The Annals of Statistics
2021, Vol. 49, No. 4, 2231–2249
https://doi.org/10.1214/20-AOS2034

ON THE RATE OF CONVERGENCE OF FULLY CONNECTED DEEP NEURAL NETWORK REGRESSION ESTIMATES

BY MICHAEL KOHLER* AND SOPHIE LANGER†

*Fachbereich Mathematik, Technische Universität Darmstadt, *kohler@mathematik.tu-darmstadt.de; †langer@mathematik.tu-darmstadt.de*

Recent results in nonparametric regression show that deep learning, that is, neural network estimates with many hidden layers, are able to circumvent the so-called curse of dimensionality in case that suitable restrictions on the structure of the regression function hold. One key feature of the neural networks used in these results is that their network architecture has a further constraint, namely the *network sparsity*. In this paper, we show that we can get similar results also for least squares estimates based on simple fully connected neural networks with ReLU activation functions. Here, either the number of neurons per hidden layer is fixed and the number of hidden layers tends to infinity suitably fast for sample size tending to infinity, or the number of hidden layers is bounded by some logarithmic factor in the sample size and the number of neurons per hidden layer tends to infinity suitably fast for sample size tending to infinity. The proof is based on new approximation results concerning deep neural networks.

1. Introduction. Neural networks belong since many years to the most promising approaches in nonparametric statistics in view of multivariate statistical applications, in particular, in pattern recognition and in nonparametric regression (see, e.g., the monographs [1, 6, 12, 15, 16, 29]). In recent years, the focus in applications is on what is called deep learning, where multilayer feedforward neural networks with many hidden layers are fitted to observed data (see, e.g., [30] and the literature cited therein). Motivated by this practical success, there is also an increasing interest in the literature in showing good theoretical properties of these neural networks; see, for example, [8, 11, 24, 26, 36, 37] and the literature cited therein for the analysis of corresponding approximation properties of neural networks.

1.1. *Nonparametric regression.* In this paper, we study these kind of estimates in connection with nonparametric regression. Here, $(\mathbf{X}, Y)$ is an $\mathbb{R}^d \times \mathbb{R}$–valued random vector satisfying $\mathbf{E}\{Y^2\} < \infty$, and given a sample of size n of $(\mathbf{X}, Y)$, that is, given a data set

$$\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\},$$

where $(\mathbf{X}, Y), (\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$ are i.i.d., the aim is to construct an estimator

$$m_n(\cdot) = m_n(\cdot, \mathcal{D}_n) : \mathbb{R}^d \to \mathbb{R}$$

of the so-called regression function $m : \mathbb{R}^d \to \mathbb{R}$, $m(\mathbf{x}) = \mathbf{E}\{Y \mid \mathbf{X} = \mathbf{x}\}$ such that the so-called L_2-error

$$\int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x})$$

is "small" (cf., e.g., [12] for a systematic introduction to nonparametric regression and a motivation for the L_2-error).

Received August 2020; revised October 2020.
MSC2020 subject classifications. Primary 62G08; secondary 41A25, 82C32.
Key words and phrases. Curse of dimensionality, deep learning, neural networks, nonparametric regression, rate of convergence.

1.2. *Neural networks*. In order to construct such regression estimates with neural networks, the first step is to define a suitable space of functions $f : \mathbb{R}^d \to \mathbb{R}$ by using neural networks. The starting point here is the choice of an activation function $\sigma : \mathbb{R} \to \mathbb{R}$. Traditionally, so-called squashing functions are chosen as activation function $\sigma : \mathbb{R} \to \mathbb{R}$, which are nondecreasing and satisfy $\lim_{x\to-\infty} \sigma(x) = 0$ and $\lim_{x\to\infty} \sigma(x) = 1$, for example, the so-called sigmoidal or logistic squasher

$$\sigma(x) = \frac{1}{1+\exp(-x)}, \quad x \in \mathbb{R}.$$

Recently, also unbounded activation functions are used, for example, the ReLU activation function

$$\sigma(x) = \max\{x, 0\}.$$

The network architecture $(L, \mathbf{k})$ depends on a positive integer L called the *number of hidden layers* and a *width vector* $\mathbf{k} = (k_1, \ldots, k_L) \in \mathbb{N}^L$ that describes the number of neurons in the first, second, ..., Lth hidden layer. A multilayer feedforward neural network with network architecture $(L, \mathbf{k})$ and ReLU activation function σ is a real-valued function defined on $\mathbb{R}^d$ of the form

$$f(\mathbf{x}) = \sum_{i=1}^{k_L} c_{1,i}^{(L)} f_i^{(L)}(\mathbf{x}) + c_{1,0}^{(L)} \tag{1}$$

for some $c_{1,0}^{(L)}, \ldots, c_{1,k_L}^{(L)} \in \mathbb{R}$ and for $f_i^{(L)}$'s recursively defined by

$$f_i^{(s)}(\mathbf{x}) = \sigma\left(\sum_{j=1}^{k_{s-1}} c_{i,j}^{(s-1)} f_j^{(s-1)}(\mathbf{x}) + c_{i,0}^{(s-1)}\right)$$

for some $c_{i,0}^{(s-1)}, \ldots, c_{i,k_{s-1}}^{(s-1)} \in \mathbb{R}$, $s \in \{2, \ldots, L\}$, and

$$f_i^{(1)}(\mathbf{x}) = \sigma\left(\sum_{j=1}^{d} c_{i,j}^{(0)} x^{(j)} + c_{i,0}^{(0)}\right)$$

for some $c_{i,0}^{(0)}, \ldots, c_{i,d}^{(0)} \in \mathbb{R}$. The space of neural networks with L hidden layers and r neurons per layer is defined by

$$\mathcal{F}(L, r) = \{f : f \text{ is of the form (1) with } k_1 = k_2 = \cdots = k_L = r\}. \tag{2}$$

As there is no further restriction on the network architecture (e.g., no sparsity restriction as in [31]) and as two neurons are only connected if and only if they belong to neighboring layers, we refer to the networks of the class $\mathcal{F}(L, r)$, similar as [37], as *fully connected feedfoward neural networks*. The representation of this kind of network as a directed acyclic graph is shown in Figure 1. Here, we get an impression of how such a network looks like and why we call those networks *fully connected*. Remark that this network class also contains networks with some weights chosen as zero.

In the sequel, the number $L = L_n$ of hidden layers and number $r = r_n$ of neurons per hidden layer of the above function space are properly chosen. Then we define the corresponding neural network regression estimator as the minimizer of the so-called empirical L_2-risk over the function space $\mathcal{F}(L_n, r_n)$, that is, we define our estimator by

$$\tilde{m}_n(\cdot) = \arg\min_{f\in\mathcal{F}(L_n,r_n)} \frac{1}{n}\sum_{i=1}^{n} |f(\mathbf{X}_i) - Y_i|^2. \tag{3}$$

For simplicity, we assume here and in the sequel that the minimum above indeed exists. When this is not the case, our theoretical results also hold for any estimate which minimizes the above empirical L_2-risk up to a small additional term.

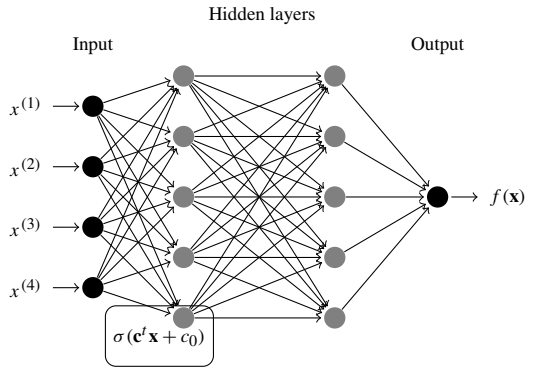

FIG. 1. *A fully connected network of the class* $\mathcal{F}(2,5)$.

1.3. *Curse of dimensionality.* In order to judge the quality of such estimates theoretically, usually the rate of convergence of the L_2-error is considered. It is well known that smoothness assumptions on the regression function are necessary in order to derive nontrivial results on the rate of convergence (see, e.g., Theorem 7.2 and Problem 7.2 in [6] and Section 3 in [7]). For that purpose, we introduce the following definition of (p, C)-smoothness.

DEFINITION 1. Let $p = q + s$ for some $q \in \mathbb{N}_0$ and $0 < s \le 1$. A function $m : \mathbb{R}^d \to \mathbb{R}$ is called (p, C)-smooth, if for every $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_d) \in \mathbb{N}_0^d$ with $\sum_{j=1}^d \alpha_j = q$ the partial derivative $\partial^q m / (\partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d})$ exists and satisfies

$$\left| \frac{\partial^q m}{\partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d}}(\mathbf{x}) - \frac{\partial^q m}{\partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d}}(\mathbf{z}) \right| \le C \|\mathbf{x} - \mathbf{z}\|^s$$

for all $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d$, where $\|\cdot\|$ denotes the Euclidean norm.

