Journal of Machine Learning Research 24 (2023) 1-52 Submitted 1/23; Revised 11/23; Published 11/23

Optimal Approximation Rates for Deep ReLU Neural Networks on Sobolev and Besov Spaces

Jonathan W. Siegel JWSIEGEL@TAMU.EDU
Department of Mathematics
Texas A&M University
College Station, TX 77843 USA

Editor: Bharath Sriperumbudur

Abstract

Let $\Omega = [0,1]^d$ be the unit cube in $\mathbb{R}^d$. We study the problem of how efficiently, in terms of the number of parameters, deep neural networks with the ReLU activation function can approximate functions in the Sobolev spaces $W^s(L_q(\Omega))$ and Besov spaces $B^s_r(L_q(\Omega))$, with error measured in the $L_p(\Omega)$ norm. This problem is important when studying the application of neural networks in a variety of fields, including scientific computing and signal processing, and has previously been solved only when $p = q = \infty$. Our contribution is to provide a complete solution for all $1 \leq p,q \leq \infty$ and $s > 0$ for which the corresponding Sobolev or Besov space compactly embeds into L_p. The key technical tool is a novel bit-extraction technique which gives an optimal encoding of sparse vectors. This enables us to obtain sharp upper bounds in the non-linear regime where $p > q$. We also provide a novel method for deriving L_p-approximation lower bounds based upon VC-dimension when $p < \infty$. Our results show that very deep ReLU networks significantly outperform classical methods of approximation in terms of the number of parameters, but that this comes at the cost of parameters which are not encodable.

1. Introduction

Deep neural networks have achieved remarkable success in both machine learning (LeCun et al., 2015) and scientific computing (Raissi et al., 2019; Han et al., 2018). However, a precise theoretical understanding of why deep neural networks are so powerful has not been attained and is an active area of research. An important part of this theory is the study of the approximation properties of deep neural networks, i.e. to understand how efficiently a given class of functions can be approximated using deep neural networks. In this work, we solve this problem for the class of deep ReLU neural networks (Nair and Hinton, 2010) when approximating functions lying in a Sobolev or Besov space with error measured in the L_p-norm. We remark that the ReLU activation functions is very widely used and is a major driver of many recent breakthroughs in deep learning (Goodfellow et al., 2016; LeCun et al., 2015; Nair and Hinton, 2010).

Let us begin by giving a description of the Sobolev function classes, which are widely used in the theory of solutions to partial differential equations (PDEs) (Evans, 2010), and the Besov function classes, which are widely used in approximation theory (DeVore and Lorentz, 1993), statistics (Donoho and Johnstone, 1995, 1998), and signal processing (DeVore et al., 1992).

Let $\Omega \subset \mathbb{R}^d$ be a bounded domain, which we take to be the unit cube $\Omega = [0,1]^d$ in the following. Due to a variety of extension theorems for Sobolev and Besov spaces (see for instance Evans

(2010); Di Nezza et al. (2012); DeVore and Lorentz (1993); Whitney (1934)), this is not a significant restriction and our results will apply to many other sufficiently well-behaved domains. We denote by $L_p(\Omega)$ the set of functions f for which the L_p-norm on Ω is finite, i.e.

$$\|f\|_{L_p(\Omega)} = \left(\int_\Omega |f(x)|^p dx\right)^{1/p} < \infty.$$

When $p = \infty$, this becomes $\|f\|_{L_\infty(\Omega)} = \operatorname{ess\,sup}_{x\in\Omega}|f(x)|$. Suppose that $s > 0$ is a positive integer. Then $f \in W^s(L_q(\Omega))$ is in the Sobolev space (see Demengel et al. (2012), Chapter 2 for instance) with s derivatives in L_q if f has weak derivatives of order s and

$$\|f\|^q_{W^s(L_q(\Omega))} := \|f\|^q_{L_q(\Omega)} + \sum_{|\alpha|=k} \|D^\alpha f\|^q_{L_q(\Omega)} < \infty.$$

Here $\alpha = (\alpha_i)_{i=1}^d$ with $\alpha_i \in \mathbb{Z}_{\geq 0}$ is a multi-index and $|\alpha| = \sum_{i=1}^d \alpha_i$ is the total degree. The $W^s(L_q(\Omega))$ semi-norm is defined by

$$|f|_{W^s(L_q(\Omega))} := \left(\sum_{|\alpha|=k} \|D^\alpha f\|^q_{L_q(\Omega)}\right)^{1/q}, \tag{1.1}$$

and the standard modifications are made when $q = \infty$.

When $s > 0$ is not an integer, we write $s = k + \theta$ with $k \geq 0$ an integer and $\theta \in (0,1)$. The Sobolev semi-norm is defined by (see Demengel et al. (2012) Chapter 4 or Di Nezza et al. (2012) Chapter 1 for instance)

$$|f|^q_{W^s(L_q(\Omega))} := \int_{\Omega\times\Omega} \frac{|D^\alpha f(x) - D^\alpha f(y)|^q}{|x-y|^{d+\theta q}} dxdy \tag{1.2}$$

when $1 \leq q < \infty$ and

$$|f|_{W^s(L_\infty(\Omega))} := \sup_{|\alpha|=k} \sup_{x,y\in\Omega} \frac{|D^\alpha f(x) - D^\alpha f(y)|}{|x-y|^\theta}.$$

We define the Sobolev norm by

$$\|f\|^q_{W^s(L_q(\Omega))} := \|f\|^q_{L_q(\Omega)} + |f|^q_{W^s(L_q(\Omega))},$$

with the usual modification when $q = \infty$. We remark that in the case of non-integral s these spaces are also called Sobolev-Slobodeckij spaces. Sobolev spaces are widely used in PDE theory and a priori estimates for PDE solutions are often given in terms of Sobolev norms (Evans, 2010). For applications of neural networks to scientific computing it is thus important to understand how efficiently neural networks can approximate functions from $W^s(L_q(\Omega))$.

Next, we consider the Besov spaces, which we define in terms of moduli of smoothness. Given a function $f \in L_q(\Omega)$ and an integer k, the k-th order modulus of smoothness of f is given by

$$\omega_k(f,t)_q = \sup_{|h|\leq t} \|\Delta_h^k f\|_{L_q(\Omega_{kh})}, \tag{1.3}$$

where $h \in \mathbb{R}^d$, the k-th order finite difference Δ_h^k is defined by

$$\Delta_h^k f(x) = \sum_{j=0}^k (-1)^j \binom{k}{j} f(x+jh),$$

and the L_q norm is taken over the set $\Omega_{kh} := \{x \in \Omega,\ x+kh \in \Omega\}$ to guarantee that all terms of the finite difference are contained in the domain Ω. Fix an integer $k > s$. The Besov space $B^s_r(L_q(\Omega))$ is defined via the norm

$$\|f\|_{B^s_r(L_q(\Omega))} := \|f\|_{L_q(\Omega)} + |f|_{B^s_r(L_q(\Omega))},$$

with Besov semi-norm given by

$$|f|_{B^s_r(L_q(\Omega))} := \left(\int_0^\infty \frac{\omega_k(f,t)_q^r}{t^{sr+1}} dt\right)^{1/r}$$

when $r < \infty$ and by

$$|f|_{B^s_\infty(L_q(\Omega))} := \sup_{t>0} t^{-s}\omega_k(f,t)_q,$$

when $r = \infty$. It can be shown that different choices of $k > s$ result in equivalent norms (DeVore and Lorentz, 1993). One can think of the Besov space $B^s_r(L_q(\Omega))$ roughly as being a space of functions with s derivatives lying in L_q, similar to the Sobolev space $W^s(L_q(\Omega))$, with the additional index r providing a finer gradation. Indeed, a variety of embedding and interpolation results relating Besov spaces and Sobolev spaces are known (see for instance DeVore and Popov (1988); DeVore and Sharpley (1984); Yuan et al. (2010); Kufner et al. (1977)).

Besov spaces are central objects in approximation theory due to their close connection with approximation by trigonometric polynomials (on the circle) and splines (DeVore and Lorentz, 1993; Petrushev, 1988). In fact, there are equivalent definitions of the Besov semi-norms in terms of approximation error by trigonometric polynomials and splines. They are also closely connected to the theory of wavelets (Daubechies, 1992), and one can give equivalent definitions of the Besov norms in terms of the wavelet coefficients of f as well (DeVore et al., 1992). For this reason, Besov spaces play an important role in signal processing (Chambolle et al., 1998; Donoho et al., 1998) and statistical recovery of functions from point samples (Donoho and Johnstone, 1995, 1998), for instance.

Our goal is to study the approximation of Sobolev and Besov functions by neural networks. One of the most important classes of neural networks are deep ReLU neural networks, which we define as follows. We use the notation $A_{\mathbf{W},b}$ to denote the affine map with weight matrix $\mathbf{W}$ and offset, or bias, b, i.e.

$$A_{\mathbf{W},b}(x) = \mathbf{W}x + b.$$

When the weight matrix $\mathbf{W}$ is an $k \times n$ and the bias $b \in \mathbb{R}^k$, the function $A_{\mathbf{W},b} : \mathbb{R}^n \to \mathbb{R}^k$ maps $\mathbb{R}^n$ to $\mathbb{R}^k$. Let σ denote the ReLU activation function (Nair and Hinton, 2010), specifically

$$\sigma(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0. \end{cases}$$

The ReLU activation function σ has become ubiquitous in deep learning in the last decade and is used in most state-of-the-art architectures. Since σ is continuous and piecewise linear, it also has the nice theoretical property that neural networks with ReLU activation function represent continuous piecewise linear functions. This property has been extensively studied in the computer science literature (Arora et al., 2018; Wang and Sun, 2005; Serra et al., 2018; Hanin and Rolnick, 2019) and has been connected with traditional linear finite element methods (He et al., 2020).

When $x \in \mathbb{R}^n$, we write $\sigma(x)$ to denote the application of the activation function σ to each component of x separately, i.e. $\sigma(x)_i = \sigma(x_i)$. The set of deep ReLU neural networks with width W and depth L mapping $\mathbb{R}^d$ to $\mathbb{R}^k$ is given by

$$\Upsilon^{W,L}(\mathbb{R}^d,\mathbb{R}^k) := \{A_{\mathbf{W}_L,b_L} \circ \sigma \circ A_{\mathbf{W}_{L-1},b_{L-1}} \circ \sigma \circ \cdots \circ \sigma \circ A_{\mathbf{W}_1,b_1} \circ \sigma \circ A_{\mathbf{W}_0,b_0}\},$$

where the weight matrices satisfy $\mathbf{W}_L \in \mathbb{R}^{k\times W}$, $\mathbf{W}_0 \in \mathbb{R}^{W\times d}$, and $\mathbf{W}_1,...,\mathbf{W}_{L-1} \in \mathbb{R}^{W\times W}$, and the biases satisfy $b_0,...,b_{L-1} \in \mathbb{R}^W$ and $b_L \in \mathbb{R}^k$. Notice that our definition of width does not include the input and output dimensions and only includes the intermediate layers. When the depth $L = 0$, i.e. when the network is an affine function, there are no intermediate layers and the width is undefined, in this case we write $\Upsilon^0(\mathbb{R}^d,\mathbb{R}^k)$. We also use the notation

$$\Upsilon^{W,L}(\mathbb{R}^d) := \Upsilon^{W,L}(\mathbb{R}^d,\mathbb{R})$$

to denote the set of deep ReLU neural networks with width W and depth L which represent scalar functions. We note that our notation only allows neural networks with fixed width. We do this to avoid excessively cumbersome notation. We remark that the dimension of any hidden layer can naturally be expanded and thus any fully connected network can be made to have a fixed width, so that this restriction is without any significant loss of generality.

The problem we study in this work is to determine optimal L_p-approximation rates

$$\sup_{\|f\|_{W^s(L_q(\Omega))}\leq 1} \left(\inf_{f_L\in\Upsilon^{W,L}(\mathbb{R}^d)} \|f - f_L\|_{L_p(\Omega)}\right) \text{ and } \sup_{\|f\|_{B^s_r(L_q(\Omega))}\leq 1} \left(\inf_{f_L\in\Upsilon^{W,L}(\mathbb{R}^d)} \|f - f_L\|_{L_p(\Omega)}\right) \tag{1.4}$$

for the class of Sobolev and Besov functions using very deep ReLU networks, i.e. using networks with a fixed (large enough) width W and depth $L \to \infty$. We will prove that this gives the best possible approximation rate in terms of the number of parameters. One can more generally consider approximation error in terms of both the width W and depth L simultaneously (Shen et al., 2022), but we leave this more general analysis as future work.

This problem has been previously solved (up to logarithmic factors) in the case where $p = q = \infty$, where the optimal rate is given by

$$\inf_{f_L\in\Upsilon^{W,L}(\mathbb{R}^d)} \|f - f_L\|_{L_\infty(\Omega)} \leq C\|f\|_{W^s(L_\infty(\Omega))}L^{-2s/d} \tag{1.5}$$

for a sufficiently large but fixed width W. Specifically, this result was obtained for $0 < s \leq 1$ in Yarotsky (2018) and for all $s > 0$ (up to logarithmic factors) in Lu et al. (2021). An analogous result also holds for $B^s_r(L_\infty(\Omega))$ for $1 \leq r \leq \infty$. Further, the best rate when both the width and depth vary (which generalizes (1.5)) has been obtained in Shen et al. (2022).

The method of proof in these cases uses the bit-extraction technique introduced in Bartlett et al. (1998) and developed further in Bartlett et al. (2019) to approximate piecewise polynomial functions on a fixed regular grid with N cells using only $O(\sqrt{N})$ parameters. This enables an approximation rate of $CN^{-2s/d}$ in terms of the number of parameters N, which is significantly faster than traditional methods of approximation. This phenomenon has been called the *super-convergence* of deep ReLU networks (Yarotsky, 2018; Shen et al., 2022; DeVore et al., 2021; Daubechies et al., 2022b). The super-convergence has a limit, however, and the rate (1.5) is shown to be optimal using the VC-dimension of deep ReLU neural networks (Yarotsky, 2018; Shen et al., 2022; Bartlett et al., 2019).

In this work, we generalize this analysis to determine the optimal approximation rates (1.4) for all $1 \leq p,q \leq \infty$ and $s > 0$, i.e. to the approximation of any Sobolev or Besov class in $L_p(\Omega)$, with the possible exception of the Sobolev embedding endpoint (described below). This was posed as a significant open problem in DeVore et al. (2021). We remark that the existing upper bounds in L_∞ clearly imply corresponding upper bounds in L_p for $p < \infty$. The key problem lies in extending the upper bounds to that case where $q < \infty$, in which case we must approximate a larger function class. A further problem is the extension of the lower bounds to the case $p < \infty$, in which we are measuring error in a weaker norm.

A necessary condition that we have any approximation rate in (1.4) at all is for the Sobolev space $W^s(L_q(\Omega))$ or Besov space $B^s_r(L_q(\Omega)$ to be contained in L_p, i.e. $W^s(L_q(\Omega)), B^s_r(L_q(\Omega) \subset L_p(\Omega)$. Indeed, any deep ReLU neural network represents a continuous function and so if $f \notin L_p(\Omega)$ it cannot be approximated at all by deep ReLU networks. We will in fact consider the case where we have a compact embedding $W^s(L_q(\Omega)), B^s_r(L_q(\Omega) \subset\subset L_p(\Omega)$. Here the symbol $A \subset\subset B$ for two Banach spaces A and B means that A is contained in B and the unit ball of A is a compact subset of B. This compact embedding is guaranteed for both Besov and Sobolev spaces by the strict Sobolev embedding condition

$$\frac{1}{q} - \frac{1}{p} - \frac{s}{d} < 0. \tag{1.6}$$

We determine the optimal rates in (1.4) under this condition. Specifically, we prove the following Theorems. The first two give an upper bound on the approximation rate by deep ReLU networks on Sobolev and Besov spaces, respectively.

Theorem 1 *Let $\Omega = [0,1]^d$ be the unit cube in $\mathbb{R}^d$ and let $0 < s < \infty$ and $1 \leq p,q \leq \infty$. Assume that $\frac{1}{q} - \frac{1}{p} < \frac{s}{d}$, which guarantees that we have the compact embedding*

$$W^s(L_q(\Omega)) \subset\subset L^p(\Omega).$$

Then we have that

$$\inf_{f_L \in \Upsilon^{25d+31,L}(\mathbb{R}^d)} \|f - f_L\|_{L_p(\Omega)} \leq C\|f\|_{W^s(L_q(\Omega))} L^{-2s/d}$$

for a constant $C := C(s,q,p,d) < \infty$.

Theorem 2 *Let $\Omega = [0,1]^d$ be the unit cube in $\mathbb{R}^d$ and let $0 < s < \infty$ and $1 \leq r,p,q \leq \infty$. Assume that $\frac{1}{q} - \frac{1}{p} < \frac{s}{d}$, which guarantees that we have the compact embedding*

$$B^s_r(L_q(\Omega)) \subset\subset L^p(\Omega).$$

Then we have that

$$\inf_{f_L \in \Upsilon^{25d+31,L}(\mathbb{R}^d)} \|f - f_L\|_{L_p(\Omega)} \leq C\|f\|_{B^s_r(L_q(\Omega))} L^{-2s/d}$$

for a constant $C := C(s,r,q,p,d) < \infty$.

Note that the width $W = 25d + 31$ of our networks are fixed as $L \to \infty$, but scale linearly with the input dimension d. We remark that a linear scaling with the input dimension is necessary since if $d \geq W$, then the set of deep ReLU networks is known to not be dense in $C(\Omega)$ (Hanin, 2019). The next Theorem gives a lower bound which shows that the rates in Theorems 1 and 2 are sharp in terms of the number of parameters.

Theorem 3 *Let $r,p,q \geq 1$ and $s > 0$, $\Omega = [0,1]^d$ be the unit cube, and $W,L \geq 1$ be integers. Then there exists an f with $\|f\|_{W^s(L_q(\Omega))} \leq 1$ and $\|f\|_{B^s_r(L_q(\Omega))} \leq 1$ such that*

$$\inf_{f_{W,L}\in\Upsilon^{W,L}(\mathbb{R}^d)} \|f - f_{W,L}\|_{L_p(\Omega)} \geq C(p,d,s)\min\{W^2L^2\log(WL), W^3L^2\}^{-s/d}.$$

We remark that if the embedding condition (1.6) strictly fails, then a simply scaling argument shows that $W^s(L_q(\Omega)), B^s_r(L_q(\Omega)) \not\subseteq L_p(\Omega)$ and we cannot get any approximation rate. On the boundary where the embedding condition (1.6) holds with equality it is not a priori clear whether one has an embedding or not (this depends on the precise values of s,p,q and r). Consequently this boundary case is much more subtle and we leave this for future work.

The key technical difficulty in proving Theorem 1 is to deal with the case when $p > q$, i.e. when the target function's (weak) derivatives are in a weaker norm than the error. Classical methods of approximation using piecewise polynomials or wavelets can attain an approximation rate of $CN^{-s/d}$ with N wavelet coefficients or piecewise polynomials with N pieces. When $p \leq q$ this rate can be achieved by linear methods, while for $p > q$ nonlinear, i.e. adaptive, methods are required. For the precise details of this theory, see for instance DeVore and Lorentz (1993); Lorentz et al. (1996); DeVore (1998).

Thus, in the linear regime where $p \leq q$ we can use piecewise polynomials on a fixed *uniform* grid to approximate f, while in the non-linear regime we need to use piecewise polynomials on an adaptive (i.e. depending upon f) *non-uniform* grid. This greatly complicates the bit-extraction technique used to obtain super-convergence, since the methods in Yarotsky (2018); Shen et al. (2022); Shijun (2021) are only applicable to regular grids. The tool that we develop to overcome this difficulty is a novel bit-extraction technique, presented in Theorem 14, which optimally encodes *sparse* vectors using deep ReLU networks. Specifically, suppose that $\mathbf{x} \in \mathbb{Z}^N$ is an N-dimensional integer vector with ℓ^1-norm bounded by

$$\|\mathbf{x}\|_{\ell^1} \leq M.$$

In Theorem 14 we give (depending upon N and M) a deep ReLU neural network construction which optimally encodes $\mathbf{x}$.

