Smooth Function Approximation by Deep Neural Networks with General Activation Functions

Ilsang Ohn and **Yongdai Kim** *

Department of Statistics, Seoul National University, Seoul 08826, Korea
* Correspondence: ydkim0903@gmail.com; Tel.: +82-2-880-9091

Abstract: There has been a growing interest in expressivity of deep neural networks. However, most of the existing work about this topic focuses only on the specific activation function such as ReLU or sigmoid. In this paper, we investigate the approximation ability of deep neural networks with a broad class of activation functions. This class of activation functions includes most of frequently used activation functions. We derive the required depth, width and sparsity of a deep neural network to approximate any Hölder smooth function upto a given approximation error for the large class of activation functions. Based on our approximation error analysis, we derive the minimax optimality of the deep neural network estimators with the general activation functions in both regression and classification problems.

Keywords: function approximation; deep neural networks; activation functions; Hölder continuity; convergence rates

1. Introduction

Neural networks are learning machines motivated by the architecture of the human brain. Neural networks are comprised of multiple hidden layers, and each of the hidden layers has multiple hidden nodes which consist of an affine map of the outputs from the previous layer and a nonlinear map called an activation function. Deep neural networks have been leading tremendous success in various pattern recognition and machine learning tasks such as object recognition, image segmentation, machine translation and others. For an overview on the empirical success of deep neural networks, we refer to the review paper [1] and recent book [2].

Inspired by the success of deep neural networks, many researchers have tried to give theoretical supports for the success of deep neural networks. Much of the work upto date has focused on the expressivity of deep neural networks, i.e., ability to approximate a rich class of functions efficiently. The well-known classical result on this topic is the universal approximation theorem, which states that every continuous function can be approximated arbitrarily well by a neural network [3–7]. But these results do not specify the required numbers of layers and nodes of a neural network to achieve a given approximation accuracy.

Recently, several results about the effects of the numbers of layers and nodes of a deep neural network to its expressivity have been reported. They provide upper bounds of the numbers of layers and nodes required for neural networks to uniformly approximate all functions of interest. Examples of a class of functions include the space of rational functions of polynomials [8], the Hölder space [9–12], Besov and mixed Besov spaces [13] and even a class of discontinuous functions [14,15].

The nonlinear activation function is a central part that makes neural networks differ from the linear models, that is, a neural network becomes a linear function if the linear activation function is used. Therefore, the choice of an activation function substantially influences on the performance and computational efficiency. Numerous activation functions have been suggested to improve neural network learning [16–21]. We refer to the papers [21,22] for an overview of this topic.

There are also many recent theoretical studies about the approximation ability of deep neural networks. However, most of the studies focus on a specific type of the activation function such as ReLU [9,10,13–15], or small classes of activation functions such as sigmoidal functions with additional monotonicity, continuity, and/or boundedness conditions [23–27] and m-admissible functions which are sufficiently smooth and bounded [11]. For definitions of sigmoidal and m-admissible functions, see [24] and [11], respectively. Thus a unified theoretical framework still lacks.

In this paper, we investigate the approximation ability of deep neural networks with a quite general class of activation functions. We derive the required numbers of layers and nodes of a deep neural network to approximate any Hölder smooth function upto a given approximation error for the large class of activation functions. Our specified class of activation functions and the corresponding approximation ability of deep neural networks include most of previous results [9–11,23] as special cases.

Our general theoretical results of the approximation ability of deep neural networks enables us to study statistical properties of deep neural networks. Schmidt-Hieber [10] and Kim et al. [28] proved the minimax optimality of a deep neural network estimator with the ReLU activation function in regression and classification problems, respectively. In this paper, we derive similar results for general activation functions.

This paper is structured as follows. In Section 2, we introduce some notions about deep neural networks. In Section 3, we introduce two large classes of activation functions. In Section 4, we present our main result on the approximation ability of a deep neural network with the general activation function considered in Section 3. In Section 5, we apply the result in Section 4 to the supervised learning problems of regression and classification. Conclusions are given in Section 6. The proofs of all results are given in Appendix.

Notation

We denote by $\mathbb{1}(\cdot)$ the indicator function. Let $\mathbb{R}$ be the set of real numbers and $\mathbb{N}$ be the set of natural numbers. For a real valued vector $\mathbf{x} \equiv (x_1, \ldots, x_d)$, we let $|\mathbf{x}|_0 := \sum_{j=1}^{d} \mathbb{1}(x_j \neq 0), |\mathbf{x}|_p := \left(\sum_{j=1}^{d} |x_j|^p\right)^{1/p}$ for $p \in [1, \infty)$ and $|\mathbf{x}|_\infty := \max_{1 \leq j \leq d} |x_j|$. For simplicity, we let $|\mathbf{x}| := |\mathbf{x}|_1$. For a real valued function $f(x) : \mathbb{R} \to \mathbb{R}$, we let $f'(a), f''(a)$ and $f'''(a)$ are the first, second and third order derivatives of f at a, respectively. We let $f'(a+) := \lim_{\epsilon \downarrow 0}(f(a + \epsilon) - f(a))/\epsilon$ and $f'(a-) := \lim_{\epsilon \downarrow 0}(f(a - \epsilon) - f(a))/\epsilon$. For $x \in \mathbb{R}$, we write $(x)_+ := \max\{x, 0\}$.

2. Deep Neural Networks

In this section we provide a mathematical representation of deep neural networks. A neural network with $L \in \mathbb{N}$ layers, $n_l \in \mathbb{N}$ many nodes at the l-th hidden layer for $l = 1, \ldots, L$, input of dimension n_0, output of dimension n_{L+1} and nonlinear activation function $\sigma : \mathbb{R} \to \mathbb{R}$ is expressed as

$$N_\sigma(\mathbf{x}|\boldsymbol{\theta}) := \mathsf{A}_{L+1} \circ \sigma_L \circ \mathsf{A}_L \circ \cdots \circ \sigma_1 \circ \mathsf{A}_1(\mathbf{x}), \tag{1}$$

where $\mathsf{A}_l : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ is an affine linear map defined by $\mathsf{A}_l(\mathbf{x}) = \mathbf{W}_l \mathbf{x} + \mathbf{b}_l$ for given $n_l \times n_{l-1}$ dimensional weight matrix $\mathbf{W}_l$ and n_l dimensional bias vector $\mathbf{b}_l$ and $\sigma_l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$ is an element-wise nonlinear activation map defined by $\sigma_l(\mathbf{z}) := (\sigma(z_1), \ldots, \sigma(z_{n_l}))^\top$. Here, $\boldsymbol{\theta}$ denotes the set of all weight matrices

and bias vectors $\boldsymbol{\theta} := \left((\mathbf{W}_1, \mathbf{b}_1), (\mathbf{W}_2, \mathbf{b}_2), \dots, (\mathbf{W}_{L+1}, \mathbf{b}_{L+1}) \right)$, which we call $\boldsymbol{\theta}$ the parameter of the neural network, or simply, a network parameter.

We introduce some notations related to the network parameter. For a network parameter $\boldsymbol{\theta}$, we write $L(\boldsymbol{\theta})$ for the number of hidden layers of the corresponding neural network, and write $n_{\max}(\boldsymbol{\theta})$ for the maximum of the numbers of hidden nodes at each layer. Following a standard convention, we say that $L(\boldsymbol{\theta})$ is the depth of the deep neural network and $n_{\max}(\boldsymbol{\theta})$ is the width of the deep neural network. We let $|\boldsymbol{\theta}|_0$ be the number of nonzero elements of $\boldsymbol{\theta}$, i.e.,

$$|\boldsymbol{\theta}|_0 := \sum_{l=1}^{L+1} \left(\left| \text{vec}(\mathbf{W}_l) \right|_0 + |\mathbf{b}_l|_0 \right),$$

where $\text{vec}(\mathbf{W}_l)$ transforms the matrix $\mathbf{W}_l$ into the corresponding vector by concatenating the column vectors. We call $|\boldsymbol{\theta}|_0$ sparsity of the deep neural network. Let $|\boldsymbol{\theta}|_\infty$ be the largest absolute value of elements of $\boldsymbol{\theta}$, i.e.,

$$|\boldsymbol{\theta}|_\infty := \max \left\{ \max_{1 \leq l \leq L+1} \left| \text{vec}(\mathbf{W}_l) \right|_\infty, \max_{1 \leq l \leq L+1} |\mathbf{b}_l|_\infty \right\}.$$

We call $|\boldsymbol{\theta}|_\infty$ magnitude of the deep neural network. We let $\text{in}(\boldsymbol{\theta})$ and $\text{out}(\boldsymbol{\theta})$ be the input and output dimensions of the deep neural network, respectively. We denote by $\Theta_{d,o}(L, N)$ the set of network parameters with depth L, width N, input dimension d and output dimension o, that is,

$$\Theta_{d,o}(L, N) := \left\{ \boldsymbol{\theta} : L(\boldsymbol{\theta}) \leq L, n_{\max}(\boldsymbol{\theta}) \leq N, \text{in}(\boldsymbol{\theta}) = d, \text{out}(\boldsymbol{\theta}) = o \right\}.$$

We further define a subset of $\Theta_{d,o}(L, N)$ with restrictions on sparsity and magnitude as

$$\Theta_{d,o}(L, N, S, B) := \left\{ \boldsymbol{\theta} \in \Theta_{d,o}(L, N) : |\boldsymbol{\theta}|_0 \leq S, |\boldsymbol{\theta}|_\infty \leq B \right\}.$$

3. Classes of Activation Functions

In this section, we consider two classes of activation functions. These two classes include most of commonly used activation functions. Definitions and examples of each class of activation functions are provided in the consecutive two subsections.

3.1. Piecewise Linear Activation Functions

We first consider piecewise linear activation functions.

Definition 1. *A function $\sigma : \mathbb{R} \to \mathbb{R}$ is continuous piecewise linear if it is continuous and there exist a finite number of break points $a_1 \leq a_2 \leq \dots \leq a_K \in \mathbb{R}$ with $K \in \mathbb{N}$ such that $\sigma'(a_k-) \neq \sigma'(a_k+)$ for every $k = 1, \dots, K$ and $\sigma(x)$ is linear on $(-\infty, a_1], [a_1, a_2], \dots, [a_{K-1}, a_K], [a_K, \infty)$.*

Throughout this paper, we write "picewise linear" instead of "continuous picewise linear" for notational simplicity unless there is a confusion. The representative examples of piecewise linear activation functions are as follows:

- ReLU: $\sigma(x) = \max\{x, 0\}$.
- Leaky ReLU: $\sigma(x) = \max\{x, ax\}$ for $a \in (0, 1)$.

The ReLU activation function is the most popular choice in practical applications due to better gradient propagation and efficient computation [22]. In this reason, most of the recent results on the function approximation by deep neural networks are based on the ReLU activation function [9,10,13–15].

In Section 4, as Yarotsky [9] did, we extend these results to any continuous piecewise linear activation function by showing that the ReLU activation function can be exactly represented by a linear combination of piecewise linear activation functions. A formal proof for this argument is presented in Appendix A.1.

