Hardness of Noise-Free Learning for Two-Hidden-Layer Neural Networks

Sitan Chen* UC Berkeley
Aravind Gollakota† UT Austin
Adam R. Klivans‡ UT Austin
Raghu Meka§ UCLA

November 13, 2022

Abstract

We give superpolynomial statistical query (SQ) lower bounds for learning two-hidden-layer ReLU networks with respect to Gaussian inputs in the standard (noise-free) model. No general SQ lower bounds were known for learning ReLU networks of any depth in this setting: previous SQ lower bounds held only for adversarial noise models (agnostic learning) [KK14, GGK20, DKZ20] or restricted models such as correlational SQ [GGJ+20, DKKZ20].

Prior work hinted at the impossibility of our result: Vempala and Wilmes [VW19] showed that general SQ lower bounds cannot apply to any real-valued family of functions that satisfies a simple non-degeneracy condition.

To circumvent their result, we refine a lifting procedure due to Daniely and Vardi [DV21] that reduces Boolean PAC learning problems to Gaussian ones. We show how to extend their technique to other learning models and, in many well-studied cases, obtain a more efficient reduction. As such, we also prove new cryptographic hardness results for PAC learning two-hidden-layer ReLU networks, as well as new lower bounds for learning constant-depth ReLU networks from label queries.

1 Introduction

In this paper we extend a central line of research proving representation-independent hardness results for learning classes of neural networks. We will consider arguably the simplest possible setting: given samples $(x_1, y_1), \ldots, (x_n, y_n)$ where for every $i \in [n]$, x_i is sampled independently from some distribution $\mathcal{D}$ over $\mathbb{R}^d$ and $y_i = f(x_i)$ for an unknown neural network $f : \mathbb{R}^d \to \mathbb{R}$, the goal is to output any function $\widehat{f}$ for which $\mathbb{E}_{x\sim\mathcal{D}}[(f(x) - \widehat{f}(x))^2]$ is small. This model is often referred to as the *realizable* or *noise-free* setting.

This problem has long been known to be computationally hard for discrete input distributions. For example, if $\mathcal{D}$ is supported over a discrete domain like the Boolean hypercube, then we have a variety of hardness results based on cryptographic/average-case assumptions [KS09, DLSS14, DSS16, DV20, DV21].

Over the last few years there has been a very active line of research on the complexity of learning with respect to continuous distributions, the most widely studied case being the assumption that $\mathcal{D}$

*sitanc@berkeley.edu. This work was supported in part by NSF Award 2103300.

†aravindg@cs.utexas.edu. Supported by NSF awards AF-1909204, AF-1717896, and the NSF AI Institute for Foundations of Machine Learning (IFML).

‡klivans@cs.utexas.edu. Supported by NSF awards AF-1909204, AF-1717896, and the NSF AI Institute for Foundations of Machine Learning (IFML).

§raghum@cs.ucla.edu. Supported by NSF CAREER Award CCF-1553605.

is a standard Gaussian in d dimensions. A rich algorithmic toolbox has been developed for the Gaussian setting [JSA15, ZSJ+17, BG17, LY17, Tia17, GKM18, GLM18, BJW19, ZYWG19, DGK+20, LMZ20, DK20, ATV21, CKM20, SZB21, VSS+22], but all known efficient algorithms can only handle networks with a single hidden layer, that is, functions of the form $f(x) = \sum_{i=1}^{k} \lambda_i \sigma(\langle w_i, x \rangle)$. This motivates the following well-studied question:

Are there fundamental barriers to learning neural networks with two *hidden layers?* (1)

Two distinct lines of research, one using cryptography and one using the statistical query (SQ) model, have made progress towards solving this question.

In the cryptographic setting, [DV21] showed that the existence of a certain class of pseudorandom generators, specifically local pseudorandom generators with polynomial stretch, implies superpolynomial lower bounds for learning ReLU networks with *three* hidden layers.

For SQ learning, work of [GGJ+20] and [DKKZ20] gave the first superpolynomial *correlational* SQ (CSQ) lower bounds for learning even one-hidden-layer neural networks. Notably, however, there are strong separations between SQ and CSQ [APVZ14, ADHV19, CKM20], and the question of whether a general SQ algorithm exists remained an interesting open problem. In fact, Vempala and Wilmes [VW19] showed that general SQ lower bounds might be impossible to achieve for learning real-valued neural networks. For any family of networks satisfying a simple non-degeneracy condition (see Section 1.1), they gave an algorithm that succeeded using only polynomially many statistical queries. As such, the prevailing conventional wisdom was that noise was required in the model to obtain full SQ lower bounds.

The main contribution of this paper is to answer Question 1 by giving both general SQ lower bounds and cryptographic hardness results (based on the Learning with Rounding or LWR assumption) for learning ReLU networks with two hidden layers and polynomially bounded weights.[1] We note that our SQ lower bound is the first of its kind for learning ReLU networks of *any* depth. We also show how to extend our results to the setting where the learner has label query access to the unknown network.

Reference	Num. hidden layers	Model of hardness
[DKKZ20, GGJ+20]	1	Correlational SQ
[DV21]	3	Cryptographic (assuming existence of local PRGs)
This work	2	Full SQ
This work	2	Cryptographic (assuming hardness of LWR)

Table 1: Summary of known and new superpolynomial lower bounds for learning noise-free shallow ReLU networks over Gaussian inputs up to sufficiently small (but non-negligible) error. (Definitions and terminology may be found in Section 2.)

SQ Lower Bound We state an informal version of our main SQ lower bound:

[1] Note that if the weights were allowed to be arbitrarily large, it is well-known to be trivial to obtain hardness over Gaussian inputs from hardness over Boolean inputs: simply approximate the sign function arbitrarily well and convert all but an arbitrarily small fraction of Gaussian inputs to bitstrings.

Theorem 1.1 (Full SQ lower bound for two hidden layers (informal), see Theorem 4.1)**.** *Any SQ algorithm for learning* $\mathrm{poly}(d)$*-sized two-hidden-layer ReLU networks over* $\mathcal{N}(0, \mathrm{Id}_d)$ *up to squared loss* $1/\mathrm{poly}(d)$ *must use at least* $d^{\omega(1)}$ *queries, or have query tolerance that is negligible in* d*.*

We stress that this bound holds unconditionally, independent of any cryptographic assumptions. This simultaneously closes the gap between the hardness result of [DV21] and the positive results on one-hidden-layer networks [JSA15, ZSJ+17, GLM18, ATV21, DK20] and goes against the conventional wisdom that one cannot hope to prove full SQ lower bounds for learning real-valued functions in the realizable setting.

We also note that unlike previous CSQ lower bounds which are based on orthogonal function families and crucially exploit cancellations specific to the Gaussian distribution, our Theorem 1.1 and other hardness results in this paper easily extend to any reasonably anticoncentrated and symmetric product distribution over $\mathbb{R}^d$; see Remark 3.11.

Cryptographic Lower Bound While Theorem 1.1 rules out almost all known approaches for provably learning neural networks (e.g. method of moments/tensor decomposition [JSA15, ZSJ+17, GLM18, BJW19, DGK+20, DK20, ATV21], noisy gradient descent [BG17, LY17, Tia17, GKM18, ZYWG19, LMZ20], and filtered PCA [CKM20]), it does not preclude the existence of a non-SQ algorithm for doing so. Indeed, a number of recent works [BRST21, SZB21, ZSWB22, DK21] have ported algorithmic techniques like lattice basis reduction [LLL82], traditionally studied in the context discrete settings like cryptanalysis, to learning problems over continuous domains for which there is no corresponding SQ algorithm.

Our next result shows however that under a certain cryptographic assumption, namely hardness of *Learning with Rounding (LWR) with polynomial modulus* [BPR12, AKPW13, BGM+16] (see Section 2), no polynomial-time algorithm can learn two-hidden-layer neural networks from Gaussian examples.

Theorem 1.2 (Cryptographic hardness result (informal), see Theorem 5.1)**.** *Suppose there exists a* $\mathrm{poly}(d)$*-time algorithm for learning* $\mathrm{poly}(d)$*-sized two-hidden-layer ReLU networks over* $\mathcal{N}(0, \mathrm{Id}_d)$ *up to squared loss* $1/\mathrm{poly}(d)$*. Then there exists a quasipolynomial-time algorithm for* LWR *with polynomial modulus.*

Note that here we may actually improve the LWR hardness assumption required from quasipolynomial to any mildly superpolynomial function of the security parameter (see Remark 5.2).

Under LWR with polynomial modulus, we also show the first hardness result for learning *one hidden layer* ReLU networks over the uniform distribution on $\{0,1\}^d$ (see Theorem 5.3).

In Section 2, we discuss existing hardness evidence for LWR as well as its relation to more standard assumptions like Learning with Errors. From a negative perspective, Theorem 1.2 suggests that the aforementioned lattice-based algorithms for continuous domains are unlikely to yield new learning algorithms for two-hidden-layer networks, because even their more widely studied *discrete* counterparts have yet to break LWR. From a positive perspective, in light of the prominent role LWR and its variants have played in a number of practical proposals for post-quantum cryptography [CKLS18, BGML+18, JZ16, DKRV18], Theorem 1.2 offers a new avenue for stress-testing these schemes.

Query Learning Lower Bound One additional benefit of our techniques is that they are flexible enough to accommodate other learning models beyond traditional PAC learning. To illustrate this, for our final result we show hardness of learning neural networks from *label queries*. In this setting, the learner is much more powerful: rather than sample or SQ access, they are given the ability to

query the value $f(x)$ of the unknown function f at any desired point x in $\mathbb{R}^d$, and the goal is still to output a function $\widehat{f}$ for which $\mathbb{E}[(f(x) - \widehat{f}(x))^2]$ is small. The expectation here is with respect to some specified distribution, which we will take to be $\mathcal{N}(0, \mathrm{Id}_d)$, though as before, our techniques will apply to any reasonably anticoncentrated, symmetric product distribution over $\mathbb{R}^d$.

In recent years, this question has received renewed interest from the security and privacy communities in light of *model extraction attacks*, which attempt to reverse-engineer neural networks found in publicly deployed systems [TJ+16, MSDH19, PMG+17, JCB+20, RK20, JWZ20, DG21]. Recent work [CKM21] has shown that in this model, there is an efficient algorithm for learning arbitrary one-hidden-layer ReLU networks that is truly polynomial in all relevant parameters. We show that under plausible cryptographic assumptions about the existence of simple pseudorandom function (PRF) families (see Section 6) which may themselves be based on standard number theoretic or lattice-based cryptographic assumptions, such a guarantee is impossible for general *constant-depth* ReLU networks.

Theorem 1.3 (Label query hardness (informal), see Theorem 6.1)**.** *If either the decisional Diffie–Hellman or the Learning with Errors assumption holds, then the class of* $\mathrm{poly}(d)$*-sized constant-depth ReLU networks from* $\mathbb{R}^d$ *to* $\mathbb{R}$ *is not learnable up to small constant squared loss* ε *over* $\mathcal{N}(0, \mathrm{Id}_d)$ *even using label queries over all of* $\mathbb{R}^d$*.*

Note that the connection between PRFs and hardness of learning from label queries over *discrete domains* is a well-known connection dating back to Valiant [Val84]. To our knowledge, however, Theorem 1.3 is the first hardness result for query learning over continuous domains.

1.1 Discussion and Related Work

Hardness for learning neural networks. There are a number of works [BR89, Vu06, KS09, LSSS14, GKKT17, DV20] showing hardness for *distribution-free* learning of various classes of neural networks.

As for hardness of distribution-specific learning, several works have established lower bounds with respect to the Gaussian distribution. Apart from the works [GGJ+20, DKKZ20, DV21] from the introduction which are most closely related to the present work, we also mention the works of [KK14, GKK19, GGK20, DKZ20] which showed hardness for *agnostically* learning halfspaces and ReLUs, [Sha18] which showed hardness for learning periodic activations with gradient-based methods, [SVWX17] which showed lower bounds against SQ algorithms for learning one-hidden-layer networks using *Lipschitz* statistical queries and large tolerance, and [SZB21] which showed lattice-based hardness of learning one-hidden-layer networks when the labels y_i have been perturbed by bounded *adversarially chosen* noise. Our approach has similarities to the "Gaussian lift" as studied by Klivans and Kothari [KK14]. Their approach, however, required noise in the labels, whereas we are interested in hardness in the strictly *realizable setting*. We also remark that [DGKP20, AAK21] showed *correlational* SQ lower bounds for learning random depth-$\omega(\log n)$ neural networks over *Boolean inputs* which are uniform over a halfspace.

There have also been works on hardness of learning from label queries over *discrete domains* and for more "classical" concept classes like Boolean circuits [Fel09, CGV15, Val84, Kha95, AK95].

Lastly, we remark on how our results relate to [CKM20], which gives the only known upper bound for learning neural networks over Gaussian inputs beyond one hidden layer. They showed that learning ReLU networks of arbitrary depth is "fixed-parameter tractable" in the sense that there is a fixed function $g(k, \varepsilon)$ in the size k of the network and target error ε for which the time complexity is at most $g(k, \varepsilon) \cdot \mathrm{poly}(d)$, and their algorithm can be implemented in SQ. That said, this

does not contradict our lower bounds for two reasons: 1) their algorithm only applies to networks without biases, 2) in our lower bound constructions, k scales polynomially in d.

SQ lower bounds for real-valued functions. A recurring conundrum in the literature on SQ lower bounds for supervised learning has been whether one can show SQ hardness for learning *real-valued* functions. SQ lower bounds for Boolean functions are typically shown by lower bounding the *statistical dimension* of the function class, which essentially corresponds to the largest possible set of functions in the class which are all approximately pairwise orthogonal. Indeed, the content of the hardness results of [GGJ+20, DKKZ20] was to prove lower bounds on the statistical dimension of one-hidden-layer networks. Unfortunately, for real-valued functions, statistical dimension lower bounds only imply CSQ lower bounds. As discussed in [GGJ+20], the class of d-variate Hermite polynomials of degree-ℓ is pairwise orthogonal and of size $d^{O(\ell)}$, which translates to a CSQ lower bound of $d^{\Omega(\ell)}$. Yet there exist SQ algorithms for learning Hermite polynomials in far fewer queries [APVZ14, ADHV19].

Further justification for the difficulty of proving SQ lower bounds for real-valued functions came from [VW19], which observed that for any real-valued learning problem satisfying a seemingly innocuous non-degeneracy assumption—namely that for any pair of functions f, g in the class, the probability under the input distribution $\mathcal{D}$ that $f(x) = g(x)$ is zero—there is an efficient "cheating" SQ algorithm (see Proposition 4.1 therein). The SQ lower bound shown in the present work circumvents this proof barrier by exhibiting a family of neural networks for which any pair of networks agrees on a set of inputs with Gaussian measure *bounded away from zero.*

Open questions While our results settle Question 1, a number of intriguing gaps between our lower bounds and existing upper bounds remain open:

- **General one-hidden-layer networks.** Despite the considerable amount of work on learning one-hidden-layer networks over Gaussian inputs, all known positive results that run in polynomial time in all parameters (input dimension d, network size k, inverse error $1/\varepsilon$) still need to make various assumptions on the structure of the network. Remarkably, it is even open whether one-hidden-layer ReLU networks with *positive output layer weights* (i.e. "sums of ReLUs") can be learned in polynomial time, the best known guarantee being the $(k/\varepsilon)^{\log^2 k} \cdot \mathrm{poly}(d/\varepsilon)$-time algorithm of [DK20]. As for general one-hidden-layer ReLU networks, it is still open whether they can even be learned in time $d^{O(k)} \cdot \mathrm{poly}(1/\varepsilon)$, the best known guarantee being the $k^{\mathrm{poly}(k/\varepsilon)} \cdot \mathrm{poly}(d)$-time algorithm of [CKM20].
- **Query learning shallow networks.** While Theorem 1.3 establishes that above a certain constant depth, ReLU networks cannot be learned even from label queries over the Gaussian distribution. It would be interesting to close the gap between this and the positive result of [CKM21] which only applies to one-hidden-layer networks, although fully settling this seems closely related to the question of what are the shallowest possible Boolean circuits needed to implement pseudorandom functions, a longstanding open question in circuit complexity.

1.2 Technical Overview

Our work will build on a recent approach of Daniely and Vardi [DV21], who developed a simple and clever technique for lifting discrete functions to the Gaussian domain entirely in the realizable setting. Our main contributions are to (1) make their lifting procedure more efficient so that two hidden layers suffice and (2) show how to apply the lift in a variety of models beyond PAC. For the

purposes of this overview we will take the domain of our discrete functions to be $\{0,1\}^d$, but our techniques extend to $\mathbb{Z}_q^d$ with $q = \mathrm{poly}(d)$.