Stone [32] showed that the optimal minimax rate of convergence in nonparametric regression for (p, C)-smooth functions is $n^{-2p/(2p+d)}$. This rate suffers from a characteristic feature in case of high-dimensional functions: If d is relatively large compared to p, then this rate of convergence can be extremely slow (so-called curse of dimensionality). As was shown in [33, 34], it is possible to circumvent this curse of dimensionality by imposing structural assumptions like additivity on the regression function. This is also used, for example, in so-called single index models, in which

$$m(\mathbf{x}) = g\big(\mathbf{a}^\top \mathbf{x}\big), \quad \mathbf{x} \in \mathbb{R}^d$$

is assumed to hold, where $g : \mathbb{R} \to \mathbb{R}$ is a univariate function and $\mathbf{a} \in \mathbb{R}^d$ is a d-dimensional vector (see, e.g., [13, 14, 22, 38]). Related to this is the so-called projection pursuit, where the regression function is assumed to be a sum of functions of the above form, that is,

$$m(\mathbf{x}) = \sum_{k=1}^{K} g_k\big(\mathbf{a}_k^\top \mathbf{x}\big), \quad \mathbf{x} \in \mathbb{R}^d$$

for $K \in \mathbb{N}$, $g_k : \mathbb{R} \to \mathbb{R}$ and $\mathbf{a}_k \in \mathbb{R}^d$ (see, e.g., [10]). If we assume that the univariate functions in these postulated structures are (p, C)-smooth, adequately chosen regression estimates can achieve the above univariate rates of convergence up to some logarithmic factor (cf., e.g., Chapter 22 in [12]).

Horowitz and Mammen [17] studied the case of a regression function, which satisfies

$$m(\mathbf{x}) = g\left(\sum_{l_1=1}^{L_1} g_{l_1}\left(\sum_{l_2=1}^{L_2} g_{l_1,l_2}\left(\dots \sum_{l_r=1}^{L_r} g_{l_1,\dots,l_r}\big(\mathbf{x}^{l_1,\dots,l_r}\big)\right)\right)\right),$$

where $g, g_{l_1}, \dots, g_{l_1,\dots,l_r} : \mathbb{R} \to \mathbb{R}$ are (p, C)-smooth univariate functions and $\mathbf{x}^{l_1,\dots,l_r}$ are single components of $\mathbf{x} \in \mathbb{R}^d$ (not necessarily different for two different indices $(l_1, \dots, l_r)$). With the use of a penalized least squares estimate, they proved that in this setting the rate $n^{-2p/(2p+1)}$ can be achieved.

The rate of convergence of neural network regression estimates has been analyzed by [2–5, 18–20, 25, 27, 28, 31, 35]. For the L_2-error of a single hidden layer neural network, [4] proves a dimensionless rate of $n^{-1/2}$ (up to some logarithmic factor), provided the Fourier transform has a finite first moment (which basically requires that the function becomes smoother with increasing dimension d of $\mathbf{X}$). McCaffrey and Gallant [25] showed a rate of $n^{(-2p/(2p+d+5))+\varepsilon}$ for the L_2-error of suitably defined single hidden layer neural network estimators for (p, C)-smooth functions, but their study was restricted to the use of a certain cosine squasher as an activation function.

The rate of convergence of neural network regression estimates based on two-layer neural networks has been analyzed in [19]. Therein, interaction models were studied, where the regression function satisfies

$$m(\mathbf{x}) = \sum_{I \subseteq \{1,\dots,d\}, |I| = d^*} m_I(\mathbf{x}_I), \quad \mathbf{x} = \left(x^{(1)}, \dots, x^{(d)}\right)^\top \in \mathbb{R}^d$$

for some $d^* \in \{1, \dots, d\}$ and $m_I : \mathbb{R}^{d^*} \to \mathbb{R}$ ($I \subseteq \{1, \dots, d\}$, $|I| \le d^*$), where

$$\mathbf{x}_{\{i_1,\dots,i_{d^*}\}} = \left(x^{(i_1)}, \dots, x^{(i_{d^*})}\right) \quad \text{for } 1 \le i_1 < \cdots < i_{d^*} \le d,$$

and in case that all m_I are (p, C)-smooth for some $p \le 1$ it was shown that suitable neural network regression estimators achieve a rate of convergence of $n^{-2p/(2p+d^*)}$ (up to some logarithmic factor), which is again a convergence rate independent of d. In [20], this result was extended to so-called (p, C)-smooth generalized hierarchical interaction models of order d^*, which are defined as follows.

DEFINITION 2. Let $d \in \mathbb{N}$, $d^* \in \{1, \dots, d\}$ and $m : \mathbb{R}^d \to \mathbb{R}$.

(a) We say that m satisfies a generalized hierarchical interaction model of order d^* and level 0, if there exist $\mathbf{a}_1, \dots, \mathbf{a}_{d^*} \in \mathbb{R}^d$ and $f : \mathbb{R}^{d^*} \to \mathbb{R}$ such that

$$m(\mathbf{x}) = f\left(\mathbf{a}_1^\top \mathbf{x}, \dots, \mathbf{a}_{d^*}^\top \mathbf{x}\right) \quad \text{for all } \mathbf{x} \in \mathbb{R}^d.$$

(b) We say that m satisfies a generalized hierarchical interaction model of order d^* and level $l + 1$, if there exist $K \in \mathbb{N}$, $g_k : \mathbb{R}^{d^*} \to \mathbb{R}$ ($k \in \{1, \dots, K\}$) and $f_{1,k}, \dots, f_{d^*,k} : \mathbb{R}^d \to \mathbb{R}$ ($k \in \{1, \dots, K\}$) such that $f_{1,k}, \dots, f_{d^*,k}$ ($k \in \{1, \dots, K\}$) satisfy a generalized hierarchical interaction model of order d^* and level l and

$$m(\mathbf{x}) = \sum_{k=1}^{K} g_k\left(f_{1,k}(\mathbf{x}), \dots, f_{d^*,k}(\mathbf{x})\right) \quad \text{for all } \mathbf{x} \in \mathbb{R}^d.$$

(c) We say that the generalized hierarchical interaction model defined above is (p, C)-smooth, if all functions f and g_k occurring in its definition are (p, C)-smooth according to Definition 1.

It was shown that for such models least squares estimators based on suitably defined multilayer neural networks (in which the number of hidden layers depends on the level of the generalized interaction model) achieve the rate of convergence $n^{-2p/(2p+d^*)}$ (up to some logarithmic factor) in case $p \le 1$. Bauer and Kohler [5] showed that this result even holds for $p > 1$ provided the squashing function is suitably chosen. Similar rate of convergence results as in [5] have been shown in [31] for neural network regression estimates using the ReLU activation function. Here, slightly more general function spaces, which fulfill some composition

assumption, were studied. Related results have been shown in [35] in case of Besov spaces as a model for the smoothness of the regression function and in [27] in case of non-ReLU activation functions. Imaizumi and Fukumizu [18] derived results concerning estimation by neural networks of piecewise polynomial regression functions with partitions having rather general smooth boundaries. In [28], the rate of convergence of ResNet-type convolutional neural networks have been analyzed. Here, the convolutional neural networks correspond to a fully connected deep neural network with constant width and depth converging to infinity for sample size tending to infinity. The class of neural networks uses the ReLU activation function and very small bounds on the absolute value of the weights in the hidden layers and a large bound on the absolute value of the weights in the output layer. In case of a (p, C)-smooth regression function up to a logarithmic factor, the rate of convergence $n^{-2p/(2p+d)}$ is shown.

The main results in [5] and [31] are new approximation results for neural networks. Here, [31] bounds the supremum norm error of the approximation of smooth functions on a cube, while the corresponding approximation bound in [5] holds only on a subset of the cube of measure close to one, which is sufficient in order to bound the approximation error of the neural network in L_2. In both papers, a further restriction of the network architecture, in form of a sparsity constraint, is needed to show their theoretical results. Thus the topology of the neural network is difficult in view of an implementation of the corresponding least squares estimate. In particular, in [31] the topology of the neural network was not completely specified, it was described how many weights are nonzero but not which of the weights are nonzero.

1.4. *Main results in this article.* The above results lead to the conjecture that network sparsity is necessary in order to be able to derive good rates of convergence of neural network regression estimates. Our main result in this article is that this is not the case. To show this, we derive similar rate of convergence results as in [5] and in [31] for least squares estimators based on simple fully connected feedforward neural networks. In these networks, either the number of neurons per hidden layer is fixed and the number of hidden layers tends to infinity suitably fast for sample size tending to infinity, or the number of hidden layers is bounded by some logarithmic factor in the sample size and the number of neurons per hidden layer tends to infinity suitably fast for sample size tending to infinity. In the first case, the networks will be much deeper than the class of networks considered for the least squares estimates in [5] and [31], where the number of hidden layers is either bounded by a constant or by some logarithmic factor in the sample size. From an approximation theoretical point of view, we derive two new error bounds for the approximation of (p, C)-smooth functions by (very wide or very deep) neural networks using the ReLU activation function, which are essential to show our convergence result. In particular, we generalize the approximation result from [36] from Hölder-smooth to (p, C)-smooth functions. Compared to previous works based on sparse neural network estimates, our result does not focus on the number of nonzero parameters but on the overall number of parameters in the network. In particular, we show that in case of networks with constant width and W weights we can achieve an approximation error of size $W^{-2p/d}$ instead of $W^{-p/d}$ as stated in [5] and [31]. By bounding the number of parameters in this sense, the topology of our neural networks is much easier in view of an implementation of the corresponding least squares estimate. For instance, as shown in Listing 1, using Python's packages `tensorflow` and `keras` enable us an easy and fast implementation. Although sparsely connected networks are often preferred in practical applications, there are some open questions about an efficient implementation of these networks. So-called pruning methods, for instance, start with large strongly connected neural networks and delete redundant parameters during the training process. The main drawback is that due to the large initial

```
model = Sequential()
model.add(Dense(d, activation="relu", input_shape=(d,)))
for i in np.arange(L):
    model.add(Dense(K, activation="relu"))
model.add(Dense(1))
model.compile(optimizer="adam",
                    loss="mean_squared_error")
model.fit(x=x_learn,y=y_learn)
```

LISTING 1. *Python code for fitting of fully connected neural networks to data* x_{learn} *and* y_{learn}.

size of the networks the computational costs of the method are high. That is why the implementation of sparsely connected networks is critically questioned (see, e.g., [9, 23]). With regard to our convergence result, we analyze a slightly more general function space, which includes all the other types of structures of m mentioned earlier.

