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

Sitan Chen  
UC Berkeley  
sitanc@berkeley.edu  

Aravind Gollakota  
UT Austin  
aravindg@cs.utexas.edu

Adam R. Klivans  
UT Austin  
klivans@cs.utexas.edu

Raghu Meka  
UCLA  
raghum@cs.ucla.edu

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 \(\hat{f}\) for which \(\mathbb{E}_{x \sim \mathcal{D}}[(f(x) - \hat{f}(x))^2]\) 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 \(\mathcal{D}\) 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, ZYW19, 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 \(f(x) = \sum_{i=1}^{d} \lambda_i \sigma(\langle w_i, x \rangle)\).
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. 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.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Num. hidden layers</th>
<th>Model of hardness</th>
</tr>
</thead>
<tbody>
<tr>
<td>[DKKZ20, GGJ+20]</td>
<td>1</td>
<td>Correlational SQ</td>
</tr>
<tr>
<td>[DV21]</td>
<td>3</td>
<td>Cryptographic (assuming existence of local PRGs)</td>
</tr>
<tr>
<td>This work</td>
<td>2</td>
<td>Full SQ</td>
</tr>
<tr>
<td>This work</td>
<td>2</td>
<td>Cryptographic (assuming hardness of LWR)</td>
</tr>
</tbody>
</table>

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 Appendix A.)

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

**Theorem 1.1** (Full SQ lower bound for two hidden layers (informal), see Theorem 3.1). *Any SQ algorithm for learning poly(d)-sized two-hidden-layer ReLU networks over \( \mathcal{N}(0, I_d) \) up to squared loss \( 1/\text{poly}(d) \) 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 extend to any reasonably anticoncentrated product distribution over \( \mathbb{R}^d \), see Remark C.5.

**Cryptographic Lower Bound** While Theorem 1.1 rules out most known approaches for provably learning neural networks (e.g. method of moments/tensor decomposition [JSA15, ZSJ+17, ...]}
GLM18, BJW19, DGK²⁰, DK20, ATV21], noisy gradient descent [BG17, LY17, Tia17, GKM18, ZYW19, 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], no polynomial-time algorithm can learn two-hidden-layer neural networks from Gaussian examples. The LWR problem 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 1.2.** 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 $f_w: \mathbb{Z}_q^n \to \mathbb{Z}_q/p$ by $f_w(x) = \frac{1}{p}\lfloor w \cdot x \rfloor_p = \frac{1}{p}(w \cdot x \mod q)$, where $\lfloor t \rfloor$ is the closest integer to $t$. In the 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 Unif($\mathbb{Z}_q/p$). LWE is similar, except that $y \sim \mathbb{Z}_q/q$ for some $e \in \mathbb{Z}_q$ sampled from a carefully chosen noise distribution, or is drawn from Unif($\mathbb{Z}_q/q$).

**Theorem 1.3** (Cryptographic hardness result (informal), see Theorem 4.1). Suppose there exists a poly$(d)$-time algorithm for learning poly$(d)$-sized two-hidden-layer ReLU networks over $N(0, \text{Id}_d)$ up to squared $1/poly(d)$. Then there exists a quasipolynomial-time algorithm for LWR with polynomial modulus (i.e., in the regime where $n = d$, $p, q = poly(n)$, and $q/p = poly(n)$).

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 4.2).

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 4.3).

We discuss existing hardness evidence for LWR as well as its relation to more standard assumptions like LWE in Appendix A.3. From a negative perspective, Theorem 1.3 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 1.3 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 $\hat{f}$ for which $\mathbb{E}[(f(x) - \hat{f}(x))^2]$ is small. The expectation here is with respect to some specified distribution, which we will take to be $\mathcal{N}(0, \text{Id}_d)$.

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 5) 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 1.4** (Label query hardness (informal), see Theorem 5.1). If either the decisional Diffie–Hellman or the Learning with Errors assumption holds, then the class of poly$(d)$-sized constant-depth ReLU networks from $\mathbb{R}^d$ to $\mathbb{R}$ is not learnable up to small constant squared loss $\varepsilon$ over $\mathcal{N}(0, \text{Id}_d)$ 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 1.4 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].

**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-$\ell$ 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 $D$ 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 question.** All known positive results for one hidden layer that run in time polynomial in all parameters require various assumptions on the underlying network. This leaves open the tantalizing possibility of strengthening our results to apply to worst-case one hidden layer networks.

### 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 $q = \text{poly}(d)$. 
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 \((z, \tilde{y})\) labeled by a (real-valued) ReLU network \(f^{DV}\) that behaves similarly to \(f\) in that \(f^{DV}(z)\) approximates \(f(\text{sign}(z))\), where for us \(\text{sign}(t)\) denotes \(1[t > 0]\) and is applied elementwise. The key idea is to use a continuous approximation \(\tilde{\text{sign}}\) of the sign function, and to pair it with a “soft indicator” function \(\text{bad}: \mathbb{R}^d \rightarrow \mathbb{R}_+\) that is large whenever \(\text{sign}(z) \neq \tilde{\text{sign}}(z)\), 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 \(f^{DV}(z) = \text{ReLU}(f(\text{sign}(z)) - \text{bad}(z))\) can be implemented as an \((L + 2)\)-hidden-layer network satisfying
\[
f^{DV}(z) = \text{ReLU}(f(\text{sign}(z)) - \text{bad}(z)).
\]
This property allows us to generate synthetic Gaussian labeled examples \((z, f^{DV}(z))\) from Boolean labeled examples \((x, f(x))\), and thereby reduce the problem of learning \(f\) to that of learning \(f^{DV}\).

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 \([0, \text{poly}(d)]\) of polynomial size, and for some function \(\sigma: T \rightarrow [0, 1]\). For such compressible functions (see Definition 2.1), 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 \(f^\Delta: \mathbb{R}^d \rightarrow \mathbb{R}\) using \(\text{sign}\) and \(\text{bad}\) so that it approximates \(f(\text{sign}(z))\) except when \(\text{bad}\) indicates that \(\text{sign} \neq \tilde{\text{sign}}\). To this end, we might hope to implement, say, 
\[
f^\Delta(z) = \sum_{t \in T} \sigma(t^*) \mathbb{1}[h(\tilde{\text{sign}}(z)) = t^*] \mathbb{1}[\forall j: \text{bad}(z_j) \ll 1].
\]
Here we now view \(\text{bad}\) as a univariate function, and whenever it is small, we can be sure \(\tilde{\text{sign}} = \text{sign}\). 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 \(f^\Delta\) as an \((L + 1)\)-hidden-layer network:
\[
f^\Delta(z) = \sum_{t \in T} \sigma(t^*) N(\text{bad}(z_1), \ldots, \text{bad}(z_d); h(\tilde{\text{sign}}(z)) = t^*).
\]
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:
\[
\psi(s_1, s_2, s_3) - \psi(1, s_2, s_3) - \psi(s_1, 1, s_3) - \psi(s_1, s_2, 1) \\
+ \psi(s_1, 1, 1) + \psi(1, s_2, 1) + \psi(s_1, 1, 1) - \psi(1, 1, 1).
\]
For a suitable choice of \(\psi\), one might hope to build \(N\) out of such a formula by taking \(s_j = \text{bad}(z_j)\) 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 2.6) 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 1.3, 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.
2 Compressing the Daniely–Vardi Lift

In this section we show how to refine the lifting procedure of Daniely and Vardi [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 2.1. Let \( q > 0 \) be a modulus. 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

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

Remark 2.2. 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 \(|T| \leq \text{poly}(d)\), \( \sigma \) is expressible as (the restriction to \( T \)) of a piecewise linear function on \( \mathbb{R} \) whose size and maximum slope are \( \text{poly}(d) \), and hence as a \( \text{poly}(d) \)-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. By a standard padding argument, we also obtain Corollary C.6, which lets us work with polynomial-sized neural networks.