3.2. Locally Quadratic Activation Functions

One of the basic building blocks in approximation by deep neural networks is the square function, which should be approximated precisely. Piecewise linear activation functions have zero curvature (i.e., constant first-order derivative) inside each interval divided by its break points, which makes it relatively difficult to approximate the square function efficiently. But if there is an interval on which the activation function has nonzero curvature, the square function can be approximated more efficiently, which is a main motivation of considering a new class of activation functions that we call locally quadratic.

Definition 2. *A function $\sigma : \mathbb{R} \to \mathbb{R}$ is locally quadratic if there exits an open interval $(a, b) \subset \mathbb{R}$ on which σ is three times continuously differentiable with bounded derivatives and there exists $t \in (a, b)$ such that $\sigma'(t) \neq 0$ and $\sigma''(t) \neq 0$.*

We now give examples of locally quadratic activation functions. First of all, any nonlinear smooth activation function with nonzero second derivative, is locally quadratic. Examples are:

- Sigmoid: $\sigma(x) = \dfrac{1}{1 + \mathrm{e}^{-x}}$.
- Tangent hyperbolic: $\sigma(x) = \dfrac{\mathrm{e}^x - \mathrm{e}^{-x}}{\mathrm{e}^x + \mathrm{e}^{-x}}$.
- Inverse square root unit (ISRU) [18]: $\sigma(x) = \dfrac{x}{\sqrt{1 + ax^2}}$ for $a > 0$.
- Soft clipping [19]: $\sigma(x) = \dfrac{1}{a} \log \left(\dfrac{1 + \mathrm{e}^{ax}}{1 + \mathrm{e}^{a(x-1)}} \right)$ for $a > 0$.
- SoftPlus [22]: $\sigma(x) = \log(1 + \mathrm{e}^x)$.
- Swish [21]: $\sigma(x) = \dfrac{x}{1 + \mathrm{e}^{-x}}$.

In addition, piecewise smooth function having nonzero second derivative on at least one piece, is also locally quadratic. Examples are:

- Rectified power unit (RePU) [12]: $\sigma(x) = \max\{x^k, 0\}$ for $k \in \mathbb{N} \setminus \{1\}$.
- Exponential linear unit (ELU) [17]: $\sigma(x) = a(\mathrm{e}^x - 1)\mathbb{1}\,(x \leq 0) + x\mathbb{1}\,(x > 0)$ for $a > 0$.
- Inverse square root linear unit (ISRLU) [18]: $\sigma(x) = \dfrac{x}{\sqrt{1 + ax^2}}\mathbb{1}\,(x \leq 0) + x\mathbb{1}\,(x > 0)$ for $a > 0$.
- Softsign [16]: $\sigma(x) = \dfrac{x}{1 + |x|}$.
- Square nonlinearity [20]:
  $\sigma(x) = \mathbb{1}\,(x > 2) + (x - x^2/4)\mathbb{1}\,(0 \leq x \leq 2) + (x + x^2/4)\mathbb{1}\,(-2 \leq x < 0) - \mathbb{1}\,(x < -2)$.

4. Approximation of Smooth Functions by Deep Neural Networks

In this section we introduce the function class we consider and show the approximation ability of the deep neural networks with a activation function considered in Section 3.

4.1. Hölder Smooth Functions

We recall the definition of Hölder smooth functions. For a d-dimensional multiple index $\mathbf{m} \equiv (m_1, \ldots, m_d) \in \mathbb{N}_0^d$ where $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$, we let $\mathbf{x}^{\mathbf{m}} := x_1^{m_1} \cdots x_d^{m_d}$ for $\mathbf{x} \in \mathbb{R}^d$. For a function $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ denotes the domain of the function, we let $\|f\|_\infty := \sup_{\mathbf{x} \in \mathcal{X}} |f(\mathbf{x})|$. We use notation

$$\partial^{\mathbf{m}} f := \frac{\partial^{|\mathbf{m}|} f}{\partial \mathbf{x}^{\mathbf{m}}} = \frac{\partial^{|\mathbf{m}|} f}{\partial x_1^{m_1} \cdots \partial x_d^{m_d}},$$

for $\mathbf{m} \in \mathbb{N}_0^d$ to denote a derivative of f of order $\mathbf{m}$. We denote by $\mathcal{C}^m(\mathcal{X})$, the space of m times differentiable functions on $\mathcal{X}$ whose partial derivatives of order $\mathbf{m}$ with $|\mathbf{m}| \leq m$ are continuous. We define the Hölder coefficient of order $s \in (0, 1]$ as

$$[f]_s := \sup_{\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}, \mathbf{x}_1 \neq \mathbf{x}_2} \frac{|f(\mathbf{x}_1) - f(\mathbf{x}_2)|}{|\mathbf{x}_1 - \mathbf{x}_2|^s}.$$

For a positive real value α, the Hölder space of order α is defined as

$$\mathcal{H}^\alpha(\mathcal{X}) := \left\{ f \in \mathcal{C}^{\lfloor \alpha \rfloor}(\mathcal{X}) : \|f\|_{\mathcal{H}^\alpha(\mathcal{X})} < \infty \right\},$$

where $\|f\|_{\mathcal{H}^\alpha(\mathcal{X})}$ denotes the Hölder norm defined by

$$\|f\|_{\mathcal{H}^\alpha(\mathcal{X})} := \sum_{\mathbf{m} \in \mathbb{N}_0^d : |\mathbf{m}| \leq \lfloor \alpha \rfloor} \|\partial^{\mathbf{m}} f\|_\infty + \sum_{\mathbf{m} \in \mathbb{N}_0^d : |\mathbf{m}| = \lfloor \alpha \rfloor} [\partial^{\mathbf{m}} f]_{\alpha - \lfloor \alpha \rfloor}.$$

We denote by $\mathcal{H}^{\alpha, R}(\mathcal{X})$ the closed ball in the Hölder space of radius R with respect to the Hölder norm, i.e.,

$$\mathcal{H}^{\alpha, R}(\mathcal{X}) := \left\{ f \in \mathcal{H}^\alpha(\mathcal{X}) : \|f\|_{\mathcal{H}^\alpha(\mathcal{X})} \leq R \right\}.$$

4.2. Approximation of Hölder Smooth Functions

We present our main theorem in this section.

Theorem 1. *Let $d \in \mathbb{N}$, $\alpha > 0$ and $R > 0$. Let the activation function σ be either continuous piecewise linear or locally quadratic. Let $f \in \mathcal{H}^{\alpha, R}([0, 1]^d)$. Then there exist positive constants L_0, N_0, S_0 and B_0 depending only on d, α, R and σ such that, for any $\epsilon > 0$, there is a neural network*

$$\boldsymbol{\theta}_\epsilon \in \Theta_{d,1} \left(L_0 \log(1/\epsilon), N_0 \epsilon^{-d/\alpha}, S_0 \epsilon^{-d/\alpha} \log(1/\epsilon), B_0 \epsilon^{-4(d/\alpha+1)} \right) \tag{2}$$

satisfying

$$\sup_{\mathbf{x} \in [0,1]^d} \left| f(\mathbf{x}) - N_\sigma(\mathbf{x} | \boldsymbol{\theta}_\epsilon) \right| \leq \epsilon. \tag{3}$$

The result of Theorem 1 is equivalent to the results on the approximation by ReLU neural networks [9,10] in a sense that the upper bounds of the depth, width and sparsity are the same orders of those for ReLU, namely, depth $= O(\log(1/\epsilon))$, width $= O(\epsilon^{-d/\alpha})$ and sparsity $= O(\epsilon^{-d/\alpha} \log(1/\epsilon))$. We remark that each upper bound is equivalent to the corresponding lower bound established by [9] up to logarithmic factor.

For piecewise linear activation functions, Yarotsky [9] derived similar results to ours. For locally quadratic activation functions, only special classes of activation functions were considered in the previous work. Li et al. [12] considered the RePU activation function and Bauer and Kohler [11] considered sufficiently smooth and bounded activation functions which include the sigmoid, tangent hyperbolic, ISRU and soft clipping activation functions. However, soft plus, swish, ELU, ISRLU, softsign and square nonlinearity activation functions are new ones only considered in our results.

Even if the orders of the depth, width and sparsity are the same for both both piecewise linear and locally quadratic activation functions, the ways of approximating a smooth function by use of these two activation function classes are quite different. To describe this point, let us provide an outline of the proof. We first consider equally spaced grid points with length $1/M$ inside the d-dimensional unit hypercube $[0, 1]^d$. Let $\mathbb{G}_{d,M}$ be the set of such grid points, namely,

$$\mathbb{G}_{d,M} := \left\{ \frac{1}{M}(m_1, \ldots, m_d) : m_j \in \{0, 1, \ldots, M\}, j = 1, \ldots, d \right\}.$$

For a given Hölder smooth function f of order α, we first find a "local" function for each grid that approximates the target function near the grid point but vanishes at apart from the grid point. To be more specific, we construct the local functions $g_{\mathbf{z}}$, $\mathbf{z} \in \mathbb{G}_{d,M}$ which satisfies:

$$\sup_{x \in [0,1]^d} \left| f(\mathbf{x}) - \sum_{\mathbf{z} \in \mathbb{G}_{d,M}} g_{\mathbf{z},M}(\mathbf{x}) \right| \le C |\mathbb{G}_{d,M}|^{-\alpha/d}, \tag{4}$$

for some universal constant $C > 0$. The inequality (4) implies that the more grid points we used, the more accurate approximation we get. Moreover, the quality of approximation is improved when the target function is more smooth (i.e., large α) and low dimensional (i.e., small d). In fact, $g_{\mathbf{z},M}(\mathbf{x})$ is given by a product of the Taylor polynomial $P_{\mathbf{z},M}(\mathbf{x}) := \sum_{\mathbf{m} \in \mathbb{N}_0^d : |\mathbf{m}| \le \alpha} (\partial^{\mathbf{m}} f)(\mathbf{z}) \frac{(\mathbf{x} - \mathbf{z})^{\mathbf{m}}}{\mathbf{m}!}$ at $\mathbf{z}$ and the local basis function $\phi_{\mathbf{z},M}(\mathbf{x}) := \prod_{j=1}^d (1/M - |x_j - z_j|)_+$, where $\mathbf{m}! := \prod_{j=1}^d m_j!$. By simple algebra, we have

$$P_M(\mathbf{x}) := \sum_{\mathbf{z} \in \mathbb{G}_{d,M}} g_{\mathbf{z},M}(\mathbf{x}) := \sum_{\mathbf{z} \in \mathbb{G}_{d,M}} P_{\mathbf{z},M}(\mathbf{x}) \phi_{\mathbf{z},M}(\mathbf{x})$$

$$= \sum_{\mathbf{z} \in \mathbb{G}_{d,M}} \sum_{\mathbf{m} : |\mathbf{m}| \le \alpha} \beta_{\mathbf{z},\mathbf{m}} \mathbf{x}^{\mathbf{m}} \phi_{\mathbf{z},M}(\mathbf{x}),$$

where $\beta_{\mathbf{z},\mathbf{m}} := \sum_{\tilde{\mathbf{m}} : \tilde{\mathbf{m}} \ge \mathbf{m}, |\tilde{\mathbf{m}}| \le \alpha} (\partial^{\tilde{\mathbf{m}}} f)(\mathbf{z}) \frac{(-\mathbf{z})^{\tilde{\mathbf{m}} - \mathbf{m}}}{\mathbf{m}!(\tilde{\mathbf{m}} - \mathbf{m})!}$.