Independent of us, [37] published a similar result for the approximation of smooth functions by simple fully connected deep neural networks. For a network with width $2d + 10$ and W weights, they also showed an approximation rate of $W^{-2p/d}$. After the original version of our paper, a related arXiv article was uploaded by [24]. Therein our approximation result, where either width or depth are varied, was generalized to ReLU networks where both width and depth are varied simultaneously.

1.5. *Notation.* Throughout the paper, the following notation is used: The sets of natural numbers and real numbers are denoted by $\mathbb{N}$ and $\mathbb{R}$, respectively. Furthermore, we set $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. For $z \in \mathbb{R}$, we denote the smallest integer greater than or equal to z by $\lceil z \rceil$ and the largest integer smaller or equal to z by $\lfloor z \rfloor$. We set $z_+ = \max\{z, 0\}$. Vectors are denoted by bold letters, for example, $\mathbf{x} = (x^{(1)}, \dots, x^{(d)})^T$. We define $\mathbf{1} = (1, \dots, 1)^T$ and $\mathbf{0} = (0, \dots, 0)^T$. A d-dimensional multiindex is a d-dimensional vector $\mathbf{j} = (j^{(1)}, \dots, j^{(d)})^T \in \mathbb{N}_0^d$. As usual, we define $\|\mathbf{j}\|_1 = j^{(1)} + \cdots + j^{(d)}$, $\mathbf{j}! = j^{(1)}! \cdots j^{(d)}!$,

$$\mathbf{x}^{\mathbf{j}} = \left(x^{(1)}\right)^{j^{(1)}} \cdots \left(x^{(d)}\right)^{j^{(d)}} \quad \text{and} \quad \partial^{\mathbf{j}} = \frac{\partial^{j^{(1)}}}{\partial (x^{(1)})^{j^{(1)}}} \cdots \frac{\partial^{j^{(d)}}}{\partial (x^{(d)})^{j^{(d)}}}.$$

Let $D \subseteq \mathbb{R}^d$ and let $f : \mathbb{R}^d \to \mathbb{R}$ be a real-valued function defined on $\mathbb{R}^d$. We write $\mathbf{x} = \arg\min_{\mathbf{z} \in D} f(\mathbf{z})$ if $\min_{\mathbf{z} \in \mathcal{D}} f(\mathbf{z})$ exists and if $\mathbf{x}$ satisfies $\mathbf{x} \in D$ and $f(\mathbf{x}) = \min_{\mathbf{z} \in \mathcal{D}} f(\mathbf{z})$. The Euclidean and the supremum norms of $\mathbf{x} \in \mathbb{R}^d$ are denoted by $\|\mathbf{x}\|$ and $\|\mathbf{x}\|_\infty$, respectively. For $f : \mathbb{R}^d \to \mathbb{R}$,

$$\|f\|_\infty = \sup_{\mathbf{x} \in \mathbb{R}^d} |f(\mathbf{x})|$$

is its supremum norm, and the supremum norm of f on a set $A \subseteq \mathbb{R}^d$ is denoted by

$$\|f\|_{\infty, A} = \sup_{\mathbf{x} \in A} |f(\mathbf{x})|.$$

Furthermore, we define the norm $\|\cdot\|_{C^q(A)}$ of the smooth function space $C^q(A)$ by

$$\|f\|_{C^q(A)} := \max\left\{\left\|\partial^{\mathbf{j}} f\right\|_{\infty, A} : \|\mathbf{j}\|_1 \le q, \mathbf{j} \in \mathbb{N}^d\right\}$$

for any $f \in C^q(A)$. Let $\mathbf{z}_1, \dots, \mathbf{z}_n \in \mathbb{R}^d$, set $\mathbf{z}_1^n := (\mathbf{z}_1, \dots, \mathbf{z}_n)$, let $\mathcal{F}$ be a set of functions $f : \mathbb{R}^d \to \mathbb{R}$ and let $\epsilon > 0$. We denote by $\mathcal{N}_1(\epsilon, \mathcal{F}, \mathbf{z}_1^n)$ the $\epsilon - \|\cdot\|_1$-covering number on $\mathbf{z}_1^n$,

that is, the minimal number $N \in \mathbb{N}$ such that there exist functions $f_1, \dots, f_N : \mathbb{R}^d \to \mathbb{R}$ with the property that for every $f \in \mathcal{F}$ there is a $j = j(f) \in \{1, \dots, N\}$ such that

$$\frac{1}{n} \sum_{i=1}^{n} |f(\mathbf{z}_i) - f_j(\mathbf{z}_i)| < \epsilon.$$

For $u \in \mathbb{R}$ and $\beta > 0$, we define $T_\beta u = \max\{-\beta, \min\{\beta, u\}\}$. Furthermore, for $f : \mathbb{R}^d \to \mathbb{R}$ we define $T_\beta f : \mathbb{R}^d \to \mathbb{R}$ by $(T_\beta f)(\mathbf{x}) = T_\beta(f(\mathbf{x}))$. And if $\mathcal{F}$ is a set of functions $f : \mathbb{R}^d \to \mathbb{R}$ we set

$$T_\beta \mathcal{F} = \{T_\beta f : f \in \mathcal{F}\}.$$

1.6. *Outline.* The main result and its proof are presented in Section 2. Our new results concerning the approximation of (p, C)-smooth functions by deep neural networks are described in Section 3. Section 4 deals with a result concerning the approximation of hierarchical composition models (see Definition 3 below) by neural networks.

2. Main result. As already mentioned above, the only possible way to avoid the so-called curse of dimensionality is to restrict the underlying function class. We therefore consider functions, which fulfill the following definition.

DEFINITION 3. Let $d \in \mathbb{N}$ and $m : \mathbb{R}^d \to \mathbb{R}$ and let $\mathcal{P}$ be a subset of $(0, \infty) \times \mathbb{N}$.

(a) We say that m satisfies a hierarchical composition model of level 0 with order and smoothness constraint $\mathcal{P}$, if there exists a $K \in \{1, \dots, d\}$ such that

$$m(\mathbf{x}) = x^{(K)} \quad \text{for all } \mathbf{x} = \big(x^{(1)}, \dots, x^{(d)}\big)^\top \in \mathbb{R}^d.$$

(b) We say that m satisfies a hierarchical composition model of level $l + 1$ with order and smoothness constraint $\mathcal{P}$, if there exist $(p, K) \in \mathcal{P}$, $C > 0$, $g : \mathbb{R}^K \to \mathbb{R}$ and $f_1, \dots, f_K : \mathbb{R}^d \to \mathbb{R}$, such that g is (p, C)-smooth, $f_1, \dots, f_K$ satisfy a hierarchical composition model of level l with order and smoothness constraint $\mathcal{P}$ and

$$m(\mathbf{x}) = g\big(f_1(\mathbf{x}), \dots, f_K(\mathbf{x})\big) \quad \text{for all } \mathbf{x} \in \mathbb{R}^d.$$

For $l = 1$ and some order and smoothness constraint $\mathcal{P} \subseteq (0, \infty) \times \mathbb{N}$, our space of hierarchical composition models becomes

$$\begin{aligned} \mathcal{H}(1, \mathcal{P}) = \big\{ & h : \mathbb{R}^d \to \mathbb{R} : h(\mathbf{x}) = g\big(x^{(\pi(1))}, \dots, x^{(\pi(K))}\big), \text{ where} \\ & g : \mathbb{R}^K \to \mathbb{R} \text{ is } (p, C)\text{-smooth for some } (p, K) \in \mathcal{P}, \\ & C > 0 \text{ and } \pi : \{1, \dots, K\} \to \{1, \dots, d\}\big\}. \end{aligned}$$

For $l > 1$, we recursively define

$$\begin{aligned} \mathcal{H}(l, \mathcal{P}) := \big\{ & h : \mathbb{R}^d \to \mathbb{R} : h(\mathbf{x}) = g\big(f_1(\mathbf{x}), \dots, f_K(\mathbf{x})\big), \text{ where} \\ & g : \mathbb{R}^K \to \mathbb{R} \text{ is } (p, C)\text{-smooth for some } (p, K) \in \mathcal{P}, \\ & C > 0 \text{ and } f_i \in \mathcal{H}(l - 1, \mathcal{P})\big\}. \end{aligned}$$

In practice, it is conceivable, that there exist input–output relationships, which can be described by a regression function contained in $\mathcal{H}(l, \mathcal{P})$. Particulary, our assumption is motivated by applications in connection with complex technical systems, which are constructed in a modular form. Here, each modular part can be again a complex system, which also explains the recursive construction in Definition 3. It is shown in [5] and in [31] that the function

classes used therein generalize all other models mentioned in our article. As the function class of [5] (see Definition 2) forms some special case of $\mathcal{H}(l, \mathcal{P})$ in form of an alternation between summation and composition, this is also true for our more general model. Compared to the function class studied in [31], our definition forms a slight generalization, since we allow different smoothness and order constraints within the same level in the composition. In particular, also the additional examples mentioned in [31] are contained in our function class.

Our main result is the following theorem.

