Statistically guided deep learning *

Michael Kohler [1] and Adam Krzyżak[2,†]
[1] *Fachbereich Mathematik, Technische Universität Darmstadt, Schlossgartenstr. 7, 64289 Darmstadt, Germany, email: kohler@mathematik.tu-darmstadt.de*
[2] *Department of Computer Science and Software Engineering, Concordia University, 1455 De Maisonneuve Blvd. West, Montreal, Quebec, Canada H3G 1M8, email: krzyzak@cs.concordia.ca*

April 11, 2025

Abstract
We present a theoretically well-founded deep learning algorithm for nonparametric regression. It uses over-parametrized deep neural networks with logistic activation function, which are fitted to the given data via gradient descent. We propose a special topology of these networks, a special random initialization of the weights, and a data-dependent choice of the learning rate and the number of gradient descent steps. We prove a theoretical bound on the expected L_2 error of this estimate, and illustrate its finite sample size performance by applying it to simulated data.

Our results show that a theoretical analysis of deep learning which takes into account simultaneously optimization, generalization and approximation can result in a new deep learning estimate which has an improved finite sample performance.

AMS classification: Primary 62G08; secondary 62G20.

Key words and phrases: Deep neural networks, gradient descent, nonparametric regression, rate of convergence, over-parametrization.

1 Introduction

1.1 Scope of this paper

Due to its tremendous success in applications, e.g., in image classification (cf., e.g., Krizhevsky, Sutskever and Hinton (2012)), in language recognition (cf., e.g., Kim (2014)) in machine translation (cf., e.g., Wu et al. (2016)) or in mastering of games (cf., e.g., Silver et al. (2017)), deep learning is currently changing the world. This big success of deep learning in the past relies on two things: the massive increase of computing power and availability of the huge data sets. However, it seems that both cannot be much more increased: Firstly, there is already a shortage of computer chips for deep learning, and also the increasing electricity demand of the computers used for computing the deep

*Running title: *Statistically deep learning*
†Corresponding author. Tel: +1-514-848-2424 ext. 3007, Fax:+1-514-848-2830

learning estimates seems problematic. And secondly, e.g. for large language models, all available text data has been already used for the training, so it is not clear how the size of the used data sets can be further increased.

But there remains one different approach to improve the deep learning estimates: one can try to improve the used estimation methods. In the past new methods have been mainly constructed by trial and error, and not based on a rigorous theoretical analysis. In this paper we investigate whether a theoretical approach succeeds in improving deep learning estimates.

1.2 Nonparametric regression

We study deep learning estimates in the context of nonparametric regression. Here (X, Y), (X_1, Y_1), ..., (X_n, Y_n) are independent and identically $\mathbb{R}^d \times \mathbb{R}$–valued random vectors with $\mathbf{E}Y^2 < \infty$, and given the data set

$$\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\} \tag{1}$$

the task is to estimate the so–called regression function

$$m : \mathbb{R}^d \to \mathbb{R}, \quad m(x) = \mathbf{E}\{Y|X = x\}.$$

More precisely, the goal is to construct an estimate

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

such that the so-called L_2 error

$$\int |m_n(x) - m(x)|^2 \mathbf{P}_X(dx)$$

is close to zero.

A detailed introduction to nonparametric regression, its estimates and known theoretical results can be found, e.g., in Györfi et al. (2002).

1.3 Least squares estimates estimates

Since

$$\mathbf{E}\{|m_n(X) - Y|^2 | \mathcal{D}_n\} = \mathbf{E}\{|m(X) - Y|^2\} + \int |m_n(x) - m(x)|^2 \mathbf{P}_X(dx)$$

(cf., e.g., Chapter 1 in Györfi et al. (2002)), the aim of minimizing the L_2 error means that one wants to find an estimate such that its so–called L_2 risk (or mean squared prediction error)

$$\mathbf{E}\{|m_n(X) - Y|^2 | \mathcal{D}_n\} \tag{2}$$

is close to the optimal value $\mathbf{E}\{|m(X) - Y|^2\}$.

This way of considering the estimation task immediately suggest a way of solving it: One can try to use the given data (1) to estimate the L_2 risk (2) by the so–called empirical L_2 risk

$$\frac{1}{n}\sum_{i=1}^{n}|m_n(X_i)-Y_i|^2 \tag{3}$$

and can try to minimize (3) over some space of functions. This leads to so–called least squares estimates

$$m_n(\cdot)=\arg\min_{f\in\mathcal{F}_n}\frac{1}{n}\sum_{i=1}^{n}|f(X_i)-Y_i|^2 \tag{4}$$

which depend on spaces $\mathcal{F}_n$ of functions $f:\mathbb{R}^d\to\mathbb{R}$. Here the right choice of the function space is crucial, since it must be on the one hand so rich that functions in it are able to approximate the (unknown) regression function well, and on the other hand it should be such that the empirical L_2 risk of the function which minimizes the empirical L_2 risk is close to its expectation. Usually the latter is shown by showing that the maximal deviation between the L_2 risk and the empirical L_2 risk on the function space is small, which holds if the function space is not too complex.

1.4 Neural networks

For neural network estimates one considers in this context spaces of neural networks. In their simplest form of fully connected feedfoward neural networks they are defined as follows: One chooses an activation function $\sigma:\mathbb{R}\to\mathbb{R}$, e.g.,

$$\sigma(x)=\max\{x,0\} \tag{5}$$

(so-called ReLU-activation function) or

$$\sigma(x)=\frac{1}{1+e^{-x}} \tag{6}$$

(so-called logistic squasher), and selects the number $L\in\mathbb{N}$ of hidden layers of the network and the numbers $k_s\in\mathbb{N}$ of neurons in the s-th hidden layer ($s\in\{1,\dots,L\}$). Then the feedforward neural network $f_{\mathbf{w}}$ with L hidden layers, k_s neurons in layer $s\in\{1,\dots,L\}$ and with weight vector $\mathbf{w}=(w_{i,j}^{(l)})_{l,i,j}$ is the function $f_{\mathbf{w}}:\mathbb{R}^d\to\mathbb{R}$ defined by

$$f_{\mathbf{w}}(x)=\sum_{j\in\{1,\dots,k_L\}}w_{1,j}^{(L)}\cdot f_j^{(L)}(x), \tag{7}$$

where

$$f_i^{(s)}(x)=\sigma\left(\sum_{j\in\{1,\dots,k_{s-1}\}}w_{i,j}^{(s-1)}\cdot f_j^{(s-1)}(x)+w_{i,0}^{(s-1)}\right)\quad\text{for } s\in\{2,\dots,L\}\text{ and } i>0 \tag{8}$$

and

$$f_i^{(1)}(x) = \sigma\left(\sum_{j\in\{1,\dots,d\}} w_{i,j}^{(0)} \cdot x^{(j)} + w_{i,0}^{(0)}\right) \quad \text{for } i > 0. \tag{9}$$

Here $w_{i,j}^{(s-1)}$ is the weight between neuron j in layer $s-1$ and neuron i in layer s. And $w_{i,0}^{(s-1)}$ is the bias in the computation of the output of neuron i in layer s.

