Neural Network Approximation of Refinable Functions

Ingrid Daubechies, Ronald DeVore, Nadav Dym, Shira Faigenbaum-Golovin, Shahar Z. Kovalsky, Kung-Ching Lin, Josiah Park, Guergana Petrova, Barak Sober [*]

October 27, 2022

Abstract

In the desire to quantify the success of neural networks in deep learning and other applications, there is a great interest in understanding which functions are efficiently approximated by the outputs of neural networks. By now, there exists a variety of results which show that a wide range of functions can be approximated with sometimes surprising accuracy by these outputs. For example, it is known that the set of functions that can be approximated with exponential accuracy (in terms of the number of parameters used) includes, on one hand, very smooth functions such as polynomials and analytic functions and, on the other hand, very rough functions such as the Weierstrass function, which is nowhere differentiable. In this paper, we add to the latter class of rough functions by showing that it also includes refinable functions. Namely, we show that refinable functions are approximated by the outputs of deep ReLU neural networks with a fixed width and increasing depth with accuracy exponential in terms of their number of parameters. Our results apply to functions used in the standard construction of wavelets as well as to functions constructed via subdivision algorithms in Computer Aided Geometric Design.

1 Introduction

Neural Network Approximation (NNA) is concerned with how efficiently a function, or a class of functions, is approximated by the outputs of neural networks. One overview of NNA is given in [9] but there are other noteworthy expositions on this subject such as [14, 22]. The main theme of NNA is to understand for specific functions or classes of functions, how fast the approximation error tends to zero as the number n of parameters of the neural net grows. In this paper, we prove bounds on the rate of NNA for univariate refinable functions (see (1.2)) when using deep networks with *Rectified Linear Unit* (ReLU) activation.

We follow the notation and use the results in [9] for neural networks. In particular, we denote by $\Upsilon^{W,L}(\mathrm{ReLU}; d, N)$ the set of vector valued functions generated by a fully-connected neural network with width W, depth L, input dimension d, output dimension N, and ReLU as the activation function. Since we shall use deep networks for the approximation of univariate functions, we introduce the notation

$$\Sigma_{Cn} := \Sigma_{Cn}(C') := \Upsilon^{C',Cn}(\mathrm{ReLU}; 1, 1), \quad n \geq 1, \tag{1.1}$$

[*]This research was supported by the NSF grants DMS 18-17603 (RD-GP), Tripods Grant CCF-1934904 (RD-KL-JP-GP), ONR Contract N00014-20-1-278 (RD-GP), THEORINET Simons award 814643 (ND).

where C' and C are fixed. We use C, C' to denote generic constants the value of which may change throughout the text black. The set Σ_{Cn} is a nonlinear parametrized set depending on at most $\tilde{C}n$ parameters, where $\tilde{C} = \tilde{C}(C', C)$ depends only on C' and C. The elements in Σ_{Cn} are known to be Continuous Piece-wise Linear (CPwL) functions on $\mathbb{R}$.

We consider a univariate function $\phi : \mathbb{R} \to \mathbb{R}$ which is refinable in the sense that there are constants $c_j \in \mathbb{R}$, $j = 0, \ldots, N$, such that

$$\phi(x) = \sum_{j=0}^{N} c_j \phi(2x - j), \quad x \in \mathbb{R}. \tag{1.2}$$

The sequence $\bar{c} := (c_j)_{j=0}^N$ is called the *mask* of the refinement equation (1.2). Because of (1.2), refinable functions ϕ are self-similar. Note that the functions satisfying (1.2) are not unique since, for example, any multiple of ϕ also satisfies the same equation. However, under very minimal requirements on the mask $\bar{c}$, there is a unique solution to (1.2) up to scaling.

There is by now a vast literature on refinable functions (see for example [2]) which derives various properties of the function ϕ from assumptions on the mask $\bar{c}$.