THEOREM 1. *Let* $(\mathbf{X}, Y), (\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$ *be independent and identically distributed random values such that* $\operatorname{supp}(\mathbf{X})$ *is bounded and*

$$\mathbf{E}\{\exp(c_1 \cdot Y^2)\} < \infty$$

for some constant $c_1 > 0$. *Let the corresponding regression function* m *be contained in the class* $\mathcal{H}(l, \mathcal{P})$ *for some* $l \in \mathbb{N}$ *and* $\mathcal{P} \subseteq [1, \infty) \times \mathbb{N}$. *Each function* g *in the definition of* m *can be of different smoothness* $p_g = q_g + s_g$ ($q_g \in \mathbb{N}_0$ *and* $s_g \in (0, 1]$) *and of different input dimension* K_g, *where* $(p_g, K_g) \in \mathcal{P}$. *Denote by* $K_{\max}$ *the maximal input dimension and by* $p_{\max}$ *the maximal smoothness of one of the functions* g. *Assume that for each* g *all partial derivatives of order less than or equal to* q_g *are bounded, that is,*

$$\|g\|_{C^{q_g}(\mathbb{R}^d)} \leq c_2$$

for some constant $c_2 > 0$ *and that* $p_{\max}, K_{\max} < \infty$. *Let each function* g *be Lipschitz continuous with Lipschitz constant* $C_{\text{Lip}} \geq 1$. *Let* $\tilde{m}_n$ *be defined as in* (3) *for some* $L_n, r_n \in \mathbb{N}$, *and define* $m_n = T_{c_3 \cdot \log(n)} \tilde{m}_n$ *for some* $c_3 > 0$ *sufficiently large.*

(a) *Choose* $c_4, c_5 > 0$ *sufficiently large and set*

$$L_n = \lceil c_4 \cdot \log n \rceil \quad \textit{and} \quad r_n = \left\lceil c_5 \cdot \max_{(p,K) \in \mathcal{P}} n^{\frac{K}{2(2p+K)}} \right\rceil.$$

Then

$$\mathbf{E} \int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x}) \leq c_6 \cdot (\log n)^6 \cdot \max_{(p,K) \in \mathcal{P}} n^{-\frac{2p}{2p+K}}$$

holds for sufficiently large n.

(b) *Choose* $c_7, c_8 > 0$ *sufficiently large and set*

$$L_n = \left\lceil c_7 \cdot \max_{(p,K) \in \mathcal{P}} n^{\frac{K}{2(2p+K)}} \cdot \log n \right\rceil \quad \textit{and} \quad r_n = r = \lceil c_8 \rceil.$$

Then

$$\mathbf{E} \int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x}) \leq c_9 \cdot (\log n)^6 \cdot \max_{(p,K) \in \mathcal{P}} n^{-\frac{2p}{2p+K}}$$

holds for sufficiently large n.

REMARK 1. Theorem 1 shows that in case that the regression function satisfies an hierarchical composition model with smoothness and order constraint $\mathcal{P}$ the L_2-errors of least squares neural network regression estimates based on a set of fully connected neural networks achieve the rate of convergence $\max_{(p,K) \in \mathcal{P}} n^{-2p/(2p+K)}$ (up to some logarithmic factor), which does not depend on d and which does therefore circumvent the so-called *curse of dimensionality*.

REMARK 2. In the special case that each hierarchical composition model of some level i ($i \in \{1, \ldots, l\}$) has some smoothness p_i and dimension K_i and that $K_l \leq K_{l-1} \leq \cdots \leq K_1 \leq d$, the rate of convergence in Theorem 1 is optimal up to some logarithmic factor (see Theorem 3 in [31]).

REMARK 3. Due to the fact that some parameters in the definition of the estimator in Theorem 1 are usually unknown in practice, they have to be chosen in a data-dependent way. Out of a set of different numbers of hidden layers and neurons per layer, the best estimator is then chosen adaptively. One simple possibility to do this is to use the so-called *splitting of the sample* method; cf., for example, Section 2.4 and Chapter 7 in [12]. Here, the sample is split into a learning sample of size n_l and a testing sample of size n_t, where $n_l + n_t = n$ (e.g., $n_l \approx n/2 \approx n_t$); the estimator is computed for several different selections of width and depth using only the learning sample, the empirical L_2-risks of these estimators are then computed on the testing sample and finally the parameter value is chosen for which the empirical L_2-risk on the testing sample is minimal.

PROOF. (a) Standard bounds of empirical process theory (cf. Lemma 18 in Supplement B, [21]) lead to

$$\begin{aligned}&\mathbf{E}\int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x})\\ &\quad\le \frac{c_{10}\cdot(\log n)^2\cdot(\sup_{\mathbf{x}_1^n\in(\mathbb{R}^d)^n}\log(\mathcal{N}_1(\frac{1}{n\cdot c_3\cdot\log n}, T_{c_3\cdot\log(n)}\mathcal{F}(L_n,r_n),\mathbf{x}_1^n))+1)}{n}\\ &\qquad+2\inf_{f\in\mathcal{F}(L_n,r_n)}\int |f(\mathbf{x})-m(\mathbf{x})|^2\mathbf{P}_{\mathbf{X}}(d\mathbf{x}).\end{aligned}$$

Set

$$(\bar p,\bar K)\in\mathcal{P} \text{ such that } \quad (\bar p,\bar K) = \arg\min_{(p,K)\in\mathcal{P}}\frac{p}{K}.$$

The fact that $1/n^{c_{11}} \le 1/(n\cdot c_3\cdot\log n) \le c_3\cdot(\log n)/8$, $L_n \le c_{12}\cdot\log n$ and $r_n \le c_{13}\cdot n^{\frac{1}{2(\bar p/\bar K+1)}}$ holds for $c_{11}, c_{12}, c_{13} > 0$, allows us to apply Lemma 19 in Supplement B, [21], to bound the first summand by

$$\begin{aligned}&\frac{c_{10}\cdot(\log n)^2\cdot c_{14}\cdot(\log n)^3\cdot\log(c_{14}\cdot(\log n)\cdot n^{\frac{2}{(2(2\bar p/\bar K+1))}})\cdot c_{14}\cdot n^{\frac{1}{(2\bar p/\bar K+1)}}}{n}\\ &\quad\le \frac{c_{15}\cdot(\log n)^6\cdot n^{\frac{1}{2\bar p/\bar K+1}}}{n} \le c_{15}\cdot(\log n)^6\cdot n^{-\frac{2\bar p}{2\bar p+\bar K}}\end{aligned} \tag{4}$$

for a sufficiently large n. Regarding the second summand, we apply Theorem 3(a) (see below), where we choose

$$M_{j,i} = \lceil n^{\frac{1}{2(2p_j^{(i)}+K_j^{(i)})}}\rceil$$

(see Section 4 for the definition of $p_j^{(i)}$ and $K_j^{(i)}$). Set

$$a_n = (\log n)^{\frac{1}{4\cdot(p_{\max}+1)}}.$$

W.l.o.g. we assume $\operatorname{supp}(\mathbf{X}) \subseteq [-a_n, a_n]^d$. Theorem 3(a) allows us to bound

$$\inf_{f\in\mathcal{F}(L_n,r_n)}\int |f(\mathbf{x})-m(\mathbf{x})|^2\mathbf{P}_{\mathbf{X}}(d\mathbf{x})$$

by

$$c_{16}\cdot\big(a_n^{4(p_{\max}+1)}\big)^2\cdot\max_{j,i}M_{j,i}^{-4p_j^{(i)}} = c_{16}\cdot(\log n)^2\cdot\max_{j,i}n^{-\frac{2p_j^{(i)}}{2p_j^{(i)}+K_j^{(i)}}}.$$

This together with (4) and the fact that

$$\max_{(p,K)\in\mathcal{P}} n^{-\frac{2p}{2p+K}} = n^{-\frac{2\bar{p}}{2\bar{p}+\bar{K}}} \geq \max_{j,i} n^{-\frac{2p_j^{(i)}}{2p_j^{(i)}+K_j^{(i)}}}$$

implies the assertion.

Part (b) follows by a slight modification of the proof of Theorem 1(a), where we use Theorem 3(b) (see below) instead of (a) to bound the approximation error. □

3. Approximation of smooth functions by fully connected deep neural networks with ReLU activation function. The aim of this section is to present a new result concerning the approximation of (p, C)-smooth functions by deep neural networks. This result is later needed in the proof of Theorem 3 (see below) to approximate hierarchical composition models by deep neural networks.

THEOREM 2. *Let $d \in \mathbb{N}$, let $f : \mathbb{R}^d \to \mathbb{R}$ be (p, C)-smooth for some $p = q + s$, $q \in \mathbb{N}_0$ and $s \in (0, 1]$, and $C > 0$. Let $a \geq 1$ and $M \in \mathbb{N}$ sufficiently large (independent of the size of a but*

$$M \geq 2 \quad \text{and} \quad M^{2p} \geq c_{17} \cdot \left(\max\{a, \|f\|_{C^q([-a,a]^d)}\}\right)^{4(q+1)}$$

must hold for some sufficiently large constant $c_{17} \geq 1$).

