On the rate of convergence of fully connected very deep neural network regression estimates

Michael Kohler and Sophie Langer†
Fachbereich Mathematik, Technische Universität Darmstadt, Schloßgartenstr. 7, 64289 Darmstadt, Germany, email: kohler@mathematik.tu-darmstadt.de, langer@mathematik.tu-darmstadt.de

August 29, 2019

Abstract
Recent results in nonparametric regression show that deep learning, i.e., neural networks 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 they are not fully connected. In this paper we show that we can get similar results also for fully connected multilayer feedforward neural networks with ReLU activation functions, provided the number of neurons per hidden layer is fixed and the number of hidden layers tends to infinity for sample size tending to infinity. The proof is based on new approximation results concerning fully connected deep neural networks.

AMS classification: Primary 62G08; secondary 41A25, 82C32.

Key words and phrases: curse of dimensionality, deep learning, neural networks, non-parametric regression, rate of convergence.

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 Anthony and Bartlett (2009); Devroye et al. (1996); Györfi et al. (2002); Haykin (1998); Hertz et al. (1991); Ripley and Hjort (1995)). 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., Schmidhuber (2015) 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, e.g., Mhaskar and Poggio (2016); Eldan and Shamir (2016) and the literature cited therein for the analysis of corresponding approximation properties of neural networks.

Running title: On the rate of convergence of fully connected very deep neural network regression estimates

Corresponding author. Tel: +49-6151-16-23371
1.1 Nonparametric regression

In this paper we study these kind of estimates in connection with nonparametric regression. Here, \((X, Y)\) is an \(\mathbb{R}^d \times \mathbb{R}\)-valued random vector satisfying \(\mathbb{E}\{Y^2\} < \infty\), and given a sample of size \(n\) of \((X, Y)\), i.e., given a data set
\[
D_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\},
\]
where \((X, Y), (X_1, Y_1), \ldots, (X_n, Y_n)\) are i.i.d., the aim is to construct an estimate
\[
m_n(\cdot) = m_n(\cdot, D_n) : \mathbb{R}^d \rightarrow \mathbb{R}
\]
of the so-called regression function \(m : \mathbb{R}^d \rightarrow \mathbb{R}, m(x) = \mathbb{E}\{Y|X = x\}\) such that the so-called \(L_2\) error
\[
\int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx)
\]
is “small” (cf., e.g., Györfi et al. (2002) for a systematic introduction to nonparametric regression and a motivation for the \(L_2\) error).

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 \(f : \mathbb{R}^d \rightarrow \mathbb{R}\) by using neural networks. The starting point here is the choice of an activation function \(\sigma : \mathbb{R} \rightarrow \mathbb{R}\). Traditionally, so-called squashing functions are chosen as activation function \(\sigma : \mathbb{R} \rightarrow \mathbb{R}\), which are nondecreasing and satisfy \(\lim_{x \rightarrow -\infty} \sigma(x) = 0\) and \(\lim_{x \rightarrow \infty} \sigma(x) = 1\), e.g., the so-called sigmoidal or logistic squasher
\[
\sigma(x) = \frac{1}{1 + \exp(-x)}, \quad x \in \mathbb{R}. \tag{1}
\]
Recently, also unbounded activation functions are used, e.g., the ReLU–activation function
\[
\sigma(x) = \max\{x, 0\}.
\]
The network architecture \((L, k)\) depends on a positive integer \(L\) called the number of hidden layers and a width vector \(k = (k_1, \ldots, k_L) \in \mathbb{N}^L\) that describes the number of neurons in the first, second, \ldots, \(L\)-th hidden layer. A multilayer feedforward neural network with network architecture \((L, k)\) and ReLU activation function \(\sigma\) is a real-valued function defined on \(\mathbb{R}^d\) of the form
\[
f(x) = \sum_{i=1}^{k_L} c_{1,i}^{(L)} f_i^{(L)}(x) + c_{1,0}^{(L)} \tag{2}
\]
for some \(c_{1,0}^{(L)}, \ldots, c_{1,k_L}^{(L)} \in \mathbb{R}\) and for \(f_i^{(L)}\)'s recursively defined by
\[
f_i^{(s)}(x) = \sigma \left( \sum_{j=1}^{k_{i-1}} c_{i,j}^{(s-1)} f_j^{(s-1)}(x) + c_{i,0}^{(s-1)} \right) \tag{3}
\]
for some $c_{i,0}^{(s-1)}, \ldots, c_{i,k_{s-1}}^{(s-1)} \in \mathbb{R}$, $s \in \{2, \ldots, L\}$, and
\[
f_i^{(1)}(x) = \sigma \left( \sum_{j=1}^{d} c_{i,j}^{(0)} x^{(j)} + c_{i,0}^{(0)} \right)
\]
for some $c_{i,0}^{(0)}, \ldots, c_{i,d}^{(0)} \in \mathbb{R}$. The space of multilayer neural networks with $L$ hidden layers and $r$ neurons per layer is defined by
\[
\mathcal{F}(L, r) = \{ f : f \text{ is of the form (2) with } k_1 = k_2 = \ldots = k_L = r \}. \tag{5}
\]
Using these function spaces with some properly chosen number $L = L_n$ of hidden layers and number $r = r_n$ of neurons per hidden layer a neural network regression estimate can be defined by using the principle of least squares. To do this, one defines the neural network regression estimate as the minimizer of the so-called empirical $L^2$ risk over the function space $\mathcal{F}(L_n, r_n)$, which results in
\[
m_{n}(\cdot) = \arg \min_{f \in \mathcal{F}(L_n, r_n)} \frac{1}{n} \sum_{i=1}^{n} |f(X_i) - Y_i|^2.
\]
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.

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 non-trivial results on the rate of convergence (see, e.g., Theorem 7.2 and Problem 7.2 in Devroye et al. (1996) and Section 3 in Devroye and Wagner (1980)). 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 $0 < s \leq 1$. A function $m : \mathbb{R}^d \to \mathbb{R}$ is called $(p, C)$-smooth, if for every $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$ with $\sum_{j=1}^{d} \alpha_j = q$ the partial derivative $\partial^\alpha m / (\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d})$ exists and satisfies
\[
\left| \frac{\partial^\alpha m}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}(x) - \frac{\partial^\alpha m}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}(z) \right| \leq C \| x - z \|^s
\]
for all $x, z \in \mathbb{R}^d$, where $\| \cdot \|$ denotes the Euclidean norm.

Stone (1982) 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 Stone (1985, 1994) it is possible to circumvent this curse of dimensionality by imposing structural assumptions like additivity on the regression function. This is also used, e.g., in so-called single index models, in which
\[ m(x) = g(a^\top x), \quad x \in \mathbb{R}^d \]
is assumed to hold, where \( g : \mathbb{R} \to \mathbb{R} \) is a univariate function and \( a \in \mathbb{R}^d \) is a \( d \)-dimensional vector (see, e.g., Härdle et al. (1993); Härdle and Stoker (1989); Kong and Xia (2007); Yu and Ruppert (2002)). 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, i.e.,
\[ m(x) = \sum_{k=1}^K g_k(a_k^\top x), \quad x \in \mathbb{R}^d \]
for \( K \in \mathbb{N}, g_k : \mathbb{R} \to \mathbb{R} \) and \( a_k \in \mathbb{R}^d \) (see, e.g., Friedman and Stuetzle (1981)). 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 Györfi et al. (2002)).

Horowitz and Mammen (2007) studied the case of a regression function, which satisfies
\[
m(x) = g \left( \sum_{l_1=1}^{L_1} g_{l_1} \left( \sum_{l_2=1}^{L_2} g_{l_1,l_2} \left( \cdots \sum_{l_r=1}^{L_r} g_{l_1,...,l_r}(x^{l_1,...,l_r}) \right) \right) \right),
\]
where \( g, g_{l_1}, \ldots, g_{l_1,...,l_r} : \mathbb{R} \to \mathbb{R} \) are \((p, C)\)-smooth univariate functions and \( x^{l_1,...,l_r} \) are single components of \( x \in \mathbb{R}^d \) (not necessarily different for two different indices \((l_1, \ldots, l_r)\)). 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 networks regression estimates has been analyzed by Barron (1991, 1993, 1994); Bauer and Kohler (2019); Kohler and Krzyżak (2005, 2017); McCaffrey and Gallant (1994); Schmidt-Hieber (2019). For the \( L_2 \) error of a single hidden layer neural network, Barron (1994) 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 \( X \)). McCaffrey and Gallant (1994) showed a rate of \( n^{(-2p/(2p+d+5)) + \epsilon} \) for the \( L_2 \) error of suitably defined single hidden layer neural network estimate for \((p, C)\)-smooth functions, but their study was restricted to the use of a certain cosine squasher as the activation function.