We remark, however, that super-convergence comes at the cost of parameters which are non-encodable, i.e. cannot be encoded using a fixed number of bits, and this makes the numerical realization of this approximation rate inherently unstable. In order to better understand this, we recall the notion of metric entropy first introduced by Kolmogorov. The metric entropy numbers $\varepsilon_N(A)$ of a set $A \subset X$ in a Banach space X are given by (see for instance Lorentz et al. (1996), Chapter 15)

$$\varepsilon_N(A)_H = \inf\{\varepsilon > 0 : \text{ } A \text{ is covered by } 2^N \text{ balls of radius } \varepsilon\}.$$

An encodable approximation method consists of two maps, an encoding map $E : A \rightarrow \{0,1\}^N$ mapping the class A to a bit-string of length N, and a decoding map $D : \{0,1\}^N \rightarrow X$ which maps each bit-string to an element of X. This reflects the fact that any method which is implemented on a classical computer must ultimately encode all parameters using some number of bits. The metric entropy numbers give the minimal reconstruction error of the best possible encoding scheme.

Let $U^s(L_q(\Omega)) := \{f : \|f\|_{W^s(L_q(\Omega))} \leq 1\}$ denote the unit ball of the Sobolev space $W^s(L_q(\Omega))$. The metric entropy of this function class is given by

$$\varepsilon_N(U^s(L_q(\Omega)))_{L^p(\Omega)} \eqsim N^{-s/d}$$

whenever the Sobolev embedding condition (1.6) is strictly satisfied. This is known as the Birman-Solomyak Theorem (Birman and Solomyak, 1967). The same asymptotics for the metric entropy also hold for the unit balls in the Besov spaces $B^s_r(L_q(\Omega))$ if the compact embedding condition (4.10) is satisfied. So the approximation rates in Theorems 1 and 2 are significantly smaller than the metric entropy of the Sobolev and Besov classes. This manifests itself in the fact that in the construction of the upper bounds in Theorems 1 and 2 the parameters of the neural network cannot be specified using a fixed number of bits, but rather need to be specified to higher and higher accuracy as the network grows (Yarotsky and Zhevnerchuk, 2020), which is a direct consequence of the bit-extraction technique.

Concerning the lower bounds, the key difficulty in proving Theorem 3 is to extend the VC-dimension arguments used to obtain lower bounds when the error is measured in L_∞ to the case when the error is measured in the weaker norm L_p for $p < \infty$. We do this by proving Theorem 22, which gives a general lower bound for L_p-approximation of Sobolev spaces by classes with bounded VC dimension. We have recently learned of a different approach to obtaining L_p lower bounds using VC-dimension (Achour et al., 2022), which is more generally applicable but introduces additional logarithmic factors in the lower bound.

We remark that there are other results in the literature which obtain approximation rates for deep ReLU networks on Sobolev spaces, but which do not achieve superconvergence, i.e. for which the approximation rate is only $CN^{-s/d}$ (up to logarithmic factors), where N is the number of parameters (Yarotsky, 2017; Gühring et al., 2020). In addition, the approximation of other novel function classes (other than Sobolev spaces, which suffer the curse of dimensionality) by neural networks has been extensively studied recently, see for instance Daubechies et al. (2022b); Petersen and Voigtlaender (2018); Daubechies et al. (2022a); Siegel and Xu (2020, 2022a,b); Bach (2017); Klusowski and Barron (2018).

Finally, we remark that although we focus on the ReLU activation function due to its popularity and to simplify the presentation, our results also apply to more general activation functions as well. Specifically, the lower bounds in Theorem 3 based upon VC-dimension hold for any piecewise polynomial activation function. The upper bounds in Theorems 1 and 2 hold as long as we can approximate the ReLU to arbitrary accuracy on compact subsets (i.e. finite intervals) using a network with a fixed size. Using finite differences this can be done for the ReLU^k activation functions, defined by

$$\sigma_k(x) = \begin{cases} 0 & x < 0 \\ x^k & x \geq 0, \end{cases}$$

when $k \geq 1$ for instance. In fact, a similar construction using finite differences can approximate the ReLU as long as the activation function is a continuous piecewise polynomial which is not identically a polynomial.

The rest of the paper is organized as follows. First, in Section 2 we describe a variety of deep ReLU neural network constructions which will be used to prove Theorem 1. Many of these constructions are trivial or well-known, but we collect them for use in the following Sections. Then, in Section 3 we prove Theorem 14 which gives an optimal representation of sparse vectors using deep ReLU networks and will be key to proving superconvergence in the non-linear regime $p > q$. In Section 4 we give the proof of the upper bounds in Theorems 1 and 2. Finally, in Section 5 we prove the lower bound Theorem 3 and also prove the optimality of Theorem 14. We remark that throughout the paper, unless otherwise specified, C will represent a constant which may change from line to line, as is standard in analysis. The constant C may depend upon some parameters and this dependence will be made clear in the presentation.

2. Basic Neural Network Constructions

In this section, we collect some important deep ReLU neural network constructions which will be fundamental in our construction of approximations to Sobolev and Besov functions. Many of these constructions are well-known and will be used repeatedly to construct more complex networks later on, so we collect them here for the reader's convenience.

We being by making some fundamental observations and constructing some basic networks. Much of these are trivial consequences of the definitions, but we collect them here for future reference. We begin by noting that by definition we can compose two networks by summing their depths.

Lemma 4 (Composing Networks) *Suppose $L_1, L_2 \geq 1$ and that $f \in \Upsilon^{W,L_1}(\mathbb{R}^d, \mathbb{R}^k)$ and $g \in \Upsilon^{W,L_2}(\mathbb{R}^k, \mathbb{R}^l)$. Then the composition satisfies*

$$g(f(x)) \in \Upsilon^{W,L_1+L_2}(\mathbb{R}^d, \mathbb{R}^l).$$

Further, if f is affine, i.e. $f \in \Upsilon^0(\mathbb{R}^d, \mathbb{R}^k)$, then

$$g(f(x)) \in \Upsilon^{W,L_2}(\mathbb{R}^d, \mathbb{R}^l).$$

Finally, if instead g is affine, i.e. $g \in \Upsilon^0(\mathbb{R}^k, \mathbb{R}^l)$ then

$$g(f(x)) \in \Upsilon^{W,L_1}(\mathbb{R}^d, \mathbb{R}^l)$$

We remark that combining this with the simple fact that we can always increase the width of a network, we can apply Lemma 4 to networks with different widths and the width of the resulting network will be the maximum of the two widths. We will use this extension without comment in the following.

Next, we give a simple construction allowing us to apply two networks networks in parallel.

Lemma 5 (Concatenating Networks) *Let $d = d_1 + d_2$ and $k = k_1 + k_2$ with $d_i, k_i \geq 1$. Suppose that $f_1 \in \Upsilon^{W_1,L}(\mathbb{R}^{d_1}, \mathbb{R}^{k_1})$ and $f_2 \in \Upsilon^{W_2,L}(\mathbb{R}^{d_2}, \mathbb{R}^{k_2})$. We view $\mathbb{R}^d = \mathbb{R}^{d_1} \oplus \mathbb{R}^{d_2}$ and $\mathbb{R}^k = \mathbb{R}^{k_1} \oplus \mathbb{R}^{k_2}$. Then the function $f = f_1 \oplus f_2 : \mathbb{R}^d \to \mathbb{R}^k$ defined by*

$$(f_1 \oplus f_2)(x_1 \oplus x_2) = f_1(x_1) \oplus f_2(x_2)$$

satisfies $f_1 \oplus f_2 \in \Upsilon^{W_1+W_2,L}(\mathbb{R}^d, \mathbb{R}^k)$.

Proof This follows by setting the weight matrices $\mathbf{W}_i = \mathbf{W}_i^1 \oplus \mathbf{W}_i^2$ and $b_i = b_i^1 \oplus b_i^2$, where $\mathbf{W}_i^1, b_i^1$ and $\mathbf{W}_i^2, b_i^2$ represent the parameters defining f_1 and f_2 respectively. Recally that the direct sum of matrices is simply given by

$$A \oplus B = \begin{pmatrix} A & \mathbf{0} \\ \mathbf{0} & B \end{pmatrix}.$$

■

Note that this result can be applied recursively to concatenate multiple networks. Combining this with the trivial fact that the identity map is in $\Upsilon^{2,1}(\mathbb{R}, \mathbb{R})$ we see that a network can be applied to only a few components of its input.

Lemma 6 *Let $m \geq 0$ and suppose that $f \in \Upsilon^{W,L}(\mathbb{R}^d, \mathbb{R}^k)$. Then the function $f \oplus I$ on $\mathbb{R}^{d+m}$ defined by*

$$(f \oplus I)(x_1 \oplus x_2) = f(x_1) \oplus x_2$$

satisfies $f \oplus I \in \Upsilon^{W+2m,L}(\mathbb{R}^{d+m}, \mathbb{R}^{k+m})$.

Using these basic lemmas we obtain the well-known construction of a deep network which represents the sum of a collection of smaller networks.

Proposition 7 (Summing Networks) *Let $f_i \in \Upsilon^{W,L_i}(\mathbb{R}^d, \mathbb{R}^k)$ for $i = i, ..., n$. Then we have*

$$\sum_{i=1}^{n} f_i \in \Upsilon^{W+2d+2k,L}(\mathbb{R}^d, \mathbb{R}^k),$$

where $L = \sum_{i=1}^{n} L_i$.

For completeness we give a detailed proof in Appendix A. An important application of this is the following well-known result showing how piecewise linear continuous functions can be represented using deep networks.

Proposition 8 *Suppose that $f : \mathbb{R} \rightarrow \mathbb{R}$ is a continuous piecewise linear function with k pieces. Then $f \in \Upsilon^{5,k-1}(\mathbb{R})$.*

For the readers convenience, we give the proof in Appendix A.

Next we describe how to approximate products using deep ReLU networks. This will be necessary in the following to approximate piecewise polynomial functions. The method for doing this is based upon the construction in Telgarsky (2016) and was first applied to approximating smooth functions using neural networks in Yarotsky (2017). This construction has since become an important tool in the analysis of deep ReLU networks and has been used by many different authors (DeVore et al., 2021; Lu et al., 2021; Petersen and Voigtlaender, 2018). For the readers convenience, we reproduce a complete description of the construction in Appendix B.

Proposition 9 (Product Network, Proposition 3 in Yarotsky (2017)) *Let $k \geq 1$. Then there exists a network $f_k \in \Upsilon^{13,6k+3}(\mathbb{R}^2)$ such that for all $x, y \in [-1, 1]$ we have*

$$|f_k(x,y) - xy| \leq 6 \cdot 4^{-k}.$$

The key to obtaining superconvergence for deep ReLU networks is the bit extraction technique, which was first introduced in Bartlett et al. (1998) with the goal of lower bounding the VC dimension of the class of neural networks with polynomial activation function. This technique as also been used to obtain sharp approximation results for deep ReLU networks (Yarotsky, 2018; Shen et al., 2022). In the following Proposition, which is a minor modification of Lemma 11 in Bartlett et al. (2019), we construct the bit extraction networks that we will need in our approximation of Sobolev and Besov functions. For the readers convenience, we give the complete proof in Appendix C.

Proposition 10 (Bit Extraction Network) *Let $n \geq m \geq 0$ be an integer. Then there exists a network $f_{n,m} \in \Upsilon^{9,4m}(\mathbb{R}, \mathbb{R}^2)$ such that for any input $x \in [0, 1]$ with at most n non-zero bits, i.e.*

$$x = 0.x_1x_2 \cdots x_n \tag{2.1}$$

with bits $x_i \in \{0, 1\}$, we have

$$f_{n,m}(x) = \begin{pmatrix} 0.x_{m+1} \cdots x_n \\ x_1x_2 \cdots x_m.0 \end{pmatrix}.$$

Finally, in order to deal with the case when the error is measured in L_∞, we will need the following technical construction. We construct a ReLU network which takes an input in $\mathbb{R}^d$ and returns the k-th largest entry. The first step is the following simple Lemma, whose proof can be found in Appendix A.

Lemma 11 (Max-Min Networks) *There exists a network $p \in \Upsilon^{4,1}(\mathbb{R}^2, \mathbb{R}^2)$ such that*

$$p\left(\begin{pmatrix} x \\ y \end{pmatrix}\right) = \begin{pmatrix} \max(x,y) \\ \min(x,y) \end{pmatrix}.$$

Using these networks as building blocks, we can implement a sorting network using deep ReLU neural networks.

Proposition 12 *Let $k \geq 1$ and $d = 2^k$ be a power of 2. Then there exists a network $s \in \Upsilon^{4d,L}(\mathbb{R}^d, \mathbb{R}^d)$ where $L = \binom{k+1}{2}$ which sorts the input components.*

Note that the power of 2 assumption is for simplicity and is not really necessary. It is also known that the depth $\binom{k+1}{2}$ can be replaced by a multiple Ck where C is a very large constant (Ajtai et al., 1983; Paterson, 1990), but this will not be important in our argument.

Proof Suppose that $(i_1, j_1), ..., (i_{2^{k-1}}, j_{2^{k-1}})$ is a pairing of the indices of $\mathbb{R}^d$. By Lemma 11 and Lemma 5, there exists a network $g \in \Upsilon^{4d,1}(\mathbb{R}^d, \mathbb{R}^d)$ which satisfies for all $l = 1, ..., k-1$

$$g(x)_{i_l} = \max(x_{i_l}, x_{j_l}),\ g(x)_{j_l} = \min(x_{i_l}, x_{j_l}),$$

i.e. which sorts the entries in each pair. By a well-known construction of sorting networks (for instance bitonic sort (Batcher, 1968)), composing $\binom{k+1}{2}$ such functions can be used to sort the input. ■

Finally, we note that by selecting a single output (which is an affine map), we can obtain a network which outputs any order statistic.

Corollary 13 *Let $1 \leq \tau \leq d$ and $d = 2^k$ is a power of 2. Then there exists a network $g_\tau \in \Upsilon^{4d,L}(\mathbb{R}^d)$ with $L = \binom{k+1}{2}$ such that*

$$g_\tau(x) = x_{(\tau)},$$

where $x_{(\tau)}$ is the τ-th largest entry of x.

3. Optimal Representation of Sparse Vectors using Deep ReLU Networks

In this section, we prove the main technical result which enables the efficient approximation of Sobolev and Besov functions in the non-linear regime when $q < p$. Specifically, we have the following Theorem showing how to optimally represent sparse integer vectors using deep ReLU neural networks.

Theorem 14 *Let $M \geq 1$ and $N \geq 1$ and $\mathbf{x} \in \mathbb{Z}^N$ be an N-dimensional vector satisfying*

$$\|\mathbf{x}\|_{\ell^1} \leq M. \tag{3.1}$$

If $N \geq M$, then there exists a neural network $g \in \Upsilon^{17,L}(\mathbb{R}, \mathbb{R})$ with depth

$$L \leq C\sqrt{M(1 + \log(N/M))}$$

which satisfies $g(n) = \mathbf{x}_n$ *for* $n = 1,...,N$.
Further, if $N < M$*, then there exists a neural network* $g \in \Upsilon^{17,L}(\mathbb{R},\mathbb{R})$ *with depth*

$$L \leq C\sqrt{N(1+\log(M/N))}$$

which satisfies $g(n) = \mathbf{x}_n$ *for* $n = 1,...,N$.

Before proving this theorem, we explain the meaning of the result and give some intuition. We let

$$S_{N,M} = \{\mathbf{x} \in \mathbb{Z}^N,\ \|\mathbf{x}\|_{\ell^1} \leq M\} \tag{3.2}$$

denote the set of integer vectors which we wish to encode. We can estimate the cardinality of this set as follows. Using a stars and bars argument we see that

$$|\{\mathbf{x} \in \mathbb{Z}^N_{\geq 0},\ \|\mathbf{x}\|_{\ell^1} \leq M\}| = \binom{N+M}{N} = \binom{N+M}{M}.$$

Further, the signs of each non-zero entry of the above set can be chosen arbitrarily. The number of such choices of sign is equal to the number of non-zero entries and is at most $\min\{M,N\}$. This gives the bound

$$|S_{N,M}| \leq \begin{cases} 2^M\binom{N+M}{M} & N \geq M \\ 2^N\binom{N+M}{N} & N < M. \end{cases}$$

Taking logarithms and utilizing the bound from Lemma 24 (proved later), we estimate

$$\log|S_{N,M}| \leq C\begin{cases} M(1+\log(N/M)) & N \geq M \\ N(1+\log(M/N)) & N < M, \end{cases} \tag{3.3}$$

and this controls the number of bits required to encode the set $S_{N,M}$. Theorem 14 implies that using deep ReLU neural networks, the number of parameters required is the *square root* of the number of bits required for such an encoding. This is analogous to the original application of bit extraction (Bartlett et al., 1998) and underlies the superconvergence phenomenon. Finally, we note that in Theorem 25 from Section 5 we prove that Theorem 14 itself is optimal as long as M is not exponentially small or exponentially large relative to N.

Proof of Theorem 14 Let $M \geq 1$ and $N \geq 1$ be fixed. There are two cases to consider, when $N \geq M$ and when $N < M$. The key to the construction in both cases will be an explicit length k binary encoding of the set $S_{N,M}$ defined in equation (3.2).

By a length k binary encoding we mean a pair of maps:

- $E : S_{N,M} \to \{0,1\}^{\leq k}$ (an encoding map which maps $S_{N,M}$ to a bit-string of length at most k)
- $D : \{0,1\}^{\leq k} \to S_{N,M}$ (a decoding map which recovers $\mathbf{x} \in S_{N,M}$ from a bit-string of length at most k)

which satisfy

$$D(E(\mathbf{x})) = \mathbf{x}.$$

Note that the bound in equation (3.3) implies that there exists such an encoding as long as

$$k \geq C\begin{cases} M(1+\log(N/M)) & N \geq M \\ N(1+\log(M/N)) & N < M. \end{cases}$$

However, in order to construct deep ReLU networks which prove Theorem 14, we will need to construct encoding and decoding maps E and D which are given by an *explicit, simple algorithm*. These will then be used to construct the neural network g.

Let us begin with the first case, when $N \geq M$. In this case, we set $k = 2M(3 + \lceil \log(N/M) \rceil)$ (note that all logarithms are taken with base 2). The encoding map E is defined as

$$E(\mathbf{x}) = f_1 t_1 f_2 t_2 \cdots f_R t_R,$$

the concatenation of $R \leq 2M$ blocks consisting of $f_i \in \{0,1\}^{1+\lceil \log(N/M) \rceil}$ and $t_i \in \{0,1\}^2$. The f_i-bits encode an offset in $\{0, 1, ..., \lceil N/M \rceil\}$ (via binary expansion), and the t_i-bits encode a value in $\{0, \pm 1\}$ (via $0 = 00$, $1 = 10$, and $-1 = 01$). The f_i and t_i are determined from the input $\mathbf{x} \in S_{N,M}$ by Algorithm 1.

It is clear that the number of blocks R produced by Algorithm 1 is at most $2M$ since in each round of the while loop either $f_i = \lceil N/M \rceil$ (which can happen at most M times before the index j reaches the end of the vector) or the entry $\mathbf{r}_j$ is decremented (which can happen at most M times since $\|\mathbf{x}\|_{\ell^1} \leq M$).

Algorithm 1 Small ℓ^1-norm Encoding Algorithm

```
Input: $\mathbf{x} \in \mathbb{Z}^N,\ \|x\|_{\ell^1} \leq M$
Set $j = 0$, $\mathbf{r} = \mathbf{x}$ {Set pointer right before the beginning of the input $\mathbf{x}$ and the residual to $\mathbf{x}$}
Set $i = 1$
while $\mathbf{r} \neq 0$ do
    $l = \min\{i : \mathbf{r}_i \neq 0\}$ {Find the first non-zero index in the residual}
    if $l - j \leq \lceil N/M \rceil$ then {If we can make it to the next non-zero index, do so}
        $f_i = l - j$
        $j = l$
    else {Otherwise go as far as we can}
        $f_i = \lceil N/M \rceil$
        $j = j + \lceil N/M \rceil$
    end if
    if $j = l$ then {If we are at the next non-zero index, $t_i$ captures its sign}
        $t_i = \mathrm{sgn}(\mathbf{r}_j)$
        $\mathbf{r}_j = \mathbf{r}_j - t_i$ {This decrements $\|\mathbf{r}\|_{\ell^1}$ which can happen at most $M$ times}
    else {This can only happen if $f_i = \lceil N/M \rceil$, which can occur at most $M$ times}
        $t_i = 0$
    end if
    $i = i + 1$
end while
```

Next, we consider the case $N < M$. In this case we set $k = 2N(3 + \lceil \log(M/N) \rceil)$, and define the encoding map E via

$$E(\mathbf{x}) = t_1 f_1 t_2 f_2 \cdots t_R f_R,$$

i.e. $E(\mathbf{x})$ is the concatenation of $R \leq 2N$ blocks consisting of $t_i \in \{0,1\}^{2+\lceil \log(M/N) \rceil}$ and $f_i \in \{0,1\}$. The f_i-bits encode an offset in $\{0,1\}$, and the t_i-bits encode a value in $\{-\lceil M/N \rceil, ..., \lceil M/N \rceil\}$. Here the first bit of each t_i determines its sign, while the remaining $1 + \lceil \log(M/N) \rceil$ bits consist of the binary expansion of its magnitude (which lies in $\{0,...,\lceil M/N\rceil\}$). The t_i and f_i are determined from the input $\mathbf{x} \in S_{N,M}$ by Algorithm 2.