The idea is then to fix the activation function, the numer of layers $L \in \mathbb{N}$, the number $k_s \in \mathbb{N}$ of neurons in layer $s \in \{1,\dots,L\}$, and to choose the weight vector $\mathbf{w}$ by minimizing the empirical L_2 risk

$$F_n(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} |f_{\mathbf{w}}(X_i) - Y_i|^2 \tag{10}$$

of $f_{\mathbf{w}}$ with respect to $\mathbf{w}$.

Usually, the activation function σ is highly nonlinear, and therefore $f_{\mathbf{w}}(X_i)$ and also $F_n(\mathbf{w})$ depend nonlinearly on $\mathbf{w}$. Due to this it is not clear how one can minimize (10) with respect to the weight vector $\mathbf{w}$.

1.5 Computation of neural network regression estimates

Minimizing of the empirical L_2 risk with respect to a class of neural networks is done in practice by using gradient descent (or one of its variants like stochastic gradient descent): One chooses a random starting vector $\mathbf{w}^{(0)}$ for the weights and computes $t_n \in \mathbb{N}$ gradient descent steps

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \lambda_n \cdot \nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t-1)}) \quad (t = 1,\dots,t_n) \tag{11}$$

with stepsize $\lambda_n > 0$. The estimate is then defined by

$$m_n(x) = f_{\mathbf{w}^{(t_n)}}(x).$$

1.6 Difficulty in the application of deep neural networks

The above definition of the neural network regression estimates requires decisions about the class of neurals networks which we fit to the data, the choice of the starting vector, the choice of the number of gradient descent steps and the choice of the stepsize.

If we consider for simplicity just fully connected neural networks with L layers and r neurons per layer (i.e., we set $k_s = r$ for $s = 1,\dots,L$), the logistic activation function, and the famous ADAM rule for the choice of the stepsize, then the remaining question is how to choose the starting vector, and how to choose the number of gradient descent steps. For the choice of the starting vector popular algorithms in the literature are the GlorotNormal-, the GlorotUniform-, the HeNormal- or the HeUniform-rule (cf., e.g., Chapter 8 in Goodfellow, Bengio and Courville (2016)), where the initial weights are chosen independently from the normal distributions or uniform distributions.

In the upper right panel in Figure 1 we apply this for a neural network with $L = 4$, $r = 20$ and the GlorotNormal-rule for initialization to an univariate regression problem (which

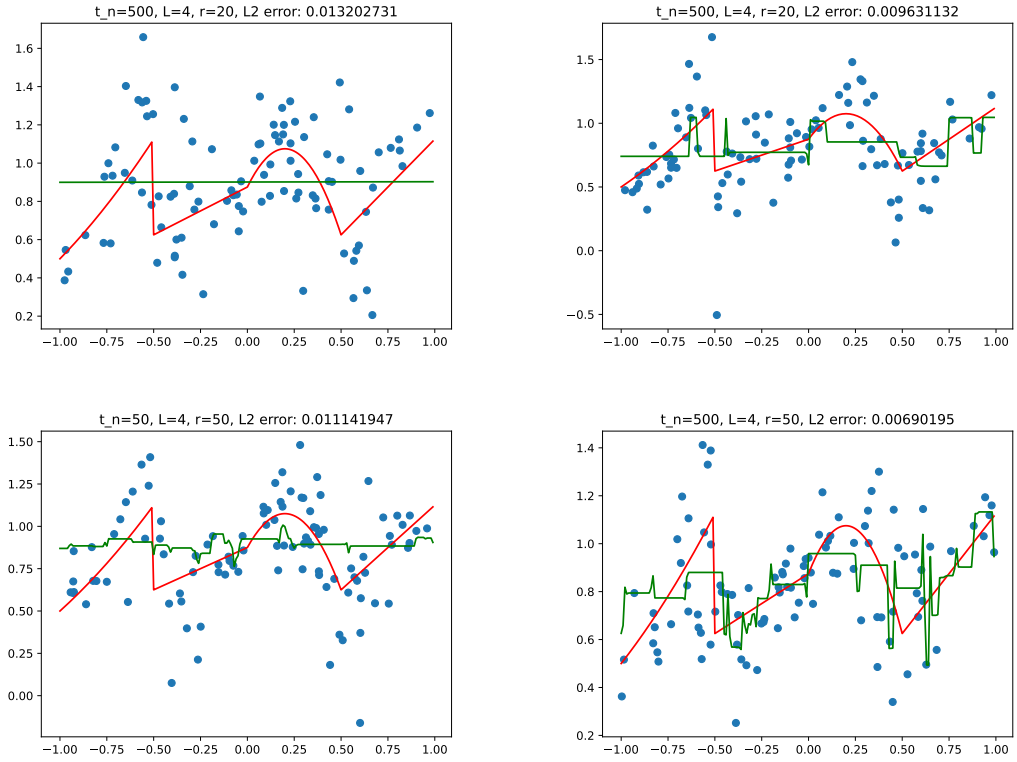

Figure 1: Neural network estimate with various initialization schemes, various topologies and various choices of the stepsize applied to the univariate regression problem with sample size $n = 100$.

is described in detail in Section 4), which leads to a constant estimate (in green) which does not approximate the regression function (in red) well. The same effect happens with the GlorotUniform-, the HeNormal- or the HeUniform-rule. The picture drastically changes if we use the uniform distribution on an interval $[-A, A]$ for the weights on the input level, the uniform distribution on an interval $[-B, B]$ for all inner weights, set all weights on the output level initially to zero, and choose a large value for A and a moderate value for B. For $A = 1000$, $B = 20$ and three different values for (L, r, t_n) the estimates are then shown in the upper right, the lower left, and the lower right panel in Figure 1, respectively.

This shows that the performance of the neural network estimate crucially depends on the chosen parameters. As mentioned on page 293 in Goodfellow, Bengio and Courville (2016) "designing improved initialization strategies is a difficult task because neural network optimization is not yet well understood." Furthermore, it is mentioned there that "our understanding of how the initial point affects generalization is especially primitive, offering little or no guidance for how to select the initial point". It should be added that the same problem also occurs in connection with the number of gradient descent steps

for the ADAM rule, or more generally with the stepsize choices during gradient descent and the number of gradient descent steps.

In this article we consider simultaneously optimization, generalization and approximation of neural networks and use this to propose a theoretically motivated way of choosing the parameters of the neural network estimates.

1.7 A theoretical approach to deep learning

In practice usually over-parametrized deep neural networks are used, where the number of weights is much larger than the sample size n, so one fits a function to the data which has much more free parameters (i.e., weights) than there are data points.

There are three main theoretical questions in this context: Firstly, why does the resulting estimate optimize well, i.e., why is gradient descent able to achieve small values of the empirical L_2 risk? Secondly, why does the resulting estimate generalize well, i.e., why is its squared error on new independent data (not contained in the training data) small? And why does it approximate well, i.e., why does the sequence of weight vectors considered during gradient descent contains a weight vector for which the corresponding neural network approximates the regression function well? Of course, if we are able to answer these questions then it should also be possible to say which activation function, which topology (i.e., number of layers and number of neurons per layer), which initialization of the weights, which stepsize and which number of gradient descent steps lead to estimates with a small L_2 error. So a theoretical understanding of the above three questions might be used to construct hints for the choice of the parameters of the estimate in applications.