The second stage is to approximate each monomial $\mathbf{x}^{\mathbf{m}}$ and each local basis function $\phi_{\mathbf{z},M}(\mathbf{x})$ by deep neural networks. Each monomial can be approximated more efficiently by a deep neural network with a locally quadratic activation function than a piecewise linear activation function since each monomial has nonzero curvature. On the other hand, the local basis function can be approximated more efficiently by a deep neural network with a piecewise linear activation than a locally quadratic activation function since the local basis function is piecewise linear itself. That is, there is a trade-off in using either a piecewise linear or a locally quadratic activation function.

We close this section by giving a comparison of our result to the approximation error analysis of [11]. Bauer and Kohler [11] studies approximation of the Hölder smooth function of order α by a two layer neural network with m-admissible activation functions with $m \ge \alpha$, where a function σ is called m-admissible if (1) σ is at least $m + 1$ times continuously differentiable with bounded derivatives; (2) a point $t \in \mathbb{R}$ exists, where all derivatives up to the order m of σ are different from zero; and (3) $|\sigma(x) - 1| \le 1/x$ for $x > 0$ and

$|\sigma(x)| \leq 1/|x|$ for $x < 0$. Our notion of locally quadratic activation functions is a generalized version of the m-admissibility.

In the proof of [11], the condition $m \geq \alpha$ is necessary because they approximate any monomial of order $\mathbf{m}$ with $|\mathbf{m}| \leq \alpha$ with a two layer neural network, which is impossible when $m < \alpha$. We drop the condition $m \geq \alpha$ by showing that any monomial of order $\mathbf{m}$ with $|\mathbf{m}| \leq \alpha$ can be approximated by deep neural network with a finite number of layers, which depends on α.

5. Application to Statistical Learning Theory

In this section, we apply our results about the approximation error of neural networks to the supervised learning problems of regression and classification. Let $\mathcal{X}$ be the input space and $\mathcal{Y}$ the output space. Let $\mathcal{F}$ be a given class of measurable functions from $\mathcal{X}$ to $\mathcal{Y}$. Let P_0 be the true but unknown data generating distribution on $\mathcal{X} \times \mathcal{Y}$. The aim of supervised learning is to find a predictive function that minimizes the population risk $\mathcal{R}(f) := \mathsf{E}_{(\mathbf{X},Y) \sim \mathsf{P}_0} \ell(Y, f(\mathbf{X}))$ with respect to a given loss function ℓ. Since P_0 is unknown, we cannot directly minimize the population risk, and thus any estimator $\hat{f}$ inevitably has the excess risk which is defined as $\mathcal{R}(\hat{f}) - \inf_{f \in \mathcal{F}} \mathcal{R}(f)$. For a given sample of size n, let $\mathcal{F}_n$ be a given subset of $\mathcal{F}$ called a sieve and let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ be observed (training) data of input–output pairs assumed to be independent realizations of $(\mathbf{X}, Y)$ following P_0. Let $\hat{f}_n$ be an estimated function among $\mathcal{F}_n$ based on the training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$. The excess risk of $\hat{f}_n$ is decomposed to approximation and estimation errors as

$$\mathcal{R}(\hat{f}_n) - \inf_{f \in \mathcal{F}} \mathcal{R}(f) = \underbrace{\left[\mathcal{R}(\hat{f}_n) - \inf_{f \in \mathcal{F}_n} \mathcal{R}(f) \right]}_{\text{Estimation error}} + \underbrace{\left[\inf_{f \in \mathcal{F}_n} \mathcal{R}(f) - \inf_{f \in \mathcal{F}} \mathcal{R}(f) \right]}_{\text{Approximation error}}. \tag{5}$$

There is a trade-off between approximation and estimation errors. If the function class $\mathcal{F}_n$ is sufficiently large to approximate the optimal estimator $f^* := \operatorname{argmin}_{f \in \mathcal{F}} \mathcal{R}(f)$ well, then the estimation error becomes large due to high variance. In contrast, if $\mathcal{F}_n$ is small, it leads to low estimation error but it suffers from large approximation error.

One of the advantages of deep neural networks is that we can construct a sieve which has good approximation ability as well as low complexity. Schmidt-Hieber [10] and Kim et al. [28] proved that a neural network estimator can achieve the optimal balance between the approximation and estimation errors to obtain the minimax optimal convergence rates in regression and classification problems, respectively. But they only considered the ReLU activation function. Based on the results of Theorem 1, we can easily extend their results to general activation functions.

The main tool to derive the minimax optimal convergence rate is that the complexity of a class of functions generated by a deep neural network is not affected much by a choice of an activation function, provided that the activation function is Lipschitz continuous. The function $\sigma : \mathbb{R} \to \mathbb{R}$ is Lipschitz continuous if there is a constant $C_\sigma > 0$ such that

$$|\sigma(x_1) - \sigma(x_2)| \leq C_\sigma |x_1 - x_2|, \tag{6}$$

for any $x_1, x_2 \in \mathbb{R}$. Here, C_σ is called the Lipschitz constant. We use the covering number with respect to the L_∞ norm $\|\cdot\|_\infty$ as a measure of complexity of function classes. We recall the definition of the covering number. Let $\mathcal{F}$ be a given class of real-valued functions defined on $\mathcal{X}$. Let $\delta > 0$. A collection $\{f_j \in \mathcal{F} : j = 1, \ldots, J\}$ is called a δ-covering set of $\mathcal{F}$ with respect to the L_∞ norm if for all $f \in \mathcal{F}$, there

exists f_j in the collection such that $\|f - f_j\|_\infty \leq \delta$. The cardinality of the minimal δ-covering set is called the δ-covering number of $\mathcal{F}$ with respect to the L_∞ norm which is denoted by $\mathcal{N}(\delta, \mathcal{F}, \|\cdot\|_\infty)$. That is,

$$\mathcal{N}(\delta, \mathcal{F}, \|\cdot\|_\infty) := \inf \left\{ J \in \mathbb{N} : \exists f_1, \ldots, f_J \text{ such that } \mathcal{F} \subset \bigcup_{j=1}^{J} B_\infty(f_j, \delta) \right\},$$

where $B_\infty(f_j, \delta) := \{f \in \mathcal{F} : \|f - f_j\|_\infty \leq \delta\}$. The following proposition provides the covering number of a class of functions generated by neural networks.

Proposition 1. *Assume that the activation function σ is Lipschitz continuous with the Lipschitz constant C_σ. Consider a class of functions generated by a deep neural network*

$$\mathcal{F}_{d,1}(L, N, S, B) := \left\{ N_\sigma(\cdot|\boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta_{d,1}(L, N, S, B) \right\}.$$

For any $\delta > 0$,

$$\log \mathcal{N}\left(\delta, \mathcal{F}_{d,1}(L, N, S, B), \|\cdot\|_\infty\right) \leq 2L(S+1) \log\left(\delta^{-1} C_\sigma L(N+1)(B \vee 1)\right), \tag{7}$$

where $B \vee 1 := \max\{B, 1\}$.

The result in Proposition 1 is very similar to the existing results in literature, e.g., Theorem 14.5 of [29], Lemma 5 of [10] and Lemma 3 of [13]. We employ similar techniques used in [10,13,29] to obtain the version presented here. We give the proof of this proposition in Appendix B.

All of the activation functions considered in Section 3 except RePU satisfy the Lipschitz condition (6) and hence Proposition 1 can be applied. An interesting implication of Proposition 1 is that the complexity of the function class generated by deep neural networks is not affected by the choice of an activation function. Hence, the remaining step to derive the convergence rate of a neural network estimator is that approximation accuracies by various activation functions are the same as that of the ReLU neural network.

5.1. Application to Regression

First we consider the regression problem. For simplicity, we let $\mathcal{X} = [0,1]^d$. Suppose that the generated model is $Y|\mathbf{X} = \mathbf{x} \sim \mathsf{N}(f_0(\mathbf{x}), 1)$ for some $f_0 : [0,1]^d \to \mathbb{R}$. The performance of an estimator is measured by the L_2 risk $\mathcal{R}_{2,f_0}(f)$, which is defined by

$$\mathcal{R}_{2,f_0}(f) := \mathsf{E}_{f_0, \mathsf{P}_{\mathbf{x}}}(Y - f(\mathbf{X}))^2 := \mathsf{E}_{Y|\mathbf{X} \sim \mathsf{N}(f_0(\mathbf{X}),1), \mathbf{X} \sim \mathsf{P}_{\mathbf{x}}}(Y - f(\mathbf{X}))^2,$$

where $\mathsf{P}_{\mathbf{x}}$ is the marginal distribution of $\mathbf{X}$. The following theorem proves that the optimal convergence rate is obtained by the deep neural network estimator of the regression function f_0 for a general activation function.

Theorem 2. *Suppose that the activation function σ is either piecewise linear or locally quadratic satisfying the Lipschitz condition (6). Then there are universal positive constants L_0, N_0, S_0 and B_0 such that the deep neural network estimator obtained by*

$$\hat{f}_n \in \operatorname*{argmin}_{f \in \mathcal{F}_{\sigma,n}} \sum_{i=1}^{n} \left(y_i - f(\mathbf{x}_i)\right)^2,$$

with

$$\mathcal{F}_{\sigma,n} := \left\{ N_\sigma(\cdot|\boldsymbol{\theta}) : \left\| N_\sigma(\cdot|\boldsymbol{\theta}) \right\|_\infty \leq 2R, \boldsymbol{\theta} \in \Theta_{d,1} \left(L_0 \log n, N_0 n^{\frac{d}{2\alpha+d}}, S_0 n^{\frac{d}{2\alpha+d}} \log n, B_0 n^\kappa \right) \right\}$$

for some $\kappa > 0$ *satisfies*

$$\sup_{f_0 \in \mathcal{H}^{\alpha,R}([0,1]^d)} \mathsf{E}\left[\mathcal{R}_{2,f_0}(\hat{f}_n) - \inf_{f \in \mathcal{F}} \mathcal{R}_{2,f_0}(f) \right] \leq C n^{-\frac{2\alpha}{2\alpha+d}} \log^3 n,$$

for some universal constant $C > 0$, *where the expectation is taken over the training data.*

5.2. Application to Binary Classification

The aim of the binary classification is to find a classifier that predicts the label $y \in \{-1, 1\}$ for any input $\mathbf{x} \in [0,1]^d$. An usual assumption on the data generating process is that $Y|\mathbf{X} = \mathbf{x} \sim 2\mathsf{Bern}(\eta(\mathbf{x})) - 1$ for some $\eta : [0,1]^d \to [0,1]$, where $\mathsf{Bern}(p)$ denotes the Bernoulli distribution with parameter p. Note that $\eta(\mathbf{x})$ is the conditional probability function $P_0(Y = 1|\mathbf{X} = \mathbf{x})$. A common approach is, instead of finding a classifier directly, to construct a real valued function f, a so-called classification function, and predict the label y based on the sign of $f(\mathbf{x})$. The performance of a classification function is measured by the misclassification error $\mathcal{R}_{01,\eta}(f)$, which is defined by

$$\mathcal{R}_{01,\eta}(f) := \mathsf{E}_{\eta, P_\mathbf{x}} \mathbb{1}\left(Yf(\mathbf{X}) < 0\right) := \mathsf{E}_{Y|\mathbf{X} \sim 2\mathsf{Bern}(\eta(\mathbf{X})) - 1, \mathbf{X} \sim P_\mathbf{x}} \mathbb{1}\left(Yf(\mathbf{X}) < 0\right).$$

It is well known that the convergence rate of the excess risk for classification is faster than that of regression when the conditional probability function $\eta(\mathbf{x})$ satisfies the following condition: there is a constant $q \in [0, \infty]$ such that for any sufficiently small $u > 0$, we have

$$P_\mathbf{x} \left(|\eta(\mathbf{X}) - 1/2| < u \right) \leq u^q. \tag{8}$$

This condition is called the Tsybakov noise condition and q is called the noise exponent [30,31]. When q is larger, the classification task is easier since the probability of generating vague samples become smaller. The following theorem proves that the optimal convergence rate can be obtained by the deep neural network estimator with an activation function considered in Section 3. As is done by [28], we consider the hinge loss $\ell_{\mathsf{hinge}}(z) := \max\{1 - z, 0\}$.