It is clear that the number of blocks R produced by Algorithm 2 is at most $2N$ since in each round of the while loop either the entry $\mathbf{r}_j$ is decremented by at least $\lceil M/N\rceil$ (which can happen at most N times since $\|\mathbf{x}\|_{\ell^1} \leq M$), or the entry $\mathbf{r}_j$ is zeroed out (which can happen at most N times before $\mathbf{r} = 0$ since there are only N entries).

Algorithm 2 Large ℓ^1-norm Encoding Algorithm

```
Input: $\mathbf{x} \in \mathbb{Z}^N,\ \|x\|_{\ell^1} \leq M$
Set $j = 0$, $\mathbf{r} = \mathbf{x}$ {Set pointer right before the beginning of the input $\mathbf{x}$ and the residual to $\mathbf{x}$}
Set $i = 1$
while $\mathbf{r} \neq 0$ do
    if $j = 0$ or $\mathbf{r}_j = 0$ then {If the value at the current index is 0, then shift the index}
        $f_i = 1$
        $j = j + 1$
    else
        $f_i = 0$
    end if
    if $|\mathbf{r}_j| \leq \lceil M/N\rceil$ then {If we can fully capture the current value, do so}
        $t_i = \mathbf{r}_j$
        $\mathbf{r}_j = 0$ {This zeros out an entry, which can happen at most $N$ times}
    else {Otherwise capture as much as we can}
        $t_i = \mathrm{sgn}(\mathbf{r}_j)\lceil M/N\rceil$
        $\mathbf{r}_j = \mathbf{r}_j - t_i$ {This reduces $\|\mathbf{r}\|_{\ell^1}$ by at least $\lceil M/N\rceil$ which can happen at most $N$ times}
    end if
    $i = i + 1$
end while
```

In both cases, the decoding map D is given by Algorithm 3. It is easy to verify that Algorithm 3 reconstructs the input $\mathbf{x}$ from the output of either Algorithm 1 or 2.

Algorithm 3 Decoding Algorithm

```
Input: A bit string $f_1 t_1 \cdots f_R t_R$
Set $\mathbf{x} = 0$ and $j = 0$ {Start with the 0 vector}
for $i = 1,...,R$ do
    $j = j + f_i$ {Shift index by $f_i$}
    $\mathbf{x}_j = \mathbf{x}_j + t_i$ {Increment value by $t_i$}
end for
```

We now show how to use these algorithms to construct an appropriate deep ReLU neural network g. Let S be a threshold parameter, to be chosen later.

Given a vector $\mathbf{x} \in \mathbb{Z}^N$, we decompose it into two pieces $\mathbf{x} = \mathbf{x}^B + \mathbf{x}^s$ (here $\mathbf{x}^B$ represents the 'big' part and $\mathbf{x}^s$ the 'small' part). We define

$$\mathbf{x}_i^B = \begin{cases} |\mathbf{x}_i| & \mathbf{x}_i \geq S \\ 0 & \mathbf{x}_i < S \end{cases}$$

and

$$\mathbf{x}_i^s = \begin{cases} 0 & \mathbf{x}_i \geq S \\ |\mathbf{x}_i| & \mathbf{x}_i < S \end{cases}$$

The large part $\mathbf{x}^B$ has small support and so can be efficiently encoded as a piecewise linear function. Specifically, the ℓ^1-norm bound (3.1) on $\mathbf{x}$ implies that the support of $\mathbf{x}^B$ is at most of size

$$|\{n : \mathbf{x}_n^B \neq 0\}| \leq \frac{\|\mathbf{x}^B\|_{\ell^1}}{S} \leq \frac{\|\mathbf{x}\|_{\ell^1}}{S} \leq \frac{M}{S}.$$

This means that there is a piecewise linear function with at most $3M/S$ pieces which matches the values of $\mathbf{x}^B$, so by Proposition 8 there exists a network

$$g_B \in \Upsilon^{5,L}(\mathbb{R})$$

with depth bounded by $L \leq 3M/S$ such that $g_B(n) = \mathbf{x}_n^B$ for $n = 1,...,N$.

The heart of the proof is an efficient encoding of the small part $\mathbf{x}^s$. This requires the encoding and decoding Algorithms 1, 2 and 3. We consider first the case $M \leq N$, which is captured in the following Proposition.

Proposition 15 *Let $M \leq N$ and suppose that $\mathbf{x} \in \mathbb{Z}^N$ and satisfies $\|\mathbf{x}\|_{\ell^l} \leq M$ and $\|\mathbf{x}\|_{\ell^\infty} < S$. Then there exists a $g \in \Upsilon^{15,L}(\mathbb{R})$ such that $g(n) = \mathbf{x}_n$ for $n = 1,...,N$ with*

$$L \leq 8M/S + 8S(5 + \lceil \log(N/M) \rceil) + 4.$$

The proof is quite technical, so let us give a high-level description of the ideas first. The idea is to take the execution of the decoding Algorithm 3 which reconstructs $\mathbf{x}$ and to divide it into blocks of length on the order of S. Each block will start at a point i in the algorithm at which $\mathbf{x}_j = 0$ before step 4 of the loop in 3. During the execution of this block, the index j increases and reaches a larger value at the end of the block. All of the entries $\mathbf{x}_n$ for n between these values is reconstructed during the given block of the reconstruction algorithm.

We now construct three networks. Given an input index n, find the block during which the value $\mathbf{x}_n$ is reconstructed. On the input index n, one network outputs the value of j at the beginning of the block, and another network outputs a real number whose binary expansion contains the bits consumed during this block. Both of these can be implemented using piecewise linear functions whose number of pieces is proportional to the number of blocks. The final network extracts the bits from the output of the second network and implements the execution of algorithm 3 in this block to reconstruct the value of $\mathbf{x}$.

Before giving the detailed proof of Proposition 15, let us complete the proof of Theorem 14 in the case $M \leq N$. We apply Proposition 15 to the small part $\mathbf{x}^s$ to get a network g^s. Then we use Proposition 7 to add this network to the network g^B representing the large part $\mathbf{x}^B$ to get a network $g \in \Upsilon^{17,L}(\mathbb{R})$ with

$$L \leq 11M/S + 8S(5 + \lceil \log(N/M) \rceil) + 4$$

such that $g(n) = \mathbf{x}_n$ for $n = 1,...,N$. Finally, we choose S optimally, namely

$$S = \sqrt{\frac{M}{5 + \lceil \log(N/M) \rceil}},$$

to get $L \leq C\sqrt{M(1+\log(N/M))}$ as desired.

Proof of Proposition 15 Let $f_1t_1\cdots f_Rt_R$ be the output of the encoding Algorithm 1 run on input $\mathbf{x}^s$. Let

$$F := \{i \in \{1,...,R\} : \ f_i = 0\}.$$

We decompose the set F into intervals, i.e.

$$F = \bigcup_{m=1}^{\mathscr{T}} [B_m, U_m],$$

where $[I,J] := \{I, I+1, ..., J\}$ for $I \leq J$ and $B_{m+1} > U_m + 1$.

Note that since $\|\mathbf{x}\|_{\ell^\infty} < S$, the length of these intervals is strictly less than S, i.e. $U_m - B_m + 1 < S$ for $m = 1, ..., \mathscr{T}$. This holds since the encoding Algorithm 1 stays at the same index for all steps $i \in [B_m, U_m]$. Hence this index is decremented $U_m - B_m + 1$ times and so this quantity must be smaller than S.

Let $\rho = \lceil R/S \rceil$ and consider steps $i_0, ..., i_\rho$ defined by $i_\rho = R+1$ (the end of the algorithm) and

$$i_k = \begin{cases} 1+kS & 1+kS \notin F \\ B_m - 1 & 1+kS \in [B_m, U_m], \end{cases}$$

for $k = 0, ..., \rho - 1$. (Note that $f_1 \neq 0$ since the index j starts at 0 in algorithm 1. Thus $1 \notin F$ and so $i_0 = 1$.)

The bound $U_m - B_m + 1 < S$ on the length of the interval $[B_m, U_m]$ implies that $i_k > 1 + (k-1)S$. This implies that $i_{k-1} < i_k$ and also that the gaps satisfy $i_k - i_{k-1} < 2S$ for all $k = 1, ..., \rho$.

Next, let indices j_k for $k = 0, ..., \rho - 1$ be the values of the index j at the beginning of step i_k in the decoding Algorithm 3. We also set $j_\rho = N$. Since by construction the intervals are not consecutive, i.e. $B_{m+1} > U_m + 1$, the steps $i_k \notin F$, i.e. $f_{i_k} > 0$. This means that $j_{k-1} < j_k$ for all $k = 1, ..., \rho$.

Observe that the steps i_k and indices j_k have been constructed such that for an integer n in the interval $j_k < n \leq j_{k+1}$, the value $\mathbf{x}_n$ is only affected during the steps $i_k, ..., i_{k+1} - 1$ in the reconstruction Algorithm 3. Further, the length of each block satisfies $i_k - i_{k-1} < 2S$.

Next, we construct two piecewise linear functions J and R as follows. For integers $n = 1, ..., N$, we set

$$J(n) = j_k \ \text{ for } \ j_k < n \leq j_{k+1},$$

and

$$R(n) = r_k \ \text{ for } \ j_k < n \leq j_{k+1},$$

where

$$r_k = 0.f_{i_k}t_{i_k}\cdots f_{i_{k+1}-1}t_{i_{k+1}-1}$$

is the real number whose binary expansion contains the encoding of $\mathbf{x}$ from step i_k to $i_{k+1} - 1$ (followed by zeros). Both J and R take at most $\rho + 1$ different values and hence can be implemented by piecewise linear functions with at most $2\rho + 1$ pieces. Thus, by Proposition 8 we have $J, R \in \Upsilon^{5,2\rho}(\mathbb{R})$.

We being our network construction as follows. We begin with the affine map

$$x \to \begin{pmatrix} x \\ x \\ x \end{pmatrix} \in \Upsilon^0(\mathbb{R}, \mathbb{R}^3),$$

and use Lemmas 4 and 6 to apply J to the first component and then apply R to the second component to get

$$x \to \begin{pmatrix} J(x) \\ x \\ x \end{pmatrix} \to \begin{pmatrix} J(x) \\ R(x) \\ x \end{pmatrix} \in \Upsilon^{9,4\rho}(\mathbb{R},\mathbb{R}^3).$$

Composing with the affine map

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} z-x \\ y \\ 0 \end{pmatrix} \in \Upsilon^{0}(\mathbb{R}^3,\mathbb{R}^3),$$

and using Lemma 4 again we get that

$$x \to \begin{pmatrix} x-J(x) \\ R(x) \\ 0 \end{pmatrix} \in \Upsilon^{9,4\rho}(\mathbb{R},\mathbb{R}^3). \tag{3.4}$$

Applied to an integer $j_k < n \leq j_{k+1}$, this network maps

$$n \to \begin{pmatrix} n-j_k \\ 0.f_{i_k}t_{i_k}\cdots f_{i_{k+1}-1}t_{i_{k+1}-1} \\ 0 \end{pmatrix}.$$

Thus the first entry is the gap between n and the index j at the beginning of step i_k and the last entry is the value of $\mathbf{x}_n$ at the beginning of step i_k, while the middle entry contains the bits used by the algorithm between steps $i_k,...,i_{k+1}-1$. The proof will now be completed by constructing a network which applies a single step of the decoding Algorithm 3 to each of these entries, this is collected in the following technical Lemma.

Lemma 16 *Given positive integers α and β there exists a network $g \in \Upsilon^{15,4\alpha+16}(\mathbb{R}^3,\mathbb{R}^3)$ such that*

$$g: \begin{pmatrix} x \\ 0.f_1t_1\cdots f_kt_k \\ \Sigma \end{pmatrix} \to \begin{pmatrix} x-f_1 \\ 0.f_2t_2\cdots f_kt_k \\ \Sigma+t_1\delta(x-f_1) \end{pmatrix}$$

whenever $x \in \mathbb{Z}$, $k \leq \beta$ and $\mathrm{len}(f_i) = \alpha$. Here the f_i denote integers encoded via binary expansion and $\mathrm{len}(f_i)$ is the length of this expansion, $t_i \in \{\pm 1, 0\}$ are encoded using two bits (specifically via $0 = 00$, $1 = 10$ and $-1 = 01$), Σ denotes a running sum, and δ is the integer Dirac delta defined by

$$\delta(z) = \begin{cases} 1 & z = 0 \\ 0 & z \neq 0 \end{cases}$$

for integer inputs z.

Before proving this Lemma, let us complete the proof of Proposition 15. We set $\alpha = 1 + \lceil \log(N/M) \rceil$ and $\beta = 2S$, and compose the map in (3.4) with $2S$ copies of the network given by Lemma 16. Then we finally compose with an affine map which selects the last coordinate. This gives a $g \in \Upsilon^{15,L}(\mathbb{R})$ with

$$\begin{aligned} L = 4\rho + 2S(4\alpha+16) &= 4\lceil R/S \rceil + 8S(5 + \lceil \log(N/M) \rceil) \\ &\leq 8M/S + 8S(5 + \lceil \log(N/M) \rceil) + 4, \end{aligned}$$

since $R \leq 2M$.

When applied to an integer $n \in \{1,...,N\}$ with $j_k < n \leq j_{k+1}$, the map in (3.4) sets the offset between n and the index j_k at the start of step i_k, outputs a number whose binary expansion contains the bits used from step i_k to step $i_{k+1}-1$, and sets a running sum to 0.

Then the $2S$ copies of the network from Lemma 16 implement Algorithm 3 from step i_k to step $i_{k+1}-1$. Note that if the number of steps is less than $2S$, the network pads with zero blocks $(f_i, t_i) = 0$, and these additional steps have no effect. Since by construction the entry $\mathbf{x}_n$ is only modified during these steps, the running sum will now be equal to $\mathbf{x}_n$. Finally, we select the last coordinate, which guarantees that $g(n) = \mathbf{x}_n$. ■

Proof of Lemma 16 We construct the desired network as follows. We use Lemma 6 to apply the bit extractor network $f_{n,\alpha}$ from Proposition 10 to the second component. Here we choose $n \geq \beta(\alpha+2)$, which is guaranteed to be larger than the length of the bit-string in the second component. This results in the map

$$\begin{pmatrix} x \\ 0.f_1t_1\cdots f_kt_k \\ \Sigma \end{pmatrix} \to \begin{pmatrix} x \\ f_1 \\ 0.t_1f_2t_2\cdots f_kt_k \\ \Sigma \end{pmatrix} \in \Upsilon^{13,4\alpha}(\mathbb{R}^3,\mathbb{R}^4).$$

Subtracting the second component from the first, this gives

$$\begin{pmatrix} x \\ 0.f_1t_1\cdots f_kt_k \\ \Sigma \end{pmatrix} \to \begin{pmatrix} x - f_1 \\ 0.t_1f_2t_2\cdots f_kt_k \\ \Sigma \end{pmatrix} \in \Upsilon^{13,4\alpha}(\mathbb{R}^3,\mathbb{R}^3), \tag{3.5}$$

and completes the first part of the construction.

Next, we implement a network which extracts the two bits corresponding to t and then adds t to the third component iff the first component is 0. Let $h(z)$ denote the continuous piecewise linear function

$$h(z) = \begin{cases} 0 & z \leq -1 \\ z+1 & -1 < z \leq 0 \\ 1-z & 0 < z \leq 1 \\ 0 & z > 1. \end{cases} \tag{3.6}$$

For integer inputs, h is simply the delta function, i.e. $h(z) = \delta(z)$ for $z \in \mathbb{Z}$, and by Proposition 8 we have $h \in \Upsilon^{5,3}(\mathbb{R})$. We first apply an affine map which duplicates the first coordinate

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} \to \begin{pmatrix} z_1 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} \in \Upsilon^0(\mathbb{R}^3,\mathbb{R}^4).$$

Then, we use Lemma 6 to apply h to the second coordinate and apply the bit extractor network $f_{n,1}$ from Proposition 10 to the third component. As before, we choose $n \geq \beta(\alpha+2)$ which is guaranteed to be larger than the length of the bit-string in the second component. This gives (note that we write $b_1 b_2$ for the two bits corresponding to t_1)

$$\begin{pmatrix} z_1 \\ 0.b_1 b_2 f_2 t_2 ... f_k t_k \\ z_3 \end{pmatrix} \to \begin{pmatrix} z_1 \\ h(z_1) \\ b_1 \\ 0.b_2 f_2 t_2 ... f_k t_k \\ z_3 \end{pmatrix} \in \Upsilon^{15,7}(\mathbb{R}^3, \mathbb{R}^5).$$

Now we compose this using Lemma 4 with the map

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{pmatrix} \to \begin{pmatrix} z_1 \\ z_2 + z_3 - 1 \\ z_4 \\ z_5 \end{pmatrix} \to \begin{pmatrix} z_1 \\ \sigma(z_2 + z_3 - 1) \\ z_4 \\ z_5 \end{pmatrix} \to \begin{pmatrix} z_1 \\ z_4 \\ z_5 + \sigma(z_2 + z_3 - 1) \end{pmatrix} \in \Upsilon^{7,1}(\mathbb{R}^5, \mathbb{R}^3).$$

Here the first and last maps in the composition are affine and the middle map is in $\Upsilon^{7,1}(\mathbb{R}^4, \mathbb{R}^4)$ by Lemma 6. This gives

$$\begin{pmatrix} z_1 \\ 0.b_1 b_2 f_2 t_2 ... f_k t_k \\ z_3 \end{pmatrix} \to \begin{pmatrix} z_1 \\ 0.b_2 f_2 t_2 ... f_k t_k \\ z_3 + \sigma(h(z_1) + b_1 - 1) \end{pmatrix} \in \Upsilon^{15,8}(\mathbb{R}^3, \mathbb{R}^3). \tag{3.7}$$

Notice that $\sigma(h(z_1) + b_1 - 1)$ equals 1 precisely when $z_1 = 0$ and $b_1 = 1$ and equals zero otherwise (for integral z_1).

In an analogous manner, we get

$$\begin{pmatrix} z_1 \\ 0.b_2 f_2 t_2 ... f_k t_k \\ z_3 \end{pmatrix} \to \begin{pmatrix} z_1 \\ 0.f_2 t_2 ... f_k t_k \\ z_3 - \sigma(h(z_1) + b_2 - 1) \end{pmatrix} \in \Upsilon^{15,8}(\mathbb{R}^3, \mathbb{R}^3). \tag{3.8}$$

Composing the networks in (3.7) and (3.8) will extract $t_1 \in \{0, \pm 1\}$ (recall the encoding $0 = 00$, $1 = 10$ and $-1 = 01$) and add t_1 to the last coordinate iff the first coordinate is 0. Composing this with the network in (3.5) gives a network $g \in \Upsilon^{15, 4\alpha+16}(\mathbb{R}^3, \mathbb{R}^3)$ as stated in the Lemma. ■

Next, we consider the case $M > N$, which is somewhat complicated by the fact that the threshold parameter S and the spacing of the blocks are no longer equal in this case. The key construction is contained in the following Proposition.

Proposition 17 *Let $M > N$ and suppose that $\mathbf{x} \in \mathbb{Z}^N$ and satisfies $\|\mathbf{x}\|_{\ell^1} \leq M$ and $\|\mathbf{x}\|_{\ell^\infty} < S$. Then there exists a $g \in \Upsilon^{15,L}(\mathbb{R})$ such that $g(n) = \mathbf{x}_n$ for $n = 1, ..., N$ with*

$$L \leq 8M/S + 8(SN/M + 1)(4 + \lceil \log(M/N) \rceil) + 4.$$

Utilizing this Proposition, we complete the proof of Theorem 14 in the case $M > N$. We apply Proposition 17 to $\mathbf{x}^s$ and use Proposition 7 to add the network to the network representing $\mathbf{x}^B$ to get a network $g \in \Upsilon^{17,L}(\mathbb{R})$ representing $\mathbf{x}$ with

$$L \leq 11M/S + 8(SN/M + 1)(4 + \lceil \log(M/N) \rceil) + 4.$$

Finally, we optimize in S, resulting in a value

$$S = \frac{M\sqrt{4 + \lceil \log(M/N) \rceil}}{\sqrt{N}}$$

to get $L \leq C\sqrt{N(1 + \log(M/N)}$ as desired.