Theorem 2.3 (Compressed DV lift). Let \( q = \text{poly}(d) \) be a modulus. Let \( C \) be a class of compressible \( L \)-hidden-layer \( \text{poly}(d) \)-sized ReLU networks mapping \( \mathbb{Z}_q^d \) to \([0, 1] \). Let \( m = m(d) = \omega(1) \) be a size parameter that grows slowly with \( d \). There exists a class \( C^\oplus \) 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 \( C^\oplus \) over \( \mathcal{N}(0, \text{Id}_d) \) up to squared loss \( d^{-\Theta(m)} \). Then there is an efficient algorithm \( B \) capable of weakly predicting \( C \) over \( \text{Unif}(\mathbb{Z}_q^d) \) with advantage \( d^{-\Theta(m)} \) over guessing the constant 1/2 in the following sense: given access to labeled examples \((x, f(x))\) for \( x \sim \text{Unif}(\mathbb{Z}_q^d) \) and an unknown \( f \in C \), \( B \) satisfies \( E \left[ (B(x) - f(x))^2 \right] < E \left[ \frac{1}{d} - f(x)^2 \right] - d^{-\Theta(m)} \), where the probability is taken over both \( x \) and the internal randomness of \( B \). We refer to \( C^\oplus \) as the lifted class corresponding to \( C \).

The proof of Theorem 2.3 leverages certain one-hidden-layer gadgets. The first two gadgets are inherent to the original DV lift (extended to work with general \( \mathbb{Z}_q \) as opposed to just \([0, 1]\)), while the third is one of our main technical contributions and essential to obtaining an improvement in depth. Proofs are deferred to Appendix C.

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, which we denote by \(|I_j|\), and the shortest one will be the one closest to the origin. Still, by Gaussian anti-concentration, we know that each \(|I_j| \geq \Theta(1/q)\). Let \( \text{thres} : \mathbb{R} \to \mathbb{Z}_q \) be the piecewise constant function that takes on value \( k \) on \( I_k \). Clearly, when \( t \sim \mathcal{N}(0, 1) \), \( \text{thres}(t) \sim \text{Unif}(\mathbb{Z}_q) \). 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 \( \delta/q \) 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 \( S_k \) contains \( 2\delta/q \). By Gaussian anti-concentration again, each \(|S_k| \geq \Theta(\delta/q)\). Notice that by construction, \( P_{x \sim \mathcal{N}(0,1)} [z \in \bigcup_k R_k] = \delta \) and \( P_{z \sim \mathcal{N}(0,1)} [z \in \bigcup_k S_k] = 2\delta \).

Lemma 2.4. 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 \( t \notin \bigcup_k R_k \).

\(^1\)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.
Lemma 2.5. 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

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

Note that when $q = 2$, $N_1$ and $N_2$ play the role of “sign” and “bad” from the technical overview.

To motivate the third gadget, recall from the technical overview that one might hope to build $N_3(s_1, \ldots, s_d; t)$ that behaves like $1[t = 0]1[\forall j : s_j \ll 1]$. Slightly more generally, one can show that it would suffice to build a one-hidden-layer network $N_3$ with the following properties:

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

Unfortunately, most natural attempts to construct $N_3$ with such ideal properties run into difficulties and appear to require two hidden layers (see Appendix D for discussion).

The key idea that lets us make progress 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 2.3, then by setting $\delta$ in Lemmas 2.4 and 2.5 appropriately, we can ensure that with overwhelming probability over $z \sim \mathcal{N}(0, \text{Id})$, no more than $m$ of the $N_2(z_j)$ are simultaneously 1. Accordingly, we focus on constructing $N_3$ such that

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

Our construction for $N_3$ has size $d^{O(m)}$, and satisfies the first and second properties exactly. It also “approximately” satisfies the third in the sense that it takes on a nonzero value with nonnegligible probability over its inputs. As we will see, this turns out to be enough for the reduction to go through. And even though the size of $N_3$ is slightly superpolynomial in the dimension, because the SQ lower bounds for Boolean functions that we build on are exponential, by a simple padding argument we will still obtain a superpolynomial SQ lower bound for our lifted functions.

Lemma 2.6 (Main lemma). Let $m = m(d) = \omega_d(1)$ be a size parameter. There exists a one-hidden-layer neural network $N_3 : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$ such that

(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) $N_3(0, \ldots, 0, s; 0) = s$ for any $s \in [0, \frac{1}{2}]$ (there are $d - 1$ zeroes in front of $s$).

Proof sketch of Theorem 2.3. For each $f \in C$ given by $f = \sigma \circ h$, let $f^\delta \in C^\delta$ be given by

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

where $N_1$ and $N_2$ are from Lemmas 2.4 and 2.5, with the $\delta$ parameter set to $d^{-10m}$, and $N_3$ is from Lemma 2.6. 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 2.6(c), the size of this network is $S = d^{O(m)}$.

Note that for $z$ such that $N_2(z_1), \ldots, N_2(z_d) < 1$, we have $N_1(z) = \text{thres}_q(z)$, and the only $t^*$ for which one of the summands in Eq. (4) is potentially nonzero is the one given by $t^* = h(\text{thres}_q(z))$. So in this case $f^\delta$ simplifies to

$$f^\delta(z) = f(\text{thres}_q(z)) N_3(N_2(z_1), \ldots, N_2(z_d); 0).$$

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 $f^\delta(z) = 0$. 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 $f^b$, 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 $P_{z_j}[N_2(z_j) = 1] = P_{z_j}[z_j \in \cup_k R_k] = \delta = d^{-10m}$. 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

$$\sum_{i=m}^{d} \binom{d-1}{i} (d^{-10m})^i (1 - d^{-10m})^{d-i} \leq (d - m + 1) \binom{d}{m} d^{-20m^2} \leq \frac{d}{2} d^{-10m} \leq d^{-9m^2}$$

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

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

The modified label is given by

$$\tilde{y} = \tilde{y}(y, z) = \begin{cases} 0 & \text{if more than } m \text{ of the } N_2(z_j) \text{ are 1} \\ 0 & \text{if between } 1 \text{ and } m \text{ of the } N_2(z_j) \text{ are 1} \\ y & \text{otherwise} \end{cases}$$

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

So $B$ can feed these new labeled examples $(z, \tilde{y})$ to $A$. Suppose $A$ outputs a hypothesis $\hat{f} : \mathbb{R}^d \to \mathbb{R}$ such that $\mathbb{E}_{z \sim N_d}[(\hat{f}(z) - f^b(z))^2] \leq \varepsilon$. 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 $f^b$ 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 $N_2(z_d) \in \left(\frac{1}{2d}, \frac{1}{d}\right)$. Observe that when $z \in G$, by Eq. (5) and Lemma 2.6(d) we have

$$f^b(z) = f(\text{thres}_q(z)) N_2(N_2(z_1), \ldots, N_2(z_{d-1}), N_2(z_d); 0) = f(x) N_2(0, \ldots, 0, N_2(z_d); 0) = y N_2(z_d),$$

where we use the fact that $\text{thres}_q(z) = x$, so that $f(\text{thres}_q(z)) = f(x) = y$.

One can show that $G$ has non-negligible probability mass. The discrete learner $B$ can now adapt $\hat{f}$ as follows. Given a fresh test point $x \sim \text{Unif}(Z^d)$, the learner forms $z = z(x)$ such that for each coordinate $j \in [d]$, $z_j$ is drawn from $\mathcal{N}(0, 1)$ conditioned on $z_j \in I_x$. If $z \in G$, then $B$ predicts $\tilde{y} = \tilde{y}(N_2(z_d))$ (recall that when $z \in z$, $N_2(z_d) > \frac{1}{2d}$), and otherwise it simply predicts $\tilde{y} = \frac{1}{2}$. By exploiting the fact that this is a good prediction at least on the region $G$, it is not hard to show that $B$’s overall square loss is non-negligibly better than random.

### 3 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. We obtain this by lifting the problem of learning parities under $U_d$, which is well-known to require exponentially many queries.

**Theorem 3.1.** Fix any $\alpha \in (0, 1)$. Any SQ learner capable of learning poly($d$)-sized two-hidden-layer ReLU networks under $\mathcal{N}(0, 1d_d)$ up to squared loss $\varepsilon$ (for some sufficiently small $\varepsilon = 1/\text{poly}(d)$) using bounded queries of tolerance $\tau \geq 2^{-\log d^{2-\alpha}}$ must use at least $\Omega(2^{d^{\log d^{2-\alpha}}})$ such queries.
This theorem is proven using the following key reduction, which adapts the compressed DV lift (Theorem 2.3) to the SQ setting. The proof is deferred to Appendix E.