Theorem 3. *Assume the Tsybakov noise condition* (8) *with the noise exponent* $q \in [0, \infty]$. *Suppose that the activation function* σ, *which is either piecewise linear or locally quadratic satisfying the Lipschitz condition* (6), *is used for all hidden layers except the last one and the ReLU activation function is used for the last hidden layer. Then there are universal positive constants* L_0, N_0, S_0 *and* B_0 *such that the deep neural network estimator obtained by*

$$\hat{f}_n \in \operatorname*{argmin}_{f \in \mathcal{F}_{\sigma,n}} \sum_{i=1}^n \ell_{\mathsf{hinge}}(y_i f(\mathbf{x}_i)),$$

with

$$\mathcal{F}_{\sigma,n} := \left\{ N_\sigma(\cdot|\boldsymbol{\theta}) : \left\| N_\sigma(\cdot|\boldsymbol{\theta}) \right\|_\infty \leq 1, \boldsymbol{\theta} \in \Theta_{d,1} \left(L_0 \log n, N_0 n^\nu \log^{-3\nu} n, S_0 n^\nu \log^{-3\nu+1} n, B_0 n^\kappa \right) \right\},$$

for $\nu := d / \{\alpha(q+2)+d\}$ and some $\kappa > 0$ satisfies

$$\sup_{\eta \in \mathcal{H}^{\alpha,R}([0,1]^d)} \mathsf{E}\left[\mathcal{R}_{01,\eta}(\hat{f}_n) - \inf_{f \in \mathcal{F}} \mathcal{R}_{01,\eta}(f)\right] \le C \left(\frac{\log^3 n}{n}\right)^{\frac{\alpha(q+1)}{\alpha(q+2)+d}},$$

for some universal constant $C > 0$, where the expectation is taken over the training data.

Note that the Bayes classifier $f^* := \mathrm{argmin}_{f \in \mathcal{F}} \mathcal{R}_{01,\eta}(f)$ is given by

$$f^*(\mathbf{x}) = 2\mathbb{1}\left(2\eta(\mathbf{x}) - 1 \ge 0\right) - 1,$$

which is an indicator function. Since a neural network with the ReLU activation function can approximate indicator functions well [14,15,28], we use the ReLU activation function in the last layer in order to approximate the Bayes classifier more precisely and thus to achieve the optimal convergence rate.

6. Conclusions

In this study, we established the upper bounds of the required depth, width and sparsity of deep neural networks to approximate any Hölder continuous function for the general classes of activation functions. These classes of activation functions include most of the popularly used activation functions. The derived upper bounds of the depth, width and sparsity are optimal in a sense that they are equivalent to the lower bounds up to logarithmic factors. We used this generalization of the approximation error analysis to extend the statistical optimality of the deep neural network estimator in regression and classification problems, where the activation function is other than the ReLU.

Our construction of neural networks for approximation reveals that the piecewise linear activation functions are more efficient in approximating local basis functions while locally quadratic activation functions are more efficient in approximating polynomials. Hence if the activation function has both piecewise linear region and locally quadratic region, we could have a better approximation result. We leave the development of such activation functions as a future work.

Appendix A. Proof of Theorem 1

Appendix A.1. Proof of Theorem 1 for Piecewise Linear Activation Functions

The main idea of the proof is that any deep neural network with the ReLU activation function can be exactly reconstructed by a neural network with a piecewise activation function whose proof is in the next lemma that is a slight modification of Proposition 1 (b) of [9].

Lemma A1. *Let σ be an any continuous peicewise linear activation function, and ρ be the ReLU activation function. Let $\boldsymbol{\theta} \in \Theta_{d,1}(L, N, S, B)$. Then there exists $\boldsymbol{\theta}^* \in \Theta_{d,1}(L, 2N, 4S + 2LN + 1, C_1 B)$ such that*

$$\sup_{\mathbf{x} \in [0,1]^d} \left|N_\sigma(\mathbf{x}|\boldsymbol{\theta}^*) - N_\rho(\mathbf{x}|\boldsymbol{\theta})\right| = 0,$$

where $C_1 > 0$ is a constant depending on the activation function σ.

Proof. Let a be any break point of σ. Note that $\sigma(a-) \neq \sigma(a+)$. Let r_0 be the distance between a and the closest other break point. Then σ is linear on $[a - r_0, a]$ and $[a, a + r_0]$. Then for any $r > 0$, the ReLU activation function $\rho(x) := (x)_+$ is expressed as

$$
\rho(x) = \frac{\sigma\left(a + \frac{r_0}{2r}x\right) - \sigma\left(a - \frac{r_0}{2} + \frac{r_0}{2r}x\right) - \sigma(a) + \sigma\left(a - \frac{r_0}{2}\right)}{(\sigma'(a+) - \sigma'(a-))\frac{r_0}{2r}}
$$
$$
=: u_1\sigma\left(a + \frac{r_0}{2r}x\right) + u_2\sigma\left(a - \frac{r_0}{2} + \frac{r_0}{2r}x\right) + v
$$
(A1)

for any $x \in [-r, r]$, where we define $u_1 := 1/\left((\sigma'(a+) - \sigma'(a-))\frac{r_0}{2r}\right)$, $u_2 := -1/\left((\sigma'(a+) - \sigma'(a-))\frac{r_0}{2r}\right)$ and $v := (-\sigma(a) + \sigma(a - r_0/2))/\left((\sigma'(a+) - \sigma'(a-))\frac{r_0}{2r}\right)$.

Let $\boldsymbol{\theta} \equiv ((\mathbf{W}_1, \mathbf{b}_1), \dots, (\mathbf{W}_{L+1}, \mathbf{b}_{L+1})) \in \Theta_{d,1}(L, N, S, B)$ be given. Since both input $\mathbf{x} \in [0, 1]^d$ and the network parameter $\boldsymbol{\theta}$ are bounded, we can take a sufficiently large r so that Equation (A1) holds for any hidden nodes of the network $\boldsymbol{\theta}$. We consider the deep neural network $\boldsymbol{\theta}^* \equiv ((\mathbf{W}_1^*, \mathbf{b}_1^*), \dots, (\mathbf{W}_{L+1}^*, \mathbf{b}_{L+1}^*)) \in \Theta_{d,1}(L, 2N)$, where we set

$$
\mathbf{W}_l^* := \frac{r_0}{2r}\begin{pmatrix} u_1\mathbf{W}_l & u_2\mathbf{W}_l \\ u_1\mathbf{W}_l & u_2\mathbf{W}_l \end{pmatrix} \in \mathbb{R}^{2n_l \times 2n_{l-1}},
$$

$$
\mathbf{b}_l^* := \begin{pmatrix} a\mathbf{1}_{n_l} + \frac{r_0}{2r}(v\mathbf{W}_l\mathbf{1}_{n_{l-1}} + \mathbf{b}_l) \\ \left(a - \frac{r_0}{2}\right)\mathbf{1}_{n_l} + + \frac{r_0}{2r}(v\mathbf{W}_l\mathbf{1}_{n_{l-1}} + \mathbf{b}_l) \end{pmatrix} \in \mathbb{R}^{2n_l},
$$

for $l = 1, \dots, L$ and

$$
\mathbf{W}_{L+1}^* := \begin{pmatrix} u_1\mathbf{W}_{L+1} & u_2\mathbf{W}_{L+1} \end{pmatrix}, \quad \mathbf{b}_{L+1}^* := v.
$$

Here, $\mathbf{1}_n$ denotes the n-dimensional vector of $1's$. Then by Equation (A1) and some algebra, we have that $N_\sigma(\mathbf{x}|\boldsymbol{\theta}^*) = N_\rho(\mathbf{x}|\boldsymbol{\theta})$ for any $\mathbf{x} \in [0, 1]^d$. For the sparsity of $\boldsymbol{\theta}^*$, we note that

$$
\left|\text{vec}(\mathbf{W}_l^*)\right|_0 + \left|\mathbf{b}_l^*\right|_0 \leq 4\left|\text{vec}(\mathbf{W}_l)\right|_0 + 2n_l
$$

which implies that $|\boldsymbol{\theta}^*|_0 \leq 4|\boldsymbol{\theta}|_0 + 2L(\boldsymbol{\theta})n_{\max}(\boldsymbol{\theta}) + 1$. $\quad\square$

Thanks to Lemma A1, to prove Theorem 1 for piecewise linear activation functions, it suffices to show the approximation ability of the ReLU networks, which is already done by [10] as in the next lemma.

Lemma A2 (Theorem 5 of [10]). *Let ρ be the ReLU activation function. For any $f \in \mathcal{H}^{\alpha,R}([0,1]^d)$ and any integers $m \geq 1$ and $M \geq \max\left\{(\alpha+1)^d, (R+1)e^d\right\}$, there exists a network parameter $\boldsymbol{\theta} \in \Theta_{d,1}(L, N, S, 1)$ such that*

$$
\sup_{\mathbf{x} \in [0,1]^d} \left|N_\rho(\mathbf{x}|\boldsymbol{\theta}) - f(\mathbf{x})\right| \leq (2R+1)(1 + d^2 + \alpha^2)6^d M2^{-m} + R3^\alpha M^{-\alpha/d},
$$
(A2)

where $L = 8 + (m+5)(1 + \lceil \log_2(d \vee \alpha) \rceil)$, $N = 6(d + \lceil\alpha\rceil)M$, and $S = 141(d + \alpha + 1)^{3+d}M(m+6)$.

Theorem 1 for piecewise linear activation functions is a direct consequence of Lemmas A1 and A2, which is summarized as follows.