Proof of Proposition 17 The proof proceeds in a very similar manner to the proof of Proposition 15 and we only indicate the differences here.

We begin with the same set F and its decomposition into intervals $[B_m, U_m]$, except that $f_1t_1 \cdots f_Rt_R$ is now the output of the encoding Algorithm 2.

Our bound on the block length becomes $U_m - B_m + 1 \leq SN/M$. This holds since the encoding algorithm stays at the same index for all steps $i \in [B_m, U_m]$, and thus this index in decremented by an amount $\lceil M/N \rceil$ a total of $U_m - B_m + 1$ times. The bound on the ℓ_∞-norm implies that $(M/N)(U_m - B_m + 1) < S$, which gives the desired bound.

Thus, in this case we set $T = \lceil SN/M \rceil$ and $\rho = \lceil R/T \rceil$ and consider steps $i_0, ..., i_\rho$ defined by $i_\rho = R + 1$ (the end of the algorithm) and

$$i_k = \begin{cases} 1 + kT & 1 + kT \notin F \\ B_m - 1 & 1 + kT \in [B_m, U_m], \end{cases}$$

for $k = 0, ..., \rho - 1$.

We now proceed with the same argument as in Proposition 15, except that the bound on $U_m - B_m + 1 < T$ implies that all block lengths are bounded by $2T$. The proof is finally completed with the following variant of Lemma 16, which implements a step of the decoding Algorithm 3 with the values f_i and t_i encoded as they are for $M > N$.

Lemma 18 *Given positive integers α and β there exists a network $g \in \Upsilon^{15,4\alpha+8}(\mathbb{R}^3, \mathbb{R}^3)$ such that*

$$g : \begin{pmatrix} x \\ 0.f_1t_1 \cdots f_kt_k \\ \Sigma \end{pmatrix} \to \begin{pmatrix} x - f_1 \\ 0.f_2t_2 \cdots f_kt_k \\ \Sigma + t_1\delta(x - f_1) \end{pmatrix}$$

whenever $x \in \mathbb{Z}$, $k \leq \beta$ and $\operatorname{len}(t_i) = \alpha$. Here $f_i \in \{0, 1\}$ are single bits, $t_i \in \mathbb{Z}$ is encoded via binary expansion with a single bit giving its sign and $\operatorname{len}(t_i)$ is the length of this expansion, Σ denotes a running sum, and δ is the integer Dirac delta defined by

$$\delta(z) = \begin{cases} 1 & z = 0 \\ 0 & z \neq 0 \end{cases}$$

for integer inputs z.

Given this lemma, we complete the proof as before, setting $\alpha = 2+\lceil\log(M/N)\rceil$ and $\beta = 2T$ and composing the network implementing the maps J and R with $2T$ copies of the network from Lemma 18. This gives a network $g \in \Upsilon^{15,L}(\mathbb{R})$ with

$$\begin{aligned} L \leq 4\rho + 2T(4\alpha+8) &= 4\lceil R/T\rceil + 8T(4+\lceil\log(M/N)\rceil) \\ &\leq 8N/T + 8T(4+\lceil\log(M/N)\rceil)+4 \\ &\leq 8M/S + 8(SN/M+1)(4+\lceil\log(M/N)\rceil)+4, \end{aligned}$$

since $R \leq 2N$ and $T = \lceil SN/M\rceil$. ■

Proof of Lemma 18 We use Lemma 6 to apply the bit extractor network $f_{n,1}$ from Proposition 10 to the second component. Here we choose $n \geq \beta(\alpha+1)$, which is guaranteed to be larger than the length of the bit-string in the second component. Then we subtract the second component from the first. This results in the map

$$\begin{pmatrix} x \\ 0.f_1t_1\cdots f_kt_k \\ \Sigma \end{pmatrix} \to \begin{pmatrix} x-f_1 \\ 0.t_1f_2t_2\cdots f_kt_k \\ \Sigma \end{pmatrix} \in \Upsilon^{13,4}(\mathbb{R}^3,\mathbb{R}^3),$$

and completes the first part of the construction.

Now we wish to extract the integer t_1 and add it to Σ iff the first coordinate (which is an integer) is 0. We do this by using Lemma 6 to apply the bit extractor network $f_{n,1}$ to the second coordinate and then apply $f_{n,\alpha-1}$ to the third coordinate of the result to get

$$\begin{pmatrix} z_1 \\ 0.b_1b_2...b_\alpha f_2t_2...f_kt_k \\ z_3 \end{pmatrix} \to \begin{pmatrix} z_1 \\ b_1 \\ b_2...b_\alpha \\ 0.f_2t_2...f_kt_k \\ z_3 \end{pmatrix} \in \Upsilon^{15,4\alpha}(\mathbb{R}^3,\mathbb{R}^5), \tag{3.9}$$

where we have written $b_1b_2...b_\alpha$ for the bits of t_1.

Next, consider the following sequence of compositions, where h is the function defined in (3.6),

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{pmatrix} \to \begin{pmatrix} z_1 \\ z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{pmatrix} \to \begin{pmatrix} z_1 \\ h(z_1) \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{pmatrix} \to \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_3 - 2^\alpha(1-h(z_1)+z_2) \\ z_4 \\ z_5 \end{pmatrix} \to$$

$$\to \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ \sigma(z_3 - 2^\alpha(1-h(z_1)+z_2)) \\ z_4 \\ z_5 \end{pmatrix} \to \tag{3.10}$$

$$\to \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 + \sigma(z_3 - 2^\alpha(1-h(z_1)+z_2)) \end{pmatrix} \in \Upsilon^{15,4}(\mathbb{R}^5, \mathbb{R}^5).$$

Using a sequence of applications of Lemmas 6 and 4, we obtain that this map can be implemented by a network in $\Upsilon^{15,4}(\mathbb{R}^5, \mathbb{R}^5)$.

Note that when $z_1 \in \mathbb{Z}$ and $z_2 \in \{0, 1\}$, we have that (recall that $h(z) = \delta(z)$ for integer z)

$$(1-h(z_1)+z_2) = \begin{cases} 0 & z_1 = 0 \text{ and } z_2 = 0 \\ 1 & z_1 \neq 0 \text{ and } z_2 = 0 \\ 1 & z_1 = 0 \text{ and } z_2 = 1 \\ 2 & z_1 \neq 0 \text{ and } z_2 = 1. \end{cases}$$

If we also have that $z_3 \in \{0, ..., 2^\alpha\}$, then it follows that

$$\sigma(z_3 - 2^\alpha(1-h(z_1)+z_2)) = \begin{cases} z_3 & z_1 = 0 \text{ and } z_2 = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Thus, if we compose the network in (3.9) with the network in (3.10), we will add the number $b_2...b_\alpha$ (which is less than 2^α) to the last coordinate iff $z_1 = b_1 = 0$. As $b_1 = 0$ to indicate that t_1 is positive and $b_2...b_\alpha$ contain the value of t_1, his handles the case where t_1 is positive and has no effect when t_1 is negative.

Next, we construct a network which handles the negative part of t_1. This is given by the following composition

$$\begin{pmatrix} z_1\\ z_2\\ z_3\\ z_4\\ z_5 \end{pmatrix} \to \begin{pmatrix} z_1\\ z_1\\ z_2\\ z_3\\ z_4\\ z_5 \end{pmatrix} \to \begin{pmatrix} z_1\\ h(z_1)\\ z_2\\ z_3\\ z_4\\ z_5 \end{pmatrix} \to \begin{pmatrix} z_1\\ z_2\\ z_3\\ z_3-2^{\alpha}(2-h(z_1)-z_2)\\ z_4\\ z_5 \end{pmatrix} \to$$
$$\to \begin{pmatrix} z_1\\ z_2\\ z_3\\ \sigma(z_3-2^{\alpha}(2-h(z_1)-z_2))\\ z_4\\ z_5 \end{pmatrix} \to \tag{3.11}$$
$$\to \begin{pmatrix} z_1\\ z_2\\ z_3\\ z_4\\ z_5-\sigma(z_3-2^{\alpha}(2-h(z_1)-z_2)) \end{pmatrix} \in \Upsilon^{15,4}(\mathbb{R}^5,\mathbb{R}^5).$$

When $z_1 \in \mathbb{Z}$ and $z_2 \in \{0,1\}$, we have that

$$(2-h(z_1)-z_2) = \begin{cases} 1 & z_1 = 0 \text{ and } z_2 = 0\\ 2 & z_1 \neq 0 \text{ and } z_2 = 0\\ 0 & z_1 = 0 \text{ and } z_2 = 1\\ 1 & z_1 \neq 0 \text{ and } z_2 = 1. \end{cases}$$

So, if we compose the network in (3.9) with the network in (3.11), we will subtract the number $b_2...b_\alpha$ (which is less than 2^α) from the last coordinate iff $z_1 = 0$ and $b_1 = 1$. As $b_1 = 1$ to indicate that t_1 is negative and $b_2...b_\alpha$ contain the value of t_1, this handles the case where t_1 is negative and has no effect when t_1 is positive.

We obtain the final network g by successively composing the network in (3.9) with the networks in (3.10) and (3.11) and then dropping the second and third components. ■

■

4. Optimal Approximation of Sobolev Functions Using Deep ReLU Networks

In this section, we give the main construction and the proof of Theorems 1 and 2. A key component of the proof is the approximation of piecewise polynomial functions using deep ReLU neural networks. To describe this, we first introduce some notation.

Throughout this section, unless otherwise specified, let $b \geq 2$ be a fixed integer. To avoid excessively cumbersome notation, we suppress the dependence on b in the following notation. Let

$l \geq 0$ be an integer and consider the b-adic decomposition of the cube $\Omega = [0,1)^d$ (note that by removing a zero-measure set it suffices to consider this half-open cube in the proof) at level l given by

$$\Omega = \bigcup_{\mathbf{i} \in I_l} \Omega^l_{\mathbf{i}}, \tag{4.1}$$

where the index $\mathbf{i}$ lies in the index set $I_l := \{0,...,b^l - 1\}^d$, and $\Omega^l_{\mathbf{i}}$ is defined by

$$\Omega^l_{\mathbf{i}} = \prod_{j=1}^d [b^{-l}\mathbf{i}_j, b^{-l}(\mathbf{i}_j + 1)). \tag{4.2}$$

Note that for each l, the b^{dl} subcubes $\Omega^l_{\mathbf{i}}$ form a partition of the original cube Ω. For an integer $k \geq 0$, we let P_k denote the space of polynomials of degree at most k and consider the space

$$\mathscr{P}^l_k = \left\{ f : \Omega \rightarrow \mathbb{R},\ f_{\Omega^l_{\mathbf{i}}} \in P_k \text{ for all } \mathbf{i} \in I_l \right\}$$

of (non-conforming) piecewise polynomials subordinate to the partition (4.1). The space $\mathscr{P}^l_k$ has dimension $\binom{d+k}{k} b^{dl}$ and a natural (L_∞-normalized) basis

$$\rho^\alpha_{l,\mathbf{i}}(x) = \begin{cases} \prod_{j=1}^d (b^l x_j - \mathbf{i}_j)^{\alpha_j} & x \in \Omega^l_{\mathbf{i}} \\ 0 & x \notin \Omega^l_{\mathbf{i}} \end{cases}$$

indexed by $\mathbf{i} \in I_l$ and α a d-dimensional multi-index with $|\alpha| \leq k$.

In our construction, we will approximate piecewise polynomial functions from $\mathscr{P}^l_k$ by deep ReLU neural networks. However, since a deep ReLU network can only represent a piecewise continuous function, this approximation will not be over the full cube Ω. Rather, we will need to remove an arbitrarily small region from Ω. This idea is from the method in Shen et al. (2022), where this region was called the trifling region. Given $\varepsilon > 0$ we define sets

$$\Omega^l_{\mathbf{i},\varepsilon} = \prod_{j=1}^d \begin{cases} [b^{-l}\mathbf{i}_j, b^{-l}(\mathbf{i}_j + 1) - \varepsilon) & \mathbf{i}_j < b^l - 1 \\ [b^{-l}\mathbf{i}_j, b^{-l}(\mathbf{i}_j + 1)) & \mathbf{i}_j = b^l - 1, \end{cases} \tag{4.3}$$

which are slightly shrunk sub-cubes (except at one edge) from (4.2). We then define the good region to be

$$\Omega_{l,\varepsilon} := \bigcup_{\mathbf{i} \in I_l} \Omega^l_{\mathbf{i},\varepsilon}.$$

Next, we will show how to approximate piecewise polynomials from $\mathscr{P}^l_k$ on the set $\Omega_{l,\varepsilon}$. For this, we begin with the following Lemma, which first appears in Shen et al. (2022). This Lemma is essentially a minor modification of the bit-extraction technique used to prove Proposition 10. We give a detailed proof for the reader's convenience in Appendix C.

Lemma 19 *Let $l \geq 0$ be an integer and $0 < \varepsilon < b^{-l}$. Then there exists a deep ReLU neural network $q_d \in \Upsilon^{9d,2(b-1)l}(\mathbb{R}^d)$ such that*

$$q_d(\Omega^l_{\mathbf{i},\varepsilon}) = \mathrm{ind}(\mathbf{i}) := \sum_{j=1}^d b^{l(j-1)}\mathbf{i}_j.$$

Note that here $\mathrm{ind}(\mathbf{i}) \in \{0,...,b^{dl}-1\}$ is just an integer index corresponding to the sub-cube position $\mathbf{i}$.

Using this Lemma we prove the following key technical Proposition, which shows how to efficiently approximate piecewise polynomial functions on the good set $\Omega_{l,\varepsilon}$.

Proposition 20 *Let $l \geq 0$ be an integer and $\varepsilon > 0$. Suppose that $f \in \mathscr{P}_k^l$ is expanded in terms of the bases $\rho_{l,\mathbf{i}}^\alpha$,*

$$f(x) = \sum_{|\alpha|\leq k,\ \mathbf{i}\in I_l} a_{\mathbf{i}}^\alpha \rho_{l,\mathbf{i}}^\alpha(x).$$

Let $1 \leq q \leq p \leq \infty$ and choose a parameter $\delta > 0$ and an integer $m \geq 1$. Then there exists a deep ReLU network $f_{\delta,m} \in \Upsilon^{22d+18,L}(\mathbb{R}^d)$ such that

$$\|f - f_{\delta,m}\|_{L_p(\Omega_{l,\varepsilon})} \leq C\left(\delta \min\left\{1, b^{-dl}\delta^{-q}\right\}^{1/p} + 4^{-m}\right)\left(\sum_{|\alpha|\leq k,\ \mathbf{i}\in I_l} |a_{\mathbf{i}}^\alpha|^q\right)^{1/q}$$

(with the standard modification when $q = \infty$), and whose depth satisfies

$$L \leq C\begin{cases} m + l + \delta^{-q/2}\sqrt{1 + dl\log(b) + q\log(\delta)} & \delta^{-q} \leq b^{dl} \\ m + l + b^{dl/2}\sqrt{1 - \log\delta - (dl/q)\log(b)} & \delta^{-q} > b^{dl}. \end{cases}$$

Here the constants $C := C(p,q,d,k,b)$ only depend upon p,q,d,k and the base b, but not on f, δ, l, ε, or m.

Before we prove this Proposition, let us explain the intuition behind it and the meaning of the parameter δ. The parameter δ represents a discretization level for the coefficients $a_{\mathbf{i}}^\alpha$. Specifically, we will round each coefficient down (in absolute value) to the nearest multiple of δ to produce an approximation to f. Then, we will represent this approximation by encoding these discretized coefficients using deep ReLU networks. This reduces to encoding an integer vector which can be done optimally using Theorem 14. The two regimes $\delta^{-q} \leq b^{dl}$ and $\delta^{-q} > b^{dl}$ correspond to the case of *dense* and *sparse* coefficients, which are handled differently in Theorem 14.

Proof of Proposition 20 We begin by decomposing $f = \sum_{|\alpha|\leq k} f_\alpha$ where

$$f_\alpha(x) = \sum_{\mathbf{i}\in I_d} a_{\mathbf{i}}^\alpha \rho_{l,\mathbf{i}}^\alpha(x).$$

By Proposition 7 and the triangle inequality, it suffices to prove the result for each f_α individually with width $W = 20d + 17$ (at the expense of larger constants). So in the following we assume that $f = f_\alpha$ and write $a_{\mathbf{i}} := a_{\mathbf{i}}^\alpha$. By normalizing f we may assume also without loss of generality that

$$\left(\sum_{\mathbf{i}\in I_l} |a_{\mathbf{i}}|^q\right)^{1/q} \leq 1. \tag{4.4}$$

We construct the following network. First, duplicate the input $x \in \mathbb{R}^d$ three times using an affine map

$$x \to \begin{pmatrix} x \\ x \\ x \end{pmatrix} \in \Upsilon^0(\mathbb{R}^d, \mathbb{R}^{3d}).$$

Next, we use Lemmas 6 and 4 to apply the network q_d from Lemma 19 to the last coordinate and apply q_1 from Lemma 19 to each entry of the first coordinate to get

$$x \to \begin{pmatrix} q_1(x_1) \\ \vdots \\ q_1(x_d) \\ x \\ q_d(x) \end{pmatrix} \in \Upsilon^{20d,2(b-1)l}(\mathbb{R}^d, \mathbb{R}^{2d+1}).$$

We now compose with the affine map

$$\begin{pmatrix} x \\ y \\ r \end{pmatrix} \to \begin{pmatrix} b^l y - x \\ r \end{pmatrix} \in \Upsilon^0(\mathbb{R}^{2d+1}, \mathbb{R}^{d+1}),$$

where $x, y \in \mathbb{R}^d$ and $r \in \mathbb{R}$, to get

$$x \to \begin{pmatrix} b^l x_1 - q(x_1) \\ \vdots \\ b^l x_d - q(x_d) \\ q_d(x) \end{pmatrix} \in \Upsilon^{20d,2(b-1)l}(\mathbb{R}^d, \mathbb{R}^{d+1}). \tag{4.5}$$

On the set $\Omega^l_{\mathbf{i},\varepsilon}$ from (4.3) this map becomes

$$x \to \begin{pmatrix} b^l x_1 - \mathbf{i}_1 \\ \vdots \\ b^l x_d - \mathbf{i}_d \\ \mathrm{ind}(\mathbf{i}) \end{pmatrix}.$$

The next step in the construction will be to approximate the coefficients $a_{\mathbf{i}}$. To do this we round the $a_{\mathbf{i}}$ down to the nearest multiple of δ (in absolute value) to get approximate coefficients

$$\tilde{a}_{\mathbf{i}} := \delta\, \mathrm{sgn}(a_{\mathbf{i}}) \left\lfloor \frac{|a_{\mathbf{i}}|}{\delta} \right\rfloor.$$

We estimate the ℓ^p-norm of the error this incurs as follows. Write

$$\|a - \tilde{a}\|_{\ell^p} = \left(\sum_{\mathbf{i} \in I_l} |a_{\mathbf{i}} - \tilde{a}_{\mathbf{i}}|^p \right)^{1/p}$$

with the standard modification when $p = \infty$. Note that

$$\|a - \tilde{a}\|_{\ell^q} \le \|a\|_{\ell^q} \le 1$$

by (4.4). In addition, it is clear from the rounding procedure that $\|a - \tilde{a}\|_{\ell^\infty} \le \delta$. Hölder's inequality thus implies that (since $p \ge q$)

$$\|a - \tilde{a}\|_{\ell^p} \le \|a - \tilde{a}\|_{\ell^q}^{q/p} \|a - \tilde{a}\|_{\ell^\infty}^{1-q/p} \le \delta^{1-q/p}.$$

On the other hand, using that $|I_l| = b^{dl}$, we can use the bound $\|a-\tilde{a}\|_{\ell^\infty} \leq \delta$ to get

$$\|a-\tilde{a}\|_{\ell^p} \leq b^{dl/p}\delta.$$

Putting these together, we get

$$\|a-\tilde{a}\|_{\ell^p} \leq \delta \min\{b^{dl}, \delta^{-q}\}^{1/p}. \tag{4.6}$$

Next we construct a ReLU neural network which maps the index $\mathrm{ind}(\mathbf{i})$ to the rounded coefficients $\tilde{a}_{\mathbf{i}}$. For this Theorem 14 will be key. We set $N = b^{dl}$ and write $\tilde{a}_{\mathbf{i}} = \delta \mathbf{x}_{\mathrm{ind}(\mathbf{i})}$ for a vector $\mathbf{x} \in \mathbb{Z}^N$ defined by

$$\mathbf{x}_{\mathrm{ind}(\mathbf{i})} = \mathrm{sgn}(a_{\mathbf{i}}) \left\lfloor \frac{|a_{\mathbf{i}}|}{\delta} \right\rfloor.$$