**Theorem 3.2.** Let \( q = \text{poly}(d) \) be a modulus, and let \( m = m(d) = \omega_d(1) \) be a size parameter. Let \( C \) be a class of compressible \( L \)-hidden-layer poly\((d)\)-sized ReLU networks mapping \( \mathbb{Z}^d_q \) to \([0,1]\), and let \( \triangle^C \) be the lifted class of \((L+1)\)-hidden-layer \( d^{\Theta(m)} \)-sized ReLU networks corresponding to \( C \), mapping \( \mathbb{R}^d \) to \( \mathbb{R} \) (as in Theorem 2.3). Suppose there is an SQ learner \( A \) capable of learning \( \triangle^C \) over \( \mathcal{N}(0,1d) \) up to squared loss \( d^{-\Theta(m^2)} \) using queries of tolerance \( \tau \), 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 \( \tilde{B} \) for \( C \) over \( \text{Unif}(\mathbb{Z}^d_q) \) with advantage \( d^{-\Theta(m)} \) over guessing the constant \( 1/2 \) (in expectation over both the data and the internal randomness of \( \tilde{B} \)).

**Proof of Theorem 3.1.** 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 \( d = 2^{\log^{1/(1+\alpha)} d} \). It is easy to see that the class \( C \) of parities on \([0,1]^d\) can be implemented by compressible one-hidden-layer poly\((d)\)-sized ReLU networks. Indeed, for any \( S \subseteq [d] \) recall that \( \chi_S(x) = 1(\sum_{j \in S} x_j \text{ is odd}) \), which is a compressible one-hidden-layer network with the inner depth-0 network being \( x \mapsto \sum_{j \in S} x_j \) and \( \sigma(t) = 1[t \text{ is odd}] \). Thus the lifted class \( \triangle^C \) 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 3.2 (with \( q = 2 \)), we obtain an SQ algorithm capable of distinguishing parities from random labels using queries of tolerance \( \tau/2 \), assuming \( \tau \geq d^{-\Theta(m^2)} = 2^{-\log^{2c+1} d} = 2^{-\log\frac{2c+1}{2} d'} \). It is well-known [Kea98, BFI+94] that the lower bound for learning parities is \( \Omega(2^{d/\tau^2}) \), which becomes \( \Omega(2^{\log^3 d' \cdot \tau^2}) \). Substituting \( \alpha = \frac{1}{1+c} \) gives the result. \( \square \)

By way of an alternative construction that arguably remains hard even for non-SQ algorithms, in Appendix F we provide a different proof of this SQ lower bound using the LWRF functions in place of parities. We stress that this alternative proof remains unconditional and relies only on the LWRF function family, not on the LWRF hardness assumption itself.

### 4 Cryptographic Hardness Based on LWRF

In this section we show hardness of learning two-hidden-layer ReLU networks over Gaussian inputs based on LWRF. This is a direct application of the compressed DV lift (Theorem 2.3) to the LWRF problem, which by definition a hard learning problem over \( \text{Unif}(\mathbb{Z}^d_q) \).

**Theorem 4.1.** Let \( n \) be the security parameter, and fix moduli \( p, q \geq 1 \) such that \( p, q = \text{poly}(n) \) and \( p/q = \text{poly}(n) \). Let \( d = n \). Let \( c > 0 \), \( m = m(d) = \log^c d \) and \( d' = d^m \). Suppose there exists a \( \text{poly}(d') \)-time algorithm capable of learning \( \text{poly}(d') \)-sized depth-2 ReLU networks under \( \mathcal{N}(0,1d) \) up to squared loss \( 1/\text{poly}(d') \). Then there exists a \( \text{poly}(d') \) time algorithm for \( \text{LWRF}_{n,p,q} \).

**Proof.** We claim that the class \( C_{\text{LWRF}} \) is implementable by compressible \( \text{poly}(d) \)-sized one-hidden-layer ReLU networks over \( \mathbb{Z}^d_q \), or, after padding, over \( \mathbb{Z}^d_q \). Indeed, by definition we have \( f_w(x) = \frac{1}{p} [(w \cdot x) \mod q]_p \), 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 \( \sigma(t) = \frac{1}{p} [t \mod q]_p \). Let \( \triangle^{C_{\text{LWRF}}} \) denote the corresponding lifted class of \( \text{poly}(d') \)-sized two-hidden-layer ReLU networks, padded to have domain \( \mathbb{R}^d \). Applying Corollary C.6 to the assumed learner for \( \triangle^{C_{\text{LWRF}}} \), we obtain a \( \text{poly}(d') \)-time weak predictor predictor for \( C_{\text{LWRF}} \), which readily yields a corresponding distinguisher for the LWRF\(n,p,q\) problem. Using the facts that \( d' = d^m = 2^{\log^{1+c} d} \) and \( d = n \), we may translate \( \text{poly}(d') \) into \( 2^{\Theta((\log^{1+c}) n)} \), yielding the result. \( \square \)

**Remark 4.2.** Note that the choice of \( m = m(d) = \log^c d \) in Theorem 4.1 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 LWRF algorithm that is as mildly superpolynomial as we like.
In addition, we also obtain a hardness result for one-hidden-layer networks under $\text{Unif}\{0, 1\}^d$, improving on the hardness result of [DV21] (see Theorem 3.4 therein) for two-hidden-layer networks under $\text{Unif}\{0, 1\}^d$. For this application, we let $d = n \log q = \tilde{O}(n)$, so that we may identify the domain $\mathbb{Z}_n^d$ with $\{0, 1\}^d$ via the binary representation. This also identifies $\text{Unif}(\mathbb{Z}_n^d)$ with $\text{Unif}\{0, 1\}^d$.

**Corollary 4.3.** Let $n, p, q$ be such that $p, q = \text{poly}(n)$ and $p/q = \text{poly}(n)$, and let $d = n \log q = \tilde{O}(n)$. Suppose there exists an efficient algorithm for learning poly($d$)-sized one-hidden-layer ReLU networks under $U_d$ up to squared loss $1/4$. Then there exists an efficient algorithm for $\text{LWR}_{n,p,q}$.

### 5 Hardness of Learning using Label Queries

Here we show hardness of learning constant-depth ReLU networks over Gaussians from label queries by lifting pseudorandom function (PRF) families. For preliminaries on PRFs and their connection to hardness of learning, see Appendix A.4. Since PRFs are not necessarily compressible, we will simply use the original DV lift (Theorem B.1).

**Theorem 5.1.** Assume there exists a family of PRF’s mapping $\{0, 1\}^d$ to $\{0, 1\}$ implemented by poly($d$)-sized $L$-hidden-layer ReLU networks. Then there does not exist an efficient learner that, given query access to an unknown poly($d$)-sized $(L + 2)$-hidden-layer ReLU network $f : \mathbb{R}^d \rightarrow \mathbb{R}$, is able to output a hypothesis $h : \mathbb{R}^d \rightarrow \mathbb{R}$ such that $\mathbb{E}_{z \sim \mathcal{N}(0,d_{id})}[(h(z) - f(z))^2] \leq 1/16$.

**Proof.** Let $f_s : \{0, 1\}^d \rightarrow \{0, 1\}$ 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 $f_s^{DV} : \mathbb{R}^d \rightarrow \mathbb{R}$ from Eq. (10), given by $f_s^{DV}(z) = \text{ReLU}(f_s(N_1(z)) - N_2(z))$, where $N_1$ and $N_2$ are from Lemmas 2.4 and 2.5, 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 B.1), $A$ yields an efficient predictor $B$ achieving small constant error w.r.t. the unknown $f_s$, contradicting Lemma A.3. 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 $\text{sign}(z)$ and return $\text{ReLU}(f_s(\text{sign}(z)) - N_2'(z)) = f_s^{DV}(z)$, as this was the key property satisfied by $f_s^{DV}$. This completes the reduction and proves the theorem.

**Acknowledgments and Disclosure of Funding**

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. SC is supported in part by NSF Award 2103300. AG and ARK are supported by NSF awards AF-1909204, AF-1717896, and the NSF AI Institute for Foundations of Machine Learning (IFML). RM is supported by NSF CAREER Award CCF-1553605.

**References**


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


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See open question in Section 1.1.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] All proofs are present in either the main body or the forthcoming supplement.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]
A Technical preliminaries

A.1 Notation

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

For $q > 0$, $\mathbb{Z}_q$ will denote the integers modulo $q$, which we will identify with $\{0,\ldots,q-1\}$. We use $\mathbb{Z}_q/q$ to denote $\{0,1/q,\ldots,(q-1)/q\}$. 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, $\text{sign} : \mathbb{R} \to \{0,1\}$ is defined as $\text{sign}(t) = I[t > 0]$. We will extend this to $\mathbb{Z}_q$ by defining $\text{thres}_q : \mathbb{R} \to \mathbb{Z}_q$ in terms of a certain partition of $\mathbb{R}$ into $q$ intervals $I_0,\ldots,I_{q-1}$ (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 $\text{negl}(n)$, 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

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

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

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

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.