The rate of convergence of neural network regression estimates based on two layer neural networks has been analyzed in Kohler and Krzyżak (2017). Therein, interaction models were studied, where the regression function satisfies
\[
m(x) = \sum_{I \subseteq \{1, \ldots, d\} \mid |I|=d^*} m_I(x_I), \quad x = (x^{(1)}, \ldots, x^{(d)})^\top \in \mathbb{R}^d
\]
for some \( d^* \in \{1, \ldots, d\} \) and \( m_I : \mathbb{R}^{d^*} \to \mathbb{R} (I \subseteq \{1, \ldots, d\}, |I| \leq d^*) \), where
\[
x_{i_1, \ldots, i_{d^*}}(i) = (x^{(i_1)}, \ldots, x^{(i_{d^*})}) \quad \text{for } 1 \leq i_1 < \ldots < i_{d^*} \leq d,
\]
and in case that all \( m_I \) are \((p, C)\)-smooth for some \( p \leq 1 \) it was shown that suitable neural network estimates 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 Kohler and Krzyżak (2017), 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 \( d \in \mathbb{N}, d^* \in \{1, \ldots, d\} \) 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 \( a_1, \ldots, a_{d^*} \in \mathbb{R}^d \) and \( f : \mathbb{R}^{d^*} \to \mathbb{R} \) such that
\[
m(x) = f(a_1^\top x, \ldots, a_{d^*}^\top x) \quad \text{for all } x \in \mathbb{R}^d.
\]

b) We say that \( m \) satisfies a generalized hierarchical interaction model of order \( d^* \) and level \( \ell + 1 \), if there exist \( K \in \mathbb{N}, g_k : \mathbb{R}^{d^*} \to \mathbb{R} \) \((k \in \{1, \ldots, K\})\) and \( f_{1,k}, \ldots, f_{d^*,k} : \mathbb{R}^d \to \mathbb{R} \) \((k \in \{1, \ldots, K\})\) such that \( f_{1,k}, \ldots, f_{d^*,k} \) satisfy a generalized hierarchical interaction model of order \( d^* \) and level \( \ell \) and
\[
m(x) = \sum_{k=1}^{K} g_k(f_{1,k}(x), \ldots, f_{d^*,k}(x)) \quad \text{for all } x \in \mathbb{R}^d.
\]

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 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 \( p \leq 1 \). Bauer and Kohler (2019) showed that this result even holds for \( p > 1 \) provided the squashing function is suitably chosen. Similar rate of convergence results as in Bauer and Kohler (2019) have been shown in Schmidt-Hieber (2019) for neural network regression estimates using the ReLU-activation function. Here slightly more general function spaces, which fulfill some composition assumption, were studied.

The main results in Bauer and Kohler (2019) and Schmidt-Hieber (2019) are new approximation results for neural networks. Here Schmidt-Hieber (2019) bounds the supremum norm error of the approximation of smooth functions on a cube, while the corresponding approximation bound in Bauer and Kohler (2019) 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 the neural networks are not fully connected, which makes the topology of the neural network difficult in view of an implementation of the estimate. In particular, in Schmidt-Hieber (2019) 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 result in this article

These results lead to the question whether it is really necessary in view of the derivation of good rates of convergence results for neural network regression estimates that the considered networks are not fully connected. In this articles we show that this is not the case. To do this, we derive similar rate of convergence results as in Bauer and Kohler (2019) and in Schmidt-Hieber (2019) for fully connected multilayer feedforward neural networks. In these networks the number of neurons per hidden layer is fixed and the number of hidden layers tends to infinity for sample size tending to infinity, so they are much deeper than the networks considered by Bauer and Kohler (2019) and Schmidt-Hieber (2019). From an approximation theoretical point of view we derive a new error bound for the approximation of \((p,C)\)-smooth functions by fully connected neural networks using the ReLU activation function, which is essential to show our convergence result. In particular, we generalize the approximation result from Yarotsky (2018) 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 non–zero parameters but on the overall number of parameters in the network. By bounding the number of parameters in this sense, the topology of our neural networks is much easier in view of an implementation. 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.

1.5 Notation

Throughout the paper, the following notation is used: The sets of natural numbers, natural numbers including 0 and real numbers are denoted by \(\mathbb{N}\), \(\mathbb{N}_0\) and \(\mathbb{R}\), respectively. 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 \(\lfloor z \rfloor\). Furthermore we set \(z_+ = \max\{z, 0\}\). 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 \(x = \arg\min_{z \in D} f(z)\) if \(\min_{z \in D} f(z)\) exists and if \(x\) satisfies \(x \in D\) and \(f(x) = \min_{z \in D} f(z)\). The Euclidean and the supremum norms of \(x \in \mathbb{R}^d\) are denoted by \(\|x\|\) and \(\|x\|_{\infty}\), respectively. For \(f: \mathbb{R}^d \to \mathbb{R}\)

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

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

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

We denote \(n\) fixed points in \(\mathbb{R}^d\) by \(z_1^n := (z_1, \ldots, z_n)\). Let \(\mathcal{F}\) be a set of functions \(f: \mathbb{R}^d \to \mathbb{R}\) and let \(\epsilon > 0\). We denote by \(N_1(\epsilon, \mathcal{F}, z_1^n)\) the \(\epsilon - \|\cdot\|_{1}\)-covering number, i.e. the minimal number \(N \in \mathbb{N}\) such that there exist functions \(f_1, \ldots, f_N: \mathbb{R}^d \to \mathbb{R}\) with the property that for every \(f \in \mathcal{F}\) there is a \(j = j(g) \in \{1, \ldots, N\}\) such that

\[
\frac{1}{n} \sum_{i=1}^{n} |f(z_i) - f_j(z_i)| < \epsilon.
\]
We define the truncation operator $T_\beta$ with level $\beta > 0$ as

$$T_\beta u = \begin{cases} u & \text{if } |u| \leq \beta \\ \beta \cdot \text{sign}(u) & \text{otherwise.} \end{cases}$$

1.6 Outline

The main result is presented in Section 2. The approximation of $(p, C)$–smooth functions by fully connected deep neural networks is analyzed in Section 3. Section 4 deals with a result concerning the approximation of hierarchical composition models (see Definition 3 below) by neural networks. Section 5 contains the proof of the main result.

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 $m : \mathbb{R}^d \to \mathbb{R}$.

a) We say that $m$ satisfies a hierarchical composition model of level $0$, if there exists a $K \in \{1, \ldots, d\}$ such that

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

b) We say that $m$ satisfies a hierarchical composition model of level $\ell + 1$, if there exist $K \in \mathbb{N}$, $g : \mathbb{R}^K \to \mathbb{R}$ and $f_1, \ldots, f_K : \mathbb{R}^d \to \mathbb{R}$, such that $f_1, \ldots, f_K$ satisfy a hierarchical composition model of level $\ell$ and

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

c) We say that a hierarchical composition model satisfies the smoothness and order constraint $\mathcal{P}$, where $\mathcal{P}$ is a subset of $(0, \infty) \times \mathbb{N}$, if in its definition all functions $g$ occurring in part b) satisfy $g : \mathbb{R}^K \to \mathbb{R}$ and $g (p, K)$–smooth for some $(p, K) \in \mathcal{P}$ and $C > 0$.

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

$$\mathcal{H}(1, \mathcal{P}) = \{ h : \mathbb{R}^d \to \mathbb{R} : h(x) = g(x^{(1)}, \ldots, x^{(K^{(1)})}), \text{where} \}
\begin{align*}
g : \mathbb{R}^{K^{(1)}} \to \mathbb{R} & \text{ is } (p^{(1)}, C)–\text{smooth for some } (p^{(1)}, K^{(1)}) \in \mathcal{P} \\
\text{and } \pi : \{1, \ldots, K^{(1)}\} & \to \{1, \ldots, d\} \}.\end{align*}$$

For $\ell > 1$, we recursively define

$$\mathcal{H}(\ell, \mathcal{P}) := \{ h : \mathbb{R}^d \to \mathbb{R} : h(x) = g(f_1(x), \ldots, f_{K^{(\ell)}}(x)), \text{where}$$
Let \( \sigma \) be described by a regression function contained in \( H \) and by input dimension random variables with values in \( \mathcal{P} \). Theorem 1.

In practice, it is conceivable, that there exist input–output–relationships, which can be described by a regression function contained in \( \mathcal{H}(\ell, \mathcal{P}) \). 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 recursiv construction in Definition 3. With regard to other function classes studied in the literature our function class generalizes previous results, as the function class of Bauer and Kohler (2019) (see Definition 2) forms some special case of \( \mathcal{H}(\ell, \mathcal{P}) \) in form of an alternation between summation and composition. Compared to the function class studied in Schmidt-Hieber (2019), our definition forms a slight generalization, since we allow different smoothness and order constraints within the same level in the composition. Our main result is the following theorem.