Daniely–Vardi (DV) lift. At a high level, the DV lift is a transformation mapping a Boolean example (x, y) labeled by a hard-to-learn Boolean function f to a Gaussian example $(z, \widetilde{y})$ labeled by a (real-valued) ReLU network f^{DV} that behaves similarly to f in that $f^{\mathsf{DV}}(z)$ approximates $f(\mathrm{sign}(z))$, where for us $\mathrm{sign}(t)$ denotes $\mathbb{1}[t > 0]$ and is applied elementwise. The key idea is to use a continuous approximation $\widetilde{\mathrm{sign}}$ of the sign function, and to pair it with a "soft indicator" function $\mathrm{bad} : \mathbb{R}^d \to \mathbb{R}_+$ that is large whenever $\mathrm{sign}(z) \neq \widetilde{\mathrm{sign}}(z)$, and that can be implemented as a one-hidden-layer network independent of the target function. One can show that whenever f is realizable as an L-hidden-layer network over $\{0,1\}^d$, the function $f^{\mathsf{DV}}(z) = \mathrm{ReLU}(f(\widetilde{\mathrm{sign}}(z)) - \mathrm{bad}(z))$ can be implemented as an $(L+2)$-hidden-layer network satisfying

$$f^{\mathsf{DV}}(z) = \mathrm{ReLU}(f(\mathrm{sign}(z)) - \mathrm{bad}(z)).$$

This property allows us to generate synthetic Gaussian labeled examples $(z, f^{\mathsf{DV}}(z))$ from Boolean labeled examples $(x, f(x))$, and thereby reduce the problem of learning f to that of learning f^{DV}. For a fuller overview, see Section 3.1.

Improving the DV lift. Our first technical contribution is to introduce a more efficient lift which only requires one extra hidden layer. Our starting point is to observe that a variety of hard-to-learn Boolean functions f like parity and LWR take the form $f(x) = \sigma(h(x))$ for some ReLU network h whose range T over Boolean inputs is a discrete subset of $[0, \mathrm{poly}(d)]$ of *polynomially bounded* size, and for some function $\sigma : T \to [0,1]$. For such *compressible* functions (see Definition 3.1), one can write $f(x) = \sigma(h(x)) = \sum_{t^* \in T} \sigma(t^*)\mathbb{1}[h(x) = t^*]$. Again, we would like to implement lifted function $f^{\triangle} : \mathbb{R}^d \to \mathbb{R}$ using $\widetilde{\mathrm{sign}}$ and bad so that it approximates $f(\mathrm{sign}(z))$ except when bad indicates that $\widetilde{\mathrm{sign}} \neq \mathrm{sign}$. To this end, we might hope to implement, say,

$$f^{\triangle}(z) = \sum_{t^* \in T} \sigma(t^*)\mathbb{1}[h(\widetilde{\mathrm{sign}}(z)) = t^*]\mathbb{1}[\forall j : \mathrm{bad}(z_j) \ll 1].$$

Here we now view bad as a univariate function, and whenever it is small, we can be sure $\widetilde{\mathrm{sign}} = \mathrm{sign}$. Suppose that we could build a one-hidden-layer network $N(s_1, \ldots, s_d; t)$ that behaves like $\mathbb{1}[t = 0]\mathbb{1}[\forall j : s_j \ll 1]$. Then we could realize $f^{\triangle}$ as an $(L+1)$-hidden-layer network:

$$f^{\triangle}(z) = \sum_{t^* \in T} \sigma(t^*) N(\mathrm{bad}(z_1), \ldots, \mathrm{bad}(z_d);\ h(\widetilde{\mathrm{sign}}(z)) - t^*).$$

While many natural attempts to build such an N run into difficulties, we construct a suitably relaxed version of N that turns out to suffice for the reduction. To gain some intuition for our construction, the starting observation is that the following inclusion-exclusion type formula vanishes identically whenever any of the s_j is 1:

$$\begin{aligned}&\psi(s_1, s_2, s_3) - \psi(1, s_2, s_3) - \psi(s_1, 1, s_3) - \psi(s_1, s_2, 1)\\ &+ \psi(s_1, 1, 1) + \psi(1, s_2, 1) + \psi(s_1, 1, 1) - \psi(1, 1, 1).\end{aligned}$$

For a suitable choice of ψ, one might hope to build N out of such a formula by taking $s_j = \mathrm{bad}(z_j)$ for every j. But the natural generalization of this expression to d inputs would have size 2^d, which runs the risk of rendering the resulting SQ lower bounds vacuous. Our final construction (Lemma 3.10)

instead resembles a truncated inclusion-exclusion type formula of only quasipolynomial size, which may be of independent interest. Since the SQ lower bounds for Boolean functions that we build on are exponential, by a simple padding argument we still obtain a superpolynomial SQ lower bound for our lifted functions.

Hard one-hidden-layer Boolean functions and LWR. To use this lift for Theorems 1.1 and 1.2, we need one-hidden-layer networks that are *compressible* and hard to learn over uniform Boolean inputs. For SQ lower bounds, we can simply start from parities, for which there are exponential SQ lower bounds, and which turn out to be easily implementable by compressible one-hidden-layer networks. For cryptographic hardness, Daniely and Vardi [DV21] used certain one-hidden-layer Boolean networks that arise from the cryptographic assumption that local PRGs exist (see Section A.4.1 therein). Unfortunately, these functions are not compressible. For this reason, we work instead with LWR: it turns out that the LWR functions are compressible and, conveniently, the hardness assumption directly involves uniform discrete inputs.

Hardness beyond PAC. While the DV lift is *a priori* only for showing hardness of example-based PAC learning, we can extend it to the SQ and label query models by simple simulation arguments.

2 Preliminaries

2.1 Notation

We use $\mathrm{Unif}(S)$ to denote the uniform distribution over a set S. We use U_d as shorthand for $\mathrm{Unif}\{0,1\}^d$. We use $\mathcal{N}(0, \mathrm{Id}_d)$ (or sometimes $\mathcal{N}_d$ for short) to denote the standard Gaussian, and $|\mathcal{N}(0, \mathrm{Id}_d)|$ (or $|\mathcal{N}_d|$ for short) to denote the positive standard half-Gaussian (i.e., $g \sim |\mathcal{N}(0, \mathrm{Id}_d)|$ if $g = |z|$ for $z \sim \mathcal{N}(0, \mathrm{Id}_d)$). We use $[n]$ to denote $\{1, \dots, n\}$.

For $q > 0$, $\mathbb{Z}_q$ will denote the integers modulo q, which we will identify with $\{0, \dots, q-1\}$. We use $\mathbb{Z}_q/q$ to denote $\{0, 1/q, \dots, (q-1)/q\}$. Our discrete functions will in general have domain $\mathbb{Z}_q^d$ for some q. The $q = 2$ case, namely Boolean functions, have domain $\{0,1\}^d$. For the purposes of this paper, $\mathrm{sign} : \mathbb{R} \to \{0,1\}$ is defined as $\mathrm{sign}(t) = \mathbb{1}[t > 0]$. We will extend this to $\mathbb{Z}_q$ by defining $\mathrm{thres}_q : \mathbb{R} \to \mathbb{Z}_q$ in terms of a certain partition of $\mathbb{R}$ into q intervals $I_0, \dots, I_{q-1}$ (formally defined later) as the piecewise constant function that takes on value k on I_k for each $k \in \mathbb{Z}_q$. Scalar functions and scalar arithmetic applied to vectors act elementwise. We say a quantity is *negligible* in a parameter n, denoted $\mathrm{negl}(n)$, if it decays as $1/n^{\omega(1)}$.

A *one-hidden-layer* ReLU network mapping $\mathbb{R}^d$ to $\mathbb{R}$ is a linear combination of ReLUs, that is, a function of the form

$$F(x) = W_1 \,\mathrm{ReLU}\left(W_0 x + b_0\right) + b_1,$$

where $W_0 \in \mathbb{R}^{k \times d}$, $W_1 \in \mathbb{R}^{1 \times k}$, $b_0 \in \mathbb{R}^k$, and $b_1 \in \mathbb{R}$. A *two-hidden-layer* ReLU network mapping $\mathbb{R}^d$ to $\mathbb{R}$ is a linear combination of ReLUs of one-hidden-layer networks, that is, a function of the form

$$F(x) = W_2 \,\mathrm{ReLU}\left(W_1 \,\mathrm{ReLU}\left(W_0 x + b_0\right) + b_1\right) + b_2,$$

where $W_0 \in \mathbb{R}^{k_0 \times d}$, $W_1 \in \mathbb{R}^{k_1 \times k_0}$, $W_2 \in \mathbb{R}^{1 \times k_1}$, $b_0 \in \mathbb{R}^{k_0}$, $b_1 \in \mathbb{R}^{k_1}$, and $b_2 \in \mathbb{R}$. Our usage of the term *hidden layer* thus corresponds to a *nonlinear* layer.

2.2 Learning models

Let $\mathcal{C}$ be a function class mapping $\mathbb{R}^d$ to $\mathbb{R}$, and let $\mathcal{D}$ be a distribution on $\mathbb{R}^d$. We consider various learning models where the learner is given access in different ways to labeled data $(x, f(x))$ for an unknown $f \in \mathcal{C}$ and must output a (possibly randomized) predictor that achieves (say) squared loss ε for any desired $\varepsilon > 0$. In the traditional PAC model, access to the data is in the form of iid labeled examples $(x, f(x))$ where $x \sim \mathcal{D}$, and the learner is considered efficient if it succeeds using $\mathrm{poly}(d, 1/\epsilon)$ time and sample complexity. In the Statistical Query (SQ) model [Kea98, Rey20], access to the data is through an SQ oracle. Given a bounded query $\phi : \mathbb{R}^d \times \mathbb{R} \to [-1, 1]$ and a tolerance $\tau > 0$, the oracle may respond with any value v such that $|v - \mathbb{E}_{x\sim\mathcal{D}}[\phi(x, f(x))]| \leq \tau$. A correlational query is one that is linear in y, i.e. of the form $\phi(x, y) = \widetilde{\phi}(x)y$ for some $\widetilde{\phi}$, and a correlational SQ (CSQ) learner is one that is only allowed to make CSQs. An SQ learner is considered efficient if it succeeds using $\mathrm{poly}(d, 1/\epsilon)$ queries and tolerance $\tau \geq 1/\mathrm{poly}(d, 1/\epsilon)$. Finally, in the label query model, the learner is allowed to request the value of $f(x)$ for any desired x, and is considered efficient if it succeeds using $\mathrm{poly}(d, 1/\epsilon)$ time and queries.

2.3 Learning with Rounding

The Learning with Rounding (LWR) problem [BPR12] is a close cousin of the well-known Learning with Errors (LWE) problem [Reg09], except with deterministic rounding in place of random additive errors.

Definition 2.1. Fix moduli $p, q \in \mathbb{N}$, where $p < q$, and let n be the security parameter. For any $w \in \mathbb{Z}_q^n$, define $f_w : \mathbb{Z}_q^n \to \mathbb{Z}_p/p$ by

$$f_w(x) = \frac{1}{p}\lfloor w \cdot x \rceil_p = \frac{1}{p}\lfloor \frac{p}{q}(w \cdot x \bmod q) \rceil,$$

where $\lfloor t \rceil$ is the closest integer to t. In the $\mathsf{LWR}_{n,p,q}$ problem, the secret w is drawn randomly from $\mathbb{Z}_q^n$, and we must distinguish between labeled examples (x, y) where $x \sim \mathbb{Z}_q^n$ and either $y = f_w(x)$ or y is drawn independently from $\mathrm{Unif}(\mathbb{Z}_p/p)$. The $\mathsf{LWE}_{n,q,B}$ problem is similar, except that $y \in \mathbb{Z}_q/q$ is either $\frac{1}{q}((w \cdot x + e) \bmod q)$ for some $e \in \mathbb{Z}_q$ sampled from a carefully chosen distribution, e.g. discrete Gaussian, such that $|e| \leq B$ except with $\mathrm{negl}(n)$ probability, or is drawn from $\mathrm{Unif}(\mathbb{Z}_q/q)$.

Remark 2.2. Traditionally the LWR problem is stated with labels lying in $\mathbb{Z}_p$ instead of $\mathbb{Z}_p/p$, although both are equivalent since the moduli p, q may be assumed to be known to the learner. The choice of $\mathbb{Z}_p/p$ is simply a convenient way to normalize labels to lie in $[0, 1]$. For consistency, we similarly normalize LWE labels to lie in $\mathbb{Z}_q/q$.

It is known that $\mathsf{LWE}_{n,q,B}$ is as hard as worst-case lattice problems when $q = \mathrm{poly}(n)$ and $B = q/\mathrm{poly}(n)$ (see e.g. [Reg10, Pei16] for surveys). Yet this is not known to directly imply the hardness of $\mathsf{LWR}_{n,p,q}$ in the regime in which p, q are both $\mathrm{poly}(n)$, which is the one we will be interested in as p, q will dictate the size of the hard networks that we construct in the proof of our cryptographic lower bound.

Unfortunately, in this polynomial modulus regime, it is only known how to reduce from LWE to LWR *when the number of samples is bounded relative to the modulus* [AKPW13, BGM+16]. For instance, the best known reduction in this regime obtains the following hardness guarantee:

Theorem 2.3 ([BGM+16])**.** *Let n be the security parameter, let $p, q \geq 1$ be moduli, and let $m, B \geq 0$. Assuming $q \geq \Omega(mBp)$, any distinguisher capable of solving $\mathsf{LWR}_{n,p,q}$ using m samples implies an efficient algorithm for $\mathsf{LWE}_{n,q,B}$.*

For our purposes, Theorem 2.3 is not enough to let us base our Theorem 1.2 off of LWE, as we are interested in the regime where the learner has an *arbitrary* polynomial number of samples.

LWR with polynomial modulus and arbitrary polynomial samples is nevertheless conjectured to be as hard as worst-case lattice problems [BPR12] and has already formed the basis for a number of post-quantum cryptographic proposals [DKRV18, CKLS18, BGML+18, JZ16]. We remark that one piece of evidence in favor of this conjecture is a reduction from a less standard variant of LWE in which the usual discrete Gaussian errors are replaced by errors uniformly sampled from the integers $\{-q/2p, \ldots, q/2p\}$ [BGM+16].

Note also that for our purposes we require *quasipolynomial*-time hardness (or $T(n)$-hardness for $T(n)$ being any other fixed, mildly superpolynomial function of the security parameter) of LWR. While slightly stronger than standard polynomial-time hardness, this remains a reasonable assumption since algorithms for worst-case lattice problems are still believed to require at least subexponential time.

2.4 Partial assignments

Let $\alpha \in \{0, 1, \star\}^d$ be a *partial assignment*. We refer to $S(\alpha) : \{i \in [d] : \alpha_i = \star\} \subseteq [d]$ as the set of *free variables* and $[d] \backslash S(\alpha)$ as the set of *fixed variables*. Given two partial assignments α, β, let the *resolution* $\alpha \searrow \beta$ denote the partial assignment γ obtained by substituting α into β. That is,

$$\gamma_i = \begin{cases} \star & i \in S(\alpha) \cap S(\beta) \\ \beta_i & i \in [d] \backslash S(\beta) \\ \alpha_i & i \in S(\beta) \backslash S(\alpha) \end{cases}$$

In this case we say that γ is a *refinement* of β that is the *result of applying* α. We write $\gamma \in \mathrm{App}(\alpha)$ to denote that γ is a result of applying α. Note that the set of refinements of β consists of all $3^{|S(\beta)|}$ partial assignments $\gamma \in \{0, 1, \star\}^d$ which agree with β on all fixed variables of β.

Given α, let $w(\alpha)$ denote $|\{i : \alpha_i = 1\}|$, that is, the Hamming weight of its fixed variables. Note that $w(\alpha \searrow \beta) \leq w(\alpha) + w(\beta)$.

Given a function $h : \mathbb{R}^d \to \mathbb{R}$ and partial assignment γ, we use $h_\gamma : \mathbb{R}^d \to \mathbb{R}$ to denote its partial restriction given by substituting in γ_i into the i-th input coordinate if $\gamma_i \in \{0, 1\}$. Note that given two partial restrictions α, β,

$$(h_\beta)_\alpha = h_{\alpha \searrow \beta} \tag{2}$$

We say that α is *sorted* if the restriction of α to its fixed variables is sorted in nonincreasing order, e.g. $\alpha = (1, \star, 1, \star, \star, 0, 0)$ is sorted, but $\alpha = (1, \star, 0, \star, \star, 0, 1)$ is not. Given α which is not necessarily sorted, denote its *sorting* by $\overline{\alpha}$. In general, we will use overline notation to denote sorted partial assignments.

3 Compressing the Daniely–Vardi Lift

In this section we show how to refine the lifting procedure of Daniely and Vardy [DV21] such that whenever the underlying discrete functions satisfy a property we term *compressibility*, we obtain hardness under the Gaussian for networks with just one extra hidden layer.

Definition 3.1. Let $q > 0$ be a modulus.[2] We call an L-hidden-layer ReLU network $f : \mathbb{Z}_q^d \to [0, 1]$ *compressible* if it is expressible in the form $f(x) = \sigma(h(x))$, where

[2]Our results are stronger when q is taken to be a large polynomial in the dimension, but the Boolean $q = 2$ case is illustrative of all the main ideas.