Kohler (2024) has developed a theory answering these questions, which applies to over-parametrized deep neural networks with smooth activation function. It uses the observation that for a proper choice of λ_n and t_n the weights computed during gradient descent stay in a local neighborhood of the starting value. More precisely, if $\lambda_n = \frac{1}{L_n}$ and the gradient of $F_n(\mathbf{w})$ is Lipschitz continuous with Lipschitz constant L_n around the starting weight vector, i.e., if

$$\|\nabla_{\mathbf{w}} F_n(\mathbf{w}_1) - \nabla_{\mathbf{w}}(F_n(\mathbf{w}_2)\| \leq L_n \cdot \|\mathbf{w}_1 - \mathbf{w}_2\|$$

holds for $\mathbf{w}_1$ and $\mathbf{w}_2$ "close" to the starting weight vector $\mathbf{w}^{(0)}$, and additionally the gradient is suitably bounded in this neighborhood, then during gradient descent

$$\|\mathbf{w}^{(t)} - \mathbf{w}^{(0)}\| \leq \sqrt{c_1 \cdot \lambda_n \cdot t}$$

holds for all $t \in \{1, \ldots, t_n\}$ (cf., Lemma A.1 in Braun et al. (2023)). Since

$$\|\mathbf{w}^{(t)} - \mathbf{w}^{(0)}\|_\infty \leq \|\mathbf{w}^{(t)} - \mathbf{w}^{(0)}\|$$

this implies that if we choose λ_n and t_n such that $\lambda_n \cdot t_n$ is bounded by some constant, then any bounds which we impose on the absolute value of the weights during the random initialization enable us to derive bounds on the absolute value of the weights during gradient descent.

Kohler (2024) uses such bounds to ensure the estimates generalize well. This is possible, since the smoothness of the activation function together with the bounds on the weights enables one to derive bounds on the derivative of the networks. And using these bounds one can approximate the corresponding set of deep networks by piecewise polynomials and bound the complexity of the set of deep networks by a suitable covering number of the set of piecewise polynomials. In this context Kohler (2024) uses a special topology of the network, where a huge linear combination of many small networks of fixed depth L and width r are computed. It turns out that neither the number K_n of these small networks nor the bounds on the absolute value of the coefficients in the linear combination have a crucial influence on the covering number above as long as they grow not faster than some polynomial in the sample size. In this way it is possible to define over-parametrized deep neural networks which generalize well (since during gradient descent they are always contained in some function space with a finite complexity). Furthermore, Kohler (2024) uses different bounds A_n and B_n for the absolute value of the weights in the input layer and the absolute value of the weights between the hidden layers. Here B_n is chosen as a large constant, and then A_n is the main parameter controlling the complexity of the over-parametrized deep networks chosen by $A_n = c_2 \cdot (\log n) \cdot n^\tau$ for some $\tau \in (0, 1)$.

In order to analyze the approximation error, Kohler (2024) derives an approximation result for the approximation of a smooth function by networks where the weights are bounded as above. Here the number of networks and the size of A_n are related and they control the approximation error of the deep network.

Furthermore, Kohler (2024) uses a relation between the gradient descent applied to the empirical L_2 risk of the deep network and the gradient descent applied to the empirical L_2 risk of the linear Taylor approximation of the deep network in order to analyze the gradient descent. This makes it possible to use techniques which have been developed for analysis of gradient descent applied to smooth convex functions.

In Kohler (2024) the regression function is assumed to be (p, C)–smooth in the following sense.

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 $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$ with $\sum_{j=1}^d \alpha_j = q$ the partial derivative $\partial^q m / (\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d})$ exists and satisfies*

$$\left| \frac{\partial^q m}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}(x) - \frac{\partial^q m}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}(z) \right| \leq C \cdot \|x - z\|^s$$

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

Kohler (2024) considered a neural network topology consisting of $K_n \in \mathbb{N}$ in parallel computed neural network with logistic squasher activation function, and with depth L and width r, where

$$\frac{K_n}{n^\kappa} \to 0 \quad (n \to \infty) \quad \text{and} \quad \frac{K_n}{n^{4 \cdot r \cdot (r+1) \cdot (L-1) + r \cdot (4d+6) + 6}} \to \infty \quad (n \to \infty)$$

for some $\kappa > 0$ and

$$L = \lceil \log_2(q+d) \rceil + 1 \quad \text{and} \quad r = 2 \cdot \lceil (2p+d)^2 \rceil.$$

The weights are initialized such that all weights of the input level are uniformly distributed on $[-c_2 \cdot (\log n) \cdot n^{1/(2p+d)}, c_2 \cdot (\log n) \cdot n^{1/(2p+d)}]$, all inner weights are uniformly distributed on $[-c_3, c_3]$ and all the output weights are set to zero. Then

$$t_n = \left\lceil c_4 \cdot \frac{K_n^3}{\beta_n} \right\rceil$$

gradient descent steps with stepsize

$$\lambda_n = \frac{c_5}{n \cdot K_n^3}$$

are performed. It is shown in Theorem 1 in Kohler (2024) that a truncated version of this estimate satisfies for any $\epsilon > 0$

$$\mathbf{E} \int |m_n(x) - m(x)|^2 \mathbf{P}_X(dx) \leq c_6 \cdot n^{-\frac{2p}{2p+d}+\epsilon},$$

provided $supp(X)$ is compact, $\mathbf{E}\left\{e^{c_7 \cdot Y^2}\right\} < \infty$ and the regression functions is (p, C)–smooth.

The main problem in using this result for defining a neural network estimate applied to data is that the parameters K_n and t_n are so large that the estimate cannot be computed in practice.

1.8 Main results

In this article we define neural network estimates with logistic squasher activation function where a linear combination of K_n fully connected neural networks of depth L and width r is computed in parallel. We use uniform distributions on the intervals $[-A_n, A_n]$ and $[-B_n, B_n]$ for initialization of the input weights and the inner weights, resp. All outer weights are initially set to zero. We perform t_n gradient descent steps with stepsize λ_n, where we choose

$$\lambda_n = \frac{1}{\hat{t}_n} \quad \text{and} \quad t_n = \min\left\{\hat{t}_n, \lceil (\log n)^{c_8} \cdot K_n^3 \rceil\right\}$$

such that

$$\hat{t}_n \in \left\{2^i \cdot t_{min} \quad : \quad i \in \mathbb{N}_0\right\}$$

satisfies (with high probability) the following three conditions:

$$\frac{1}{t_n} \cdot \sum_{t=0}^{t_n-1} \lambda_n \cdot \left\| \nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)}) \right\|^2 \leq \frac{c_9}{n},$$

$$F_n(\mathbf{w}^{(t_n)}) \leq \frac{1}{t_n} \cdot \sum_{t=0}^{t_n-1} F_n(\mathbf{w}^{(t)}) + \frac{c_9}{n},$$

and

$$\max_{t=1,\ldots,t_n} \|\mathbf{w}^{(0)} - \mathbf{w}^{(t)}\|^2 \leq \frac{c_9 \cdot \log n}{n}.$$