**Theorem 1.** Let \((X, Y), (X_1, Y_1), \ldots, (X_n, Y_n)\) be independent and identically distributed random variables with values in \( \mathbb{R}^d \times \mathbb{R} \) such that \( \text{supp}(X) \) is bounded and

\[
\mathbb{E}\{\exp(c_1 \cdot Y^2)\} < \infty
\]

for some constant \( c_1 > 0 \). Let the corresponding regression function \( m \) be contained in the class \( \mathcal{H}(\ell, \mathcal{P}) \) for some \( \ell \in \mathbb{N} \) 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 \( K_{\text{max}} \) the maximal input dimension and by \( p_{\text{max}} \) 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, i.e.,

\[
\max_{j_1, \ldots, j_{K_g} \in \{0, 1, \ldots, q_g\}} \max_{j_1 + \cdots + j_{K_g} \leq q_g} \left\| \frac{\partial^{j_1 \cdots j_{K_g}} g}{\partial^{j_1} x(1) \cdots \partial^{j_{K_g}} x(K_g)} \right\| \leq c_2
\]

for some constant \( c_2 > 0 \) and that \( p_{\text{max}}, K_{\text{max}} < \infty \). Let each \( g \) be Lipschitz continous with Lipschitz constant \( C_{\text{Lip}} \geq 1 \). Set

\[
(i) \quad L_n = \ell \cdot K_{\text{max}}^{\ell-1} \cdot \left( 4 \cdot \left[ \max_{(p, K) \in \mathcal{P}} \frac{n^{-K}}{2(2p+K)} \right] - 1 \right.
\]

\[
+ \log_4 \left( \left[ \max_{(p, K) \in \mathcal{P}} \frac{n^{-2p+4K+1}}{2(2p+K)} \right] \cdot \log_2 \left( \max\{K_{\text{max}}, p_{\text{max}}\} + 1 \right) \right)
\]

\[
(ii) \quad r_n = 2K_{\text{max}}^{\ell-1} + 2d + 2K_{\text{max}} \cdot \left( 4K_{\text{max}}^2 + 18K_{\text{max}} \right.
\]

\[
+ 2\left( K_{\text{max}} + p_{\text{max}} \right) + \left( 4\left[ e^{K_{\text{max}}} \right] + \max\left\{ 2K_{\text{max}}, 9p_{\text{max}} \right\} + 13 \right)
\]

Let \( \sigma : \mathbb{R} \to \mathbb{R} \) be the ReLU activation function \( \sigma(x) = \max\{x, 0\} \). Let \( \tilde{m}_n \) be the least squares estimate defined by

\[
\tilde{m}_n(\cdot) = \arg \min_{h \in \mathcal{F}(L_n, r_n)} \frac{1}{n} \sum_{i=1}^{n} |Y_i - h(X_i)|^2
\]
and define $m_n = T_{c_3 \log(n)} \tilde{n}_n$ for some $c_3 > 0$ sufficiently large. Then

$$E \int |m_n(x) - m(x)|^2 P_X(dx) \leq c_4 \cdot (\log(n))^{1/4} \max_{(p,K) \in \mathcal{P}} n^{-2p/(2p+K)}$$

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 multilayer feedforward neural networks with a fixed number of neurons per layer (corresponding to the hierarchical composition model) 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.** Due to the fact that some parameters in the definition of the estimate in Theorem 1 are normally 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 estimate is then chosen adaptively. Several possible methods and their effects can be found in Györfi et al. (2002).

### 3 Approximation of smooth functions by 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 fully connected very deep neural networks.

**Theorem 2.** Let $d \in \mathbb{N}$, let $f : \mathbb{R}^d \to \mathbb{R}$ be $(p,C)$-smooth for some $p = q + s$, $q \in \mathbb{N}$ 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 $M^2 \geq c_5 \cdot a^{4(q+1)}$ must hold for some sufficiently large constant $c_5 > 0$). Let $\sigma : \mathbb{R} \to \mathbb{R}$ be the ReLU activation function

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

Then there exists a neural network

$$f_{net} \in \mathcal{F}(L,r)$$

with

$$L = 4M^d - 1 + (2p + 4(q+1)d) \cdot \log_4(M) \cdot \lceil \log_2(\max\{d,q\} + 1) \rceil$$

hidden layers and at most

$$r = 2d \cdot \left( 4d^2 + 18d + 2 \left( \frac{d + q}{d} \right) \cdot (4|e^d| + \max\{2d,9q\} + 13) \right)$$

neurons per layer, such that

$$\sup_{x \in [-a,a]^d} |f(x) - f_{net}(x)| \leq c_6 \cdot a^{4(q+1)} \cdot M^{-2p}.$$
Outline of the proof. The overall idea of this proof is based on the approximation result from Yarotsky (2018). We generalize this result from Hölder-smooth functions to \((p,C)\)-smooth functions. As in Yarotsky (2018) the trick is to approximate the function first on a coarse grid and then to use a local refinement of the grid, where the approximation is constructed using correction terms encoded in a single number computed with a suitably chosen basis.

The complete proof of Theorem 2 is given in the supplement, now we explain its general idea for the one-dimensional case. We start by subdividing the interval \([-a, a]\) in \(M\) and \(M^2\) smaller equidistant intervals of sidelength \(2a/M\) and \(2a/M^2\) and define the corresponding partitions by

\[
P_1 = \{[a_0, a_1), [a_1, a_2), \ldots, [a_{M-1}, a_M]\}
\]

and

\[
P_2 = \{[b_0, b_1), [b_1, b_2), \ldots, [b_{M^2-1}, b_{M^2}]\}.
\]

The interval \(S\) of \(P_1\) and \(P_2\) satisfying \(x \in S\) is defined by \(S_{P_1}(x)\) and \(S_{P_2}(x)\) and the corresponding left border is denoted by \((S_{P_1}(x))_{\text{left}}\) and \((S_{P_2}(x))_{\text{left}}\), respectively. Our aim is to approximate a \((p,C)\)-smooth function by a fully connected neural network within \(c_7 \cdot M\) layers (constant \(c_7 > 0\)) and with an accuracy of order \(M^{2p}\). Our construction is motivated by the fact that the Taylor polynomial

\[
T_{f,q,(S_{P_2}(x))_{\text{left}}}(x) = \sum_{l=0}^{q} f^{(l)}((S_{P_2}(x))_{\text{left}}) \cdot \frac{(x-(S_{P_2}(x))_{\text{left}})^l}{l!}
\]

approximates our function \(f(x)\) (according to Lemma 1 in Kohler (2014)) with an error of size \(1/M^{2p}\). That is why one of the essential parts of our network approximate this Taylor polynomial. This part, denoted by \(f_{\text{net},P_2}(x)\), proceeds in two steps. In the first step it computes

\[
f^{(l)}((S_{P_1}(x))_{\text{left}}), \quad l \in \{0, 1, \ldots, q\}
\]

and suitably defined numbers

\[
b_k^{(l)} \in \mathbb{Z}, \quad |b_k^{(l)}| \leq e + 1, \quad l \in \{0, \ldots, q\}, k \in \{1, \ldots, M\},
\]

which depend on \(S_{P_1}(x)\).

In order to describe the construction we assume that \(S_{P_1}(x) = [a_i, a_{i+1})\) for some \(i \in \{0, \ldots, M - 1\}\) and we denote by \(I_{1,i}, \ldots, I_{M,i}\) those sets of \(P_2\) which are contained in \(S_{P_1}(x)\). Here we number the sets such that

\[
(I_{1,i})_{\text{left}} < (I_{2,i})_{\text{left}} < \cdots < (I_{M,i})_{\text{left}}
\]

holds.

In the second step \(f_{\text{net},P_2}\) computes successively approximations

\[
\tilde{f}^{(l)}((I_{k,i})_{\text{left}}), \quad l \in \{0, \ldots, q\}
\]
of

\[ f^{(l)}((I_{k,i})_{\text{left}}), \ l \in \{0, \ldots, q\} \]

for \( k \in \{1, \ldots, M\} \). To do this, we set

\[ \hat{f}^{(l)}((I_{1,i})_{\text{left}}) = f^{(l)}((S_{P_1}(x))_{\text{left}}), \ l \in \{0, \ldots, q\}. \]

By construction of the first step and \((I_{1,i})_{\text{left}} = (S_{P_1}(x))_{\text{left}}\) this estimate has error zero. As soon as we have computed the above estimates for some \( k \in \{1, \ldots, M - 1\} \) we use the Taylor polynomials with these coefficients around \((I_{k,i})_{\text{left}}\) in order to compute

\[ \sum_{s=0}^{q-l} \frac{\hat{f}^{(l+s)}(I_{k,i})_{\text{left}}}{s!} ((I_{k+1,i})_{\text{left}} - (I_{k,i})_{\text{left}})^s, \ l \in \{0, \ldots, q\} \]

and define

\[ \hat{f}^{(l)}((I_{k+1,i})_{\text{left}}) = \sum_{s=0}^{q-l} \frac{\hat{f}^{(l+s)}(I_{k,i})_{\text{left}}}{s!} ((I_{k+1,i})_{\text{left}} - (I_{k,i})_{\text{left}})^s + b_k^{(l)} \cdot c_8 \cdot \left( \frac{2a}{M^2} \right)^{p-l}, \]

\( l \in \{0, \ldots, q\} \). Choosing some suitable values of \( b_k^{(l)} \in \mathbb{Z} \) such that

\[ |b_k^{(l)}| \leq e + 1 \]

it can be shown that

\[ |\hat{f}^{(l)}((I_{k+1,i})_{\text{left}}) - f^{(l)}((I_{k+1,i})_{\text{left}})| \leq c_8 \cdot \left( \frac{2a}{M^2} \right)^{p-l} \]

holds for all \( l \in \{0, \ldots, q\} \). Observe that in this way we have defined the coefficients \( b_k^{(l)} \) for each interval \([a_i, a_{i+1})\). We will encode these coefficients for each \( i \in \{0, \ldots, M - 1\} \) and each \( l \in \{0, \ldots, q\} \) in the single number

\[ b_i^{(l)} = \sum_{k=0}^{M-1} (b_k^{(l)} + [e] + 2) \cdot (4 + 2[e])^{-k-1} \in [0, 1]. \]

Finally our neural networks \( f_{\text{net}, P_2} \) computes

\[ \hat{f}_{P_2}(x) = \sum_{l=0}^{q} \frac{\hat{f}^{(l)}((S_{P_2}(x))_{\text{left}})}{l!} (x - (S_{P_2}(x))_{\text{left}})^l, \]

where we use that by construction we have \( S_{P_2}(x) = I_{k,i} \) for some \( k \in \{1, \ldots, M\} \). It can be shown that we have

\[ |T_{f,q,(S_{P_2}(x))_{\text{left}}}(x) - \hat{f}_{P_2}(x)| \leq e \cdot c_8 \cdot \left( \frac{2a}{M^2} \right)^p, \]

(8)
and that it is therefore enough to show, that our network \( f_{\text{net}, P_2} \) approximates \( \hat{P}_2 \) up to an error of size \( 1/M^{2p} \).