We proceed to estimate $\|\mathbf{x}\|_{\ell^1}$. We observe that by (4.4)

$$\sum_{i=1}^N |\mathbf{x}_i|^q \leq \sum_{\mathbf{i}\in I_l} \left(\frac{|a_{\mathbf{i}}|}{\delta}\right)^q \leq \delta^{-q}. \tag{4.7}$$

Thus $\|\mathbf{x}\|_{\ell^q} \leq \delta^{-1}$. Moreover, since $\mathbf{x} \in \mathbb{Z}^N$, (4.7) implies that the number of non-zero entries in $\mathbf{x}$ satisfies

$$|\{i : \mathbf{x}_i \neq 0\}| \leq \min\{\delta^{-q}, N\}.$$

We can thus use Hölder's inequality to get the bound

$$\|\mathbf{x}\|_{\ell^1} \leq |\{i : \mathbf{x}_i \neq 0\}|^{1-1/q} \|\mathbf{x}\|_{\ell^q} \leq \delta^{-1} \min\{\delta^{-q}, N\}^{1-1/q}.$$

Using this we apply Theorem 14 with $M = \delta^{-1}\min\{\delta^{-q}, N\}^{1-1/q}$ to the vector $\mathbf{x}$. We calculate that if $\delta^{-q} \leq N$, then

$$M = \delta^{-1}\delta^{-q(1-1/q)} = \delta^{-q} \leq N,$$

while if $\delta^q > N$, then

$$M = \delta^{-1}N^{(1-1/q)} = N(\delta^{-q}N^{-1})^{1/q} \geq N.$$

Thus, Theorem 14 (combined with a scaling by δ) gives a network $g \in \Upsilon^{17,L}(\mathbb{R})$ such that $g(\mathrm{ind}(\mathbf{i})) = \tilde{a}_{\mathbf{i}}$, whose depth is bounded by

$$L \leq C \begin{cases} \delta^{-q/2}\sqrt{1 + dl\log(b) + q\log(\delta)} & \delta^{-q} \leq b^{dl} \\ b^{dl/2}\sqrt{1 - \log\delta - (dl/q)\log(b)} & \delta^{-q} > b^{dl}. \end{cases}$$

Using Lemma 6 to apply g to the last coordinate of the output in (4.5) gives a network $\tilde{f}_\delta \in \Upsilon^{20d+17,L}$ with depth bounded by

$$L \leq 2(b-1)l + C \begin{cases} \delta^{-q/2}\sqrt{1 + dl\log(b) + q\log(\delta)} & \delta^{-q} \leq b^{dl} \\ b^{dl/2}\sqrt{1 - \log\delta - (dl/q)\log(b)} & \delta^{-q} > b^{dl}, \end{cases}$$

such that for $x \in \Omega^l_{\mathbf{i},\varepsilon}$ we have

$$\tilde{f}_\delta(x) = \begin{pmatrix} b^l x_1 - \mathbf{i}_1 \\ \vdots \\ b^l x_d - \mathbf{i}_d \\ \tilde{a}_{\mathbf{i}} \end{pmatrix}. \tag{4.8}$$

Finally, to obtain the network $f_{\delta,m}$ we use Lemma 4 to compose $\tilde{f}_\delta$ with a network P_m which approximates the product

$$\begin{pmatrix} z_1 \\ \vdots \\ z_d \\ z_{d+1} \end{pmatrix} \rightarrow z_{d+1}\prod_{j=1}^d z_j^{\alpha_j}$$

on the set where $|z_j| \leq 1$ for all $j = 1,...,d+1$. Note from the bound (4.4) we see that $|\tilde{a}_{\mathbf{i}}| \leq |a_{\mathbf{i}}| \leq \|a\|_{\ell^q} \leq 1$. In addition, it is easy to see that for $x \in \Omega^l_{\mathbf{i},\varepsilon}$ we have $|b^l x_j - \mathbf{i}_j| \leq 1$ for $j = 1,...,d$. Thus the output of $\tilde{f}_\delta$ satisfies these assumptions for any $x \in \Omega_{l,\varepsilon}$.

We construct the network P_m using Proposition 9 as follows. Choose a parameter $m \geq 1$. We first approximate a function which multiplies the last entry z_{d+1} by the i-th entry z_i. We do this by duplicating the i-th entry using an affine map and then applying Lemma 6 to apply the network f_m from Proposition 9 to the i-th and last entries

$$\begin{pmatrix} z_1 \\ \vdots \\ z_d \\ z_{d+1} \end{pmatrix} \rightarrow \begin{pmatrix} z_1 \\ \vdots \\ z_d \\ z_{d+1} \\ z_i \end{pmatrix} \rightarrow \begin{pmatrix} z_1 \\ \vdots \\ f_m(z_{d+1}, z_i) \end{pmatrix} \in \Upsilon^{2d+13,6m+3}(\mathbb{R}^{d+1}, \mathbb{R}^{d+1}).$$

In order to ensure that the resulting approximate product is still bounded in magnitude by 1 (so that we can recursively apply these products), we apply the map $z \rightarrow \max(\min(z,-1),1) \in \Upsilon^{5,2}(\mathbb{R})$ to the last component. This gives a network $P_i^m \in \Upsilon^{2d+13,6m+5}(\mathbb{R}^{d+1}, \mathbb{R}^{d+1})$, which maps

$$P_i^m : \begin{pmatrix} z_1 \\ \vdots \\ z_d \\ z_{d+1} \end{pmatrix} \rightarrow \begin{pmatrix} z_1 \\ \vdots \\ z_d \\ z_{d+1} \\ z_i \end{pmatrix} \rightarrow \begin{pmatrix} z_1 \\ \vdots \\ \tilde{f}_m(z_{d+1}, z_i) \end{pmatrix},$$

where $\tilde{f}_m(z_{d+1}, z_i) = \max(\min(f_m(z_{d+1}, z_i), -1), 1)$. Observe that since the true product $z_{d+1}z_i \in [-1,1]$ the truncation cannot increase the error, so that Proposition 9 implies

$$|\tilde{f}_m(z_{d+1}, z_i) - z_i z_{d+1}| \leq |f_m(z_{d+1}, z_i) - z_i z_{d+1}| \leq 6 \cdot 4^{-m}.$$

We construct P_m by composing (using Lemma 4) α_j copies of P_j^m and then applying an affine map which selects the last coordinate. Thus $P_m \in \Upsilon^{2d+13,L}(\mathbb{R}^{d+1})$ with $L \leq k(6m+5)$. Moreover, since all entries z_i are bounded by 1, we calculate that

$$\left| P_m(\mathbf{z}) - z_{d+1}\prod_{j=1}^d z_j^{\alpha_j} \right| \leq \sum_{j=1}^d \alpha_j |\tilde{f}_m(z_{d+1}, z_j) - z_j z_{d+1}| \leq 6k \cdot 4^{-m}. \tag{4.9}$$

We obtain the network $f_{\delta,m} \in \Upsilon^{20d+17,L}(\mathbb{R}^d, \mathbb{R})$ by composing $\tilde{f}_\delta$ and P_m using Lemma 4. Its depth is bounded by

$$L \leq 2(b-1)l + k(6m+5) + C \begin{cases} \delta^{-q/2}\sqrt{1 + dl\log(b) + q\log(\delta)} & \delta^{-q} \leq b^{dl} \\ b^{dl/2}\sqrt{1 - \log\delta - (dl/q)\log(b)} & \delta^{-q} > b^{dl}, \end{cases}$$

and we note that $k(6m+5) \leq Cm$ for integers $m \geq 1$ and a constant $C := C(k)$ which depends upon k.

We bound the error using equations (4.6), (4.8), (4.9), and the fact that the basis $\rho^{\alpha}_{l,\mathbf{i}}$ is normalized in L_∞ and has disjoint support for fixed α to get

$$\|f - f_{\delta,m}\|^p_{L_p(\Omega_{l,\varepsilon})} \leq 2^{-ld} \sum_{\mathbf{i} \in I_l} |a_{\mathbf{i}} - \tilde{a}_{\mathbf{i}}|^p + (6k \cdot 4^{-m})^p,$$

so that

$$\|f - f_{\delta,m}\|_{L_p(\Omega_{l,\varepsilon})} \leq 2^{-ld/p} \|a - \tilde{a}\|_{\ell^p} + 6k \cdot 4^{-m} \leq C\left(\delta \min\left\{1, 2^{-dl}\delta^{-q}\right\}^{1/p} + 4^{-m}\right),$$

which completes the proof. ■

Next, we use the construction in Proposition 20 to approximate a target function $f \in W^s(L_q(\Omega))$ in $L_p(\Omega)$ using deep ReLU neural networks, again removing an arbitrarily small trifling set in the spirit of Shen et al. (2022).

Proposition 21 *Let $\Omega = [0,1)^d$, $1 \leq q \leq p \leq \infty$ and $f \in W^s(L_q(\Omega))$ with $\|f\|_{W^s(L_q(\Omega))} \leq 1$ for $s > 0$. Suppose that the Sobolev embedding condition is strictly satisfied, i.e.*

$$\frac{1}{q} - \frac{1}{p} - \frac{s}{d} < 0, \tag{4.10}$$

which guarantees the compact embedding $W^s(L_q(\Omega)) \subset\subset L_p(\Omega)$ holds. Let $\varepsilon > 0$ and $l_0 \geq 1$ be an integer and set $l^* = \lfloor \kappa l_0 \rfloor$ with

$$\kappa := \frac{s}{s + d/p - d/q}.$$

Note that $1 \leq \kappa < \infty$ by the Sobolev embedding condition. Then there exists a network $f_{l_0,\varepsilon} \in \Upsilon^{24d+20,L}(\mathbb{R}^d)$ such that

$$\|f - f_{l_0,\varepsilon}\|_{L_p(\Omega_{l^*,\varepsilon})} \leq Cb^{-sl_0}$$

and whose depth is bounded by

$$L \leq Cb^{dl_0/2}.$$

Here the constants $C := C(s,p,q,d,b)$ do not depend upon l_0, f or ε.

Before giving the detailed proof, let us comment on the intuition and the meaning of κ and l^*. The idea is to decompose the function f into different scales which consist of piecewise polynomial functions. We then appoximate these piecewise polynomial functions using neural networks via Proposition 20 to varying degrees of accuracy dependent on the parameter δ used at each level. The parameter l_0 gives the finest level at which we approximate the coefficients in the dense regime $\delta^{-q} > b^{dl}$, while the level l^* is the finest level which appears in the approximation. All levels between l_0 and l^* are approximated in the sparse regime $\delta^{-q} \leq b^{dl}$. The parameter κ controls the gap between l_0 and l^*, and essentially measures how adaptive the approximation must be. The proof is completed by choosing δ optimally at each level, analogous to the proof of the Birman-Solomyak Theorem (Birman and Solomyak, 1967) which calculates the metric entropy of the Sobolev unit ball.

Proof of Proposition 21 For a function $f \in L_q(\Omega)$, we write

$$\Pi^l_k(f) = \arg\min_{p \in \mathscr{P}^l_k} \|f - p\|_{L^q(\Omega)}$$

for the L_q-projection of f onto the space of piecewise polynomials of degree k. We will utilize the following well-known multiscale dyadic decomposition of the function f, which is a common tool in harmonic analysis (Birman and Solomyak, 1967; Mallat, 1999; Littlewood and Paley, 1931) and the analysis of multigrid methods (Bramble et al., 1990),

$$f = \sum_{l=0}^{\infty} f_l,$$

where the components at level l are defined by $f_0 = \Pi_k^0(f)$ and $f_l = \Pi_k^l(f) - \Pi_k^{l-1}(f)$ for $l \geq 1$. Expanding the components f_l in the basis $\rho_{l,\mathbf{i}}^{\alpha}$, we write

$$f_l(x) = \sum_{|\alpha| \leq k,\ \mathbf{i} \in I_l} a_{l,\mathbf{i}}^{\alpha} \rho_{l,\mathbf{i}}^{\alpha}(x). \tag{4.11}$$

The key estimate in the proof is to establish the following coefficient bound

$$|a_{l,\mathbf{i}}^{\alpha}| \leq C b^{(d/q-s)l} |f|_{W^s(L_q(\Omega_{\mathbf{i}^-}^{l-1}))}, \tag{4.12}$$

where $\Omega_{\mathbf{i}^-}^{l-1} \supset \Omega_{\mathbf{i}}^{l}$ is the parent domain of $\Omega_{\mathbf{i}}^{l}$ when $l \geq 1$. When $l = 0$, we have the simple modification

$$|a_{0,\mathbf{0}}^{\alpha}| \leq C \|f\|_{W^s(L_q(\Omega))}.$$

We prove (4.12) by utilizing the Bramble-Hilbert lemma (Bramble and Hilbert, 1970) and a well-known scaling argument. For $l \geq 1$ consider the scaling map $S_{l,\mathbf{i}}$ which scales the small domain $\Omega_{\mathbf{i}^-}^{l-1}$ up to the large domain Ω, defined by

$$S_{l,\mathbf{i}}(f)(x) = f(b^{l-1}x - \mathbf{i}^-) \in L_q(\Omega)$$

for $f \in L_q(\Omega_{\mathbf{i}^-}^{l-1})$. We verify the following simple facts

$$\begin{aligned}
&|S_{l,\mathbf{i}}(f)|_{W^s(L_q(\Omega))} = b^{(d/q-s)(l-1)} |f|_{W^s(L_q(\Omega_{\mathbf{i}^-}))} \\
&S_{l,\mathbf{i}}(f_l) = S_{l,\mathbf{i}}(\Pi_k^l(f) - \Pi_k^{l-1}(f)) = \Pi_k^1(S_{l,\mathbf{i}}(f)) - \Pi_k^0(S_{l,\mathbf{i}}(f)) \\
&S_{l,\mathbf{i}}(\rho_{l,\mathbf{i}}^{\alpha}) = \rho_{1,\mathbf{j}}^{\alpha},
\end{aligned} \tag{4.13}$$

where $\mathbf{j} \in \{0,1,...,b\}^d$ is the index of $\Omega_{\mathbf{i}}^{l}$ in $\Omega_{\mathbf{i}^-}^{l-1}$, i.e. $\mathbf{j} \equiv \mathbf{i} \pmod{b}$. From the last two facts we deduce that

$$\Pi_k^1(S_{l,\mathbf{i}}(f)) - \Pi_k^0(S_{l,\mathbf{i}}(f)) = \sum_{\mathbf{j} \in I_1} a_{l,(b\mathbf{i}^- + \mathbf{j})}^{\alpha} \rho_{1,\mathbf{j}}^{\alpha},$$

where the $a_{l,(b\mathbf{i}^- + \mathbf{j})}^{\alpha}$ are the coefficients from the expansion (4.11) of f_l. Combining this with the first fact from (4.13), it suffices to prove (4.12) when $l = 1$ and apply this to $S_{l,\mathbf{i}}(f)$.

To prove (4.12) when $l = 1$, we use the Bramble-Hilbert lemma (Bramble and Hilbert, 1970). We calculate using the Bramble-Hilbert lemma that

$$\begin{aligned}
&\|\Pi_k^0(f) - f\|_{L_q(\Omega_{\mathbf{i}}^1)} \leq \|\Pi_k^0(f) - f\|_{L_q(\Omega)} \leq C|f|_{W^s(L_q(\Omega))} \\
&\|\Pi_k^1(f) - f\|_{L_q(\Omega_{\mathbf{i}}^1)} \leq C|f|_{W^s(L_q(\Omega_{\mathbf{i}}^1))} \leq C|f|_{W^s(L_q(\Omega))}.
\end{aligned}$$

Combining these two estimates, we get

$$\|\Pi_k^0(f)-\Pi_k^1(f)\|_{L_q(\Omega_{\mathbf{i}}^1)} \leq C|f|_{W^s(L_q(\Omega))}.$$

When $l=0$ we make the modification

$$\|\Pi_k^0(f)\|_{L_q(\Omega)} \leq \|f\|_{L_q(\Omega)} + \|\Pi_k^0(f)-f\|_{L_q(\Omega)} \leq C\|f\|_{W^s(L_q(\Omega))}.$$

Now we use the fact that all norms on the finite dimensional space of polynomials of degree at most k are equivalent to transfer the L_q bound to a bound on the coefficients. This implies (4.12).

From (4.12), we deduce the following bound on the ℓ^q-norm of the coefficients of f_l:

$$\left(\sum_{|\alpha|\leq k,\ \mathbf{i}\in I_l} |a_{l,\mathbf{i}}^\alpha|^q\right)^{1/q} \leq Cb^{(d/q-s)l}\left(\sum_{|\alpha|\leq k,\ \mathbf{i}\in I_l} |f|^q_{W^s(L_q(\Omega_{\mathbf{i}^-}^{l-1}))}\right)^{1/q} \leq Cb^{(d/q-s)l}, \tag{4.14}$$

since $\|f\|_{W^s(L_q(\Omega))} \leq 1$. This follows from the sub-additivity of the Sobolev norm,

$$\sum_{\mathbf{i}\in I_{l-1}} |f|^q_{W^s(L_q(\Omega_{\mathbf{i}}^{l-1}))} \leq |f|^q_{W^s(L_q(\Omega))}, \tag{4.15}$$

since each $\Omega_{\mathbf{i}^-}^{l-1}$ appears a finite number of times in the sum (4.14) (namely $\binom{k+d}{d}b^d$ which is independent of l).

We remark that the Sobolev sub-additivity (4.15) immediately follows from the definitions (1.1) and (1.2). Note also that the bound (4.14) also easily follows when the standard modifications are made for $q=\infty$.

Next, we derive the following bound, which follows from (4.14), the L^∞-normalization of the basis functions $\rho_{l,\mathbf{i}}^\alpha$, the fact that for fixed α the functions $\rho_{l,\mathbf{i}}^\alpha$ have disjoint support, and the assumption that $p\geq q$:

$$\begin{aligned}\|f_l\|_{L_p(\Omega)} &\leq \sum_{|\alpha|\leq k} b^{-dl/p}\left(\sum_{\mathbf{i}\in I_l}|a_{l,\mathbf{i}}^\alpha|^p\right)^{1/p} \\ &\leq b^{-dl/p}\binom{k+d}{d}^{1-1/p}\left(\sum_{|\alpha|\leq k,\ \mathbf{i}\in I_l}|a_{l,\mathbf{i}}^\alpha|^p\right)^{1/p} \\ &\leq Cb^{(d/q-d/p-s)l}.\end{aligned} \tag{4.16}$$

We now complete the proof by using Proposition 20 to approximate each f_l for $l=1,\dots,l^*$, for which we must choose appropriate parameters. First, we choose $\tau>0$ such that

$$\frac{d}{q}-\frac{d}{p}-s+\left(1-\frac{q}{p}\right)\tau < 0.$$

Note that this condition can be satisfied since $q\leq p$ and the Sobolev embedding condition (4.10) holds. For each level l we choose parameters

$$\delta = \delta(l) = \begin{cases} b^{-dl_0/q+\tau(l-l_0)} & l\geq l_0 \\ b^{-dl/q+(s+1)(l-l_0)} & l<l_0\end{cases} \tag{4.17}$$

and

$$m = m(l) = K_1 l_0 + K_2 l \tag{4.18}$$

in Proposition 20, where $K_1, K_2 > 0$ are parameters to be chosen later. Note that $\delta(l)^{-q} \leq b^{dl}$ when $l \geq l_0$ and $\delta(l)^{-q} > b^{dl}$ when $l < l_0$. This means that the coarser levels are discretized finely and the coefficients are dense, while the finer levels are discretized coarsely so that the coefficients are sparse.

So, we define the network $f_{l_0,\varepsilon}$ using Proposition 7 to be

$$f_{l_0,\varepsilon} = \sum_{l=0}^{l^*} f_{\delta(l),m(l)},$$

where $f_{\delta(l),m(l)}$ is constructed using Proposition 20 applied to f_l with parameters $\delta = \delta(l)$ and $m = m(l)$.