- $h : \mathbb{Z}_q^d \to T$ is an $(L-1)$-hidden-layer network such that $|h(x)| \le \mathrm{poly}(d)$ for all x;
- h has range $T = h(\mathbb{Z}_q^d)$ such that $T \subseteq \mathbb{Z}$ and $|T| \le \mathrm{poly}(d)$; and
- $\sigma : T \to [0,1]$ is a mapping from h's possible output values to $[0,1]$.

Remark 3.2. To see why such an f is an L-hidden-layer network in z, consider the function $\sigma : T \to \mathbb{R}$. Because $T \subseteq \mathbb{Z}$ and $|T| \le \mathrm{poly}(d)$, σ is expressible as (the restriction to T of) a piecewise linear function on $\mathbb{R}$ whose size and maximum slope are $\mathrm{poly}(d)$, and hence as a $\mathrm{poly}(d)$-sized one-hidden-layer ReLU network from $\mathbb{R}$ to $\mathbb{R}$. By composition, $x \mapsto \sigma(h(x))$ can be represented by an L-hidden-layer network.

We now formally state a theorem which captures our "compressed" version of the DV lift. The version of this theorem for $L+2$ layers is implicit in [DV21]. In technical terms, our improvement consists of removing the single outer ReLU present in their construction. Thus, while our construction still has three *linear* layers, it has only two *non-linear* layers.

Theorem 3.3 (Compressed DV lift)**.** *Let $q = \mathrm{poly}(d)$ be a modulus. Let $\mathcal{C}$ be a class of compressible L-hidden-layer $\mathrm{poly}(d)$-sized ReLU networks mapping $\mathbb{Z}_q^d$ to $[0,1]$. Let $m = m(d) = \omega_d(1)$ be a size parameter that grows slowly with d. There exists a class $\mathcal{C}^\triangle$ of $(L+1)$-hidden-layer $d^{\Theta(m)}$-sized ReLU networks mapping $\mathbb{R}^d$ to $[0,1]$ such that the following holds:*

Suppose there is an efficient algorithm A capable of learning $\mathcal{C}^\triangle$ over $\mathcal{N}(0, \mathrm{Id}_d)$ up to squared loss $d^{-\Theta(m)}$. Then there is an efficient algorithm B capable of weakly predicting *$\mathcal{C}$ over $\mathrm{Unif}(\mathbb{Z}_q^d)$ with advantage $d^{-\Theta(m)}$ over guessing the constant $1/2$ in the following sense: given access to labeled examples $(x, f(x))$ for $x \sim \mathrm{Unif}(\mathbb{Z}_q^d)$ and an unknown $f \in \mathcal{C}$, B satisfies*

$$\mathbb{E}\left[\big(B(x) - f(x)\big)^2\right] < \mathbb{E}\left[\big(\frac{1}{2} - f(x)\big)^2\right] - d^{-\Theta(m)},$$

where the probability is taken over both x and the internal randomness of B. We refer to $\mathcal{C}^\triangle$ as the lifted class *corresponding to $\mathcal{C}$.*

By a standard padding argument, we obtain the following corollary which lets us work with polynomial-sized neural networks.

Corollary 3.4 (Compressed DV lift with padding)**.** *Let q, m and d be as above, and let $d' = d^m$. View $\mathcal{C}$ and $\mathcal{C}^\triangle$ as function classes on $\mathbb{Z}_q^{d'}$ and $\mathbb{R}^{d'}$ respectively, defined using only the first d coordinates, so that $\mathcal{C}^\triangle$ is now a $\mathrm{poly}(d')$-sized class over $\mathbb{R}^{d'}$. Then an algorithm capable of learning $\mathcal{C}^\triangle$ over $\mathcal{N}_{d'}$ up to squared loss $1/\mathrm{poly}(d')$ implies a weak predictor for $\mathcal{C}$ over $\mathrm{Unif}(\mathbb{Z}_q^{d'})$ with advantage $1/\mathrm{poly}(d')$.*

3.1 The DV Lift

Before proceeding to the proof of Theorem 3.3, we first outline the idea of the original DV lift in the setting of Boolean functions ($q = 2$). The goal is to approximate any given $f \in \mathcal{C}$ by a ReLU network $f^{\mathsf{DV}} : \mathbb{R}^d \to \mathbb{R}$ in such a way that f^{DV} under $\mathcal{N}_d$ behaves similarly to f under U_d. As a first attempt, one might consider the function $f^\star(z) = f(\mathrm{sign}(z))$ (also studied in [KK14]), where recall that $\mathrm{sign}(t) = \mathbb{1}[t > 0]$. We could implement the following reduction: given a random example (x, y) where $x \sim U_d$ and $y = f(x)$, draw a fresh half-Gaussian $g \sim |\mathcal{N}_d|$ and output $((2x-1)g, y)$ (where the arithmetic in defining the vector $(2x-1)g$ is done elementwise). Since $2x-1$ is distributed uniformly over $\{\pm 1\}^d$, the marginal is exactly $\mathcal{N}_d$, and the labels are consistent with $f^\star$ since $\mathrm{sign}((2x-1)g) = x$ and so $f(\mathrm{sign}((2x-1)g)) = f(x)$. However, the issue is that the sign function is discontinuous, and so $f^\star$ is not realizable as a ReLU network.

Daniely and Vardi address this concern by devising a clever construction for f^{DV} that interpolates between two desiderata:

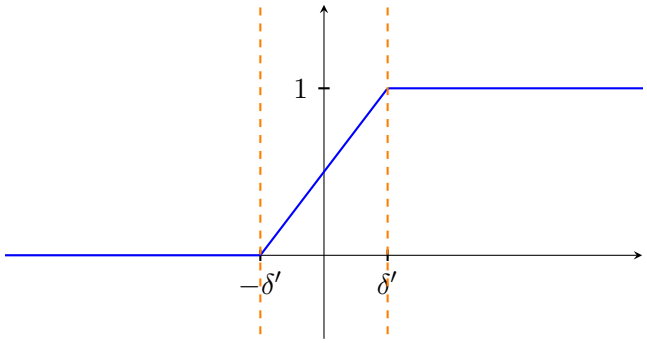
- For all but a small fraction of inputs, an initial layer successfully "Booleanizes" the input. In this case, one would like $f^{\mathsf{DV}}(z)$ to simply behave as $f(\mathrm{sign}(z))$.
- For the remaining fraction of inputs, we would ideally like f^{DV} to output an uninformative value such as zero, but this would violate continuity of f^{DV}.

The trick is to use a continuous approximation of the sign function, N_1, that interpolates linearly between 0 and 1 on an interval $[-\delta, \delta]$ (see Fig. 1a), and to pair it with a "soft indicator" function $N_2 : \mathbb{R} \to \mathbb{R}$ for the region where $N_1 \neq \mathrm{sign}$. Concretely, $N_2(t)$ is constructed as a one-hidden-layer ReLU network that (a) is always nonnegative, (b) equals 0 when $|t| \geq 2\delta$, and (c) equals 1 when $|t| \leq \delta$ (see Fig. 1b). Now let $N_2'(z) = \sum_j N_2(z_j)$, and define

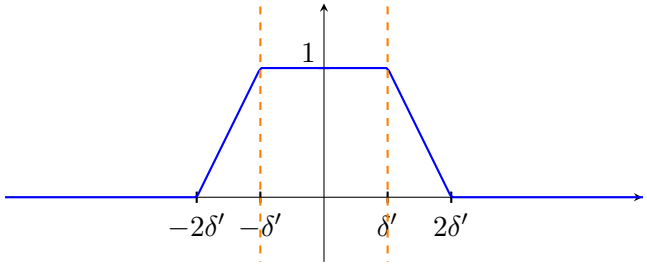
$$f^{\mathsf{DV}}(z) = \mathrm{ReLU}(f(N_1(z)) - N_2'(z)). \tag{3}$$

One can show that f^{DV} satisfies $f^{\mathsf{DV}}(z) = \mathrm{ReLU}(f(\mathrm{sign}(z)) - N_2'(z))$, since N_2' "zeroes out" f^{DV} wherever $N_1 \neq \mathrm{sign}$ for any coordinate. This lets us perform the following reduction: given examples (x, y) where $x \sim U_d$ and $y = f(x)$, draw a fresh $g \sim |\mathcal{N}_d|$ and output $(z, \widetilde{y}) = ((2x-1)g, \mathrm{ReLU}(y - N_2'((2x-1)g)))$. The marginal is again $\mathcal{N}_d$, and the labels are easily seen to be consistent with f^{DV}. Correctness of the reduction can be established by using Gaussian anticoncentration to argue that f^{DV} is a good approximation of f. Formally, one can prove the following theorem.

Theorem 3.5 (Original DV lift, implicit in [DV21])**.** *Let $\mathcal{C}$ be a class of L-hidden-layer $\mathrm{poly}(d)$-sized ReLU networks mapping $\{0,1\}^d$ to $[0,1]$. There exists a class $\mathcal{C}^{DV}$ of $(L+2)$-hidden-layer $\mathrm{poly}(d)$-sized ReLU networks mapping $\mathbb{R}^d$ to $[0,1]$ such that the following holds. Suppose there is an efficient algorithm A capable of learning $\mathcal{C}^{DV}$ over $\mathcal{N}(0, \mathrm{Id}_d)$ up to squared loss $\frac{1}{64}$. Then there is an efficient algorithm B capable of weakly predicting $\mathcal{C}$ over $\mathrm{Unif}\{0,1\}^d$ with squared loss $\frac{1}{16}$.*



(a) $N_1 : \mathbb{R} \to [-1, 1]$



(b) $N_2 : \mathbb{R} \to [0, 1]$

Figure 1: Schematic plots of N_1 and N_2 in the $q = 2$ case, where $N_2'(z)$ may be realized as $\sum_{j \in [d]} N_2(z_j)$. Here, $\delta' = \Theta(\delta)$ where δ is the parameter from Lemmas 3.6 and 3.7.

We now show how to construct the gadgets N_1 and N_2, extending them to make them suitable for working with $\mathbb{Z}_q$ for general q as opposed to just $\{0,1\}$. These constructions utilize the simple but important property that piecewise linear functions on the real line are readily and efficently realized as linear combination of ReLUs.

Start by letting $I_0, I_1, \ldots, I_{q-1}$ be a partition of $\mathbb{R}$ into q consecutive intervals each of mass $1/q$ under $\mathcal{N}(0,1)$ (e.g., when $q=2$, $I_0 = (-\infty, 0)$ and $I_1 = (0, \infty)$). Note that these intervals will have differing lengths, and the shortest ones will be the ones closest to the origin. Still, by Gaussian anti-concentration, we know that each $|I_j| \geq \Theta(1/q)$. Let $\text{thres}_q : \mathbb{R} \to \mathbb{Z}_q$ be the piecewise constant function that takes on value k on I_k. Clearly, when $t \sim \mathcal{N}(0,1)$, $\text{thres}_q(t) \sim \text{Unif}(\mathbb{Z}_q)$. Let $R_1, \ldots, R_q$ be intervals such that $R_k \subseteq I_{k-1} \cup I_k$ and R_k contains the boundary point between I_{k-1} and I_k, and such that each R_k has mass δ/q for some $\delta \ll 1$ to be picked later. Let $S_1, \ldots, S_q$ be slightly larger intervals such that $R_k \subset S_k$ for each $k \in [q-1]$, and each S_k has mass $2\delta/q$. By Gaussian anti-concentration again, each $|S_k| \geq \Theta(\delta/q)$. Notice that by construction, $\mathbb{P}_{z \sim \mathcal{N}(0,1)}[z \in \cup_k R_k] = \delta$ and $\mathbb{P}_{z \sim \mathcal{N}(0,1)}[z \in \cup_k S_k] = 2\delta$.

Lemma 3.6. *Let $\delta > 0$, $q > 0$, and intervals I_k, R_k, S_k for $k \in \mathbb{Z}_q$ be as above. There exists a one-hidden-layer ReLU network $N_1 : \mathbb{R} \to \mathbb{R}$ with $O(q)$ units and weights of magnitude $O(q/\delta)$ such that $N_1(t) = \text{thres}_q(t)$ if $t \notin \cup_k R_k$.*

Proof. This can be done by considering the piecewise linear function that approximates the function thres_q by matching it exactly on $\mathbb{R} \setminus \cup_k R_k$, and interpolating linearly between values $k-1$ and k on the interval R_k for each $k \in [q-1]$. □

Lemma 3.7. *Let $\delta > 0$, $q > 0$, and intervals I_k, R_k, S_k for $k \in \mathbb{Z}_q$ be as above. There exists a one-hidden-layer ReLU network $N_2 : \mathbb{R} \to [0,1]$ with $O(q)$ units and weights of magnitude $O(q/\delta)$ such that*

$$N_2(t) \text{ is } \begin{cases} = 1 & \text{if } t \in \cup_k R_k \\ = 0 & \text{if } t \in \mathbb{R} \setminus \cup_k S_k \\ \geq 0 & \text{otherwise} \end{cases}.$$

Proof. Consider the piecewise linear function that is 0 on $\mathbb{R} \setminus \cup_k S_k$, is 1 on $\cup_k R_k$, and interpolates linearly between 0 and 1 (or 1 and 0) on $S_k \setminus R_k$ for every $k \in [q-1]$. Put differently, the graph of N_2 consists of a trapezoid on each S_k that achieves its maximum value of 1 on R_k. □

3.2 Saving One Hidden Layer via Compressibility

The starting point for exploiting compressibility to avoid a hidden layer in the lift is as follows. Compressibility lets us express $f(x)$ as $\sigma(h(x))$ for some $h : \mathbb{Z}_q^d \to T$ with a poly(d)-sized range $T \subseteq \mathbb{Z}$, and some $\sigma : T \to [0,1]$. So we can write

$$f(x) = \sigma(h(x)) = \sum_{t^* \in T} \sigma(t^*) \mathbb{1}[h(x) = t^*].$$

We would like a lifted function $f^{\triangle} : \mathbb{R}^d \to \mathbb{R}$ (where we introduce $f^{\triangle}$ as notation to distinguish our lift from the original DV lift, denoted f^{DV}) such that $f^{\triangle}(z)$ behaves like $\sigma(h(\text{thres}_q(z)))$ except when N_2 indicates that $N_1 \neq \text{thres}_q$, in which case we want $f^{\triangle}(z) = 0$. To this end, we might hope to write

$$f^{\triangle}(z) = \sum_{t^* \in T} \sigma(t^*) \mathbb{1}[h(N_1(z)) = t^*] \mathbb{1}[\forall j : N_2(z_j) < 1].$$

Suppose that we could build a one-hidden-layer network $N_3(s_1, \ldots, s_d; t)$ that behaves like $\mathbb{1}[t = 0]\mathbb{1}[\forall j : s_j < 1]$. Then we could realize $f^{\triangle}$ as

$$f^{\triangle}(z) = \sum_{t^* \in T} \sigma(t^*) N_3(N_2(z_1), \ldots, N_2(z_d);\ h(N_1(z)) - t^*).$$

Notice that whenever $N_2(z_j) = 1$ for any coordinate j, this expression vanishes. Otherwise, we know that $h(N_1(z)) = h(\mathrm{thres}_q(z))$, which takes values in T, so that only the summand with $t^* = h(\mathrm{thres}_q(z))$ survives and the expression simplifies to $f(\mathrm{thres}_q(z))N_3(N_2(z_1), \ldots, N_2(z_d); 0)$. It is not hard to show that this is sufficient to let us complete the required reduction. Moreover, because N_3 is a one-hidden-layer network in its arguments, and because both $h \circ N_1$ and N_2 have at most L hidden layers (for $h \circ N_1$, one comes from N_1 and $L - 1$ from h; for N_2, it itself has just one hidden layer), this implementation of $f^{\triangle}$ would have only $L + 1$ hidden layers.

Slightly more generally, one can show that it would suffice to build a one-hidden-layer network N_3 with the following properties:

$$N_3(s_1, \ldots, s_d; t) = \begin{cases} 0 & \text{if } \exists j : s_j = 1 \\ 0 & \text{if } t \in \mathbb{Z} \setminus \{0\} \\ 1 & \text{if } \forall j : s_j = 0 \text{ and } t = 0 \end{cases} \tag{4}$$

Unfortunately, most natural attempts to construct N_3 with such ideal properties — in particular, all formulations of N_3 purely as a function of two variables, $\sum_j s_j$ and t, which was the approach taken in [DV21] — run into difficulties and appear to require *two* hidden layers (see Appendix A for discussion). One approach that does almost work, however, comes at the cost of exponential size. Let $\psi(s_1, \ldots, s_d; t)$ be any function that vanishes whenever $t \in \mathbb{Z} \setminus \{0\}$ (for all $s_1, \ldots, s_d \in [0, 1]^d$). For simplicity, let us consider the $d = 3$ case. Consider the following expression that resembles the inclusion-exclusion formula:

$$\begin{aligned} &\psi(s_1, s_2, s_3; t) - \psi(1, s_2, s_3; t) - \psi(s_1, 1, s_3; t) - \psi(s_1, s_2, 1; t) \\ &+ \psi(s_1, 1, 1; t) + \psi(1, s_2, 1; t) + \psi(s_1, 1, 1; t) - \psi(1, 1, 1; t) \end{aligned} \tag{5}$$

Notice that whenever any $s_j = 1$, this expression vanishes identically. Moreover, for any $t \in \mathbb{Z} \setminus \{0\}$ (and any $s_1, \ldots, s_d$), the expression vanishes again because each summand vanishes. Thus the first two properties are satisfied; the third property turns out to be more subtle, and we will ignore it for the moment. The natural generalization of this expression to general d can be stated in the language of partial assignments.