Before we explain the detailed construction of the network \( f_{\text{net}, P_2} \), we introduce a corresponding network of functions. This network has two parts. In the first part we compute the approximation of \((S\tilde{P}_1(x))_{\text{left}}, f((S\tilde{P}_1(x))_{\text{left}}), \ldots, f^{(q)}((S\tilde{P}_1(x))_{\text{left}})\) on our coarse grid and the corresponding values of \( b_1^{(l)} \), where we assume \( S\tilde{P}_1(x) = [a_i, a_{i+1}) \) for some \( i \in \{0, \ldots, M - 1\} \). To do this, we define

\[
\text{out}_1^{(0)} = x \quad \text{and} \quad \text{out}_k^{(0)} = 0, \quad k \in \{2, \ldots, 4 + 2q\}
\]

For \( j \in \{1, \ldots, M\} \) we set

\[
\text{out}_1^{(j)} = \text{out}_1^{(j-1)},
\]

\[
\text{out}_2^{(j)} = a_{j-1} \cdot 1_{[a_{j-1}, a_j]}(\text{out}_1^{(j-1)}) + \text{out}_2^{(j-1)},
\]

\[
\text{out}_3^{(j)} = f^{(l)}(a_{j-1}) \cdot 1_{[a_{j-1}, a_j]}(\text{out}_1^{(j-1)}) + \text{out}_3^{(j-1)}
\]

\((l \in \{0, \ldots, q\})\) and

\[
\text{out}_4^{(j)} = b_1^{(l)} \cdot 1_{[a_{j-1}, a_j]}(\text{out}_1^{(j-1)}) + \text{out}_4^{(j-1)}
\]

\((l \in \{0, \ldots, q\})\). It is easy to see that for \( x \in [a_i, a_{i+1}) \), the above definitions imply

\[
\text{out}_1^{(M)} = x, \quad \text{out}_2^{(M)} = a_i, \quad \text{out}_3^{(M)} = f^{(l)}(a_i) \quad \text{and} \quad \text{out}_4^{(M)} = b_1^{(l)} \quad (9)
\]

for \( l \in \{0, \ldots, q\} \).

Next we describe the second part of our network of functions. For \( j \in \{1, \ldots, M - 1\} \) we set

\[
\text{out}_1^{(M+j)} = \text{out}_1^{(M+j-1)},
\]

\[
\text{out}_2^{(M+j)} = \text{out}_2^{(M+j-1)} + \frac{2a}{M^2},
\]

\[
\text{out}_3^{(M+j)} = \sum_{s=0}^{q-l} \frac{\text{out}_3^{(M+j-1)}}{s!} \cdot \left( \frac{2a}{M^2} \right)^s
\]

\[
\quad + \left( [4 + 2 \cdot [e] \cdot \text{out}_3^{(M+j-1)}] - [e] - 2 \right) \cdot c_8 \cdot \left( \frac{2a}{M^2} \right)^{p-l},
\]

\(l \in \{0, \ldots, q\}\),

\[
\text{out}_4^{(M+j)} = (4 + 2 \cdot [e]) \cdot \text{out}_4^{(M+j-1)} - [(4 + 2 \cdot [e]) \cdot \text{out}_4^{(M+j-1)}],
\]

\(l \in \{0, \ldots, q\}\),

\[
\text{out}_5^{(M+j)} = \text{out}_2^{(M+j-1)} + 2a/(M^2) \cdot \text{out}_1^{(M+j-1)} \cdot \text{out}_2^{(M+j-1)}
\]

\(l \in \{0, \ldots, q\}\).
\[
\text{out}^{(M+j-1)}_{5+2q+l} + \text{out}^{(M+j-1)}_{5+2q+l}
\]

\(l \in \{0, 1, \ldots, q+1\}\), where we set \(\text{out}^{(M)}_{5+2q+l} = 0\). Again it is easy to see that for

\[
x \in [a_i + j \cdot 2a/M^2, a_i + (j + 1) \cdot 2a/M^2)
\]

for some \(j \in \{0, \ldots, M-1\}\) we have

\[
\text{out}^{(2M-1)}_{5+2q} = a_i + j \cdot 2a/M^2
\]

and

\[
\text{out}^{(2M-1)}_{6+2q+l} = \tilde{f}^{(l)}(a_i + j \cdot 2a/M^2), \quad l \in \{0, \ldots, q\}.
\]

To compute the value of (7) we compute in a final step

\[
\text{out}^{(2M)}_{6+3q} = \sum_{l=0}^{q} \frac{\text{out}^{(2M-1)}_{6+2q+l}}{l!} \left(\text{out}^{(2M-1)}_{1} - \text{out}^{(2M-1)}_{5+2q}\right)^l.
\]

(10)

This network of functions can then be approximated by smaller blocks of neural networks. Here we use the neural networks

\[
f_{\text{id}}(x) = \sigma(x) + \sigma(-x) = x
\]

which computes the identity function without an error,

\[
f_{\text{ind},[a,\infty)}(x) = B_M \cdot \sigma(x-a) - B_M \cdot \sigma(x-a - \frac{1}{B_M}),
\]

which approximates the indicator function \(1_{[a,\infty)}(x)\) for some properly chosen \(B_M \in \mathbb{N}\),

\[
f_{\text{ind},[a,b]}(x) = B_M \cdot \sigma(x-a) - B_M \cdot \sigma(x-a - \frac{1}{B_M}) - B_M \cdot \sigma(x-b + \frac{1}{B_M}) + B_M \cdot \sigma(x-b)
\]

which approximates the indicator function \(1_{[a,b]}(x)\), and

\[
f_{\text{trunc},i}(x) = \sum_{j=1}^{4+2\lceil e \rceil} f_{\text{ind},i,[j,\infty)}(x), \quad i \in \{1, \ldots, M\}
\]

where

\[
f_{\text{ind},i,[j,\infty)}(x) = B_{M,i} \cdot \sigma(x-j) - B_{M,i} \cdot \sigma(x-j - \frac{1}{B_{M,i}}), \quad B_{M,i} \in \mathbb{N},
\]

which approximates \(\lfloor x \rfloor\) for \(x \in [0, \ldots, 5 + 2\lceil e \rceil]\).

To compute (10) it can be shown that there exists a network \(f_{\text{prod}}\) which approximates this value within \(c_9 \cdot \log_q(M)\) hidden layers and with an error of order \(1/M^{2p}\).
In order to define our neural network we start with
\[ \text{neur}_1^{(0)} = x \quad \text{and} \quad \text{neur}_k^{(0)} = 0, \quad k \in \{2, \ldots, 4 + 2q\}. \]

For \( j \in \{1, \ldots, M\} \) and \( l \in \{0, \ldots, q\} \) we set
\[
\text{neur}_1^{(j)} = \text{id}(\text{neur}_1^{(j-1)}), \\
\text{neur}_2^{(j)} = a_{j-1} \cdot \text{id}_{[a_{j-1}, a_j]}(\text{neur}_1^{(j-1)}) + \text{id}(\text{neur}_2^{(j-1)}), \\
\text{neur}_{3+l}^{(j)} = f(l) (a_{j-1} \cdot \text{id}_{[a_{j-1}, a_j]}(\text{neur}_1^{(j-1)}) + f(id)(\text{neur}_{3+l}^{(j-1)}),
\]
and
\[
\text{neur}_{4+q+l}^{(j)} = b_{j-1}^{(l)} \cdot \text{id}_{[a_{j-1}, a_j]}(\text{neur}_1^{(j-1)}) + f(id)(\text{neur}_{4+q+l}^{(j-1)}).
\]

Here it is easy to see that this construction leads to neural networks with \( M \) hidden layers.