Our interest in refinable functions arose from the various settings in which these functions appear. The B-splines [8] are refinable functions; subdivision schemes for computer-aided design and computer graphics implicitly use refinable functions for the fast generation of smooth curves and surfaces [2, 12]. The multiresolution analysis framework [18] for the construction of orthonormal [4] or biorthogonal [3] wavelet bases are another noteworthy example: the "scaling functions" (out of which the basic wavelets are built) are refinable functions. Wavelets and wavelet bases are not only fundamental tools for various applications in signal analysis; they are also of particular interest to approximation theory because they provide a natural framework for nonlinear approximation. Additional examples are various fractal-like objects, such as the continuous, nowhere differentiable snowflake curve of von Koch [17], or the curves constructed by de Rham [28, 27], or attractors of hyperbolic iterated function systems on $\mathbb{R}^n$ (see [1]). They occur also in the study of nonhomogeneous Markov chains [16, 23] and in probabilistic automata [21]. In summary, refinable function are the pillars of various applications, and thus studying their NNA will shed light on how neural networks can be utilized in these applications

Our presentation relies heavily on the results and techniques from [6, 7] and to keep it as transparent as possible, we only consider refinable functions that satisfy the two scale relationship (1.2). There are various generalizations of (1.2), including the replacement of the dilation factor 2 by k as well as generalizations of the definition of refinablity to the multivariate settings where the dilation is given by general linear mappings (matrices). Generalizations of the results of the present paper to these broader settings is left to future work.

We next introduce the Banach space $C(\Omega)$ of continuous and bounded functions $f : \Omega \to \mathbb{R}$, defined on an interval $\Omega \subset \mathbb{R}$ (which can be all of $\mathbb{R}$), and the uniform norm

$$\|f\|_{C(\Omega)} := \sup_{x \in \Omega} |f(x)|, \quad f \in C(\Omega). \tag{1.3}$$

We consider the linear operator $V = V_{\bar{c}}$, $V : C(\mathbb{R}) \to C(\mathbb{R})$, given by

$$Vg(x) := \sum_{j=0}^{N} c_j g(2x - j), \quad g \in C(\mathbb{R}), \tag{1.4}$$

and its composition with itself n times

$$V^n g := \underbrace{V \circ V \circ \ldots \circ V}_{n \text{ times}} g. \tag{1.5}$$

The main contribution of our article is described formally in the following theorem.

Theorem 1.1. *Let $\bar{c} = (c_j)_{j=0}^N$ be any refinement mask, let g be any CPwL function which vanishes outside of $[0, N]$, and let V be the linear operator given by (1.4). Then, the function $V^n g$ is in $\Sigma_{Cn}(C') = \Upsilon^{C',Cn}(\mathrm{ReLU}; 1, 1)$ with C, C' depending only on N and the number m of breakpoints of g.*

As a corollary, under certain standard assumptions on $\bar{c}$, we show in Section **??** that a refinable function ϕ can be approximated by the elements of $\Upsilon^{C',Cn}(\mathrm{ReLU}; 1, 1)$ with exponential accuracy. More precisely, we prove that under certain assumptions on the mask $\bar{c}$, the normalized solution ϕ of (1.2) satisfies the inequality

$$E_n(\phi) := \mathrm{dist}(\phi, \Upsilon^{C',Cn}(\mathrm{ReLU}; 1, 1))_{C(\mathbb{R})} \leq \tilde{C}\lambda^n, \quad n = 1, 2, \ldots, \tag{1.6}$$

where $0 < \lambda < 1$ and $\tilde{C}$ depend on the mask.

Besides its implications for approximation of refinable functions, Theorem 1.1 can be seen as an example of a *depth separation* result. We know that the number of breakpoints of $V^n g$ grows exponentially with n (see Remark 2.1) and (see [19]) that the number of breakpoints of any univariate function generated by a neural network with width W and depth L is upper bounded by W^L. Thus, it is clear that any 'shallow network' (network with constant depth) that would generate $V^n g$ must have width which grows exponentially in n. In contrast, the upper bound does not rule out the possibility of 'deep networks' (networks with constant width and depth which grows *linearly in* n) generating $V^n g$. Indeed, Theorem 1.1 shows that $V^n g$ can be expressed by such 'deep networks'. We stress that our result does not follow from the upper bound: while deep neural networks can express some functions which have much more breakpoints than the network's parameters, they most certainly cannot express all such functions (see a proof in [11]). For more examples of depth separation results we refer the reader to [15, 24, 25].

Our main vehicle for proving Theorem 1.1 is the *cascade algorithm* which is used to compute $V^n g$. We describe this algorithm in Section 2. Note that in [6] the term cascade algorithm was used more narrowly to indicate that as a consequence of (1.2), the numerical values of $f(\ell 2^{-j})$ could be computed easily from a few $f(k2^{-j+1})$, where $2k$ is close to ℓ. Now we are using this terminology in a more general sense, including also what in [6] was given the more cumbersome name of *two-scale difference equation.* In essence, while the straightforward algorithm for the computation of $V^n g$ at a point x would require an exponential number of evaluations, the cascade algorithm provides a method to do it with a linear number of evaluations of a discontinuous function. Due to the discontinuous nature of this construction, the cascade algorithm cannot be directly implemented by ReLU NNs. This is the main technical challenge we face in this paper. Our approach for overcoming this difficulty is described in the proofs of the various lemmas that lead to Theorem 1.1, as portrayed in Section 3.