Propositions 7 and 20 imply that $f_{l_0,\varepsilon} \in \Upsilon^{24d+20,L}(\mathbb{R}^d)$ with

$$L \leq \sum_{l=0}^{l^*} L_l,$$

where the depths L_l for each level are bounded using Proposition 20 by

$$L_l \leq C \begin{cases} m(l) + l + \delta(l)^{-q/2}\sqrt{1 + dl\log(b) + q\log(\delta(l))} & \delta(l)^{-q} \leq b^{dl} \\ m(l) + l + b^{dl/2}\sqrt{1 - \log\delta(l) - (dl/q)\log(b)} & \delta(l)^{-q} > b^{dl}. \end{cases}$$

Plugging in the expressions for $\delta(l)$ and $m(l)$ given in (4.17) and (4.18), and using that $\delta(l)^{-q} \leq b^{dl}$ when $l \geq l_0$ and $\delta(l)^{-q} > b^{dl}$ when $l < l_0$, we get the bound

$$L \leq C\left(\sum_{l=0}^{l^*} K_1 l_0 + (K_2+1)\,l + b^{dl_0/2}\sum_{l=0}^{l_0-1} b^{d(l-l_0)/2}\sqrt{1+\log(b)(s+1)(l-l_0)}\right.$$
$$\left. + \, b^{dl_0/2}\sum_{l=l_0}^{l^*} b^{-\tau(l-l_0)/2}\sqrt{1+\log(b)(d/q+\tau)(l-l_0)}\right).$$

Summing the series above (and noting that the latter two are bounded by convergent geometric series), we get

$$L \leq C((l^*)^2 + b^{dl_0/2}).$$

Note that here the constant C depends upon the choice of parameters K_1 and K_2. Since $l^* \leq \kappa l_0$ is a linear function of l_0, the quadratic term $(l^*)^2 \leq (\kappa l_0)^2$ is dominated by the exponential second term. Thus we get $L \leq C b^{dl_0/2}$ (for a potentially larger constant C).

Finally, we bound the error. For this we use Proposition 20 to bound

$$\|f_{\delta(l),m(l)} - f_l\|_{L^p(\Omega_{l,\varepsilon})} \leq C\left(\delta(l)\min\left\{1, b^{-dl}\delta(l)^{-q}\right\}^{1/p} + 4^{-m(l)}\right)\left(\sum_{|\alpha|\leq k,\ \mathbf{i}\in I_l} |a^{\alpha}_{l,\mathbf{i}}|^q\right)^{1/q}.$$

Combining this with the bound (4.14), plugging in the choices (4.17) and (4.18) (here again we have $b^{-dl}\delta(l)^{-q} \leq 1$ when $l \geq l_0$ and $b^{-dl}\delta(l)^{-q} > 1$ when $l < l_0$), and noting that $\Omega_{l,\varepsilon} \supset \Omega_{l^*,\varepsilon}$ if $l \leq l^*$, we get

$$\sum_{l=0}^{l^*} \|f_{\delta(l),m(l)} - f_l\|_{L^p(\Omega_{l^*,\varepsilon})} \leq C\left(\sum_{l=0}^{l_0-1} b^{(d/q-s)l}\left[\delta(l) + 4^{-m(l)}\right] + \right.$$
$$\left. \sum_{l=l_0}^{l^*} b^{(d/q-s)l}\left[\delta(l)^{1-q/p} b^{-(d/p)l} + 4^{-m(l)}\right]\right).$$

Plugging in our choices for $\delta(l)$ and $m(l)$, we calculate

$$\sum_{l=0}^{l^*} \|f_{\delta(l),m(l)} - f_l\|_{L^p(\Omega_{l^*,\varepsilon})} \leq C\left(b^{-sl_0} \sum_{l=0}^{l_0-1} b^{l-l_0}\right.$$
$$+ b^{-sl_0} \sum_{l=l_0}^{l^*} b^{(d/q-d/p-s+\tau(1-q/p))(l-l_0)} \tag{4.19}$$
$$\left. + \sum_{l=0}^{l^*} b^{(d/q-s)l} 4^{-K_1 l_0 - K_2 l}\right).$$

The first sum above is a convergent geometric series and is bounded by Cb^{-sl_0}. Due to the choice of τ, the second sum is also a convergent gemoetric series, and is also bounded by Cb^{-sl_0}. Choosing K_1 and K_2 large enough so that $4^{-K_1} \leq b^{-s}$ and $4^{-K_2} < b^{(s-d/q)}$, the final sum is also a convergent geometric series which is bounded by Cb^{-sl_0}. Thus, we obtain

$$\sum_{l=0}^{l^*} \|f_{\delta(l),m(l)} - f_l\|_{L^p(\Omega_{l^*,\varepsilon})} \leq Cb^{-sl_0}$$

for an appropriate constant C. Finally, we estimate

$$\|f - f_{l_0,\varepsilon}\|_{L^p(\Omega_{l^*,\varepsilon})} \leq \sum_{l=0}^{l^*} \|f_{\delta(l),m(l)} - f_l\|_{L^p(\Omega_{l^*,\varepsilon})} + \sum_{l=l^*+1}^{\infty} \|f_l\|_{L^p(\Omega_{l^*,\varepsilon})}.$$

Utilizing (4.19) and (4.16) we get

$$\|f - f_{l_0,\varepsilon}\|_{L^p(\Omega_{l^*,\varepsilon})} \leq Cb^{-sl_0} + C\sum_{l=l^*+1}^{\infty} b^{(d/q-d/p-s)l}.$$

The compact Sobolev embedding condition implies that the second sum is a convergent geometric series, bounded by a multiple of its first term. This gives

$$\|f - f_{l_0,\varepsilon}\|_{L^p(\Omega_{l^*,\varepsilon})} \leq C(b^{-sl_0} + b^{(d/q-d/p-s)l^*}).$$

Finally, we use the definition of l^* and κ to see that

$$b^{(d/q-d/p-s)l^*} \leq Cb^{-sl_0},$$

which completes the proof. ■

We note that a completely analogous Proposition holds for the Besov spaces $B^s_r(L_q(\Omega))$, i.e. Proposition 21 holds with the Sobolev space $W^s(L_q(\Omega))$ replaced by $B^s_r(L_q(\Omega))$. The proof is exactly the same, utilizing a piecewise polynomial approximation, with the main difference being that the Bramble-Hilbert lemma is replaced by the following bound on piecewise polynomial approximation of Besov functions, known as Whitney's theorem (see DeVore and Popov (1988), Section 3 for instance)

$$\|\Pi^0_{k-1}(f) - f\|_{L^q(\Omega)} \leq C\omega_k(f,1)_q,$$

where $\Omega = [0,1]^d$ is the unit cube, ω_k is the modulus of smoothness introduced in (1.3), and the constant depends upon d,q and k. This is easily seen to imply that

$$\|\Pi^0_k(f) - f\|_{L^q(\Omega)} \leq C\|f\|_{B^s_r(L_q(\Omega))}$$

as soon as $k > s-1$ (for a different constant C which depends upon d,q,k,s and r), and the proof proceeds utilizing the same scaling argument. In addition, the sub-additivity (4.15) must be replaced by the corresponding result for Besov spaces

$$\sum_{\mathbf{i}\in I_{l-1}} |f|^q_{B^s_r(L_q(\Omega^{l-1}_{\mathbf{i}}))} \leq C|f|^q_{B^s_r(L_q(\Omega))},$$

which holds with a constant C depending upon the dimension and the particular space (see for instance DeVore and Sharpley (1993)). With these modifications, the proof proceeds in exactly the same way.

Finally, we show how to remove the trifling region to give a proof of Theorem 1. This is a technical construction similar to the method in Shen et al. (2022); Lu et al. (2021); Shijun (2021), but we significantly reduce the size of the required network (in particular the width no longer depends exponentially on the input dimension) by using the sorting network construction from Corollary 13.

In addition to using sorting network, in our approach we use different bases b_i to create minimally overlapping trifling regions. This is somewhat different than the aforementioned approaches (Shen et al., 2022; Lu et al., 2021; Shijun, 2021), which shift the grid to achieve the same effect. The reason we did this is to avoid the use of Sobolev and Besov extension theorems, as our method allows everything to stay within the unit cube. Although such extension theorems could be used, they become quite technical in full generality, and so we have found our approach simpler.

The proof of Theorem 2 is completely analogous using Proposition 21 with the Sobolev spaces replaced by Besov spaces, and is omitted.

Proof of Theorem 1 We assume without loss of generality that $f \in W^s(L_q(\Omega))$ has been normalized, i.e. so that $\|f\|_{W^s(L_q(\Omega))} \leq 1$.

In order to remove the trifling region from the preceding construction we will make use of different bases b. Let r be the smallest integer such that $2^r \geq 2d+2$ (so that $2^r \leq 4d+4$), set $m = 2^r$, and set $b_i = \pi_i$ (the i-th prime number) for $i = 1,...,m$.

Let $n \geq b_m$ be an integer. We will construct a network $f_L \in \Upsilon^{30d+24,L}(\mathbb{R}^d)$ such that

$$\|f - f_L\|_{L_p(\Omega)} \leq Cn^{-s}$$

with depth $L \leq Cn^{d/2}$, which will complete the proof.

For $i = 1,...,m$, set $l_i = \lfloor \log(n)/\log(b_i) \rfloor$ to be the largest power of b_i which is at most n, and write $l_i^* = \lfloor \kappa l_i \rfloor$ where κ is defined as in Proposition 21. Note that since the π_i are all pairwise relatively prime, the numbers

$$S := \left\{ \frac{1}{\pi_1^{l_1^*}}, ..., \frac{\pi_1^{l_1^*} - 1}{\pi_1^{l_1^*}}, \frac{1}{\pi_2^{l_2^*}}, ..., \frac{\pi_2^{l_2^*} - 1}{\pi_2^{l_2^*}}, ..., \frac{1}{\pi_m^{l_m^*}}, ..., \frac{\pi_m^{l_m^*} - 1}{\pi_m^{l_m^*}} \right\}$$

are all distinct. Choose an $\varepsilon > 0$ which satisfies

$$\varepsilon < \min_{x \neq y \in S} |x - y|, \tag{4.20}$$

i.e. which is smaller than the distance between the two closest elements of S. This ε has the property that any $x \in [0,1]$ is contained in at most one of the sets

$$[j\pi_i^{-l_i^*} - \varepsilon, j\pi_i^{-l_i^*}) \text{ for } i = 1,...,m \text{ and } j = 1,...,\pi_i^{l_i^*} - 1. \tag{4.21}$$

This means that for any $x \in \Omega$, we have $x \notin \Omega_{l_i^*,\varepsilon}$ for at most d different values i. Here $\Omega_{l_i^*,\varepsilon}$ is the good region at level l_i^* with base b_i. This holds since x has d coordinates and each coordinate can be contained in at most one bad set from (4.21).

We now use Proposition 21, setting $l_0 = l_i$ and using an ε satisfying (4.20), to construct $f_i \in \Upsilon^{24d+20,L}(\mathbb{R}^d)$ which satisfies

$$\|f - f_i\|_{L_p(\Omega_{l_i^*,\varepsilon})} \leq C\pi_i^{-sl_i} \leq Cn^{-s}$$

and has depth bounded by

$$L \leq C\pi_i^{dl_i/2} \leq Cn^{d/2}.$$

Finally, we construct the following network. We sequentially duplicate the input and apply the network f_i to the new copy using Lemma 6 to get

$$x \to \begin{pmatrix} x \\ x \end{pmatrix} \to \begin{pmatrix} x \\ f_1(x) \end{pmatrix} \to \begin{pmatrix} x \\ x \\ f_1(x) \end{pmatrix} \to \begin{pmatrix} x \\ f_2(x) \\ f_1(x) \end{pmatrix} \to \cdots \to \begin{pmatrix} f_m(x) \\ \vdots \\ f_2(x) \\ f_1(x) \end{pmatrix} \in \Upsilon^{30d+24,L}(\mathbb{R}^d, \mathbb{R}^m) \tag{4.22}$$

with $L \leq C\sum_{i=1}^m \pi_i^{dl_i/2} \leq Cn^{d/2}$.

We construct the network $f_L \in \Upsilon^{30d+24,L}(\mathbb{R}^d)$ by composing the network from (4.22) with the order statistic network which selects the median, i.e. the $m/2$-largest value. By construction the network depth of f_L satisfies

$$L \leq Cn^{d/2} + \binom{m+1}{2} \leq Cn^{d/2},$$

since $\binom{m+1}{2}$ is a constant independent of n.

To bound the approximation error of f_L we introduce the following notation. Given $x \in [0,1)^d$, we write

$$\mathscr{K}(x) = \{i : \ x \in \Omega_{l_i^*,\varepsilon}\}$$

for the set of indices such that x is contained in the good region for the base b_i decomposition. Since x fails to be in $\Omega_{l_i^*,\varepsilon}$ for at most d values of i, we get

$$|\mathscr{K}(x)| \geq m - d \geq m/2 + 1$$

since $m \geq 2d + 2$. Thus the $m/2$-largest element among the $f_1(x),...,f_m(x)$ is both smaller and larger than some element of $\{f_i(x),\ i \in \mathscr{K}(x)\}$, which implies

$$\min_{i \in \mathscr{K}(x)} f_i(x) \leq f_L(x) \leq \max_{i \in \mathscr{K}(x)} f_i(x),$$

so that

$$|f_L(x) - f(x)| \leq \max_{i\in\mathscr{K}(x)} |f_i(x) - f(x)|.$$

This completes the proof when $p = \infty$, since if $i \in \mathscr{K}(x)$ then $|f_i(x) - f(x)| \leq Cn^{-s}$ by Proposition 21 and the definition of $\mathscr{K}(x)$.

For $p < \infty$, we note that

$$\begin{aligned}\int_\Omega |f_L(x) - f(x)|^p dx \leq \int_\Omega \max_{i\in\mathscr{K}(x)} |f_i(x) - f(x)|^p dx &\leq \int_\Omega \sum_{i\in\mathscr{K}(x)} |f_i(x) - f(x)|^p dx \\ &\leq \sum_{i=1}^m \|f_i - f\|^p_{L^p(\Omega_{l_i^*,\varepsilon})} \\ &\leq Cn^{-sp}.\end{aligned}$$

Taking p-th roots completes the proof. ■

5. Lower Bounds

In this section, we study lower bounds on the approximation rates that deep ReLU neural networks can achieve on Sobolev spaces. Our main result is to prove Theorem 3, which shows that the construction of Theorem 1 is optimal in terms of the number of parameters. In addition, we show that the representation of sparse vectors proved in Theorem 14 is optimal.

The key concept is the notion of VC dimension, which was used in Yarotsky (2018); Shen et al. (2022) to prove lower bounds for approximation in the L_∞-norm. We generalize these results to obtain sharp lower bounds on the approximation in L_p as well. Let K be a class of functions defined on $\mathbb{R}^d$. The VC-dimension (Vapnik and Chervonenkis, 2015) of K is defined to be the largest number n such that there exists a set of points $x_1, ..., x_n \in \Omega$ such that

$$|\{(\mathrm{sgn}(g(x_1)), ..., \mathrm{sgn}(g(x_n))),\ g \in K\}| = 2^n,$$

i.e. such that every sign pattern at the points $x_1, ..., x_n$ can be matched by a function from K. Such a set of points is said to be shattered by K.

The VC dimension of classes of functions defined by neural networks has been extensively studied and the most precise results are available for piecewise polynomial activation functions. We will discuss two main results concerning the VC dimension of $\Upsilon^{W,L}(\mathbb{R}^d)$. The first bound is most useful when the depth L is fixed and the width W is large and is given by

$$\text{VC-dim}(\Upsilon^{W,L}(\mathbb{R}^d)) \leq C(W^2L^2\log(WL)). \tag{5.1}$$

This was proved in Theorem 6 of Bartlett et al. (2019). The second bound, which is most informative when the width W is fixed and the depth L is large is

$$\text{VC-dim}(\Upsilon^{W,L}(\mathbb{R}^d)) \leq C(W^3L^2). \tag{5.2}$$

This was proved in Theorem 8 of Bartlett et al. (2019) using a technique developed in Goldberg and Jerrum (1993). In either case, the VC-dimension of a deep ReLU neural network with $P = O(W^2L)$

parameters is bounded by CP^2, with this bound achieved up to a constant only in the case where the width W is fixed and the depth L grows. This bound on the VC-dimension was used in Yarotsky (2018); Shen et al. (2022) to prove Theorem 3 in the case $p = \infty$. However, in order to extend the lower bound to $p < \infty$ a more sophisticated analysis is required. The key argument is captured in the following Proposition.

Theorem 22 *Let $p > 0$, $\Omega = [0,1]^d$ and suppose that K is a translation invariant class of functions whose VC-dimension is at most n. By translation invariant we mean that $f \in K$ implies that $f(\cdot - v) \in K$ for any fixed vector $v \in \mathbb{R}^d$. Then there exists an $f \in W^s(L_\infty(\Omega)) \cap B^s_1(L_\infty(\Omega))$ such that*

$$\inf_{g\in K} \|f-g\|_{L^p(\Omega)} \geq C(p,d,s)n^{-\frac{s}{d}} \max\left\{\|f\|_{W^s(L_\infty(\Omega))}, \|f\|_{B^s_1(L_\infty(\Omega))}\right\}.$$

Although the translation invariance holds for many function classes of interest, it is an interesting problem whether it can be removed. Before proving this result, we first show how Theorem 3 follows from this.

Proof of Theorem 3 Note that the class of deep ReLU networks $\Upsilon^{W,L}(\mathbb{R}^d)$ is translation invariant. Combining this with the VC-dimension bounds (5.1) and (5.2), Theorem 22 implies Theorem 3 in the case $q = \infty$ and $r = 1$. The general case follows trivially since $W^s(L_\infty(\Omega)) \subset W^s(L_q(\Omega))$ for any $q \leq \infty$, and $B^s_1(L_\infty(\Omega) \subset B^s_r(L_q(\Omega))$ for $r \geq 1$ and $q \leq \infty$. ■

Let us turn to the proof of Theorem 22. A key ingredient is the well-known Sauer-Shelah lemma (Sauer, 1972; Shelah, 1972).

Lemma 23 (Sauer-Shelah Lemma) *Suppose that K has VC-dimension at most n. Given any collection of N points $x_1,...,x_N \in \Omega$, we have*

$$|\{(\text{sgn}(g(x_1)),...,\text{sgn}(g(x_N))),\ g\in K\}| \leq \sum_{i=0}^n \binom{N}{i}.$$

We will also utilize the following elementary bound on the size of a Hamming ball.

Lemma 24 *Suppose that $N \geq 2n$, then*

$$\sum_{i=0}^n \binom{N}{i} \leq 2^{NH(n/N)},$$

where $H(p)$ is the entropy function

$$H(p) = -p\log(p) - (1-p)\log(1-p).$$

(Note that all logarithms here are taken base 2.)

Proof Observe that since $N - n \geq n$, we have

$$\begin{aligned}\left(\frac{N-n}{N}\right)^{N-n}\left(\frac{n}{N}\right)^n \sum_{i=0}^n \binom{N}{i} &\leq \sum_{i=0}^n \binom{N}{i}\left(\frac{N-n}{N}\right)^{N-i}\left(\frac{n}{N}\right)^i \\ &< \sum_{i=0}^N \binom{N}{i}\left(\frac{N-n}{N}\right)^{N-i}\left(\frac{n}{N}\right)^i = 1.\end{aligned}$$

This means that

$$\sum_{i=0}^{n}\binom{N}{i} \leq \left[\left(\frac{N-n}{N}\right)^{N-n}\left(\frac{n}{N}\right)^{n}\right]^{-1}.$$

Taking logarithms, we obtain

$$\log\left(\sum_{i=0}^{n}\binom{N}{i}\right) \leq -N\left((N-n)\log\left(\frac{N-n}{N}\right)+n\log\left(\frac{n}{N}\right)\right) = NH(n/N)$$

as desired. ■

Utilizing these lemmas, we give the proof of Theorem 22.

Proof of Theorem 22 Let $c < 1/2$ be chosen so that $H(c) < 1/2$ (for instance $c = 0.1$ will work) and fix

$$k := \lceil \sqrt[d]{n/c} \rceil \leq C(d)n^{1/d},$$

and $\varepsilon := k^{-1}$. Next, we consider shifts of an equally spaced grid with side length ε. Specifically, for each $\lambda \in [0,\varepsilon)^d$, define the point set

$$X_\lambda = \left\{\lambda + \varepsilon z,\ z \in [k]^d\right\},$$

where we have written $[k] := \{0,...,k-1\}$ for the set of integers from 0 to $k-1$.

Let us now investigate the set of sign patterns which the class K can match on X_λ. To do this, we will introduce some notation. For a function $g \in K$, we write

$$\text{sgn}(g|_{X_\lambda}) \in \{\pm 1\}^{[k]^d},\ \text{sgn}(g|_{X_\lambda})(z) = \text{sgn}(g(\lambda+\varepsilon z))$$

for the set of signs which g takes at the (shifted) grid points X_λ. Here the vector $\text{sgn}(g|_{X_\lambda})$ is indexed by the coordinate $z \in [k]^d$ which specifies the location of a point in the shifted grid X_λ.