A.2 Learning models

Let $C$ be a function class mapping $\mathbb{R}^d$ to $\mathbb{R}$, and let $D$ 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 $f \in C$ and must output a (possibly randomized) predictor that achieves (say) squared loss $\varepsilon$ 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 $x \sim D$, and the learner is considered efficient if it succeeds using $\text{poly}(d,1/\varepsilon)$ time and sample complexity. In the Statistical Query (SQ) model [Kearns98, 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 $|v - E_{x \sim D}[\phi(x,f(x))]| \leq \tau$. A correlational query is one that is linear in $y$, i.e. of the form $\phi(x,y) = \hat{\phi}(x)y$ for some $\phi$, 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 $\text{poly}(d,1/\varepsilon)$ queries and tolerance $\tau \geq 1/\text{poly}(d,1/\varepsilon)$. 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 $\text{poly}(d,1/\varepsilon)$ time and queries.

A.3 Learning with Rounding

Here we provide some further details about $\text{LWR}$ (Definition 1.2) and known hardness results.

**Remark A.1.** Traditionally the $\text{LWR}$ 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 $\text{LWE}$ labels to lie in $\mathbb{Z}_q/q$.

It is known that $\text{LWR}_{n,q,B}$ is as hard as worst-case lattice problems when $q = \text{poly}(n)$ and $B = \lceil q/\text{poly}(n) \rceil$ (see e.g. [Reg10, Pei16] for surveys). Yet this is not known to directly imply the hardness of $\text{LWR}_{n,p,q}$ in the regime in which $p,q$ are both $\text{poly}(n)$, 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 A.2** ([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 \( \text{LWR}_{n,p,q} \) using \( m \) samples implies an efficient algorithm for \( \text{LWE}_{n,q,B} \).

For our purposes, Theorem A.2 is not enough to let us base our Theorem 1.3 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, BGLM+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.

### A.4 Pseudorandom functions and hardness of learning

We 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 A.3.** Let \( C = \{ f_s \} \) 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 \( f_s \in C \), satisfies

\[
\mathbb{P}_{x,s} [L(x) = f_s(x)] \geq \frac{1}{2} + \frac{1}{\text{poly}(d)},
\]

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

There exist multiple candidate constructions of PRF families in the class \( \text{TC}^0 \) 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 \( V_1 \) from Lemma 2.4, any \( \text{TC}^0 \) (meaning depth-\( L \)) function \( f : \{0,1\}^d \to \{0,1\} \) may be implemented as a \( \text{poly}(d) \)-sized \( L \)-hidden-layer ReLU network (see e.g. [VRPS21, Lemma A.3]). Thus we may leverage the following candidate PRF constructions in \( \text{TC}^0 \) for our hardness result:

* PRFs in \( \text{TC}^0 \) based on the decisional Diffie-Hellman (DDH) assumption [KL01] (improving on [NR97]), yielding hardness for depth-6 ReLU networks
* PRFs in \( \text{TC}^0 \) 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 \( \text{TC}^0 \) is a longstanding open question in circuit complexity [Raz92, HMP+93, RR97, KL01]. Under less widely-believed assumptions, [BGM+18] have also proposed candidate PRFs in \( \text{ACC}^0 \).

### A.5 Partial assignments

Let \( \alpha \in \{0,1\}^d \) be a partial assignment. We refer to \( S(\alpha) : \{ i \in [d] : \alpha_i = \ast \} \subseteq [d] \) as the set of free variables and \( [d] \setminus S(\alpha) \) as the set of fixed variables. Given two partial assignments \( \alpha, \beta \), let the

\[
\alpha \oplus \beta = \alpha \text{ or } \beta,
\]

\[
\alpha \odot \beta = \min(\alpha, \beta).
\]

Note that what the authors term a depth-(\( L + 1 \)) network is in fact an \( L \)-hidden-layer network in our terminology.
resolution $\alpha \searrow \beta$ denote the partial assignment $\gamma$ obtained by substituting $\alpha$ into $\beta$. That is,

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

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

Given $\alpha$, 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 $\gamma$, we use $h_\gamma : \mathbb{R}^d \to \mathbb{R}$ to denote its partial restriction given by substituting in $\gamma_i$ into the $i$-th input coordinate if $\gamma_i \in \{0,1\}$. Note that given two partial restrictions $\alpha, \beta$,

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

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

B  The original DV lift

In order to set the stage for our compressed DV lift, we briefly 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 $f^{DV} : \mathbb{R}^d \to \mathbb{R}$ 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 $f_{\mathcal{N}}$ between two desiderata:

- For all but a small fraction of inputs, an initial layer successfully “Booleanizes” the input. In this case, one would like $f^{DV}(z)$ to simply behave as $f(\text{sign}(z))$.
- 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 $N_1 \neq \text{sign}$. 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

$$
f^{DV}(z) = \text{ReLU}(f(N_1(z)) - N_2(z)) \quad (10)
$$

One can show that $f^{DV}$ satisfies $f^{DV}(z) = \text{ReLU}(f(\text{sign}(z)) - N_2(z))$, since $N_2$ “zeroes out” $f^{DV}$ wherever $N_1 \neq \text{sign}$ 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 $(z,y) = ((2x - 1)g, \text{ReLU}(y - N_2((2x - 1)g)))$. 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 B.1 (Original DV lift, implicit in [DV21]). Let $\mathcal{C}$ be a class of $L$-hidden-layer $\text{poly}(d)$-sized ReLU networks mapping $\{0,1\}^d$ to $[0,1]$. There exists a class $C^{DV}$ of $(L + 2)$-hidden-layer
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 $C_{DV}$ over $N(0, \text{Id}_d)$ up to squared loss $\frac{1}{16}$. Then there is an efficient algorithm $B$ capable of weakly predicting $C$ over $\text{Unif} \{0, 1\}^d$ with squared loss $\frac{1}{16}$.

C Full proofs for Section 2

C.1 Gadget constructions

We begin with the constructions of $N_1$ and $N_2$.

**Proof of Lemma 2.4.** This can be done by considering the piecewise linear function that approximates the function $\text{thres}_q$ by matching it exactly on $R \cup kR_k$, and interpolating linearly between values $k-1$ and $k$ on the interval $R_k$ for each $k \in [q-1]$.

**Proof of Lemma 2.5.** Consider the piecewise linear function that is 0 on $R \cup kS_k$, is 1 on $\cup kR_k$, 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 $R_k$.

We now detail the construction of the gadget $N_3$. First let us briefly look at one approach that almost satisfies Eq. (2), except 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:

$$
\begin{align*}
\psi(s_1, s_2, s_3; t) &= \psi(1, s_2, s_3; t) - \psi(s_1, 1, s_3; t) - \psi(s_1, s_2, 1; t) \\
&+ \psi(s_1, 1, 1; t) + \psi(1, s_2, 1; t) + \psi(s_1, 1, 1; t) - \psi(1, 1, 1; t)
\end{align*}
$$

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 C.1. \( \psi : \mathbb{R}^d \rightarrow \mathbb{R} \) be any function. \( \mathcal{P}_i \) denote the set of partial assignments \( \gamma \in \{1, \star \}^d \) with \( i \) Is. The expression
\[
\sum_{i=0}^{d} \sum_{\gamma \in \mathcal{P}_i} (-1)^i \psi_{\gamma}
\] (12)
vanishes whenever any \( s_j = 1 \). (We may view \( t \) as an additional parameter that is always left free, as in Eq. (11))

Proof. For concreteness, suppose \( s_1 = 1 \). Let \( \mathcal{P}_i \) (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 \( \mathcal{P}_i \) and \( \mathcal{P}_i^{1} \) using the map \( \gamma \mapsto \gamma' \) where \( \gamma' = (1, \gamma_2, \ldots, \gamma_d) \). When \( s_1 = 1 \), for every such pair \( (\gamma, \gamma') \), we have \( \psi_{\gamma} = \psi_{\gamma'} \), and moreover they occur in (12) with opposite signs. Thus the entire expression vanishes.

Let us assume for now that \( \psi \) 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 2.3, this would indeed be the case). This construction has size \( 2^d \), meaning that the resulting lifted functions would have size \( S = \text{poly}(2^d) \). But by Theorem 2.5, the SQ lower bound for the LWR functions over \( \mathbb{Z}_2^d \) with \( n = d \) and \( q = \text{poly}(n) \) 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 \( d \) by changing our goal to be closer to Eq. (3) (although still only approximately), while retaining the spirit of constructing \( N_3 \) using a linear combination of partial restrictions. The key lemma in proving Lemma 2.6 is the following.