Lemma 3.8. *Let $\psi : \mathbb{R}^d \to \mathbb{R}$ be any function. Let $\mathcal{P}_i$ denote the set of partial assignments $\gamma \in \{1, \star\}^d$ with i 1s. The expression*

$$\sum_{i=0}^{d} \sum_{\gamma \in \mathcal{P}_i} (-1)^i \psi_\gamma \tag{6}$$

vanishes whenever any $s_j = 1$. (We may view t as an additional parameter that is always left free, as in Eq. (5))

Proof. For concreteness, suppose $s_1 = 1$. Let $\mathcal{P}_i^\star$ (resp. $\mathcal{P}_i^1$) denote the set of $\gamma \in \mathcal{P}_i$ with $s_1 = \star$ (resp. $s_1 = 1$). For every $i \in \{0, \ldots, d-1\}$, we can form a bijection between $\mathcal{P}_i^\star$ and $\mathcal{P}_{i+1}^1$ using the map $\gamma \mapsto \gamma'$ where $\gamma' = (1, \gamma_2, \ldots, \gamma_d)$. When $s_1 = 1$, for every such pair (γ, γ'), we have $\psi_\gamma = \psi_{\gamma'}$, and moreover they occur in (6) with opposite signs. Thus the entire expression vanishes. ∎

Let us assume for now that ψ is picked suitably and the rest of the reduction goes through with this construction (as one can verify when we come to the proof of Theorem 3.3, this would indeed be the case). This construction has size 2^d, meaning that the resulting lifted functions would have size $S = \mathrm{poly}(2^d)$. But by Theorem 4.5, the SQ lower bound for the LWR functions over $\mathbb{Z}_q^n$ with $n = d$ and $q = \mathrm{poly}(n)$ scales as $q^{\Omega(n)} = 2^{\Omega(d \log d)} = S^{\Omega(\log \log S)}$, which is still superpolynomial in S. Thus after padding the dimension to $d' = 2^d$, this construction would actually still yield a superpolynomial SQ lower bound for two-hidden-layer ReLU networks over $\mathbb{R}^{d'}$.

Instead of pursuing this route, however, we give a more efficient construction that has size only slightly superpolynomial in d. The key idea is to restrict attention to those possibilities for $(s_1, \ldots, s_d) = (N_2(z_1), \ldots, N_2(z_d))$ that are the most likely. Specifically, if $m = \omega_d(1)$ is the size parameter from Theorem 3.3, then by setting δ in Lemmas 3.6 and 3.7 appropriately, we can ensure that with overwhelming probability over $z \sim \mathcal{N}(0, \mathrm{Id})$, no more than m of the $N_2(z_j)$ are simultaneously 1. Accordingly, we focus on constructing N_3 such that

$$N_3(s_1, \ldots, s_d; t) = \begin{cases} 0 & \text{if between 1 and } m \text{ of the } s_i \text{ are 1} \\ 0 & \text{if } t \in \mathbb{Z} \setminus \{0\} \\ 1 & \text{otherwise} \end{cases}.$$

We now describe a $d^{\Theta(m)}$-sized construction for N_3 that satisfies the first and second properties exactly, and "approximately" satisfies the third in the sense that it takes on a nonzero value with nonnegligible probability over its inputs. As we will see later, this turns out to be enough for the reduction to go through. The construction retains the spirit of using a linear combination of partial restrictions.

Lemma 3.9 (Main lemma)**.** *Let $m = m(d) = \omega_d(1)$ be a size parameter. Let $\mathcal{A}$ denote the set of all partial assignments $\alpha \in \{0, 1, \star\}^d$ for which $|S(\alpha)| = m$ and $w(\alpha) = 1$. Let $\mathcal{B}$ denote the set of all sorted partial assignments given by refining some element of $\mathcal{A}$ and sorting. Given $i, j \geq 0$, let $\mathcal{B}_{i,j}$ denote the set of $\overline{\beta} \in \mathcal{B}$ for which $|S(\overline{\beta})| = i$ and $w(\overline{\beta}) = j$. For any symmetric function $\psi : \mathbb{R}^d \to \mathbb{R}$, define the function*

$$\psi^* \triangleq \psi - \sum_{i=0}^{m} \sum_{j=1}^{m+1-i} (-1)^{m-i} \cdot \lambda_{i+j} \sum_{\overline{\beta} \in \mathcal{B}_{i,j}} \psi_{\overline{\beta}}, \qquad \text{for } \lambda_k \triangleq \binom{d-k-1}{m-k+1}$$

Then

(a) $|\mathcal{B}| \leq \binom{d}{m}(d-m) \cdot 3^m$

(b) ψ^* *is symmetric*

(c) $\psi^*_\alpha : \mathbb{R}^d \to \mathbb{R}$ *is the identically zero function for all* $\alpha \in \mathcal{A}$.

Lemma 3.10. *Let*

$$\psi(s_1, \ldots, s_d; t) = \sum_{i=1}^{d} \mathrm{ReLU}\left(t - \left(s_i - \frac{1}{d-1} \sum_{j \neq i} s_j\right)\right) - \mathrm{ReLU}(dt),$$

viewed as a function of $s_1, \ldots, s_d$ parameterized by t, and let ψ^* be as above. Define $N_3(s_1, \ldots, s_d; t) = \psi^*(s_1, \ldots, s_d; t)$. Then

(a) $N_3(s_1, \ldots, s_d; t) = 0$ *for any $t \in \mathbb{R}$ if between 1 and m of the s_j are 0*

(b) $N_3(s_1, \ldots, s_d; t) = 0$ *for any* $s_1, \ldots, s_d \in [0,1]^d$ *if* $t \in \mathbb{Z} \setminus \{0\}$

(c) N_3 *has size at most* d^{2m}

(d) $N_3(\underbrace{0, \ldots, 0}_{d-1}, s; 0) = s$ *for any* $s \in [0, \frac{1}{d}]$.

Before proceeding to the proofs of Lemmas 3.9 and 3.10, let us see how to use them to prove Theorem 3.3.

Proof of Theorem 3.3. For each $f \in \mathcal{C}$ given by $f = \sigma \circ h$, let $f^\triangle \in \mathcal{C}^\triangle$ be given by

$$f^\triangle(z) = \sum_{t^* \in T} \sigma(t^*) N_3(N_2(z_1), \ldots, N_2(z_d);\ h(N_1(z)) - t^*), \tag{7}$$

where N_1 and N_2 are from Lemmas 3.6 and 3.7, with the δ parameter set to d^{-10m}, and N_3 is from Lemma 3.10. This is an $(L+1)$-hidden layer network since $h \circ N_1$ and N_2 each have at most L hidden layers, and N_3 adds an additional layer. By Lemma 3.10(c), the size of this network is $S = d^{\Theta(m)}$. Note that whenever z is such that $N_2(z_1), \ldots, N_2(z_d) < 1$, then:

- $N_1(z) = \text{thres}_q(z)$, and so $h(N_1(z)) = h(\text{thres}_q(z))$ takes only integer values in $T = h(\mathbb{Z}_q^d)$; and
- the only t^* for which one of the summands in Eq. (7) is potentially nonzero is the one given by $t^* = h(\text{thres}_q(z))$.

Thus in this case $f^\triangle$ simplifies to

$$\begin{aligned} f^\triangle(z) &= \sigma(h(\text{thres}_q(z)))\ N_3(N_2(z_1), \ldots, N_2(z_d);\ 0) \\ &= f(\text{thres}_q(z))\ N_3(N_2(z_1), \ldots, N_2(z_d);\ 0). \end{aligned} \tag{8}$$

Further, for z such that between 1 and m of the $N_2(z_j)$ are 1, we know that $\psi(N_2(z_1), \ldots, N_2(z_d);\ t) = 0$ identically (for all $t \in \mathbb{R}$), so in this case $f^\triangle(z) = 0$. And finally, for z such that more than m of the $N_2(z_j)$ are 1, we have no guarantees on the behavior of $f^\triangle$, but as we now show, we have set parameters such that this case occurs only with negligible probability, and we can pretend that 0 is still a valid label in this case. Indeed, by standard Gaussian anti-concentration, for each coordinate z_j we have $\mathbb{P}_{z_j}[N_2(z_j) = 1] = \mathbb{P}_{z_j}[z_j \in \cup_k R_k] = \delta = d^{-10m}$. The number of coordinates j for which $N_2(z_j) = 1$ thus follows a binomial distribution $B(d, d^{-10m})$, which has a decreasing pdf with unique mode at $\lfloor (d+1)d^{-10m} \rfloor = 0$. Thus the probability of having at least m 1s is at most

$$\sum_{i=m}^{d} \binom{d}{i} (d^{-10m})^i (1 - d^{-10m})^{d-i} \leq (d - m + 1) \binom{d}{m} d^{-10m^2} \leq d d^m d^{-10m^2} \leq d^{-9m^2} \tag{9}$$

for sufficiently large d. This is negligibly small not only in d but in the size of the network, $S = d^{\Theta(m)}$.

We now describe the reduction. For each labeled example (x, y) that the discrete learner B receives, where $x \sim \text{Unif}(\mathbb{Z}_q^d)$ and $y = f(x)$ for an unknown $f \in \mathcal{C}$, B forms a labeled example $(z, \widetilde{y})$ for the Gaussian learner A as follows. For each coordinate $j \in [d]$, z_j is drawn from $\mathcal{N}(0,1)$ conditioned on $z_j \in I_{x_j}$. Notice that this way $\text{thres}_q(z) = x$, and the marginal distribution on z is exactly $\mathcal{N}_d$. The modified label is given by

$$\widetilde{y} = \widetilde{y}(y, z) = \begin{cases} 0 & \text{if more than } m \text{ of the } N_2(z_j) \text{ are } 1 \\ 0 & \text{if between 1 and } m \text{ of the } N_2(z_j) \text{ are } 1 \\ y\ N_3(N_2(z_1), \ldots, N_2(z_d);\ 0) & \text{otherwise} \end{cases} \tag{10}$$

Note that in the bottom two cases, $\widetilde{y} = f^{\triangle}(z)$ exactly; in the top case $\widetilde{y}$ is in general inconsistent with $f^{\triangle}$, but as we have seen, this case occurs with $\mathrm{negl}(S)$ probability. In particular, with overwhelming probability, no $\mathrm{poly}(S)$-time algorithm will ever see non-realizable samples.

So B can feed these new labeled examples $(z, \widetilde{y})$ to A. Suppose A outputs a hypothesis $\widehat{f} : \mathbb{R}^d \to \mathbb{R}$ such that $\mathbb{E}_{z \sim N_d}[(\widehat{f}(z) - f^{\triangle}(z))^2] \leq \varepsilon$. We need to show B can convert this hypothesis into a nontrivial one for its discrete problem. We first define a "good region" $G \subseteq \mathbb{R}^d$ where $f^{\triangle}$ is guaranteed to be nonzero and nontrivially related to the original f by saying $z \in G$ iff $N_2(z_1), \ldots, N_2(z_{d-1}) = 0$, and $N_2(z_d) \in (\frac{1}{2d}, \frac{1}{d})$. Observe that when $z \in G$, by Eq. (8) and Lemma 3.10(d) we have

$$\begin{aligned} f^{\triangle}(z) &= f(\mathrm{thres}_q(z)) N_3(N_2(z_1), \ldots, N_2(z_{d-1}), N_2(z_d); 0) \\ &= f(x) N_3(0, \ldots, 0, N_2(z_d); 0) \\ &= y N_2(z_d), \end{aligned} \tag{11}$$

where we use the fact that $\mathrm{thres}_q(z) = x$, so that $f(\mathrm{thres}_q(z)) = f(x) = y$. Let us compute the probability mass of G. For coordinates $j \in [d-1]$, note that $\mathbb{P}[N_2(z_j) = 0] = \mathbb{P}[z_j \notin \cup_k S_k] = 1 - 2\delta = 1 - d^{-\Theta(m)}$. For z_d, we need a lower bound on the probability that $N_2(z_d) \in (\frac{1}{2d}, \frac{1}{d})$. Consider the behavior of N_2 on just the interval S_k that is closest to the origin (which will be $k = \lceil q/2 \rceil$): it changes linearly from 0 to 1 (and again from 1 to 0) on $S_k \setminus R_k$. It is not hard to see that N_2 takes values in $(\frac{1}{2d}, \frac{1}{d})$ on a $O(1/d)$ fraction of S_k. Since the Gaussian pdf will be at least some constant on all of S_k, the probability that z_d lands in this fraction of S_k is $\Omega(|S_k|/d) = \Omega(\delta/qd) \geq d^{-\Theta(m)}$. Overall, we get that

$$\mathbb{P}[z \in R] = \mathbb{P}\left[N_2(z_d) \in \left(\frac{1}{2d}, \frac{1}{d}\right)\right] \prod_{j \in [d-1]} \mathbb{P}[N_2(z_j) = 0] \geq (1 - d^{-\Theta(m)})^{d-1} d^{-\Theta(m)} = d^{-\Theta(m)},$$

which is still $1/\mathrm{poly}(S)$ and hence non-negligible in the size S of the network.

The discrete learner B can now adapt $\widehat{f}$ as follows. Given a fresh test point $x \sim \mathrm{Unif}(\mathbb{Z}_q^d)$, the learner forms z such that for each coordinate $j \in [d]$, z_j is drawn from $\mathcal{N}(0,1)$ conditioned on $z_j \in I_{x_k}$; for brevity, we shall denote the random variable z conditioned on x (formed in this way) by $z|x$. If $z \in G$, then B predicts $\widehat{y} = \frac{\widehat{f}(z)}{N_2(z_d)}$ (recall that when $z \in z$, $N_2(z_d) > \frac{1}{2d}$), and otherwise

it simply predicts $\widetilde{y} = \frac{1}{2}$. The square loss of this predictor is given by

$$\begin{aligned}
\mathop{\mathbb{E}}_{x\sim\mathrm{Unif}(\mathbb{Z}_q^d)}[(\widehat{y} - f(x))^2] &= \mathop{\mathbb{E}}_{x}\mathop{\mathbb{E}}_{z|x}[(\widehat{y} - f(x))^2] \\
&= \mathop{\mathbb{E}}_{x,z|x}[(\widehat{y} - f(x))^2 \mid z \in G]\,\mathbb{P}[z \in G] + \mathop{\mathbb{E}}_{x,z|x}[(\widehat{y} - f(x))^2 \mid z \notin G]\,\mathbb{P}[z \notin G] \\
&= \mathop{\mathbb{E}}_{x,z|x}\left[\left(\frac{\widehat{f}(z)}{N_2(z_d)} - f(x)\right)^2 \mid z \in G\right]\mathbb{P}[z \in G] + \mathop{\mathbb{E}}_{x,z|x}\left[\left(\frac{1}{2} - f(x)\right)^2 \mid z \notin G\right]\mathbb{P}[z \notin G] \\
&= \mathop{\mathbb{E}}_{x,z|x}\left[\left(\frac{\widehat{f}(z)}{N_2(z_d)} - \frac{f^{\triangle}(z)}{N_2(z_d)}\right)^2 \mid z \in G\right]\mathbb{P}[z \in G] + \mathop{\mathbb{E}}_{x}\left[\left(\frac{1}{2} - f(x)\right)^2\right]\mathbb{P}[z \notin G] \\
&\qquad\qquad \text{(by Eq. (11), when } z \in G,\ f^{\triangle}(z) = f(x)N_2(z_d)) \\
&< 4d^2\,\mathop{\mathbb{E}}_{z}[(\widehat{f}(z) - f^{\triangle}(z))^2 \mid z \in G]\,\mathbb{P}[z \in G] + \mathop{\mathbb{E}}_{x}\left[\left(\frac{1}{2} - f(x)\right)^2\right]\mathbb{P}[z \notin G] \\
&\qquad\qquad \text{(when } z \in G,\ N_2(z_d) > \tfrac{1}{2d}) \\
&\le 4d^2\,\mathop{\mathbb{E}}_{z}[(\widehat{f}(z) - f^{\triangle}(z))^2] + \mathop{\mathbb{E}}_{x}\left[\left(\frac{1}{2} - f(x)\right)^2\right]\mathbb{P}[z \notin G] \\
&\le 4d^2\varepsilon + \mathop{\mathbb{E}}_{x}\left[\left(\frac{1}{2} - f(x)\right)^2\right]\mathbb{P}[z \notin G] \\
&= \mathop{\mathbb{E}}_{x}\left[\left(\frac{1}{2} - f(x)\right)^2\right] + 4d^2\varepsilon - \mathop{\mathbb{E}}_{x}\left[\left(\frac{1}{2} - f(x)\right)^2\right]\mathbb{P}[z \in G].
\end{aligned}$$

In the case of the hard classes $\mathcal{C}$ that we consider, we may assume without loss of generality that $\mathbb{E}_{x\sim\mathrm{Unif}(\mathbb{Z}_q^d)}[(\frac{1}{2} - f(x))^2] \ge 1/\operatorname{poly}(d)$, since otherwise the problem of learning $\mathcal{C}$ is trivial (in fact, in our applications we will have $\mathbb{E}_{x\sim\mathrm{Unif}(\mathbb{Z}_q^d)}[(\frac{1}{2} - f(x))^2] = \Theta(1)$). This means that by taking

$$\varepsilon = \mathbb{P}[z \in G]/\operatorname{poly}(d) = d^{-\Theta(m)}/\operatorname{poly}(d) = d^{-\Theta(m)}$$

sufficiently small (but still $1/\operatorname{poly}(S)$), we may ensure that the square loss of the discrete learner B is at most $\mathbb{E}_{x\sim\mathrm{Unif}(\mathbb{Z}_q^d)}[(\frac{1}{2} - f(x))^2] - d^{-\Theta(m)}$, as desired. □

Remark 3.11. The only property of the Gaussian $\mathcal{N}(0, \mathrm{Id}_d)$ used crucially in the proof above is that it is a product distribution $P = \otimes_{i\in[d]} P_i$ where each P_i is suitably anti-concentrated. By some simple changes to the parameters of N_1, N_2 and N_3 (depending on P), the proof can be made to work more generally for such distributions P.