We write

$$\text{sgn}(K|_{X_\lambda}) := \left\{\text{sgn}(g|_{X_\lambda}),\ g \in K\right\} \subset \{\pm 1\}^{[k]^d}$$

for the set of sign patterns attained by the class K on X_λ. Observe that since K is assumed to be translation invariant, the set $\text{sgn}(K|_{X_\lambda})$ is independent of the shift λ. To see this, let $\lambda, \mu \in [0,\varepsilon)^d$ be two different shifts and let $g \in K$. By the translation invariance, we find that the function g' defined by

$$g'(x) = g(x+\lambda-\mu)$$

is also in K. We easily calculate that

$$\text{sgn}(g|_{X_\lambda}) = \text{sgn}(g'|_{X_\mu}),$$

which implies that $\text{sgn}(K|_{X_\lambda}) = \text{sgn}(K|_{X_\mu})$. In the following we simplify notation and write $\text{sgn}(K) \subset \{\pm 1\}^{[k]^d}$ for this set.

Next, we will show that there exists a choice of signs $\alpha \in \{\pm 1\}^{[k]^d}$ which differs from every element of $\text{sgn}(K)$ in a constant fraction of its entries. To do this, it is convenient to use the notion of the Hamming distance between two sign patterns, which is defined as the number of indices in which they differ, i.e.

$$d_H(\alpha,\beta) := |\{z \in [k]^d :\ \alpha(z) \neq \beta(z)\}|.$$

We also use the notion of the Hamming ball of radius m around a sign pattern $\alpha \in \{\pm 1\}^{[k]^d}$, which is defined to be the set of sign patterns which differ from α by at most m entries, i.e.

$$B_H(\alpha, m) = \{\beta \in \{\pm 1\}^{[k]^d},\ d_H(\alpha, \beta) \leq m\}.$$

We note that Lemma 24 implies the following estimate on the size of $B_H(\alpha, m)$ when $2m < k^d$:

$$|B_H(\alpha, m)| = \sum_{i=0}^{m} \binom{k^d}{i} \leq 2^{k^d H(m/k^d)}.$$

Further, our assumption on the VC-dimension of K combined with Lemmas 23 and 24 implies that

$$|\operatorname{sgn}(K)| \leq 2^{k^d H(n/k^d)} \leq 2^{k^d H(c)} < 2^{k^d/2}$$

from our choice of c. If we choose $m := \lfloor ck^d \rfloor \leq ck^d$, it follows that

$$\left| \bigcup_{\beta \in \operatorname{sgn}(K)} B_H(\beta, m) \right| < 2^{k^d/2} 2^{k^d H(m/k^d)} < 2^{k^d/2} 2^{k^d H(c)} < 2^{k^d},$$

so that there must exist an $\alpha \in \{\pm 1\}^{[k]^d}$ such that

$$\alpha \notin \bigcup_{\beta \in \operatorname{sgn}(K)} B_H(\beta, m),$$

and hence

$$\inf_{\beta \in \operatorname{sgn}(K)} d_H(\alpha, \beta) \geq m + 1 \geq ck^d. \tag{5.3}$$

Finally, we choose a compactly supported smooth positive bump function ϕ whose support is strictly contained in the unit cube Ω and consider the function

$$f(x) = \sum_{z \in [k]^d} \alpha(z) \phi(kx - z).$$

Since the supports of the functions $\phi(kx - z)$ are all disjoint, we calculate

$$\|f\|_{W^s(L_\infty(\Omega))} = \|\phi(kx - z)\|_{W^s(L_\infty(\Omega))} \leq k^s \|\phi\|_{W^s(L_\infty(\Omega))}, \tag{5.4}$$

and also

$$\|f\|_{B_1^s(L_\infty(\Omega))} \leq Ck^s \|\phi\|_{B_1^s(L_\infty(\Omega))}, \tag{5.5}$$

for an appropriate constant $C = C(d, s)$. Next, let $g \in K$ be arbitrary. We calculate

$$\begin{aligned} \int_\Omega |f(x) - g(x)|^p dx &= \int_{[0,\varepsilon)^d} \sum_{z \in [k]^d} |f(\lambda + \varepsilon z) - g(\lambda + \varepsilon z)|^p d\lambda \\ &= \int_{[0,\varepsilon)^d} \sum_{z \in [k]^d} |\alpha(z)\phi(k\lambda) - g(\lambda + \varepsilon z)|^p d\lambda. \end{aligned}$$

From equation (5.3) and the fact that $\operatorname{sgn}(g|_{X_\lambda}) \in \operatorname{sgn}(K)$, we see that

$$|\{z \in [k]^d,\ \alpha(z) \neq \operatorname{sgn}(g(\lambda + \varepsilon z))\}| \geq ck^d.$$

Further, if $\alpha(z) \neq \mathrm{sgn}(g(\lambda + \varepsilon z))$, then we have the lower bound

$$|\alpha(z)\phi(k\lambda) - g(\lambda + \varepsilon z)| \geq \phi(k\lambda)$$

since $\phi \geq 0$. This implies that for every $\lambda \in [0,\varepsilon)^d$ we have the lower bound

$$\sum_{z \in [k]^d} |\alpha(z)\phi(k\lambda) - g(\lambda + \varepsilon z)|^p \geq ck^d \phi(k\lambda)^p.$$

We thus obtain

$$\int_\Omega |f(x) - g(x)|^p dx \geq ck^d \int_{[0,\varepsilon)^d} \phi(k\lambda)^p d\lambda = c\int_\Omega \phi(x)^p dx,$$

from which we deduce

$$\|f-g\|_{L^p(\Omega)} \geq c^{\frac{1}{p}} \|\phi\|_{L^p(\Omega)}.$$

Combining this with the bounds (5.4) and (5.5), using that $k \leq C(d)n^{1/d}$, that ϕ is a fixed function, and that $g \in K$ was arbitrary, we get

$$\inf_{g\in K} \|f-g\|_{L^p(\Omega)} \geq C(d,p)k^{-s}\|f\|_{W^s(L_\infty(\Omega))} \geq C(d,k)n^{-s/d} \max\left\{\|f\|_{W^s(L_\infty(\Omega))}, \|f\|_{B^s_1(L_\infty(\Omega))}\right\},$$

as desired. ■

We conclude this section by proving that Theorem 14 is optimal up to a constant as long as the ℓ^1-norm M is not too large and not too small. Specifically, we have the following.

Theorem 25 *Let $M,N \geq 1$ be integers and define*

$$S_{N,M} = \{\mathbf{x} \in \mathbb{Z}^N,\ \|\mathbf{x}\|_{\ell^1} \leq M\}$$

as in the proof of Theorem 14. Suppose that $W,L \geq 1$ are integers and that for any $\mathbf{x} \in S_{N,M}$ there exists an $f \in \Upsilon^{W,L}(\mathbb{R})$ such that $f(i) = \mathbf{x}_i$ for $i = 1,..,N$. There exists a constant $C < \infty$ such that if $C\log(N) < M \leq N$, then

$$W^4L^2 \geq C^{-1}M(1+\log(N/M)),$$

and if $N \leq M < \exp(N/C)$, then

$$W^4L^2 \geq C^{-1}N(1+\log(M/N)).$$

This result implies that if $\Upsilon^{W,L}(\mathbb{R})$ can match the values of any vector in $S_{N,M}$ for M in the range $(C\log(N), \exp(N/C))$, then the number of parameters must be larger than a constant multiple of the upper bound proved in Theorem 14. Thus Theorem 14 is sharp in this range. If $M < C\log(N)$ then piecewise linear functions with $O(M)$ pieces can fit $S_{N,M}$, and if $M > \exp(N/C)$ then piecewise linear functions with $O(N)$ pieces can fit $S_{N,M}$. This implies that Theorem 14 is no longer sharp outside this range.

Proof Suppose first that $N/2 < M \leq 2N$, i.e. that M is of the same order as N. For any subset $S \subset \{1,...,N/2\}$ it is easy to construct an $\mathbf{x} \in S_{N,M}$ such that $\mathbf{x}_i > 0$ iff $i \in S$. Thus the class $\Upsilon^{W,L}(\mathbb{R})$ must shatter a set of size at least $N/2$ and the VC-dimension bound (5.2) implies the result.

In the case where $M << N$ or $M >> N$, the proof proceeds in a similar manner as the VC-dimension bounds from Goldberg and Jerrum (1993); Bartlett et al. (2019) although the VC-dimension cannot directly be used.

We begin with the case where $M \leq N/2$. We will bound the total number of sign patterns that $\Upsilon^{W,L}(\mathbb{R})$ can match on the input set $X = \{1,...,N\}$. For $i = 1,...,L$, let $\varepsilon_i \in \{0,1\}^W$ be a sign pattern. Given an input $x \in X$ and a neural network with parameters $\mathbf{W}_i$ and b_i, consider the signs of the following quantities

$$\begin{aligned} (A_{\mathbf{W}_0,b_0}(x))_j,\ j = 1,...,W \\ (A_{\mathbf{W}_1,b_1}\circ\varepsilon_1\circ A_{\mathbf{W}_0,b_0}(x))_j,\ j = 1,...,W \\ (A_{\mathbf{W}_2,b_2}\circ\varepsilon_2\circ A_{\mathbf{W}_1,b_1}\circ\varepsilon_1\circ A_{\mathbf{W}_0,b_0}(x))_j,\ j = 1,...,W \\ \vdots \\ (A_{\mathbf{W}_{L-1},b_{L-1}}\circ\varepsilon_{L-1}\circ\cdots\circ\varepsilon_2\circ A_{\mathbf{W}_1,b_1}\circ\varepsilon_1\circ A_{\mathbf{W}_0,b_0}(x))_j,\ j = 1,..,W \\ A_{\mathbf{W}_L,b_L}\circ\varepsilon_L\circ A_{\mathbf{W}_{L-1},b_{L-1}}\circ\varepsilon_{L-1}\circ\cdots\circ\varepsilon_2\circ A_{\mathbf{W}_1,b_1}\circ\varepsilon_1\circ A_{\mathbf{W}_0,b_0}(x). \end{aligned} \tag{5.6}$$

Here ε_i represents pointwise multiplication by the sign vector ε_i. For any input $x \in \mathbb{R}$ the definition of the ReLU activation function implies that if we recursively set

$$\varepsilon_i = \mathrm{sgn}(A_{\mathbf{W}_{i-1},b_{i-1}}\circ\varepsilon_{i-1}\circ\cdots\circ\varepsilon_2\circ A_{\mathbf{W}_1,b_1}\circ\varepsilon_1\circ A_{\mathbf{W}_0,b_0}(x)), \tag{5.7}$$

then we will have

$$A_{\mathbf{W}_L,b_L}\circ\varepsilon_L\circ\cdots\circ\varepsilon_2\circ A_{\mathbf{W}_1,b_1}\circ\varepsilon_1\circ A_{\mathbf{W}_0,b_0}(x) = A_{\mathbf{W}_L,b_L}\circ\sigma\circ\cdots\circ\sigma\circ A_{\mathbf{W}_1,b_1}\circ\sigma\circ A_{\mathbf{W}_0,b_0}(x).$$

This implies that the signs of the quantities in (5.6) ranging over all sign vectors $\varepsilon_1,...,\varepsilon_L \in \{0,1\}^W$ uniquely determine the value of the network at x. Thus the number of sign patterns achieved on the set X is bounded by the number of sign patterns achieved in (5.6) as x ranges over the input set X, the ε_i range over the sign vectors $\{0,1\}^W$, and the parameters $\mathbf{W}_i$, b_i range of the set of all real numbers. As the ε_i range over the sign vectors $\{0,1\}^W$ and x ranges over X, the quantities in (5.6) range over $N(WL+1)2^{WL}$ polynomials in the $P \leq CW^2L$ parameter variables $\mathbf{W}_i$, b_i of degree at most L. We can thus use Warren's Theorem (Warren (1968), Theorem 3) to bound the total number of sign patterns by

$$\left(\frac{4eLN(WL+1)2^{WL}}{P}\right)^P \leq (4eLN(WL+1)2^{WL})^{CW^2L}. \tag{5.8}$$

Suppose that $\Upsilon^{W,L}(\mathbb{R})$ can match the values of any element in $S_{N,M}$. Since the set $S_{N,M}$ contains the indicator function of every subset of $\{1,...,N\}$ of size M, we get that

$$\binom{N}{M} \leq (4eLN(WL+1)2^{WL})^{CW^2L}.$$

Taking logarithms, we get

$$\begin{aligned} M\log(N/M) &\leq CW^3L^2 + CW^2L\log(N) + CW^2L\log(4eL(WL+1)) \\ &\leq CW^3L^2 + CW^2L\log(N) \leq CW^4L^2 + CW^2L\log(N). \end{aligned}$$

Since $M \leq N/2$, we conclude that

$$M(1+\log(N/M)) \leq CM\log(N/M) \leq C\max\{W^4L^2, W^2L\log(N)\}. \tag{5.9}$$

In the next few equations, let C denote the constant in (5.9). Suppose that $W^4L^2 < C^{-1}M(1+\log(N/M))$. Then equation (5.9) implies that

$$W^2L \geq M\frac{1+\log(N/M)}{C\log(N)}.$$

But this would mean that

$$M\frac{1+\log(N/M)}{C\log(N)} \leq W^2L = \sqrt{W^4L^2} < \sqrt{C^{-1}M(1+\log(N/M))}.$$

Rearranging this, we get the inequality

$$\sqrt{M} < \frac{\sqrt{C}\log(N)}{\sqrt{1+\log(N/M)}},$$

from which we deduce that $M \leq C\log(N)$ for a (potentially larger) new constant C.

Next, we consider the case where $M > 2N$. In this case we consider the following modification of (5.6)

$$\begin{aligned}
&(A_{\mathbf{W}_0,b_0}(x))_j,\ j=1,...,W\\
&(A_{\mathbf{W}_1,b_1}\circ\varepsilon_1\circ A_{\mathbf{W}_0,b_0}(x))_j,\ j=1,...,W\\
&(A_{\mathbf{W}_2,b_2}\circ\varepsilon_2\circ A_{\mathbf{W}_1,b_1}\circ\varepsilon_1\circ A_{\mathbf{W}_0,b_0}(x))_j,\ j=1,...,W\\
&\vdots\\
&(A_{\mathbf{W}_{L-1},b_{L-1}}\circ\varepsilon_{L-1}\circ\cdots\circ\varepsilon_2\circ A_{\mathbf{W}_1,b_1}\circ\varepsilon_1\circ A_{\mathbf{W}_0,b_0}(x))_j,\ j=1,..,W\\
&A_{\mathbf{W}_L,b_L}\circ\varepsilon_L\circ A_{\mathbf{W}_{L-1},b_{L-1}}\circ\varepsilon_{L-1}\circ\cdots\circ\varepsilon_2\circ A_{\mathbf{W}_1,b_1}\circ\varepsilon_1\circ A_{\mathbf{W}_0,b_0}(x)-k,\ k=0,...\lfloor M/N\rfloor-1.
\end{aligned}\tag{5.10}$$

The number of sign patterns that can be obtained as x ranges over X, the ε_i range over $\{0,1\}^W$, and the parameters range over the set of real numbers is bounded (using Warren's Theorem Warren (1968) by

$$\left(\frac{4eLN(WL+M)2^{WL}}{P}\right)^P \leq (4eLN(WL+M)2^{WL})^{CW^2L}.$$

The only difference to the previous bound (5.8) is that for each choice of $x \in X$ and each choice of signs $\varepsilon_i \in \{0,1\}^W$, the number of equations in (5.10) is $WL + \lfloor M/N\rfloor \leq WL+M$.

The set $S_{N,M}$ contains all $(\lfloor M/N\rfloor+1)^{N-1}$ vectors whose first $N-1$ coordinates are arbitrary integers in $\{0,1,...,\lfloor M/N\rfloor\}$ and whose last coordinate is chosen to make the ℓ^1-norm equal to M. Thus, setting ε_i recursively according to (5.7), we see that if every vector in $S_{N,M}$ can be represented by an element of $\Upsilon^{W,L}(\mathbb{R})$, then

$$(\lfloor M/N\rfloor+1)^{N-1} \leq (4eLN(WL+M)2^{WL})^{CW^2L}.$$

Taking logarithms and calculating in a similar manner as before, we get

$$N(1+\log(M/N)) \leq CW^4L^2 + CW^2L\log(M).$$

As before we now deduce that if $W^4L^2 < C^{-1}N(1+\log(M/N))$, then $N \leq C\log(M)$. ■

Acknowledgments

We would like to thank Ron DeVore for suggesting this problem, and Andrea Bonito, Geurgana Petrova, Zuowei Shen, George Karniadakis, Jinchao Xu, and Qihang Zhou for helpful comments while preparing this manuscript. We would also like to thank the anonymous reviewers for their careful reading and helpful comments. This work was supported by the National Science Foundation (DMS-2111387 and CCF-2205004) as well as a MURI ONR grant N00014-20-1-2787.

Appendix A. Elementary Constructions

Here we give the constructions of sums and of piecewise linear continuous functions using deep ReLU networks references in Section 2.

Proof of Proposition 7 We will show by induction on j that

$$\begin{pmatrix} x \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} x \\ \sum_{i=1}^{j} f_i(x) \end{pmatrix} \in \Upsilon^{W+2d+2k,L}(\mathbb{R}^{d+k},\mathbb{R}^{d+k})$$

for $L = \sum_{i=1}^{j} L_i$. The base case $j=0$ is trivial since the identity map is affine. Suppose we have shown this for $j-1$, i.e.

$$\begin{pmatrix} x \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} x \\ \sum_{i=1}^{j-1} f_i(x) \end{pmatrix} \in \Upsilon^{W+2d+2k,L}(\mathbb{R}^{d+k},\mathbb{R}^{d+k}),$$

where $L = \sum_{i=1}^{j-1} L_i$. Compose this map with an affine map which duplicates the first entry to get

$$\begin{pmatrix} x \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} x \\ x \\ \sum_{i=1}^{j-1} f_i(x) \end{pmatrix} \in \Upsilon^{W+2d+2k,L}(\mathbb{R}^{d+k},\mathbb{R}^{2d+k}).$$

Now, we use Lemma 6 to apply f_j to the middle entry. This gives

$$\begin{pmatrix} x \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} x \\ f_j(x) \\ \sum_{i=1}^{j-1} f_i(x) \end{pmatrix} \in \Upsilon^{W+2d+2k,L+L_j}(\mathbb{R}^{d+k},\mathbb{R}^{d+2k}).$$

We finally compose with the affine map

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y+z \end{pmatrix} \in \Upsilon^{0}(\mathbb{R}^{d+2k},\mathbb{R}^{d+k}),$$

and apply Lemma 4 to obtain

$$\begin{pmatrix} x \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} x \\ \sum_{i=1}^{j} f_i(x) \end{pmatrix} \in \Upsilon^{W+2d+2k,L+L_j}(\mathbb{R}^{d+k},\mathbb{R}^{d+k}),$$

which completes the inductive step.

Applying the induction up to $j = n$, we have that

$$x \to \begin{pmatrix} x \\ 0 \end{pmatrix} \to \begin{pmatrix} x \\ \sum_{i=1}^n f_i(x) \end{pmatrix} \to \sum_{i=1}^n f_i(x) \in \Upsilon^{W+2k+2,L}(\mathbb{R}^d, \mathbb{R}^k),$$

where $L = \sum_{i=1}^n L_i$, since the first and last maps above are affine (applying Lemma 4). ■

Proof of Proposition 8 First observe that any piecewise linear function f with k pieces can be written as

$$f(x) = a_0 x + c + \sum_{i=1}^{k-1} a_i \sigma(x - b_i) \tag{A.1}$$

for appropriate weights $a_0, ..., a_{k-1}$ and $b_1, ..., b_{k-1}$. Specifically, the b_i are simply equal to the breakpoints points at which the derivative of f is discontinuous, while the a_i give the jump in derivative at those points. a_0 is set equal to the derivative in the left-most component and c is set to match the value at 0.