(a) *Let $L, r \in \mathbb{N}$ such that*

(i) $L \geq 5 + \lceil \log_4(M^{2p}) \rceil \cdot (\lceil \log_2(\max\{q, d\} + 1) \rceil + 1)$,
(ii) $r \geq 2^d \cdot 64 \cdot \binom{d+q}{d} \cdot d^2 \cdot (q+1) \cdot M^d$

hold. There exists a neural network $\hat{f}_{\text{wide}} \in \mathcal{F}(L, r)$ with the property that

$$\|f - \hat{f}_{\text{wide}}\|_{\infty,[-a,a]^d} \leq c_{18} \cdot \left(\max\{a, \|f\|_{C^q([-a,a]^d)}\}\right)^{4(q+1)} \cdot M^{-2p}. \tag{5}$$

(b) *Let $L, r \in \mathbb{N}$ such that*

(i) $L \geq 5M^d + \lceil \log_4(M^{2p+4\cdot d\cdot(q+1)} \cdot e^{4\cdot(q+1)\cdot(M^d-1)}) \rceil \cdot \lceil \log_2(\max\{q, d\} + 1) \rceil + \lceil \log_4(M^{2p}) \rceil$,
(ii) $r \geq 132 \cdot 2^d \cdot \lceil e^d \rceil \cdot \binom{d+q}{d} \cdot \max\{q + 1, d^2\}$

hold. There exists a neural network $\hat{f}_{\text{deep}} \in \mathcal{F}(L, r)$ such that (5) holds with $\hat{f}_{\text{wide}}$ replaced by $\hat{f}_{\text{deep}}$.

REMARK 4. The above result focuses on the convergence rate and no attempt has been made to minimize the constants in the definition of L and r.

The following corollary translates Theorem 2(b) in terms of the number of overall parameters versus the approximation accuracy.

COROLLARY 1. *Let $d \in \mathbb{N}$, let $f : \mathbb{R}^d \to \mathbb{R}$ be (p, C)-smooth for some $p = q + s$, $q \in \mathbb{N}_0$ and $s \in (0, 1]$, and $C > 0$. Let $a \geq 1$ and $\epsilon \in (0, 1)$. Then there exists a fully connected neural network $\hat{f}$ with $c_{19} \cdot \epsilon^{-d/(2p)}$ parameters, such that*

$$\|f - \hat{f}\|_{\infty,[-a,a]^d} \leq \epsilon.$$

PROOF. The number of overall weights W in a neural network with L hidden layers and r neurons per layer can be computed by

$$W = (d+1)\cdot r + (L-1)\cdot(r+1)\cdot r + (r+1).$$

Using Theorem 2(b), where we choose $M = \lceil c_{20}\cdot\epsilon^{-1/(2p)}\rceil$ for some constant $c_{20} > 0$, implies the assertion. □

REMARK 5. Compared with [31] and [5], where the total number of parameters is $c_{21}\cdot\epsilon^{-d/p}$ for some constant $c_{21} > 0$ in case of an approximation error of ϵ, Corollary 1 gives a quadratic improvement.

SKETCH OF THE PROOF OF THEOREM 2. The proof is deferred to Supplement A, [21]. The basic idea is to construct deep neural networks which approximate a piecewiece Taylor polynomial with respect to a partition of $[-a,a]^d$ into M^{2d} equivolume cubes. Our approximation starts on a coarse grid with M^d equivolume cubes and calculates the position of the cube C with $\mathbf{x}\in C$. This cube is then subpartitioned into M^d smaller cubes to finally compute the values of our Taylor polynomial on the finer grid with M^{2d} cubes. Parts (a) and (b) use a different approach to achieve this.

In part (a), we exploit the fact that a network with $c_{22}\cdot M^d$ neurons per layer has $c_{23}\cdot M^{2d}$ connections between two consecutive layers. Then each of the $c_{23}\cdot M^{2d}$ weights in our network is matched to one of the $c_{23}\cdot M^{2d}$ possible values of the derivatives of f. To detect the right values of the derivatives for our Taylor polynomial, we proceed in two steps: In the first two hidden layers, our network approximates the indicator function for every cube on the coarse grid. The output layer of those networks is then multiplied by the derivatives of f on the cube, respectively. And those values are the input of the $c_{24}\cdot M^d$ networks in the next two hidden layers, which approximate the indicator function multiplied by the values of the derivatives, respectively, on the M^d smaller cubes of the subpartition of C with $\mathbf{x}\in C$. Using this two-step approximation, we finally detect the right values of the derivatives on the M^{2d} equivolume cubes. In the remaining layers, we compute the Taylor polynomial.

In part (b) in the first $c_{25}\cdot M^d$ layers of the network the values of the derivatives of f necessary for the computation of a piecewise Taylor polynomial of f with respect to the partition on the coarse grid are determined. Then additional $c_{26}\cdot M^d$ layers of the network are used to compute a piecewise Taylor polynomial of f on the subpartition (into M^d smaller cubes) of the cube C with $\mathbf{x}\in C$ (where C is again one of the cubes of the coarse grid). Here, the values of the derivatives are computed successively by computing them one after another by a Taylor approximation using the previously computed values and suitably defined correction terms. □

4. Approximation of hierarchical composition models by neural networks. In this section, we show one of the key elements of the proof of Theorem 1, that is, a result concerning the approximation of hierarchical composition models with smoothness and order constraint $\mathcal{P}\subseteq[1,\infty)\times\mathbb{N}$ by deep neural networks. In order to formulate this result, we observe in a first step, that one has to compute different hierarchical composition models of some level i ($i\in\{1,\dots,l-1\}$) to compute a function $h_1^{(l)}\in\mathcal{H}(l,\mathcal{P})$. Let $\tilde{N}_i$ denote the number of hierarchical composition models of level i, needed to compute $h_1^{(l)}$. We denote in the following by

$$h_j^{(i)}:\mathbb{R}^d\to\mathbb{R} \tag{6}$$

the jth hierarchical composition model of some level i ($j\in\{1,\dots,\tilde{N}_i\}$, $i\in\{1,\dots,l\}$) that applies a $(p_j^{(i)},C)$-smooth function $g_j^{(i)}:\mathbb{R}^{K_j^{(i)}}\to\mathbb{R}$ with $p_j^{(i)} = q_j^{(i)} + s_j^{(i)}$, $q_j^{(i)}\in\mathbb{N}_0$ and

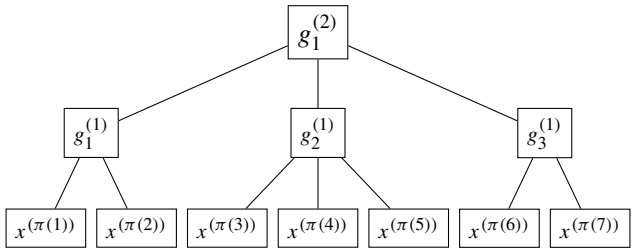

FIG. 2. *Illustration of a hierarchical composition model of the class* $\mathcal{H}(2,\mathcal{P})$ *with the structure* $h_1^{(2)}(\mathbf{x}) = g_1^{(2)}(h_1^{(1)}(\mathbf{x}), h_2^{(1)}(\mathbf{x}), h_3^{(1)}(\mathbf{x}))$, $h_1^{(1)}(\mathbf{x}) = g_1^{(1)}(x^{(\pi(1))}, x^{(\pi(2))})$, $h_2^{(1)}(\mathbf{x}) = g_2^{(1)}(x^{(\pi(3))}, x^{(\pi(4)}, x^{(\pi(5))})$ *and* $h_3^{(1)}(\mathbf{x}) = g_3^{(1)}(x^{(\pi(6))}, x^{(\pi(7))})$, *defined as in* (7) *and* (8).

$s_j^{(i)} \in (0,1]$, where $(p_j^{(i)}, K_j^{(i)}) \in \mathcal{P}$. The computation of $h_1^{(l)}(\mathbf{x})$ can then be recursively described as follows:

$$h_j^{(i)}(\mathbf{x}) = g_j^{(i)}\big(h^{(i-1)}_{\sum_{t=1}^{j-1} K_t^{(i)}+1}(\mathbf{x}), \ldots, h^{(i-1)}_{\sum_{t=1}^{j} K_t^{(i)}}(\mathbf{x})\big) \tag{7}$$

for $j \in \{1,\ldots,\tilde{N}_i\}$ and $i \in \{2,\ldots,l\}$ and

$$h_j^{(1)}(\mathbf{x}) = g_j^{(1)}\big(x^{(\pi(\sum_{t=1}^{j-1} K_t^{(1)}+1))}, \ldots, x^{(\pi(\sum_{t=1}^{j} K_t^{(1)}))}\big) \tag{8}$$

for some function $\pi : \{1,\ldots,\tilde{N}_1\} \to \{1,\ldots,d\}$. Furthermore for $i \in \{1,\ldots,l-1\}$ the recursion

$$\tilde{N}_l = 1 \quad \text{and} \quad \tilde{N}_i = \sum_{j=1}^{\tilde{N}_{i+1}} K_j^{(i+1)} \tag{9}$$

holds.

The exemplary structure of a function $h_1^{(2)} \in \mathcal{H}(2,\mathcal{P})$ is illustrated in Figure 2. Here, one can get a perception of how the hierarchical composition models of different levels are stacked on top of each other. The approximation result of such a function $h_1^{(l)}$ by a neural network is summarized in the following theorem.