We wish to stress here that some of the lemmas we use in our proofs may be applicable to other settings of NNA. In particular, we draw the reader's attention to the result of §**??** and its utilization, which show that in some special cases we can represent the multiplication of two functions from Σ_{Cn} as a function in Σ_{Cn}.

In Section **??**, we discuss how our main theorem can be combined with the existing theory of refinable functions and the existing convergence results for the cascade algorithm to prove that under standard conditions on $\bar{c}$, the solution ϕ to the refinement equation can be approximated to exponential accuracy by the elements of Σ_{Cn}. Finally, in Section **??**, we discuss how our results put in perspective the approximation properties of deep NN and we present some new results on approximation by deep neural networks.

2 Preliminaries

In this section, we touch upon some of the necessary tools that describe the cascade algorithm as outlined in the works of Daubechies-Lagarias [6, 7].

2.1 The operator V

We consider the action of V on any continuous function g supported on $[0, N]$. It is hard to do a direct analysis of Vg because the points $2x - j$, $j = 0, \dots, N$, are spread out. However, note two important facts. The first is that the points appearing in (1.4) are all equal to $2x$ modulo one. This means that there is at most one point from each interval $[j, j+1]$ and all these points differ by an integer amount. Secondly, since g is supported on $[0, N]$, only the points $2x - j$ that land in $[0, N]$ appear in (1.4). More precisely, the following statement about Vg holds.

Remark 2.1. *If $g : \mathbb{R} \to \mathbb{R}$ is a CPwL function supported on $[0, N]$, then Vg is also a CPwL supported on $[0, N]$. Moreover, each breakpoint ξ' of Vg satisfies $2\xi' = j + \xi$, $j = 0, \dots, N$, where ξ is a breakpoint of g. For example, given a CPwL function g_0 supported on $[0, N]$ with breakpoints at the integers $\{0, \dots, N\}$, $V^n g_0$ has breakpoints at $j/2^n$, $j = 0, 1, \dots, N2^n$.*

Indeed, the fact that Vg is supported on $[0, N]$ follows from the observation that each of the functions $g(2x - j)$, $j = 0, \dots, N$, is supported on $[j/2, (N + j)/2] \subset [0, N]$.

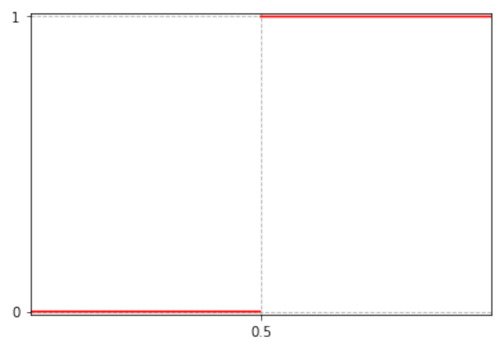
It follows from Remark 2.1 that if g_0 is a CPwL function supported on $[0, N]$ and has breakpoints at the integers $\{0, \dots, N\}$, then $V^n g_0$ is a CPwL supported on $[0, N]$ with breakpoints $j/2^n$, $j = 0, 1, \dots, N2^n$, and as such, $V^n g_0 \in \Upsilon^{3,N2^n}(\mathrm{ReLU}; 1, 1)$ [9]. We shall show that $V^n g_0$ is actually an output of an NN with much smaller depth.

In going forward, we put ourselves in the setting of the works of Daubechies-Lagarias [6, 7], where a better understanding of Vg is facilitated by the introduction of the operator $\mathbf{Vec}$. It assigns to each $g \in C(\mathbb{R})$ a vector valued function $G := \mathbf{Vec}(g)$, where $G := (g_1, \dots, g_N)^T$ with

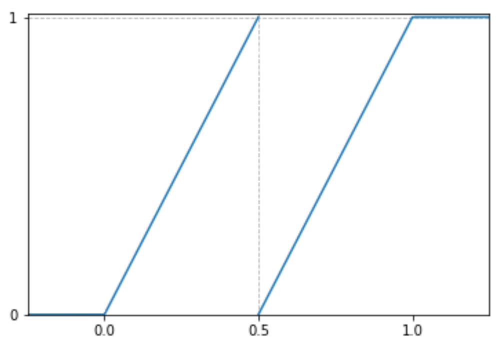
$$g_k(x) := g(x + k - 1), \quad x \in \mathbb{R},\ k = 1, 2, \dots, N. \tag{2.1}$$