Now we apply Proposition 7 to the sum (A.1) to get the desired result, since we easily see that

$$x \to a_0 x + c \in \Upsilon^0(\mathbb{R})$$

and

$$x \to a_i \sigma(x - b_i) \in \Upsilon^{1,1}(\mathbb{R}).$$

■

Proof of Lemma 11 We observe the basic formulas:

$$\max(x,y) = x + \sigma(y - x), \ \min(x,y) = x - \sigma(x - y).$$

We begin with the affine map

$$\begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} x \\ y - x \\ x - y \end{pmatrix} \in \Upsilon^0(\mathbb{R}^2, \mathbb{R}^3).$$

Next, we use the fact that $\sigma \in \Upsilon^{1,1}(\mathbb{R})$ and Lemmas 4 and 6 to apply σ to the last two coordinates. We get

$$\begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} x \\ \sigma(y - x) \\ \sigma(x - y) \end{pmatrix} \in \Upsilon^{4,1}(\mathbb{R}^2, \mathbb{R}^3).$$

Finally, we use Lemma 4 to compose with the affine map

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} x + y \\ x - z \end{pmatrix}.$$

■

Appendix B. Product Network Construction

Proof of Proposition 9 Observe that the piecewise linear hat function

$$f(x) = \begin{cases} 2x & x \leq 1/2 \\ 2(1-x) & x > 1/2 \end{cases}$$

satisfies $f \in \Upsilon^{5,1}(\mathbb{R})$ by Proposition 8. On the interval $[0,1]$, f composed with itself n times is the sawtooth function

$$f^{\circ n}(x) := (f \circ \cdots \circ f)(x) = f(2^{n-1}x - \lfloor 2^{n-1}x \rfloor),$$

and one can calculate that (see Yarotsky (2017))

$$x^2 = x - \sum_{n=1}^{\infty} 4^{-n} f^{\circ n}(x) \tag{B.1}$$

for $x \in [0,1]$.

Using this, we construct a network $g_k \in \Upsilon^{7,k}(\mathbb{R})$ such that

$$\sup_{x\in[0,1]} |x^2 - g_k(x)| \leq 4^{-k}. \tag{B.2}$$

To do this, we first apply the affine map which duplicates the input

$$x \to \begin{pmatrix} x \\ x \end{pmatrix} \in \Upsilon^0(\mathbb{R},\mathbb{R}^2). \tag{B.3}$$

Next, we show by induction on k that the map

$$x \to \begin{pmatrix} x - \sum_{n=1}^{k} 4^{-n} f^{\circ n}(x) \\ f^{\circ k}(x) \end{pmatrix} \in \Upsilon^{7,k}(\mathbb{R},\mathbb{R}^2). \tag{B.4}$$

The base case $k = 0$ is simply (B.3).

For the inductive step suppose that (B.4) holds for $k \geq 0$. We use Lemma 6 to apply $f \in \Upsilon^{5,1}(\mathbb{R})$ to the second coordinate, showing that

$$\begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} x \\ f(y) \end{pmatrix} \in \Upsilon^{7,1}(\mathbb{R}^2,\mathbb{R}^2).$$

Using the inductive assumption and Lemma 4, we compose this with the map in (B.4) to get

$$x \to \begin{pmatrix} x - \sum_{n=1}^{k} 4^{-n} f^{\circ n}(x) \\ f^{\circ (k+1)}(x) \end{pmatrix} \in \Upsilon^{7,k+1}(\mathbb{R},\mathbb{R}^2).$$

We again use Lemma 4 and compose with the affine map

$$\begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} x+y \\ y \end{pmatrix} \in \Upsilon^0(\mathbb{R}^2,\mathbb{R}^2)$$

to complete the inductive step.

To construct g_k we then simply compose the map in (B.4) with the affine map

$$\begin{pmatrix} x \\ y \end{pmatrix} \to x \in \Upsilon^0(\mathbb{R}^2, \mathbb{R})$$

which forgets the second coordinate. Then for $x \in [0,1]$ we have

$$g_k(x) = x - \sum_{n=1}^{k} 4^{-n} f^{\circ n}(x)$$

and by (B.1) we get the bound (B.2). This gives us a network which approximates x^2 on the interval $[0,1]$.

In order to obtain a network which approximates x^2 on $[-1,1]$ we observe that if $x \in [-1,1]$, then $\sigma(x), \sigma(-x) \in [0,1]$, and

$$x^2 = \sigma(x)^2 + \sigma(-x)^2.$$

We begin with the single layer network

$$x \to \begin{pmatrix} \sigma(x) \\ \sigma(-x) \end{pmatrix} \in \Upsilon^{2,1}(\mathbb{R}, \mathbb{R}^2). \tag{B.5}$$

Further, applying Lemma 6, we see that

$$\begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} g_k(x) \\ y \end{pmatrix} \in \Upsilon^{9,k}(\mathbb{R}^2, \mathbb{R}^2)$$

and also

$$\begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} x \\ g_k(y) \end{pmatrix} \in \Upsilon^{9,k}(\mathbb{R}^2, \mathbb{R}^2)$$

Finally, composing all of these and then applying the affine summation map

$$\begin{pmatrix} x \\ y \end{pmatrix} \to x + y \in \Upsilon^0(\mathbb{R}^2),$$

we get, using Lemma 4 (note that we can expand the width of the network in (B.5)), a function $h_k \in \Upsilon^{9,2k+1}(\mathbb{R})$ such that on $[-1,1]$, we have

$$|x^2 - h_k(x)| \leq |\sigma(x)^2 - h_k(\sigma(x))| + |\sigma(-x)^2 - h_k(\sigma(-x))| \leq 4^{-k}.$$

(Since one of $\sigma(x)$ and $\sigma(-x)$ is 0.)

Finally, to construct a network which approximates products, we use the formula

$$xy = 2\left(\left(\frac{x+y}{2}\right)^2 - \left(\frac{x}{2}\right)^2 - \left(\frac{y}{2}\right)^2\right)$$

If $x, y \in [-1,1]$, then all of the terms which are squared in the previous equation are also in $[-1,1]$, so that we can approximate these squares using the network h_k. Applying the affine map

$$\begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} (x+y)/2 \\ x/2 \\ y/2 \end{pmatrix} \in \Upsilon^0(\mathbb{R}^2, \mathbb{R}^3),$$

then successively applying h_k to the first, second, and third coordinates using Lemmas 6 and 4, and finally applying the affine map

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to 2(x-y-z) \in \Upsilon^0(\mathbb{R}^3),$$

we obtain a network $f_k \in \Upsilon^{13,6k+3}(\mathbb{R}^2)$ such that for $x, y \in [-1,1]$ we have

$$|f_k(x,y) - xy| \leq 6 \cdot 4^{-k},$$

as desired. ■

Appendix C. Bit Extraction Network Construction

Proof of Proposition 10 We begin by noting that for any $\varepsilon > 0$ the piecewise linear maps

$$b_\varepsilon(x) = \begin{cases} 0 & x \leq 1/2 - \varepsilon \\ \varepsilon^{-1}(x - 1/2 + \varepsilon) & 1/2 - \varepsilon < x \leq 1/2 \\ 1 & x > 1/2 \end{cases}$$

and

$$g_\varepsilon(x) = \begin{cases} x & x \leq 1 - \varepsilon \\ \frac{1-\varepsilon}{\varepsilon}(1-x) & 1 - \varepsilon < x \leq 1 \\ x - 1 & x > 1 \end{cases}$$

satisfy $b_\varepsilon, g_\varepsilon \in \Upsilon^{5,2}(\mathbb{R})$ by Proposition 8. In addition, these functions have been designed so that if $\varepsilon < 2^{-n}$, we have for any x of the form (2.1) that

$$b_\varepsilon(x) = x_1,\ g_\varepsilon(2x) = 0.x_2x_3\cdots x_n.$$

We now construct the network $f_{n,m}$ by induction on m. In what follows, we assume that all of our inputs x are of the form (2.1). The base case when $m = 0$ is simply the affine map

$$x \to \begin{pmatrix} x \\ 0 \end{pmatrix} \in \Upsilon^0(\mathbb{R}, \mathbb{R}^2).$$

For the inductive step, we suppose that we have constructed a map

$$f_{n,m-1}(x) = \begin{pmatrix} 0.x_m x_{m+1} \cdots x_n \\ x_1 x_2 \cdots x_{m-1}.0 \end{pmatrix} \in \Upsilon^{9,4(m-1)}(\mathbb{R}, \mathbb{R}^2)$$

We then compose this network with an affine map which doubles and duplicates the first component

$$\begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} 2x \\ x \\ y \end{pmatrix} \in \Upsilon^0(\mathbb{R}^2, \mathbb{R}^3)$$

to get the map

$$x \to \begin{pmatrix} x_m.x_{m+1}\cdots x_n \\ 0.x_mx_{m+1}\cdots x_n \\ x_1x_2\cdots x_{m-1}.0 \end{pmatrix} \in \Upsilon^{9,4(m-1)}(\mathbb{R},\mathbb{R}^3).$$

Next we choose $\varepsilon < 2^{-n}$ and use Lemmas 4 and 6 to apply g_ε to the first component and then b_ε to the second component. This gives a map

$$x \to \begin{pmatrix} 0.x_{m+1}\cdots x_n \\ x_m \\ x_1x_2\cdots x_{m-1}.0 \end{pmatrix} \in \Upsilon^{9,4m}(\mathbb{R},\mathbb{R}^3).$$

Finally, we complete the inductive step by composing with the affine map

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} x \\ 2z+y \end{pmatrix} \in \Upsilon^0(\mathbb{R}^3,\mathbb{R}^2).$$

■

Proof of Lemma 19 Start with the following piecewise linear function

$$g_\varepsilon(x) := \begin{cases} 0 & x \leq 0 \\ (j+\varepsilon^{-1}(x-j/b)) & j/b-\varepsilon < x \leq j/b, \text{ for } j = 1,...,b-1 \\ j & j/b < x \leq (j+1)/b-\varepsilon, \text{ for } j = 0,...,b-1 \\ b-1 & x > 1. \end{cases}$$

Note that this function has $2b-1$ pieces and so by Proposition 8 we have $g_\varepsilon \in \Upsilon^{5,2(b-1)}(\mathbb{R})$.

We set $x_0 = x$ and $q_0 = 0$ and consider the following recursion

$$x_{n+1} = bx_n - g_\varepsilon(x_n),\ q_{n+1} = bq_n + g_\varepsilon(x_n).$$

It is easy to verify that if $x_0 = x \in [jb^{-l},(j+1)b^{-l}-\varepsilon)$, then $q_l = j$, since in this case all iterates $x_n \notin \cup_{j=1}^b (j/b-\varepsilon, j/b)$ so that g_ε extracts the first bit in the b-ary expansion of x_n

$$g_\varepsilon(x_n) = j \text{ if } j/b \leq x_n < (j+1)/b.$$

In addition, when $x_0 = x \in [1-b^{-l},1]$, then $q_l = b^l - 1$ since $g_\varepsilon(x_n) = b-1$ for all n.

We implement this recursion using a deep ReLU network as follows. We begin with the affine map

$$x \to \begin{pmatrix} x \\ x \\ 0 \end{pmatrix} = \begin{pmatrix} x_0 \\ x_0 \\ q_0 \end{pmatrix} \in \Upsilon^0(\mathbb{R},\mathbb{R}^3).$$

Now, we use induction. Suppose that the map

$$x \to \begin{pmatrix} x_n \\ x_n \\ q_n \end{pmatrix} \in \Upsilon^{9,2(b-1)n}(\mathbb{R},\mathbb{R}^3)$$

has already been implemented. Then we use Lemmas 4 and 6 to apply g_ε to only the second coordinate. This gives the map

$$x \to \begin{pmatrix} x_n \\ g_\varepsilon(x_n) \\ q_n \end{pmatrix} \in \Upsilon^{9,2(b-1)(n+1)}(\mathbb{R},\mathbb{R}^3).$$

Finally, we compose with the affine map

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} 2(x-y) \\ 2(x-y) \\ 2z+y \end{pmatrix} \in \Upsilon^0(\mathbb{R}^3,\mathbb{R}^3)$$

to complete the inductive step. After l steps of induction, we then compose with the affine map which selects the last coordinate to get the network $q_1 \in \Upsilon^{9,2(b-1)l}(\mathbb{R})$.

For higher dimensional cubes $\Omega = [0,1)^d$, we construct an indexing network $q_d \in \Upsilon^{9d,2(b-1)l}(\mathbb{R}^d)$. We use Lemma 5 to apply q_1 to each coordinate of the input. Then, we compose with the affine map

$$x \to \sum_{j=1}^d b^{l(j-1)} x_j$$

to get $q_d \in \Upsilon^{9d,2(b-1)l}(\mathbb{R}^d)$ with

$$q_d(\Omega^l_{\mathbf{i},\varepsilon}) = \text{ind}(\mathbf{i}) := \sum_{j=1}^d b^{l(j-1)}\mathbf{i}_j.$$

■

References

El Mehdi Achour, Armand Foucault, Sébastien Gerchinovitz, and François Malgouyres. A general approximation lower bound in L^p norm, with applications to feed-forward neural networks. *arXiv preprint arXiv:2206.04360*, 2022.

Miklós Ajtai, János Komlós, and Endre Szemerédi. An $O(n \log n)$ sorting network. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of computing*, pages 1–9, 1983.

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

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

Peter Bartlett, Vitaly Maiorov, and Ron Meir. Almost linear VC dimension bounds for piecewise polynomial networks. *Advances in Neural Information Processing Systems*, 11, 1998.

Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *The Journal of Machine Learning Research*, 20(1):2285–2301, 2019.

Kenneth E Batcher. Sorting networks and their applications. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, pages 307–314, 1968.

Mikhail Shlemovich Birman and Mikhail Zakharovich Solomyak. Piecewise-polynomial approximations of functions of the classes W_p^α. *Matematicheskii Sbornik*, 115(3):331–355, 1967.

James H Bramble and SR Hilbert. Estimation of linear functionals on Sobolev spaces with application to Fourier transforms and spline interpolation. *SIAM Journal on Numerical Analysis*, 7(1):112–124, 1970.

James H Bramble, Joseph E Pasciak, and Jinchao Xu. Parallel multilevel preconditioners. *Mathematics of Computation*, 55(191):1–22, 1990.

Antonin Chambolle, Ronald A DeVore, Nam-Yong Lee, and Bradley J Lucier. Nonlinear wavelet image processing: variational problems, compression, and noise removal through wavelet shrinkage. *IEEE Transactions on Image Processing*, 7(3):319–335, 1998.

Ingrid Daubechies. *Ten Lectures on Wavelets*. SIAM, 1992.

Ingrid Daubechies, Ronald DeVore, Nadav Dym, Shira Faigenbaum-Golovin, Shahar Z Kovalsky, Kung-Chin Lin, Josiah Park, Guergana Petrova, and Barak Sober. Neural network approximation of refinable functions. *IEEE Transactions on Information Theory*, 2022a.

Ingrid Daubechies, Ronald DeVore, Simon Foucart, Boris Hanin, and Guergana Petrova. Nonlinear approximation and (deep) ReLU networks. *Constructive Approximation*, 55(1):127–172, 2022b.

Françoise Demengel, Gilbert Demengel, and Reinie Erné. *Functional Spaces for the Theory of Elliptic Partial Differential Equations*. Springer, 2012.

Ronald DeVore, Boris Hanin, and Guergana Petrova. Neural network approximation. *Acta Numerica*, 30:327–444, 2021.

Ronald A DeVore. Nonlinear approximation. *Acta Numerica*, 7:51–150, 1998.

Ronald A DeVore and George G Lorentz. *Constructive Approximation*, volume 303. Springer Science & Business Media, 1993.

Ronald A DeVore and Vasil A Popov. Interpolation of besov spaces. *Transactions of the American Mathematical Society*, 305(1):397–414, 1988.

Ronald A DeVore and Robert C Sharpley. Maximal functions measuring smoothness. *Memoirs of the American Mathematical Society*, 47(293), 1984.

Ronald A DeVore and Robert C Sharpley. Besov spaces on domains in $\mathbb{R}^d$. *Transactions of the American Mathematical Society*, 335(2):843–864, 1993.

Ronald A DeVore, Björn Jawerth, and Bradley J Lucier. Image compression through wavelet transform coding. *IEEE Transactions on Information Theory*, 38(2):719–746, 1992.

Eleonora Di Nezza, Giampiero Palatucci, and Enrico Valdinoci. Hitchhikers guide to the fractional Sobolev spaces. *Bulletin des Sciences Mathématiques*, 136(5):521–573, 2012.

David L Donoho and Iain M Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432):1200–1224, 1995.

David L Donoho and Iain M Johnstone. Minimax estimation via wavelet shrinkage. *The Annals of Statistics*, 26(3):879–921, 1998.

David L. Donoho, Martin Vetterli, Ronald A. DeVore, and Ingrid Daubechies. Data compression and harmonic analysis. *IEEE Transactions on Information Theory*, 44(6):2435–2476, 1998.

Lawrence C Evans. *Partial Differential Equations*, volume 19. American Mathematical Soc., 2010.

Paul Goldberg and Mark Jerrum. Bounding the Vapnik-Chervonenkis dimension of concept classes parameterized by real numbers. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 361–369, 1993.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016.

Ingo Gühring, Gitta Kutyniok, and Philipp Petersen. Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms. *Analysis and Applications*, 18(05):803–859, 2020.

Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

Boris Hanin. Universal function approximation by deep neural nets with bounded width and ReLU activations. *Mathematics*, 7(10):992, 2019.

Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, pages 2596–2604. PMLR, 2019.

Juncai He, Lin Li, Jinchao Xu, and Chunyue Zheng. ReLU deep neural networks and linear finite elements. *Journal of Computational Mathematics*, 38(3):502–527, 2020.

Jason M Klusowski and Andrew R Barron. Approximation by combinations of ReLU and squared ReLU ridge functions with ℓ^1 and ℓ^0 controls. *IEEE Transactions on Information Theory*, 64(12): 7649–7656, 2018.

Alois Kufner, Oldrich John, and Svatopluk Fucik. *Function Spaces*, volume 3. Springer Science & Business Media, 1977.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

John E Littlewood and Raymond EAC Paley. Theorems on Fourier series and power series. *Journal of the London Mathematical Society*, 1(3):230–233, 1931.

George G Lorentz, Manfred v Golitschek, and Yuly Makovoz. *Constructive Approximation: Advanced Problems*, volume 304. Springer, 1996.

Jianfeng Lu, Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network approximation for smooth functions. *SIAM Journal on Mathematical Analysis*, 53(5):5465–5506, 2021.

Stéphane Mallat. *A Wavelet Tour of Signal Processing*. Elsevier, 1999.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, 2010.

Michael S Paterson. Improved sorting networks with $O(\log N)$ depth. *Algorithmica*, 5(1):75–92, 1990.

Philipp Petersen and Felix Voigtlaender. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Networks*, 108:296–330, 2018.

Pencho P Petrushev. Direct and converse theorems for spline and rational approximation and Besov spaces. In *Function Spaces and Applications: Proceedings of the US-Swedish Seminar held in Lund, Sweden, June 15–21, 1986*, pages 363–377. Springer, 1988.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Norbert Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1): 145–147, 1972.

Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. Bounding and counting linear regions of deep neural networks. In *International Conference on Machine Learning*, pages 4558–4566. PMLR, 2018.

Saharon Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41(1):247–261, 1972.

Zuowei Shen, Haizhao Yang, and Shijun Zhang. Optimal approximation rate of ReLU networks in terms of width and depth. *Journal de Mathématiques Pures et Appliquées*, 157:101–135, 2022.

Zhang Shijun. *Deep neural network approximation via function compositions*. PhD thesis, National University of Singapore (Singapore), 2021.

Jonathan W Siegel and Jinchao Xu. Approximation rates for neural networks with general activation functions. *Neural Networks*, 128:313–321, 2020.

Jonathan W Siegel and Jinchao Xu. High-order approximation rates for shallow neural networks with cosine and ReLU^k activation functions. *Applied and Computational Harmonic Analysis*, 58: 1–26, 2022a.

Jonathan W Siegel and Jinchao Xu. Sharp bounds on the approximation rates, metric entropy, and n-widths of shallow neural networks. *Foundations of Computational Mathematics*, pages 1–57, 2022b.

Matus Telgarsky. Benefits of depth in neural networks. In *Conference on Learning Theory*, pages 1517–1539. PMLR, 2016.

Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of Complexity*, pages 11–30. Springer, 2015.

Shuning Wang and Xusheng Sun. Generalization of hinging hyperplanes. *IEEE Transactions on Information Theory*, 51(12):4425–4431, 2005.

Hugh E Warren. Lower bounds for approximation by nonlinear manifolds. *Transactions of the American Mathematical Society*, 133(1):167–178, 1968.

Hassler Whitney. Analytic extensions of differentiable functions defined in closed sets. *Transactions of the American Mathematical Society*, 36(1):63–89, 1934.

Dmitry Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94: 103–114, 2017.

Dmitry Yarotsky. Optimal approximation of continuous functions by very deep ReLU networks. In *Conference on Learning Theory*, pages 639–649. PMLR, 2018.

Dmitry Yarotsky and Anton Zhevnerchuk. The phase diagram of approximation rates for deep neural networks. *Advances in Neural Information Processing Pystems*, 33:13005–13015, 2020.

Wen Yuan, Winfried Sickel, and Dachun Yang. *Morrey and Campanato Meet Besov, Lizorkin and Triebel*. Springer, 2010.