ and $d = d(C) \in \mathbb{N}$ such that the following holds. No $O(2^{\delta N})$-time algorithm can, given an instance (G, κ) of DκS where each vertex of G has degree at most d and an integer ℓ, distinguish between the following two cases:*

- *(Completeness)* $\text{den}_\kappa(G) \geq \ell$.
- *(Soundness)* $\text{den}_\kappa(G) < \ell/C$.

[3] Equivalently, $\text{den}_\kappa(G) := \max_{T \subseteq V, |T|=\kappa} |E(T)|$.

[4] As C increases, δ and d decreases.

While this hypothesis is new (we are the first to introduce it), it seems fair to say that refuting it will require a breakthrough in current algorithms for the DκS problem. There are also other supporting evidences for the validity of this hypothesis: please see the beginning of Appendix C for an additional discussion. As mentioned earlier, assuming this hypothesis, we can prove the tight running time lower bound for the bounded 1-ReLU Training problem:

Theorem 3 (Tight Running Time Lower Bound for 1-ReLU Training)**.** *Assuming Hypothesis 1, there is no algorithm that, for all given $\epsilon > 0$, can solve the bounded 1-ReLU training problem within an additive error ϵ in time $2^{o(1/\epsilon^2)}poly(n, m)$.*

We remark that, akin to standard conventions in the area of fine-grained and parameterized complexity, all lower bounds are stated against algorithms that work for *all* values of ϵ with the specified running time. Indeed, it is possible to significantly speed up the time bound $2^{O(1/\epsilon^2)}\text{poly}(n, m)$ for extreme values of ϵ; for instance, enumerating all possible $\mathbf{w}$ over a $\Theta(\epsilon)$-net[5] of $\mathcal{B}^n$ gives an algorithm that runs in time $O(1/\epsilon)^{O(n)}\text{poly}(m)$, which is asymptotically smaller than $2^{O(1/\epsilon^2)}\text{poly}(n, m)$ when $\epsilon = o\left(\frac{1}{\sqrt{n \log n}}\right)$. Nonetheless, our lower bounds can be extended to include a large range of "reasonable" ϵ. Further discussion on such an extension is provided before Section 1.3.

An interesting consequence of Theorem 3 is that it gives a separation between proper and improper agnostic learning of 1-ReLU. Specifically, [GKKT17] shows that improper agnostic learning of 1-ReLU can be done in $2^{O(1/\epsilon)}\text{poly}(n)$ time, while Theorem 3 rules out such a possibility for proper agnostic learning. (See Appendix F.2 for the relation between learning and training.)

Training k-ReLU: The Realizable Case. An important special case of the k-ReLU Training problem is the realizable case, where there is an unknown k-ReLU that labels every training sample correctly. When $k = 1$, it is straightforward to see that the realizable case of 1-ReLU Training can be phrased as a linear program and hence can be solved in polynomial time. On the other hand, we show that, once $k > 1$, the problem becomes NP-hard:

Theorem 4 (Hardness of Training k-ReLU in the Realizable Case)**.** *For any constant $k \geq 2$, the k-ReLU training problem is NP-hard even in the realizable case.*

Our result is in fact slightly stronger than stated above: specifically, we show that, when the samples can be realizable by a (non-negative) sum of k ReLUs (i.e. k-ReLU when $\mathbf{a}$ is the all-one vector), it is still NP-hard to find a k-ReLU that realizes the samples even if negative coefficients in $\mathbf{a}$ are allowed. Furthermore, while we assume in this theorem that k is a constant independent of n, one can also prove an analogous hardness result, when k grows sufficiently slowly as a function of n. We refer the reader to Appendix E for more details.

Observe that Theorem 4 implies that efficient *multiplicative* approximation for the k-ReLU Training problem is impossible (assuming P$\neq$NP) for $k \geq 2$. As a result, we once again turn to additive approximation. On this front, we can improve the running time of the algorithm in Theorem 2 when we assume that the samples are realizable, as stated below.

Theorem 5 (Training Algorithm in the Realizable Case)**.** *When the given samples are realizable by some k-ReLU, there is a (randomized) algorithm that can solve the bounded k-ReLU training problem to within any additive error $\epsilon > 0$ in time $2^{O((k^3/\epsilon)\log^3(k/\epsilon))}poly(n, m)$.*

[5]Recall that an *δ-net* (also refer to as an *δ-cover*) of a set $S \subseteq \mathbb{R}^n$ is a set $T \subseteq \mathbb{R}^n$ such that, for every $x \in S$, there exists $y \in T$ where $\|x - y\|_2 \leq \delta$. It is well-known that, for any $\delta \in [0, 1]$, there is a δ-net of the unit ball $\mathcal{B}^n$ of size $(3/\delta)^n$ and that it can be found in $(3/\delta)^{O(n)}$ time.

Importantly, the dependency of ϵ in the exponent is $\tilde{O}(1/\epsilon)$, instead of $1/\epsilon^2$ that appeared in the non-realizable case (i.e. Theorems 2 and 3). We can also show that this dependency is tight (up to log factors), in the realizable case, under the Gap Exponential Time Hypothesis (Gap-ETH) [Din16, MR17], a standard complexity theoretic assumption in parameterized complexity (see e.g. [CCK+17]). Gap-ETH states that there exists $\delta > 0$ such that no $2^{o(n)}$-time algorithm can, given a CNF formula with n Boolean variables, distinguish between (i) the case where the formula is satisfiable, and (ii) the case where any assignment violates at least δ fraction of the clauses. Our running time lower bound can be stated more formally as follows.

Theorem 6 (Tight Running Time Lower Bound for the Realizable Case)**.** *Assuming Gap-ETH, for any constant $k \geq 2$, there is no algorithm that, for all given $\epsilon > 0$, can solve the bounded k-ReLU training problem within an additive error ϵ in time $2^{o(1/\epsilon)}poly(n, m)$ even when the input samples are realizable by some k-ReLU.*

Relation to Learning ReLUs. k-ReLU Training is closely related to the problem of *proper learning* of k-ReLU. In fact, an algorithm for the latter also solves the former. Hence, our hardness results immediately implies hardness of proper learning of k-ReLU as well. Furthermore, our algorithm also works for the learning problem. Please refer to Section F.2 for more details.

Stronger Quantifier in Running Time Lower Bounds. As stated earlier, our running time lower bounds in Theorems 3 and 6 hold only against algorithms that work *for all* $\epsilon > 0$. A natural question is whether one can prove lower bounds against algorithms that work only *for some* "reasonable" values of ϵ. As explained in more detail below, we can quite easily also get a lower bound with this latter (stronger) quantifier, for any "reasonable" value of ϵ.

First, our lower bounds in Theorems 3 and 6 both apply in the regime where the lower bounds themselves are $2^{\Theta(n)}$; in other words, $\epsilon = \Theta(1/\sqrt{n})$ in Theorem 3 and $\epsilon = \Theta(1/n)$ in Theorem 6. These are essentially the smallest possible value of ϵ for which the lower bounds in Theorems 3 and 6 can hold, because the aforementioned algorithm that enumerates over an ϵ-net of $\mathcal{B}^n$ solves the problem in time $O(1/\epsilon)^{O(n)}\text{poly}(n)$. On the other hand, for smaller values of ϵ, we can get a running time lower bound easily by "padding" the dimension by "dummy" coordinates that are always zero. For instance, if we start with $\epsilon = \Theta(1/\sqrt{n})$, then we may pad the instance to say $n' = n^2$ dimensions, resulting in the relationship $\epsilon = \Theta(1/\sqrt[4]{n'})$. To summarize, this simple padding technique immediately gives the following stronger quantifier version of Theorem 3:

Theorem 7. *For any non-increasing and efficiently computable*[6] *function $\epsilon : \mathbb{N} \to \mathbb{R}^+$ such that $\omega(\sqrt{\log n}) \leq \frac{1}{\epsilon(n)} \leq o(\sqrt{n})$, assuming Hypothesis 1, there is no algorithm that can solve the bounded 1-ReLU training problem within an additive error $\epsilon(n)$ in time $2^{o(1/\epsilon(n)^2)}poly(n, m)$.*

Notice that the constraint $\omega(\sqrt{\log n}) \leq \frac{1}{\epsilon(n)}$ is also essentially necessary, because for $\epsilon > \sqrt{\frac{\log\log n}{\log n}}$ our algorithm (Theorem 2) already runs in polynomial time. A strong quantifier version of Theorem 6 similar to above can be shown as well (but with $\omega(\log n) \leq \frac{1}{\epsilon(n)} \leq o(n)$). We omit the full (straightforward) proof via padding of Theorem 7; interested readers may refer to the proof of Lemma 3.4 of [DKM19b] which employs the same padding technique.

[6] That is, we assume that computing $\epsilon(n)$ can be done in time poly(n) for any $n \in \mathbb{N}$.

1.2 Independent and concurrent work

There have been several concurrent and independent works to ours that we mention here. We remark that the techniques in these works are markedly different than the ones in this paper. For a single ReLU, [DWX18] proved that the 1-ReLU Training problem is NP-hard. With respect to two ReLUs, [BJW19] showed that finding weights minimizing the squared error of a 2-ReLU is NP-hard, even in the realizable case. The work of [BDL18] considered the problem of training a network with a slightly different architecture, in which there are two ReLUs in the first hidden layer and the final output gate is also a ReLU (instead of a sum gate as in our case); they showed that, for such networks with three ReLUs (two in the hidden layer, one in the output layer), the training problem is NP-hard even for the realizable case. As a result of having an output gate computing a ReLU, our NP-hardness result (regarding training a sum of two ReLUs) does not imply their result and their hardness result does not imply our hardness result for training a sum of 2 ReLUs.

1.3 Related work

The computational aspects of training and learning neural networks has been extensively studied. Due to this, we only focus on those directly related to our results.

We are not aware of a previous work showing that the general k-ReLU training problem is NP-hard for $k > 2$, nor are we aware of previous results regarding the hardness of *approximating* the squared error of a single ReLU. The $k = 2$ case and the $k > 2$ case seem to require different ideas and indeed our proof technique for Theorem 6 is different than those of [BDL18, BJW19]. Moreover, the question of generalizing the NP-hardness result from $k = 2$ to $k > 2$ is mentioned explicitly in [BDL18]. Finally, we remark that neither [DWX18] nor [BJW19] provides explicit running time lower bounds in terms of $1/\epsilon$ for the problem of training k ReLUs within an additive error of ϵ. To the best of our knowledge, our work is the first to obtain such lower bounds.

[Vu98] has proven that finding weights minimizing the squared error of a k-ReLU is NP-hard when $\mathbf{a}$ is the all-one vector (or alternatively, when all the coefficients of the units are restricted to be positive) for every $k \geq 2$.

Some sources (e.g. [ABMM18, Bac17]) attribute (either implicitly or explicitly) the NP-hardness of the k-ReLU Training problem to [BR89], who consider training a neural network with *threshold units*. However, it is unclear (to us) how to derive the NP-hardness of training ReLUs from the hardness results of [BR89]. Several NP-hardness results for training neural networks with architectures differing from the fully connected architecture considered here are known. For example, in [BG17], the training problem is shown to be hard for a depth-2 *convolutional network* with (at least two) *non-overlapping* patches. To the best of our knowledge, these architectural differences render those previous results inapplicable for deriving the hardness results regarding the networks considered in this work.

Several papers have studied a slightly different setting of *improper learning* of neural networks. An example is [LSSS14] who show that improper learning of depth-2 networks of $\omega(1)$ ReLUs is hard, assuming certain average case assumptions. More recently, [GKKT17] show that even for a single ReLU, when $|\langle \mathbf{w}, \mathbf{x} \rangle|$ tends to infinity with n, learning $[\langle \mathbf{w}, \mathbf{x} \rangle]_+$ improperly in time $g(\epsilon) \cdot poly(n)$ is unlikely as it will result in an efficient algorithm for the problem of learning sparse parities with noise which is believed to be intractable. These hardness results for improper learning do imply hardness for the corresponding training problems. Nonetheless, it should be noted that the fact that these results have to rely on assumptions other than P $\neq$ NP is not a coincidence: it

is known that basing hardness of improper learning on P $\neq$ NP alone will result in a collapse of the Polynomial Hierarchy [ABX08].

On the algorithmic side, Arora et al. [ABMM18] provide a simple and elegant algorithm that exactly solves the ReLU training problem in polynomial time assuming the dimension n of the data points is an absolute constant; Arora et al.'s algorithm is for the networks we consider, and it has since been also extended to other types of networks [BDL18]. Additionally, there have also been works on (agnostic) learning algorithms for ReLUs. Specifically, Goel et al. [GKKT17] consider the bounded norm setting where the inputs to the ReLUs as well as the weight vectors of the units have norms at most 1. For this setting, building on kernel methods and tools from approximation theory, they show how to *improperly* learn a single n-variable ReLU up to an additive error of ϵ in time $2^{O(1/\epsilon)} \cdot poly(n)$. Their result generalizes to depth-2 ReLUs with k units with running time of $2^{O(\sqrt{k}/\epsilon)} \cdot poly(n)$ assuming the coefficient vector $\mathbf{a}$ has norm at most 1. The algorithm they provide is quite general: it works for arbitrary distribution over input-output pairs, for ϵ that can be small as $1/\log n$ and also for the reliable setting.

A limitation of our hardness results is that they consider "pathological" training data sets that are specifically constructed to encode intractable combinatorial optimization problems. Several works in literature have tried to overcome this issue by considering the training/learning problems on more "benign" data distributions, such as log-concave distributions or those with Gaussian marginals. On this front, both algorithms and lower bounds have been shown for depth-2 networks [SVWX17, BJW19, GKK19].

Using insights from the study of exponential time algorithms towards understanding the complexity of machine learning problems as is done in this work is receiving attention lately [ST17, DKM19a, SFGP19].

1.4 Organization of the Paper

In the remainder of the main body of this paper, we provide high-level overviews of our proofs (Section 2) and discuss several potential research directions (Section 3). The appendix contains all the details of our proofs and is organized as follows. Appendix A contains several additional notations that will be used throughout the proofs. In Appendix B, we prove the NP-hardness of 1-ReLU Training (Theorem 1). We then prove the running time lower bound for the problem in Appendix C. In Appendix D, we consider the problem of training (non-negative) sum of k ReLUs, and prove hardness for the problem. We then use these hardness to prove our NP-hardness and running time lower bound of the k-ReLU Training problem (Theorems 4, 6) in Appendix E. Finally, our algorithms are presented in Appendix F.

2 Proof Overview

Below we provide the informal overviews of our proofs and intuition behind them. All full proofs can be found in the appendix.

NP-Hardness of Training 1-ReLU. Our reduction is from the (NP-hard) Set Cover problem, in which we are given subsets $T_1, \ldots, T_M$ of a universe U, and the goal is to select as few of these subsets as possible whose union covers the entire universe U. We reduce this to the problem of 1-ReLU Training, where the dimension n is equal to M. We think of each coordinate of $\mathbf{w}$ as

an unknown (i.e. variable); specifically, the desired solution will have $w_i = -1$ iff T_i is picked and 0 otherwise. From this perspective, adding a labelled sample $(\mathbf{x}, y)$ is the same as adding a "constraint" $[\mathbf{w} \cdot \mathbf{x}]_+ = y$. There are two types of constraints we will add:

- (Element Constraint) For each $u \in U$, we add a constraint of the form $\left[1 + \sum_{T_i \ni u} w_i\right]_+ = 0$. The point is that such a constraint is satisfied when u is covered by the selected subsets.
- (Subset Constraint) For each $i \in [M]$, we add a constraint of the form $[\gamma + w_i]_+ = \gamma$ for some small $\gamma > 0$. This constraint will be violated for any selected subset.

By balancing the weights (i.e. number of copies) of each constraint carefully, we can ensure that the element constriants are never unsatisfied, and that the goal is ultimately to violate as few subset constraints as possible, which is equivalent to trying to pick as few subsets as possible that can fully cover U. This completes the high-level overview of our reduction.

We remark that there is a subtle point here because we cannot directly have a constant such that 1 or γ in the constraints themselves. Rather, we need to have "constraint coordinate" and adding the constants through this coordinate. This will also be done in the other reductions presented below, and we will not mention this again.

The outlined proof, together with the $\Theta(\log |U|)$ inapproximability of Set Cover [LY94, Fei98], already gives a hardness of approximation of a multiplicative factor of $\Theta(\log(nm))$ for the 1-ReLU Training problem. To further improve this inapproximability ratio to $(nm)^{1/\text{poly}\log\log(nm)}$, we reduce from the *Minimum Monotone Circuit Satisfiability (MMCS)* problem, which is a generalization of Set Cover. In MMCS, we are given a monotone circuit and the goal is to set as few input wires to true as possible under the condition that the circuit's output must be true. Strong inapproximability results for MMCS are known (e.g. [DHK15]). Our reduction from MMCS proceeds in a similar manner as that of the Set Cover reduction above. Roughly speaking, the modification is that each unknown is now whether each wire is set/evaluated to true, whereas the constraints are now to ensure that the evaluation at each gate is correct and that the output is true.

Tight Running Time Hardness of 1-ReLU Training. We now move on to the proof overview of the tight running time lower bound for 1-ReLU Training. Recall that we will be reducing from the Densest κ-Subgraph (DκS) problem, in which we are given a graph $G = (V, E)$ and $\kappa \in \mathbb{N}$. The goal is to find a set of κ vertices that induces the maximum number of edges.

To motivate our construction, a simple combination of dimensionality reduction and δ-net can in fact find a ReLU that point-wise approximates the optimal ReLU to within an additive factor of δ in time $2^{\tilde{O}(1/\delta^2)}\text{poly}(n)$. That is, if the ReLU that achieves the optimal error has weight vector $\mathbf{w}^*$, then we can find a weight vector $\mathbf{w}$ such that $|[\mathbf{w} \cdot \mathbf{x}]_+ - [\mathbf{w}^* \cdot \mathbf{x}]_+| \leq \delta$ for all input samples $(\mathbf{x}, y)$ in time[7] $2^{\tilde{O}(1/\delta^2)}\text{poly}(n)$.