Proof of Theorem 1 for piecewise linear activation functions. Let ρ be the ReLU activation function. By letting $M = 3^d(2R)^{d/\alpha}\epsilon^{-d/\alpha}$ and $m = \log_2\left(2(2R+1)(1+d^2+\alpha^2)18^d(2R)^{d/\alpha}\epsilon^{-d/\alpha-1}\right)$, Lemma A2 implies that there exists a network parameter θ' such that $\sup_{\mathbf{x}\in[0,1]^d}|N_\rho(\mathbf{x}|\theta') - f(\mathbf{x})| \le \epsilon$ with $L(\theta') \le L'_0\log(1/\epsilon)$, $n_{\max}(\theta') \le N'_0\epsilon^{-d/\alpha}$ and $|\theta'|_0 \le S'_0\epsilon^{-d/\alpha}\log(1/\epsilon)$ for some positive constants L'_0, N'_0, and S'_0 depending only on α, d and R. Hence by Lemma A1, there is a network parameter θ producing the same output of the ReLU neural network $N_\rho(\cdot|\theta)$ with $L(\theta) = L(\theta')$, $n_{\max}(\theta) = 2n_{\max}(\theta')$, $|\theta|_0 \le 4|\theta'|_0 + 2L(\theta')n_{\max}(\theta') + 1 \le S_0\epsilon^{-d/\alpha}\log(1/\epsilon)$ and $|\theta|_\infty \le B_0|\theta'|_\infty$ for some $S_0 > 0$ depending only on α, d, R and σ, and some $B_0 > 0$ depending only on σ, which completes the proof. $\square$

Appendix A.2. Proof of Theorem 1 for Locally Quadratic Activation Functions

Lemma A3. *Assume that an activation function σ is locally quadratic. There is a constant K_0 depending only on the activation function such that for any $K > K_0$ the following results hold.*

(a) *There is a neural network $\theta_2 \in \Theta_{1,1}(1,3)$ with $|\theta_2|_\infty \le K^2$ such that*

$$\sup_{x\in[-1,1]}\left|N_\sigma(x|\theta_2) - x^2\right| \le \frac{C_1}{K},$$

where $C_1 > 0$ is a constant depending only on σ.

(b) *Let $A > 0$. There is a neural network parameter $\theta_{\times,A} \in \Theta_{2,1}(1,9)$ with $|\theta_{\times,A}|_\infty \le \max\{K^2, 2A^2\}$ such that*

$$\sup_{\mathbf{x}\in[-A,A]^2}\left|N_\sigma(\mathbf{x}|\theta_{\times,A}) - x_1x_2\right| \le \frac{6A^2C_1}{K}.$$

(c) *Let α be a positive integer. For any multi-index $\mathbf{m} \in \mathbb{N}_0^d$ with $|\mathbf{m}| \le \alpha$, there is a network parameter $\theta_{\mathbf{m}} \in \Theta_{d,1}(\lceil\log_2\alpha\rceil, 9\alpha)$ with $|\theta_{\mathbf{m}}|_\infty \le \max\{K^2, C_2\}$ such that*

$$\sup_{\mathbf{x}\in[0,1]^d}\left|N_\sigma(\mathbf{x}|\theta_{\mathbf{m}}) - \mathbf{x}^{\mathbf{m}}\right| \le \frac{C_3}{K},$$

for some positive constants C_2 and C_3 depending only on σ and α.

(d) *There is a network parameter $\theta_{1/2} \in \Theta_{1,1}(\lceil\log K\rceil, 15)$ with $|\theta_{1/2}|_\infty \le \max\{K^2, C_4\}$ such that*

$$\sup_{x\in[0,2]}\left|N_\sigma(x|\theta_{1/2}) - \sqrt{x}\right| \le C_5\frac{\log K}{K}$$

for some positive constants C_4 and C_5 depending only on σ.

(e) *There is a network parameter $\theta_{\mathrm{abs}} \in \Theta_{1,1}(\lceil\log K\rceil, 15)$ with $|\theta_{\mathrm{abs}}|_\infty \le \max\{K^2, C_6\}$ such that*

$$\sup_{x\in[-1,1]}\left|N_\sigma(x|\theta_{\mathrm{abs}}) - |x|\right| \le \frac{C_7}{\sqrt{K}},$$

for some positive constants C_6 and C_7 depending only on σ.

Proof. Recall that there is an interval (a,b) on which $\sigma(x)$ is three times continuously differentiable with bounded derivatives and there is $t \in (a,b)$ such that $\sigma'(t) \ne 0$ and $\sigma''(t) \ne 0$

Proof of (a). Take K large so that $2/K < \min\{|t-b|, |t-a|\}$. Consider a neural network

$$N_\sigma(x|\boldsymbol{\theta}_2) := \sum_{k=0}^{2}(-1)^{k-1}\frac{K^2}{\sigma''(t)}\binom{2}{k}\sigma\left(\frac{k}{K}x + t\right). \tag{A3}$$

Since σ is three times continuously differentiable on (a,b) and $(k-1)x/K + t \in (a,b)$ if $x \in [0,1]$, it can be expanded in the Taylor series with Lagrange remainder around t to have

$$
\begin{aligned}
N_\sigma(x|\boldsymbol{\theta}_2) &= \frac{K^2}{\sigma''(t)}\sum_{k=0}^{2}(-1)^{k}\binom{2}{k}\left\{\sigma(t) + \sigma'(t)\frac{kx}{K} + \frac{\sigma''(t)}{2}\frac{(kx)^2}{K^2} + \frac{\sigma''(\xi_k)}{6}\frac{(kx)^3}{K^3}\right\} \\
&= \frac{K^2}{\sigma''(t)}\left\{\sigma''(t)\frac{x^2}{K^2} + \sum_{k=1}^{2}(-1)^{k}\binom{2}{k}\frac{\sigma'''(\xi_k)}{6}\frac{(kx)^3}{K^3}\right\} \\
&= x^2 + \frac{x^3}{6K\sigma''(t)}\sum_{k=1}^{2}(-1)^{k}k^3\binom{2}{k}\sigma'''(\xi_k),
\end{aligned}
$$

where $\xi_k \in [t-k|x|/K, t+k|x|/K] \subset (a,b)$. Since the third order derivative is bounded on (a,b), we get the desired assertion by retaking $K \leftarrow \sqrt{2/\sigma''(t)}K$.

Proof of (b). The proof can be done straightforwardly by the polarization type identity:

$$x_1 x_2 = 2A^2\left\{\left(\frac{x_1 + x_2}{2A}\right)^2 - \left(\frac{x_1}{2A}\right)^2 - \left(\frac{x_1}{2A}\right)^2\right\}.$$

We construct the network as

$$N_\sigma(\mathbf{x}|\boldsymbol{\theta}_{\times,A}) := 2A^2\left\{N_\sigma\left(\frac{x_1+x_2}{2A}\bigg|\boldsymbol{\theta}_2\right) - N_\sigma\left(\frac{x_1}{2A}\bigg|\boldsymbol{\theta}_2\right) - N_\sigma\left(\frac{x_2}{2A}\bigg|\boldsymbol{\theta}_2\right)\right\}, \tag{A4}$$

where $\boldsymbol{\theta}_2$ is defined in (A3). Since $(x_1+x_2)/2A, x_1/2A, x_2/2A \in [-1,1]$ for $\mathbf{x} \in [-A,A]^2$, the triangle inequality implies that $|N_\sigma(\mathbf{x}|\boldsymbol{\theta}_{\times,A}) - x_1 x_2| \leq 6A^2 C_1/K$.

Proof of(c). Let $q := \lceil \log_2 \alpha \rceil$. We construct $\boldsymbol{\theta}_\mathbf{m}$ as follows. Fix $\mathbf{x} \equiv (x_1, \ldots, x_d) \in [0,1]^d$. We first consider the affine map that transforms $(x_1, \ldots, x_d)$ to $\mathbf{z} \in [0,1]^{2^q}$ which is given by

$$\mathbf{z} := (\underbrace{x_1, \ldots, x_1}_{m_1 \text{ times}}, \underbrace{x_2, \ldots, x_2}_{m_2 \text{ times}}, \ldots, \underbrace{x_d, \ldots, x_d}_{m_d \text{ times}}, \underbrace{1, \ldots, 1}_{2^q - |\mathbf{m}| \text{ times}}).$$

The first hidden layer of $\boldsymbol{\theta}_\mathbf{m}$ pairs neighboring entries in $\mathbf{z}$ and applies the network $\boldsymbol{\theta}_{\times,A_1}$ defined in (b) with $A_1 = 1$ to each pair. That is, the first hidden layer of $\boldsymbol{\theta}_\mathbf{m}$ produces

$$\left\{g_{1,j} := N_\sigma((z_{2j-1}, z_{2j})|\boldsymbol{\theta}_{\times,1}) : j = 1, \ldots, 2^{q-1}\right\}.$$

Note that $\sup_{1 \leq j \leq 2^{q-1}}|g_{1,j} - z_{2j-1}z_{2j}| \leq 6C_1/K$ and $\sup_{1 \leq j \leq 2^{q-1}}|g_{1,j}| \leq 6C_1/K + 1$, where $6C_1/K + 1$ can be bounded by some constant $A_2 > 1$ depending only on C_1 and K_0. Then the second hidden layer of $\boldsymbol{\theta}_\mathbf{m}$ pairs neighboring entries of $\left\{g_{1,j} : j = 1, \ldots, 2^{q-1}\right\}$ and applies $\boldsymbol{\theta}_{\times,A_2}$ to each pair to have

$$\left\{g_{2,j} := N_\sigma((g_{1,2j-1}, g_{1,2j})|\boldsymbol{\theta}_{\times,A_2}) : j = 1, \ldots, 2^{q-2}\right\}.$$

Note that $\sup_{1\le j\le 2^{q-2}}|g_{2,j} - g_{1,2j-1}g_{1,2j}| \le 6C_1 A_2^2/K$ and $\sup_{1\le j\le 2^{q-2}}|g_{2,j}| \le 6C_1 A_2^2/K + 1 \le A_3$ for some $A_3 > 1$ depending only on C_1 and K_0. We repeat this procedure to produce $\left\{g_{k,j} : j = 1, \ldots, 2^{q-k}\right\}$ for $k = 3, \ldots, q$ with

$$\sup_{1\le j\le 2^{q-k}}\left|g_{k,j} - g_{k-1,2j-1}g_{k-1,2j}\right| \le \frac{6C_1 A_k^2}{K}, \quad \sup_{1\le j\le 2^{q-k}}\left|g_{k,j}\right| \le A_{k+1},$$

for some $A_{k+1} > 1$, and we set $N_\sigma(\mathbf{x}|\boldsymbol{\theta_m})$ equal to $g_{q,1}$.

By applying the triangle inequality repeatedly, we have

$$\left|g_{q,1} - \mathbf{x^m}\right| \le \left|g_{q,1} - g_{q-1,1}g_{q-1,2}\right| + \left|g_{q-1,1} - \prod_{j=1}^{2^{q-1}} z_j\right|\left|g_{q-1,2}\right| + \left|g_{q-1,2} - \prod_{j=2^{q-1}+1}^{2^q} z_j\right|\left|\prod_{j=1}^{2^{q-1}} z_j\right|$$

$$\le \frac{6C_1 A_q^2}{K} + A_q\left|g_{q-1,1} - \prod_{j=1}^{2^{q-1}} z_j\right| + \left|g_{q-1,2} - \prod_{j=2^{q-1}+1}^{2^q} z_j\right|$$

$$\le \frac{6C_1 A_q^2}{K} + (A_q + 1)\frac{6C_1 A_{q-1}^2}{K} + A_q A_{q-1}\left|g_{q-2,1} - \prod_{j=1}^{2^{q-2}} z_j\right| + A_q\left|g_{q-2,2} - \prod_{j=2^{q-2}+1}^{2\times 2^{q-2}} z_j\right|$$

$$+ A_{q-1}\left|g_{q-2,3} - \prod_{j=2\times 2^{q-2}+1}^{3\times 2^{q-2}} z_j\right| + \left|g_{q-2,4} - \prod_{j=3\times 2^{q-2}+1}^{4\times 2^{q-2}} z_j\right|$$

$$\le \cdots \le \sum_{k=0}^{q-1}\left\{A_{q-k}^2 \prod_{h=q-k+1}^{q}(A_h + 1)\right\}\frac{6C_1}{K} \le C_1'\frac{1}{K},$$

for some $C_1' > 0$ depending only on C_1, K_0 and q. Since we set $\mathbf{x}$ arbitrary, the proof is done.