Lemma C.2. 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 \( \tilde{\beta} \in \mathcal{B} \) for which \( |S(\tilde{\beta})| = i \) and \( w(\tilde{\beta}) = j \). For any symmetric function \( \psi : \mathbb{R}^d \rightarrow \mathbb{R} \), define the function
\[
\psi^* \triangleq \psi - \sum_{i=0}^{m} \sum_{j=1}^{m+1-i} (-1)^{m-i} \cdot \lambda_{i+j} \sum_{\tilde{\beta} \in \mathcal{B}_{i,j}} \psi_{\tilde{\beta}}, \quad \text{for } \lambda_k \triangleq \binom{d-k-1}{m-k+1}
\]
Then
\[
(a) \ |\mathcal{B}| \leq \binom{d}{m}(d-m) \cdot 3^m
\]
\[
(b) \ \psi^* \text{ is symmetric}
\]
\[
(c) \ \psi^*_a : \mathbb{R}^d \rightarrow \mathbb{R} \text{ is the identically zero function for all } \alpha \in \mathcal{A}.
\]

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

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 \( j \) Is for any \( 0 \leq i \leq m \) and \( j \geq 1 \) satisfying \( i + j \leq m + 1 \).

To prove the second part of the lemma, it suffices to show that
\[
\sum_{\tilde{\beta} \in \mathcal{B}_{i,j}} h_{\tilde{\beta}}
\] (13)
is symmetric for all \( i, j \). As transpositions generate the symmetric group on \( d \) elements, it suffices to show that (13) is invariant under swapping two input coordinates, call them \( a, b \in [d] \). For all \( \tilde{\beta} \in \mathcal{B}_{i,j} \) for which \( a, b \) are either both present or both absent in \( S(\tilde{\beta}) \), this clearly does not affect the value of \( h_{\tilde{\beta}} \). Now consider the set \( S_a \) (resp. \( S_b \)) of partial assignments \( \tilde{\beta} \in \mathcal{B}_{i,j} \) for which \( a \) (resp. only \( b \)) is present in \( S(\tilde{\beta}) \). There is a clear bijection \( f : S_a \rightarrow S_b \); given \( \tilde{\beta} \in S_a \), swap the
α- and β-th entries, and vice-versa, and for any \( \overline{\beta} \in S_\alpha \), the function \( h_\pi + h_f(\pi) \) 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 \( \alpha \in A \), as \( h^* \) is symmetric. So consider \( \pi = \{1, 0, \cdots, 0, \ast, \cdots, \ast\} \). We apply (9) to get

\[
\begin{align*}
    h_\pi^* &= h_\pi^\ast - \sum_{i=0}^{m} \sum_{j=1}^{m+1-i} (-1)^{m-i} \cdot \lambda_{i+j} \sum_{\overline{\beta} \in B_{i,j}} h_\pi \setminus \overline{\beta} \\
    &= h_\pi^\ast - \sum_{\pi \in B \cap \text{App}(\alpha) \text{ sorted}} h_\pi^\ast \sum_{i=0}^{m} \sum_{j=1}^{m+1-i} (-1)^{m-i} \cdot \lambda_{i+j} \sum_{\overline{\beta} \in B_{i,j}} 1_{[\overline{\pi} \setminus \overline{\beta} = \pi]} 
\end{align*}
\]

(14)

Note that for \( \overline{\pi} = \overline{\alpha} \), the only \( \overline{\beta} \in B \) for which \( \overline{\pi} \setminus \overline{\beta} = \pi \) is \( \overline{\beta} = \overline{\alpha} \). Indeed, for \( \overline{\beta} \) to be such that \( \overline{\pi} \setminus \overline{\beta} = \overline{\alpha} \), it must have \( S(\overline{\beta}) = S(\overline{\alpha}) \) and exactly one 1, from which it follows that \( \overline{\beta} = \overline{\alpha} \). Since \( \overline{\alpha} \in B_{m,1} \), its coefficient in (14) is given by

\[(1 - 1)^{m-r}, \lambda_{m+1} = 1, \]

and so the \( h_\alpha \) in (14) cancels with the \( \overline{\pi} = \overline{\alpha} \)-th summand in (14).

In the rest of the proof, we can thus focus on sorted \( \pi \in B \cap \text{App}(\alpha) \setminus \{\overline{\pi}\} \). Note that such \( \pi \) satisfy

\[|S(\pi)| < m.\]

(15)

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

Observe that we must have \( \pi_1 = 1 \). Indeed, it cannot be 0 because \( \pi \) is sorted and has at least one 1. It also cannot be \( \ast \). To see this, consider any \( \overline{\beta} \) for which \( \overline{\pi} \setminus \overline{\beta} = \pi \). If we had \( \overline{\beta}_1 \neq \ast \), then clearly \( \pi_1 \neq \ast \). If we had \( \overline{\beta}_1 = \ast \), then \( (\overline{\pi} \setminus \overline{\beta})_1 = 1 \), so \( \pi = \overline{\pi} \setminus \overline{\beta} \) must also have first entry given by 1.

We are now ready to calculate the coefficient of \( h_\pi^* \) (for each \( \pi \in B \cap \text{App}(\alpha) \setminus \{\overline{\pi}\} \)) in (14) by adding the coefficients of all the \( \overline{\beta} \in B \) for which

\[\overline{\pi} \setminus \overline{\beta} = \pi.\]

(16)

First let us consider the contribution of \( \overline{\beta} \in B \) for which \( \overline{\beta}_1 = 1 \). Observe that such \( \overline{\beta} \) must have exactly \( w(\pi) \) 1s. Furthermore, such a \( \overline{\beta} \) is an element of \( B \) if and only if it has at most \( m + 1 - w(\pi) \) free variables, and the set of free variables in \( \overline{\beta} \) must be \( S(\pi) \cup V \) where \( V \) is any subset of \( [d] \setminus \{1\} \cup S(\pi) \). The contribution of all such \( \overline{\beta} \) to the coefficient of \( h_\pi^* \) in (14) is thus

\[
\sum_{i=|S(\pi)|}^{m+1-w(\pi)} (-1)^{m-i} \cdot \lambda_{i+w(\pi)} \cdot \binom{d-m-1}{i-|S(\pi)|},
\]

(17)

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

It remains to consider the contribution from \( \overline{\beta} \in B \) 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 \( B \). The only possibility is \( \overline{\beta}_1 = \ast \), which we split into two cases based on \( w(\pi) \).

**Case 1:** \( w(\pi) = 1 \). In this case, we claim that there are no \( \overline{\beta} \in B \) simultaneously satisfying (16) and \( \overline{\beta}_1 = \ast \). Suppose to the contrary. Then such a \( \overline{\beta}_1 \) must have at least one 1 in some other entry (as \( \overline{\beta} \in B \)), but this would imply that the resolution \( \overline{\pi} \setminus \overline{\beta} \) has at least two 1s, a contradiction. The total coefficient of \( h_\pi^* \) in this case is thus exactly given by (17). Upon substituting \( w(\pi) = 1 \), this simplifies to

\[
\sum_{i=|S(\pi)|}^{m+1-w(\pi)} (-1)^{m-i} \cdot \lambda_{i+1} \cdot \binom{d-m-1}{i-|S(\pi)|} = \sum_{i=|S(\pi)|}^{m+1-w(\pi)} (-1)^{m-i} \cdot \binom{d-i-2}{i-|S(\pi)|} \cdot \binom{d-m-1}{i-|S(\pi)|} = 0,
\]

where in the last step we use Lemma C.3 (which we can apply because of (15)).
Case 2: \( w(\tau) > 1 \). Observe that we must have \( w(\overline{\beta}) = w(\tau) - 1 \) (as the only entry of \( \tau \) equal to 1 is the first entry, and the first entry of \( \overline{\beta} \) is \( \ast \)). As \( w(\tau) - 1 > 0 \) in the current case, such a \( \overline{\beta} \) is an element of \( B \) if and only if it has at most \( m + 2 - w(\tau) \) free variables, and the set of free variables in \( \overline{\beta} \) must be \( \{1\} \cup S(\tau) \cup V \) where \( V \) is any subset of \( \{d\} \setminus (\{1\} \cup S(\tau)) \). Thus in this second case, the contribution of all \( \overline{\beta} \) with \( \overline{\beta}_1 = \ast \) to the coefficient of \( h_{\tau} \) in (14) is

\[
\sum_{i=|S(\tau)|+1}^{m+2-w(\tau)} (-1)^{m-i} \cdot 1_{i+1}(1) \cdot \binom{d-1}{i} \cdot 1_{|S(\tau)|-1}(1) = \sum_{j=|S(\tau)|}^{m+1-w(\tau)} (-1)^{m-j} \cdot 1_{j}(1) \cdot \binom{d-1}{j} \cdot 1_{|S(\tau)|-1}(1),
\]

where here the index \( i \) denotes the total number of free variables in \( \overline{\beta} \), the factor of \( \binom{d-1}{i} \cdot 1_{|S(\tau)|-1}(1) \) is the number of ways to choose \( V \) (note that \( |V| = i - |S(\tau)| - 1 \)), and in the second expression we made the change of variable \( j = i - 1 \). We conclude that in this case, the coefficient of \( h_{\tau} \) in (14) is given by the sum of (17) and (18), which is 0.