For the second part of our network we define
\[
\text{id}_{[a_0, a_1]}(z) = z, \quad z \in \mathbb{R} \\
\text{id}_{1+1}(z) = f(id)(\text{id}(z)) = z, \quad t \in \mathbb{N}_0, z \in \mathbb{R}.
\]

Then we set for \( j \in \{1, \ldots, M - 1\} \)
\[
\text{neur}_1^{(M+j)} = f_2^2(\text{neur}_1^{(M+j-1)}), \\
\text{neur}_2^{(M+j)} = f_2^2 \left( \text{neur}_2^{(M+j-1)} + \frac{2a}{M^2} \right), \\
\text{neur}_{3+l}^{(M+j)} = f(id) \left( f(id) \left( \sum_{s=0}^{q-l} \frac{\text{neur}_{3+l+s}^{(M+j-1)}}{s!} \cdot \left( \frac{2a}{M^2} \right)^s \right) \\
+ \left( \text{trunc}_j((4 + 2 \cdot [e]) \cdot \text{neur}_{4+q+l}^{(M+j-1)}) - [e] - 2 \right) \cdot c_8 \cdot \left( \frac{2a}{M^2} \right)^{p-l} \right),
\]

\( l \in \{0, \ldots, q\}, \)
\[
\text{neur}_{4+q+l}^{(M+j)} = f(id) \left( (4 + 2 \cdot [e]) \cdot \text{neur}_{4+q+l}^{(M+j-1)} \right. \\
- \left. \text{trunc}_j((4 + 2 \cdot [e]) \cdot \text{neur}_{4+q+l}^{(M+j-1)}) \right),
\]

\( l \in \{0, \ldots, q\}. \) For the approximation of \( \text{out}_{5+2q+l}^{(M+j)}, \) \( l \in \{0, \ldots, q\} \), we exploit the fact that \( \sigma \) gets 0 in case that its input is negative. In particular, we set \( \text{neur}_{5+2q+l}^{(M)} = 0 \) and
\[
\text{neur}_{5+2q+l}^{(M+j)}
\]
\[ f_{id}(\text{neur}_{2+l}^{(M+j-1)}) - B_M \cdot \sigma \left( \text{neur}_{2}^{(M+j-1)} - \text{neur}_{1}^{(M+j-1)} \right) \]

\[ -B_M \cdot \sigma \left( \text{neur}_{1}^{(M+j-1)} - \text{neur}_{2}^{(M+j-1)} - (2a)/M^2 \right) \]

\[ -\sigma \left( -f_{id}(\text{neur}_{2+l}^{(M+j-1)}) - B_M \cdot \sigma \left( \text{neur}_{2}^{(M+j-1)} - \text{neur}_{1}^{(M+j-1)} \right) \right) \]

\[ -B_M \cdot \sigma \left( \text{neur}_{1}^{(M+j-1)} - \text{neur}_{2}^{(M+j-1)} - (2a)/M^2 \right) \]

\[ +f_{id}(\text{neur}_{5+2q+l}^{(M+j-1)}) \]  

\[ l \in \{0, \ldots, q+1\}, \] where one can see that the first two summands get zero for in case that \( x \) (contained in \( \text{neur}_{j}^{(j)} \) for all \( j \)) is not contained in \( [\text{neur}_{2}^{(M+j-1)} - 1/B_M, \text{neur}_{2}^{(M+j-1)} + 2a/M^2 + 1/B_M] \) for \( j \in \{1, \ldots, M-1\} \). Again it is easy to see, that this second part needs \( 2M - 2 \) layers.

The final approximation follows by applying \( f_{prod} \) to the values of \( \text{neur}_{1}^{(2M-1)} \) and \( \text{neur}_{5+2q+l}^{(2M-1)} \cdot l \in \{0, \ldots, q\} \). Summarizing this our network \( f_{net,P_2} \) needs \( 3M - 2 \) hidden layers and it can be shown that this network approximates (10) with an error of order \( 1/M^{2p} \) in case that \( x \) is not close to the borders of some interval.

One main throwback in the construction of \( f_{net,P_2} \) is, that the approximation of the indicator functions therein is not exact on the borders of the intervals. The construction of some network \( f_{ind,[a_{i-1},a_i]} \) is illustrated in Figure 1. Here one sees that the value gets linear in the small regions close to \( a_{i-1} \) and \( a_i \). As a consequence the value of our approximation gets bad in case that \( x \) lies close to one of the borders. In order to circumvent this phenomenon, we approximate two networks of the above described form in parallel. Here the second network uses a different partition, where each value in the intervals of \( P_1 \) and \( P_2 \) is shifted by \( 2a/M^2 \). Thus we have at least one network, that approximates the value of the corresponding Taylor polynomial with an error of size \( 1/M^{2p} \). By using a linear combination of both networks we are able to show that our final network approx-
imates the value of \( f(x) \) for all \( x \in [-a, a] \) with an error of size \( 1/M^{2p} \). Here the weight of each network is chosen in a way, that it gets small (of order \( 1/M^{2p} \)) in case that \( x \) is close to one of the borders of the underlying partition.

4 Approximation of hierarchical composition models by neural networks

The aim of this section is to prove a result concerning the approximation of hierarchical composition models with smoothness and order constraint \( \mathcal{P} \subseteq [1, \infty) \times \mathbb{N} \) by fully connected 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 \( i \) (\( i \in \{1, \ldots, \ell - 1\} \)) to compute a function \( h_1^{(\ell)} \in \mathcal{H}(\ell, \mathcal{P}) \). Let \( \tilde{N}_i \) denote the number of hierarchical composition models of level \( i \), needed to compute \( h_1^{(\ell)} \). We denote in the following by

\[
h_j^{(i)} : \mathbb{R}^d \to \mathbb{R}
\]

(12)

the \( j \)-th hierarchical composition model of some level \( i \) (\( j \in \{1, \ldots, \tilde{N}_i\}, i \in \{1, \ldots, \ell\} \)), 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 \( s_j^{(i)} \in (0, 1] \), where \( (p_j^{(i)}, K_j^{(i)}) \in \mathcal{P} \). The computation of \( h_1^{(\ell)}(x) \) can then be recursively described as follows:

\[
h_j^{(i)}(x) = g_j^{(i)} \left( \frac{h_1^{(i-1)}(x)}{\sum_{t=1}^{i-1} K_t^{(i)}+1} + \ldots + \frac{h_1^{(i-1)}(x)}{\sum_{t=1}^{i-1} K_t^{(i)}+1} \right)
\]

(13)

for \( j \in \{1, \ldots, \tilde{N}_i\} \) and \( i \in \{2, \ldots, \ell\} \) and

\[
h_j^{(1)}(x) = g_j^{(1)} \left( x^{(\pi(\sum_{t=1}^{i-1} K_t^{(1)}+1))} + \ldots + x^{(\pi(\sum_{t=1}^{i-1} K_t^{(1)}+1))} \right)
\]

(14)

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

\[
\tilde{N}_i = 1 \text{ and } \tilde{N}_i = \sum_{j=1}^{\tilde{N}_{i+1}} K_j^{(i+1)}
\]

(15)

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 \( h_1^{(\ell)} \) by a neural network is summarized in the following theorem:
Figure 2: Illustration of a hierarchical composition model of the class $\mathcal{H}(2, \mathcal{P})$ with the structure $h_1^{(2)}(x) = g_1^{(2)}(h_1^{(1)}(x), h_2^{(1)}(x), h_3^{(1)}(x))$, $h_1^{(1)}(x) = g_1^{(1)}(x^{(\pi(1))}, x^{(\pi(2))})$, $h_2^{(1)}(x) = g_2^{(1)}(x^{(\pi(3))}, x^{(\pi(4))})$ and $h_3^{(1)}(x) = g_3^{(1)}(x^{(\pi(6))}, x^{(\pi(7))})$, defined as in (13) and (14).

**Theorem 3.** Let $X$ be a $\mathbb{R}^d$-valued random variable and let $m : \mathbb{R}^d \rightarrow \mathbb{R}$ be contained in the class $\mathcal{H}(\ell, \mathcal{P})$ for some $\ell \in \mathbb{N}$ and $\mathcal{P} \subseteq [1, \infty) \times \mathbb{N}$. Let $\tilde{N}_i$ be defined as in (15). Each $m$ consists of different functions $h_j^{(i)}$ $(j \in \{1, \ldots, \tilde{N}_i\}, i \in \{1, \ldots, \ell\})$ defined as in (12), (13) and (14). Assume that the corresponding functions $g_j^{(i)}$ are Lipschitz continuous with Lipschitz constant $C_{\text{Lip}} \geq 1$ and satisfy

$$\max_{j_1, \ldots, j_{K_j^{(i)}} \in \{0, 1, \ldots, q_j^{(i)}\}} \left\| \frac{\partial^{j_1 + \cdots + j_{K_j^{(i)}}} g_j^{(i)}}{\partial x^{(1)} \cdots \partial x^{(K_j^{(i)})}} \right\|_\infty \leq c_{10}$$

for some constant $c_{10} > 0$. Denote by $K_{\text{max}} = \max_{i,j} K_j^{(i)} < \infty$ the maximal input dimension and by $p_{\text{max}} = \max_{i,j} p_j^{(i)} < \infty$ the maximal smoothness of the functions $g_j^{(i)}$.