Proof of (d). By (b), it is easy to verify that there is a network $\boldsymbol{\theta}_1 \in \Theta_{1,1}(1,6)$ with $|\boldsymbol{\theta}_1|_\infty \le \max\{K^2, 2\}$ such that $|\sigma(x) - x| \le C_1'/K$ for any $x \in [-1, 1]$ and some constant $C_1' > 0$. The Taylor series with Lagrange remainder around 1 of $\sqrt{x}$ is given by

$$\sqrt{x} = \sum_{k=0}^{J}\frac{(x-1)^k}{k!} + \frac{1}{(J+1)!}\frac{d^{J+1}\sqrt{x}}{dx^{J+1}}\Big|_{x=\xi}(x-1)^{J+1},$$

where $\xi \in [0, 2]$, and thus

$$\sup_{x\in[0,2]}\left|\sqrt{x} - \sum_{k=0}^{J}\frac{(x-1)^k}{k!}\right| \le C_1'\frac{1}{(J+1)!} \le e\left(\frac{e}{J+1}\right)^{J+1}.$$

for some $C_1' > 0$, where the last inequality is because $n! \ge (n/e)^n e$.

Now, we will construct a neural network $\boldsymbol{\theta}_{p,J}$ that approximates the polynomial $\sum_{k=0}^{J}\frac{(x-1)^k}{k!}$ as follows. The first hidden layer computes $(N_\sigma(x-1|\boldsymbol{\theta}_2)/2, N_\sigma(x-1|\boldsymbol{\theta}_1))$ from the input x. Then

$$\left|(N_\sigma(x-1|\boldsymbol{\theta}_2)/2, N_\sigma(x-1|\boldsymbol{\theta}_1)) - \left((x-1)^2/2, (x-1)\right)\right|_\infty \le C_2'\frac{1}{K},$$

for any $x \in [0,1]$ and some constant $C_2' > 0$. The next hidden layer computes $(N_\sigma((u,v)|\boldsymbol{\theta}_{\times,1+C_2'/K})/3, N_\sigma(u+v|\boldsymbol{\theta}_1))$ from the input (u,v) from the first hidden layer. Using the triangle inequality, we have that the second hidden layer approximates the vector $((x-1)^3/3!, (x-1)^2/2 + (x-1))$ by error $\leq 2C_3'/K$ for some $C_3' > 0$. Repeating this procedure, we construct the network $\boldsymbol{\theta}_{p,J} \in \Theta_{1,1}(J,15)$ which approximates $\sum_{k=0}^J \frac{(x-1)^k}{k!}$ by error $\leq C_4'J/K$ for some $C_4' > 0$. Taking $J = \lceil \log K \rceil$, we observe that $(e/J+1)^{J+1} \leq (e/\log K)^{\log K+1} \leq eK/(\log K)^{\log K} \leq 1/K$ for all sufficiently large K, which implies the desired result.

Proof of (e). Let $\zeta \in (0,1)$. Since for any $x \in \mathbb{R}$,

$$\sqrt{x^2 + \zeta^2} - |x| \leq \frac{\zeta^2}{\sqrt{x^2 + \zeta^2} + |x|} \leq \frac{\zeta^2}{\zeta} = \zeta,$$

the function $\sqrt{x^2 + \zeta^2}$ approximates the absolute value function $|x|$ by error ζ. For $\boldsymbol{\theta}_2$ in (a) and $\boldsymbol{\theta}_{1/2}$ in (d), we have that

$$\left| N_\sigma\left(N_\sigma(x|\boldsymbol{\theta}_2) + \zeta^2 \middle| \boldsymbol{\theta}_{1/2}\right) - |x| \right| \leq \left| N_\sigma\left(N_\sigma(x|\boldsymbol{\theta}_2) + \zeta^2 \middle| \boldsymbol{\theta}_{1/2}\right) - \sqrt{x^2 + \zeta^2} \right| + \zeta$$

$$\leq \left| N_\sigma\left(N_\sigma(x|\boldsymbol{\theta}_2) + \zeta^2 \middle| \boldsymbol{\theta}_{1/2}\right) - \sqrt{N_\sigma(x|\boldsymbol{\theta}_2) + \zeta^2} \right|$$

$$+ \left| \sqrt{N_\sigma(x|\boldsymbol{\theta}_2) + \zeta^2} - \sqrt{x^2 + \zeta^2} \right| + \zeta$$

$$\leq C_1' \left(\frac{\log K}{K} + \frac{1}{K\zeta} \right) + \zeta$$

for some constant $C_1' > 0$. We now set $\zeta = 1/\sqrt{K}$ and $N_\sigma(x|\boldsymbol{\theta}_{\text{abs}}) := N_\sigma(N_\sigma(x|\boldsymbol{\theta}_2) + K^{-1}|\boldsymbol{\theta}_{1/2})$. Since $(\log K)/K = o(1/\sqrt{K})$, the proof is done. $\square$

Proof of Theorem 1 for locally quadratic activation functions. Recall that

$$P_M(\mathbf{x}) = \sum_{\mathbf{z} \in \mathbb{G}_{d,M}} \sum_{\mathbf{m} \in \mathbb{N}_0^d : |\mathbf{m}| \leq \alpha} \beta_{\mathbf{z},\mathbf{m}} \mathbf{x}^{\mathbf{m}} \phi_{\mathbf{z},M}(\mathbf{x}).$$

Then by Lemma B.1 of [10],

$$\sup_{\mathbf{x} \in [0,1]^d} |P_M(\mathbf{x}) - f(\mathbf{x})| \leq RM^{-\alpha}.$$

From the equivalent representation of the ReLU function $(x)_+ = (x + |x|)/2$, we can easily check that the neural network $N_\sigma(x|\boldsymbol{\theta}_{\text{relu}}) := (N(x|\boldsymbol{\theta}_{\text{abs}}) + N_\sigma(x|\boldsymbol{\theta}_1))/2$ with $\boldsymbol{\theta}_{\text{relu}} \in \Theta_{1,1}(\lceil \log K \rceil, 21)$ approximates the ReLU function by error $\leq C_1'/\sqrt{K}$ for some $C_1' > 0$, where $\boldsymbol{\theta}_1 \in \Theta_{1,1}(1,6)$ is defined in the proof of (d) of Lemma A3 and $\boldsymbol{\theta}_{\text{abs}} \in \Theta_{1,1}(\lceil \log K \rceil, 15)$ is defined in (e) of Lemma A3. For $z \in (0,1)$ and $M \in \mathbb{N}$, we define

$$N_\sigma(x|\boldsymbol{\theta}_{\phi,z,M}) := N_\sigma\left(1/M - N_\sigma((x-z)|\boldsymbol{\theta}_{\text{abs}}) \middle| \boldsymbol{\theta}_{\text{relu}}\right).$$

Then it approximates the function $(1/M - |x-z|)_+$ by error $\leq C_2'/\sqrt{K}$ for some $C_2' > 0$. In turn, for $\mathbf{z} \in \mathbb{G}_{d,M}$, by invoking the similar construction used in (c) of Lemma A3 to approximates the product of d

components, we can construct the network $\boldsymbol{\theta}_{\phi,\mathbf{z},M} \in \Theta_{1,1}(\lceil \log K \rceil + \lceil \log_2 d \rceil, 21d)$ with $\left| \boldsymbol{\theta}_{\phi,\mathbf{z},M} \right|_\infty \leq C_3' K^2$ for some $C_3' > 0$ such that

$$\sup_{\mathbf{x} \in [0,1]^d} \left| N(\mathbf{x}|\boldsymbol{\theta}_{\phi,\mathbf{z},M}) - \prod_{j=1}^d \left(\frac{1}{M} - |x_j - z_j| \right)_+ \right| \leq C_4' \frac{1}{\sqrt{K}},$$

for some $C_4' > 0$. For each $\mathbf{m} \in \mathbb{N}_0^d$ with $|\mathbf{m}| \leq \alpha$, we have the neural network $\boldsymbol{\theta}_\mathbf{m}$ in (c) of Lemma A3 that approximates $\mathbf{x}^\mathbf{m}$. The number of these networks is $\binom{d+\alpha}{\alpha}$, which is denoted by A_α. Also there are $|\mathbb{G}_{d,M}| = (M+1)^d$ networks $\boldsymbol{\theta}_{\phi,\mathbf{z},M}$ for $\mathbf{z} \in \mathbb{G}_{d,M}$. We need approximation of each product $\mathbf{x}^\mathbf{m} \phi_{\mathbf{z},M}$, which requires additional $A_\alpha (M+1)^d$ many networks $\boldsymbol{\theta}_{\times,A} \in \Theta_{2,1}(1,9)$, where $\boldsymbol{\theta}_{\times,A}$ is defined as in (A4) for some $A > 1$ not depending on M and K. Finally we construct the output layer which computes the weighted sum of $\left\{ N_\sigma \left((N_\sigma(\mathbf{x}|\boldsymbol{\theta}_\mathbf{m}), N_\sigma(\mathbf{x}|\boldsymbol{\theta}_{\phi,\mathbf{z},M})) | \boldsymbol{\theta}_{\times,A} \right) : \mathbf{m} \in \mathbb{N}_0^d, |\mathbf{m}| \leq \alpha, \mathbf{z} \in \mathbb{G}_{d,M} \right\}$. Letting $\boldsymbol{\theta}_{f,K,M}$ be the network constructed above, we can check that

$$\sup_{\mathbf{x} \in [0,1]^d} \left| N(\mathbf{x}|\boldsymbol{\theta}_{f,K,M}) - P_M(\mathbf{x}) \right| \leq C_5' A_\alpha (M+1)^d \left(\frac{1}{K} + \frac{1}{\sqrt{K}} \right) \leq C_6' \frac{(M+1)^d}{\sqrt{K}},$$

for some positive constants C_5' and C_6'. In addition, we have $L(\boldsymbol{\theta}_{f,K,M}) \leq 1 + (\lceil \log K \rceil + \lceil \log_2(\alpha \vee d) \rceil) \leq C_7' \lceil \log K \rceil$ and $n_{\max}(\boldsymbol{\theta}_{f,K,M}) \leq C_8' A_\alpha (M+1)^d$ for some positive constants C_7' and C_8'. For sparsity of the network, we have

$$\begin{aligned} \left| \boldsymbol{\theta}_{f,K,M} \right|_0 &\leq A_\alpha (M+1)^d \left| \boldsymbol{\theta}_{\times,A} \right|_0 + (M+1)^d \left| \boldsymbol{\theta}_{\phi,\mathbf{z},M} \right|_0 + A_\alpha \left| \boldsymbol{\theta}_\mathbf{m} \right|_0 \\ &\leq C_9' \lceil \log K \rceil (M+1)^d, \end{aligned}$$

for some $C_9' > 0$. Taking $M + 1 = \epsilon^{-1/\alpha}$ and $K = \epsilon^{-2d/\alpha-2}$, we have

$$\boldsymbol{\theta}_{f,K,M} \in \Theta \left(L_0 \log(1/\epsilon), N_0 \epsilon^{-d/\alpha}, S_0 \epsilon^{-d/\alpha} \log(1/\epsilon), B_0 \epsilon^{-4(d/\alpha+1)} \right),$$

so that $\left\| P_M - N_\sigma(\cdot | \boldsymbol{\theta}_{f,K,M}) \right\|_\infty \leq C_{10}' \epsilon$ for some $C_{10}' > 0$. Since $\left\| f - P_M \right\|_\infty \leq RM^{-\alpha} \leq C_{11}' \epsilon$ for some $C_{11}' > 0$, the proof is done. $\square$