Even though $\mathbf{Vec}(g)$ is defined on all of $\mathbb{R}$, we are mainly concerned with its values on $[0, 1]$. Note that the solution ϕ of (1.2) turns out to be supported on $[0, N]$. Therefore, knowing the restriction to $[0, 1]$ of $\mathbf{Vec}(\phi)$ is equivalent to knowing ϕ on its full support. On that interval, g_k is the piece of g with support on $[k - 1, k]$, reparameterized to be supported on $[0, 1]$. Note that

$$g_k(1) = g_{k+1}(0), \quad k = 1, \dots, N - 1. \tag{2.2}$$



(a) Graph of Q.



(b) Graph of R.

Figure 2.1: Graphs of Q and R.

Further, we define for $x \in \mathbb{R}$ and $n \geq 1$

$$G_n(x) := \mathbf{Vec}(V^n g)(x) := (V^n g(x), \dots, V^n g(x+N-1))^T, \tag{2.3}$$

Before describing the cascade algorithm which represents G_n via bit extraction, we recall in the next subsection how we find the binary bits of a number $x \in [0,1]$.

2.2 Binary bits and quantization

Any $x \in [0,1]$, can be represented as

$$x = \sum_{k=1}^{\infty} B_k(x) 2^{-k},$$

where the bits $B_k(x) \in \{0,1\}$. While such a representation of x is not unique, we shall use one particular representation where the bits are found using the quantizer function

$$Q(x) := \chi_{[1/2,1]}(x), \quad x \in [0,1],$$

with χ_I denoting the characteristic function of a set I. The first bit of x and its residual are defined as

$$\begin{array}{rcl} B_1(x) & = & Q(x), \\ R(x) & := & R_1(x) := 2x - B_1(x) \\ & = & 2x - Q(x) \in [0,1], \end{array} \tag{2.4}$$

respectively. The graph of R has two linear pieces, one on $[0,1/2)$ and the other on $[1/2,1]$ and a jump discontinuity at $x = 1/2$. Each linear piece for R has slope 2.

While for the most part we consider $R(x)$ only for $x \in [0,1]$, there are occasions where we need R to be defined for x outside this interval. For such x, we define $R(x) := 0$ when $x \leq 0$ and $R(x) := 1$ when $x \geq 1$. Figure 2.1 shows the graphs of Q and R.

We find the later bits and residuals recursively as

$$\begin{array}{rcl} B_j(x) & = & Q(R_{j-1}(x)), \\ R_j(x) & := & 2R_{j-1}(x) - B_j(x) \\ & = & 2R_{j-1}(x) - Q(R_{j-1}(x)), \quad j = 2,3,\dots . \end{array} \tag{2.5}$$

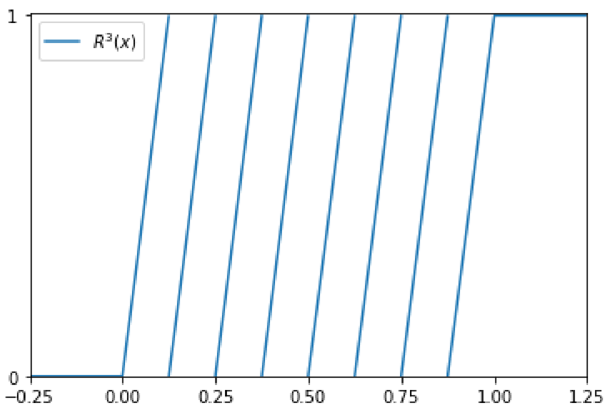


Figure 2.2: Graph of R^3.

Note that on $[0,1]$

$$R^j := \underbrace{R \circ \ldots \circ R}_{j \text{ times}} = R_j, \quad B_{j+1} = B_1 \circ R^j, \quad j = 1, 2, \ldots. \tag{2.6}$$

The R_j's are piecewise linear functions with jump discontinuities at the dyadic integers $k2^{-j}$, $k = 1, 2, \ldots, 2^j - 1$, see for example, Figure 2.2 for the graph of $R^3 = R_3$. Note that, as in the case of R_1, we define $R_j(x) := 0$ for $x \leq 0$ and $R_j(x) := 1$ for $x \geq 1$. Then we will have that $R^j = R_j$ on the whole real line.