We propose an algorithm which chooses λ_n and t_n such that these three conditions are satisfied (with high probability). We show that the truncated version of the corresponding estimate satisfies

$$\begin{aligned}
&\mathbf{E} \int |m_n(x) - m(x)|^2 \mathbf{P}_X(dx) \\
&\leq c_{10} \cdot \left(\mathbf{E} \left\{ \inf_{\mathbf{w}: \|\mathbf{w} - \mathbf{w}^{(0)}\| \leq \frac{1}{n}} \int |f_{\mathbf{w}}(x) - m(x)|^2 \mathbf{P}_X(dx) \right\} + \frac{A_n^d \cdot B_n^{(L-1)\cdot d}}{n^{1-\epsilon}} \right)
\end{aligned}$$

and, in case of (p, C) smooth regression function and A_n and B_n chosen suitably,

$$\mathbf{E} \int |m_n(x) - m(x)|^2 \mathbf{P}_X(dx) \leq c_{11} \cdot n^{-\frac{2p}{2p+d}+\epsilon},$$

where $\epsilon > 0$ is an arbitrary small number and where $c_{10}, c_{11} > 0$ are constants depending on ϵ. Furthermore, we implement this estimate and study its finite sample size performance in an univariate regression estimation problem, showing that it achieves a good performance on simulated data. In particular, we observe that for our simulated data the above mentioned algorithm for the choice of λ_n and t_n leads to an estimate which can be computed in a reasonable time.

Our main contributions can be summarized as follows: Motivated by a theoretical analysis of the expected L_2 error of a neural network regression estimated learned by gradient descent we propose a special topology of the neural networks, a special initialization (where we use uniform distributions whose parameters can be considered as smoothing parameters), and a special way to choose the stepsize and the number of gradient descent steps. We derive a theoretical bound on the expected L_2 error of this estimate, and propose an algorithm where all parameters are chosen data-dependent which leads to an estimate which outperforms the traditional regression estimates (including neural networks estimates) on simulated data in an univariate case. This shows that theoretical analysis of deep neural network estimates can lead to new estimates which have an improved performance on simulated data.

1.9 Discussion of related results

The huge success of deep learning in applications has motivated many researchers to investigate theoretically why these methods are so successful. This has been studied, e.g., in approximation theory, where quite a few results concerning the approximation of smooth functions by deep neural networks have been derived, see Yarotsky (2018), Yarotsky and Zhevnerchute (2019), Lu et al. (2020), Langer (2021) and the literature

cited therein. Here it is investigated what kind of topology and how many nonzero weights are necessary to approximate a smooth function up to some given error. In applications, the functions which one wants to approximate has to be estimated from observed data, which usually contain some random error. One interesting question in this context is how well a neural network learned from such noisy data generalizes on a new independent test data. Classically this is done within the framework of the VC theory, and here e.g. the result of Bartlett et al. (2019) can be used to bound the VC dimension of classes of neural networks. For over-parametrized deep neural networks (where the number of free parameters adjusted to the observed data set is much larger than the sample size) the analysis of the generalization error can be done by using bounds on the Rademacher complexity (cf., e.g., Liang, Rakhlin and Sridharan (2015), Golowich, Rakhlin and Shamir (2019), Lin and Zhang (2019), Wang and Ma (2022) and the literature cited therein). By combining these results it was possible to analyze the error of least squares regression estimates. Here results have been shown which indicate why deep learning performs well in high-dimensional applications: they show that least squares regression estimates based on deep neural networks can achieve a dimension reduction in case that the function to be estimated satisfies a hierarchical composition model, i.e., in case that it is a composition of smooth functions which do either depend only on a few components or are rather smooth. One of the first results in this respect was shown in Kohler and Krzyżak (2017), and later extended by Bauer and Kohler (2019), Schmidt-Hieber (2020) and Kohler and Langer (2021). The main trick in these papers is the use of the network structure of deep neural networks, which implies that the composition of neural networks is itself a deep neural network. Consequently, any approximation result for some functions by deep neural networks can be extended to approximation of a composition of such functions by a deep neural network representing a composition of the approximating networks. Since in this setting neither the number of weights nor the depth of the network, which determine the VC dimension and hence the complexity of the neural network in case that it is not over-parametrized (cf., Bartlett et al. (2019)), changes much, these neural networks share the approximation properties and the complexity of neural networks for low dimensional predictors and hence can achieve dimension reduction. Bhattacharya, Fan and Mukherjee (2025) showed that a suitably defined least squares neural network estimate can also achieve (up to logarithmic factors) optimal rate of convergence results in interaction models with diverging dimensions.

In practice, least squares estimates cannot be applied because the corresponding optimization problem cannot be solved efficiently. Instead, gradient descent is used to compute the estimate, and then it is natural to investigate theoretically whether estimates learned by gradient descent have nice properties. It was shown in a series of papers, cf., e.g., Zou et al. (2018), Du et al. (2019), Allen-Zhu, Li and Song (2019) and Kawaguchi and Huang (2019), that the application of gradient descent to over-parameterized deep neural networks can lead to neural networks which (globally) minimize the empirical risk considered. Unfortunately, as was shown in Kohler and Krzyżak (2021), the corresponding estimates do not behave well on a new independent data.

In applications it is essential to control the approximation, generalization and optimization errors simultaneously (cf., Kutyniok (2020)). Unfortunately, none of the results

mentioned above controls all these three aspects simultaneously.

One way to study these three aspects simultaneously is to use some equivalent model of deep learning. The most prominent approach here is the neural tangent kernel setting, which was proposed by Jacot, Gabriel and Hongler (2020). In this approach a kernel estimate is studied and its error is used to bound the error of the neural network estimate (see also Hanin and Nica (2019) and the literature cited therein). It was observed by Nitanda and Suzuki (2021) that in most studies in the neural tangent kernel setting the equivalence to deep neural networks holds only pointwise and not for the global L_2 error, which is crucial for predictions problems in practice. So from results derived in the neural tangent kernel setting it is often not clear how the L_2 error of the deep neural network estimate behaves. An exception is the article Nitanda and Suzuki (2021), where the global error of an over-parametrized shallow neural network learned by gradient descent was studied based on the neural tangent kernel approach. However, due to the use of the neural tangent kernel, the smoothness assumption on the function to be estimated has to be defined with the aid of a norm involving the kernel, which does not lead to classical smoothness conditions usually considered, which makes it hard to interpret the obtained results. In addition, it is required that the number of neurons be sufficiently large, but it was not specified what this exactly means, i.e., it is not clear whether the number of neurons must grow e.g. exponentially in the sample size or not.

Another approach where the estimate is studied in some asymptotically equivalent model is the mean field approach, cf., e.g., Mei, Montanari, and Nguyen (2018), Chizat and Bach (2018), Nguyen and Pham (2020), Ba et al. (2020), Arous, Gheissari and Jagannath (2021), Bietti et al. (2022), and the literature cited therein. Here it is again unclear how close the behaviour of the deep networks in the equivalent model is to the behaviour of the deep networks in the applications, because the equivalent model is based on some approximation of the deep neural networks using, e.g., some asymptotic expansions.

In a online stochastic gradient setting, where in each gradient descent step a new independent data point is given, Abbe, Adsera, and Misiakiewicz (2023) studies the rate of convergence of a shallow neural network estimate learned by the layerwise gradient descent for special regression functions. Here upper and lower bounds on the rate of convergence (or more precisely: the number of gradient descent steps required to achieve a given error bound) are derived.