Indeed, this is an explanation why, in the realizable case, we can get ϵ squared error in $2^{\tilde{O}(1/\delta)}\text{poly}(n)$ time by simply picking $\delta = \sqrt{\epsilon}$. Now, since we need our hardness here (for the non-realizable case) to hold with stronger running time lower bound of $2^{\Theta(1/\epsilon^2)}\text{poly}(n)$, we have to make sure that whenever $\delta \gg \epsilon$, the aforementioned point-wise approximation of δ is not sufficient to get an error of ϵ. Suppose that, for an input labelled sample $(\mathbf{x}, y)$, the optimal ReLU outputs y' and our approximation outputs y'' (where $|y'' - y'| \leq \delta$). Notice that the difference in the square

[7] We assume throughout that $m = \text{poly}(1/\delta)$, which is w.l.o.g. due to standard generalization bounds. See Section F.

error between the two for this sample is only at most $O((y' - y)\delta) + \delta^2$. Now, if we want this quantity to be at least ϵ for any $\delta \geq \Omega(\epsilon)$, then it must be that $|y' - y| = \Omega(1)$. In other words, we have to make our samples so that even the optimal ReLU is "wrong" by $\Omega(1)$ additive factor (on average); this indeed means that, if the ReLU we find is "more wrong" by an additive factor of $\Theta(\epsilon)$, then the increase in the average squared error would be $\Omega(\epsilon)$ as desired.

With the observation in the previous paragraph in mind, we will now provide a rough description of our gadget; they will all be formalized later in the proof of Lemma 14. Given a DκS instance $(G = (V, E), \kappa)$, our samples will have $|V|$ dimensions, one corresponding to each vertex. In the YES case where there is $T \subseteq V$ of size κ that induces many edges, we aim to have our ReLU weight assigning $\frac{1}{\sqrt{\kappa}}$ to all coordinates corresponding to vertices in T, and zero to all other coordinates. To enforce this, we first add a sample for every vertex $v \in V$ that corresponds to the constraint

$$\left[\mathbf{w}_v - \frac{1}{2\sqrt{\kappa}}\right]_+ = 1.$$

We refer to these as the *cardinality constraints*. While this may look peculiar at first glance, the effect is that it ensures that roughly speaking $\mathbf{w}$ has κ coordinates that are "approximately" $\frac{1}{\sqrt{\kappa}}$ and the remaining coordinates are "small". To see that this is the case, observe that the average mean squared error here is $1 - \frac{2}{|V|}\sum_{v\in V}\left[\mathbf{w}_v - \frac{1}{2\sqrt{\kappa}}\right]_+ + \frac{1}{|V|}\sum_{v\in V}\left[\mathbf{w}_v - \frac{1}{2\sqrt{\kappa}}\right]_+^2$. The last term is small and may be neglected. Hence, we essentially have to maximize $\sum_{v\in V}\left[\mathbf{w}_v - \frac{1}{2\sqrt{\kappa}}\right]_+$. This term is indeed maximized when $\mathbf{w}$ has κ coordinates equal to $\frac{1}{\sqrt{\kappa}}$, and zeros in the remaining coordinates. Notice here that this also fits with our intuition from the previous paragraph: even in the optimal ReLU, the value out put by the value (which is either 0 or $\frac{1}{2\sqrt{\kappa}}$) is $\Omega(1)$ away from the input label of the sample (i.e. 1).

So far, the cardinality constraints have ensured that $\mathbf{w}$ "represents" a set $T \subseteq V$ of size roughly κ. However, we have not used the fact that T contains many edges at all. Thus, for every edge $e = \{u, v\} \in E$, we also add the example corresponding to the following constraint to our distribution:

$$\frac{1}{2}\left[\mathbf{w}_u + \mathbf{w}_v - \frac{1.75}{\sqrt{\kappa}}\right]_+ = 1.$$

We call these the *edge constraints*. The point here is that, if e is not an induced edge in T, then the output of the ReLU will be zero. On the other hand, if e is an edge in T, then the output of the ReLU will be $\frac{0.25}{\sqrt{\kappa}}$. Hence, the more edges T induces, the smaller the error.

By carefully selecting weights (i.e. number of copies) of each sample, one can indeed show that the average square error incurred in the completeness and soundness case of Hypothesis 1 differs by $\epsilon = \Omega\left(\frac{1}{\sqrt{|V|}}\right)$. Hence, if we can solve the 1-ReLU Training problem to within an additive error of ϵ in time $2^{o(1/\epsilon^2)}\text{poly}(n, m)$, we can also solve the problem in Hypothesis 1 in time $2^{o(|V|)}$, which breaks the hypothesis.

Hardness of Training k-ReLU in the Realizable Case. We next consider the problems of Training k-ReLU for $k \geq 2$ in the realizable case. Both the NP-hardness result (Theorem 4) and the tight running time lower bound (Theorem 6) employ similar reductions. These reductions proceed

in two steps. First is to reduce from the NP-hard k-coloring problem to the problem of training *non-negative sum of k ReLUs*, in which we fix the coefficient vector $\mathbf{a}$ to be the all-one vector and only seeks to find $\mathbf{w}^1, \ldots, \mathbf{w}^k$ that minimizes the squared error. Then, in the second step, we reduce this to the original problem of k-ReLU Training (where the coefficient vector $\mathbf{a}$ can be negative).

Step I: From k-Coloring to Training Sum of k ReLUs. The NP-hardness of Sum of k ReLUs Training in fact follows directly from a reduction of [Vu98]. We will now sketch Vu's reduction, since it will be helpful in our subsequent discussions below. Vu's reduction starts from the k-coloring problem, in which we are given a hypergraph $G = (V, E)$ and the goal is to determine whether there is a proper k-coloring[8] of the hypergraph. Given an instance $G = (V, E)$ of k-coloring, the number of dimensions in the training problem will be $n = |V|$ where we associate each dimension with a vertex. Notice that now we have k unknowns associated to each vertex v: $w_v^1, \ldots, w_v^k$. In the desired solution, these variables will tell us which color v is assigned to: specifically, $w_v^i > 0$ iff v is colored i and $w_v^i \leq 0$ otherwise.

Adding a labelled sample $(\mathbf{x}, y)$ is the same as adding a "constraint" $[\mathbf{w}^1 \cdot \mathbf{x}]_+ + \cdots + [\mathbf{w}^k \cdot \mathbf{x}]_+ = y$. There are two types of constraints we will add:

- (Vertex Constraint) For every vertex $v \in V$, we add a constraint[9] $[w_v^1]_+ + \cdots + [w_v^k]_+ = 1$. This constraint ensures that, for every $v \in V$, we must have $w_v^{i_v} > 0$ for at least one $i_v \in [k]$, meaning that the vertex v is assigned at least one color.
- (Hyperedge Constraint) For every hyperedge $e = \{v_1, \ldots, v_\ell\} \in E$, we add a constraint[10] $[w_{v_1}^1 + \cdots + w_{v_\ell}^1]_+ + \cdots + [w_{v_1}^k + \cdots + w_{v_\ell}^k]_+ = 0$. This ensures that the hyperedge e is not monochromatic. Otherwise, we have $i_{v_1} = \cdots = i_{v_\ell}$ meaning that $w_{v_1}^{i_{v_1}} + \cdots + w_{v_\ell}^{i_{v_1}} > 0$, which violates the hyperedge constraint.

This finishes our summary of Vu's reduction, which gives the NP-hardness of training a (non-negative) sum of k ReLUs.

Step II: Handling Negative Coefficients. The argument above, especially for the hyperedge constraints, relies on the fact that the coefficient vector $\mathbf{a}$ is the all-one vector. In other words, even if the input hypergraph is not k-coloring, it is still possible that there is a k-ReLU (possibly negative weight vector $\mathbf{a}$) that realizes the samples. Hence, the reduction above does not yet work for our original problem of k-ReLU Training. To handle this issue, we use an additional gadget which is simply a set of labelled samples with the following properties: these samples can be realized by a k-ReLU only when the weight vectors $\mathbf{a}$ is the all-one vector. Essentially speaking, by adding these samples also to our sample set, we have forced $\mathbf{a}$ to be the all-one vector, at which point we restrict ourselves back to the case of (non-negative) sum of k ReLUs and we can use the hard instance from the above reduction from k-coloring. These are the main ideas of the proof of Theorem 4.

Tight Running Time Lower Bound. As stated earlier, the tight running time lower bound for the bounded k-ReLU Training problem (Theorem 6) follows from a similar reduction, except that we now have to (1) carefully select the number of copies of each sample and (2) scale the labels y_i's down so that the norm of each of $\mathbf{w}^1, \ldots, \mathbf{w}^k$ is at most one. Roughly speaking, this means that the labels for the vertex constraints become $\Theta(1/\sqrt{|V|})$ instead of 1 as before. In other words, each violated constraint roughly contributes to $\Theta(1/|V|)$ squared error. Since it is known (assuming

[8] A proper k-coloring is a mapping $\chi : V \to [k]$ such that no hyperedge is monochromatic, or equivalently $|\chi(e)| > 1$ for all $e \in E$.

[9] This constraint corresponds to $\mathbf{x}$ being the v-th vector in the standard basis and $y = 1$.

[10] This constraint corresponds to $\mathbf{x}$ being the indicator vector of e and $y = 0$.

Gap-ETH) that distinguishing between a k-colorable hypergraph and a hypergraph for which every k-coloring violates a constant fraction of the edges takes $2^{\Omega(|V|)}$ time (e.g. [Pet94]), we can arrive at the conclusion that solving the bounded k-ReLU Training problem to within an additive squared error of $\epsilon = \Theta(1/|V|)$ must take $2^{\Omega(1/|V|)} = 2^{\Omega(1/\epsilon)}$ time as desired.

We remark here that, interestingly, [Vu98] used the reduction from k-coloring only for the case of $k = 2$ units and employed an additional gadget to handle the case $k > 2$. To the best of our knowledge, this approach seems to decrease the resulting error ϵ, which means that the running time lower bound is not of the form $2^{\Omega(1/\epsilon)}$. On the other hand, we argue the hardness directly from k-coloring for any constant $k \geq 2$. This, together with a careful selection of the number of copies of each sample, allows us to achieve the running time lower bound in Theorem 6.

Training and Learning Algorithms. Our k-ReLU training algorithm is based on the approach of [ABMM18]. The main idea behind the algorithm is to iterate over all possible sign patterns (whether each ReLU is active or not) of the inputs and subsequently solve the so formed convex optimization for each fixed pattern. The best hypothesis over all different sign patterns is chosen as the the final hypothesis. It is not hard to see that the run-time for such an algorithm would be $2^{(m+1)k}poly(n)$ since there are 2^{mk} different sign patterns.

Using standard generalization bounds, one can show that the number of samples m needed for the empirical loss to be ϵ close to the true loss is at most $O(k^4/\epsilon^2)$. Plugging this into the above algorithm gets us the desired running time ($2^{O(k^5/\epsilon^2)}poly(n, m)$ as in Theorem 2) for the agnostic setting. For the realizable setting, we use an improved generalization result of [SST10], which implies that $m = \tilde{O}(k^2/\epsilon)$ suffices; plugging this into the above algorithm yields us Theorem 5.

3 Conclusions and Open Questions

We have studied the computational complexity of training depth-2 networks with the ReLU activation function providing both NP-hardness results and algorithms for training ReLU's. Along the way we have introduced and used a new hypothesis regarding the hardness of approximating the Denset κ-Subgraph problem in subexponential time that may find applications in other settings. Our results provide a separation between proper and improper learning showing that for a single ReLU, proper learning is likely to be harder than improper learning. Our hardness results regarding properly learning shallow networks suggest that improperly learning such networks (for example, learning overparametrized networks whose number of units far exceeds the dimension of the labeled vectors [AZLL19, DLL$^+$18]) might be necessary to allow for tractable learning problems.

We stress here that our hardness results apply to minimizing the population loss[11] as well, since one may simply create an instance where the population is just the training data. Furthermore, the standard procedure for training neural networks is to perform ERM which is essentially minimizing the training loss. In fact, a bulk of theoretical work in the field focuses on generalization error assuming training error is small (often 0). Therefore, we believe it is a natural question to study the hardness of minimizing training loss.

Neural networks offer many choices (e.g., number of units, depth, choice of activation function, weight restrictions). Indicating which architectures are NP-hard to train can prove useful in guiding the search for a mathematical model of networks that can be trained efficiently. It should be

[11] The population loss is the expected square loss with respect to the distribution of data points.

remembered that our NP-hardness results are worst-case. Therefor they do not preclude efficient algorithms under additional distributional or structural assumptions [Rou20]. Finally, as we focus on networks having significantly fewer units than data-points, the NP-hardness results reported here are not at odds with the ability to train neural networks in the overparmeterized regime where there are polynomial time algorithms that can fit the data with zero error [ZBH$^+$16].

While we restrict our attention to algorithms for training networks with bounded weights, our exponential dependency of the running time on k (the number of units) makes these algorithms impractical. It remains an interesting question whether the dependency of the running time on k can be improved, or alternatively whether strong running time lower bounds can be shown in terms of k (similar to what is done for ϵ in this work).

While we have focused on depth-2 networks, algorithms and lower bounds for deeper networks are of interest as well, especially given the multitude of their practical applications. It would be interesting to see whether the algorithms and hardness results extend to the setting of depth greater than 1. An interesting concrete question here is whether training/learning becomes harder as the network becomes deeper. For instance, is it possible to prove running time lower bounds that grow with the depth of the network?

Acknowledgments

We would like to thank Amit Daniely, Amir Globerson, Meena Jagadeesan and Cameron Musco for interesting discussions.

References

[AAM$^+$11] Noga Alon, Sanjeev Arora, Rajsekar Manokaran, Dana Moshkovitz, and Omri Weinstein. Inapproximabilty of densest k-subgraph from average case hardness. Unpublished Manuscript, 2011.

[ABMM18] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*, 2018.

[ABMP01] Michael Alekhnovich, Samuel R. Buss, Shlomo Moran, and Toniann Pitassi. Minimum propositional proof length is NP-hard to linearly approximate. *J. Symb. Log.*, 66(1):171–191, 2001.

[ABX08] Benny Applebaum, Boaz Barak, and David Xiao. On basing lower-bounds for learning on worst-case assumptions. In *FOCS*, pages 211–220, 2008.

[AOW15] Sarah R. Allen, Ryan O'Donnell, and David Witmer. How to refute a random CSP. In *FOCS*, pages 689–708, 2015.

[AZLL19] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pages 6155–6166, 2019.

[B+15] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[Bac17] Francis Bach. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.

[BCV+12] Aditya Bhaskara, Moses Charikar, Aravindan Vijayaraghavan, Venkatesan Guruswami, and Yuan Zhou. Polynomial integrality gaps for strong SDP relaxations of densest k-subgraph. In *SODA*, pages 388–405, 2012.

[BDL18] Digvijay Boob, Santanu S Dey, and Guanghui Lan. Complexity of training relu neural network. *arXiv preprint arXiv:1809.10787*, 2018.

[BG17] Alon Brutzkus and Amir Globerson. Globally optimal gradient descent for a convnet with gaussian inputs. *arXiv preprint arXiv:1702.07966*, 2017.

[Bha18] Amey Bhangale. NP-hardness of coloring 2-colorable hypergraph with poly-logarithmically many colors. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[BJW19] Ainesh Bakshi, Rajesh Jayaram, and David P Woodruff. Learning two layer rectified neural networks in polynomial time. In *Conference on Learning Theory*, pages 195–268, 2019.

[BM02] Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[BR89] Avrim Blum and Ronald L Rivest. Training a 3-node neural network is NP-complete. In *Advances in neural information processing systems*, pages 494–501, 1989.

[CCK+17] Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From Gap-ETH to FPT-inapproximability: Clique, dominating set, and more. In *FOCS*, pages 743–754, 2017.

[CMMV17] Eden Chlamtác, Pasin Manurangsi, Dana Moshkovitz, and Aravindan Vijayaraghavan. Approximation algorithms for label cover and the log-density threshold. In *SODA*, pages 900–919, 2017.

[Cyb89] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[DHK15] Irit Dinur, Prahladh Harsha, and Guy Kindler. Polynomially low error PCPs with polyloglog n queries via modular composition. In *STOC*, pages 267–276, 2015.

[Din16] Irit Dinur. Mildly exponential reduction from gap 3SAT to polynomial-gap label-cover. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:128, 2016.

[DKM19a] Ilias Diakonikolas, Daniel Kane, and Pasin Manurangsi. Nearly tight bounds for robust proper learning of halfspaces with a margin. In *Advances in Neural Information Processing Systems*, pages 10473–10484, 2019.

[DKM19b] Ilias Diakonikolas, Daniel M. Kane, and Pasin Manurangsi. Nearly tight bounds for robust proper learning of halfspaces with a margin. *CoRR*, abs/1908.11335, 2019.

[DLL+18] Simon S Du, Jason D Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. *arXiv preprint arXiv:1811.03804*, 2018.

[DRS05] Irit Dinur, Oded Regev, and Clifford D. Smyth. The hardness of 3-uniform hypergraph coloring. *Combinatorica*, 25(5):519–535, 2005.

[DS04] Irit Dinur and Shmuel Safra. On the hardness of approximating label-cover. *Inf. Process. Lett.*, 89(5):247–254, 2004.

[DWX18] Santanu S Dey, Guanyi Wang, and Yao Xie. An approximation algorithm for training one-node relu neural network. *arXiv preprint arXiv:1810.03592*, 2018.

[Fei98] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[GKK19] Surbhi Goel, Sushrut Karmalkar, and Adam Klivans. Time/accuracy tradeoffs for learning a relu with respect to gaussian marginals. In *Advances in Neural Information Processing Systems*, pages 8582–8591, 2019.