2.3 The cascade algorithm

We now look at the computation of $G_1 = ((Vg)_1, \ldots, (Vg)_N)^T$ on $[0,1]$ for general continuous functions g supported on $[0,N]$. Note that for $x \in [0,1]$, and $k = 1, \ldots, N$ we have

$$\begin{aligned} (Vg)_k(x) &:= (Vg)(x+k-1) \\ &= \textstyle\sum_{j=0}^{N} c_j g(2x+2k-2-j), \end{aligned} \tag{2.7}$$

and that we can write

$$\begin{aligned} g(2x+2k-j-2) &= (R(x)+Q(x)+2k-j-2) \\ &= \begin{cases} g_{2k-j-1}(R(x)), & x \in [0,1/2), \\ \\ g_{2k-j}(R(x)), & x \in [1/2,1]. \end{cases} \end{aligned} \tag{2.8}$$

In this way, we get two different formulas depending on whether $x \in [0,1/2)$ or $x \in [1/2,1]$. For example, when $x \in [0,1/2)$, using the fact that $c_k = 0$ if k is not in $\{0,\ldots,N\}$ and that $g_j(x) := g(x+j-1) = 0$ for $x \in [0,1]$ when $j \leq 0$ or $j \geq N+1$, we have for $k = 1, \ldots, N$,

$$
\begin{aligned}
(Vg)_k(x) &= \sum_{j=0}^{N} c_j g_{2k-j-1}(R(x)) \\
&= \sum_{j=2k-N-1}^{2k-1} c_{2k-j-1} g_j(R(x)) \\
&= (T_0 G(R(x)))_k, \quad x \in [0, 1/2),
\end{aligned}
$$

where T_0 is the $N \times N$ matrix with (i, j)-th entry equal to c_{2i-j-1},

$$
T_0 = (c_{2i-j-1})_{ij}, \quad i, j = 1, \ldots, N. \tag{2.9}
$$

A similar derivation gives

$$
(Vg)_k(x) = (T_1 G(R(x)))_k, \quad x \in [1/2, 1], \quad k = 1, \ldots, N,
$$

where now T_1 is the $N \times N$ matrix with (i, j)-th entry equal to c_{2i-j},

$$
T_1 = (c_{2i-j})_{ij}, \quad i, j = 1, \ldots, N. \tag{2.10}
$$

More succinctly, we have for $x \in [0, 1]$

$$
G_1(x) = \mathbf{Vec}(Vg)(x) = T_{Q(x)} G(R(x)) = T_{B_1(x)} G(R(x)). \tag{2.11}
$$

Then, using (2.5) we get

$$
\begin{aligned}
G_2(x) &= \mathbf{Vec}(V(Vg))(x) \\
&= T_{B_1(x)} \mathbf{Vec}(Vg)(R(x)) \\
&= T_{B_1(x)} T_{B_1(R(x))} G(R^2(x)) \\
&= T_{B_1(x)} T_{B_2(x)} G(R^2(x)), \quad x \in [0, 1],
\end{aligned} \tag{2.12}
$$

and if we iterate this computation we get the cascade algorithm. Since $B_1(R^j(x)) = B_{j+1}(x)$, we have for $x \in [0, 1],\ n = 1, 2, \ldots$

$$
G_n(x) = T_{B_1(x)} \cdots T_{B_n(x)} G(R^n(x)). \tag{2.13}
$$

It is useful to keep in mind what $T_{B_n(x)}$ looks like as x traverses $[0, 1]$. It alternately takes the values T_0 and T_1 with the switch coming at the dyadic integers $j2^{-n}$, $1 \leq j < 2^n$.

3 Proof of the main theorem

Before we start proving Theorem 1.1, we observe a simple fact regarding how V^n behaves with respect to the translation operator. This fact, which is described in the following lemma, will help us simplify the proof of Theorem 1.1.

Lemma 3.1. *Let g be a continuous function on $\mathbb{R}$ and for any δ, consider the translated function*

$$\tilde{g}(\cdot) := g(\cdot - \delta).$$

Then, for each $n \geq 1$,

$$V^n(g)(x) = [V^n(\tilde{g})](x + 2^{-n}\delta), \quad x \in \mathbb{R}. \tag{3.1}$$

Proof: Let us first see the action of V on g. Since $g(x) = \tilde{g}(x + \delta)$, we have

$$\begin{aligned} Vg(x) &= \sum_{k=0}^{N} c_k g(2x - k) \\ &= \sum_{k=0}^{N} c_k \tilde{g}(2x - k + \delta) \\ &= \sum_{k=0}^{N} c_k \tilde{g}(2(x + \delta/2) - k) \\ &= [V\tilde{g}](x + \delta/2). \end{aligned} \tag{3.2}$$