The results presented in this paper are based on the statistical theory for deep neural networks developed by the authors together with various co-authors, see, e.g. Braun et al. (2023), Drews and Kohler (2023, 2024), Kohler and Krzyżak (2022, 2023) and Kohler (2024). Here Braun et al. (2023) investigates the rate for convergence of a shallow neural network estimate learned by gradient descent. All other papers consider deep neural networks with the same kind of topology used in the current paper. Kohler and Krzyżak (2023) uses over-parametrized deep ReLU neural network learned by gradient descent. Due to the use of Rademacher complexity to control the generalization error the rate of convergence derived in case of a (p, C)–smooth regression function is of the order $n^{-p/(2p+d)+\epsilon}$ instead of $n^{-2p/(2p+d)+\epsilon}$ as in the current paper. Drews and Kohler (2024) derives result concerning the consistency of the estimates, and in Kohler and

Krzyżak (2022) and in Drews and Kohler (2023) the same rate as in the current paper is shown but only for the special case $p = 1/2$. Here Kohler and Krzyżak (2022) use an additional regularization of the estimate, and Drews and Kohler (2023) shows that this regularization is not necessary. For general p the above rate of convergence is derived in Kohler (2024) (again without additional regularization). Our paper is closely based on the approach there and shows that the rate of convergence there can be also achieved with an estimate which uses a data-dependent choice of the number of gradient descent steps which is in applications much smaller than the number of gradient descent steps required in the theoretical result in Kohler (2024).

1.10 Notation

The sets of natural numbers and real numbers are denoted by $\mathbb{N}$ and $\mathbb{R}$, respectively. For $z \in \mathbb{R}$, we denote the smallest integer greater than or equal to z by $\lceil z \rceil$. The Euclidean norm of $x \in \mathbb{R}^d$ is denoted by $\|x\|$. For a closed and convex set $A \subseteq \mathbb{R}^d$ we denote by $Proj_A x$ that element $Proj_A x \in A$ with

$$\|x - Proj_A x\| = \min_{z \in A} \|x - z\|.$$

1.11 Outline

The newly proposed deep learning regression estimate is introduced in Section 2. Section 3 presents theoretical results concerning its rate of convergence. Its finite sample size performance is illustrated in Section 4. Section 5 contains the proofs.

2 Definition of the estimate

In the sequel we will use the logistic squasher (6) as activation function.

2.1 Topology of the network

We let $K_n, L, r \in \mathbb{N}$ be parameters of our estimate and using these parameters we set

$$f_{\mathbf{w}}(x) = \sum_{j=1}^{K_n} w_{j,1,1}^{(L)} \cdot f_{j,1}^{(L)}(x) \tag{12}$$

for some $w_{1,1,1}^{(L)}, \ldots, w_{K_n,1,1}^{(L)} \in \mathbb{R}$, where $f_{j,1}^{(L)} = f_{\mathbf{w},j,1}^{(L)}$ are recursively defined by

$$f_{k,i}^{(l)}(x) = f_{\mathbf{w},k,i}^{(l)}(x) = \sigma\left(\sum_{j=1}^{r} w_{k,i,j}^{(l-1)} \cdot f_{k,j}^{(l-1)}(x) + w_{k,i,0}^{(l-1)}\right) \tag{13}$$

for some $w_{k,i,0}^{(l-1)}, \ldots, w_{k,i,r}^{(l-1)} \in \mathbb{R}$ $(l = 2, \ldots, L)$ and

$$f_{k,i}^{(1)}(x) = f_{\mathbf{w},k,i}^{(1)}(x) = \sigma\left(\sum_{j=1}^{d} w_{k,i,j}^{(0)} \cdot x^{(j)} + w_{k,i,0}^{(0)}\right) \tag{14}$$

for some $w_{k,i,0}^{(0)}, \ldots, w_{k,i,d}^{(0)} \in \mathbb{R}$.

This means that we consider neural networks which consist of K_n fully connected neural networks of depth L and width r computed in parallel and compute a linear combination of the outputs of these K_n neural networks. The weights in the k-th such network are denoted by $(w_{k,i,j}^{(l)})_{i,j,l}$, where $w_{k,i,j}^{(l)}$ is the weight between neuron j in layer l and neuron i in layer $l+1$.

2.2 Initialization of the weights

We initialize the weights $\mathbf{w}^{(0)} = ((\mathbf{w}^{(0)})_{k,i,j}^{(l)})_{k,i,j,l}$ as follows: We set

$$(\mathbf{w}^{(0)})_{k,1,1}^{(L)} = 0 \quad (k = 1, \ldots, K_n),$$

we choose $(\mathbf{w}^{(0)})_{k,i,j}^{(l)}$ uniformly distributed on $[-B, B]$ if $l \in \{1, \ldots, L-1\}$, and we choose $(\mathbf{w}^{(0)})_{k,i,j}^{(0)}$ uniformly distributed on $[-A, A]$, where $A, B \geq 0$ are parameters of the estimate. Here the random values are defined such that all components of $\mathbf{w}^{(0)}$ are independent.

2.3 Gradient descent

Our aim is to choose the weight vector $\mathbf{w}$ by minimizing the empirical L_2 risk

$$F_n(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} |f_{\mathbf{w}}(X_i) - Y_i|^2 \tag{15}$$

of $f_{\mathbf{w}}$ with respect to $\mathbf{w}$.

We do this by using gradient descent: Given the random starting vector $\mathbf{w}^{(0)}$ for the weights from Subsection 2.2 we compute $t_n \in \mathbb{N}$ gradient descent steps

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \lambda_n \cdot \nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t-1)}) \quad (t = 1, \ldots, t_n) \tag{16}$$

with stepsize $\lambda_n > 0$.

2.4 Choice of the stepsize and the number of gradient descent steps

We choose

$$\lambda_n = \frac{1}{\hat{t}_n} \quad \text{and} \quad t_n = \min\left\{\hat{t}_n, \lceil (\log n)^{c_8} \cdot K_n^3 \rceil\right\}$$

such that

$$\hat{t}_n \in \left\{2^i \cdot t_{min} \quad : \quad i \in \mathbb{N}_0\right\}$$

satisfies either the following three conditions

$$\frac{1}{t_n} \cdot \sum_{t=0}^{t_n - 1} \lambda_n \cdot \left\| \nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)}) \right\|^2 \leq \frac{c_9}{n}, \tag{17}$$

$$F_n(\mathbf{w}^{(t_n)}) \leq \frac{1}{t_n} \cdot \sum_{t=0}^{t_n-1} F_n(\mathbf{w}^{(t)}) + \frac{c_9}{n} \tag{18}$$

and

$$\max_{t=1,\ldots,t_n} \|\mathbf{w}^{(0)} - \mathbf{w}^{(t)}\|^2 \leq \frac{c_9 \cdot \log n}{n} \tag{19}$$

simultaneously, or such that

$$n \cdot (\log n)^{c_8} \cdot K_n^3 \leq \hat{t}_n \leq 2 \cdot n \cdot (\log n)^{c_8} \cdot K_n^3 \tag{20}$$

holds. We do this by using Algorithm 1 below.