THEOREM 3. *Let* $m : \mathbb{R}^d \to \mathbb{R}$ *be contained in the class* $\mathcal{H}(l,\mathcal{P})$ *for some* $l \in \mathbb{N}$ *and* $\mathcal{P} \subseteq [1,\infty) \times \mathbb{N}$. *Let* $\tilde{N}_i$ *be defined as in* (9). *Each* m *consists of different functions* $h_j^{(i)}$ $(j \in \{1,\ldots,\tilde{N}_i\}, i \in \{1,\ldots,l\})$ *defined as in* (6), (7) *and* (8). *Assume that the corresponding functions* $g_j^{(i)}$ *are Lipschitz continuous with Lipschitz constant* $C_{\mathrm{Lip}} \geq 1$ *and satisfy*

$$\big\|g_j^{(i)}\big\|_{C^{q_j^{(i)}}(\mathbb{R}^d)} \leq c_{27}$$

for some constant $c_{27} > 0$. *Denote by* $K_{\max} = \max_{i,j} K_j^{(i)} < \infty$ *the maximal input dimension and by* $p_{\max} = \max_{i,j} p_j^{(i)} < \infty$ *the maximal smoothness of the functions* $g_j^{(i)}$. *Let* $a \geq 1$ *and* $M_{j,i} \in \mathbb{N}$ *sufficiently large* (*each independent of the size of* a, *but* $\min_{j,i} M_{j,i}^2 > c_{28} \cdot a^{4(p_{\max}+1)}/(2^l K_{\max} C_{\mathrm{Lip}})^l$ *must hold for some constant* $c_{28} \geq 1$ *sufficiently large*).

(a) *Let* $L, r \in \mathbb{N}$ *be such that*

(i) $L \geq l \cdot (5 + \lceil \log_4(\max_{j,i} M_{j,i}^{2p_j^{(i)}}) \rceil \cdot (\lceil \log_2(\max\{K_{\max}, p_{\max}\}+1) \rceil + 1))$,

(ii) $r \geq \max_{i \in \{1,\ldots,l\}} \sum_{j=1}^{\tilde{N}_i} 2^{K_j^{(i)}} \cdot 64 \cdot \binom{K_j^{(i)}+q_j^{(i)}}{K_j^{(i)}} \cdot (K_j^{(i)})^2 \cdot (q_j^{(i)}+1) \cdot M_{j,i}^{K_j^{(i)}}$

hold. Then there exists a neural network t_1 of the network class $\mathcal{F}(L,r)$ with the property that

$$\|t_1 - m\|_{\infty,[-a,a]^d} \le c_{29} \cdot a^{4(p_{\max}+1)} \cdot \max_{j,i} M_{j,i}^{-2p_j^{(i)}}. \tag{10}$$

(b) *Let $L, r \in \mathbb{N}$ such that*

(i) $L \ge \sum_{i=1}^{l} \sum_{j=1}^{\tilde{N}_i} (5M_{j,i}^{K_j^{(i)}} + \lceil \log_4(M_{j,i}^{2p_j^{(i)}+4\cdot K_j^{(i)}\cdot(q_j^{(i)}+1)} \cdot e^{4\cdot(q_j^{(i)}+1)\cdot(M_{j,i}^{K_j^{(i)}}-1)}) \rceil \cdot \lceil \log_2(\max\{K_j^{(i)}, q_j^{(i)}\}+1) \rceil + \lceil \log_4(M_{j,i}^{2p_j^{(i)}}) \rceil)$,

(ii) $r \ge 2\sum_{t=1}^{l-1} \tilde{N}_t + 2d + 132 \cdot 2^{K_{\max}} \cdot \lceil e^{K_{\max}} \rceil \cdot \binom{K_{\max}+\lceil p_{\max} \rceil}{K_{\max}} \cdot \max\{\lceil p_{\max} \rceil + 1, K_{\max}^2\}$

hold. Then there exists a neural network t_2 of the network class $\mathcal{F}(L,r)$ with the property that (10) holds with t_1 replaced by t_2.

In the construction of our network, we will compose smaller subnetworks to successively build the final network. For two networks $f \in \mathcal{F}(L_f, r_f)$ and $g \in \mathcal{F}(L_g, r_g)$ with $L_f, L_g, r_f, r_g \in \mathbb{N}$, the *composed* neural network $f \circ g$ is contained in the function class $\mathcal{F}(L_f + L_g, \max\{r_f, r_g\})$. In the literature (see, e.g., [31]), the composition of two networks is often defined by $f \circ \sigma(g)$. Thus for every composition an additional layer is added. We follow a different approach. Instead of using an additional layer, we "melt" the weights of both networks f and g to define $f \circ g$. The following example clarifies our idea: Let

$$f(x) = \beta_f \cdot \sigma(\alpha_f \cdot x) \quad \text{and} \quad g(x) = \beta_g \cdot \sigma(\alpha_g \cdot x) \quad \text{for } \alpha_f, \alpha_g, \beta_f, \beta_g \in \mathbb{R},$$

then we have

$$(f \circ g)(x) = f\big(g(x)\big) = \beta_f \cdot \sigma\big(\alpha_f \cdot \beta_g \cdot \sigma(\alpha_g \cdot x)\big).$$

Figure 3 illustrates our idea by the network representation as an acyclic graph. This clearly shows why we do not need an additional layer in our composed network.

PROOF. (a) The computation of the function $m(\mathbf{x}) = h_1^{(l)}(\mathbf{x})$ can be recursively described as in (7) and (8). The basic idea of the proof is to define a composed network, which approximately computes the functions $h_1^{(1)}, \dots, h_{\tilde{N}_1}^{(1)}, h_1^{(2)}, \dots, h_{\tilde{N}_2}^{(2)}, \dots, h_1^{(l)}$.

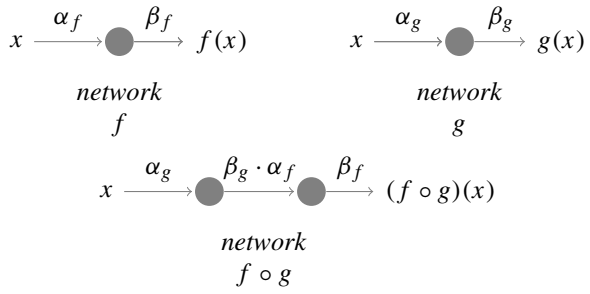

FIG. 3. *Illustration of the composed network $f \circ g$.*

For the approximation of $g_j^{(i)}$, we will use the networks

$$\hat{f}_{\text{wide},g_j^{(i)}} \in \mathcal{F}\big(L_0, r_j^{(i)}\big)$$

described in Theorem 2(a), where

$$L_0 = 5 + \Big\lceil \log_4\Big(\max_{j,i} M_{j,i}^{2p_j^{(i)}}\Big)\Big\rceil \cdot \big(\lceil \log_2(\max\{K_{\max}, p_{\max}\}+1)\rceil + 1\big)$$

and

$$r_j^{(i)} = 2^{K_j^{(i)}} \cdot 64 \cdot \binom{K_j^{(i)} + q_j^{(i)}}{K_j^{(i)}} \cdot \big(K_j^{(i)}\big)^2 \cdot \big(q_j^{(i)}+1\big) \cdot M_{j,i}^{K_j^{(i)}}$$

for $j \in \{1, \dots, \tilde{N}_i\}$ and $i \in \{1, \dots, l\}$.

To compute the values of $h_1^{(1)}, \dots, h_{\tilde{N}_1}^{(1)}$, we use the networks

$$\hat{h}_1^{(1)}(\mathbf{x}) = \hat{f}_{\text{wide},g_1^{(1)}}\big(x^{(\pi(1))}, \dots, x^{(\pi(K_1^{(1)}))}\big)$$

$$\vdots$$

$$\hat{h}_{\tilde{N}_1}^{(1)}(\mathbf{x}) = \hat{f}_{\text{wide},g_{\tilde{N}_1}^{(1)}}\big(x^{(\pi(\sum_{t=1}^{\tilde{N}_1-1} K_t^{(1)}+1))}, \dots, x^{(\pi(\sum_{t=1}^{\tilde{N}_1} K_t^{(1)}))}\big).$$

To compute the values of $h_1^{(i)}, \dots, h_{\tilde{N}_i}^{(i)}$ ($i \in \{2, \dots, l\}$). we use the networks

$$\hat{h}_j^{(i)}(\mathbf{x}) = \hat{f}_{\text{wide},g_j^{(i)}}\big(\hat{h}^{(i-1)}_{\sum_{t=1}^{j-1} K_t^{(i)}+1}(\mathbf{x}), \dots, \hat{h}^{(i-1)}_{\sum_{t=1}^{j} K_t^{(i)}}(\mathbf{x})\big)$$

for $j \in \{1, \dots, \tilde{N}_i\}$. Finally, we set

$$t_1(\mathbf{x}) = \hat{h}_1^{(l)}(\mathbf{x}).$$

Figure 4 illustrates the computation of the network t_1. It is easy to see that t_1 forms a composed network, where the networks $\hat{h}_1^{(i)}, \dots, \hat{h}_{\tilde{N}_i}^{(i)}$ are computed in parallel (i.e., in the

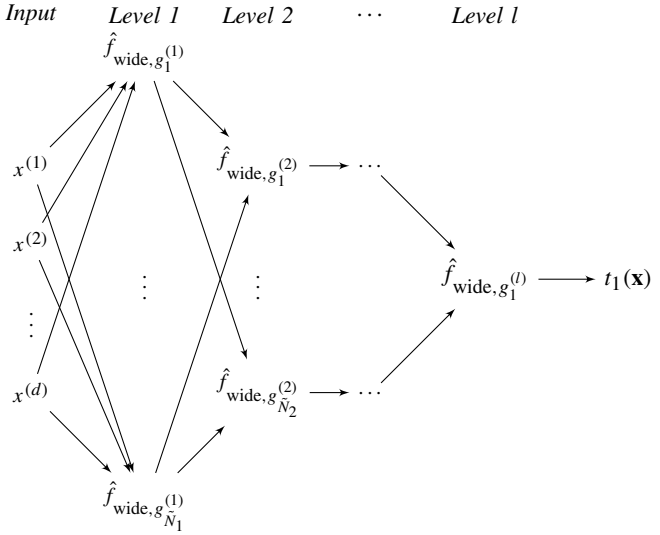

FIG. 4. *Illustration of the neural network* $t_1(\mathbf{x})$.