This proves the case $n = 1$ in (3.1). We next complete the proof of (3.1) for all $n \geq 1$ by induction. Suppose that we have established the result for a given value of n. Consider the function $h := V^n(g)$. Formula (3.1) says that $\overline{h} := V^n(\tilde{g})$ satisfies

$$\overline{h}(x) = h(x - 2^{-n}\delta), \quad x \in \mathbb{R}. \tag{3.3}$$

So we can apply (3.2) with h in place of g and obtain

$$\begin{aligned} V^{n+1}g(x) &= V(h)(x) \\ &= V(\overline{h})(x + 2^{-n-1}\delta) \\ &= V^{n+1}(\tilde{g})(x + 2^{-n-1}\delta). \end{aligned}$$

This advances the induction and proves (3.1). □

We turn now to discuss the proof of Theorem 1.1. We first show how to prove the theorem when $g = g_0$, where g_0 is any CPwL function that has support in $[0, N]$ and has breakpoints at the integers. We make this assumption on the breakpoints g only for transparency of the proof. We remark later how the same method of proof gives the theorem for arbitrary CPwL functions g.

We represent g_0 as a linear combination of hat functions each of which has a translate with support in $[1/8, 7/8]$. The fact that these functions are supported on this sub-interval will be pivotal in many of the lemmas and theorems below and we will therefore use the following terminology throughout the paper. We call a univariate function g *special* if:

- g is a non-negative CPwL function defined on $\mathbb{R}$.
- the support of $g \subset [1/8, 7/8]$.

Therefore, the translates of the hat functions in the representation of g are special functions and all our results below, proven for special functions, can be applied to these translates.

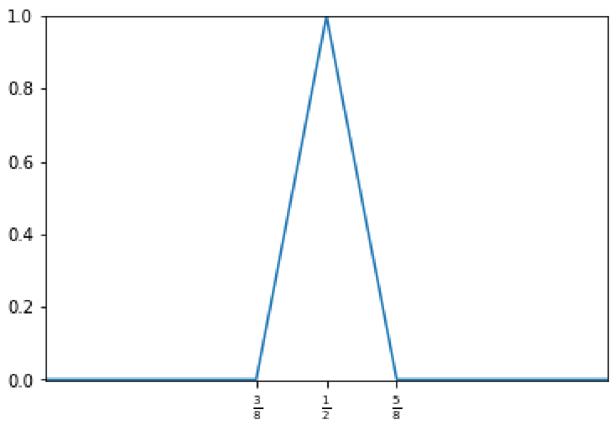


Figure 3.3: The graph of $H := H_4$.

Let H be the 'hat function' with break points at $\{3/8, 1/2, 5/8\}$ and $H(\frac{1}{2}) = 1$, see Figure 3.3. That is, H is given by the formula

$$H(x) = \begin{cases} 0, & x \leq \frac{3}{8},\ x \geq \frac{5}{8}, \\ 8x - 3, & \frac{3}{8} \leq x \leq \frac{1}{2}, \\ -8x + 5, & \frac{1}{2} \leq x \leq \frac{5}{8}. \end{cases} \tag{3.4}$$

Clearly H is a special function with $m = 3$ break points. Although the function under investigation g_0 is not special, it can be written as a linear combination of shifts of the special function H,

$$g_0(x) = \sum_{j=1}^{8N-1} g_0(j/8) H_j(x), \quad H_j(\cdot) := H(\cdot - \frac{j-4}{8}). \tag{3.5}$$

Therefore, from (3.5), Lemma 3.1 and the fact that V^n is a linear operator it follows that

$$\begin{aligned} V^n g_0(x) &= \sum_{j=1}^{8N-1} g_0(j/8) V^n(H(\cdot - \frac{j-4}{8}))(x) \\ &= \sum_{j=1}^{8N-1} g_0(j/8) V^n H(x - 2^{-n-3}(j-4)). \end{aligned} \tag{3.6}$$

Accordingly, the discussion in the following subsections concentrate on special functions. We will come back to finalize the proof of the main theorem in the closing subsection.