[GKKT17] Surbhi Goel, Varun Kanade, Adam R. Klivans, and Justin Thaler. Reliably learning the relu in polynomial time. In *COLT*, pages 1004–1042, 2017.

[Hau92] David Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Inf. Comput.*, 100(1):78–150, 1992.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[Jud88] Stephen Judd. On the complexity of loading shallow neural networks. *Journal of Complexity*, 4:177–192, 1988.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, pages 85–103, 1972.

[KSS94] Michael J. Kearns, Robert E. Schapire, and Linda Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2-3):115–141, 1994.

[KST08] Sham M. Kakade, Karthik Sridharan, and Ambuj Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual*

Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008, pages 793–800, 2008.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[Lov73] Laszlo Lovasz. Coverings and colorings of hypergraphs. In *Proc. 4th Southeastern Conf. on Comb.*, pages 3–12, 1973.

[LSSS14] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. On the computational efficiency of training neural networks. In *Advances in Neural Information Processing Systems*, pages 855–863, 2014.

[LT91] Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces: isoperimetry and processes*. Springer, Berlin, May 1991.

[LY94] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994.

[Man15] Pasin Manurangsi. On approximating projection games. Master's thesis, Massachusetts Institute of Technology, January 2015.

[Man17] Pasin Manurangsi. Almost-polynomial ratio ETH-hardness of approximating densest k-subgraph. In *STOC*, pages 954–961, 2017.

[Meg88] Nimrod Megiddo. On the complexity of polyhedral separability. *Discrete & Computational Geometry*, 3(4):325–337, 1988.

[MR17] Pasin Manurangsi and Prasad Raghavendra. A birthday repetition theorem and complexity of approximating dense csps. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 78:1–78:15, 2017.

[MR18] Pasin Manurangsi and Daniel Reichman. The computational complexity of training relu(s). *CoRR*, abs/1810.04207, 2018.

[Pet94] Erez Petrank. The hardness of approximation: Gap location. *Computational Complexity*, 4:133–157, 1994.

[PS16] Xingyuan Pan and Vivek Srikumar. Expressiveness of rectifier networks. In *International Conference on Machine Learning*, pages 2427–2435, 2016.

[Rou20] Tim Roughgarden. Beyond the worst-case analysis of algorithms, 2020.

[Sch08] Grant Schoenebeck. Linear level lasserre lower bounds for certain k-CSPs. In *FOCS*, pages 593–602, 2008.

[SFGP19] Kirill Simonov, Fedor Fomin, Petr Golovach, and Fahad Panolan. Refined complexity of PCA with outliers. In *International Conference on Machine Learning*, pages 5818–5826, 2019.

[SST10] Nathan Srebro, Karthik Sridharan, and Ambuj Tewari. Smoothness, low noise and fast rates. In John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, editors, *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pages 2199–2207. Curran Associates, Inc., 2010.

[ST17] Rocco A Servedio and Li-Yang Tan. What circuit classes can be learned with non-trivial savings? In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[Sto73] Larry Stockmeyer. Planar 3-colorability is polynomial complete. *SIGACT News*, 5(3):19–25, July 1973.

[SVWX17] Le Song, Santosh Vempala, John Wilmes, and Bo Xie. On the complexity of learning neural networks. In *Advances in Neural Information Processing Systems*, pages 5514–5522, 2017.

[Tul09] Madhur Tulsiani. CSP gaps and reductions in the lasserre hierarchy. In *STOC*, pages 303–312, 2009.

[Vu98] Van H Vu. On the infeasibility of training neural networks with small mean-squared error. *IEEE Transactions on Information Theory*, 44(7):2892–2900, 1998.

[VW19] Santosh Vempala and John Wilmes. Polynomial convergence of gradient descent for training one-hidden-layer neural networks. In *Conference on Learning Theory*, pages 3115–3117, 2019.

[ZBH+16] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

A Preliminaries and Notation

We use $\mathcal{B}^n = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\|^2 \leq 1\}$ to denote the (closed) unit ball and $\mathcal{S}^{n-1} = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\|^2 = 1\}$ to denote the unit sphere in n dimensions. Moreover, we use $\mathbf{e}_i$ to denote the i-th vector in the standard basis, i.e., $\mathbf{e}_i$ has its i-th coordinate being 1 and other coordinates being zeros.

For any $n, k \in \mathbb{N}$, $\mathbf{w}^1, \ldots, \mathbf{w}^k \in \mathbb{R}^n$, $\mathbf{a} \in \{-1, 1\}^k$ and distribution $\mathcal{D}$ over $\mathbb{R}^n \times \mathbb{R}$, let

$$\mathcal{L}(\mathbf{w}^1, \ldots, \mathbf{w}^k, \mathbf{a}; \mathcal{D}) := \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\left(\mathrm{RELU}_{\mathbf{w}^1,\ldots,\mathbf{w}^k,\mathbf{a}}(\mathbf{x}) - y_i \right)^2 \right]$$

to denote the expected squared loss with respect to $\mathcal{D}$. We may write a sequence of labelled samples $S = ((\mathbf{x}_i, y_i))_{i\in[m]}$ in place of $\mathcal{D}$ to denote the expression when the distribution is uniform over S.

With this notation, the k-ReLU Training problem is: given a multiset of labelled samples $S = \{(\mathbf{x}_i, y_i)\}_{i\in[m]}$ where $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^n, y_1, \ldots, y_m \in \mathbb{R}$, find $\mathbf{w}^1, \ldots, \mathbf{w}^k \in \mathbb{R}^n$ and $\mathbf{a} \in \{-1, 1\}^k$ that minimizes $\mathcal{L}(\mathbf{w}^1, \ldots, \mathbf{w}^k, \mathbf{a}; S)$. The bounded k-ReLU Training problem is similar except that $\mathbf{x}_1, \ldots, \mathbf{x}_m \subseteq \mathcal{B}^n, y_1, \ldots, y_m \in [-k, k]$ and the minimization is over $\mathbf{w}^1, \ldots, \mathbf{w}^k \in \mathcal{B}^n, \mathbf{a} \in \{-1, 1\}^k$.

Additionally, we define the *bounded sum of k-ReLU Training* problem to be the restriction of the bounded k-ReLU Training in which we only consider $\mathbf{a} = (1, \ldots, 1)$.

For brevity, when $\mathbf{a}$ is the all-one vector (i.e. $a_1 = \cdots = a_k = 1$), we may drop $\mathbf{a}$ from the notation and simply write $\mathcal{L}(\mathbf{w}; \mathcal{D}) := \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[(\mathrm{RELU}_{\mathbf{w}^1,\ldots,\mathbf{w}^k,(1,\ldots,1)}(\mathbf{x}) - y_i)^2]$.

Furthermore, we use $\mathbf{x} \circ \mathbf{x}'$ to denote the concatenation between vectors $\mathbf{x}$ and $\mathbf{x}'$, and $\mathbf{0}_p, \mathbf{1}_p$ for $p \in \mathbb{N}$ to denote the p-dimensional all-zero vector and the p-dimensional all-one vector respectively.

B Hardness of training a single ReLU

In this section, we prove our hardness of 1-ReLU Training (Theorem 1). Specifically, our first result is the NP-hardness of the problem stated below, which proves the first part of Theorem 1.

Theorem 8. *1-ReLU Training problem is NP-hard, even when the samples* $\mathbf{x}_i$ *belong to* $\{-1, 0, 1\}^n$.

In all of our hardness reductions for 1-ReLU (both in this section and Section C), we will always consider non-negative labels y_i, which means that it is always better to pick a_1 to be 1 and not -1. For convenience, we will assume this throughout and do not explicitly state that $a_1 = 1$.

Proof of Theorem 8. We reduce the Set Cover problem to the 1-ReLU Training problem. Recall that, in the Set Cover problem, we are given a finite set U along with a family $\mathcal{T} = \{T_1, \ldots, T_M\}$ of M subsets of U. Our goal is to determine if one can choose t subsets from $\mathcal{T}$ whose union equals U. Set Cover is well known to be NP-hard [Kar72].

We consider a ReLU on $n = M+2$ dimensions, where we view each coordinate of the (unknown) weight vector $\mathbf{w}$ as a variable. Specifically, for each $T_i \in \mathcal{T}$, we have a variable w_{T_i}. In addition, we have two dummy variable w_1 and w_γ. We let $\gamma = 0.01/M^2$.

We introduce the following labelled samples. First, for each $u \in U$, let $\mathbf{x}_u$ be n-dimensional vector having 1 for the coordinate corresponding to the dummy variable w_1, -1 in all coordinates that correspond to a subset $T_i \in \mathcal{T}$ containing u, and 0 to all other coordinates. (In other words, $\mathbf{x}_u = \mathbf{e}_1 + \sum_{T_i \ni u} \mathbf{e}_{T_i}$.) We label this vector by $y_u = 0$. This labelled sample corresponds to the constraint

$$\left[w_1 + \sum_{T_i \ni u} w_{T_i} \right]_+ = 0 \tag{2}$$

Second, for every $T_i \in \mathcal{T}$ let $\mathbf{x}_{T_i}$ be the n-dimensional vector having -1 in the T_i-th coordinate, 1 in the coordinate corresponding to w_γ and 0 for all other coordinates. We label this vector by γ. (In other words, $\mathbf{x}_{T_i} = \mathbf{e}_\gamma + \mathbf{e}_{T_i}$ and $y_{T_i} = \gamma$.) This corresponds to the constraint

$$[w_\gamma + w_{T_i}]_+ = \gamma \tag{3}$$

We add the n-dimensional vector having 1 in the coordinate corresponding to w_1 and 0 elsewhere. We label these vectors by 1. This corresponds to the constraint

$$[w_1]_+ = 1 \tag{4}$$

We add the n-dimensional vector having 1 in the coordinate corresponding to w_γ and 0 elsewhere. We label these vectors by γ. This corresponds to the constraint

$$[w_\gamma]_+ = \gamma \tag{5}$$

In summary, the sample multiset is $S = \{(\mathbf{x}_u, y_u)\}_{u \in U} \cup \{(\mathbf{x}_{T_i}, y_{T_i})\}_{T_i \in \mathcal{T}} \cup \{(\mathbf{e}_1, 1)\} \cup \{(\mathbf{e}_\gamma, \gamma)\}$. Clearly this reduction runs in polynomial time. We now prove the correctness of this reduction.

(YES Case) Assume there is a cover of size t for the set cover instance; suppose without loss of generality that this cover consists of the first t subsets $T_1, \ldots, T_t$ in $\mathcal{T}$. Assigning $w_{T_1} = w_{T_2} = \ldots = w_{T_t} = -1, w_1 = 1$, $w_\gamma = \gamma$ and 0 to all other variables results in an average squared error of $\frac{\gamma^2 \cdot t}{|S|}$. This because exactly t of the constraints from (3) are violated and each violated constraint contributes γ^2 to the squared error. All other constraints are satisfied.

(NO Case) Suppose contrapositively that there is a weight vector $\mathbf{w}$ such that $\mathcal{L}(\mathbf{w}; S) \leq \frac{\gamma^2 \cdot t}{|S|}$. First, observe that $w_1 \geq 0.9$; otherwise, the squared error from (4) alone is more than $(0.1)^2 \geq \gamma^2 t$. Observe also that $w_\gamma \leq 0.2/M$; otherwise, the squared error from (5) must be more than $(0.2/M - \gamma)^2 \geq (0.1/M)^2 > \gamma^2 t$.

Our main observation is that the family $\mathcal{T}_{<-w_\gamma} = \{T_i : w_{T_i} < -w_\gamma\}$ is a set cover. The reason is as follows: by the definition of γ we have that $\sum_{T_i \in (\mathcal{T} \setminus \mathcal{T}_{<-w_\gamma})} w_{T_i} \geq -w_\gamma \cdot M \geq -0.2$. As a result, if there is an element $u \in U$ that is not covered then the corresponding constraint (2) for u will incur already a square error of at least $(0.7)^2 > \gamma^2 t$ (recall that t is no larger than m). Thus, the observation follows.

The last step of the proof is to show that the family $\mathcal{T}_{<-w_\gamma}$ contains at most k subsets. To see that this is the case, observe that, for every $T_i \in \mathcal{T}_{<-w_\gamma}$, we have $[w_\gamma + w_{T_i}]_+ = 0$, meaning that the corresponding constraint (3) incurs a squared error of γ^2. Since the total squared error is at most $\gamma^2 t$, we can immediately concludes that at most t subsets belong to $\mathcal{T}_{<-w_\gamma}$.

Thus, $\mathcal{T}_{<-w_\gamma}$ is a set cover with at most t subsets, which completes the NO case of the proof. □

Observe that in the hardness result above the set of samples is not realizable. This is not a coincidence as it is a simple result that when there are set of weights with zero error the training problem for a single ReLU is solvable in polynomial time via a simple application of linear programming.

B.1 Hardness of Approximating Minimum Training Error for a single ReLU

The reduction above coupled with the fact that set cover is hard to approximate within a factor $O(\log |U|)$ [LY94, Fei98] in fact immediately implies that the problem of approximating the minimum squared error to within a *multiplicative* factor of $O(\log(nm))$ is also hard. In this subsection, we will substantially improve this inapproximability ratio. Specifically, we will show that this problem is hard to approximate to within an almost polynomial (i.e. $(nm)^{1/\text{poly}\log\log(nm)}$) factor, thereby proving the second part of Theorem 1:

Theorem 9. *1-ReLU Training problem is NP-hard to approximate to within a factor of* $(nm)^{1/(\log\log(nm))^{O(1)}}$ *where n is the dimension of the samples and m is the number of samples.*

To prove Theorem 9, we will reduce from the Minimum Monotone Circuit Satisfiability problem, which is formally defined below.

Definition 3. *A* monotone circuit *is a circuit where each gate is either an OR or and AND gate. We use $|C|$ to denote the number of wires in the circuit.*

Definition 4. *In the* Minimum Monotone Circuit Satisfiability$_i$ (MMCS$_i$) *problem, we are given a monotone circuit of depth i, and the objective is to assign as few* Trues *as possible to the input wires while ensuring that the circuit is satisfiable (i.e. output wire is evaluated to* True*).*

The hardness of approximating MMCS has long been studied (e.g. [ABMP01, DS04]). The problem was known to be NP-hard to approximate to within a factor of $2^{\log^{1-\epsilon}|C|}$ for any constant $\epsilon > 0$ [DS04]. This has recently been improved to $|C|^{1/(\log\log|C|)^{O(1)}}$ by Dinur et al. [DHK15][12].

Theorem 10 ([DHK15])**.** MMCS_3 *is NP-hard to approximate to within* $|C|^{1/(\log\log|C|)^{O(1)}}$ *factor.*

The main result of this subsection is that, for any constant $\ell > 0$, there is a polynomial-time reduction from MMCS_ℓ to the problem of minimizing the training error in single ReLU such that the optimum of the latter is proportional to the optimum of the former. From Theorem 10 above, this immediately implies Theorem 9. The reduction is stated and proved below.

Theorem 11. *For every* $\ell > 0$*, there is a polynomial-time reduction that takes in a depth-*ℓ *monotone circuit* C *and produces samples* $\{(\mathbf{x}_i, y_i)\}_{i\in[m]}$ *where* $\mathbf{x}_i \in \{0,1\}^n$ *such that the minimum squared training error*[13] *for these samples among all single ReLUs is exactly* $\text{OPT}_{\text{MMCS}}(C)/(10|C|)^{2\ell+2}$.

Proof. Let $\gamma := 1/(10|C|)^{\ell+1}$. We consider a ReLU with $n = |C| + 1$ variables. For each wire j, we create a variable w_j. Additionally, we have a dummy variable w_γ. (Note that, in the desired solution, we want w_j to be 1 iff the wire is evaluated to TRUE and 0 otherwise, and $w_\gamma = \gamma$.)

Dummy Variable Constraint. We add the following constraint

$$[w_\gamma]_+ = \gamma. \tag{6}$$

Input Wire Constraint. For each input wire i, we add the constraint

$$[w_\gamma - w_i]_+ = \gamma. \tag{7}$$

Output Wire Constraint. For the output wire o, we add the constraint

$$[w_o]_+ = 1. \tag{8}$$

OR Gate Constraint. For each OR gate with input wires $i_1, \ldots, i_k$ and output wire j, we add the constraint

$$[w_j - w_{i_1} - \cdots - w_{i_k}]_+ = 0. \tag{9}$$

AND Gate Constraint. For each AND gate with input wires $i_1, \ldots, i_k$ and output wire j, we add the following k constraints:

$$[w_j - w_{i_1}]_+ = 0, \cdots, [w_j - w_{i_k}]_+ = 0. \tag{10}$$

We will now show that the minimum squared error possible is exactly $\text{OPT}_{\text{MMCS}}(C) \cdot \gamma^2$.

First, we will show that the error is at most $\text{OPT}_{\text{MMCS}}(C) \cdot \gamma^2$. Suppose that ϕ is an assignment to C with $\text{OPT}_{\text{MMCS}}(C)$ TRUEs that satisfies the circuit. We assign $w_\gamma = \gamma$, and, for each wire j,

[12] It should be noted that Dinur et al. [DHK15] in fact shows that there exists a PCP with $D = (\log\log n)^{O(1)}$ query over alphabet of size $n^{O(1/D)}$ with perfect completeness and soundness at most $1/n$. The result we use (Theorem 10) follows from their result and from the reduction in Section 3 of [DS04] which shows how to reduce D-query PCP over alphabet F with perfect completeness and soundness s to an MMCS_3 instance of size $F^D\text{poly}(n)$ and gap $O(1/s)^{1/D}/D$. Plugging this in immediately implies the hardness we use.