Data: $(x_1, y_1), \ldots, (x_n, y_n)$
K, L, r, A, B
$t_{min} = 50$, $t_{max,1} = (\log n)^{c_8} \cdot K^3$, $t_{max,2} = n \cdot t_{max,1}$, $c_9 = 10$
begin
  i=0
  repeat
    $\lambda = \frac{1}{2^i \cdot t_{min}}$
    $t = 0$
    $\mathbf{w}^{(0)} = InitializeWeights(K, L, r, A, B)$
    repeat
      $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \lambda \cdot \nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)})$
      $t = t + 1$
    until $t \geq min(2^i \cdot t_{min}, t_{max,1})$ *or* $\frac{1}{2^i \cdot t_{min}} \cdot \sum_{t=0}^{t-1} \lambda \cdot \left\|\nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)})\right\|^2 > \frac{c_9}{n}$ *or* $\|\mathbf{w}^{(0)} - \mathbf{w}^{(t)}\| > \frac{c_9}{n}$;
    $i = i + 1$
  until $\left(\frac{1}{t} \cdot \sum_{s=0}^{t-1} \lambda \cdot \left\|\nabla_{\mathbf{w}} F_n(\mathbf{w}^{(s)})\right\|^2 \leq \frac{c_9}{n}$ *and* $F_n(\mathbf{w}^{(t)}) \leq \frac{1}{t} \cdot \sum_{s=0}^{t-1} F_n(\mathbf{w}^{(s)}) + \frac{c_9}{n}$ *and* $\max_{s=1,\ldots,t} \|\mathbf{w}^{(0)} - \mathbf{w}^{(s)}\|^2 \leq \frac{c_9 \cdot \log n}{n} \right)$ *or* $t \geq t_{max,2}$;
end
Result: $f_{\mathbf{w}^{(t)}}$

Algorithm 1: Pseudo code for the choice of the stepsize and the number of gradient descent steps.

In Algorithm 1 the second and the third condition in the inner repeat-until loop imply that the first or the third condition in the outer repeat-until loop cannot be satisfied if we continue the inner loop and therefore the inner loop is terminated if one of these conditions holds.

2.5 Definition of the estimate

For the theoretical analysis we consider a truncated version of the neural network with weight vector $\mathbf{w}^{(t_n)}$, i.e., we define the estimate by

$$m_n(x) = T_{\beta_n}(f_{\mathbf{w}^{(t_n)}}(x)) \tag{21}$$

where $\beta_n = c_{12} \cdot \log n$ and $T_\beta z = \max\{\min\{z, \beta\}, -\beta\}$ for $z \in \mathbb{R}$ and $\beta > 0$.

3 Rate of convergence

In this section we present our theoretical results concerning the estimate introduced in Section 2.

3.1 A general result

Our first result is a general bound on the expected L_2 error of our estimate.

Theorem 1 *Let $n \in \mathbb{N}$, let (X, Y), (X_1, Y_1), ..., (X_n, Y_n) be independent and identically distributed $\mathbb{R}^d \times \mathbb{R}$–valued random variables such that supp(X) is bounded, the regression function is bounded in absolute value, and*

$$\mathbf{E}\left\{e^{c_7 \cdot Y^2}\right\} < \infty \tag{22}$$

holds. Let $K_n \in \mathbb{N}$ be such that

$$\frac{K_n}{n^\kappa} \to 0 \quad (n \to \infty) \tag{23}$$

for some $\kappa > 0$, set $A = A_n$ and $B = B_n$ for some

$$1 \le A_n \le n \quad \text{and} \quad 1 \le B_n \le c_{13} \cdot \log n, \tag{24}$$

set $\beta_n = c_{12} \cdot \log n$ and define the estimate as in Section 2. Assume $c_7 \cdot c_{12} \ge 3$ and $c_8 > 2L$. Then we have for any $\epsilon > 0$

$$\begin{aligned}
&\mathbf{E} \int |m_n(x) - m(x)|^2 \mathbf{P}_X(dx) \\
&\le c_{14} \cdot \left(\mathbf{E}\left\{ \inf_{\mathbf{w}: \|\mathbf{w} - \mathbf{w}^{(0)}\| \le \frac{1}{n}} \int |f_{\mathbf{w}}(x) - m(x)|^2 \mathbf{P}_X(dx) \right\} + \frac{A_n^d \cdot B_n^{(L-1)\cdot d}}{n^{1-\epsilon}} \right).
\end{aligned}$$

Remark 1. The right-hand side above is a sum of two terms. The first term

$$\mathbf{E}\left\{ \inf_{\mathbf{w}: \|\mathbf{w} - \mathbf{w}^{(0)}\| \le \frac{1}{n}} \int |f_{\mathbf{w}}(x) - m(x)|^2 \mathbf{P}_X(dx) \right\}$$

can be considered as the approximation error of the estimate and describes how well the unknown regression function can be approximated by deep neural networks whose inner weights are close to the randomly initialized weights at the beginning of the gradient descent. The second term

$$\frac{A_n^d \cdot B_n^{(L-1)\cdot d}}{n^{1-\epsilon}}$$

can be considered as the estimation error of the estimate. It is related to the fact that we use gradient descent to minimize the empirical L_2 risk of the estimate (i.e., the empirical L_2 risk on the training data) and not the L_2 risk.

3.2 Rate of convergence in case of a (p, C)–smooth regression function

If we impose some smoothness condition on the regression function we can derive an upper bound on the approximation error of the estimate and use it to derive a bound on the rate of convergence of the estimate. Our main result in this respect is the following corollary to Theorem 1.

Corollary 1 *Let $n \in \mathbb{N}$, let (X, Y), (X_1, Y_1), ..., (X_n, Y_n) be independent and identically distributed $\mathbb{R}^d \times \mathbb{R}$–valued random variables such that supp(X) is bounded and that (22) holds for some $c_7 > 0$. Let $p, C > 0$ where $p = q + \beta$ for some $q \in \mathbb{N}_0$ and $\beta \in (0, 1]$, and assume that the regression function $m : \mathbb{R}^d \to \mathbb{R}$ is (p, C)–smooth.*

Set $\beta_n = c_{12} \cdot \log n$ for some $c_{12} > 0$ which satisfies $c_7 \cdot c_{12} \geq 3$, and assume $c_8 > 2L$. Let $K_n \in \mathbb{N}$ be such that (23) holds for some $\kappa > 0$ and such that

$$\frac{K_n}{n^{175 \cdot (2p+d)^4 \cdot \lceil \log_2(p+d) \rceil}} \to \infty \quad (n \to \infty)$$

holds. Set

$$A = A_n = c_{15} \cdot n^{\frac{1}{2p+d}} \cdot \log n \quad \text{and} \quad B = B_n = c_{16} \cdot \log n,$$
$$L = \lceil \log_2(q + d) \rceil + 1 \quad \text{and} \quad r = 2 \cdot \lceil (2p + d)^2 \rceil$$

and define the estimate as in Section 2.