3.3 Proofs of Lemmas 3.9 and 3.10

We now detail the proofs involved in the construction of the gadget N_3.

Proof of Lemma 3.9. Note that $|\mathcal{A}| = \binom{d}{m}(d-m)$. Any partial assignment β has at most $3^{|S(\beta)|}$ refinements, and $\mathcal{B}$ is a subset of all refinements of partial assignments from $\mathcal{A}$, so $|\mathcal{B}| \le \binom{d}{m}(d-m)\cdot 3^m$.

For the remaining parts of the lemma, it will be useful to observe that $\mathcal{B}$ consists exactly of all partial assignments with i free variables and j 1s for any $0 \le i \le m$ and $j \ge 1$ satisfying $i + j \le m + 1$.

To prove the second part of the lemma, it suffices to show that

$$\sum_{\overline{\beta}\in\mathcal{B}_{i,j}} h_{\overline{\beta}} \tag{12}$$

is symmetric for all i, j. As transpositions generate the symmetric group on d elements, it suffices to show that (12) is invariant under swapping two input coordinates, call them $a, b \in [d]$. For all $\overline{\beta} \in \mathcal{B}_{i,j}$ for which a, b are either both present or both absent in $S(\overline{\beta})$, this clearly does not affect the value of $h_{\overline{\beta}}$. Now consider the set S_a (resp. S_b) of partial assignments $\overline{\beta} \in \mathcal{B}_{i,j}$ for which only a (resp. only b) is present in $S(\overline{\beta})$. There is a clear bijection $f : S_a \to S_b$: given $\overline{\beta} \in S_a$, swap the a- and b-th entries, and vice-versa, and for any $\overline{\beta} \in S_a$, the function $h_{\overline{\beta}} + h_{f(\overline{\beta})}$ is unaffected by the swapping of input coordinates a, b. This concludes the proof of the second part of the lemma.

Finally, to prove the third part of the lemma, it suffices to verify it for a single $\alpha \in \mathcal{A}$, as h^* is symmetric. So consider $\overline{\alpha} = \{1, 0, \cdots, 0, \star, \cdots, \star\}$. We apply (2) to get

$$\begin{aligned} h^*_{\overline{\alpha}} &= h_{\overline{\alpha}} - \sum_{i=0}^{m} \sum_{j=1}^{m+1-i} (-1)^{m-i} \cdot \lambda_{i+j} \sum_{\overline{\beta} \in \mathcal{B}_{i,j}} h_{\overline{\alpha} \searrow \overline{\beta}} \\ &= h_{\overline{\alpha}} - \sum_{\overline{\gamma} \in \mathcal{B} \cap \mathrm{App}(\alpha) \text{ sorted}} h_{\overline{\gamma}} \cdot \sum_{i=0}^{m} \sum_{j=1}^{m+1-j} (-1)^{m-i} \cdot \lambda_{i+j} \sum_{\overline{\beta} \in \mathcal{B}_{i,j}} \mathbb{1}[\overline{\overline{\alpha} \searrow \overline{\beta}} = \overline{\gamma}] \end{aligned} \tag{13}$$

Note that for $\overline{\gamma} = \overline{\alpha}$, the only $\overline{\beta} \in \mathcal{B}$ for which $\overline{\overline{\alpha} \searrow \overline{\beta}} = \overline{\gamma}$ is $\overline{\beta} = \overline{\alpha}$. Indeed, for $\overline{\beta}$ to be such that $\overline{\overline{\alpha} \searrow \overline{\beta}} = \overline{\alpha}$, it must have $S(\overline{\beta}) = S(\overline{\alpha})$ and exactly one 1, from which it follows that $\overline{\beta} = \overline{\alpha}$. Since $\overline{\alpha} \in \mathcal{B}_{m,1}$, its coefficient in (13) is given by

$$(-1)^{m-m} \cdot \lambda_{m+1} = 1,$$

and so the h_α in (13) cancels with the $\overline{\gamma} = \overline{\alpha}$-th summand in (13).

In the rest of the proof, we can thus focus on sorted $\overline{\gamma} \in \mathcal{B} \cap \mathrm{App}(\alpha) \backslash \{\overline{\alpha}\}$. Note that such $\overline{\gamma}$ satisfy

$$|S(\overline{\gamma})| < m. \tag{14}$$

To see this, recall that any $\overline{\gamma} \in \mathcal{B}$ with $|S(\overline{\gamma})| = m$ must have exactly one 1, and since $\overline{\gamma} \in \mathrm{App}(\overline{\alpha})$ it must be that $\overline{\gamma}$ must have $S(\overline{\gamma}) = S(\overline{\alpha})$ and so $\overline{\gamma} = \overline{\alpha}$.

Observe that we must have $\overline{\gamma}_1 = 1$. Indeed, it cannot be 0 because $\overline{\gamma}$ is sorted and has at least one 1. It also cannot be $\star$. To see this, consider any $\overline{\beta}$ for which $\overline{\overline{\alpha} \searrow \overline{\beta}} = \overline{\gamma}$. If we had $\overline{\beta}_1 \neq \star$, then clearly $\overline{\gamma}_1 \neq \star$. If we had $\overline{\beta}_1 = \star$, then $(\overline{\alpha} \searrow \overline{\beta})_1 = 1$ (as $\overline{\alpha}_1 = 1$), so $\overline{\gamma} = \overline{\overline{\alpha} \searrow \overline{\beta}}$ must also have first entry given by 1.

We are now ready to calculate the coefficient of $h_{\overline{\gamma}}$ (for each $\overline{\gamma} \in \mathcal{B} \cap \mathrm{App}(\alpha) \backslash \{\overline{\alpha}\}$) in (13) by adding the coefficients of all the $\overline{\beta} \in \mathcal{B}$ for which

$$\overline{\overline{\alpha} \searrow \overline{\beta}} = \overline{\gamma}. \tag{15}$$

First let us consider the contribution of $\overline{\beta} \in \mathcal{B}$ for which $\overline{\beta}_1 = 1$. Observe that such $\overline{\beta}$ must have exactly $w(\overline{\gamma})$ 1s. Furthermore, such a $\overline{\beta}$ is an element of $\mathcal{B}$ if and only if it has at most $m + 1 - w(\overline{\gamma})$ free variables, and the set of free variables in $\overline{\beta}$ must be $S(\overline{\gamma}) \cup V$ where V is any subset of $[d] \backslash (\{1\} \cup S(\overline{\alpha}))$. The contribution of all such $\overline{\beta}$ to the coefficient of $h_{\overline{\gamma}}$ in (13) is thus

$$\sum_{i=|S(\overline{\gamma})|}^{m+1-w(\overline{\gamma})} (-1)^{m-i} \cdot \lambda_{i+w(\overline{\gamma})} \cdot \binom{d-m-1}{i-|S(\overline{\gamma})|}, \tag{16}$$

where here the index i denotes the total number of free variables in $\overline{\beta}$, and the factor of $\binom{d-m-1}{i-|S(\overline{\gamma})|}$ is the number of ways to choose V.

It remains to consider the contribution from $\overline{\beta} \in \mathcal{B}$ for which $\overline{\beta}_1 \neq 1$. First note that clearly we cannot have $\overline{\beta}_1 = 0$, as $\overline{\beta}$ is sorted and has at least one 1 because it lies in $\mathcal{B}$. The only possibility is $\overline{\beta}_1 = \star$, which we split into two cases based on $w(\overline{\gamma})$.

Case 1: $w(\overline{\gamma}) = 1$**.** In this case, we claim that there are no $\overline{\beta} \in \mathcal{B}$ simultaneously satisfying (15) and $\overline{\beta}_1 = \star$. Suppose to the contrary. Then such a $\overline{\beta}_1$ must have at least one 1 in some other entry (as $\overline{\beta} \in \mathcal{B}$), but this would imply that the resolution $\overline{\alpha} \searrow \overline{\beta}$ has at least two 1s, a contradiction. The total coefficient of $h_{\overline{\gamma}}$ in this case is thus exactly given by (16). Upon substituting $w(\overline{\gamma}) = 1$, this simplifies to

$$\sum_{i=|S(\overline{\gamma})|}^{m+1-w(\overline{\gamma})} (-1)^{m-i} \cdot \lambda_{i+1} \cdot \binom{d-m-1}{i-|S(\overline{\gamma})|} = \sum_{i=|S(\overline{\gamma})|}^{m+1-w(\overline{\gamma})} (-1)^{m-i} \cdot \binom{d-i-2}{d-m-2} \cdot \binom{d-m-1}{i-|S(\overline{\gamma})|} = 0,$$

where in the last step we use Lemma B.1 (which we can apply because of (14)).

Case 2: $w(\overline{\gamma}) > 1$**.** Observe that we must have $w(\overline{\beta}) = w(\overline{\gamma}) - 1$ (as the only entry of $\overline{\alpha}$ equal to 1 is the first entry, and the first entry of $\overline{\beta}$ is $\star$). As $w(\overline{\gamma}) - 1 > 0$ in the current case, such a $\overline{\beta}$ is an element of $\mathcal{B}$ if and only if it has at most $m + 2 - w(\overline{\gamma})$ free variables, and the set of free variables in $\overline{\beta}$ must be $\{1\} \cup S(\overline{\gamma}) \cup V$ where V is any subset of $[d] \backslash (\{1\} \cup S(\overline{\alpha}))$. Thus in this second case, the contribution of all $\overline{\beta}$ with $\overline{\beta}_1 = \star$ to the coefficient of $h_{\overline{\gamma}}$ in (13) is

$$\sum_{i=|S(\overline{\gamma})|+1}^{m+2-w(\overline{\gamma})} (-1)^{m-i} \cdot \lambda_{i+w(\overline{\gamma})-1} \cdot \binom{d-m-1}{i-|S(\overline{\gamma})|-1} = \sum_{j=|S(\overline{\gamma})|}^{m+1-w(\overline{\gamma})} (-1)^{m-j-1} \cdot \lambda_{j+w(\overline{\gamma})} \cdot \binom{d-m-1}{j-|S(\overline{\gamma})|}, \quad (17)$$

where here the index i denotes the total number of free variables in $\overline{\beta}$, the factor of $\binom{d-m-1}{i-|S(\overline{\gamma})|-1}$ is the number of ways to choose V (note that $|V| = i - |S(\overline{\gamma})| - 1$), and in the second expression we made the change of variable $j = i - 1$. We conclude that in this case, the coefficient of $h_{\overline{\gamma}}$ in (13) is given by the sum of (16) and (17), which is 0.

Overall, we conclude that the entire RHS of (13) vanishes for $\alpha \in \mathcal{A}$, proving the third part of the lemma. □

The next lemma formally constructs N_3 and verifies that it has the required properties, is of acceptable size, and that it takes on nonzero values on a significant part of its domain.

Proof of Lemma 3.10. Part (a) follows directly from Lemma 3.9(c). Part (b) follows by verifying that for any $t \in \mathbb{Z} \setminus \{0\}$, $\psi(s_1, \ldots, s_d; t) = 0$ for any $s_1, \ldots, s_d \in [0, 1]^d$; this means that ψ^*, which is a combination of partial restrictions of ψ, also vanishes for such t. First suppose that t is a positive integer. Observe that $t \geq 1$ while $s_i - \frac{1}{d-1} \sum_{j \neq i} s_j \in [-1, 1]$, so each ReLU in the definition of ψ is activated and we get

$$\psi(s_1, \ldots, s_d; t) = \sum_{i=1}^{d} \left[t - \left(s_i - \frac{1}{d-1} \sum_{j \neq i} s_j \right) \right] - dt = -\sum_{i=1}^{d} \left(s_i - \frac{1}{d-1} \sum_{j \neq i} s_j \right) = 0.$$

Next suppose that t is a negative integer. Then $t \leq -1$ while $s_i - \frac{1}{d-1} \sum_{j \neq i} s_j \in [-1, 1]$, so each ReLU in the definition of h is inactive and we get $\psi(s_1, \ldots, s_d; t) = 0$.

For part (c), observe that by the size bound in Lemma 3.9(a) and the fact that ψ contains $O(d)$ ReLUs, the size of N_3 may be bounded by

$$S \le O(d) \cdot (\binom{d}{m}(d-m) \cdot 3^m + 1) \le O(d)(\frac{d^{m+1} \cdot 3^m}{m!} + 1) \le d^{m+2} \le d^{2m}$$

for m larger than some absolute constant.

It remains to prove part (d). For brevity, we will omit the parameter t and just refer to $\psi(0,\dots,0,s;t)$ and $\psi^*(0,\dots,0,s;t)$ as $\psi(0,\dots,0,s)$ and $\psi^*(0,\dots,0,s)$. We first compute $\psi(0,\dots,0,s)$: for $s \in [0,1]$,

$$\psi(0,\dots,0,s) = \mathrm{ReLU}(-s) + (d-1)\,\mathrm{ReLU}(\frac{1}{d-1} \cdot s) = s.$$

Next, for any $\overline{\beta} \in \mathcal{B}$, if $w(\overline{\beta}) = j$ for some $0 \le j \le m+1$, then if $\overline{\beta}_d = \star$,

$$\begin{aligned}
&\psi_{\overline{\beta}}(0,\dots,0,s) \\
&\quad = \psi(\underbrace{1,\dots,1}_{j},\underbrace{0,\cdots 0}_{d-j-1},s) \\
&\quad = j \cdot \mathrm{ReLU}\left(-1 + \frac{1}{d-1}(j-1+s)\right) + (d-j-1) \cdot \mathrm{ReLU}\left(\frac{1}{d-1} \cdot j + \frac{1}{d-1} \cdot s\right) \\
&\qquad + \mathrm{ReLU}\left(-s + \frac{1}{d-1} \cdot j\right) \\
&\quad = \frac{d-j-1}{d-1} \cdot (j+s) + \mathrm{ReLU}\left(-s + \frac{1}{d-1} \cdot j\right)
\end{aligned}$$

Note that when $s \in [0, 1/(d-1)]$, because $j \ge 1$ (as $\overline{\beta} \in \mathcal{B}$) this simplifies to

$$= \frac{(d-j-s)j}{d-1}.$$

On the other hand, if $\overline{\beta}_d \in \{0,1\}$, then

$$\begin{aligned}
\psi_{\overline{\beta}}(0,\dots,0,s) &= \psi(\underbrace{1,\dots,1}_{j},\underbrace{0,\cdots 0}_{d-j}) \\
&= j \cdot \mathrm{ReLU}\left(-1 + \frac{1}{d-1}(j-1)\right) + (d-j) \cdot \mathrm{ReLU}\left(\frac{1}{d-1} \cdot j\right) = \frac{(d-j)j}{d-1}.
\end{aligned}$$

As there are $\binom{d-1}{i-1}$ (resp. $\binom{d-1}{i}$) partial assignments in $\mathcal{B}_{i,j}$ for which $\overline{\beta}_d = \star$ (resp. $\overline{\beta}_d \in \{0,1\}$), we can thus explicitly compute $h^*(0,\dots,0,s)$ for $s \in [0,1/(d-1)]$ to be

$$\psi(0,\dots,0,s) - \sum_{i=0}^{m} \sum_{j=1}^{m+1-i} (-1)^{m-i} \binom{d-i-j-1}{m-i-j+1} \left(\binom{d-1}{i-1} \cdot \frac{(d-j-s)j}{d-1} + \binom{d-1}{i} \cdot \frac{(d-j)j}{d-1}\right).$$

By Lemma B.2, the double sum is equal to sero, so $h^*(0,\dots,0,s) = h(0,\dots,0,s) = s$ for $s \in [0,1/(d-1)]$ as claimed. □

4 Statistical Query Lower Bound

We prove a superpolynomial SQ lower bound (for general queries as opposed to only correlational or Lipschitz queries) for weakly learning two-hidden-layer ReLU networks under the standard Gaussian.