3.1 The function $g(R^n)$ is an output of an NN for special functions g

In this section, we shall show that for certain choices of g, the function $g(R^n)$ is in the set $\Upsilon^{C,n+1}(\mathrm{ReLU}; 1, 1)$, where C depends only on the mask $\bar{c}$ and the function g. If g is a special function, then the corresponding vector function G, viewed as a function on $[0, 1]$, is

$$G = \mathbf{Vec}(g) = (g, 0, \ldots, 0)^T.$$

Namely, all its coordinates are zero except the first one, which is the nonzero function g supported on $[1/8, 7/8]$. Therefore $G(R^n(x)) = (g(R^n(x)), 0, \dots, 0)^T$, when considered as a function on $[0, 1]$, and is the zero vector when $R^n(x)$ takes a value outside $[1/8, 7/8]$. Since the formula for R^n is $R^n(x) = 2^n x - k$ on $[k2^{-n}, (k+1)2^{-n})$, $0 \le k < 2^n$, $R^n(x) = 1$, $x \ge 1$, this gives

$$g(R^n(x)) = 0, \quad x \in \Lambda, \tag{3.7}$$

where

$$\Lambda := [0,1] \cap \bigcup_{0 \le j \le 2^n} [j2^{-n} - 2^{-n-3}, j2^{-n} + 2^{-n-3}].$$

In particular, the support of $g(R^n(x))$ is contained in $[2^{-n-3}, 1 - 2^{-n-3}]$.

The first step in our argument to prove that $g(R^n)$ is an output of an NN is to replace the discontinuous functions R^j by the CPwL function $\hat{R}^j := \hat{R}^j_{\alpha,\beta}$. We shall give a family of possible replacements which depend on the choice of two parameters α, β satisfying

$$7/16 < \alpha < \beta < 1/2. \tag{3.8}$$

Definition of $\hat{R}_{\alpha,\beta}$: *We let $\hat{R} := \hat{R}_{\alpha,\beta}$ be the CPwL function defined on $\mathbb{R}$, see Figure* **??***(a), with breakpoints at $\{0, \alpha, \beta, 1/2, 1\}$ which satisfies:*

- $\hat{R}(x) = 0$*, $x \le 0$;*
- $\hat{R}(x) = R(x)$*, for $0 \le x \le \alpha$ and for $x \ge 1/2$;*
- *on $[\alpha, \beta]$, $\hat{R}$ is the linear function that interpolates $R(\alpha)$ at α and interpolates 0 at β;*
- $\hat{R}(x) = 0$ *on $[\beta, 1/2]$.*

In the next lemma, we summarize the properties of $\hat{R}^n := \underbrace{\hat{R} \circ \dots \circ \hat{R}}_{n \text{ times}}$ that we will need in going forward. For its statement, we introduce for $\delta = \delta(n) := 1/2 - \alpha < 2^{-n}$ and the sets

$$E_j := \bigcup_{0<i<2^j} [i2^{-j} - \delta 2^{-j+1}, i2^{-j}], \tag{3.9}$$
$$j = 1, 2, , \dots, n, \tag{3.10}$$
$$E = \bigcup_{j=1}^n E_j, \quad E' = [0,1] \setminus E. \tag{3.11}$$

Clearly, the sets E_j are

$$E_1 = \left[\tfrac{1}{2} - \delta, \tfrac{1}{2}\right], \tag{3.12}$$
$$E_2 = \left[\tfrac{1}{4} - \tfrac{\delta}{2},\right.$$

References

[1] M. Barnsley, *Fractals everywhere.* Academic Press, (2014).

[2] A. Caveretta, W. Dahmen, C. Micchelli, *Stationary Subdivision*, American Math. Society, Providence, 1991.

[3] A. Cohen, I, Daubechies, J. C. Feauveau, *Biorthogonal bases of compactly supported wavelets*, Communications on Pure and Applied Mathematics **45**(1992), 485–560.

[4] I. Daubechies, *Ten Lectures on Wavelets*, SIAM, Philadelphia, (1992).

[5] I. Daubechies, R. DeVore, S. Foucart, B. Hanin, G. Petrova, *Nonlinear approximation and (deep) relu networks*, arXiv:1905.02199, Constructive Approximation, to appear.

[6] I. Daubechies, J. Lagarias, *Two scale difference equations I: existence and global regularity of solutions*, SIAM J. Math. Anal. **22**(1991), 1388–1410.

[7] I. Daubechies, J. Lagarias, *Two scale difference equations II: local regularity, infinite products of matrices and fractals*, SIAM J. Math. Anal. **23**(1) (1992), 1031–1079.

[8] C. De Boor, *A practical guide to splines.* Springer-Verlag, New York, (1978).