Then we have for any $\epsilon > 0$:

$$\mathbf{E} \int |m_n(x) - m(x)|^2 \mathbf{P}_X(dx) \leq c_{17} \cdot n^{-\frac{2p}{2p+d} + \epsilon}.$$

Remark 2. According to Stone (1982) the optimal minimax rate of convergence of the expected L_2 error in case of a (p, C)–smooth regression function is

$$n^{-\frac{2p}{2p+d}}$$

(cf., e.g., Chapter 3 in Györfi et al. (2002)) so the rate of convergence above is optimal up to the arbitrarily small $\epsilon > 0$ in the exponent. It is an open problem whether a corresponding result can also be shown with $\epsilon = 0$. In our proof this ϵ appears due to our use of the metric entropy bounds for bounding the complexity of our over-parametrized space of deep neural networks.

4 Application to simulated data

In this section we investigate how the estimate behaves on the simulated data.

For the simulated data we use an example from Györfi et al. (2002). Here we have $d = 1$ (so the predictor is univariate) and we choose the distribution of X to be standard normal restricted to $[-1, 1]$, i.e., the distribution of X has a density which is zero outside

of $[-1,1]$, and which is proportional to the density of the standard normal distribution on $[-1,1]$. Then we define

$$m(x) = \begin{cases} (x+2)^2/2 & \text{if } -1 \le x < -0.5, \\ x/2 + 0.875 & \text{if } -0.5 \le x < 0, \\ 5 \cdot (x-0.2)^2 + 1.075 & \text{if } 0 \le x < 0.5, \\ x + 0.125 & \text{if } 0.5 \le x \le 1, \end{cases}$$

$$\sigma(x) = 0.2 - 0.1 \cdot \cos(2 \cdot \pi \cdot x)$$

and set

$$Y = m(X) + \sigma(X) \cdot N$$

where N is a standard normally distributed random variable independent of X.

We implemented our estimate in R using the logistic squasher activation function, and the topology of the network as in (12)-(14), i.e., our network is computing a linear combination of K_n neural networks with depth L and width r. The initialization is done as described in the previous section with parameters A and B, i.e., all outer weights are initialized by zero and the weights between the hidden layers and the weights at the input layer are uniformly distributed on the intervals $[-B, B]$ and $[-A, A]$, respectively. Then we perform t_n gradient descent steps with stepsize λ_n (maybe adapted to the data as described in the previous section). Here we use the standard formulas for backpropagation in order to compute the gradient.

4.1 Do the estimates generalize well despite over-parametrization?

We compute our estimate with parameters $K \in \{100, 200, 400, 800, 1600\}$, $L = 4$, $r = 8$, $t_n = K/2$, $\lambda = 1/t_n$, $A = 1000$ and $B = 20$ for 25 data sets of sample size $n = 100$ and compute the median L_2 error and its interquartilerange (IQR). Here the deep neural network has

$$\begin{aligned} &K * (1 + (r+1) + (L-2) * r * (r+1) + r * (d+1)) \\ &= K * (1 + (8+1) + (4-2) * 8 * (8+1) + 8 * (1+1)) = K * 170 \end{aligned}$$

many weights, so it is clearly over-parametrized. The median values of the L_2 errors and the corresponding IQRs are reported in Table 1.

Typical estimates for various values of K are shown in Figure 2.

As we can see from Table 1 and Figure 2 the error of the estimate decreases with increasing K as long as $K \le 800$, and although the estimate has much more parameters than there are data points, there is no overfitting of the data visible even for $K = 1600$. We believe that the slight increase in the L_2 error of the estimate for $K = 1600$ is either due to the fact that the simple choice $t_n = 2/K$ is not optimal for a very large value of K, or that this just occurs because of random fluctuations of the median errors.

Value of K	number of parameters	median L_2 error (IQR)
100	17,000	0.0010 (0.0597)
200	34,000	0.0065 (0.0018)
400	68,000	0.0039 (0.0014)
800	136,000	0.0032 (0.0010)
1600	272,000	0.0036 (0.0016)

Table 1: Median L_2 errors (and IQRs) in 25 simulations with $n = 100$, $L = 4$, $r = 8$, $t_n = K/2$, $\lambda = 1/t_n$, $A = 1000$, $B = 20$ and $K \in \{100, 200, 400, 800, 1600\}$.

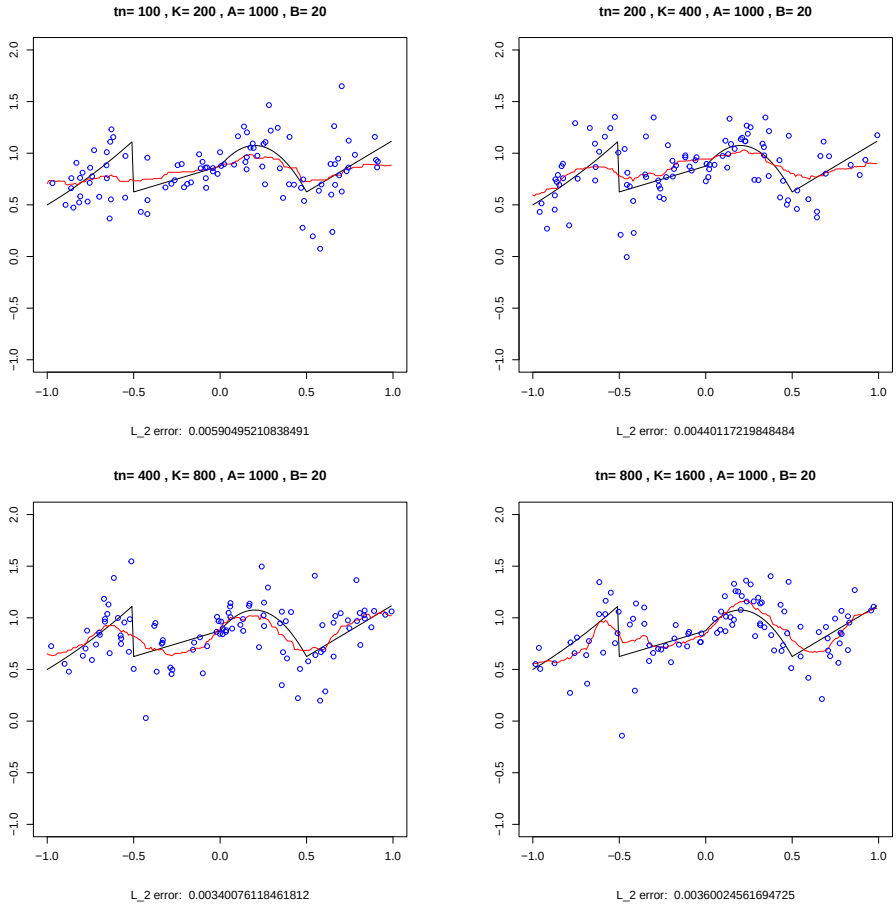

Figure 2: Estimate applied to a sample of size $n = 100$, with parameters $K \in \{200, 400, 800, 1600\}$, $L = 4$, $r = 8$, $\lambda = 2/K$, $t_n = K/2$, $A = 1000$ and $B = 20$.