[13] For convenience, we are using the *total* squared error (i.e. $|S| \cdot \mathcal{L}(\mathbf{w}; S)$), not the average squared error (i.e. $\mathcal{L}(\mathbf{w}; S)$), in the theorem statement and its proof.

we assign w_j to be 1 if and only if the wire j is evaluated to be TRUE on input ϕ and 0 otherwise. It is clear that every constraint is satisfied, except the input wire constraints (7) for the wires that are assigned to TRUE by ϕ. There are exactly $\mathrm{OPT}_{\mathrm{MMCS}}(C)$ such wires, and each contributes γ^2 to the error; as a result, the training error of such weights is exactly $\mathrm{OPT}_{\mathrm{MMCS}}(C) \cdot \gamma^2$.

Next, we will show that the minimum squared training error has to be at least $\mathrm{OPT}_{\mathrm{MMCS}}(C) \cdot \gamma^2$. Suppose for the sake of contradiction that the minimum error δ is less than $\mathrm{OPT}_{\mathrm{MMCS}}(C) \cdot \gamma^2$. Observe that, from $\mathrm{OPT}_{\mathrm{MMCS}}(C) \leq |C|$ and from our choice of γ, we have

$$\delta < |C| \cdot \gamma^2 < 0.1 \tag{11}$$

Consider an assignment ϕ that assigns each input wire i to be TRUE iff $w_i \geq w_\epsilon$. At the heart of this proof is the following proposition, which bounds the weight of every FALSE wire.

Proposition 12. *For any wire j at height h that is evaluated to* FALSE *on ϕ, $w_j \leq (2|C|)^h \cdot (\gamma + \sqrt{\delta})$.*

We note here that we define the height recursively by first letting the heights of all input wires be zero and then let the height of the output wire of each gate G be one plus the maximum of the heights among all input wires of G.

Proof of Proposition 12. We will prove by induction on the height h.

Base Case. Consider any input wire i (of height 0) that is assigned FALSE by ϕ. By definition of ϕ, we have $w_i < w_\gamma$. Note that w_γ must be at most $\gamma + \sqrt{\delta}$, as otherwise the squared error incurred in (6) is already more than δ. Thus, we have $w_i \leq \gamma + \sqrt{\delta}$ as claimed.

Inductive Step. Let $h \in \mathbb{N}$ and suppose that the statement holds for every FALSE wire at height less than h. Let j be any FALSE at height h. Let us consider two cases:

- j is an output of an OR gate. Let $i_1, \ldots, i_k$ be the inputs of the gate. Since j is evaluated to FALSE, $i_1, \ldots, i_k$ must all be evaluated to FALSE. From our inductive hypothesis, we have $w_{i_1}, \ldots, w_{i_k} \leq (2|C|)^{h-1} \cdot (\gamma + \sqrt{\delta})$. Now, observe that w_j can be at most $\sqrt{\delta} + w_{i_1} + \cdots + w_{i_k}$, as otherwise the squared error incurred in (9) would be more than δ. As a result, we have

$$w_j \leq \sqrt{\delta} + k \cdot (2|C|)^{h-1} \cdot (\gamma + \sqrt{\delta}) = \sqrt{\delta} + |C| \cdot (2|C|)^{h-1} \cdot (\gamma + \sqrt{\delta}) \leq (2|C|)^h \cdot (\gamma + \sqrt{\delta}).$$

- j is an output of an AND gate. Let $i_1, \ldots, i_k$ be the inputs of the gate. Since j is evaluated to FALSE, at least one of $i_1, \ldots, i_k$ must all be evaluated to FALSE. Let i be one such wire. Observe that w_j can be at most $\sqrt{\delta} + w_i$, as otherwise the squared error incurred in (10) would be more than δ. Hence, we have

$$w_j \leq \sqrt{\delta} + w_i \leq \sqrt{\delta} + (2|C|)^{h-1} \cdot (\gamma + \sqrt{\delta}) \leq (2|C|)^h \cdot (\gamma + \sqrt{\delta}).$$

  where the second inequality comes from the inductive hypothesis.

In both cases, we have $w_j < (2|C|)^h \cdot (\gamma + \sqrt{\delta})$, which concludes the proof of Proposition 12. □

Now, consider the output wire o. We claim that o must be evaluated to TRUE on ϕ. This is because, if o is a FALSE wire, then Proposition 12 ensures that w_o is at most

$$(2|C|)^\ell \cdot (\gamma + \sqrt{\delta}) \overset{(11)}{<} (2|C|)^\ell \cdot (\gamma + \sqrt{|C|} \cdot \gamma) \leq 0.1,$$

where the second inequality comes from our choice of γ. This would mean that the squared error incurred in (8) is at least $0.81 > \delta$. Thus, it must be that ϕ satisfies C.

Finally, observe that, since ϕ assigns each input wire i to be TRUE iff $w_i \geq w_\gamma$, each input wire that is assigned TRUE incurs a squared error of γ^2 from (7). As a result, the number of input wires assigned TRUE is at most $\frac{\delta}{\gamma^2} < \mathrm{OPT}_{\mathrm{MMCS}}(C)$, which is a contradiction as we argued above that ϕ satisfies C. This concludes our proof. □

C Running Time Lower Bound for 1-ReLU Training

In this section, we prove our nearly tight running time lower bound for bounded 1-ReLU Training (Theorem 3). Recall that our lower bound relies on the hypothesis that there is no $2^{o(N)}$-time algorithm that can approximate Densest κ-Subgraph to within any (multiplicative) constant factor (Hypothesis 1). While this hypothesis might seem strong (especially given the fact that there is no known large constant factor inapproximability for DκS although we have such hardness under stronger assumptions, e.g. [AAM+11, Man17]), it should be noted that refuting it seems to be out of reach of known techniques. In particular, it is known that $o(N)$-level of the Sum-of-Squares Hierarchies do not give constant factor approximation for DκS even for bounded degree graphs [BCV+12, CMMV17, Man15]. Furthermore, these Sum-of-Squares lower bounds are proved via reductions from a certain family of random CSPs, whose Sum-of-Squares lower bounds are shown in [Sch08, Tul09]. This means that, if Hypothesis 1 is false, then one can refute this family of sparse random CSPs in subexponential time. This would constitute an arguably surprising development in the area of refuting random CSPs, which has been extensively studied for decades (see [AOW15] and references therein).

We stress here that our lower bound in Theorem 3 only holds if we are only allowed to consider a ReLU with weight vector $\mathbf{w}$ within $\mathcal{B}^n$ (i.e. having norm at most 1). If we modify the problem so that we are allowed to output a ReLU with arbitrary weight, then our lower bound in Theorem 3 does not hold. It is an interesting open problem whether one can extend our lower bound to such modified problem as well, or whether a faster algorithm exists in that case.

The rest of this section is devoted to proving Theorem 3. The proof is easier to state if we consider a slight modification of Hypothesis 1 where in the soundness we do not only guarantee that $\mathrm{den}_\kappa(G)$ is small but also that $\mathrm{den}_{B\kappa}(G)$ is small for some large constant B, as stated below.

Hypothesis 2. *For any constants $C, B \geq 1$, there exist $\delta = \delta(C, B) > 0$ and $d = d(C, B) \in \mathbb{N}$ such that the following holds. No $O(2^{\delta N})$-time algorithm can, given an instance (G, κ) of DκS where each vertex of G has degree at most d and an integer ℓ, distinguish between the following two cases:*

- *(Completeness)* $\mathrm{den}_\kappa(G) \geq \ell$.
- *(Soundness)* $\mathrm{den}_{B\kappa}(G) < \ell/C$.

It turns out that the two hypotheses are actually equivalent:

Proposition 13. *Hypothesis 1 and Hypothesis 2 are equivalent.*

Since the proof of their equivalence is just a simple observation, we defer them to Appendix C.2. With Hypothesis 2 in mind, we can now state (the properties of) the heart of our proof: the reduction from DκS to the problem of .

Lemma 14. *For some constants $C, B \geq 1$, there is a polynomial time algorithm that takes in an N-vertex graph G with bounded degree d and integers κ, ℓ, and produces a multiset of samples $S = \{(\mathbf{x}_i, y_i)\}_{i\in[m]} \subseteq \mathcal{B}^n \times [0,1]$ and two positive real numbers $\mathrm{OPT}, \epsilon \in \mathbb{R}^+$ such that*

- *(Completeness) If $\mathrm{den}_\kappa(G) \geq \ell$, there $\mathbf{w} \in \mathcal{B}^n$ such that $\mathcal{L}(\mathbf{w}; S) \leq \mathrm{OPT}$*
- *(Soundness) If $\mathrm{den}_{B\kappa}(G) < \ell/C$, then, for any $\mathbf{w} \in \mathcal{B}^n$, we have $\mathcal{L}(\mathbf{w}; S) > \mathrm{OPT} + \epsilon$.*
- *(Error bound) $\epsilon \geq \Omega_{d,C,B}\left(\frac{1}{\sqrt{N}}\right)$*

By plugging in appropriate parameters, it is simple to see that Lemma 14 implies Theorem 3.

Proof of Theorem 3. Suppose for the sake of contradiction that there is an $2^{o(1/\epsilon^2)}\mathrm{poly}(n)$-time algorithm $\mathbf{A}$ that solve the 1-ReLU Training problem to within an additive error of ϵ. We may solve the distinguishing problem in Hypothesis 2 as follows. Given an instance (G, κ, ℓ), we first apply the reduction in Lemma 14 to produce a multiset of samples $S = \{(\mathbf{x}_i, y_i)\}_{i\in[m]}$ where $\mathbf{x}_i \in \mathcal{B}^n$ and positive real numbers $\mathrm{OPT}, \epsilon = \Omega(1/\sqrt{N})$. We then run the algorithm $\mathbf{A}$ with accuracy ϵ to obtain a ReLU weight vector $\mathbf{w}$ with norm at most one. By checking whether $\mathcal{L}(\mathbf{w}; S) \leq \mathrm{OPT} + \epsilon$, we have distinguished the two cases in Hypothesis 2. Furthermore, our algorithm runs in time $2^{o(1/\epsilon^2)}\mathrm{poly}(n) = 2^{o(n)}\mathrm{poly}(n)$. Hence, this violates Hypothesis 2.

We conclude the proof by recalling that Hypotheses 1 and 2 are equivalent by Proposition 13. □

C.1 Reducing Densest k-Subgraph to Agnostic Learning of ReLUs

We proceed to the main technical contribution of this section, the proof of Lemma 14. The proof closely follows the intuition given in Section 2.

Proof of Lemma 14. We will give the reduction for $C, B = 1000$. Before we specify S, recall that we use N and M to denote the number of vertices and the number of edges of G respectively. Furthermore, let us define several additional parameters that will be used throughout:

- Let $\delta = \frac{1}{2\sqrt{\kappa}}$.
- Let $\gamma = \frac{1}{1000d}$ and $\zeta = \frac{1}{10^{10}d^2}$
- Let $\mathrm{OPT} = (1-\gamma)\zeta \cdot \left(1 - \frac{\kappa\delta}{\sqrt{2}N} + \frac{\kappa\delta^2}{8N}\right) + \gamma\zeta \cdot \left(1 - \frac{\ell\delta}{2\sqrt{2}M} + \frac{\ell\delta^2}{32M}\right)$.
- Let $\epsilon = \gamma\zeta \cdot \frac{\ell\delta}{4\sqrt{2}M} - (1-\gamma)\zeta \cdot \frac{\kappa\delta^2}{8N} - \gamma\zeta \cdot \frac{\ell\delta^2}{32M}$.

Now that, a priori, it may not be clear that ϵ is even positive. However, note that both of the terms $(1-\gamma)\zeta \cdot \frac{\kappa\delta^2}{8N}$ and $\gamma\zeta \cdot \frac{\ell\delta^2}{32M}$ are $O_d(\frac{1}{N})$. However, $\gamma\zeta \cdot \frac{\ell\delta}{4\sqrt{2}M} = \Omega_d(\frac{\ell}{\sqrt{\kappa}N})$. Now, notice that, we may assume w.l.o.g. that[14] $\ell \geq \kappa/3$ and that[15] $\kappa \geq \Omega(N)$. Hence, this positive term is at least $\Omega_d(\frac{1}{\sqrt{N}})$. In other words, for sufficiently large N, ϵ is positive and furthermore $\epsilon = \Omega_d(\frac{1}{\sqrt{N}})$ as desired.

[14]Note that, in the non-trivial case, it is always simple to find κ vertices that induce $\lfloor \kappa/2 \rfloor$ edges, by repeatedly adding one edge at a time. This bound is at least $\kappa/3$ for any $\kappa \geq 2$.

[15]In particular, if $\kappa = o(N)$, then the algorithm that enumerate all subsets of size κ already runs in time polynomial in $\binom{N}{\kappa} = 2^{o(N)}$.

We can now define the multiset of samples $S = \{(\mathbf{x}_i, y_i)\}_{i\in[m]}$ where $\mathbf{x}_i \in \mathcal{B}^n$ as follows. First, we let $n = N + 1$; we associate each of the first N coordinates by each vertex of G and we name the last coordinate $*$.

- We create $(10^{20} \cdot d^3NM) \cdot (1-\zeta)$ copies of the labelled sample $(\mathbf{e}_*, \frac{1}{\sqrt{2}})$ in S. This corresponds to the constraint

$$[\mathbf{w}_*]_+ = \frac{1}{\sqrt{2}}.$$

  We refer to this as the *constant constraint for* $*$.

- For each vertex $v \in V$, we add $(10^{20} \cdot d^3NM) \cdot \frac{(1-\gamma)\zeta}{N}$ copies of the sample $(\frac{1}{2}(\mathbf{e}_v - \delta\mathbf{e}_*), 1)$ to S. This corresponds to the constraint

$$\frac{1}{2}[\mathbf{w}_v - \delta\mathbf{w}_*]_+ = 1.$$

  This is referred to as the *cardinality constraint for* v.

- Finally, for each edge $e = \{u, v\} \in E$, we add $(10^{20} \cdot d^3NM) \cdot \frac{\gamma\zeta}{M}$ copies of the sample $(\frac{1}{2}(\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_*), 1)$ to S. This corresponds to the constraint

$$\frac{1}{2}[\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_*]_+ = 1.$$

  We refer to this as the *edge constraint for* e.

For notational convenience, let us separate the average square error $\mathcal{L}(\mathbf{w}; S)$ into three parts, based on the type of constraints. More specifically, we let

$$\begin{aligned}
\mathcal{L}^*(\mathbf{w}; S) &= (1-\zeta) \cdot \left(\frac{1}{\sqrt{2}} - [\mathbf{w}_*]_+\right)^2, \\
\mathcal{L}^{\text{card}}(\mathbf{w}; S) &= \frac{(1-\gamma)\zeta}{N} \cdot \left(\sum_{v\in V}\left(1 - \frac{1}{2}[\mathbf{w}_v - \delta\mathbf{w}_*]_+\right)^2\right), \text{ and} \\
\mathcal{L}^{\text{edge}}(\mathbf{w}; S) &= \frac{\gamma\zeta}{M} \cdot \left(\sum_{\{u,v\}\in E}\left(1 - \frac{1}{2}[\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_*]_+\right)^2\right).
\end{aligned}$$

By definition, we of course have $\mathcal{L}(\mathbf{w}; S) = \mathcal{L}^*(\mathbf{w}; S) + \mathcal{L}^{\text{card}}(\mathbf{w}; S) + \mathcal{L}^{\text{edge}}(\mathbf{w}; S)$. It will also be useful to expand out the term $\mathcal{L}^{\text{card}}(\mathbf{w}; S)$ and $\mathcal{L}^{\text{edge}}(\mathbf{w}; S)$ as follows:

$$\begin{aligned}
\frac{\mathcal{L}^{\text{card}}(\mathbf{w}; S)}{(1-\gamma)\zeta/N} &= N - \left(\sum_{v\in V}[\mathbf{w}_v - \delta\mathbf{w}_*]_+\right) + \frac{1}{4}\left(\sum_{v\in V}[\mathbf{w}_v - \delta\mathbf{w}_*]_+^2\right), \\
\frac{\mathcal{L}^{\text{edge}}(\mathbf{w}; S)}{\gamma\zeta/M} &= M - \left(\sum_{\{u,v\}\in E}[\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_*]_+\right) + \frac{1}{4}\left(\sum_{\{u,v\}\in E}[\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_*]_+^2\right).
\end{aligned}$$

(Completeness) Suppose that there exists a set $T \subseteq V$ of size k that induces at least ℓ edges. Then, we can set $\mathbf{w}_* = \frac{1}{\sqrt{2}}$, $\mathbf{w}_v$ to be $\delta\sqrt{2}$ iff $v \in T$ and zero otherwise. It is obvious to see that the $\|\mathbf{w}\|_2 = 1$ as desired. Moreover, we have $\mathcal{L}^*(\mathbf{w}; S) = 0$,

$$\mathcal{L}^{\text{card}}(\mathbf{w}; S) = (1-\gamma)\zeta \cdot \left(1 - \frac{\kappa\delta}{\sqrt{2}N} + \frac{\kappa\delta^2}{8N}\right),$$

and

$$\mathcal{L}^{\text{edge}}(\mathbf{w}; S) \leq \gamma\zeta \cdot \left(1 - \frac{\ell\delta}{2\sqrt{2}M} + \frac{\ell\delta^2}{32M}\right).$$

In total, we have $\mathcal{L}(\mathbf{w}; S) \leq \text{OPT}$ as desired.

(Soundness) Suppose for the sake of contradiction that $\text{den}_{B\kappa}(G) \leq \ell/C$ but there exists $\mathbf{w} \in \mathcal{B}^n$ such that $\mathcal{L}(\mathbf{w}; S) \leq \text{OPT} + \epsilon$. Let $\lambda_1 = \delta\mathbf{w}_*$ and $\lambda_2 = 2.5\delta\mathbf{w}_*$. We partition the set of vertices V into three sets:

- $V_{\geq\lambda_2} := \{v \in V \mid \mathbf{w}_v \geq \lambda_2\}$.
- $V_{(\lambda_1,\lambda_2)} := \{v \in V \mid \mathbf{w}_v \in (\lambda_1, \lambda_2)\}$.
- $V_{\leq\lambda_1} := \{v \in V \mid \mathbf{w}_v \leq \lambda_1\}$.