same layers) for $i \in \{1, \dots, l\}$, respectively. Since each $\hat{h}_j^{(i)}$ ($j \in \{1, \dots, \tilde{N}_i\}$) needs L_0 layers and $r_j^{(i)}$ neurons per layer, this network is contained in the class

$$\mathcal{F}\left(l \cdot L_0, \max_{i \in \{1,\dots,l\}} \sum_{j=1}^{\tilde{N}_i} r_j^{(i)}\right) \subseteq \mathcal{F}(L, r).$$

Here, we use that by successively applying $\hat{f}_{\mathrm{id}}$ to the output of the network t_1, we can easily enlarge the number of hidden layers in the network. Using induction on i, it is easy to see that t_1 satisfies

$$\|t_1 - m\|_{\infty,[-a,a]^d} \le c_{30} \cdot a^{4\cdot(p_{\max}+1)} \cdot \max_{j,i} M_{j,i}^{-2p_j^{(i)}}. \tag{11}$$

A complete proof can be found in Supplement A, [21]. This shows the assertion of the theorem.

(b) Denote $h_1^{(1)}, \dots, h_{\tilde{N}_1}^{(1)}, \dots, h_1^{(l-1)}, \dots, h_{\tilde{N}_{l-1}}^{(l-1)}, h_1^{(l)}$ by $h_1, h_2, \dots, h_{\sum_{t=1}^{l} \tilde{N}_t}$, such that

$$h_j^{(i)}(\mathbf{x}) = h_{N_j^{(i)}}(\mathbf{x}),$$

where

$$N_j^{(i)} = \sum_{t=1}^{i-1} \tilde{N}_t + j$$

for $i \in \{1, \dots, l\}$ and $j \in \{1, \dots, \tilde{N}_i\}$. Then we have

$$h_j(\mathbf{x}) = g_j^{(1)}\big(x^{(\pi(\sum_{t=1}^{j-1} K_t^{(1)}+1))}, \dots, x^{(\pi(\sum_{t=1}^{j} K_t^{(1)}))}\big) \tag{12}$$

for $j \in \{1, \dots, \tilde{N}_1\}$ and

$$h_{N_j^{(i)}}(\mathbf{x}) = g_j^{(i)}\big(h_{\sum_{t=1}^{j-1} K_t^{(i)}+1}(\mathbf{x}), \dots, h_{\sum_{t=1}^{j} K_t^{(i)}}(\mathbf{x})\big) \tag{13}$$

for $j \in \{1, \dots, \tilde{N}_i\}$ and $i \in \{2, \dots, l\}$.

In our neural network, we will compute $h_1, h_2, \dots, h_{\sum_{t=1}^{l} \tilde{N}_t}$ successively. In the construction of the network, each $g_j^{(i)}$ will be approximated by a network

$$\hat{f}_{\mathrm{deep}, g_j^{(i)}} \in \mathcal{F}\big(L_j^{(i)}, r_0\big)$$

described in Theorem 2(b) where

$$L_j^{(i)} = 5M_{j,i}^{K_j^{(i)}} + \lceil \log_4\big(M_{j,i}^{2p_j^{(i)}+4\cdot K_j^{(i)}\cdot(q_j^{(i)}+1)} \cdot e^{4\cdot(q_j^{(i)}+1)\cdot(M_{j,i}^{K_j^{(i)}}-1)}\big)\rceil$$

$$\cdot \big(\lceil \log_2\big(\max\{K_j^{(i)}, q_j^{(i)}\}+1\big)\rceil\big) + \lceil \log_4\big(M_{j,i}^{2p_j^{(i)}}\big)\rceil$$

and

$$r_0 = 132 \cdot 2^{K_{\max}} \cdot \lceil e^{K_{\max}} \rceil \cdot \binom{K_{\max} + \lceil p_{\max} \rceil}{K_{\max}} \cdot \max\{\lceil p_{\max} \rceil + 1, K_{\max}^2\}$$

with $M_{j,i} \in \mathbb{N}$ sufficiently large. Furthermore, we use the identity network

$$\hat{f}_{\mathrm{id}}(z) = \sigma(z) - \sigma(-z) = z$$

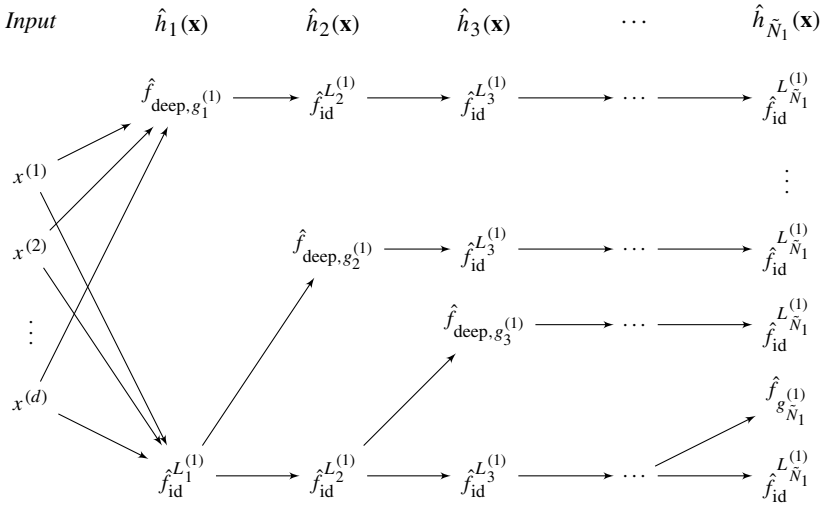

FIG. 5. *Illustration of the neural network, which computes* $h_1, \dots, h_{\tilde{N}_1}$.

with

$$\hat{f}_{\text{id}}^{0}(z) = z, \quad z \in \mathbb{R},$$
$$\hat{f}_{\text{id}}^{t+1}(z) = \hat{f}_{\text{id}}\big(\hat{f}_{\text{id}}^{t}(z)\big) = z, \quad z \in \mathbb{R}, t \in \mathbb{N}_0$$

and

$$\hat{f}_{\text{id}}^{t}\big(x^{(1)}, \dots, x^{(d)}\big) = \big(\hat{f}_{\text{id}}^{t}\big(x^{(1)}\big), \dots, \hat{f}_{\text{id}}^{t}\big(x^{(d)}\big)\big) = \big(x^{(1)}, \dots, x^{(d)}\big)$$

for $x^{(1)}, \dots, x^{(d)} \in \mathbb{R}$ to shift some values or vectors in the next hidden layers, respectively. We set

$$\tilde{L}_{N_j^{(i)}} = L_j^{(i)}$$

for $i \in \{1, \dots, l\}$ and $j \in \{1, \dots, \tilde{N}_i\}$.

Figure 5 illustrates how the functions $h_1, \dots, h_{\tilde{N}_1}$ are computed by our network and gives an idea of how the smaller networks are stacked on top of each other. The main idea is that we successively apply the network $\hat{f}_{\text{deep},g_j^{(i)}}$ in consecutive layers. Here, we make use of the identity network $\hat{f}_{\text{id}}$, which enables us to shift the input value as well as every already computed function in the next hidden layers without an error. As described in (12) and (13), our network successively computes

$$\hat{h}_i(\mathbf{x}) = \hat{g}_i^{(1)}(\mathbf{x}) := \hat{f}_{\text{deep},g_i^{(1)}}\big(x^{(\pi(\sum_{t=1}^{j-1} K_t^{(1)}+1))}, \dots, x^{(\pi(\sum_{t=1}^{j} K_t^{(1)}))}\big)$$

for $i \in \{1, \dots, \tilde{N}_1\}$ and

$$\hat{h}_{N_j^{(i)}}(\mathbf{x}) = \hat{f}_{\text{deep},g_j^{(i)}}\big(\hat{h}_{N^{(i-1)}_{\sum_{t=1}^{j-1} K_t^{(i)}+1}}(\mathbf{x}), \dots, \hat{h}_{N^{(i-1)}_{\sum_{t=1}^{j} K_t^{(i)}}}(\mathbf{x})\big)$$

for $i \in \{2, \dots, l\}$ and $j \in \{1, \dots, \tilde{N}_i\}$ Finally, we set

$$t_2(\mathbf{x}) = \hat{h}_{N_1^{(l)}}(\mathbf{x}) = \hat{h}_{\sum_{t=1}^{l} \tilde{N}_t}(\mathbf{x}).$$

Remark that for notational simplicity we have substituted every network $\hat{f}_{\text{id}}$ in the input of the functions $\hat{h}_j$ by the real value (since $\hat{f}_{\text{id}}$ computes this value without an error). Since

each network $\hat{h}_j$ for $j \in \{1, \dots, \sum_{t=1}^{l} \tilde{N}_t\}$ needs $\tilde{L}_j$ layers and r_0 neurons per layer, and we further need $2d$ neurons per layer to successively apply $\hat{f}_{\mathrm{id}}$ to the input $\mathbf{x}$ and 2 neurons per layer to apply $\hat{f}_{\mathrm{id}}$ to the at most $\sum_{t=1}^{l-1} \tilde{N}_t$ already computed functions in our network the final network t_2 is contained in the class

$$\mathcal{F}\left(\sum_{j=1}^{\sum_{t=1}^{l} \tilde{N}_t} \tilde{L}_j, 2 \cdot \sum_{t=1}^{l-1} \tilde{N}_t + 2d + r_0\right).$$

Using induction on i, it is easy to see that t_2 satisfies

$$\|t_2 - m\|_{\infty,[-a,a]^d} \le c_{31} \cdot a^{4(p_{\max}+1)} \cdot \max_{j,i} M_{j,i}^{-2p_j^{(i)}}. \tag{14}$$

A complete proof can be found in Supplement A, [21]. As described in part (a), we can easily enlarge the number of hidden layers by successively apply $\hat{f}_{\mathrm{id}}$ to the output of the network t_2. □

Acknowledgments. The authors are grateful to the many comments and suggestions that were brought up by the Associate Editor and four referees improving an early version of this manuscript.