Theorem 4.1. *Fix any $\alpha \in (0,1)$. Any SQ learner capable of learning $\mathrm{poly}(d)$-sized two-hidden-layer ReLU networks under $\mathcal{N}(0,\mathrm{Id}_d)$ up to squared loss ε (for some sufficiently small $\varepsilon = 1/\mathrm{poly}(d)$) using bounded queries of tolerance $\tau \geq 2^{-(\log d)^{2-\alpha}}$ must use at least $\Omega(2^{2^{(\log d)^{\alpha}}}\tau^2) = d^{\omega(1)}\tau^2$ such queries.*

For instance, taking $\alpha = \frac{1}{2}$ gives a slightly subexponential (but super-quasipolynomial) in d query lower bound for queries of tolerance at least inverse quasipolynomial in d.

This theorem is proven using the following key reduction, which adapts the compressed DV lift (Theorem 3.3) to the SQ setting.

Theorem 4.2. *Let $q = \mathrm{poly}(d)$ be a modulus, and let $m = m(d) = \omega_d(1)$ be a size parameter. Let $\mathcal{C}$ be a class of compressible L-hidden-layer $\mathrm{poly}(d)$-sized ReLU networks mapping $\mathbb{Z}_q^d$ to $[0,1]$, and let $\mathcal{C}^{\triangle}$ be the lifted class of $(L+1)$-hidden-layer $d^{\Theta(m)}$-sized ReLU networks corresponding to $\mathcal{C}$, mapping $\mathbb{R}^d$ to $\mathbb{R}$ (as in Theorem 3.3). Suppose there is an SQ learner A capable of learning $\mathcal{C}^{\triangle}$ over $\mathcal{N}(0,\mathrm{Id}_d)$ up to squared loss $d^{-\Theta(m)}$ using queries of tolerance τ, where $\tau \geq d^{-\Theta(m^2)}$. Then there is an SQ learner B that, using the same number of queries of tolerance $\tau/2$, produces a weak predictor $\widetilde{B}$ for $\mathcal{C}$ over $\mathrm{Unif}(\mathbb{Z}_q^d)$ with advantage $d^{-\Theta(m)}$ over guessing the constant $1/2$ (in expectation over both the data and the internal randomness of $\widetilde{B}$).*

Proof. Recall that B is given SQ access to a distribution of pairs (x,y) where $x \sim \mathrm{Unif}(Z_q^d)$ and $y = f(x)$ for an unknown $f \in \mathcal{C}$. A can request estimates $\mathbb{E}[\phi(x,y)] \pm \tau$ for arbitrary bounded queries $\phi : \mathbb{Z}_q^d \times [0,1] \to [-1,1]$ and any desired τ. We know that given (x,y), the distribution of $(z,\widetilde{y})$, where $z = z(x)$ is defined by drawing each z_j from $\mathcal{N}(0,1)$ conditioned on $z_j \in I_{x_j}$ and $\widetilde{y} = \widetilde{y}(y,z)$ is as in Eq. (10)), is consistent with some $f^{\triangle} \in \mathcal{C}^{\triangle}$ except on a region of probability mass at most d^{-9m^2} (recall Eq. (9)). Suppose we could simulate SQ access to the distribution of $(z, f^{\triangle}(z))$ using only SQ access to that of $(x, f(x))$. Then by the argument in Theorem 3.3, simulating A on the $(z, f^{\triangle}(z))$ distribution would give us a weak predictor $\widetilde{B}$ for the distribution of $(x, f(x))$, satisfying

$$\mathbb{E}\left[\left(\widetilde{B}(x) - f(x)\right)^2\right] < \mathbb{E}\left[\left(\frac{1}{2} - f(x)\right)^2\right] - d^{-\Theta(m)}.$$

What we must describe is how B can simulate A's statistical queries. Say A requests an estimate $\mathbb{E}_z[\phi(z, f^{\triangle}(z))] \pm \tau$ for some query $\phi : \mathbb{R}^d \times \mathbb{R} \to [-1,1]$. Consider the query $\widetilde{\phi} : \mathbb{Z}_q^d \times [0,1] \to [-1,1]$ given by $\widetilde{\phi}(x,y) = \mathbb{E}_{z(x)}[\phi(z(x), \widetilde{y}(y, z(x)))]$. This function can be computed without any additional SQs, since the distribution of $(z,\widetilde{y}) = (z(x), \widetilde{y}(y,z(x)))$, given (x,y), is fully determined and known to B. Observe that

$$\mathop{\mathbb{E}}_{x,y} \widetilde{\phi}(x,y) = \mathop{\mathbb{E}}_{x,z(x)}[\phi(z(x), \widetilde{y}(y,z(x)))] = \mathop{\mathbb{E}}_{z,\widetilde{y}}[\phi(z,\widetilde{y})]. \tag{18}$$

We must also account for the difference between $\mathbb{E}_z[\phi(z, f^{\triangle}(z))]$ and $\mathbb{E}_{z,\widetilde{y}}[\phi(z,\widetilde{y})]$. But because the distributions only differ on a region of mass d^{-9m^2} and ϕ is bounded, we have

$$\left|\mathop{\mathbb{E}}_{z}[\phi(z, f^{\triangle}(z))] - \mathop{\mathbb{E}}_{z,\widetilde{y}}[\phi(z,\widetilde{y})]\right| \leq \Theta(d^{-9m^2}) \leq \frac{\tau}{2} \tag{19}$$

since we assumed $\tau \geq d^{-\Theta(m^2)}$. Putting together (18) and (19), we see that B can simulate A's query ϕ to within tolerance τ by querying $\widetilde{\phi}$ with tolerance $\tau/2$. ∎

Again, by a padding argument we can obtain a corollary similar to Corollary 3.4, for which we omit the formal statement. We will use such an argument in the proof of Theorem 4.1.

4.1 SQ lower bound via parities

We can obtain an SQ lower bound for two-hidden-layer ReLU networks by lifting the problem of learning parities under U_d, which is well-known to require exponentially many queries. More precisely, we show that an SQ learner for two-hidden-layer ReLU networks would yield an SQ algorithm for the problem of distinguishing an unknown parity from random labels.

Theorem 4.3 ([Kea98, BFJ+94])**.** *Consider an SQ algorithm given SQ access either to the distribution of labeled pairs (x, y) where $x \sim U_d$ and $y = \chi_S(x)$ for an unknown parity χ_S or to the randomly labeled distribution $U_d \times \mathrm{Unif}\{0,1\}$. Any algorithm capable of distinguishing between the two cases with probability $2/3$ using queries of tolerance τ requires at least $\Omega(2^d\tau^2)$ such queries.*

Lemma 4.4. *For every $S \subseteq [d]$, the parity function $\chi_S : \{0,1\}^d \to \{0,1\}$ can be implemented as a compressible one-hidden-layer ReLU network of $\mathrm{poly}(d)$ size.*

Proof. Recall that $\chi_S(x)$ evaluates to 1 if the Hamming weight of the bits of x in S is odd, and 0 otherwise, so that $\chi_S(x) = \sigma(\sum_{j\in S} x_j)$. This satisfies the definition of a compressible one-hidden-layer network with the inner depth-0 network being $x \mapsto \sum_{j\in S} x_j$ and $\sigma(t) = \mathbb{1}[t \text{ is odd}]$. □

We can now supply one proof of Theorem 4.1.

First proof of Theorem 4.1. Let $m = m(d) = \log^c d$ for $c = \frac{1}{\alpha} - 1$, and let $d' = d^m = 2^{\log^{c+1} d}$, so that $d = 2^{\log^{1/(1+c)} d'}$. By Lemma 4.4, the class $\mathcal{C}$ of parities on $\{0,1\}^d$ can be implemented by compressible one-hidden-layer $\mathrm{poly}(d)$-sized ReLU networks, and so the lifted class $\mathcal{C}^\triangle$ can be implemented by two-hidden-layer $d^{\Theta(m)}$-sized ReLU networks over $\mathbb{R}^d$. A padding argument lets us embed these classes into dimension d'. By using the predictor from Theorem 4.2 (with $q = 2$), we could obtain an SQ algorithm capable of distinguishing parities from random labels using queries of tolerance $\tau/2$, assuming $\tau \geq d^{-\Theta(m^2)} = 2^{-\log^{2c+1} d} = 2^{-\log^{\frac{2c+1}{c+1}} d'}$. By Theorem 4.3, the lower bound for learning parities is $\Omega(2^d\tau^2) = \Omega(2^{2^{\log^{1/(1+c)} d'}}\tau^2)$. Substituting $\alpha = \frac{1}{1+c}$ gives the result. □

But the SQ lower bound obtained this way via parities is somewhat unconvincing since there is a non-SQ algorithm capable of learning the lifted function class obtained from parities. Indeed, suppose we are given examples $(z, f^\triangle(z))$ where f is an unknown parity. We know that whenever z lands in the "good region" G from the proof of Theorem 3.3 (which happens with non-negligible probability), we have $f^\triangle(z) = f(\mathrm{sign}(z))N_2(z)$ (recall Eq. (11)). This means we can simply filter out all $z \notin G$ and form a clean data set of labeled points $(\mathrm{sign}(z), f(\mathrm{sign}(z)))$. The unknown f (and hence $f^\triangle$) can now be learnt by simple Gaussian elimination. In order to give a more convincing lower bound, we now provide an alternative proof based on LWR.

4.2 SQ lower bound via the LWR functions

Here we provide an alternative proof of Theorem 4.1 using the LWR functions. The hard function class obtained this way is not only *unconditionally* hard for SQ algorithms, it is arguably hard for non-SQ algorithms as well, since LWR is believed to be cryptographically hard.

We begin by stating an SQ lower bound for the LWR functions. This theorem is proven in Appendix C using a general formulation in terms of pairwise independent function families that may be of independent interest, communicated to us by Bogdanov [Bog21].

Theorem 4.5. *Let $\mathcal{C}_{\mathsf{LWR}}$ denote the $\mathsf{LWR}_{n,p,q}$ function class. Any SQ learner capable of learning $\mathcal{C}_{\mathsf{LWR}}$ up to squared loss $1/16$ under $\mathrm{Unif}(\mathbb{Z}_q^n)$ using queries of tolerance τ requires at least $\Omega(q^{n-1}\tau^2)$ such queries.*

The following lemma shows that the LWR functions may be realized as compressible one-hidden-layer ReLU networks.

Lemma 4.6. *For every $w \in \mathbb{Z}_q^n$, the LWR function $f_w : \mathbb{Z}_q^n \to \mathbb{Z}_p/p$ can be implemented as a compressible one-hidden-layer ReLU network of size $O(q^2 n)$.*

Proof. By definition, we have $f_w(x) = \frac{1}{p}\lfloor (w \cdot x) \bmod q \rceil_p$, which is a compressible one-hidden-layer ReLU network with the inner depth-0 network (i.e., affine function) being $w \mapsto w \cdot x$ and $\sigma(t) = \frac{1}{p}\lfloor t \bmod q \rceil_p$. The size bound follows by observing that for any $x \in \mathbb{Z}_q^n$, the quantity $w \cdot x$ is an integer in $\{0, \ldots, q^2 n\}$. □

We are ready for an alternative proof of Theorem 4.1.

Alternative proof of Theorem 4.1. Let n be the security parameter, and fix moduli $p, q \geq 1$ such that $p, q = \mathrm{poly}(n)$ and $p/q = \mathrm{poly}(n)$. Let $d = n$, so that the SQ lower bound from Theorem 4.5 is $\Omega(q^{n-1}) = d^{\Omega(d)} = 2^{\widetilde{\Omega}(d)}$. Let $m = m(d) = \log^c d$ for $c = \frac{1}{\alpha} - 1$, and let $d' = d^m = 2^{\log^{c+1} d}$, so that $d = 2^{\log^{1/(1+c)} d'}$. By Lemma 4.6, the $\mathsf{LWR}_{n,p,q}$ function class $\mathcal{C}_{\mathsf{LWR}}$ is implementable by one-hidden-layer ReLU networks over $\mathbb{Z}_q^d$ of size $\mathrm{poly}(n) = \mathrm{poly}(d)$. The result now follows by Theorem 4.2 and the same padding argument as in the proof based on parities. □

5 Cryptographic Hardness Based on LWR

In this section we show hardness of learning two-hidden-layer ReLU networks over Gaussian inputs based on LWR. This is a direct application of the compressed DV lift (Theorem 3.3) to the LWR problem, which is by definition a hard learning problem over $\mathrm{Unif}(\mathbb{Z}_q^n)$, or equivalently $\mathrm{Unif}(\mathbb{Z}_q^d)$ with $d = n$.

Theorem 5.1. *Let n be the security parameter, and fix moduli $p, q \geq 1$ such that $p, q = \mathrm{poly}(n)$ and $p/q = \mathrm{poly}(n)$. Let $d = n$. Let $c > 0$, $m = m(d) = \log^c d$ and $d' = d^m$. Suppose there exists a $\mathrm{poly}(d')$-time algorithm capable of learning $\mathrm{poly}(d')$-sized depth-2 ReLU networks under $\mathcal{N}(0, \mathrm{Id}_{d'})$ up to squared loss $1/\mathrm{poly}(d')$. Then there exists a $\mathrm{poly}(d') = 2^{\Theta(\log^{1+c} n)}$ time algorithm for $\mathsf{LWR}_{n,p,q}$.*

Proof of Theorem 5.1. By Lemma 4.6, we know that the class $\mathcal{C}_{\mathsf{LWR}}$ is implementable by compressible $\mathrm{poly}(d)$-sized one-hidden-layer ReLU networks over $\mathbb{Z}_q^d$, or, after padding, over $\mathbb{Z}_q^{d'}$. Let $\mathcal{C}_{\mathsf{LWR}}^{\triangle}$ denote the corresponding lifted class of $\mathrm{poly}(d')$-sized two-hidden-layer ReLU networks, padded to have domain $\mathbb{R}^{d'}$. Applying Corollary 3.4 to the assumed learner for $\mathcal{C}_{\mathsf{LWR}}^{\triangle}$, we obtain a $\mathrm{poly}(d')$-time weak predictor predictor for $\mathcal{C}_{\mathsf{LWR}}$, which readily yields a corresponding distinguisher for the $\mathsf{LWR}_{n,p,q}$ problem. Using the facts that $d' = d^m = 2^{\log^{1+c} d}$ and $d = n$, we may translate $\mathrm{poly}(d')$ into $2^{\Theta(\log^{1+c} n)}$, yielding the result. □

Remark 5.2. Note that the choice of $m = m(d) = \log^c d$ in Theorem 5.1 is purely for simplicity. By picking $m(d) = \omega_d(1)$ to be a suitably slow-glowing function of d, such as $\log^* d$, we can obtain a running time for the final LWR algorithm that is as mildly superpolynomial as we like.

In addition, as an immediate corollary of Lemma 4.6, we also obtain a hardness result for one-hidden-layer networks under $\mathrm{Unif}\{0,1\}^d$, improving on the hardness result of [DV21] (see Theorem 3.4 therein) for *two*-hidden-layer networks under $\mathrm{Unif}\{0,1\}^d$. For this application, we let $d = n \log q = \widetilde{O}(n)$, so that we may identify the domain $\mathbb{Z}_q^n$ with $\{0,1\}^d$ via the binary representation. This also identifies $\mathrm{Unif}(\mathbb{Z}_q^n)$ with $\mathrm{Unif}\{0,1\}^d$.

Corollary 5.3. *Let n, p, q be such that $p, q = \mathrm{poly}(n)$ and $p/q = \mathrm{poly}(n)$, and let $d = n \log q = \widetilde{O}(n)$. Suppose there exists an efficient algorithm for learning $\mathrm{poly}(d)$-sized one-hidden-layer ReLU networks under U_d up to squared loss $1/4$. Then there exists an efficient algorithm for $\mathsf{LWR}_{n,p,q}$.*

6 Hardness of Learning using Label Queries

The main result of this section is to show hardness of learning constant-depth ReLU networks over Gaussians from label queries:

Theorem 6.1. *Assume there exists a family of PRFs mapping $\{0,1\}^d$ to $\{0,1\}$ implemented by $\mathrm{poly}(d)$-sized L-hidden-layer ReLU networks. Then there does not exist an efficient learner that, given query access to an unknown $\mathrm{poly}(d)$-sized $(L+2)$-hidden-layer ReLU network $f : \mathbb{R}^d \to \mathbb{R}$, is able to output a hypothesis $h : \mathbb{R}^d \to \mathbb{R}$ such that $\mathbb{E}_{z \sim \mathcal{N}(0, \mathrm{Id}_d)}[(h(z) - f(z))^2] \le 1/16$.*

We first recall the classical connection between pseudorandom functions and learning from label queries (also known as membership queries in the Boolean setting), due to Valiant [Val84] (see e.g. [BR17, Proposition 12] for a modern exposition).