From this point on, we will write each edge $\{u, v\}$ in E as an ordered tuple (u, v) such that $\mathbf{w}_u \geq \mathbf{w}_v$ (tie broken arbitrarily). We can then partition the set of edges E into three parts:

- $E_{\geq\lambda_2} := \{(u, v) \in E \mid u \in V_{\geq\lambda_2}\}$.
- $E_{(\lambda_1,\lambda_2)} := \{(u, v) \in E \mid u, v \in V_{(\lambda_1,\lambda_2)}\}$.
- $E_{\leq\lambda_1} := \{(u, v) \in E \mid v \in V_{\leq\lambda_1} \wedge u \notin V_{\geq\lambda_2}\}$.

Observe that

$$\mathcal{L}^{\text{edge}}(\mathbf{w}; S) \geq \frac{\gamma\zeta}{M}\left(M - \sum_{(u,v)\in E}[\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_*]_+\right). \tag{12}$$

Let us now write $\sum_{(u,v)\in E}[\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_*]_+$ as

$$\sum_{(u,v)\in E_{\geq\lambda_2}}[\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_*]_+ + \sum_{(u,v)\in E_{(\lambda_1,\lambda_2)}}[\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_*]_+ + \sum_{(u,v)\in E_{\leq\lambda_1}}[\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_*]_+.$$

The last summation $\sum_{(u,v)\in E_{\leq\lambda_1}}[\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_*]_+$ is simply zero because, for all $(u, v) \in E_{\leq\lambda_1}$, we have $\mathbf{w}_u < \lambda_2$ and $\mathbf{w}_v \leq \lambda_1$, meaning that $\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_* < 0$.

Let us now consider the second term $\sum_{(u,v)\in E_{(\lambda_1,\lambda_2)}}[\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_*]_+$. Observe that $\mathcal{L}(\mathbf{w}; S) \leq \text{OPT} + \epsilon \leq 0.01$ implies that $\mathbf{w}_* \geq \frac{1}{3}$, which means that $\lambda_1 \geq \frac{\delta}{3} = \frac{1}{6\sqrt{\kappa}}$. Notice that $E_{(\lambda_1,\lambda_2)}$ is exactly the set of edges induced by $V_{(\lambda_1,\lambda_2)}$. Since $\lambda_1 \geq \frac{1}{6\sqrt{\kappa}}$, we must have $|V_{(\lambda_1,\lambda_2)}| < 36\kappa$

(because otherwise $\|\mathbf{w}\|_2 > 1$). As a result, from the assumption that $\mathrm{den}_{B\kappa}(G) \leq \ell/C$, we have $|E_{(\lambda_1,\lambda_2)}| < 0.001\ell$. Hence, we have

$$\sum_{(u,v)\in E_{(\lambda_1,\lambda_2)}} [\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_*]_+ < 0.001\ell \cdot 1.5\delta w_* < 0.01\ell \cdot \delta.$$

Finally, let us bound $\sum_{(u,v)\in E_{\geq\lambda_2}} [\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_*]_+$ as follows:

$$\begin{aligned}\sum_{(u,v)\in E_{\geq\lambda_2}} [\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_*]_+ &\leq \sum_{(u,v)\in E_{\geq\lambda_2}} ([2\mathbf{w}_u - 3.5\delta\mathbf{w}_*]_+) \\ &\leq \sum_{(u,v)\in E_{\geq\lambda_2}} 2[\mathbf{w}_u - \delta\mathbf{w}_*]_+ \\ &\leq 2d \sum_{u\in V_{\geq\lambda_2}} [\mathbf{w}_u - \delta\mathbf{w}_*]_+,\end{aligned}$$

where the last inequality follows from the fact that every vertex in graph G has degree at most d.

Combining the above two inequalities, we have

$$\sum_{(u,v)\in E} [\mathbf{w}_u + \mathbf{w}_v - 3.5\delta\mathbf{w}_*]_+ < 0.01\ell\delta + 2d \sum_{u\in V_{\geq\lambda_2}} [\mathbf{w}_u - \delta\mathbf{w}_*]_+.$$

Plugging the above inequality back into (12), we arrive at

$$\mathcal{L}^{\mathrm{edge}}(\mathbf{w}; S) > \frac{\gamma\zeta}{M}\left(M - 0.01\ell\delta - 2d \sum_{u\in V_{\geq\lambda_2}} [\mathbf{w}_u - \delta\mathbf{w}_*]_+\right). \tag{13}$$

Observe also that

$$\begin{aligned}\mathcal{L}^{\mathrm{card}}(\mathbf{w}; S) &\geq \frac{(1-\gamma)\zeta}{N}\left(N - \sum_{v\in V}[\mathbf{w}_v - \delta\mathbf{w}_*]_+\right) \\ &= \frac{(1-\gamma)\zeta}{N}\left(N - \sum_{v\in V_{\geq\lambda_2}} [\mathbf{w}_v - \delta\mathbf{w}_*]_+ - \sum_{v\in V_{(\lambda_1,\lambda_2)}} [\mathbf{w}_v - \delta\mathbf{w}_*]_+\right)\end{aligned} \tag{14}$$

By summing up (13) and (14), we have

$$
\begin{aligned}
&\mathcal{L}^{\text{edge}}(\mathbf{w};S)+\mathcal{L}^{\text{card}}(\mathbf{w};S)\\
&>\frac{\gamma\zeta}{M}\left(M-0.01\ell\delta-2d\sum_{u\in V_{\geq\lambda_2}}[\mathbf{w}_u-\delta\mathbf{w}_*]_+\right)\\
&\quad+\frac{(1-\gamma)\zeta}{N}\left(N-\sum_{v\in V_{\geq\lambda_2}}[\mathbf{w}_v-\delta\mathbf{w}_*]_+-\sum_{v\in V_{(\lambda_1,\lambda_2)}}[\mathbf{w}_v-\delta\mathbf{w}_*]_+\right)\\
&\geq\frac{\gamma\zeta}{M}\left(M-0.01\ell\delta\right)+\frac{(1-\gamma)\zeta}{N}\left(N-\left(1+\frac{2\gamma dN}{(1-\gamma)M}\right)\cdot\sum_{v\in V_{\geq\lambda_2}}[\mathbf{w}_v-\delta\mathbf{w}_*]_+-\sum_{v\in V_{(\lambda_1,\lambda_2)}}[\mathbf{w}_v-\delta\mathbf{w}_*]_+\right)\\
&\geq\frac{\gamma\zeta}{M}\left(M-0.01\ell\delta\right)+\frac{(1-\gamma)\zeta}{N}\left(N-1.01\sum_{v\in V_{\geq\lambda_2}}[\mathbf{w}_v-\delta\mathbf{w}_*]_+-\sum_{v\in V_{(\lambda_1,\lambda_2)}}[\mathbf{w}_v-\delta\mathbf{w}_*]_+\right),\qquad(15)
\end{aligned}
$$

where in the last inequality we use the fact that $\frac{2\gamma}{(1-\gamma)}<\frac{0.01}{d}$ which follows from our choice of γ.

Now, for each $v\in V_{(\lambda_1,\lambda_2)}$, the AM-GM inequality implies that

$$\mathbf{w}_v^2=(\delta\mathbf{w}_*+(\mathbf{w}_v-\delta\mathbf{w}_*))^2\geq 4\delta\mathbf{w}_*(\mathbf{w}_v-\delta\mathbf{w}_*)=4\delta\mathbf{w}_*[\mathbf{w}_v-\delta\mathbf{w}_*]_+.$$

Similarly, for each $v\in V_{\geq\lambda_2}$, the AM-GM inequality implies that

$$
\begin{aligned}
\mathbf{w}_v^2&=(1.25\delta\mathbf{w}_*+(\mathbf{w}_v-1.25\delta\mathbf{w}_*))^2\\
&\geq 5\delta\mathbf{w}_*(\mathbf{w}_v-1.25\delta\mathbf{w}_*)\\
&>4.1\delta\mathbf{w}_*(\mathbf{w}_v-\delta\mathbf{w}_*)\\
&\geq 4.1\delta\mathbf{w}_*[\mathbf{w}_v-\delta\mathbf{w}_*]_+,
\end{aligned}
$$

where the second-to-last inequality follows from $\mathbf{w}_v\geq\lambda_2\mathbf{w}_*=2.5\delta\mathbf{w}_*$, which implies that $(\mathbf{w}_v-1.25\delta\mathbf{w}_*)\geq\frac{1.25}{1.5}(\mathbf{w}_v-\delta\mathbf{w}_*)$.

Plugging the above two inequalities back into (15), we get

$$
\begin{aligned}
\mathcal{L}^{\text{edge}}(\mathbf{w};S)+\mathcal{L}^{\text{card}}(\mathbf{w};S)&>\frac{\gamma\zeta}{M}\left(M-0.01\ell\cdot\delta\right)+\frac{(1-\gamma)\zeta}{N}\left(N-\frac{1}{4\delta\mathbf{w}_*}\sum_{v\in V}\mathbf{w}_v^2\right)\\
&\geq\frac{\gamma\zeta}{M}\left(M-0.01\ell\cdot\delta\right)+\frac{(1-\gamma)\zeta}{N}\left(N-\frac{1}{4\delta\mathbf{w}_*}(1-\mathbf{w}_*^2)\right),
\end{aligned}
$$

where the second inequality follows from $\|\mathbf{w}\|_2\leq 1$.

Recall also that $\mathcal{L}^*(\mathbf{w};S)=(1-\zeta)\left(\frac{1}{\sqrt{2}}-\mathbf{w}_*\right)^2$. Adding this to above, we have

$$
\begin{aligned}
\mathcal{L}(\mathbf{w};S)&>\frac{\gamma\zeta}{M}\left(M-0.01\ell\cdot\delta\right)+\frac{(1-\gamma)\zeta}{N}\left(N-\frac{1}{4\delta\mathbf{w}_*}(1-\mathbf{w}_*^2)\right)+(1-\zeta)\left(\frac{1}{\sqrt{2}}-\mathbf{w}_*\right)^2\\
&=\frac{\gamma\zeta}{M}\left(M-0.01\ell\cdot\delta\right)+(1-\gamma)\zeta-\left(\frac{(1-\gamma)\zeta}{N}\cdot\frac{1}{4\delta\mathbf{w}_*}\cdot(1-\mathbf{w}_*^2)-(1-\zeta)\left(\frac{1}{\sqrt{2}}-\mathbf{w}_*\right)^2\right).\qquad(16)
\end{aligned}
$$

We will now bound the term

$$D(\mathbf{w}_*) := \frac{(1-\gamma)\zeta}{N} \cdot \frac{1}{4\delta \mathbf{w}_*} \cdot (1 - \mathbf{w}_*^2) - (1-\zeta)\left(\frac{1}{\sqrt{2}} - \mathbf{w}_*\right)^2.$$

In particular, we will show that $D(\mathbf{w}^*) < D(1/\sqrt{2}) + \frac{5\zeta^2}{N}$.

First, notice that, if $\mathbf{w}_* \geq \frac{1}{\sqrt{2}}$, then we immediately have $D(\mathbf{w}^*) \leq D(1/\sqrt{2})$, as the former is larger than the latter term-wise. Hence, we may only consider the case $\mathbf{w}_* < \frac{1}{\sqrt{2}}$. Let $\varphi_* = \frac{1}{\sqrt{2}} - \mathbf{w}_*$. We may write $D(\mathbf{w}^*) - D(1/\sqrt{2})$ as

$$\begin{aligned} D(\mathbf{w}_*) - D(1/\sqrt{2}) &= \frac{(1-\gamma)\zeta}{4\delta N}\left(\frac{\sqrt{2}\varphi_*}{\mathbf{w}_*} + \varphi_*\right) - (1-\zeta)\varphi_*^2 \\ &= \varphi_*\left(\frac{(1-\gamma)\zeta}{4\delta N}\left(\frac{\sqrt{2}}{\mathbf{w}_*} + 1\right) - (1-\zeta)\varphi_*\right) \end{aligned}$$

Recall that $\mathbf{w}_* \geq \frac{1}{3}$. As a result, we must have

$$\begin{aligned} D(w^*) - D(1/\sqrt{2}) &\leq \varphi_*\left(\frac{2(1-\gamma)\zeta}{\delta N} - (1-\zeta)\varphi_*\right) \\ &= \frac{1}{1-\zeta} \cdot ((1-\zeta)\varphi_*)\left(\frac{2(1-\gamma)\zeta}{\delta N} - (1-\zeta)\varphi_*\right) \\ \text{(AM-GM Inequality)} &\leq \frac{1}{1-\zeta}\left(\frac{(1-\gamma)\zeta}{\delta N}\right)^2 \\ &< \frac{5\zeta^2}{N}, \end{aligned}$$

where the last inequality follows from $\zeta = 0.99$ and $\delta = \frac{1}{2\sqrt{\kappa}} \geq \frac{1}{2\sqrt{N}}$. Thus, in both cases, we have $D(\mathbf{w}_*) < D(1/\sqrt{2}) + \frac{5\zeta^2}{N}$. Plugging this back into (16), we have

$$\begin{aligned} \mathcal{L}(\mathbf{w}; S) &> \frac{\gamma\zeta}{M}(M - 0.01\ell\delta) + (1-\gamma)\zeta - \frac{(1-\gamma)\zeta}{4\sqrt{2}\delta N} - \frac{5\zeta^2}{N} \\ &> \gamma\zeta\left(1 - \frac{\ell\delta}{4\sqrt{2}M}\right) + (1-\gamma)\zeta\left(1 - \frac{1}{4\sqrt{2}\delta N}\right) + \left(\frac{0.01\ell\delta\gamma\zeta}{M} - \frac{5\zeta^2}{N}\right) \\ &= OPT + \epsilon + \left(\frac{0.01\ell\delta\gamma\zeta}{M} - \frac{5\zeta^2}{N}\right) \\ &\geq OPT + \epsilon + \frac{0.01\zeta}{N}\left(\frac{\ell\delta\gamma}{d} - 500\zeta\right) \\ &\geq OPT + \epsilon + \frac{0.01\zeta}{N}\left(\frac{(\kappa/3) \cdot \frac{1}{2\sqrt{\kappa}} \cdot \gamma}{d} - 500\zeta\right) \\ &\geq OPT + \epsilon, \end{aligned}$$

where, in the second to last inequality, we assume w.l.o.g. that $\ell \geq \kappa/3$ and the last inequality follows from our choice of ζ and γ. This is a contradiction. □

C.2 Simplifying the Hypothesis: Proof of Proposition 13

The proof is a simple "trivial" reduction; the key observation here is that $\mathrm{den}_{B\kappa}(G)$ cannot be much larger than $\mathrm{den}_{\kappa}(G)$. Hence, by picking the constant C in Hypothesis 1 to be sufficiently large, we can arrive at a hypothesis of the form stated in Hypothesis 2.

Proof of Proposition 13. It is obvious that Hypothesis 2 implies Hypothesis 1, by simply plugging $B = 1$ into the former.

To prove the converse, for any $C, B > 1$, let $C' =$ and let $\delta = \delta(C')$ and $d = d(C')$ be as in Hypothesis 1. Now, we claim that, if $\mathrm{den}_{\kappa}(G) < \ell/C'$ for any graph G and any ℓ, then $\mathrm{den}_{B\kappa}(G) < \ell/C$. To see that this is the case, suppose contrapositively that $\mathrm{den}_{B\kappa}(G) \geq \ell/C$, i.e., there exists $T \subseteq V$ of size $B\kappa$ such that $|E(T)| > \ell/C$. Then, let us consider a random subset $T' \subseteq T$ of size k. We have

$$\mathbb{E}_{T'}[|E(T')|] = |E(T)| \cdot \left(\frac{\kappa(\kappa-1)}{B\kappa(B\kappa-1)}\right) \geq \frac{\ell}{C} \cdot \frac{1}{2B^2} = \frac{\ell}{C'}.$$

where we assume w.l.o.g. that $\kappa \geq 2$ in the inequality. This indeed implies that $\mathrm{den}_{\kappa}(G) \geq \ell/C'$.

The previous paragraph means that, if we can distinguish the two cases in Hypothesis 2 (with constants C, B), then we can also distinguish the two cases in Hypothesis 1 (with constant C'). As a result, if the former cannot be done in $O(2^{\delta N})$ time, then nor does the latter. In other words, Hypothesis 1 implies Hypothesis 2. □

D Hardness of Training (Non-negative) Sum of k ReLUs

In this section, we consider the bounded sum of k-ReLU Training problem. Recall that this is the restriction of the bounded k ReLU Training problem, in which we only allow the coefficient vector $\mathbf{a}$ to be the all-one vector $\mathbf{1}_k$; hence, here we are simply looking for a sum of k ReLUs, i.e. $\sum_{j\in[k]}[\mathbf{w}^j \cdot \mathbf{x}]_+$. We prove the NP-hardness of the bounded sum of k-ReLU Training, as stated more precisely below. We note here that a hardness of similar form was already obtained in [Vu98], except that there each ReLU is allowed to have a bias term. Hence, for completeness, we include the full proof of the following theorem later in this section.

Theorem 15. *For any constant $k \geq 2$, the bounded sum of k-ReLU Training problem is NP-hard.*

Furthermore, we show that a tight running time lower bound for the task of bounded k-ReLU Training to within an error of ϵ requires $2^{\Omega(1/\epsilon)}\mathrm{poly}(n, m)$ time, even in the realizable case. Our training algorithm in the realizable case (Theorem 5) can be adapted to only consider $\mathbf{a} = \mathbf{1}_k$, with the same running time. Hence, our running time lower bound here is essentially tight in terms of ϵ.