Overall, we conclude that the entire RHS of (14) vanishes for \( \alpha \in A \), proving the third part of the lemma. \( \square \)

We are ready to formally construct \( N_3 \) and verify 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 2.6.** Let

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

viewed as a function of \( s_1, \ldots, s_d \) parameterized by \( t \), and let \( \psi^* \) be as defined in Lemma C.2. Define \( \psi^*(s_1, \ldots, s_d; t) \).

Part (a) follows directly from Lemma C.2(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 \( \psi^* \), which is a combination of partial restrictions of \( \psi \), 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 \( \psi \) 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 C.2(a) and the fact that \( \psi \) contains \( O(d) \) ReLUs, the size of \( N_3 \) may be bounded by

\[
S \leq O(d) \cdot \left( \frac{d}{m} \right) (d-m) \cdot 3^m + 1 \leq O(d) \left( \frac{d^{m+1} \cdot 3^m}{m!} \right) + 1 \leq d^{m+2} \leq d^{2m}
\]

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 \( \psi(0, \ldots, 0; s) \) and \( \psi^*(0, \ldots, 0; s) \) as \( \psi(0, \ldots, 0, s) \) and \( \psi^*(0, \ldots, 0, s) \). We first compute \( \psi(0, \ldots, 0, s) \) for \( s \in [0,1] \),

\[
\psi(0, \ldots, 0, s) = \text{ReLU}(-s) + (d-1) \text{ReLU}(\frac{1}{d-1} \cdot s) = s.
\]
Next, for any $\vec{\beta} \in \mathcal{B}$, if $w(\vec{\beta}) = j$ for some $0 \leq j \leq m + 1$, then if $\vec{\beta}_d = \ast$,

$$
\psi_{\vec{\beta}}(0, \ldots, 0, s) = \psi(1, \ldots, 1, 0, \ldots, 0)_{d-j}
$$

$$
= j \cdot \text{ReLU} \left( -1 + \frac{1}{d-1}(j-1) + s \right) + \left( \frac{d - j - 1}{d - 1} \cdot j \right)
$$

$$
= \frac{d - j - 1}{d - 1} \cdot (j + s) + \text{ReLU} \left( -s + \frac{1}{d - 1} \cdot j \right)
$$

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

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

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

$$
\psi_{\vec{\beta}}(0, \ldots, 0, s) = \psi(1, \ldots, 1, 0, \ldots, 0)_{d-j}
$$

$$
= j \cdot \text{ReLU} \left( -1 + \frac{1}{d-1}(j-1) \right) + \left( \frac{d - j}{d - 1} \cdot j \right) = \frac{(d - j)j}{d - 1}.
$$

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

$$
\psi(0, \ldots, 0, s) - \sum_{i=0}^{m} \sum_{j=1}^{m+1-i} (-1)^{m-i} \binom{d - i - j - 1}{m - i - j + 1} \binom{d - 1}{i - 1} \binom{(d - j - s)j}{d - 1} + \binom{d - 1}{i} \binom{(d - j)j}{d - 1}.
$$

By Lemma C.4, the double sum is equal to zero, so $h^*(0, \ldots, 0, s) = h(0, \ldots, 0, s) = s$ for $s \in [0, 1/(d - 1)]$ as claimed.

### C.2 Supporting technical lemmas for Lemma 2.6

**Lemma C.3.** For any $0 \leq S < m \leq d$,

$$
\sum_{i=S}^{m} (-1)^{m-i} \binom{d - i - 2}{d - m - 1} \binom{d - m - 1}{i - S} = 0. \quad (19)
$$

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

$$
\sum_{k=0}^{\ell} (-1)^{k} \binom{j - k}{\ell - 1} \binom{\ell}{k} = 0. \quad (20)
$$

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 $\binom{d - m - 1}{i - S}$ is zero on the right-hand side of (19)), and $j \geq \ell$ by our assumption that $S < m$. We conclude the identity

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

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 (21) is equal, up to a sign, to the left-hand side of (19), so we’d be done.

It remains to establish (20), which we do by following an argument due to [Ear19]. Observe that the left-hand side of (20) is simply counting via inclusion-exclusion the number of subsets of \( \{1, \ldots, j\} \) of size \( \ell - 1 \) which contain \( \{1, \ldots, \ell\} \). Indeed, the \( k = 0 \) summand counts all subsets of size \( \ell - 1 \). The \( k = 1 \) summands subtract out the contribution, for every \( 1 \leq x \leq \ell \), from the subsets of size \( \ell - 1 \) which contain \( x \). The \( k = 2 \) summands add back the contribution, for every distinct \( 1 \leq x < y \leq \ell \), from the subsets of size \( \ell - 1 \) which contain both of \( x, y \), etc. \( \square \)

**Lemma C.4.** For any integers \( m \geq 3 \) and \( a \in \{0, 1, 2\} \),

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

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

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

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

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

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

\[
- \sum_{\ell' = 1}^{m+1} \sum_{i=1}^{\ell'} (-1)^{m-i} \binom{d - 1 - \ell'}{m + 1 - \ell'} \binom{d - 1}{i - 1} (\ell' + 1 - i)^a
\]

\[
= - \sum_{\ell' = 1}^{m+1} \sum_{i=1}^{\ell'} (-1)^{m-i} \binom{d - 1 - \ell'}{m + 1 - \ell'} \binom{d - 1}{i - 1} (\ell' - i)^a
\]

\[
- \sum_{\ell' = 1}^{m+1} \sum_{i=1}^{\ell'} (-1)^{m-i} \binom{d - 1 - \ell'}{m + 1 - \ell'} \binom{d - 1}{i - 1} \sum_{b=0}^{a-1} \binom{a}{b} (\ell' - i)^b
\]

(22)

Note that the first term on the right-hand side differs from \( S_{a,m} \) only in the summands given by \( 1 \leq i = \ell' \leq m + 1 \), and those summands clearly vanish. We conclude that the first term on the right-hand side of (22) is exactly \( S_{a,m} \). For the second term on the right-hand side of (22), the part coming from any \( 0 < b \leq a - 1 \) is also zero, so we thus get

\[
= S_{a,m} - \sum_{\ell' = 1}^{m+1} \sum_{i=1}^{\ell'} (-1)^{m-i} \binom{d - 1 - \ell'}{m + 1 - \ell'} \binom{d - 1}{i - 1}
\]

\[
= S_{a,m} - S_{0,m} - \sum_{\ell' = 1}^{m+1} (-1)^{m-\ell'} \binom{d - 1 - \ell'}{m + 1 - \ell'} (\ell' - 1)^a
\]

\[
= S_{a,m} - 1 - \sum_{\ell' = 1}^{m+1} (-1)^{m-\ell'} \binom{d - 1 - \ell'}{m + 1 - \ell'} (\ell' - 1)^a
\]

\[
= S_{a,m} = 1[a = 0],
\]

(23)

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 $i' = i + 1$,

$$T_{a,m} = - \sum_{\ell=1}^{m+1} \sum_{i'=1}^{\ell} (-1)^{m-i'} \left( \frac{d-1-\ell}{m+1-\ell} \right)^{d-1} \left( i' - 1 \right)^a$$

$$= - \sum_{\ell=2}^{m+1} \sum_{i'=1}^{\ell-1} (-1)^{m-i'} \left( \frac{d-1-\ell}{m+1-\ell} \right)^{d-1} \left( i' - 1 \right)^a - \sum_{\ell=1}^{m+1} (-1)^{m-\ell} \left( \frac{d-1-\ell}{m+1-\ell} \right)^{d-1} \left( \ell - 1 \right)^a$$

$$= - \sum_{b=0}^{a} \binom{a}{b} S_{b,m} + 1 = 0,$$

where in the second step we pulled out the summands corresponding to $i' = \ell$, in the third step we used (23), and in the last step we used that for $m \geq 3$, $S_{b,m} = \mathbb{1}[b \neq 0]$ for $0 \leq b \leq 2$. \hfill \Box

### C.3 Full proof of Theorem 2.3

**Continued proof of Theorem 2.3.** Let us compute the probability mass of the “good region” $G$. 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 $N_2(z_d) \in \left( \frac{1}{2d}, \frac{1}{d} \right)$. Consider the behavior of $N_2$ on just the interval $S_k$ that is closest to the origin (which will be $k = \lfloor q/2 \rfloor$); 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 $\left( \frac{1}{2d}, \frac{1}{d} \right)$ 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 $\Omega(|S_k|/d) = \Omega(\delta/qd) = \Omega(d^{-\Theta(m)})$. 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 $1/poly(S)$ and hence non-negligible in the size $S$ of the network.