Lemma 6.2. *Let $\mathcal{C} = \{f_s\}$ be a family of PRFs from $\{0,1\}^d$ to $\{0,1\}$ indexed by the key s. Then there cannot exist an efficient learner L that, given query access to an unknown $f_s \in \mathcal{C}$, satisfies*

$$\mathop{\mathbb{P}}_{x,s}[L(x) = f_s(x)] \ge \frac{1}{2} + \frac{1}{\mathrm{poly}(d)},$$

where the probability is taken over the random key s, the internal randomness of A, and a random test point $x \sim \mathrm{Unif}\{0,1\}^d$.

There exist multiple candidate constructions of PRF families in the class TC^0 of constant-depth Boolean circuits built with AND, OR, NOT and threshold (or equivalently majority) gates. Because the majority gate can be simulated by a linear combination of ReLUs similar to N_1 from Lemma 3.6, any TC^0_L (meaning depth-L) function $f : \{0,1\}^d \to \{0,1\}$ may be implemented as a $\mathrm{poly}(d)$-sized L-hidden-layer ReLU network (see e.g. [VRPS21, Lemma A.3][3]). Thus we may leverage the following candidate PRF constructions in TC^0 for our hardness result:

- PRFs in TC^0_4 based on the decisional Diffie-Hellman (DDH) assumption [KL01] (improving on [NR97]), yielding hardness for depth-6 ReLU networks
- PRFs in TC^0 based on Learning with Errors [BPR12, BP14], yielding hardness for depth-$O(1)$ ReLU networks

Note that depth 4 is the shallowest depth for which we have candidate PRF constructions based on widely-believed assumptions, and the question of whether there exist PRFs in TC^0_3 is a longstanding open question in circuit complexity [Raz92, HMP+93, RR97, KL01]. Under less widely-believed assumptions, [BIP+18] have also proposed candidate PRFs in ACC^0_3.

We can now complete the proof of Theorem 6.1. Since pseudorandom functions are not necessarily compressibile, we will simply use the original DV lift (Theorem 3.5).

[3] Note that what the authors term a depth-$(L+1)$ network is in fact an L-hidden-layer network in our terminology.

Proof of Theorem 6.1. Let $f_s : \{0,1\}^d \to \{0,1\}$ be an unknown L-hidden-layer ReLU network obtained from the PRF family by picking the key s at random. Consider the lifted $(L+2)$-hidden-layer ReLU network $f_s^{\mathsf{DV}} : \mathbb{R}^d \to \mathbb{R}$ from Eq. (3), given by $f_s^{\mathsf{DV}}(z) = \mathrm{ReLU}(f_s(N_1(z)) - N_2'(z))$, where N_1 and N_2 are from Lemmas 3.6 and 3.7, and $N_2'(z) = \sum_j N_2(z_j)$. Suppose there were an efficient learner A capable of learning functions of the form f_s^{DV} using queries. By the DV lift (Theorem 3.5), A yields an efficient predictor B achieving small constant error w.r.t. the unknown f_s, contradicting Lemma 6.2. We only need to verify that A's query access to f_s^{DV} can be simulated by B. Indeed, suppose A makes a query to f_s^{DV} at a point $z \in \mathbb{R}^d$. Then B can make a query to f_s at the point $\mathrm{sign}(z)$ and return $\mathrm{ReLU}(f_s(\mathrm{sign}(z)) - N_2'(z)) = f_s^{\mathsf{DV}}(z)$, as this was the key property satisfied by f_s^{DV}. This completes the reduction and proves the theorem. □

Acknowledgments. We would like to thank our anonymous reviewers for pointing out an issue in the first version of our proof. Part of this work was completed while the authors were visiting the Simons Institute for the Theory of Computing.

References

[AAK21] Naman Agarwal, Pranjal Awasthi, and Satyen Kale. A deep conditioning treatment of neural networks. In *Algorithmic Learning Theory*, pages 249–305. PMLR, 2021. 1.1

[ADHV19] Alexandr Andoni, Rishabh Dudeja, Daniel Hsu, and Kiran Vodrahalli. Attribute-efficient learning of monomials over highly-correlated variables. In *Algorithmic Learning Theory*, pages 127–161. PMLR, 2019. 1, 1.1

[AK95] Dana Angluin and Michael Kharitonov. When won't membership queries help? *Journal of Computer and System Sciences*, 50(2):336–355, 1995. 1.1

[AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited. In *Annual Cryptology Conference*, pages 57–74. Springer, 2013. 1, 2.3

[APVZ14] Alexandr Andoni, Rina Panigrahy, Gregory Valiant, and Li Zhang. Learning sparse polynomial functions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 500–510. SIAM, 2014. 1, 1.1

[ATV21] Pranjal Awasthi, Alex Tang, and Aravindan Vijayaraghavan. Efficient algorithms for learning depth-2 neural networks with general relu activations. *Advances in Neural Information Processing Systems*, 34, 2021. 1, 1, 1

[BFJ+94] Avrim Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. Weakly learning dnf and characterizing statistical query learning using fourier analysis. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 253–262, 1994. 4.3

[BG17] Alon Brutzkus and Amir Globerson. Globally optimal gradient descent for a convnet with gaussian inputs. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 605–614, 2017. 1, 1

[BGM+16] Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In *Theory of Cryptography Conference*, pages 209–224. Springer, 2016. 1, 2.3, 2.3

[BGML+18] Sauvik Bhattacharya, Oscar Garcia-Morchon, Thijs Laarhoven, Ronald Rietman, Markku-Juhani O Saarinen, Ludo Tolhuizen, and Zhenfei Zhang. Round5: Compact and fast post-quantum public-key encryption. *IACR Cryptol. ePrint Arch.*, 2018:725, 2018. 1, 2.3

[BIP+18] Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J Wu. Exploring crypto dark matter. In *Theory of Cryptography Conference*, pages 699–729. Springer, 2018. 6

[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. PMLR, 2019. 1, 1

[Bog21] Andrej Bogdanov. Personal communication, 2021. 4.2, C

[BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In *Annual Cryptology Conference*, pages 353–370. Springer, 2014. 6

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 719–737. Springer, 2012. 1, 2.3, 2.3, 6

[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. 1.1

[BR17] Andrej Bogdanov and Alon Rosen. Pseudorandom functions: Three decades later. In *Tutorials on the Foundations of Cryptography*, pages 79–158. Springer, 2017. 6, C

[BRST21] Joan Bruna, Oded Regev, Min Jae Song, and Yi Tang. Continuous lwe. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 694–707, 2021. 1

[CGV15] Aloni Cohen, Shafi Goldwasser, and Vinod Vaikuntanathan. Aggregate pseudorandom functions and connections to learning. In *Theory of Cryptography Conference*, pages 61–89. Springer, 2015. 1.1

[CKLS18] Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yongsoo Song. Lizard: Cut off the tail! a practical post-quantum public-key encryption from lwe and lwr. In *International Conference on Security and Cryptography for Networks*, pages 160–177. Springer, 2018. 1, 2.3

[CKM20] Sitan Chen, Adam R Klivans, and Raghu Meka. Learning deep relu networks is fixed-parameter tractable. *arXiv preprint arXiv:2009.13512*, 2020. 1, 1, 1, 1.1, 1.1

[CKM21] Sitan Chen, Adam Klivans, and Raghu Meka. Efficiently learning one hidden layer relu networks from queries. In *Advances in Neural Information Processing Systems*, 2021. 1, 1.1

[DG21] Amit Daniely and Elad Granot. An exact poly-time membership-queries algorithm for extraction a three-layer relu network. *arXiv preprint arXiv:2105.09673*, 2021. 1

[DGK+20] Ilias Diakonikolas, Surbhi Goel, Sushrut Karmalkar, Adam R Klivans, and Mahdi Soltanolkotabi. Approximation schemes for relu regression. In *Conference on Learning Theory*, 2020. 1, 1

[DGKP20] Abhimanyu Das, Sreenivas Gollapudi, Ravi Kumar, and Rina Panigrahy. On the learnability of random deep networks. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 398–410. SIAM, 2020. 1.1

[DK20] Ilias Diakonikolas and Daniel M. Kane. Small covers for near-zero sets of polynomials and learning latent variable models. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 184–195, 2020. 1, 1, 1, 1.1

[DK21] Ilias Diakonikolas and Daniel M. Kane. Non-gaussian component analysis via lattice basis reduction, 2021. 1

[DKKZ20] Ilias Diakonikolas, Daniel M Kane, Vasilis Kontonis, and Nikos Zarifis. Algorithms and sq lower bounds for pac learning one-hidden-layer relu networks. In *Conference on Learning Theory*, pages 1514–1539. PMLR, 2020. (document), 1, **??**, 1.1, 1.1

[DKRV18] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem. In *International Conference on Cryptology in Africa*, pages 282–305. Springer, 2018. 1, 2.3

[DKZ20] Ilias Diakonikolas, Daniel M Kane, and Nikos Zarifis. Near-optimal sq lower bounds for agnostically learning halfspaces and relus under gaussian marginals. *arXiv preprint arXiv:2006.16200*, 2020. (document), 1.1

[DLSS14] Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. From average case complexity to improper learning complexity. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 441–448, 2014. 1

[DSS16] Amit Daniely and Shai Shalev-Shwartz. Complexity theoretic limitations on learning dnf's. In *Conference on Learning Theory*, pages 815–830. PMLR, 2016. 1

[DV20] Amit Daniely and Gal Vardi. Hardness of learning neural networks with natural weights. *Advances in Neural Information Processing Systems*, 33, 2020. 1, 1.1

[DV21] Amit Daniely and Gal Vardi. From local pseudorandom generators to hardness of learning. In *Conference on Learning Theory*, pages 1358–1394. PMLR, 2021. (document), 1, 1, **??**, 1, 1.1, 1.2, 1.2, 3, 3, 3.5, 3.2, 5, A

[Ear19] Mike Earnest. Proving an identity involving the alternating sum of products of binomial coefficients. Mathematics Stack Exchange, 2019. URL: https://math.stackexchange.com/q/3108805 (version: 2019-02-11). B

[Fel09] Vitaly Feldman. On the power of membership queries in agnostic learning. *The Journal of Machine Learning Research*, 10:163–182, 2009. 1.1

[GGJ+20] Surbhi Goel, Aravind Gollakota, Zhihan Jin, Sushrut Karmalkar, and Adam Klivans. Superpolynomial lower bounds for learning one-layer neural networks using gradient descent. In *International Conference on Machine Learning*, pages 3587–3596. PMLR, 2020. (document), 1, **??**, 1.1, 1.1

[GGK20] Surbhi Goel, Aravind Gollakota, and Adam Klivans. Statistical-query lower bounds via functional gradients. *Advances in Neural Information Processing Systems*, 33, 2020. (document), 1.1

[GKK19] Surbhi Goel, Sushrut Karmalkar, and Adam Klivans. Time/accuracy tradeoffs for learning a relu with respect to gaussian marginals. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 8584–8593, 2019. 1.1

[GKKT17] Surbhi Goel, Varun Kanade, Adam Klivans, and Justin Thaler. Reliably learning the relu in polynomial time. In *Conference on Learning Theory*, pages 1004–1042. PMLR, 2017. 1.1

[GKM18] Surbhi Goel, Adam R. Klivans, and Raghu Meka. Learning one convolutional layer with overlapping patches. In *ICML*, volume 80, pages 1778–1786. PMLR, 2018. 1, 1

[GLM18] Rong Ge, Jason D Lee, and Tengyu Ma. Learning one-hidden-layer neural networks with landscape design. In *6th International Conference on Learning Representations, ICLR 2018*, 2018. 1, 1, 1

[HMP+93] András Hajnal, Wolfgang Maass, Pavel Pudlák, Mario Szegedy, and György Turán. Threshold circuits of bounded depth. *Journal of Computer and System Sciences*, 46(2):129–154, 1993. 6

[JCB+20] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In Srdjan Capkun and Franziska Roesner, editors, *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pages 1345–1362. USENIX Association, 2020. 1

[JSA15] Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *arXiv preprint arXiv:1506.08473*, 2015. 1, 1, 1

[JWZ20] Rajesh Jayaram, David P. Woodruff, and Qiuyi Zhang. Span recovery for deep neural networks with applications to input obfuscation. In *ICLR*. OpenReview.net, 2020. 1

[JZ16] Zhengzhong Jin and Yunlei Zhao. Optimal key consensus in presence of noise. *arXiv preprint arXiv:1611.06150*, 2016. 1, 2.3

[Kea98] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, 1998. 2.2, 4.3

[Kha95] Michael Kharitonov. Cryptographic lower bounds for learnability of boolean functions on the uniform distribution. *Journal of Computer and System Sciences*, 50(3):600–610, 1995. 1.1

[KK14] Adam Klivans and Pravesh Kothari. Embedding hard learning problems into gaussian space. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014. (document), 1.1, 3.1

[KL01] Matthias Krause and Stefan Lucks. Pseudorandom functions in in tc0 and cryptographic limitations to proving lower bounds. *computational complexity*, 10(4):297–313, 2001. 6

[KS09] Adam R Klivans and Alexander A Sherstov. Cryptographic hardness for learning intersections of halfspaces. *Journal of Computer and System Sciences*, 75(1):2–12, 2009. 1, 1.1

[LLL82] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261:515–534, 1982. 1

[LMZ20] Yuanzhi Li, Tengyu Ma, and Hongyang R. Zhang. Learning over-parametrized two-layer neural networks beyond ntk. In *Conference on Learning Theory 2020*, volume 125, pages 2613–2682. PMLR, 2020. 1, 1

[LSSS14] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. On the computational efficiency of training neural networks. *Advances in Neural Information Processing Systems*, 27:855–863, 2014. 1.1

[LY17] Yuanzhi Li and Yang Yuan. Convergence analysis of two-layer neural networks with relu activation. In *Advances in Neural Information Processing Systems 30*, pages 597–607, 2017. 1, 1

[MSDH19] Smitha Milli, Ludwig Schmidt, Anca D. Dragan, and Moritz Hardt. Model reconstruction from model explanations. In *FAT*, pages 1–9. ACM, 2019. 1

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 458–467. IEEE, 1997. 6

[Pei16] Chris Peikert. A decade of lattice cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4):283–424, mar 2016. 2.3

[PMG+17] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In Ramesh Karri, Ozgur Sinanoglu, Ahmad-Reza Sadeghi, and Xun Yi, editors, *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*, pages 506–519. ACM, 2017. 1

[PSP17] PSPACEhard. Alternating sum of binomial coefficients identity. Mathematics Stack Exchange, 2017. URL: https://math.stackexchange.com/q/2183223 (version: 2017-03-12). B

[Raz92] Alexander A Razborov. On small depth threshold circuits. In *Scandinavian Workshop on Algorithm Theory*, pages 42–52. Springer, 1992. 6

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009. 2.3

[Reg10] Oded Regev. The learning with errors problem. *Invited survey in CCC*, 7(30):11, 2010. 2.3

[Rey20] Lev Reyzin. Statistical queries and statistical algorithms: Foundations and applications. *arXiv preprint arXiv:2004.00557*, 2020. 2.2

[RK20] David Rolnick and Konrad P. Kording. Reverse-engineering deep relu networks. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 8178–8187. PMLR, 2020. 1

[RR97] Alexander A Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997. 6

[Sha18] Ohad Shamir. Distribution-specific hardness of learning neural networks. *The Journal of Machine Learning Research*, 19(1):1135–1163, 2018. 1.1

[SVWX17] Le Song, Santosh Vempala, John Wilmes, and Bo Xie. On the complexity of learning neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5520–5528, 2017. 1.1

[SZB21] Min Jae Song, Ilias Zadik, and Joan Bruna. On the cryptographic hardness of learning single periodic neurons. *arXiv preprint arXiv:2106.10744*, 2021. 1, 1, 1.1

[Tia17] Yuandong Tian. An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, volume 70, pages 3404–3413. PMLR, 2017. 1, 1

[TJ+16] Florian Tramèr, Fan Zhang 0022, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. *CoRR*, abs/1609.02943, 2016. 1

[Val84] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984. 1, 1.1, 6

[VRPS21] Gal Vardi, Daniel Reichman, Toniann Pitassi, and Ohad Shamir. Size and depth separation in approximating natural functions with neural networks. *arXiv preprint arXiv:2102.00314*, 2021. 6

[VSS+22] Kiran Vodrahalli, Rakesh Shivanna, Mahesh Sathiamoorthy, Sagar Jain, and Ed Chi. Algorithms for efficiently learning low-rank neural networks, 2022. 1

[Vu06] VH Vu. On the infeasibility of training neural networks with small mean-squared error. *IEEE Transactions on Information Theory*, 44(7):2892–2900, 2006. 1.1

[VW19] Santosh Vempala and John Wilmes. Gradient descent for one-hidden-layer neural networks: Polynomial convergence and sq lower bounds. In *COLT*, volume 99, 2019. (document), 1, 1.1

[ZSJ+17] Kai Zhong, Zhao Song, Prateek Jain, Peter L Bartlett, and Inderjit S Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *International conference on machine learning*, pages 4140–4149. PMLR, 2017. 1, 1, 1

[ZSWB22] Ilias Zadik, Min Jae Song, Alexander S. Wein, and Joan Bruna. Lattice-based methods surpass sum-of-squares in clustering, 2022. 1

[ZYWG19] Xiao Zhang, Yaodong Yu, Lingxiao Wang, and Quanquan Gu. Learning one-hidden-layer relu networks via gradient descent. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1524–1534. PMLR, 2019. 1, 1

A Barriers for constructing N_3

We briefly discuss why one natural approach to constructing N_3 satisfying the ideal properties in Eq. (4) ultimately requires two hidden layers rather than one, unlike the construction we give in Sections 3.2 and 3.3.