Our running time lower bound is based on the Gap Exponential Time Hypothesis (Gap-ETH) [Din16, MR17], which states that there is no $2^{o(n)}$-time algorithm that can distinguish between a satisfiable 3CNF formula and one which is not even $(1-\delta)$-satisfiable for some constant $\delta > 0$. (We remark that the lower bound can also be based on the weaker Exponential Time Hypothesis (ETH) [IP01, IPZ01], but the lower bound will only be of the form $2^{\Omega\left(\frac{1}{\epsilon \cdot \mathrm{poly}\log(1/\epsilon)}\right)}\mathrm{poly}(n, m)$.)

Theorem 16. *Assuming Gap-ETH, for any constant $k \geq 2$, there is no $2^{o(1/\epsilon)}$poly(n, m)-time algorithm that can solve the bounded sum of k-ReLU Training problem within an additive square error of ϵ, even in the realizable case.*

As the reader might have noticed, Theorems 15 and 16 are similar to Theorems 4 and 6, except that the latter are for the bounded k-ReLU Training (where $\mathbf{a}$ is not restricted to $\mathbf{1}_k$). Indeed, we will use Theorems 15 and 16 to prove Theorems 4 and 6 in the upcoming section.

Both Theorems 15 and 16 are based on a single reduction from the hypergraph k-coloring problem. Recall that, in the hypergraph k-coloring problem, we are given a hypergraph $G = (V, E)$ and the goal is to find a proper coloring $\chi : V \to [k]$. (A coloring χ is said to be proper if it does not result in any hyperedge e being monochromatic, i.e., $|\chi(e)| = 1$.) The main properties of the reduction is given in the lemma below. As mentioned earlier in Section 2, this reduction is in fact almost the same as that of [Vu98], except that the number of copies of each sample are different; this is needed in order to prove the tight running time lower bound (Theorem 16).

Lemma 17. *For any integer $k \geq 2$, there exists a polynomial time reduction that takes in an N-vertex hypergraph G whose edge size is at most t, and produces a multiset of samples $S = \{(\mathbf{x}_i, y_i)\}_{i\in[m]} \subseteq \mathcal{B}^n \times [0, 1]$ and a positive integer $\epsilon \in \mathbb{R}^+$ such that*

- *(Completeness) If G is k-colorable, then there exists $\mathbf{w}^1, \ldots, \mathbf{w}^k \in \mathcal{B}^n$ such that the samples are realizable by the sum of k ReLUs $\sum_{j\in[k]}[\mathbf{w}^j \cdot \mathbf{x}]_+$ (i.e. $\mathcal{L}(\mathbf{w}^1, \ldots, \mathbf{w}^k; S) = 0$).*
- *(Soundness) If every k-coloring of G results in γ fraction of edges being monochromatic for some $\gamma \in (0, 1)$, then $\mathcal{L}(\mathbf{w}^1, \ldots, \mathbf{w}^k; S) > \frac{\gamma}{100k^2t^5} \cdot \frac{1}{N}$ for all $\mathbf{w}^1, \ldots, \mathbf{w}^k \in \mathbb{R}^n$.*

Proof. Let V and E denote the set of vertices and the set of hyperedges of G respectively. Recall that we use N and M to denote $|V|$ and $|E|$ respectively. For convenience, let us rename the vertices as $1, 2, \ldots, n$.

Let $n = N$ and[16] $m = \sum_{i\in V} \deg_G(i) + |V|$. For each vertex $i \in V$, we create $\deg_G(i)$ copies of the sample $\mathbf{x}_i = \mathbf{e}_i$ and label them by $y_i = \frac{1}{t\sqrt{tn}}$; we refer to such samples as the *vertex i samples*. Moreover, for each hyperedge $e = \{i_1, \ldots, i_q\}$, we create a sample $(\mathbf{x}_e, y_e)$ with $\mathbf{x}_e = \frac{1}{\sqrt{t}} \sum_{j=1}^{q} \mathbf{e}_{i_j}$ and label it with $y_e = 0$; similarly, we refer to this as the *hyperedge e sample*. This completes our construction.

(Completeness) Suppose that the graph is k-colorable; let $\chi : V \to [k]$ be its proper k-coloring. We define $\mathbf{w}^1, \cdots, \mathbf{w}^k$ by $\mathbf{w}^a_i = \frac{1}{t\sqrt{n}}$ iff $\chi(i) = a$ and $-\frac{1}{\sqrt{n}}$ otherwise. Consider the sum of k ReLUs $\mathbf{x} \mapsto [\mathbf{w}^1 \cdot \mathbf{x}]_+ + \cdots + [\mathbf{w}^k \cdot \mathbf{x}]_+$. For each vertex i, we have

$$[\mathbf{w}^1 \cdot \mathbf{x}_i]_+ \cdots + [\mathbf{w}^k \cdot \mathbf{x}_i]_+ = [\mathbf{w}^1_i]_+ \cdots + [\mathbf{w}^k_i]_+ = \frac{1}{t\sqrt{tn}}.$$

Moreover, for each hyperedge $e = \{i_1, \cdots, i_q\}$, we have

$$[\mathbf{w}^1 \cdot \mathbf{x}_e]_+ + \cdots + [\mathbf{w}^k \cdot \mathbf{x}_e]_+ = \left[\frac{1}{\sqrt{t}} \sum_{j=1}^{q} \mathbf{w}^1_{i_j}\right]_+ + \cdots + \left[\frac{1}{\sqrt{t}} \sum_{j=1}^{q} \mathbf{w}^k_{i_j}\right]_+ = 0,$$

where the second equality follows from the fact that the edge e is not monochromatic. Hence, the samples are realizable by a sum of k ReLUs as desired.

(Soundness) Suppose contrapositively that for some constant $k \geq 2$, there exists a sum of k ReLUs $\mathbf{x} \mapsto \sum_{\ell\in[k]}[\mathbf{w}^\ell \cdot \mathbf{x}]_+$ such that $\mathcal{L}(\mathbf{w}^1, \ldots, \mathbf{w}^k; S) \leq \epsilon := \frac{\gamma}{100k^2t^5n}$.

[16] We use $\deg_G(i)$ to denote the number of hyperedges that i belongs to.

Let T denote the set of vertices $i \in V$ such that there exists $\ell_i \in [k]$ where $\mathbf{w}_i^{\ell_i} > \frac{1}{2k} \cdot \frac{1}{t\sqrt{tn}}$. Note that, for each $i \notin T$, we have $\sum_{\ell \in [k]} [\mathbf{w}^\ell \cdot \mathbf{x}_i]_+ = \sum_{\ell \in [k]} [\mathbf{w}_i^\ell]_+ \leq \frac{1}{2t\sqrt{tn}}$. In other words, we incur a square loss of at least $\frac{1}{4t^3 n}$ for each copy of the vertex i samples. This means that $\mathcal{L}(\mathbf{w}^1, \dots, \mathbf{w}^k; S) \geq \frac{\sum_{i \in (V \setminus T)} \deg_G(i)}{m} \cdot \frac{1}{4t^3 n}$. From our assumption, this can be at most ϵ, which gives

$$\sum_{i \in (V \setminus T)} \deg_G(i) \leq \epsilon \cdot 4t^3 \cdot m \cdot n \leq \frac{\gamma M}{2}, \tag{17}$$

where the latter comes from our choice of ϵ, and from $m \leq (t+1)M$.

Now, consider the coloring $\chi : V \to [k]$ where we assign $\chi(i) = \ell_i$ for all $i \in T$, and assign $\chi(i)$ arbitrarily for all $i \notin T$. From (17), the number of hyperedges that contain at least one vertex outside of T is at most $\gamma M/2$. Next, consider each hyperedge $e = \{i_1, \dots, i_q\}$ that is contained in T (i.e. $e \subseteq T$). e is monochromatic if and only if $\ell(i_1) = \cdots = \ell(i_q)$, which means that

$$\sum_{\ell \in [k]} [\mathbf{w}^\ell \cdot \mathbf{x}_e]_+ \geq [\mathbf{w}^{\ell(i_1)} \cdot \mathbf{x}_e]_+ = \left[\frac{1}{\sqrt{t}} \left(\mathbf{w}_{i_1}^{\ell(i_1)} + \mathbf{w}_{i_2}^{\ell(i_2)} + \cdots + \mathbf{w}_{i_q}^{\ell(i_q)} \right) \right]_+ > \frac{1}{2kt^2\sqrt{n}},$$

where the last inequality comes from $\mathbf{w}_i^{\ell(i)} > \frac{1}{2kt\sqrt{tn}}$ for all $i \in S$. In other words, each monochromatic hyperedge e contained in T incurs a square loss of more than $\frac{1}{4k^2 t^4 n}$ in the hyperedge e sample. As a result, the number of such hyperedges is less than

$$\epsilon \cdot (4k^2 t^4 n) \cdot m \leq \frac{\gamma M}{2}.$$

As a result, in total, the number of monochromatic hyperedges for the coloring χ is less than γM. This concludes our proof. □

With the above reduction ready, Theorem 15 and Theorem 16 follow easily from known results on hardness of coloring. For Theorem 15, we may use the (classic) NP-hardness of coloring:

Theorem 18 ([Lov73, Sto73])**.** *For any $k \geq 2$, deciding whether a given hypergraph G is k-colorable is NP-hard. Furthermore, this holds even when G has maximum edge size at most 3.*

Proof of Theorem 15. We reduce from hardness of coloring in Theorem 18. Let $G = (V, E)$ be the input hypergraph whose hyperedges are of size at most 3. By applying the reduction from Lemma 17, we get a set of samples $\{(\mathbf{x}_i, y_i)\}_{i \in [m]}$. Lemma 17 guarantees that, if the graph is k-colorable, then there is a sum of k ReLUs $\sum_{j \in [k]} [\mathbf{w}^j \cdot \mathbf{x}]_+$ with $\mathbf{w}^1, \dots, \mathbf{w}^k \in \mathcal{B}^n$ that realizes these samples. On the other hand, if G is not k-colorable, then the soundness guarantee of Lemma 17 implies that any sum of k ReLUs incurs an average square error of at least $\frac{1/M}{100 \cdot k^2 \cdot 3^5} \cdot \frac{1}{N} = \frac{1}{24300 k^2 NM}$. Hence, the bounded sum of k ReLU Training problem is also NP-hard. □

Remark 1. *If we plug in hardness of approximate coloring (e.g. [DRS05, Bha18]) to our reduction instead of the hardness of exact coloring in Theorem 18, then we can actually get a stronger soundness where the constructed samples are not realizable even for any sum of k' ReLUs for any constant $k' > k$. In fact, using the hardness of approximation of coloring in [Bha18], k' can even be taken as large as $(\log n)^{1-o(1)}$.*

We note, however, that such strong soundness does not hold for the problem of bounded k-ReLU Training (with possibly negative coefficient). In particular, our gadget in Lemma 21 has a soundness guarantee that only holds against k-ReLU, but not even $(k+1)$-ReLU. It remains an interesting open question to extend such stronger soundness to this case as well.

We now move on to prove our running time lower bound (Theorem 16). To prove this result, we will use the following running time lower bound, which is explicit in [Pet94]:

Theorem 19 ([Pet94])**.** *Assuming Gap-ETH, for any $k \geq 2$, there exists $\gamma > 0$ such that the following holds. There is no $2^{o(N)}$ time algorithm that can, given an N-vertex $(k+1)$-uniform hypergraph G, distinguish between the following two cases:*

- *(Completeness) G is k-colorable.*
- *(Soudness) Any k-coloring of G violates more than γ fraction of its hyperedges.*

We remark that, strictly speaking, Petrank only proved the above theorem in the case of $k = 2$, for which the problem of 2-coloring 3-uniform hypergraph is equivalent to the so-called Max NAE 3SAT, which was proved to be hard to approximate in [Pet94, Theorem 4.3]; the reduction is a linear time reduction from the gap version of 3-SAT, which yields the running time lower bound we stated above. Nonetheless, it is also very simple to generalize the result to the case $k \geq 2$. We sketch the argument in Appendix D.1.

Proof of Theorem 16. We reduce from hardness of coloring in Theorem 18. Let $G = (V, E)$ be a $(k+1)$-uniform hypergraph. By applying the reduction from Lemma 17, we get a set of examples $\{(\mathbf{x}_i, y_i)\}_{i\in[m]}$. Lemma 17 guarantees that, if the graph is k-colorable, then there is a sum of k ReLUs $\sum_{j\in[k]}[\mathbf{w}^j \cdot \mathbf{x}]_+$ where $\mathbf{w}^1, \ldots, \mathbf{w}^k \in \mathcal{B}^n$ that realizes these samples. On the other hand, if any k-colorable of G results in at least γ fraction of the hyperedges being monochromatic, then the soundness guarantee of Lemma 17 implies that any sum of k ReLUs incurs an average square error of more than $\epsilon := \frac{\gamma}{100k^2t^5} \cdot \frac{1}{N} = \Omega_{\gamma,k}\left(\frac{1}{N}\right)$.

Hence, if there is a $2^{o(1/\epsilon)}\mathrm{poly}(n)$-time learning algorithm for 2-ReLUs (in the realizable case) to within square error of ϵ, then we can distinguish the two cases in Theorem 19 in time $2^{o(N)}$ time. By Theorem 19, this violates Gap-ETH. □

D.1 On hardness of coloring

In this section, we briefly sketch the proof of Theorem 18. First, Theorem 4.3 of [Pet94] immediately implies the following.

Theorem 20 ([Pet94])**.** *Assuming Gap-ETH, there exists $\gamma > 0$ such that the following holds. There is no $2^{o(N)}$ time algorithm that can, given an N-vertex 3-uniform hypergraph G, distinguish between the following two cases:*

- *(Completeness) G is 2-colorable.*
- *(Soudness) Any 2-coloring of G violates more than γ fraction of its hyperedges.*

We may now prove Theorem 18 as follows.

Proof Sketch of Theorem 15. Given an N-vertex 3-uniform input hypergraph $G = (V, E)$ from Theorem 20. We create a new hypergraph $G' = (V', E')$ where V' is simply V together with $k-2$ additional "dummy vertices" (i.e. $V' = V \cup \{u_1, \ldots, u_{k-2}\}$). We then let $E' = \{e \cup \{u_1, \ldots, u_{k-2}\} \mid e \in E\}$. It is simple to check that, for any $\nu \in [0, 1]$, there exists a k-coloring of G' with ν fraction of edges being monochromatic if and only if there exists a 2-coloring of G with ν fraction of edges being monochromatic. □

E Handling Negative Coefficients: Hardness of k-ReLU Training

In this section, we show that our hardness from the previous section can be easily extended to the case where the coefficients in front of each ReLU unit is allowed to be negative. Specifically, we will prove Theorems 4 and 6 here.

The main gadget used to translate our results from the non-negative coefficient case to the more generalized case here is just a set of points that can be realized by a weighted sum of k ReLUs only when all the coefficients are positive, as stated below.

Lemma 21 (Main Gadget)**.** *For any $k \in \mathbb{N}$, there exists a set of samples $\tilde{S} = \{(\tilde{\mathbf{x}}_i, \tilde{y}_i)\}_{i \in [\tilde{m}]} \subseteq \mathcal{B}^k \times [0, k]$ and a positive real number $\tau \in \mathbb{R}^+$ such that*

- *(Completeness) The samples can be realized by $\sum_{j \in [c]} [\tilde{\mathbf{w}}^j \cdot \tilde{\mathbf{x}}]_+$ for some $\tilde{\mathbf{w}}^1, \ldots, \tilde{\mathbf{w}}^k \in \mathcal{B}^k$.*
- *(Soundness) For any $\tilde{\mathbf{w}}^1, \ldots, \tilde{\mathbf{w}}^k \in \mathbb{R}^k$, $\mathbf{a} \in \{-1, 1\}^n \setminus \{\mathbf{1}_k\}$, $\mathcal{L}(\tilde{\mathbf{w}}^1, \ldots, \tilde{\mathbf{w}}^k, \mathbf{a}; \tilde{S}) \geq \tau$.*

Moreover, the set $\tilde{S}$ can be constructed in time $2^{O(k)}$.

We will construct our gadget in the above lemma in Section E.1. Before we do so, let us use it to prove Theorem 4. The main idea is simple: we start from the NP-hard instance from the non-negative weights case and extend the dimension by k (where we simply add k zeros to the end of each sample). Then, we construct additional samples using the gadget (Lemma 21); the gadget samples are embedded in the last k coordinates and the remaining coordinates are just zeros. The key observation here is that, if a weighted sum of k ReLUs with negative weights is used, then it must incur the error from the soundness of Lemma 21. Otherwise, we are back to the case where all coefficients are non-negative, for which we already know the hardness.

At this point, we would also like to remark that, while in all our proofs we only consider constant $k \geq 2$, we can in fact take k to be as large as $O(\log n)$ for Theorem 4. The bottleneck here is the construction time $2^{O(k)}$ of the gadget in Lemma 21; since we want this to be polynomial time, we can take k to be at most $O(\log n)$.

Proof of Theorem 4. Let $S = \{(\mathbf{x}_i, y_i)\}_{i \in [m]}$ where $\mathbf{x}_i \in \{0, 1\}^n$ be the NP-hard instance from Theorem 15, and let $\tilde{S} = \{(\tilde{\mathbf{x}}_i, \tilde{y}_i)\}_{i \in [\tilde{m}]}$ where $\mathbf{x}_i \in \mathcal{B}^k$ be the samples from Lemma 21. We construct the multiset of new samples $\hat{S} = \{(\hat{\mathbf{x}}_i, \hat{y}_i)\}_{i \in [\hat{m}]}$ where $\hat{\mathbf{x}}_i \in \mathcal{B}^{\hat{n}}$ as follows.

- Let $\hat{n} = n + k$ and $\hat{m} = m + \tilde{m}$.
- For every $i \in [m]$, we add $\tilde{m}$ copies of the labelled sample $(\mathbf{x}_i \circ \mathbf{0}_k, 0.5 y_i)$ in $\hat{S}$.
- For every $i \in [\tilde{m}]$, we add m copies of the labelled sample $(\mathbf{0}_n \circ \tilde{\mathbf{x}}_i, 0.5 \tilde{y}_i)$ in $\hat{S}$.