[9] R. DeVore, B. Hanin, G. Petrova, *Neural Network Approximation*, Acta Numerica, **30**(2021), 327–444.

[10] R. DeVore, *Nonlinear Approximation*, Acta Numerica, **7**(1998), 50–151.

[11] N. Dym, B. Sober, I. Daubechies, *Expression of fractals through neural network functions*, IEEE Journal on Selected Areas in Information Theory **1**(1) (2020), 57–66.

[12] N. Dyn, *Subdivision schemes in CAGD.*, Advances In Numerical Analysis. **2**(1992), 36–104.

[13] W. E, Q. Wang, *Exponential convergence of the deep neural network approximation for analytic functions*, Sci. China Math., **61** (2018), 1733–1740.

[14] D. Elbrăchter, D. Perekrestenko, P. Grohs, H. Bȯlcskei, *Deep Neural Network Approximation Theory*, IEEE Transactions of Information Theory, **67**(5) (2021), 2581–2623.

[15] R. Eldan, O. Shamir,*The power of depth for feedforward neural networks*, Conference on learning theory, PMLR, (2016) 907-940..

[16] J. Hajnal, M. Bartlett, *Weak ergodicity in non-homogeneous Markov chains.* Mathematical Proceedings Of The Cambridge Philosophical Society. **54**(1958), 233–246.

[17] H. Koch, *Sur une courbe continue sans tangente, obtenue par une construction géométrique élémentaire.* Arkiv For Matematik, Astronomi Och Fysik. **1**(1904), 681–704.

[18] S. Mallat, *Multiresolution approximations and wavelet orthonormal bases of L^2.* Transactions Of The American Mathematical Society. **315**(1989), 69–87.

[19] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, J. Sohl-Dickstein, *On the expressive power of deep neural networks.* International Conference on Machine Learning. PMLR, 2017.

[20] J.A.A. Opschoor, Ch. Schwab, J. Zech , *Exponential ReLU DNN Expression of Holomorphic Maps in High Dimension*, Constructive Approximation (2021), doi.org/10.1007/s00365-021-09542-5.

[21] M. Paterson, *Unsolvability in* 3×3 *matrices*, Studies In Applied Mathematics. **49**(1970), 105–107.

[22] A. Pinkus, *Approximation theory of the mlp model in neural networks*, Acta Numerica **8**(1999), 143–195.

[23] E. Seneta, *On the historical development of the theory of finite inhomogeneous Markov chains.* Mathematical Proceedings Of The Cambridge Philosophical Society. **74**(1973), 507–513.

[24] M. Telgarsky, *Representation benefits of deep feedforward networks* arXiv preprint arXiv:1509.08101. 2015 Sep 27.

[25] M. Telgarsky *Benefits of depth in neural networks* , Conference on Learning Theory, PMLR, (2016), 1517-1539.

[26] D. Yarotsky, *Error bounds for approximations with deep relu networks*, Neural Networks, **94**(2017), 103–114.

[27] G. de Rham *Sur certaines équations fonctionnelles.* L'ouvrage publiéa l'occasion de son centenaire par l'ecole polytechnique de l'université de Lausanne 1853-1953. 1953, 95–97.

[28] G. de Rham *Sur une courbe plane.* Journal de Mathematiques Pares et Appliquees. **35**(1956), 25–42.

Ingrid Daubechies, Department of Mathematics, Duke University, Durham, NC 27705, (3.13)
ingrid.daubechies@duke.edu

Ronald DeVore, Department of Mathematics, Texas A& M University, College Station, TX 77843, (3.14)
rdevore@math.tamu.edu

Nadav Dym, Department of Mathematics, Duke University, Durham, NC 27705, (3.15)
nadav.dym@duke.edu

Shira Faigenbaum-Golovin, Department of Mathematics, Duke University, Durham, NC 27705, (3.16)
alexandra.golovin@duke.edu

Shahar Z. Kovalsky, Department of Mathematics, University of North Carolina, Chapel Hill, NC 27599, (3.17)
shaharko@unc.edu

Kung-Ching Lin , Department of Mathematics, Texas A& M University, College Station, TX 77840, (3.18) kclin@math.tamu.edu

Josiah Park, Department of Mathematics, Texas A& M University, College Station, TX 77840, (3.19)

j.park@math.tamu.edu

Guergana Petrova, Department of Mathematics, Texas A& M University, College Station, TX 77840, (3.20)

gpetrova@math.tamu.edu

Barak Sober, Department of Mathematics, Duke University, Durham, NC 27705, (3.21)

barakino@math.duke.edu