The most straightforward way to ensure that a function of $s_1, \dots, s_d, t$ vanishes whenever there exists j for which $s_j = 1$ would be to threshold on $\sum s_j$, e.g. by taking $\mathrm{ReLU}(1 - \sum_j s_j)$. While this function is a one-hidden-layer ReLU network, it is unclear how to modify it to satisfy the remaining desiderata in (4) while preserving the fact that it has only one hidden layer. We note that [DV21] takes this approach of thresholding on $\sum_j s_j$ but uses two hidden layers.

Here we informally argue that such an approach inherently requires an extra hidden layer. That is, we argue that no function $N : \mathbb{R}^2 \to \mathbb{R}$ that takes as inputs $s \triangleq \sum_j s_j$ and t and satisfies (4) can be implemented as a one-hidden-layer network. Concretely, $N(s,t)$ must vanish whenever $s \geq 1$ or $t \in \mathbb{Z}\backslash\{0\}$. Any function computed by a one-hidden-layer ReLU network of the form $(s,t) \mapsto \sum_i \mathrm{ReLU}(a_i s + b_i t - c_i)$, unless if it is affine linear, must in general be nowhere smooth (i.e. have a discontinuous gradient) along the entire line where a particular neuron of the network vanishes. In our example, these are the lines $\{(s,t) : a_i s + b_i t = c_i\}$. But this means that such a line cannot intersect the region $\{(s,t) : s \geq 1\}$, as otherwise it would be zero (hence smooth) on an infinite segment of the line. This can only happen if $b_i = 0$, i.e. none of the neurons of N depend on t. Such a network clearly cannot satisfy (4).

B Supporting lemmas for Section 3

Lemma B.1. *For any $0 \leq S < m \leq d$,*

$$\sum_{i=S}^{m} (-1)^{m-i} \binom{d-i-2}{d-m-2} \binom{d-m-1}{i-S} = 0. \tag{20}$$

Proof. We will show that for any integers $j \geq \ell \geq 1$,

$$\sum_{k=0}^{\ell} (-1)^k \binom{j-k}{\ell-1} \binom{\ell}{k} = 0. \tag{21}$$

We would like to substitute $\ell = d-m-1$ and $j = d-2-S$. Note that this is valid as we can assume without loss of generality that $d-m-1 \geq 1$ (otherwise $\binom{d-m-1}{i-S} = 0$ on the right-hand side of (20)), and $j \geq \ell$ by our assumption that $S < m$. We conclude the identity

$$0 = \sum_{k=0}^{d-m-1} (-1)^k \binom{d-2-S-k}{d-m-2} \binom{d-m-1}{k} = \sum_{i=S}^{d-m-1+S} (-1)^{i-S} \binom{d-i-2}{d-m-2} \binom{d-m-1}{i-S}, \tag{22}$$

where the second step is by the change of variable $i = k + S$. If $d-m-1+S \geq m$, then note that all summands $m < i \leq d-m-1+S$ vanish because in that case $d-i-2 < d-m-2$ and so $\binom{d-i-2}{d-m-2} = 0$. If $d-m-1+S < m$, then note that all summands $d-m-1+S < i \leq m$ vanish

because in that case $d-m-1 < i-S$ and so $\binom{d-m-1}{i-S} = 0$. We conclude that (22) is equal, up to a sign, to the left-hand side of (20), so we'd be done.

It remains to establish (21), which we do by following an argument due to [Ear19]. Observe that the left-hand side of (21) is simply counting via inclusion-exclusion the number of subsets of $\{1,\dots,j\}$ of size $\ell-1$ which contain $\{1,\dots,\ell\}$. Indeed, the $k=0$ summand counts all subsets of size $\ell-1$. The $k=1$ summands subtract out the contribution, for every $1 \le x \le \ell$, from the subsets of size $\ell-1$ which contain x. The $k=2$ summands add back the contribution, for every distinct $1 \le x < y \le \ell$, from the subsets of size $\ell-1$ which contain both of x, y, etc. □

Lemma B.2. *For any integers $m \ge 3$ and $a \in \{0,1,2\}$,*

$$\sum_{i=1}^{m}\sum_{j=1}^{m+1-i}(-1)^{m-i}\binom{d-i-j-1}{m-i-j+1}\binom{d-1}{i-1}\cdot j^a = \mathbb{1}[a=0]$$

$$\sum_{i=0}^{m}\sum_{j=1}^{m+1-i}(-1)^{m-i}\binom{d-i-j-1}{m-i-j+1}\binom{d-1}{i}\cdot j^a = 0$$

Proof. By taking $\ell = i+j$, we can rewrite these sums as

$$S_{a,m} \triangleq \sum_{\ell=2}^{m+1}\sum_{i=1}^{\ell-1}(-1)^{m-i}\binom{d-1-\ell}{m+1-\ell}\binom{d-1}{i-1}(\ell-i)^a$$

$$T_{a,m} \triangleq \sum_{\ell=1}^{m+1}\sum_{i=0}^{\ell-1}(-1)^{m-i}\binom{d-1-\ell}{m+1-\ell}\binom{d-1}{i}(\ell-i)^a$$

We proceed by induction on m. The base cases follow from a direct calculation. By the change of variable $\ell' = \ell - 1$, we can rewrite $S_{a,m+1}$ as

$$\begin{aligned}
&-\sum_{\ell'=1}^{m+1}\sum_{i=1}^{\ell'}(-1)^{m-i}\binom{d-1-\ell'}{m+1-\ell'}\binom{d-1}{i-1}(\ell'+1-i)^a \\
&\quad= -\sum_{\ell'=1}^{m+1}\sum_{i=1}^{\ell'}(-1)^{m-i}\binom{d-1-\ell'}{m+1-\ell'}\binom{d-1}{i-1}(\ell'-i)^a \\
&\qquad -\sum_{\ell'=1}^{m+1}\sum_{i=1}^{\ell'}(-1)^{m-i}\binom{d-1-\ell'}{m+1-\ell'}\binom{d-1}{i-1}\sum_{b=0}^{a-1}\binom{a}{b}(\ell'-i)^b \qquad (23)
\end{aligned}$$

Note that the first term on the right-hand side differs from $S_{a,m}$ only in the summands given by $1 \le i = \ell' \le m+1$, and those summands clearly vanish. We conclude that the first term on the right-hand side of (23) is exactly $S_{a,m}$. For the second term on the right-hand side of (23), the part coming from any $0 < b \le a-1$ is also zero, so we thus get

$$\begin{aligned}
&= S_{a,m} - \sum_{\ell'=1}^{m+1}\sum_{i=1}^{\ell'}(-1)^{m-i}\binom{d-1-\ell'}{m+1-\ell'}\binom{d-1}{i-1} \\
&= S_{a,m} - S_{0,m} - \sum_{\ell'=1}^{m+1}(-1)^{m-\ell'}\binom{d-1-\ell'}{m+1-\ell'}\binom{d-1}{\ell'-1} \\
&= S_{a,m} - 1 - \sum_{\ell'=1}^{m+1}(-1)^{m-\ell'}\binom{d-1-\ell'}{m+1-\ell'}\binom{d-1}{\ell'-1} \\
&= S_{a,m} = \mathbb{1}[a=0], \qquad (24)
\end{aligned}$$

where the penultimate step follows e.g. by applying the identity in [PSP17]. This completes the induction for $S_{a,m}$.

For $T_{a,m}$, note that by the change of variable $i' = i+1$,

$$\begin{aligned}
T_{a,m} &= -\sum_{\ell=1}^{m+1}\sum_{i'=1}^{\ell}(-1)^{m-i'}\binom{d-1-\ell}{m+1-\ell}\binom{d-1}{i'-1}(\ell-i'+1)^a \\
&= -\sum_{\ell=2}^{m+1}\sum_{i'=1}^{\ell-1}(-1)^{m-i'}\binom{d-1-\ell}{m+1-\ell}\binom{d-1}{i'-1}(\ell-i'+1)^a - \sum_{\ell=1}^{m+1}(-1)^{m-\ell}\binom{d-1-\ell}{m+1-\ell}\binom{d-1}{\ell-1} \\
&= -\sum_{b=0}^{a}\binom{a}{b}S_{b,m} + 1 = 0,
\end{aligned}$$

where in the second step we pulled out the summands corresponding to $i' = \ell$, in the third step we used (24), and in the last step we used that for $m \ge 3$, $S_{b,m} = \mathbb{1}[b \ne 0]$ for $0 \le b \le 2$. □

C SQ lower bound for the LWR functions

Here we prove an SQ lower bound for the LWR functions (Theorem 4.5) using a general formulation in terms of pairwise independent function families. To our knowledge, this particular formulation has not appeared explicitly before in the literature, and was communicated to us by Bogdanov [Bog21]. A variant of this argument may be found in [BR17, §7.7].

Definition C.1. Let $\mathcal{C}$ be a function family mapping $\mathcal{X}$ to $\mathcal{Y}$, and let D be a distribution on $\mathcal{X}$. We call $\mathcal{C}$ an $(1-\eta)$-pairwise independent function family if with probability $1-\eta$ over the choice of x, x' drawn independently from D, the distribution of $(f(x), f(x'))$ for f drawn uniformly at random from $\mathcal{C}$ is the product distribution $\mathrm{Unif}(\mathcal{Y}) \otimes \mathrm{Unif}(\mathcal{Y})$.

Lemma C.2. *Fix security parameter n and moduli p, q. The $\mathsf{LWR}_{n,p,q}$ function class $\mathcal{C}_{\mathsf{LWR}} = \{f_w \mid w \in \mathbb{Z}_q^n\}$ is $(1 - \frac{2}{q^{n-1}})$-pairwise independent with respect to $\mathrm{Unif}(\mathbb{Z}_q^n)$.*

Proof. This follows from the simple observation that whenever $x, x' \in \mathbb{Z}_q^n$ are linearly independent, the pair $(w \cdot x \bmod q, w \cdot x' \bmod q)$ for $w \sim \mathrm{Unif}\{Z_q^n\}$ is distributed as $\mathrm{Unif}(\mathbb{Z}_q) \otimes \mathrm{Unif}(\mathbb{Z}_q)$. For such x, x', $(f_w(x), f_w(x')) = (\frac{1}{p}\lfloor w \cdot x \bmod q \rceil_p), \frac{1}{p}\lfloor w \cdot x' \bmod q \rceil_p)$ for $f_w \sim \mathrm{Unif}(\mathcal{C}_{\mathsf{LWR}})$ is distributed

as $\mathrm{Unif}(\mathbb{Z}_p/p) \otimes \mathrm{Unif}(\mathbb{Z}_p/p)$. The probability that $x, x' \sim \mathrm{Unif}(\mathbb{Z}_q^n)$ are linearly dependent is at most

$$\mathbb{P}[x = 0] + \mathbb{P}[x \neq 0]\,\mathbb{P}[x' \text{ is a multiple of } x] \leq \frac{1}{q^n} + \frac{q}{q^n} \leq \frac{2}{q^{n-1}}.$$

□

We can now prove full SQ lower bounds for any $(1-\eta)$-pairwise independent function family as follows.

Lemma C.3. *Let $\mathcal{C}$ mapping $\mathcal{X}$ to $\mathcal{Y}$ be a $(1-\eta)$-pairwise independent function family w.r.t. a distribution D on $\mathcal{X}$. Let $\phi : \mathcal{X} \times \mathcal{Y} \to [-1, 1]$ be any bounded query function. Then*

$$\operatorname*{Var}_{f \sim \mathrm{Unif}(\mathcal{C})} \mathop{\mathbb{E}}_{x \sim D} [\phi(x, f(x))] \leq 2\eta.$$

Proof. Denote $\mathbb{E}_{x \sim D}[\phi(x, f(x))]$ by $\phi[f]$. By some algebraic manipulations (with all subscripts denoting independent draws),

$$\begin{aligned}
\operatorname*{Var}_{f \sim \mathrm{Unif}(\mathcal{C})} [\phi[f]] &= \mathop{\mathbb{E}}_{f} \left[\phi[f]^2\right] - \left(\mathop{\mathbb{E}}_{f} [\phi[f]]\right)^2 \\
&= \mathop{\mathbb{E}}_{f} [\phi[f]\phi[f]] - \mathop{\mathbb{E}}_{f} [\phi[f]] \mathop{\mathbb{E}}_{f'} \left[\phi[f']\right] \\
&= \mathop{\mathbb{E}}_{f, f'} \left[\mathop{\mathbb{E}}_{x} [\phi(x, f(x))] \mathop{\mathbb{E}}_{x'} [\phi(x', f(x'))] - \mathop{\mathbb{E}}_{x} [\phi(x, f(x))] \mathop{\mathbb{E}}_{x'} [\phi(x', f'(x'))] \right] \\
&= \mathop{\mathbb{E}}_{x, x'} \mathop{\mathbb{E}}_{f, f'} \left[\phi(x, f(x))\phi(x', f(x')) - \phi(x, f(x))\phi(x', f'(x'))\right].
\end{aligned}$$

By $(1-\eta)$-pairwise independence of $\mathcal{C}$, the inner expectation vanishes with probability $1-\eta$ over the choice of $x, x' \sim D$, and is at most 2 otherwise. This gives the claim. □

Theorem C.4. *Let $\mathcal{C}$ mapping $\mathcal{X}$ to $\mathcal{Y}$ be a $(1-\eta)$-pairwise independent function family w.r.t. a distribution D on $\mathcal{X}$. For any $f \in \mathcal{C}$, let D_f denote the distribution of $(x, f(x))$ where $x \sim D$. Let $D_{\mathrm{Unif}(\mathcal{C})}$ denote the distribution of (x, y) where $x \sim D$ and $y = f(x)$ for $f \sim \mathrm{Unif}(\mathcal{C})$ (this can be thought of as essentially $D \otimes \mathrm{Unif}(\mathcal{Y})$). Any SQ learner able to distinguish the labeled distribution D_{f^*} for an unknown $f^* \in \mathcal{C}$ from the randomly labeled distribution $D_{\mathrm{Unif}(\mathcal{C})}$ using bounded queries of tolerance τ requires at least $\frac{\tau^2}{2\eta}$ such queries.*

Proof. Let $\phi : \mathcal{X} \times \mathcal{Y} \to [-1, 1]$ be any query made by the learner. For any $f \in \mathcal{C}$, let $\phi[f]$ denote $\mathbb{E}_{x \sim D}[\phi(x, f(x))] = \mathbb{E}_{(x,y) \sim D_f}[\phi(x, y)]$. Consider the adversarial strategy where the SQ oracle responds to this query with $\overline{\phi} = \mathbb{E}_{f \sim \mathrm{Unif}(\mathcal{C})} \phi[f] = \mathbb{E}_{(x,y) \sim D_{\mathrm{Unif}(\mathcal{C})}}[\phi(x, y)]$. By Chebyshev's inequality and Lemma C.3,

$$\mathop{\mathbb{P}}_{f \sim \mathcal{C}} \left[\left|\phi[f] - \overline{\phi}\right| > \tau\right] \leq \frac{\mathrm{Var}_{f \sim \mathrm{Unif}(\mathcal{C})} \left[\phi[f]\right]}{\tau^2} \leq \frac{2\eta}{\tau^2}.$$

So each such query only allows the learner to rule out at most a $\frac{2\eta}{\tau^2}$ fraction of $\mathcal{C}$. Thus to distinguish D_{f^*} from $D_{\mathrm{Unif}(\mathcal{C})}$, the learner requires at least $\frac{\tau^2}{2\eta}$ queries. □

Theorem 4.5 now follows easily as a corollary.

Proof of Theorem 4.5. It is not hard to see that learning $\mathcal{C}_{\mathsf{LWR}}$ up to squared loss $1/16$ certainly suffices to solve the distinguishing problem in Theorem C.4. The claim now follows by Lemma C.2. □

Remark C.5. We remark that the argument in this section, specialized to the $q = 2$ case, recovers the traditional SQ lower bound for parities (Theorem 4.3) without appealing to any notion of statistical dimension.