The discrete learner $B$ can now adapt $\hat{f}$ as follows. Given a fresh test point $x \sim \text{Unif}(\mathbb{Z}_d^d)$, 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 $\hat{y} = \hat{f}(z)/N_2(z)$ (recall that when $z \in z, N_2(z_d) > \frac{1}{2d}$), and otherwise it simply predicts $\hat{y} = \frac{1}{2}$. The square loss of this predictor is given by

$$\mathbb{E}_{x \sim \text{Unif}(\mathbb{Z}_d^d)} \left[ (\hat{y} - f(x))^2 \right] = \mathbb{E}_{x \sim \text{Unif}(\mathbb{Z}_d^d)} \left[ (\hat{y} - f(x))^2 \right]$$

$$= \mathbb{E}_{x \sim \text{Unif}(\mathbb{Z}_d^d)} \left[ (\hat{f}(z) - f(x))^2 \right] \left[ z \in G \right] \mathbb{P}[z \in G] + \mathbb{E}_{x \sim \text{Unif}(\mathbb{Z}_d^d)} \left[ (\hat{y} - f(x))^2 \right] \left[ z \notin G \right] \mathbb{P}[z \notin G]$$

$$= \mathbb{E}_{x \sim \text{Unif}(\mathbb{Z}_d^d)} \left[ \left( \frac{\hat{f}(z)}{N_2(z)} - f(x) \right)^2 \right] \left[ z \in G \right] \mathbb{P}[z \in G] + \mathbb{E}_{x \sim \text{Unif}(\mathbb{Z}_d^d)} \left[ \left( \frac{1}{2} - f(x) \right)^2 \right] \left[ z \notin G \right] \mathbb{P}[z \notin G]$$

$$= \mathbb{E}_{x \sim \text{Unif}(\mathbb{Z}_d^d)} \left[ \left( \frac{\hat{f}(z)}{N_2(z)} - f^0(z) \right)^2 \right] \left[ z \in G \right] \mathbb{P}[z \in G] + \mathbb{E}_{x \sim \text{Unif}(\mathbb{Z}_d^d)} \left[ \left( \frac{1}{2} - f(x) \right)^2 \right] \left[ z \notin G \right] \mathbb{P}[z \notin G]$$

(by Eq. (8), when $z \in G, f^0(z) = f(x)N_2(z_d)$)

$$< 4d^2 \mathbb{E}_{z} \left[ (\hat{f}(z) - f^0(z))^2 \right] \left[ z \in G \right] \mathbb{P}[z \in G] + \mathbb{E}_{x} \left[ \left( \frac{1}{2} - f(x) \right)^2 \right] \mathbb{P}[z \notin G]$$

$$\leq 4d^2 \mathbb{E}_{z} \left[ (\hat{f}(z) - f^0(z))^2 \right] \left[ z \in G \right] \mathbb{P}[z \in G] + 4d^2 \mathbb{E}_{x} \left[ \left( \frac{1}{2} - f(x) \right)^2 \right] \mathbb{P}[z \notin G]$$

In the case of the hard classes $C$ that we consider, we may assume without loss of generality that $\mathbb{E}_{x \sim \text{Unif}(\mathbb{Z}_d^d)} \left[ (\frac{1}{2} - f(x))^2 \right] \geq 1/poly(d)$, since otherwise the problem of learning $C$ is trivial (in fact,
in our applications we will have \( \mathbb{E}_{x \sim \text{Unif}(Z_q^d)}[(\frac{1}{2} - f(x))^2] = \Theta(1) \). This means that by taking
\[
\varepsilon = \mathbb{P}[z \in G]/\text{poly}(d) = d^{-\Theta(m)}/\text{poly}(d) = d^{-\Theta(m)}
\]
sufficiently small (but still \( 1/\text{poly}(S) \)), we may ensure that the square loss of the discrete learner \( B \) is at most \( \mathbb{E}_{x \sim \text{Unif}(Z_q^d)}[(\frac{1}{2} - f(x))^2] - d^{-\Theta(m)} \), as desired. \( \Box \)

**Remark C.5.** The only property of the Gaussian \( \mathcal{N}(0, \text{Id}_d) \) 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 \( N_3 \) (i.e., adjusting the widths and locations of the intervals \( I_k, R_k, S_k \) depending on each \( P_i \)), the proof can be made to work more generally for such distributions \( P \).

**Corollary C.6 (Compressed DV lift with padding).** Let \( q, m \) and \( d \) be as above, and let \( d' = d^m \). View \( \mathcal{C} \) and \( \mathcal{C}^\circ \) as function classes on \( \mathbb{Z}_q^d \) and \( \mathbb{R}^d \) respectively, defined using only the first \( d \) coordinates, so that \( \mathcal{C}^\circ \) is now a poly\((d')\)-sized class over \( \mathbb{R}^d \). Then an algorithm capable of learning \( \mathcal{C}^\circ \) over \( \mathcal{N}_{d'} \) up to squared loss \( 1/\text{poly}(d') \) implies a weak predictor for \( \mathcal{C} \) over \( \text{Unif}(\mathbb{Z}_q^d) \) with advantage \( 1/\text{poly}(d') \).

### D Barriers for constructing \( N_3 \)

We briefly discuss why one natural approach to constructing \( N_3 \) satisfying the ideal properties in Eq. (2) ultimately requires two hidden layers rather than one, unlike the construction we give in Appendix C.

The most straightforward way to ensure that a function of \( s_1, \ldots, s_d, t \) vanishes whenever there exists \( j \) for which \( s_j = 1 \) would be to threshold on \( \sum_j s_j \), e.g. by taking \( \text{ReLU}(1 - \sum_j s_j) \). While this function is a one-hidden-layer \( \text{ReLU} \) network, it is unclear how to modify it to satisfy the remaining desiderata in (2) 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 (2) can be implemented as a one-hidden-layer network. Concretely, \( N(s, t) \) must vanish whenever \( s \geq 1 \) or \( t \in \mathbb{Z} \setminus \{0\} \). Any function computed by a one-hidden-layer \( \text{ReLU} \) network of the form \( (s, t) \mapsto \sum_j \text{ReLU}(a_j s + b_j t - c_j) \), 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 \( \{(s, t) : a_j s + b_j t = c_j\} \). 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 (2).