Then, for any $a \geq 1$, $M_{j,i} \in \mathbb{N}$ sufficiently large (each independent of the size of $a$, but $\min_{j,i} M_{j,i} > c_{11} \cdot a^{4(p_{\text{max}}+1)}/(K_{\text{max}} C_{\text{Lip}})^\ell$ must hold for some constant $c_{11} > 0$ sufficiently large) and any

(i) $L \geq \sum_{i=1}^\ell \sum_{j=1}^{\tilde{N}_i} \left( 4M_{j,i}^{K_j^{(i)}} - 1 + (2p_j^{(i)} + 4q_j^{(i)} + 1) \cdot K_j^{(i)} \right) \cdot \log_4(M_{j,i}) \cdot \log_2(\max\{K_j^{(i)}, q_j^{(i)}\} + 1))$

(ii) $r \geq 2 \sum_{i=1}^{\ell-1} \tilde{N}_i + 2d + 2K_{\text{max}} \cdot \left( 4K_{\text{max}}^2 + 18K_{\text{max}} ight) + 2(K_{\text{max}}^2 + p_{\text{max}}) \cdot \left( 4\lfloor eK_{\text{max}} \rfloor + \max\{2K_{\text{max}}, 9p_{\text{max}}\} + 13 \right)$

a neural network $t(x)$ of the network class $\mathcal{F}(L, r)$ exists such that

$$|t(x) - m(x)| \leq c_{12} \cdot a^{4(p_{\text{max}}+1)} \cdot \max_{j,i} M_{j,i}^{-2p_j^{(i)}}$$
holds for all \( x \in [-a, a]^d \).

**Proof.** The computation of the function \( m(x) = h_1^{(\ell)}(x) \) can be recursively described as in (13) and (14).

Denote \( h_1^{(i)}, \ldots, h_{\tilde{N}_1}^{(i)}, \ldots, h_1^{(\ell-1)}, \ldots, h_{\tilde{N}_{\ell-1}}^{(\ell-1)}, h_1^{(\ell)} \) by \( h_1, h_2, \ldots, h_{\sum_{t=1}^{\ell} \tilde{N}_t} \), so that

\[
h_j^{(i)}(x) = h_{N_j^{(i)}}(x),
\]

where

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

for \( i \in \{1, \ldots, \ell\} \) and \( j \in \{1, \ldots, \tilde{N}_i\} \). Then we have

\[
h_j(x) = g_j^{(1)} \left( x^{(\sum_{t=1}^{j-1} K_t^{(1)} + 1)}, \ldots, x^{(\sum_{t=1}^{j} K_t^{(1)})} \right)
\]

for \( j \in \{1, \ldots, \tilde{N}_1\} \) and

\[
h_{N_j^{(i)}}(x) = g_j^{(i)} \left( h_{N_{j-1}^{(i-1)}}(x), \ldots, h_{N_{j-1}^{(i-1)}} \left( x^{(\sum_{t=1}^{j-1} K_t^{(i-1)} + 1)} \right) \right)
\]

for \( j \in \{1, \ldots, \tilde{N}_i\} \) and \( i \in \{2, \ldots, \ell\} \).

In our neural network we will compute \( h_1, h_2, \ldots, h_{\sum_{t=1}^{\ell} \tilde{N}_t} \) successively. Each \( g_j^{(i)} \) will be approximated by a network

\[
f_{\text{network,}g_j^{(i)}} \in \mathcal{F}(L_j^{(i)}, r_0)
\]

described in Theorem 2, where

\[
L_j^{(i)} = 4 M_{j,i}^{K_j^{(i)}} - 1 + (2p_j^{(i)} + 4(q_j^{(i)} + 1) \cdot K_j^{(i)}) \cdot \log_4(M_{j,i}) \cdot \lceil \log_2(\max\{K_j^{(i)}, q_j^{(i)}\} + 1) \rceil
\]

and

\[
r_0 = 2^{K_{\text{max}}} \cdot \left( 4K_{\text{max}}^2 + 18K_{\text{max}} + 2\left( K_{\text{max}} + p_{\text{max}} \right) \cdot K_{\text{max}} \right) \cdot (4e^{K_{\text{max}}} + \max\{2K_{\text{max}}, 9p_{\text{max}}\} + 13)
\]

18
with $M_{j,i} \in \mathbb{N}$ sufficiently large. For the computation of $h_1^{(\ell)}(x)$ we will then need, at least
\[ L_{\text{min}} = \sum_{t=1}^{\ell} \sum_{j=1}^{N_t} L_j^{(i)} \]
many hidden layers and $r = 2 \sum_{t=1}^{\ell-1} \tilde{N}_t + 2d + r_0$ many neurons in each hidden layer, which are numbered in the following by $1, \ldots, r$

In particular, we will save the value of $h_j(x)$ in the neuron
\[ 2(j-1) + 1 \text{ and } 2j, \]
$j \in \{1, \ldots, \sum_{t=1}^{\ell-1} \tilde{N}_t\}$, we will save the values of $x^{(1)}, \ldots, x^{(d)}$ in the neuron
\[ 2 \sum_{t=1}^{\ell-1} \tilde{N}_t + 1, \ldots, 2 \sum_{t=1}^{\ell-1} \tilde{N}_t + 2d, \]
and we will compute an approximation of $g_j^{(i)}$ in the neuron
\[ 2 \sum_{t=1}^{\ell-1} \tilde{N}_t + 2d + 1, \ldots, 2 \sum_{t=1}^{\ell-1} \tilde{N}_t + 2d + r_0. \]

In the sequel we define a fully connected feedforward neural network with $L \geq L_{\text{min}}$ hidden layers and $r$ neurons per hidden layer by (2), (3) and (4). Set
\[ g_{\text{max}} = \max \{ \max_{i \in \{1, \ldots, \ell\}} \max_{j \in \{1, \ldots, N_i, \tilde{N}_i\}} \|g_j^{(i)}\|_\infty, 1\}. \]

In the construction of our neural network, we will use the network
\[ f_{id}(x) = \sigma(x) - \sigma(-x) = x \]
for $x \in \mathbb{R}$ and the networks $f_{\text{network}, g_j^{(i)}}$ from Theorem 2 satisfying
\[ |f_{\text{network}, g_j^{(i)}}(x) - g_j^{(i)}(x)| \leq \frac{c_{13}}{(K_{\text{max}} C_{\text{Lip}})^{\ell}} \cdot a^{4(p_{\text{max}}+1)} \cdot \max_{j,i} M_{j,i}^{-2p_j^{(i)}} \]
for all $x \in [-2 \max\{g_{\text{max}}, a\}, 2 \max\{g_{\text{max}}, a\}]$ and some constant $c_{13} \geq c_6 2^{d(p_{\text{max}}+1)} g_{\text{max}}^{4(p_{\text{max}}+1)} (K_{\text{max}} C_{\text{Lip}})^{\ell}$. Remark that each $g_j^{(i)}$ satisfies the assumptions of $m$ in Theorem 2.

In the first step of the proof we show how to compute $h_1^{(\ell)}(x)$ with $L_{\text{min}}$ hidden layers. We start by computing $h_1, \ldots, h_{\tilde{N}_1}$ in the layers $1, \ldots, \sum_{t=1}^{\ell} L_t^{(1)}$. Here neurons (17)
in layer $\tau$ will be used to provide the values of the input variables $x^{(1)}, \ldots, x^{(d)}$ as input in layer $\tau + 1$, $\tau \in \{1, \ldots, \sum_{s=1}^{\tilde{N}_s} L_t^{(s)}\}$. To do this, we choose the corresponding weights according to the neural network $f_{id}$ from above applied successively to the output of the corresponding two neurons in the previous layer in case $\tau > 1$ and applied to $x^{(1)}, \ldots, x^{(d)}$ in case $\tau = 1$.

Neurons (18) in the layers $\sum_{t=1}^{\tilde{N}_1} L_t^{(1)} + 1, \ldots, \sum_{t=1}^{j-1} L_t^{(1)}$ are used to compute an approximation of $g_j^{(1)}$, $j \in \{1, \ldots, \tilde{N}_1\}$, applied to the output of the neurons (17) in layer $\sum_{t=1}^{j-1} L_t^{(1)}$ in case $j > 1$, and applied to the input in case $j = 1$. To achieve this, we choose the weights for these neurons according to the neural network $f_{network,g_j^{(1)}}$. The output of this network is used as input of $f_{id}$ computed in neurons (16) in layer $\sum_{t=1}^{j} L_t^{(1)} + 1$ and here we apply in these neurons in layer $\sum_{t=1}^{j-1} L_t^{(1)} + 2, \sum_{t=1}^{j-1} L_t^{(1)} + 3, \ldots, \sum_{t=1}^{\tilde{N}_1} L_t^{(1)}$ successively $f_{id}$ to the corresponding neurons in the previous layer.

![Diagram](https://via.placeholder.com/150)

**Figure 3:** Illustration of the neural network, which computes $h_1, \ldots, h_{\tilde{N}_1}$

Figure 3 illustrates the described structure of the network and gives an idea of how the neural networks are stacked on top of each other.

After layer $\sum_{t=1}^{\tilde{N}_1} L_t^{(1)}$ the same procedure is used for $h_{N_2^{(2)}}, h_{N_2^{(2)}}, \ldots, h_{N_1^{(1)}}$. The only difference is that $f_{network,g_j^{(i)}}$ gets as input the output of the neurons

$$2 \sum_{t=1}^{i-2} \tilde{N}_t + 2 \sum_{t=1}^{j-1} K_t^{(i)} + 1, \ldots, 2 \left( \sum_{t=1}^{i-2} \tilde{N}_t + \sum_{t=1}^{j} K_t^{(i)} \right)$$

in the layer $\sum_{s=1}^{i-1} \sum_{t=1}^{\tilde{N}_s} L_t^{(s)} + \sum_{t=1}^{j-1} L_t^{(i)}$ or in case that the computation of the corresponding input of $g_j^{(i)}$ is finished in the previous layer, also the sum of the outputs of the
neurons \((18)\). The output of the network is the output of \(f_{\text{network, } g_1^{(i)}}\) contained in the neurons \((18)\) in the last layer.