(Completeness) Suppose that $\{(\mathbf{x}_i, y_i)\}_{i\in[m]}$ are realizable by a sum of k-ReLUs $\sum_{j\in[k]}[\mathbf{w}^j \cdot \mathbf{x}]_+$ where $\mathbf{w}^1, \ldots, \mathbf{w}^k \in \mathcal{B}^n$. Recall also from Lemma 21 that the samples $\{(\tilde{\mathbf{x}}_i, y_i)\}_{i\in[\tilde{m}]}$ can be realized by a sum of k-ReLUs $\sum_{j\in[k]}[\tilde{\mathbf{w}}^j \cdot \tilde{\mathbf{x}}]_+$ where $\tilde{\mathbf{w}}^1, \ldots, \tilde{\mathbf{w}}^k \in \mathcal{B}^k$. For every $j \in [k]$, let $\hat{\mathbf{w}}^j = 0.5\mathbf{w}^j \circ 0.5\tilde{\mathbf{w}}^j$. It is easy to check that $\|\hat{\mathbf{w}}^1\|, \ldots, \|\hat{\mathbf{w}}^k\| \leq 1$ and that the constructed samples can be realized by the k-ReLU $\sum_{j\in[k]}[\hat{\mathbf{w}}^j \cdot \hat{\mathbf{x}}]_+$.

(Soundness) Suppose that any sum of k-ReLUs incurs an average loss of at least $\frac{1}{\mathrm{poly}(n)}$ on the samples $\{(\mathbf{x}_i, y_i)\}_{i\in[m]}$. We will show that any weighted sum of k-ReLUs $\sum_{j\in[k]} a_j[\hat{\mathbf{w}}_j \cdot \hat{\mathbf{x}}]_+$ incurs a loss of at least $\frac{1}{\mathrm{poly}(\tilde{n})}$ on the constructed samples $\{(\hat{\mathbf{x}}_i, \hat{y}_i)\}_{i\in[\hat{m}]}$. Consider the following two cases:

- $a_1 = \cdots = a_k = +1$. In this case, the assumption on $\{(\mathbf{x}_i, y_i)\}_{i\in[m]}$ implies that the weighted sum of k-ReLUs incurs an average loss of at least $\frac{1}{\mathrm{poly}(n)}$ on the samples $\{(\mathbf{x}_i \circ \mathbf{0}_{\tilde{m}}, 0.5y_i)\}_{i\in[m]}$. Since (the copies of) these samples contribute to half of the total number of constructed samples, we have that the weighted sum of k-ReLUs incurs an average loss of at least $\frac{1}{2} \cdot \frac{1}{\mathrm{poly}(n)} \geq \frac{1}{\mathrm{poly}(\tilde{n})}$ on the constructed samples $\{(\hat{\mathbf{x}}_i, \hat{y}_i)\}_{i\in[\hat{m}]}$.
- $a_j = -1$ for some $j \in [k]$. In this case, the soundness of Lemma 21 implies that the weighted sum of k-ReLUs incurs an average loss of at least τ on the samples $\{(\mathbf{0}_m \circ \tilde{\mathbf{x}}_i, 0.5\tilde{y}_i)\}_{i\in[\tilde{m}]}$. Since (the copies of) these samples contribute to half of the total number of constructed samples, we have that the weighted sum of k-ReLUs incurs an average loss of at least 0.5τ on the constructed samples $\{(\hat{\mathbf{x}}_i, \hat{y}_i)\}_{i\in[\hat{m}]}$. Finally, recall that τ is a positive constant that depends only on c; hence, $0.5\tau \geq 1/\mathrm{poly}(\tilde{n})$.

□

We remark that it is straightforward to see that, when plugging the above reduction into Theorem 16, we also immediately get Theorem 6. We omit the full proof here, but note that the main observation is that, the error incurred in the second case ($a_j = -1$ for some $j \in [k]$) is an absolute constant that only depends on k. This means that, for any sufficiently small ϵ, we are forced to use $a_1 = \cdots = a_k = 1$, which takes us right back to Theorem 16.

E.1 Construction of the Gadget

We now move on to the construction of our main gadget (Lemma 21). Before we state the proof for general k, let us first note that the case for $k = 2$ is incredibly simple: just create two samples $(\mathbf{u}, 1)$ and $(-\mathbf{u}, 1)$ where $\mathbf{u}$ can be any unit vector. (This construction is in fact the same as a gadget used in [BJW19].) These samples can be realized by the sum of 2 ReLUs $[\tilde{\mathbf{w}}^1 \cdot \tilde{\mathbf{x}}]_+ + [\tilde{\mathbf{w}}^2 \cdot \tilde{\mathbf{x}}]_+$ where $\tilde{\mathbf{w}}_1 = \mathbf{u}$ and $\tilde{\mathbf{w}}_2 = -\mathbf{u}$. To see that they cannot be realized by weighted sum of 2 ReLUs with a negative coefficient $[\tilde{\mathbf{w}}_1 \cdot \tilde{\mathbf{x}}]_+ - [\tilde{\mathbf{w}}_2 \cdot \tilde{\mathbf{x}}]_+$, observe that $\tilde{\mathbf{w}}_1 \cdot \mathbf{u}$ or $\tilde{\mathbf{w}}_1 \cdot (-\mathbf{u})$ must be non-positive, meaning that it must output a non-positive value on at least one of $\mathbf{u}$ or $-\mathbf{u}$. Hence, the samples are not realizable by such a weighted sum of 2 ReLUs.

The above example is a special case of a more general phenomenon: if we look at the "sign pattern" (i.e. whether $\tilde{\mathbf{w}}^1 \cdot \tilde{\mathbf{x}}, \ldots, \tilde{\mathbf{w}}^k \cdot \tilde{\mathbf{x}}$ are positives), there can be as many as 2^k such patterns; $2^k - 1$ such patterns may result in a positive output in the (positive) sum of c ReLUs, with the only exception being when $\tilde{\mathbf{w}}_1 \cdot \tilde{\mathbf{x}}, \ldots, \tilde{\mathbf{w}}_c \cdot \tilde{\mathbf{x}} < 0$. However, if we look at the sign patterns for a weighted sum of k ReLUs with at least one negative coefficient, then only at most $2^k - 2$ patterns

can result in positive outputs. (For instance, if the coefficient of $[\tilde{\mathbf{w}}^k \cdot \tilde{\mathbf{x}}]_+$ is negative, then the sign pattern $\tilde{\mathbf{w}}^1 \cdot \tilde{\mathbf{x}}, \ldots, \tilde{\mathbf{w}}^{k-1} \cdot \tilde{\mathbf{x}} < 0$ and $\tilde{\mathbf{w}}_k \cdot \tilde{\mathbf{x}} \geq 0$ cannot result in a positive output.) Although we do not use these bounds directly in the above samples for $k = 2$, we do use the fact that $\mathbf{u}$ and $-\mathbf{u}$ cannot correspond to the same sign pattern. Roughly speaking, our soundness proof for the general case below also proceeds by arguing that, since there are fewer sign patterns that result in positive outputs when there is a negative coefficient, pigeonhole principle implies that some samples that should not be from the same sign pattern must be from the same sign pattern when there is a negative coefficient, which would lead to a large square error similar to the case $k = 2$ above.

To formalize our construction, recall that a set of vectors in $\mathbb{R}^d$ is said to be in *general position* if any d of these vectors are linearly independent. We use $\mathcal{B}(\mathbf{x}, r) := \{\mathbf{x}' \mid \|\mathbf{x} - \mathbf{x}'\| \leq r\}$ to denote the ball of radius r around $\mathbf{x}$. It is well known that for any $d, t \in \mathbb{N}$, $\mathbf{x} \in \mathbb{R}^d$ and $r \in \mathbb{R}^+$, it is possible to construct a set of t vectors in $\mathcal{B}(\mathbf{x}, r)$ that are in general position in $poly(d, t)$ time.

Proof of Lemma 21. Let $f(\tilde{\mathbf{x}}) = [\tilde{x}_1]_+ + \cdots + [\tilde{x}_k]_+$ be the sum of k-ReLUs, in which the j-th ReLU weight vector $\tilde{\mathbf{w}}_j$ is just the j-th standard basis vector. We construct our samples as follows:

- For every $\mathbf{u} \in \{-1, +1\}^k \setminus \{-\mathbf{1}_k\}$, let $S_{\mathbf{u}} \subseteq \mathcal{B}(\frac{\mathbf{u}}{2\sqrt{k}}, \frac{0.01}{k})$ be the set of any $2^k \cdot k$ points in general position. (As stated before the proof, this can be constructed in $2^{O(k)}$ time.)
- Let $S := \bigcup_{\mathbf{u} \in \{-1,+1\}^k \setminus \{-\mathbf{1}_k\}} S_{\mathbf{u}}$. The samples are $(\tilde{\mathbf{x}}, f(\tilde{\mathbf{x}}))$ for all $\tilde{\mathbf{x}} \in S$.

Before we prove the completeness and soundness of the gadget, we first define the real number τ that will be used in the soundness. We let

$$\tau = \frac{0.1}{k|S|} \cdot \min \left\{ 1, \min_{\substack{\mathbf{u} \in \{-1,+1\}^n \setminus \{-\mathbf{1}_k\} \\ S'_{\mathbf{u}} \subseteq S_{\mathbf{u}}, |S'_{\mathbf{u}}| = k}} \inf_{\|\mathbf{w}\|=1} \sum_{\mathbf{x} \in S'_{\mathbf{u}}} \|\mathbf{w} \cdot \mathbf{x}\|^2 \right\}.$$

A priori, it might not be clear that τ has to be positive. To see that this is the case, observe that $\inf_{\|\mathbf{w}\|=1} \sum_{\mathbf{x} \in S'_{\mathbf{u}}} \|\mathbf{w} \cdot \mathbf{x}\|^2$ is exactly equal to $\min_{j=1,\ldots,k} \lambda_j^2$ where λ_j is the j-th eigenvalue of the $(k \times k)$-matrix whose rows are $\mathbf{x} \in S'_{\mathbf{u}}$. Since the vectors in $S_{\mathbf{u}}$ are in general position, this matrix must be full rank, which implies that all its eigenvalues are non-zero. As a result, we have $\inf_{\|\mathbf{w}\|=1} \sum_{\mathbf{x} \in S'_{\mathbf{u}}} \|\mathbf{w} \cdot \mathbf{x}\|^2 > 0$, which in turn implies that $\tau > 0$ as desired. (Note here that τ only depends on k and our choices of $\{S_{\mathbf{u}}\}_{\mathbf{u}}$.)

(Completeness) By construction, the samples are realizable by f, which is a sum of k ReLUs.

(Soundness) We now consider any weighted sum of k ReLUs, such that at least one of the coefficient is negative. Without loss of generality, we may assume that this is a function of the form $g(\tilde{\mathbf{x}}) = \sum_{j=1}^{k'} [\tilde{\mathbf{w}}^j \cdot \tilde{\mathbf{x}}]_+ - \sum_{j=k'+1}^{k} [\tilde{\mathbf{w}}^j \cdot \tilde{\mathbf{x}}]_+$ for some non-negative integer $k' < k$. We will show that g incurs an average loss of at least some positive constant. To do so, we define the following notations: let $\mathrm{sgn}(z) = +1$ if $z > 0$ and 0 otherwise. For every "sign pattern" $\mathbf{s} \in \{0, 1\}^k$, let $P_{\mathbf{s}} \subseteq \mathbb{R}^k$ denote the subsets of points $\tilde{\mathbf{x}} \in \mathbb{R}^k$ such that $\mathrm{sgn}(\tilde{\mathbf{w}}^j \cdot \tilde{\mathbf{x}}) = s_j$ for all $j \in [k]$. Now, consider the following two cases:

- Case I: $(P_{(0,0,\ldots,0,0)} \cup P_{(0,0,\ldots,0,1)}) \cap S \neq \emptyset$. Let $\tilde{\mathbf{x}}$ be any element of $(P_{(0,0,\ldots,0,0)} \cup P_{(0,0,\ldots,0,1)}) \cap S$. From $\tilde{\mathbf{x}} \in S$, it is simple to check that[17] $f(\tilde{\mathbf{x}}) \geq \frac{0.4}{\sqrt{k}}$. However, from $\tilde{\mathbf{x}} \in P_{(0,0,\ldots,0,0)} \cup P_{(0,0,\ldots,0,1)}$,

[17] In particular, if $\tilde{\mathbf{x}} \in S_{\mathbf{u}}$ and $u_i = 1$, then we must have $\tilde{x}_i \geq 0.4/\sqrt{k}$, which implies that $f(\tilde{\mathbf{x}}) \geq 0.4/\sqrt{k}$.

we have $g(\tilde{\mathbf{x}}) \leq 0$. Hence, g must incur an average square loss of at least $\frac{1}{|S|} \cdot \frac{0.16}{k}$, which is at least τ by the definition of the latter.

- Case II: $(P_{(0,0,\ldots,0,0)} \cup P_{(0,0,\ldots,0,1)}) \cap S = \emptyset$.

  Fix any $\mathbf{u} \in \{-1,+1\}^k \setminus \{-\mathbf{1}_k\}$, since $|S_{\mathbf{u}}| = 2^k \cdot k$ and $\cup_{\mathbf{s} \in \{0,1\}^k} P_{\mathbf{s}} = \mathbb{R}^k$, there must exists a sign pattern $\mathbf{s}_{\mathbf{u}}$ such that $|S_{\mathbf{u}} \cap P_{\mathbf{s}_{\mathbf{u}}}| \geq k$. From the assumption of this case, we must also have that $\mathbf{s}_{\mathbf{u}} \neq (0,0,\ldots,0), (0,0,\ldots,0,1)$.

  As a result, by pigeonhole principle[18], there exists two distinct $\mathbf{u}^1, \mathbf{u}^2 \in \{-1,+1\}^k \setminus \{-\mathbf{1}_k\}$ such that $\mathbf{s}_{\mathbf{u}^1} = \mathbf{s}_{\mathbf{u}^2}$. Let $\mathbf{s}^* = \mathbf{s}_{\mathbf{u}^1} = \mathbf{s}_{\mathbf{u}^2}$; we have that $|P_{\mathbf{s}^*} \cap S_{\mathbf{u}^1}|, |P_{\mathbf{s}^*} \cap S_{\mathbf{u}^2}| \geq k$.

  Furthermore, observe that, for all $\tilde{\mathbf{x}} \in P_{\mathbf{s}^*}$, we have

$$\begin{aligned} g(\tilde{\mathbf{x}}) &= \sum_{j=1}^{k'} s_j^* \cdot \tilde{\mathbf{w}}^j \cdot \tilde{\mathbf{x}} - \sum_{j=k'+1}^{k} s_j^* \cdot \tilde{\mathbf{w}}^j \cdot \tilde{\mathbf{x}} \\ &= \left(\sum_{j=1}^{k'} s_j^* \cdot \tilde{\mathbf{w}}^j - \sum_{j=k'+1}^{k} s_j^* \cdot \tilde{\mathbf{w}}^j \right) \cdot \tilde{\mathbf{x}} \\ &= \tilde{\mathbf{w}}^* \cdot \tilde{\mathbf{x}}, \end{aligned}$$

  where we define $\tilde{\mathbf{w}}^* = \left(\sum_{j=1}^{k'} s_j^* \cdot \tilde{\mathbf{w}}^j - \sum_{j=k'+1}^{k} s_j^* \cdot \mathbf{w}^j \right)$. As a result, the average square error incurred by g on the constructed samples is at least

$$\frac{1}{|S|} \left(\sum_{\mathbf{x} \in P_{\mathbf{s}^*} \cap (S_{\mathbf{u}^1} \cup S_{\mathbf{u}^2})} (f(\tilde{\mathbf{x}}) - g(\tilde{\mathbf{x}}))^2 \right) = \frac{1}{|S|} \sum_{\mathbf{x} \in P_{\mathbf{s}^*} \cap (S_{\mathbf{u}^1} \cup S_{\mathbf{u}^2})} (f(\tilde{\mathbf{x}}) - \mathbf{w}^* \cdot \tilde{\mathbf{x}}))^2. \tag{18}$$

  Next, observe that, for all $\tilde{\mathbf{x}} \in S_{\mathbf{u}^1}$, we have $f(\tilde{\mathbf{x}}) = \operatorname{sgn}(\mathbf{u}^1) \cdot \mathbf{x}$ where define $\operatorname{sgn}(\mathbf{u})$ as $(\operatorname{sgn}(u_j))_{j \in [k]}$. Similarly, for all $\tilde{\mathbf{x}} \in S_{\mathbf{u}^2}$, we have $f(\tilde{\mathbf{x}}) = \operatorname{sgn}(\mathbf{u}^2) \cdot \tilde{\mathbf{x}}$. Moreover, since $\mathbf{u}^1 \neq \mathbf{u}^2$ both belong to $\{\pm 1\}^k$, we must have $\|\operatorname{sgn}(\mathbf{u}^1) - \operatorname{sgn}(\mathbf{u}^2)\|_2 \geq 2$. From this, we must have either $\|\mathbf{w}^* - \operatorname{sgn}(\mathbf{u}^1)\| \geq 1$ or $\|\mathbf{w}^* - \operatorname{sgn}(\mathbf{u}^2)\| \geq 1$; without loss of generality, we assume the former. We may lower bound the right hand side term in (18) by

$$\begin{aligned} \frac{1}{|S|} \sum_{\mathbf{x} \in P_{\mathbf{s}^*} \cap S_{\mathbf{u}^1}} \|(\operatorname{sgn}(\mathbf{u}^1) - \tilde{\mathbf{w}}^*) \cdot \tilde{\mathbf{x}}\|^2 &\geq \frac{1}{|S|} \inf_{\|\mathbf{w}\|=1} \sum_{\tilde{\mathbf{x}} \in P_{\mathbf{s}^*} \cap S_{\mathbf{u}^1}} \|\mathbf{w} \cdot \tilde{\mathbf{x}}\|^2 \\ &\geq \tau, \end{aligned}$$

  where the first inequality follows from $\|\operatorname{sgn}(\mathbf{u}^1) - \mathbf{w}^*\| \geq 1$ and the last inequality follows from $|P_{\mathbf{s}^*} \cap S_{\mathbf{u}^1}| \geq k$ and from our definition of τ (especially the second term with $\mathbf{u} = \mathbf{u}^1$).