Appendix B. Proofs of Proposition 1

Proof. Given a deep neural network $\boldsymbol{\theta} = ((\mathbf{W}_1, \mathbf{b}_1), \dots, (\mathbf{W}_{L+1}, \mathbf{b}_{L+1})) \in \Theta_{d,1}(L, N, S, B)$, we define $\check{N}_{l,\sigma,\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^{n_{l-1}}$ and $\hat{N}_{l,\sigma,\boldsymbol{\theta}} : \mathbb{R}^{n_l} \to \mathbb{R}$ as

$$\begin{aligned} \check{N}_{l,\sigma,\boldsymbol{\theta}}(\mathbf{x}) &:= \sigma_{l-1} \circ \mathsf{A}_{l-1} \circ \cdots \circ \sigma_1 \circ \mathsf{A}_1(\mathbf{x}), \\ \hat{N}_{l,\sigma,\boldsymbol{\theta}}(\mathbf{x}) &:= \mathsf{A}_{L+1} \circ \sigma_L \circ \mathsf{A}_L \circ \cdots \sigma_l \circ \mathsf{A}_l \circ \sigma_{l-1}(\mathbf{x}), \end{aligned}$$

for $l \in 2, \dots, L$, where $\mathsf{A}_l \mathbf{x} = \mathbf{W}_l \mathbf{x} + \mathbf{b}_l$. Corresponding to the last and first layer, we define $\check{N}_{1,\sigma,\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}$ and $\hat{N}_{L+1,\sigma,\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}$. Note that $N_\sigma(\mathbf{x}|\boldsymbol{\theta}) = \hat{N}_{l+1,\sigma,\boldsymbol{\theta}} \circ \mathsf{A}_l \circ \check{N}_{l,\sigma,\boldsymbol{\theta}}(\mathbf{x})$. For given $\delta > 0$, let $\boldsymbol{\theta} = ((\mathbf{W}_1, \mathbf{b}_1), \dots, (\mathbf{W}_{L+1}, \mathbf{b}_{L+1})) \in \Theta_{d,1}(L, N, S, B)$ and $\boldsymbol{\theta}^* = ((\mathbf{W}_1^*, \mathbf{b}_1^*), \dots, (\mathbf{W}_{L+1}^*, \mathbf{b}_{L+1}^*)) \in \Theta_{d,1}(L, N, S, B)$

be two neural network parameter such that $\left|\mathrm{vec}(\mathbf{W}_l - \mathbf{W}_l^*)\right|_\infty \leq \delta$ and $\left|\mathbf{b}_l - \mathbf{b}_l^*\right|_\infty \leq \delta$ for $l = 1, \ldots, L+1$. Let C_σ be the Lipschitz constant of σ. We observe that

$$\left\|\check{N}_{l,\sigma,\boldsymbol{\theta}}\right\|_\infty \leq C_\sigma\left(NB\left\|\check{N}_{l-1,\sigma,\boldsymbol{\theta}}\right\|_\infty + B\right)$$

$$\leq C_\sigma(B \vee 1)(N+1)\left\|\check{N}_{l-1,\sigma,\boldsymbol{\theta}}\right\|_\infty$$

$$\leq \left\{C_\sigma(B \vee 1)(N+1)\right\}^{l-1},$$

and similarly, $\left\|\hat{N}_{l,\sigma,\boldsymbol{\theta}}\right\|_\infty \leq (C_\sigma BN)^{L-l+1}$. Letting $A_l^* \mathbf{x} = \mathbf{W}_l^* \mathbf{x} + \mathbf{b}_l^*$, we have

$$\left\|N_\sigma(\cdot|\boldsymbol{\theta}) - N_\sigma(\cdot|\boldsymbol{\theta}^*)\right\|_\infty \leq \left\|\sum_{l=1}^L \left[\hat{N}_{l+1,\sigma,\boldsymbol{\theta}^*} \circ A_l \circ \check{N}_{l,\sigma,\boldsymbol{\theta}}(\cdot) - \hat{N}_{l+1,\sigma,\boldsymbol{\theta}^*} \circ A_l^* \circ \check{N}_{l,\sigma,\boldsymbol{\theta}}(\cdot)\right]\right\|_\infty$$

$$\leq \sum_{l=1}^L (C_\sigma BN)^{L-l}\left\|(A_l - A_l^*) \circ \check{N}_{l,\sigma,\boldsymbol{\theta}}(\cdot)\right\|_\infty$$

$$\leq \sum_{l=1}^L (C_\sigma BN)^{L-l}\delta\left\{C_\sigma(B \vee 1)(N+1)\right\}^{l-1}$$

$$\leq \delta L\left\{C_\sigma(B \vee 1)(N+1)\right\}^L.$$

Thus, for a fixed sparsity pattern (i.e., the location of nonzero elements in $\boldsymbol{\theta}$), the covering number is bounded by $\left[\delta/L\left\{C_\sigma(B \vee 1)(N+1)\right\}^L\right]^{-S}$. Since the number of the sparsity patterns is bounded by $\binom{(N+1)^L}{S} \leq (N+1)^{LS}$, the log of covering number is bounded above by

$$\log\left((N+1)^{LS}\left[\frac{L\left\{C_\sigma(B \vee 1)(N+1)\right\}^L}{\delta}\right]^S\right) \leq 2LS\log\left(\frac{C_\sigma L(B \vee 1)(N+1)}{\delta}\right),$$

which completes the proof.　□

Appendix C. Proof of Theorem 2

The proof Theorem 2 is based on the following oracle inequality.

Lemma A4 (Lemma 4 of [10]). *Assume that $Y|\mathbf{X} = \mathbf{x} \sim \mathsf{N}(f_0(\mathbf{x}), 1)$ for some f_0 with $\|f_0\|_\infty \leq R$. Let $\mathcal{F}^\dagger$ be a given function class from $[0,1]^d$ to $[-2R, 2R]$, and let $\hat{f}$ be any estimator in $\mathcal{F}^\dagger$. Then for any $\delta \in (0,1]$, we have*

$$\mathsf{E}\left[\mathsf{E}_{\mathbf{X}\sim P_\mathbf{x}}\left(\hat{f}(\mathbf{X}) - f_0(\mathbf{X})\right)^2\right] \leq 4\left[\inf_{f \in \mathcal{F}^\dagger} \mathsf{E}_{\mathbf{X}\sim P_\mathbf{x}}\left(f(\mathbf{X}) - f_0(\mathbf{X})\right)^2\right.$$

$$\left. + (4R)^2\frac{18\log\mathcal{N}(\delta, \mathcal{F}^\dagger, \|\cdot\|_\infty) + 72}{n} + 32\delta(4R) + \Delta_n\right],$$

with

$$\Delta_n := \mathsf{E}\left[\frac{1}{n}\sum_{i=1}^n\left(Y_i - \hat{f}(\mathbf{X}_i)\right)^2 - \inf_{f \in \mathcal{F}^\dagger}\frac{1}{n}\sum_{i=1}^n\left(Y_i - f(\mathbf{X}_i)\right)^2\right],$$

where the expectations are taken over the training data.

Proof of Theorem 2. We apply Lemma A4 to $\mathcal{F}^\dagger = \mathcal{F}_{\sigma,n}$ and $\hat{f} = \hat{f}_n \in \text{argmin}_{f \in \mathcal{F}_{\sigma,n}} \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2$.
By definition of $\hat{f}_n$, we have $\Delta_n = 0$. Also it can be easily verified that $f_0 = \text{argmin}_{f \in \mathcal{F}} \mathcal{R}_{2,f_0}(f)$ and
$E_{f_0,P_\mathbf{x}} \left(\hat{f}_n(\mathbf{X}) - f_0(\mathbf{X}) \right)^2 = \mathcal{R}_{2,f_0}(\hat{f}_n) - \mathcal{R}_{2,f_0}(f_0)$. Set $\delta = 1/n$. By Proposition 1,

$$\log \mathcal{N}\left(\frac{1}{n}, \mathcal{F}_{\sigma,n}, \|\cdot\|_\infty \right) \leq C_1' n^{\frac{d}{2\alpha+d}} \log^3 n,$$

for some $C_1' > 0$. If a function f_n is approximates f_0 by error ϵ which is sufficeintly small, then $\|f_n\|_\infty \leq 2R$
since $\|f_0\|_\infty \leq R$. Now, Theorem 1 implies that there is $f_n \in \mathcal{F}_{\sigma,n}$ such that

$$E_{f_0,P_\mathbf{x}} \left(f_n(\mathbf{X}) - f_0(\mathbf{X}) \right)^2 \leq C_2' \sup_{\mathbf{x} \in [0,1]^d} |f_n(\mathbf{x}) - f_0(\mathbf{x})|^2$$

$$\leq C_3' \left(\left(n^{\frac{d}{2\alpha+d}} \right)^{-d/\alpha} \right)^2 = C_3' n^{-\frac{2\alpha}{2\alpha+d}},$$

which completes the proof. $\square$

Appendix D. Proof of Theorem 3

For a given real-valued function f, let $\mathcal{R}_{\text{hinge},\eta}(f) := E_{Y|\mathbf{X} \sim 2\text{Bern}(\eta(\mathbf{X}))-1, \mathbf{X} \sim P_\mathbf{x}} \ell_{\text{hinge}}(Yf(\mathbf{X}))$, which we
call the hinge risk. The proof of Theorem 3 is based on the following theorem, which is given in [28].

Lemma A5 (Theorem 6 of [28]). *Assume that $\eta(\mathbf{x})$ satisfies the Tsybakov noise condition (8) with the noise
exponent $q \in [0, \infty]$. Assume that there exists a sequence $(\delta_n)_{n \in \mathbb{N}}$ such that*

- *there exists a sequence of classes of functions $\{\mathcal{F}_n\}_{n \in \mathbb{N}}$ with $\sup_{n \in \mathbb{N}} \sup_{f \in \mathcal{F}_n} \|f\|_\infty \leq F$ for some $F > 0$ such
  that there is $f_n \in \mathcal{F}_n$ with $\mathcal{R}_{\text{hinge},\eta}(f_n) - \min_{f \in \mathcal{F}} \mathcal{R}_{\text{hinge},\eta}(f) \leq C_1 \delta_n$ for some universal constant $C_1 > 0$;*
- $\log \mathcal{N}(\delta_n, \mathcal{F}_n, \|\cdot\|_\infty) \leq C_2 n \delta_n^{(q+2)/(q+1)}$ *for some universal constant $C_2 > 0$.*

Then the estimator $\hat{f}_n$ obtained by

$$\hat{f}_n \in \underset{f \in \mathcal{F}_n}{\text{argmin}} \sum_{i=1}^n \ell_{\text{hinge}}(y_i f(\mathbf{x}_i))$$

satisfies

$$E\left[\mathcal{R}_{01,\eta}(\hat{f}_n) - \min_{f \in \mathcal{F}} \mathcal{R}_{01,\eta}(f) \right] \leq C_3 \delta_n,$$

for some universal constant $C_3 > 0$, where the expectation is taken over the training data.