Set
\[
\tilde{L}_j^{(i)} = \sum_{s=1}^{i} \sum_{t=1}^{N_s} L_t^{(s)} + \sum_{t=1}^{j} L_t^{(i)}
\]
for \(i \in \{1, \ldots, \ell\}\) and \(j \in \{1, \ldots, \tilde{N}_i\}\), i.e., \(\tilde{L}_j^{(i)}\) is the layer, where in the above neural network the computation of \(h_j^{(i)}\) is completed. We define
\[
\begin{align*}
f_{id}^0(z) &= z, &z \in \mathbb{R}, \\
f_{id}^{t+1}(z) &= f_{id}(f_{id}^{t}(z)) = z, &z \in \mathbb{R}, \ t \in \mathbb{N}_0
\end{align*}
\]
and
\[
f_{id}^{t}(x^{(1)}, \ldots, x^{(d)}) = (f_{id}^{t}(x^{(1)}), \ldots, f_{id}^{t}(x^{(d)})) = (x^{(1)}, \ldots, x^{(d)}),
\]
\(x^{(1)}, \ldots, x^{(d)} \in \mathbb{R}\).

In the above neural network we compute the functions
\[
\hat{h}_j(x) = f_{\text{network, } g_1^{(1)}} \left( f_{id}^{\tilde{L}_j^{(1)}} \left( x^{(\sum_{l=1}^{\tilde{L}_j^{(i)}} K_l^{(i)})+1} \right) \right), \ldots, f_{id}^{\tilde{L}_j^{(i-1)}} \left( x^{(\sum_{l=1}^{\tilde{L}_j^{(i-1)}} K_l^{(i)})+1} \right) \right)
\]
for \(j \in \{1, \ldots, \tilde{N}_1\}\), where \(\tilde{L}_0^{(1)} = 0\), and
\[
\hat{h}_{N_j^{(i)}}(x) = f_{\text{network, } g_j^{(i)}} \left( f_{id}^{\tilde{L}_j^{(i-1)}} \left( x^{(\sum_{l=1}^{\tilde{L}_j^{(i-1)}} K_l^{(i)})+1} \hat{h}_{N_j^{(i-1)}}(x) \right) \right), \ldots, f_{id}^{\tilde{L}_j^{(i-1)}} \left( x^{(\sum_{l=1}^{\tilde{L}_j^{(i-1)}} K_l^{(i)})+1} \hat{h}_{N_j^{(0)}}(x) \right) \right)
\]
for \(i \in \{2, \ldots, \ell\}, \ j \in \{1, \ldots, \tilde{N}_i\}\). Here \(\hat{h}_{N_j^{(i)}}(x)\) is the output of the above neural network.

We show by induction that we have for all \(i \in \{1, \ldots, \ell\}\) and every \(j \in \{1, \ldots, \tilde{N}_i\}\)
\[
|\hat{h}_{N_j^{(i)}}(x) - h_{N_j^{(i)}}(x)| \leq \frac{2^i c_{13}}{(K_{\text{max}} \cdot C_{\text{Lip}})^{\ell-i}} \max_{j,i} M_{j,i}^{-2p_j^{(i)}}.
\]
(21)

By (20) and (19) we get for \(i = 1\) and every \(j \in \{1, \ldots, \tilde{N}_1\}\)
\[
|\hat{h}_{N_j^{(1)}}(x) - h_{N_j^{(1)}}(x)|
\]
and the Lipschitz continuity of $g$ now that (21) holds for some $\epsilon$, from which we can conclude for $\tilde{h}_{N+1,j} \ldots , \tilde{h}_{N+1,j}$,

$$
\left| f_{\text{network}, g^{(i)}} \left( f_{id}^{(i)} \left( x \left( \pi(\sum_{t=1}^{j-1} K_t^{(1)}) + 1 \right) \right) \right) \right|
\leq \frac{c_{13}}{(K_{\text{max}} C_{\text{Lip}})^{\ell}} \max_{j,i} M_{j,i}^{-2p_j^{(i)}}
$$

from which we can conclude for $\min_{j,i} M_{j,i}^{2p_j^{(i)}} > c_{13} a^{4(p_{\text{max}}+1)}/(K_{\text{max}} C_{\text{Lip}})^{\ell}$, that

$$
|\hat{h}_{N_{j+1}}(x)| \leq |\hat{h}_{N_{j}}(x) - h_{N_{j}}(x)| + g_{\text{max}} \leq 2g_{\text{max}}
$$

for $x \in [-a,a]^d$. Thus we have shown that (21) holds for $i = 1$ and that the output of each function $\hat{h}_{N_{j+1}}$ is contained in the interval, where inequality (19) holds. Assume now that (21) holds for some $i - 1$ and every $j \in \{1, \ldots, \tilde{N}_{i-1}\}$. Then

$$
|\hat{h}_{N_{j+1}}(x)| \leq |\hat{h}_{N_{j+1}}(x) - h_{N_{j+1}}(x)| + g_{\text{max}} \leq 2g_{\text{max}}
$$

follows directly by the induction hypothesis. This together with the induction hypothesis and the Lipschitz continuity of $g^{(i)}$ implies for $x \in [-a,a]^d$, $i \in \{2, \ldots, \ell\}$ and $j \in \{1, \ldots, \tilde{N}_{i}\}$

$$
\left| \hat{h}_{N_{j+1}}(x) - h_{N_{j+1}}(x) \right|
\leq \left| f_{\text{network}, g^{(i)}} \left( f_{id} \hat{h}_{N_{j+1}}^{(i-1)} + \sum_{i=1}^{j-1} K_t^{(1)} + 1 \right) \right|
\leq \left| f_{\text{network}, g^{(i)}} \left( h_{N_{j+1}}^{(i-1)} + \sum_{i=1}^{j-1} K_t^{(1)} + 1 \right) \right|
\leq \left| g_j^{(i)} \left( h_{N_{j+1}}^{(i-1)} + \sum_{i=1}^{j-1} K_t^{(1)} + 1 \right) \right|
$$

22
\[
\begin{align*}
&\leq \frac{c_{13}}{(K_{\text{max}} C_{\text{Lip}})^t} \cdot a^{4(p_{\text{max}} + 1)} \cdot \max_{j,i} M_{j,i}^{-2p_j(i)} \\
&\quad + K_j^{(i)} \cdot C_{\text{Lip}} \cdot \frac{2^{i-1}c_{13}}{(K_{\text{max}} C_{\text{Lip}})^{t-i+1}} \cdot a^{4(p_{\text{max}} + 1)} \cdot \max_{j,i} M_{j,i}^{-2p_j(i)} \\
&\leq \frac{2^{i} \cdot c_{13}}{(K_{\text{max}} C_{\text{Lip}})^{t-1}} \cdot a^{4(p_{\text{max}} + 1)} \cdot \max_{j,i} M_{j,i}^{-2p_j(i)}.
\end{align*}
\]

It is easy to see that there even exists a fully connected deep neural network with \( L \geq L_{\text{min}} \) hidden layers and \( r \) neurons per layer satisfying the error bound (21) for \( h_{N_1}^{(t)} \).

Therefore we apply successively the network \( f_{id} \) to the output of the network \( \hat{h}_{N_1}^{(t)} \). Here the first \( f_{id} \) in layer \( L_{\text{min}} + 1 \) gets as input the output of the neurons (18) in layer \( L_{\text{min}} \) providing the value of \( \hat{h}_{N_1}^{(t)} \). All other networks \( f_{id} \) in the layers \( L_{\text{min}} + 2, \ldots, L \) get as input the output of the network \( f_{id} \) of the layer before, i.e., the output of the neurons \( 2 \sum_{t=1}^{t} \tilde{N}_t - 1 \) and \( 2 \sum_{t=1}^{t} \tilde{N}_t \). The output of our network \( t \) is then the sum of the neurons \( 2 \sum_{t=1}^{t} \tilde{N}_t - 1 \) and \( 2 \sum_{t=1}^{t} \tilde{N}_t \) in layer \( L \). Since we have
\[
t(x) = f_{id}^{L-L_{\text{min}}} \left( \hat{h}_{N_1}^{(t)}(x) \right) = \hat{h}_{N_1}^{(t)}(x)
\]
this shows the assertion.

\( \square \)

## 5 Proof of the main result

### 5.1 An auxiliary result from the empirical process theory

In the proof of Theorem 1 we use the following bound on the expected \( L_2 \) error of the least squares estimates.

**Lemma 1.** Assume that the distribution of \((X,Y)\) satisfies
\[
\mathbb{E}\{\exp(c_{14} \cdot Y^2)\} < \infty
\]
for some constant \( c_{14} > 0 \) and that the regression function \( m \) is bounded in absolute value.