SUPPLEMENTARY MATERIAL

Supplement A: Network approximation of smooth functions (DOI: 10.1214/20-AOS2034SUPPA; .pdf). This supplementary file contains the long and rather technical proof of Theorem 2 and the induction proofs of Theorem 3 that show the accuracy of the networks.

Supplement B: Auxiliary results and further proofs (DOI: 10.1214/20-AOS2034SUPPB; .pdf). This supplementary file contains the auxiliary results and further proofs of all lemmata that follow in a straightforward modification from earlier results.

REFERENCES

[1] ANTHONY, M. and BARTLETT, P. L. (2009). *Neural Network Learning*: *Theoretical Foundations*, 1st ed. Cambridge Univ. Press, New York, NY, USA.

[2] BARRON, A. R. (1991). Complexity regularization with application to artificial neural networks. In *Nonparametric Functional Estimation and Related Topics* (*Spetses*, 1990). *NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci.* **335** 561–576. Kluwer Academic, Dordrecht. MR1154352

[3] BARRON, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Inf. Theory* **39** 930–945. MR1237720 https://doi.org/10.1109/18.256500

[4] BARRON, A. R. (1994). Approximation and estimation bounds for artificial neural networks. *Mach. Learn.* **14** 115–133.

[5] BAUER, B. and KOHLER, M. (2019). On deep learning as a remedy for the curse of dimensionality in nonparametric regression. *Ann. Statist.* **47** 2261–2285. MR3953451 https://doi.org/10.1214/18-AOS1747

[6] DEVROYE, L., GYÖRFI, L. and LUGOSI, G. (1996). *A Probabilistic Theory of Pattern Recognition*. *Applications of Mathematics* (*New York*) **31**. Springer, New York. MR1383093 https://doi.org/10.1007/978-1-4612-0711-5

[7] DEVROYE, L. P. and WAGNER, T. J. (1980). Distribution-free consistency results in nonparametric discrimination and regression function estimation. *Ann. Statist.* **8** 231–239. MR0560725

[8] ELDAN, R. and SHAMIR, O. (2016). The power of depth for feedforward neural networks. In 29*th Annual Conference on Learning Theory* (V. Feldman, A. Rakhlin and O. Shamir, eds.). *Proc. Mach Learn. Res.* (*PMLR*) **49** 907–940. PMLR.

[9] EVCI, U., PEDREGOSA, F., GOMEZ, A. and ELSEN, E. (2019). The difficulty of training sparse neural networks. CoRR, abs/1906.10732.

[10] FRIEDMAN, J. H. and STUETZLE, W. (1981). Projection pursuit regression. *J. Amer. Statist. Assoc.* **76** 817–823. MR0650892

[11] GROHS, P., PEREKRESTENKO, D., ELBRÄCHTER, D. and BÖLCSKEI, H. (2019). Deep neural network approximation theory. *IEEE Trans. Inf. Theory*.

[12] GYÖRFI, L., KOHLER, M., KRZYŻAK, A. and WALK, H. (2002). *A Distribution-Free Theory of Nonparametric Regression*. *Springer Series in Statistics*. Springer, New York. MR1920390 https://doi.org/10.1007/b97848

[13] HÄRDLE, W., HALL, P. and ICHIMURA, H. (1993). Optimal smoothing in single-index models. *Ann. Statist.* **21** 157–178. MR1212171 https://doi.org/10.1214/aos/1176349020

[14] HÄRDLE, W. and STOKER, T. M. (1989). Investigating smooth multiple regression by the method of average derivatives. *J. Amer. Statist. Assoc.* **84** 986–995. MR1134488

[15] HAYKIN, S. (1998). *Neural Networks*: *A Comprehensive Foundation*, 2nd ed. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[16] HERTZ, J., KROGH, A. and PALMER, R. G. (1991). *Introduction to the Theory of Neural Computation*. *Santa Fe Institute Studies in the Sciences of Complexity. Lecture Notes*, *I*. Addison-Wesley, Redwood City, CA. With forewords by Jack Cowan and Christof Koch. MR1096298

[17] HOROWITZ, J. L. and MAMMEN, E. (2007). Rate-optimal estimation for a general class of nonparametric regression models with unknown link functions. *Ann. Statist.* **35** 2589–2619. MR2382659 https://doi.org/10.1214/009053607000000415

[18] IMAIZUMI, M. and FUKUMIZU, K. (2019). Deep neural networks learn non-smooth functions effectively. *Proc. Mach. Learn. Res.* **89** 869–878.

[19] KOHLER, M. and KRZYŻAK, A. (2005). Adaptive regression estimation with multilayer feedforward neural networks. *J. Nonparametr. Stat.* **17** 891–913. MR2192165 https://doi.org/10.1080/10485250500309608

[20] KOHLER, M. and KRZYŻAK, A. (2017). Nonparametric regression based on hierarchical interaction models. *IEEE Trans. Inf. Theory* **63** 1620–1630. MR3625984 https://doi.org/10.1109/TIT.2016.2634401

[21] KOHLER, M. and LANGER, S. (2021). Supplement to "On the rate of convergence of fully connected deep neural network regression estimates." https://doi.org/10.1214/20-AOS2034SUPPA, https://doi.org/10.1214/20-AOS2034SUPPB

[22] KONG, E. and XIA, Y. (2007). Variable selection for the single-index model. *Biometrika* **94** 217–229. MR2367831 https://doi.org/10.1093/biomet/asm008

[23] LIU, Z., SUN, M., ZHOU, T., HUANG, G. and DARRELL, T. (2018). Rethinking the value of network pruning. CoRR, abs/1810.05270.

[24] LU, J., SHEN, Z., YANG, H. and ZHANG, S. (2020). Deep network approximation for smooth functions. CoRR, arXiv:2001.03040.

[25] MCCAFFREY, D. F. and GALLANT, A. R. (1994). Convergence rates for single hidden layer feedforward networks. *Neural Netw.* **7** 147–158. https://doi.org/10.1016/0893-6080(94)90063-9

[26] MHASKAR, H. N. and POGGIO, T. (2016). Deep vs. shallow networks: An approximation theory perspective. *Anal. Appl. (Singap.)* **14** 829–848. MR3564936 https://doi.org/10.1142/S0219530516400042

[27] OHN, I. and KIM, Y. (2019). Smooth function approximation by deep neural networks with general activation functions. *Entropy* **21** Paper No. 627. MR3988437 https://doi.org/10.3390/e21070627

[28] OONO, K. and SUZUKI, T. (2019). Approximation and nonparametric estimation of ResNet-type cpnvolutional neural networks. In *Proceedings of the* 36*th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.) **97** 4922–4931. Proc. Mach. Learn. Res. (PMLR), Long Beach, CA, USA.

[29] RIPLEY, B. D. and HJORT, N. L. (1995). *Pattern Recognition and Neural Networks*, 1st ed. Cambridge Univ. Press, New York, NY, USA.

[30] SCHMIDHUBER, J. (2015). Deep learning in neural networks: An overview. *Neural Netw.* **61** 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

[31] SCHMIDT-HIEBER, J. (2020). Nonparametric regression using deep neural networks with ReLU activation function. *Ann. Statist.* **48** 1875–1897. MR4134774 https://doi.org/10.1214/19-AOS1875

[32] STONE, C. J. (1982). Optimal global rates of convergence for nonparametric regression. *Ann. Statist.* **10** 1040–1053. MR0673642

[33] STONE, C. J. (1985). Additive regression and other nonparametric models. *Ann. Statist.* **13** 689–705. MR0790566 https://doi.org/10.1214/aos/1176349548

[34] STONE, C. J. (1994). The use of polynomial splines and their tensor products in multivariate function estimation. *Ann. Statist.* **22** 118–184. With discussion by Andreas Buja and Trevor Hastie and a rejoinder by the author. MR1272079 https://doi.org/10.1214/aos/1176325361

[35] SUZUKI, T. (2019). Adaptivity of deep ReLU network for learning in Besov and mixed smooth Besov spaces: Optimal rate and curse of dimensionality. In *International Conference on Learning Representations*.

[36] YAROTSKY, D. (2018). Optimal approximation of continuous functions by very deep ReLU networks. *COLT* **75** 639–649.

[37] YAROTSKY, D. and ZHEVNERCHUK, A. (2019). The phase diagram of approximation rates for deep neural networks. CoRR, abs/1906.09477.

[38] YU, Y. and RUPPERT, D. (2002). Penalized spline estimation for partially linear single-index models. *J. Amer. Statist. Assoc.* **97** 1042–1054. MR1951258 https://doi.org/10.1198/016214502388618861