Proof of Theorem 3. It is well known that $f^* = 2\mathbb{1}(\eta(\cdot) \geq 1/2) - 1 = \text{argmin}_{f \in \mathcal{F}} \mathcal{R}_{\text{hinge},\eta}(f)$, i.e., the
hinge risk minimizer is equal to the Bayes classifier [32]. The first step is to find a function $f_n \in \mathcal{F}_{\sigma,n}$
which approximates the Bayes classifier f^* well. Let $(\xi_n)_{n \in \mathbb{N}}$ be a given sequence of positive integers.
Since $\eta \in \mathcal{H}^{\alpha,R}([0,1]^d)$, by Theorem 6, for each ξ_n there exists $\boldsymbol{\theta}_n$ such that $\|N_\sigma(\cdot|\boldsymbol{\theta}_n) - \eta(\cdot)\|_\infty \leq \xi_n$ with
at most $O(\log(1/\xi_n))$ layers, $O(\xi_n^{-d/\alpha})$ nodes at each layer and $O(\xi_n^{-d/\alpha} \log(1/\xi_n))$ nonzero parameters.
We construct the neural network f_n by adding one ReLU layer to $N_\sigma(\cdot|\boldsymbol{\theta}_n)$ to have

$$f_n(\mathbf{x}) = 2\left\{ \rho\left(\frac{1}{\xi_n}\left(N_\sigma(\mathbf{x}|\boldsymbol{\theta}_n) - \frac{1}{2} \right) \right) - \rho\left(\frac{1}{\xi_n}\left(N_\sigma(\mathbf{x}|\boldsymbol{\theta}_n) - \frac{1}{2} \right) - 1 \right) \right\} - 1,$$

where ρ is the ReLU activation function. Note that $f_n(\mathbf{x})$ is equal to 1 if $N_\sigma(\mathbf{x}|\boldsymbol{\theta}_n) \geq 1/2 + \xi_n$, $(N_\sigma(\mathbf{x}|\boldsymbol{\theta}_n) - 1/2)/\xi_n$ if $1/2 \leq (N_\sigma(\mathbf{x}|\boldsymbol{\theta}_n) < 1/2 + \xi_n$ and -1 otherwise. Let

$$B(4\xi_n) = \{\mathbf{x} : |2\eta(\mathbf{x}) - 1| > 4\xi_n\}.$$

Then on $B(4\xi_n)$, $|f_n(\mathbf{x}) - f^*(\mathbf{x})| = 0$, since $N_\sigma(\mathbf{x}|\boldsymbol{\theta}_n) - 1/2 = (\eta(\mathbf{x}) - 1/2) - ((N_\sigma(\mathbf{x}|\boldsymbol{\theta}_n) - \eta(\mathbf{x})) \geq \xi_n$ when $2\eta(\mathbf{x}) - 1 > 4\xi_n$. Similarly we can show that $N_\sigma(\mathbf{x}|\boldsymbol{\theta}_n) - 1/2 < -\xi_n$ when $2\eta(\mathbf{x}) - 1 < -4\xi_n$. Therefore the Tsybakov noise condition (8) implies

$$\begin{aligned}
\mathcal{R}_{\text{hinge},\eta}(f_n) - \mathcal{R}_{\text{hinge},\eta}(f^*) &= \int |f_n(\mathbf{x}) - f^*(\mathbf{x})||2\eta(\mathbf{x}) - 1| d P_\mathbf{x}(\mathbf{x}) \\
&= \int_{B(4\xi_n)^c} |f_n(\mathbf{x}) - f^*(\mathbf{x})||2\eta(\mathbf{x}) - 1| d P_\mathbf{x}(\mathbf{x}) \\
&\leq 8\xi_n \Pr(|2\eta(\mathbf{x}) - 1| \leq 4\xi_n) \leq C_1' \xi_n^{q+1},
\end{aligned}$$

for some $C_1' > 0$, where the first equality follows from Theorem 2.31 of [33].

We take $\delta_n = C_1' \xi_n^{q+1}$. Then there are positive constants L_0, N_0, S_0 and B_0 such that $f_n \in \mathcal{F}_{\sigma,n}$ where

$$\mathcal{F}_{\sigma,n} := \left\{ N_\sigma(\cdot|\boldsymbol{\theta}) : \|N_\sigma(\cdot|\boldsymbol{\theta})\|_\infty \leq 1, \right.$$
$$\left. \boldsymbol{\theta} \in \Theta_{d,1}\left(L_0 \log(\delta_n^{-1}), N_0 \delta_n^{-d/\alpha(q+1)}, S_0 \delta_n^{-d/\alpha(q+1)} \log(\delta_n^{-1}), B_0 \delta_n^{-\kappa'} \right) \right\},$$

for some $\kappa' > 0$. Propostion 1 implies that the log covering number of $\mathcal{F}_{\sigma,n}$ is bounded above by

$$\log \mathcal{N}\left(\delta_n, \mathcal{F}_{\sigma,n}, \|\cdot\|_\infty \right) \leq \delta_n^{-d/\alpha(q+1)} \log^3(\delta_n^{-1}).$$

Note that to satisfy the entropy condition of Lemma A5, δ_n should satisfy

$$(\delta_n)^{\frac{d}{\alpha(q+1)} + \frac{q+2}{q+1}} \geq C_2' n^{-1} \log^3(\delta_n^{-1}) \tag{A5}$$

for some $C_2' > 0$. If we let $\delta_n = (\log^3 n/n)^{\alpha(q+1)/(\alpha(q+2)+d)}$, the condition (A5) holds and so the proof is done. $\square$

References

1. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [CrossRef]
2. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
3. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Syst.* **1989**, *2*, 303–314. [CrossRef]
4. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366. [CrossRef]
5. Funahashi, K.I. On the approximate realization of continuous mappings by neural networks. *Neural Netw.* **1989**, *2*, 183–192. [CrossRef]
6. Chui, C.K.; Li, X. Approximation by ridge functions and neural networks with one hidden layer. *J. Approx. Theory* **1992**, *70*, 131–141. [CrossRef]
7. Leshno, M.; Lin, V.Y.; Pinkus, A.; Schocken, S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Netw.* **1993**, *6*, 861–867. [CrossRef]

8. Telgarsky, M. Neural networks and rational functions. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 3387–3393.

9. Yarotsky, D. Error bounds for approximations with deep ReLU networks. *Neural Netw.* **2017**, *94*, 103–114. [CrossRef]

10. Schmidt-Hieber, J. Nonparametric regression using deep neural networks with ReLU activation function. *arXiv* **2017**, arXiv:1708.06633 .

11. Bauer, B.; Kohler, M. On deep learning as a remedy for the curse of dimensionality in nonparametric regression. *Ann. Stat.* **2019**, accepted. [CrossRef]

12. Li, B.; Tang, S.; Yu, H. Better Approximations of High Dimensional Smooth Functions by Deep Neural Networks with Rectified Power Units. *arXiv* **2019**, arXiv:1903.05858.

13. Suzuki, T. Adaptivity of deep ReLU network for learning in Besov and mixed smooth Besov spaces: Optimal rate and curse of dimensionality. *arXiv* **2018**, arXiv:1810.08033.

14. Petersen, P.; Voigtlaender, F. Optimal approximation of piecewise smooth functions using deep ReLU neural networks. *Neural Netw.* **2018**, *108*, 296–330. [CrossRef]

15. Imaizumi, M.; Fukumizu, K. Deep Neural Networks Learn Non-Smooth Functions Effectively. *arXiv* **2018**, arXiv:1802.04474.

16. Bergstra, J.; Desjardins, G.; Lamblin, P.; Bengio, Y. *Quadratic Polynomials Learn Better Image Features*; Technical Report 1337; Département d'Informatique et de Recherche Operationnelle, Université de Montréal: Montréal, QC, Canada, 2009.

17. Clevert, D.A.; Unterthiner, T.; Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv* **2015**, arXiv:1511.07289.

18. Carlile, B.; Delamarter, G.; Kinney, P.; Marti, A.; Whitney, B. Improving deep learning by inverse square root linear units (ISRLUs). *arXiv* **2017**, arXiv:1710.09967.

19. Klimek, M.D.; Perelstein, M. Neural Network-Based Approach to Phase Space Integration. *arXiv* **2018**, arXiv:1810.11509.

20. Wuraola, A.; Patel, N. SQNL: A New Computationally Efficient Activation Function. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–7.

21. Ramachandran, P.; Zoph, B.; Le, Q.V. Searching for activation functions. *arXiv* **2017**, arXiv:1710.05941.

22. Glorot, X.; Bordes, A.; Bengio, Y. Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Ft. Lauderdale, FL, USA, 11–13 April 2011; pp. 315–323.

23. Mhaskar, H.N. Approximation properties of a multilayered feedforward artificial neural network. *Adv. Comput. Math.* **1993**, *1*, 61–80. [CrossRef]

24. Costarelli, D.; Vinti, G. Saturation classes for max-product neural network operators activated by sigmoidal functions. *Results Math.* **2017**, *72*, 1555–1569. [CrossRef]

25. Costarelli, D.; Spigler, R. Solving numerically nonlinear systems of balance laws by multivariate sigmoidal functions approximation. *Comput. Appl. Math.* **2018**, *37*, 99–133. [CrossRef]

26. Costarelli, D.; Vinti, G. Estimates for the neural network operators of the max-product type with continuous and p-integrable functions. *Results Math.* **2018**, *73*, 12. [CrossRef]

27. Costarelli, D.; Sambucini, A.R. Approximation results in Orlicz spaces for sequences of Kantorovich max-product neural network operators. *Results Math.* **2018**, *73*, 15. [CrossRef]

28. Kim, Y.; Ohn, I.; Kim, D. Fast convergence rates of deep neural networks for classification. *arXiv* **2018**, arXiv:1812.03599.

29. Anthony, M.; Bartlett, P.L. *Neural Network Learning: Theoretical Foundations*; Cambridge University Press: Cambridge, UK, 2001.

30. Mammen, E.; Tsybakov, A.B. Smooth discrimination analysis. *Ann. Stat.* **1999**, *27*, 1808–1829.

31. Tsybakov, A.B. Optimal aggregation of classifiers in statistical learning. *Ann. Stat.* **2004**, *32*, 135–166. [CrossRef]
32. Lin, Y. A note on margin-based loss functions in classification. *Stat. Probab. Lett.* **2004**, *68*, 73–82. [CrossRef]
33. Steinwart, I.; Christmann, A. *Support Vector Machines*; Springer Science & Business Media: New York, NY, USA, 2008.