Let \( \tilde{m}_n \) be the least squares estimate
\[
\tilde{m}_n(\cdot) = \arg\min_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{i=1}^{n} |Y_i - f(X_i)|^2
\]

based on some function space \( \mathcal{F}_n \) and set \( m_n = T_{c_{15} \log(n)} \tilde{m}_n \) for some constant \( c_{15} > 0 \).

Then \( m_n \) satisfies
\[
\mathbb{E} \int |m_n(x) - m(x)|^2 P_X(dx) \leq \frac{c_{16} \cdot (\log(n))^2 \cdot \sup_{x \in \mathbb{R}^d} \left( \log \left( N_1 \left( \frac{1}{n c_{15} \log(n)}, T_{c_{15} \log(n)} \mathcal{F}_n, x_1^2 \right) \right) + 1 \right)}{n}
\]

23
for \( n > 1 \) and some constant \( c_{16} > 0 \), which does not depend on \( n \) or the parameters in the estimate.

\textbf{Proof.} This result follows in a straightforward way from the proof of Theorem 1 in Bagirov et al. (2009). A complete proof can be found in the supplement of Bauer and Kohler (2019).

\section{5.2 A bound on the covering number}

If the function class \( F_n \) in Lemma 1 forms a class of fully connected neural networks \( F(L, r) \) with \( r \) bounded, the following result will help to bound the covering number:

\textbf{Lemma 2.} Let \( \frac{1}{n c_{17}} \leq \epsilon < \frac{c_{15} \log(n)}{4} \) and let \( F(L, r) \) defined as in (5) where \( \sigma : \mathbb{R} \to \mathbb{R} \) with \( \sigma(x) = \max\{x, 0\} \) and \( r \leq c_{18} \) for large \( n \) and certain constants \( c_{15}, c_{17}, c_{18} > 0 \). Then

\[ \log \left( N_1(\epsilon, T_{c_{15} \log(n)} F(L, r), x^n) \right) \leq c_{19} \log(n) \log(L) L^2 \]

holds for sufficiently large \( n, x_1, \ldots, x_n \in \mathbb{R}^d \) and a constant \( c_{19} > 0 \) independent of \( n \).

\textbf{Proof.} This assertion follows by a combined application of Lemma 9.2 and Theorem 9.4 in Györfi et al. (2002) together with Theorem 6 in Bartlett et al. (2019). A complete proof can be found in the supplement.

\section{5.3 Proof of Theorem 1}

Let \( a_n = \log(n)^{4/(8 \cdot (p_{\max} + 1))} \). Application of Lemma 1 leads to

\[ \mathbb{E} \int |m_n(x) - m(x)|^2 P_X(dx) \leq c_{16} \log(n)^2 \left( \sup_{x_1 \in \mathbb{R}^d} \log \left( N_1 \left( \frac{1}{n c_{17} \log(n)}, T_{c_{13} \log(n)} F(L_n, r_n), x_1^n \right) \right) + 1 \right) + 2 \inf_{f \in F(L_n, r_n)} \int |f(x) - m(x)|^2 P_X(dx). \]

Set

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

The fact that \( \frac{1}{n c_{17}} \leq \frac{1}{n c_{13} \log(n)} \leq \frac{c_{15} \log(n)}{4} \) and \( L_n \leq c_{20} \cdot n^{\frac{1}{2(2p/2p + K + 1)}} \) holds for \( c_{17}, c_{20} > 0 \), allows us to apply Lemma 2 to bound the first summand by

\[ c_{16} \log(n)^2 c_{19} \log(n) \cdot \log \left( c_{20} n^{\frac{1}{(2p/2p + K + 1)}} \right) c_{20} n^{\frac{1}{(2p/2p + K + 1)}} \]

for \( n > 1 \) and some constant \( c_{16} > 0 \).
\[
\leq \frac{c_{21}(\log(n))^{4n^{\frac{2p}{K+1}}}}{n} \leq c_{21}(\log(n))^{4}n^{-\frac{2p}{p+K}}
\]  
(22)

for a sufficiently large \(n\). Regarding the second summand we apply Theorem 3, where we choose \(M_{j,i} = \left\lceil \frac{n^{1/2(2p^{(i)}j+K^{(i)})}}{2} \right\rceil \).

W.l.o.g. we assume \(\text{supp}(X) \subseteq [-a_n, a_n]^d\). Theorem 3 allows us to bound
\[
\inf_{f \in F(L_n, r_n)} \int |f(x) - m(X)|^2 P_X(dx)
\]
by
\[
c_{22} \left(a_n^{4(p_{\max}+1)}\right)^2 \max_{j,i} M_{j,i}^{-4p^{(i)}j} = c_{22}(\log(n))^{4} \max_{j,i} n^{-\frac{2p^{(i)}j}{2p^{(i)}j+K^{(i)}j}}.
\]
This together with (22) and the fact that
\[
n^{-\frac{2p}{p+K}} = \max_{j,i} n^{-\frac{2p^{(i)}j}{2p^{(i)}j+K^{(i)}j}}
\]
implies the assertion.

References

Anthony, M. and Bartlett, P. L. (2009). Neural Network Learning: Theoretical Foundations. Cambridge University Press, New York, NY, USA, 1st edition.

Bagirov, A. M., Clausen, C., and Kohler, M. (2009). Estimation of a regression function by maxima of minima of linear functions. IEEE Trans. Information Theory, 55(2):833–845.

Barron, A. (1993). Universal approximation bounds for superpositions of a sigmoidal function. IEEE Transactions on Information Theory, 39(3):930–945.

Barron, A. R. (1991). Complexity regularization with application to artificial neural networks. Nonparametric Functional Estimation and Related Topics, pages 561–576.

Barron, A. R. (1994). Approximation and estimation bounds for artificial neural networks. Machine Learning, 14(1):115–133.

Bartlett, P. L., Harvey, N., Liaw, C., and Mehrabian, A. (2019). Nearly-tight vc-dimension bounds for piecewise linear neural networks. Journal of Machine Learning Research, 20:1–17.

Bauer, B. and Kohler, M. (2019). On deep learning as a remedy for the curse of dimensionality in nonparametric regression. Annals of Statistics, 47:2261–2285.

Devroye, L., Györfi, L., and Lugosi, G. (1996). A Probabilistic Theory of Pattern Recognition. Springer.
Devroye, L. P. and Wagner, T. J. (1980). Distribution-free consistency results in non-parametric discrimination and regression function estimation. *Annals of Statistics*, 8(2):231–239.

Eldan, R. and Shamir, O. (2016). The power of depth for feedforward neural networks. In Feldman, V., Rakhlin, A., and Shamir, O., editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 907–940. PMLR.

Friedman, J. H. and Stuetzle, W. (1981). Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–823.

Györfi, L., Kohler, M., Krzyżak, A., and Walk, H. (2002). *A Distribution-Free Theory of Nonparametric Regression*. Springer Series in Statistics. Springer.

Härdle, W., Hall, P., and Ichimura, H. (1993). Optimal smoothing in single-index models. *Annals of Statistics*, 21(1):157–178.

Härdle, W. and Stoker, T. M. (1989). Investigating smooth multiple regression by the method of average derivatives. *Journal of the American Statistical Association*, 84(408):986–995.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition.

Hertz, J., Palmer, R. G., and Krogh, A. S. (1991). *Introduction to the Theory of Neural Computation*. Perseus Publishing, 1st edition.

Horowitz, J. L. and Mammen, E. (2007). Rate-optimal estimation for a general class of nonparametric regression models with unknown link functions. *Annals of Statistics*, 35(6):2589–2619.

Kohler, M. (2014). Optimal global rates of convergence for noiseless regression estimation problems with adaptively chosen design. *Journal of Multivariate Analysis*, 132:197–208.

Kohler, M. and Krzyżak, A. (2005). Adaptive regression estimation with multilayer feedforward neural networks. *Journal of Nonparametric Statistics*, 17(8):891–913.

Kohler, M. and Krzyżak, A. (2017). Nonparametric regression based on hierarchical interaction models. *IEEE Trans. Information Theory*, 63(3):1620–1630.

Kong, E. and Xia, Y. (2007). Variable selection for the single–index model. *Biometrika*, 94(1):217–229.

McCaffrey, D. F. and Gallant, A. R. (1994). Convergence rates for single hidden layer feedforward networks. *Neural Networks*, 7(1):147–158.
Mhaskar, H. and Poggio, T. A. (2016). Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, 14.

Ripley, B. D. and Hjort, N. L. (1995). *Pattern Recognition and Neural Networks*. Cambridge University Press, New York, NY, USA, 1st edition.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Schmidt-Hieber, J. (2019). Nonparametric regression using deep neural networks with relu activation function. *To appear in Annals of Statistics*.

Stone, C. J. (1982). Optimal global rates of convergence for nonparametric regression. *Annals of Statistics*, 10(4):1040–1053.

Stone, C. J. (1985). Additive regression and other nonparametric models. *Annals of Statistics*, 13(2):689–705.

Stone, C. J. (1994). The use of polynomial splines and their tensor products in multivariate function estimation. *Annals of Statistics*, 22(1):118–171.

Yarotsky, D. (2018). Optimal approximation of continuous functions by very deep relu networks. *COLT*, 75:639–649.

Yu, Y. and Ruppert, D. (2002). Penalized spline estimation for partially linear single-index models. *Journal of the American Statistical Association*, 97(460):1042–1054.