Hence, in both cases, we have that the average squared error incurred by g must be at least τ, which concludes our proof. □

[18] Note that there are $2^k - 1$ pigeons and only $2^k - 2$ holes.

F Training and learning algorithms

In this Section, we describe algorithms for learning and training ReLUs. We begin in Section F.1 by giving a simple training algorithm, whose running time is ostensibly super-polynomial. Then, in Section F.1, we define the learning problem and show, using standard generalization arguments, that in fact the aforementioned training algorithm gives the claimed running time lower bound (in Theorems 3 and 6).

F.1 A Simple Training Algorithm

In light of the NP-hardness from the previous sections, a polynomial time algorithm for training depth-2 ReLUs do not exist (unless P = NP). Nevertheless, it is still possible to train the ReLUs in super polynomial time. For instance, [ABMM18] gives a simple algorithm that runs in time $n^{O(km)}$ and output the optimal training error (to within arbitrarily small accuracy). Below, we observe that their approach also yields an $2^{km} \cdot poly(n, m, k)$ time algorithm. Before we proceed to the statement and the proof of the algorithm, we remark that, the NP-hardness proof from Section D in fact implies that, assuming the Exponential Time Hypothesis (ETH) [IP01, IPZ01][19], the bounded k-ReLU training problem cannot be solved exactly in $2^{o(m)}$ time for any constant $k \geq 2$. Hence, the dependency m in the exponent is tight in this sense. However, it is unclear whether the dependency on k can be improved.

Lemma 22. *There is an $2^{k(1+m)} \cdot poly(n, m, 1/\delta, C)$-time algorithm that, given samples $\{(\mathbf{x}_i, y_i)\}_{i \in [m]}$ where $\mathbf{x}_i \in \mathbb{R}^n$ and an accuracy parameter $\delta \in (0, 1)$, finds the weights $\mathbf{w}_1, \ldots, \mathbf{w}_k \in \mathcal{B}^n, \mathbf{a} \in \{-1, 1\}^k$ that minimizes $\mathcal{L}(\mathbf{w}^1, \ldots, \mathbf{w}^k, \mathbf{a}; S)$ up to an additive error of δ. We assume the bit complexity of every number appearing in the coordinates of the $\mathbf{x}_i$'s and y_i's is at most C.*

Proof. First, we iterate over all possible $\mathbf{a} \in \{-1, 1\}^k$. Moreover, for each ReLU term $[\langle \mathbf{w}_j, \mathbf{x}_i \rangle]_+$ guess whether it equals 0 or $\langle \mathbf{w}_j, \mathbf{x}_i \rangle + b_j$ and replace the term in the error function accordingly. Furthermore, if the guess $[\langle \mathbf{w}_j, \mathbf{x}_i \rangle]_+ = 0$ was made then add the linear constraint $\langle \mathbf{w}_j, \mathbf{x}_i \rangle \leq 0$. Else, add the linear constraint $\langle \mathbf{w}_j, \mathbf{x}_i \rangle \geq 0$. After all guesses are made we get a convex optimization program with linear constraints. It is well known that such a convex optimization problem can be solved in time polynomial in $n, m, 1/\delta, C$ using a separation oracles and the ellipsoid algorithm (see for example, [B$^+$15, Section 2.1]). Since the number of guesses is at most $2^k \cdot (2^m)^k$, the claim follows. □

F.2 Learning k-ReLUs

We will now use the above training algorithms to give learning algorithms for ReLUs. We follow the agnostic learning model for real-valued function from [Hau92, KSS94]. A *concept class* $\mathcal{C} : \mathcal{Y}^{\mathcal{X}}$ is any set of functions from $\mathcal{X}$ to $\mathcal{Y}$. We say that a concept class $\mathcal{C}$ is *properly agnostically learnable* with respect to loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ if, for every $\delta, \epsilon > 0$, there is an algorithm $\mathcal{A}$ such that, for any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, takes in independent random samples from $\mathcal{D}$ and outputs a hypothesis $h \in \mathcal{C}$ such that, with probability $1 - \delta$, the following holds:

$$\mathcal{L}(h; \mathcal{D}) \leq \min_{c \in \mathcal{C}} \mathcal{L}(c; \mathcal{D}) + \epsilon$$

[19] ETH states that 3SAT with n variables and $m = O(n)$ clauses cannot be solved in $2^{o(n)}$ time.

where $\mathcal{L}(f;\mathcal{D}) := \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(f(\mathbf{x}),y)]$ is the expected loss for f over $\mathcal{D}$. Furthermore, if $\mathcal{X} \subseteq \mathbb{R}^n$ and the algorithm $\mathcal{A}$ runs in time polynomial in n and $1/\delta$, then $\mathcal{C}$ is said to be *efficiently* properly agnostically learnable. Throughout this section, we only consider the quadratic loss function (i.e., $\ell(y,y') = (y-y')^2$) and this will henceforth not be explicitly stated.

The concept classes we consider are the classes of sums of k ReLUs, where each coefficient has magnitude at most one, and the distribution $\mathcal{D}$ is allowed to be any distribution on the ball. More specifically, the class k-ReLU(n), which represent the sums of k ReLUs, is defined as follows:

Definition 5 (k-ReLU(n))**.** *For any $n,k \in \mathbb{N}$ and any $\mathbf{w}^1,\ldots,\mathbf{w}^k \in \mathbb{R}^n, \mathbf{a} \in \{-1,1\}^k$, we use $\text{RELU}_{\mathbf{w}^1,\ldots,\mathbf{w}^k,\mathbf{a}} : \mathcal{B}^n \to [-k,k]$ to denote the function $\text{RELU}_{\mathbf{w}^1,\ldots,\mathbf{w}^k,\mathbf{a}}(\mathbf{x}) = \sum_{j=1}^k a_j[\langle \mathbf{w}_j, \mathbf{x}\rangle]_+$. Let k-ReLU(n) denote the class $\{\text{RELU}_{\mathbf{w}^1,\ldots,\mathbf{w}^k,\mathbf{a}} \mid \mathbf{w}^1,\ldots,\mathbf{w}^k \in \mathcal{B}^n, \mathbf{a} \in \{-1,1\}^k\}$.*

We show that, for any fixed number of ReLUs k, the class above can be efficiently agnostically properly learned, as stated below.

Theorem 23. *For any $n,k \in \mathbb{N}$, ReLU(n,k) can be efficiently agnostically properly learned for the quadratic loss function in time $2^{O(k^5/\epsilon^2)} \cdot (n/\delta)^{O(1)}$ time.*

When the sum of ReLU's is realizable, we can get better running time both in terms of k, ϵ:

Theorem 24. *For any $n,k \in \mathbb{N}$, ReLU(n,k) can be efficiently properly learned in the realizable case for the quadratic loss function in time $2^{O(k^3/\epsilon \cdot \log^3(k/\epsilon))} \cdot (n/\delta)^{O(1)}$ time.*

We remark that learning algorithms immediately imply training algorithms, by simplying letting $\mathcal{D}$ be the uniform distribution on the input set of labelled samples $S = \{(\mathbf{x}_i, y_i)\}_{i\in[m]}$. Thereby, Theorems 23 and 24 imply Theorems 2 and 5, respectively.

F.2.1 Generalization Bounds

Before we get to our proofs, we state the necessary generalization bounds.

Theorem 25 ([BM02])**.** *Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$ and let $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be a b-bounded loss function that is L-Lispschitz in its first argument. Let $\mathcal{F} \subseteq (\mathcal{Y}')^{\mathcal{X}}$ and for any $f \in \mathcal{F}$, let $\mathcal{L}(f;\mathcal{D}) := \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(f(\mathbf{x}),y)]$ and $\mathcal{L}(f;S) := \frac{1}{m}\sum_{i=1}^m \ell(f(\mathbf{x}_i),y_i)$, where each sample $(\mathbf{x}_i,y_i) \in S$ is drawn independently uniformly at random according to $\mathcal{D}$. Then, for any $\delta > 0$, with probability at least $1-\delta$, the following is true for all $f \in \mathcal{F}$:*

$$|\mathcal{L}(f;\mathcal{D}) - \mathcal{L}(f;S)| \le 4 \cdot L \cdot \mathcal{R}_m(\mathcal{F}) + 2 \cdot b \cdot \sqrt{\frac{\log(1/\delta)}{m}} \tag{19}$$

where $\mathcal{R}_m(\mathcal{F})$ is the Rademacher complexity of the function class $\mathcal{F}$.

While the above bound is generic and easy to apply it turns out to be not tight, especially when $\mathcal{L}(f;S)$ is small. Below we list such a bound from [SST10]; for simplicity of presentation, we only state the bound when $\mathcal{L}(f;S) = 0$ which suffices for us. To state the bound, we also require the notion of smoothness of the loss; we say that a loss ℓ is H*-smooth* if it is differentiable in the first variable and the derivative is H-Lipchitz.

Theorem 26 ([SST10]). *Let $\mathcal{D}, S, \ell, \mathcal{F}, \mathcal{L}, \mathcal{R}_m(\mathcal{F})$ be as in Theorem 25. Furthermore, assume that ℓ is H-smooth. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following is true for all $f \in \mathcal{F}$ such that $\mathcal{L}(f; S) = 0$:*

$$\mathcal{L}(f; \mathcal{D}) \leq C \left(H \log^3 m \cdot \mathcal{R}_m(\mathcal{F})^2 + \frac{b \cdot \log(1/\delta)}{m} \right), \tag{20}$$

where $C > 1$ is an absolute constant.

To see the differences between Theorems 25 and 26, notice that, if we ignore the second term in (20) for the moment, we only require $m = O_{b,\delta}(1/\epsilon)$ to get $\mathcal{L}(f; \mathcal{D}) \leq \epsilon$ in the latter whereas the former would need $m = O_{b,\delta}(1/\epsilon^2)$. This will indeed result in the difference in the running time of the learning algorithms for k-ReLUs in the realizable versus agnostic case.

Finally, we also use the following bounds on the Rademacher complexity:

Theorem 27 ([KST08]). *Let $\mathcal{X} \subseteq \mathcal{B}^n$ and let $\mathcal{W} = \{\mathbf{x} \mapsto \langle \mathbf{x}, \mathbf{w} \rangle \mid \|w\|_2 \leq 1\}$. Then,*

$$\mathcal{R}_m(\mathcal{W}) \leq \sqrt{1/m}.$$

Fact 1. *Let $\mathcal{F}_1, \mathcal{F}_2 \subseteq \mathbb{R}^{\mathcal{X}}$ be any function classes and let $\mathcal{F} = \{f_1 + f_2 \mid f_1 \in \mathcal{F}_1, f_2 \in \mathcal{F}_2\}$. Then,*

$$\mathcal{R}_m(\mathcal{F}) \leq \mathcal{R}_m(\mathcal{F}_1) + \mathcal{R}_m(\mathcal{F}_2).$$

Theorem 28 ([BM02, LT91]). *Let $\psi : \mathbb{R} \to \mathbb{R}$ be Lipschitz with constant L_ψ and suppose that $\psi(0) = 0$. Let $\mathcal{Y} \subseteq \mathcal{R}$, and for a function $f \in \mathcal{Y}^{\mathcal{X}}$, let $\phi \circ f$ denote the composition of ψ and f. For $\mathcal{F} \subseteq \mathcal{Y}^{\mathcal{X}}$, let $\psi \circ \mathcal{F} = \{\psi \circ f \mid f \in \mathcal{F}\}$. It holds that $\mathcal{R}_m(\psi \circ \mathcal{F}) \leq 2 \cdot L_\psi \cdot \mathcal{R}_m(\mathcal{F})$.*

F.2.2 Properly Learning ReLUs: The Agnostic Case

Our proof of Theorem 23 follows by an application of a standard generalization argument to bound the sample complexity given the algorithm in Lemma 22. It turns out that in our case, the number of samples needed only depends on k and ϵ (and not the dimension n) due to boundedness of our networks. Hence, by applying the algorithm from Lemma 22, we immediately get Theorem 23.

We now proceed to prove Theorem 23. For ease of presentation, when we invoke the algorithm from Lemma 22, we will ignore the accuracy parameter δ and pretend that the algorithm output an actual optimal solution. The presence of δ only adds an additive term which we can make sufficiently small.

Proof of Theorem 23. First, let us describe the algorithm. Given samples $S = \{(\mathbf{x}_i, y_i)\}_{i \in [m]}$ where

$$m = \left\lceil \frac{1024 \cdot k^4 \cdot (1 + \log(1/\delta))}{\epsilon^2} \right\rceil.$$

We use the algorithm from Lemma 22 to solve for $\mathbf{w}_1, \ldots, \mathbf{w}_k, \mathbf{a}$ that minimizes the training error for the m samples. Then, we simply output the hypothesis $h = \textsc{relu}_{\mathbf{w}_1, \ldots, \mathbf{w}_k, \mathbf{a}}$.

Clearly, the algorithm is a proper learning algorithm. Furthermore, the running time is $2^{O(km)} poly(n, m) = 2^{O(k^5/\epsilon^2)} poly(n, m, 1/\delta)$ as desired.

Thus, we are left to bound the error $\mathcal{L}(h; \mathcal{D})$. To do this, first, observe that, from Theorem 27 and Theorem 28, we have $\mathcal{R}_m(1\text{-ReLU}(n)) \leq \frac{2}{\sqrt{m}}$. Hence, from Fact 1, we have $\mathcal{R}_m(k\text{-ReLU}(n)) \leq \frac{2k}{\sqrt{m}}$.

Now, observe that, for the region $[-k,k]\times[-k,k]$ the loss function $\ell(y',y)=(y'-y)^2$ is $(2k)$-Lipschitz in the first argument and $(4k^2)$-bounded. As a result, from Theorem 25, the following holds for all $f\in k$-ReLU(n) with probability at least $1-\delta$:

$$|\mathcal{L}(f;\mathcal{D})-\mathcal{L}(f;S)|\le 4\cdot(2k)\cdot\frac{2k}{\sqrt{m}}+2\cdot(4k^2)\cdot\sqrt{\frac{\log(1/\delta)}{m}}\le\frac{\epsilon}{2}, \tag{21}$$

where the second inequality comes from our choice of m.

Let $f_{\mathrm{OPT}}\in k$-ReLU(n) be the minimizer of $\mathcal{L}(f;\mathcal{D})$. Since h minimizes the training error,

$$\mathcal{L}(h;S)\le\mathcal{L}(f_{\mathrm{OPT}};S). \tag{22}$$

As a result, we have

$$\mathcal{L}(h;\mathcal{D})\overset{(21)}{\le}\mathcal{L}(h;S)+\frac{\epsilon}{2}\overset{(22)}{\le}\mathcal{L}(f_{\mathrm{OPT}};S)+\frac{\epsilon}{2}\overset{(21)}{\le}\mathcal{L}(f_{\mathrm{OPT}};\mathcal{D})+\epsilon=\left(\min_{f\in k\text{-ReLU}(n)}\mathcal{L}(f;\mathcal{D})\right)+\epsilon,$$

which concludes the proof. □

The results above should be compared to those of [GKKT17] who showed similar learnability results as above, except that their algorithm is *improper*. That is, their algorithm would output a (modification of) low-degree polynomial, as opposed to sums of ReLUs (which our algorithm outputs). We remark here that, while our algorithm is advantageous to their in this sense, their algorithm is faster extends to a larger class of networks.

F.2.3 Properly Learning ReLUs: The Realizable Case

Next we prove Theorem 24. For the realizable setup we can get better guarantees by using the improved generalization bound from Theorem 26.

Proof of Theorem 24. First, let us describe the algorithm. Given samples $S=\{(\mathbf{x}_i,y_i)\}_{i\in[m]}$ such that

$$m=\left\lceil\frac{10^6C\cdot k^2\log^3(10Ck/\epsilon)}{\epsilon}+\frac{8k^2\log(1/\delta)}{\epsilon}\right\rceil$$

where C is the constant from Theorem 26.

We use the algorithm from Lemma 22 to solve for $\mathbf{w}_1,\dots,\mathbf{w}_k,\mathbf{a}$ that minimizes the training error for the m samples. Then, we simply output the hypothesis $h=\textsc{relu}_{\mathbf{w}_1,\dots,\mathbf{w}_k,\mathbf{a}}$.

Clearly, the algorithm is a proper learning algorithm, and the running time is $2^{O(km)}poly(n,m)=2^{O((k^3/\epsilon)\cdot\log^3(k/\epsilon))}poly(n,m,1/\delta)$ as desired. Furthermore, since S is realizable, we must have $\mathcal{L}(h;S)=0$.

Finally, we will apply the generalization bound from Theorem 26. To do so, first recall from the proof of Theorem 23 that $\mathcal{R}_m(k\text{-ReLU}(n))\le\frac{2k}{\sqrt{m}}$ and that, in the region $[-k,k]\times[-k,k]$ the squared loss function is $(4k^2)$-bounded. Furthermore, the squared loss is 2-smooth. As a result, we may apply Theorem 26 which implies that, with probability $1-\delta$, we have

$$\mathcal{L}(h;\mathcal{D})\le C\left(2\log^3 m\cdot\frac{4k^2}{m}+\frac{4k^2\cdot\log(1/\delta)}{m}\right)\le\epsilon,$$

where the inequality follows from our choice of m. □