### E Proof of Theorem 3.2

**Proof of Theorem 3.2.** Recall that \( B \) is given SQ access to a distribution of pairs \( (x, y) \) where \( x \sim \text{Unif}(Z_q^d) \) and \( y = f(x) \) for an unknown \( f \in \mathcal{C} \). A request estimates \( \mathbb{E}[\phi(x, y)] \pm \tau \) for arbitrary bounded queries \( \phi : Z_q^d \times [0, 1] \to [-1, 1] \) and any desired \( \tau \). We know that given \( (x, y) \), the distribution of \( (z, \tilde{y}) \), 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 \( \tilde{y} = \tilde{g}(y, z) \) as in Eq. (7), is consistent with some \( f^\circ \in \mathcal{C}^\circ \) except on a region of probability mass at most \( d^{-9m^2} \) (recall Eq. (6)). Suppose we could simulate SQ access to the distribution of \( (z, f^\circ(z)) \) using only SQ access to that of \( (x, f(x)) \). Then by the argument in Theorem 2.3, simulating \( A \) on the \( (z, f^\circ(z)) \) distribution would give us a weak predictor \( \tilde{B} \) for the distribution of \( (x, f(x)) \), satisfying
\[
\mathbb{E} [(\tilde{B}(x) - f(x))^2] < \mathbb{E} [(\frac{1}{2} - f(x))^2] - d^{-\Theta(m)}.
\]

What we must describe is how \( B \) can simulate \( A \)'s statistical queries. Say \( A \) requests an estimate \( \mathbb{E}_z[\phi(z, f^\circ(z))] \pm \tau \) for some query \( \phi : \mathbb{R}^d \times \mathbb{R} \to [-1, 1] \). Consider the query \( \phi : Z_q^d \times [0, 1] \to \mathbb{R} \):

27
dependent is at most $\frac{1}{q}$. This function can be computed without any additional SQs, since the distribution of $(z, y) = (z(x), \overline{y}(y, z(x)))$, given $(x, y)$, is fully determined and known to $B$. Observe that

$$\mathbb{E}_{x,y} \overline{\phi}(x, y) = \mathbb{E}_{x,z(x)} [\phi(z(x), \overline{y}(y, z(x)))] = \mathbb{E}_{z,y} [\phi(z, y)]. \quad (24)$$

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

$$\left| \mathbb{E}_z [\phi(z, f^h(z))] - \mathbb{E}_{z,\overline{y}} [\phi(z, \overline{y})] \right| \leq \Theta(d^{-9m^2}) \leq \frac{\tau}{2} \quad (25)$$

since we assumed $\tau \geq d^{-\Theta(m^2)}$. Putting together (24) and (25), we see that $B$ can simulate $A$’s query $\phi$ to within tolerance $\tau$ by querying $\phi$ with tolerance $\tau/2$.

## F SQ lower bound via the LWR functions

The SQ lower bound obtained 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 $(z, f^h(z))$ where $f$ is an unknown parity. We know that whenever $z$ lands in the “good region” $G$ from the proof of Theorem 2.3 (which happens with non-negligible probability), we have $f^h(z) = f(\text{sign}(z))N_2(z)$ (recall Eq. (8)). This means we can simply filter out all $z \notin G$ and form a clean data set of labeled points $(\text{sign}(z), f(\text{sign}(z)))$. The unknown $f$ (and hence $f^h$) can now be learnt by simple Gaussian elimination.

In order to give a more convincing lower bound, we now provide an alternative proof of Theorem 3.1 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 showing an SQ lower bound for the LWR functions 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 [Bog21]. A variant of this argument may be found in [BR17, §7.7].

**Definition F.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 $\text{Unif}(\mathcal{Y}) \otimes \text{Unif}(\mathcal{Y})$.

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

**Proof.** This follows from the simple observation that whenever $x, x' \in \mathbb{Z}_q^n$ are linearly independent, the pair $(w \cdot x \mod q, w \cdot x' \mod q)$ for $w \sim \text{Unif}(\mathbb{Z}_q^n)$ is distributed as $\text{Unif}(\mathbb{Z}_q) \otimes \text{Unif}(\mathbb{Z}_q)$. For such $x, x'$, $(f_w(x), f_w(x')) = (\frac{1}{p}w \cdot x \mod q, \frac{1}{p}w \cdot x' \mod q)$ for $f_w \sim \text{Unif}(\mathcal{C}_{\text{LWR}})$ is distributed as $\text{Unif}(\mathbb{Z}_p/p) \otimes \text{Unif}(\mathbb{Z}_p/p)$. The probability that $x, x' \sim \text{Unif}(\mathbb{Z}_q^n)$ 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 F.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

$$\operatorname{Var}_{f \sim \text{Unif}(\mathcal{C})} \mathbb{E}_{x \sim D} [\phi(x, f(x))] \leq 2\eta.$$
Proof. Denote $E_{x \sim D}[\phi(x, f(x))]$ by $\phi[f]$. By some algebraic manipulations (with all subscripts denoting independent draws),
\[
\begin{align*}
\var_{f \sim \text{Unif}(C)} [\phi[f]] &= \mathbb{E}_f [\phi[f]^2] - (\mathbb{E}_f [\phi[f]])^2 \\
&= \mathbb{E}_f [\phi[f] \phi[f]] - \mathbb{E}_f [\phi[f]] \mathbb{E}_f [\phi[f]'] \\
&= \mathbb{E}_{f, f'} [\mathbb{E}_{x \sim D} [\phi(x, f(x))] \mathbb{E}_{x' \sim D} [\phi(x', f(x'))] - \mathbb{E}_x [\phi(x, f(x))]{\mathbb{E}_x}_f [\phi(x', f(x'))]] \\
&= \mathbb{E}_{x, x' \sim D} \mathbb{E} \mathbb{E}_f \mathbb{E}_{x' \sim D} [\phi(x, f(x)) \phi(x', f(x')) - \phi(x, f(x)) \phi(x', f'(x'))].
\end{align*}
\]
By $(1 - \eta)$-pairwise independence of $C$, 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 F.4. Let $C$ mapping $\mathcal{X} \times \mathcal{Y}$ be a $(1 - \eta)$-pairwise independent function family w.r.t. a distribution $D$ on $\mathcal{X}$. For any $f \in C$, let $D_f$ denote the distribution of $(x, f(x))$ where $x \sim D$. Let $D_{\text{Unif}(C)}$ denote the distribution of $(x, y)$ where $x \sim D$ and $y = f(x)$ for $f \sim \text{Unif}(C)$ (this can be thought of as essentially $D \otimes \text{Unif}(\mathcal{Y})$). Any SQ learner able to distinguish the labeled distribution $D_f^*$ for an unknown $f^* \in C$ from the randomly labeled distribution $D_{\text{Unif}(C)}$ using bounded queries of tolerance $\tau$ requires at least $\frac{\tau^2}{2\eta}$ such queries.

Proof. Let $\phi : \mathcal{X} \times \mathcal{Y} \rightarrow [-1, 1]$ be any query made by the learner. For any $f \in C$, let $\phi[f]$ denote $E_{x \sim D}[\phi(x, f(x))] = E_{(x, y) \sim D_f} [\phi(x, y)]$. Consider the adversarial strategy where the SQ oracle responds to this query with $\overline{\phi} = E_{f \sim \text{Unif}(C)} \phi[f] = E_{(x, y) \sim D_{\text{Unif}(C)}} [\phi(x, y)]$. By Chebyshev’s inequality and Lemma F.3,
\[
P_{f \sim C} [\phi[f] - \overline{\phi}] > \tau \leq \frac{\var_{f \sim \text{Unif}(C)} [\phi[f]]}{\tau^2} \leq \frac{2\eta}{\tau^2}.
\]
So each such query only allows the learner to rule out at most a $\frac{2\eta}{\tau^2}$ fraction of $C$. Thus to distinguish $D_{f^*}$ from $D_{\text{Unif}(C)}$, the learner requires at least $\frac{\tau^2}{2\eta}$ queries. 

Theorem F.5 now follows easily as a corollary.

Theorem F.5. Let $\mathcal{C}_{\text{LWR}}$ denote the $\text{LWR}_{n, p, q}$ function class. Any SQ learner capable of learning $\mathcal{C}_{\text{LWR}}$ up to squared loss $1/16$ under $\text{Unif}(\mathbb{Z}_q^n)$ using queries of tolerance $\tau$ requires at least $\Omega(q^{n-1}\tau^2)$ such queries.

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

We are ready for an alternative proof of Theorem 3.1.

Alternative proof of Theorem 3.1. Let $n$ be the security parameter, and fix moduli $p, q \geq 1$ such that $p, q = \text{poly}(n)$ and $p/q = \text{poly}(n)$. Let $d = n$, so that the SQ lower bound from Theorem F.5 is $\Omega(q^{n-1}) = d^{\Omega(d)} = 2^{\Omega(d)}$. Let $m = m(d) = \log_c d$ for $c = \frac{1}{n} - 1$, and let $d' = d^m = 2^{\log_c^{1+\epsilon} d}$, so that $d = 2^{\log_c^{1/(1+c\epsilon)} d'}$. Recall from the proof of Theorem 4.1 that the $\text{LWR}_{n, p, q}$ function class $\mathcal{C}_{\text{LWR}}$ is implementable by one-hidden-layer ReLU networks over $\mathbb{Z}_q^d$ of size $\text{poly}(n) = \text{poly}(d)$. The result now follows by Theorem 3.2 and the same padding argument as in the proof based on parities.