4.2 Are A and B really the smoothing parameters of the estimate?

To see whether the parameters A and B of the uniform distribution are really the smoothing parameters of the estimate, we apply our estimate with $n = 100$, $K = 800$, $L = 4$,

	$A = 10$	$A = 100$	$A = 1,000$
$B = 2$	0.0106 (0.0010)	0.0095 (0.0005)	0.0097 (0.0007)
$B = 20$	0.0034 (0.0011)	0.0033 (0.0011)	0.0034 (0.0013)
$B = 200$	0.0032 (0.0018)	0.0032 (0.0010)	0.0032 (0.0022)
$B = 2000$	0.0035 (0.0010)	0.0030 (0.0016)	0.0034 (0.0021)

Table 2: Median L_2 errors (and IQRs) in 25 simulations with $n = 100$, $K = 800$, $L = 4$, $r = 8$, $t_n = 400$, $\lambda = 1/400$, $A \in \{10, 100, 1000\}$ and $B \in \{2, 20, 200, 2000\}$.

Value of K	median L_2 error (IQR)	number simulations with $t_n \neq K/2$
100	0.0082 (0.0015)	22
200	0.0059 (0.0011)	7
400	0.0044 (0.0007)	2
800	0.0034 (0.0016)	1
1600	0.0034 (0.0016)	0

Table 3: Median L_2 errors (and IQRs) in 25 simulations of the adaptive estimate for $n = 100$, $L = 4$, $r = 8$, $A = 1000$, $B = 20$ and $K \in \{100, 200, 400, 800, 1600\}$.

$r = 8$, $\lambda = 1/400$, $t_n = 400$, $A \in \{10, 100, 1000\}$ and $B \in \{2, 20, 200, 2000\}$ to 25 different data sets and report the median L_2 error and the corresponding IQR in Table 2.

We clearly see that A and B have an influence on the L_2 error. If B is too small the L_2 errors get large. Otherwise it is not clear how the values of A and B influence the errors. We think this is due to the fact that they influence simultaneously the generalization error (where larger values increase the error) and the approximation error (where large values of A decrease the approximation error, and where very large values of B might decrease the approximation error again because large values of A might result in input neurons with an nearly constant output for which a larger value of B might be an adavantage).

4.3 Do the data-dependent choices of the stepsize and the number of gradient descent steps work?

In this subsection we investigate whether the proposed data-dependent choice of the stepsize and the number of gradient descent steps improves the estimate. To do this, we apply our adaptive estimate, where the number of gradient descent steps and the stepsize is chosen as in Subsection 2.4 with $n = 100$, $K \in \{100, 200, 400, 800\}$, $L = 4$, $r = 8$, $A = 1000$ and $B = 20$ to 25 different data sets. The median values of the L_2 errors and their IQRs are reported in Table 3. There we also report in how many of the 25 simulations the adaptive estimate chooses $t_n \neq K/2$ (and hence uses a different value than the non-adaptive estimate in Table 1). In Figure 3 we show plots of typical estimates which we get for different values of K.

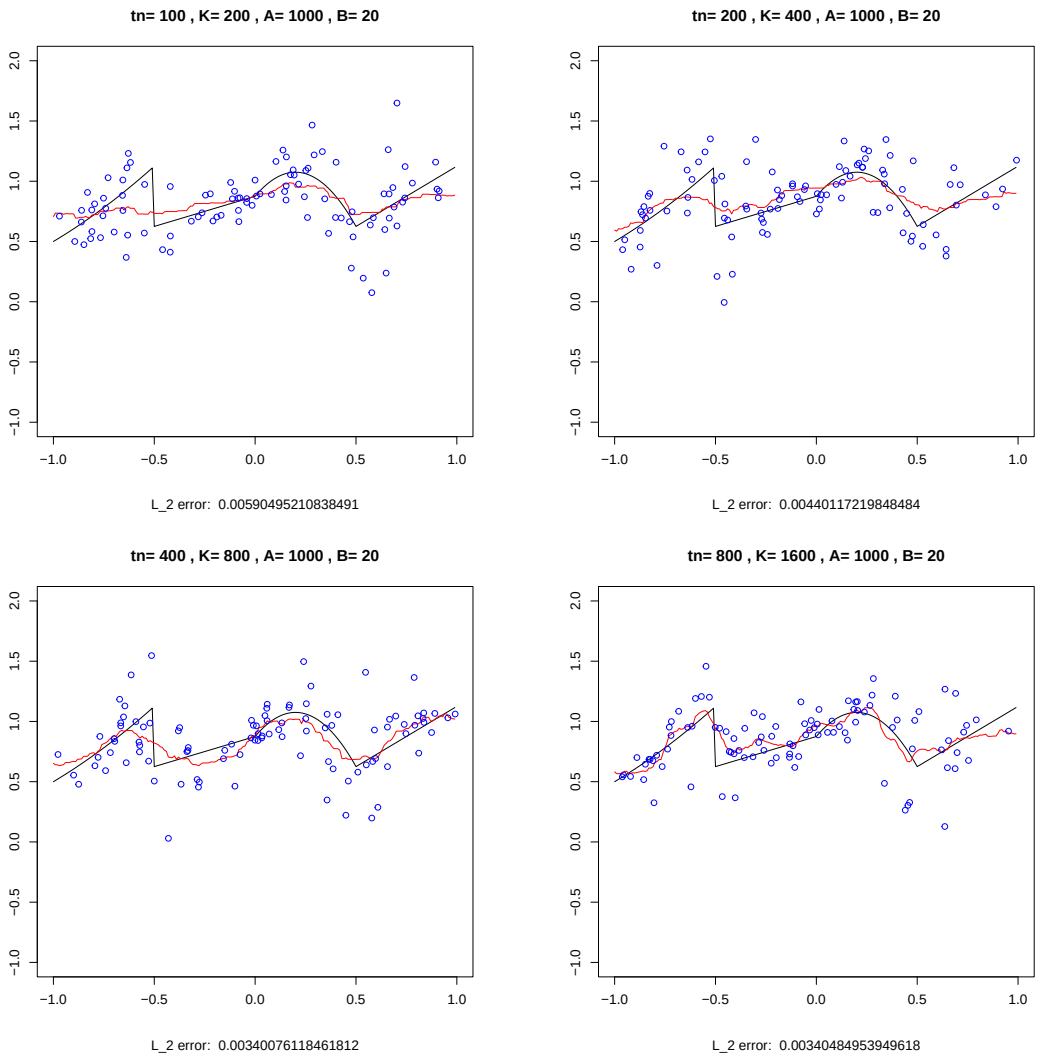

Figure 3: Adaptive estimates applied to a sample of size $n = 100$, with parameters $K \in \{200, 400, 800, 1600\}$, $L = 4$, $r = 8$, $A = 1000$ and $B = 20$.

Again the error of the estimate gets smaller with increasing K. For large values of K the adaptive algorithm always chooses $t_n = 2/K$ which explains why there is no improvement in comparison with the non adaptive choice of t_n and λ_n. However, for small values of K the median errors with the adaptive choice of λ and t_n are smaller than the median errors for $t_n = K/2$ and $\lambda_n = 1/t_n$ in Table 1.

Value of K	median L_2 error (IQR)
100	0.0069 (0.0016)
200	0.0051 (0.0011)
400	0.0046 (0.0015)
800	0.0036 (0.0014)

Table 4: Median L_2 errors (and IQRs) in 25 simulations of the adaptive estimate for $n = 100$, $L = 4$, $r = 8$ and $K \in \{100, 200, 400, 800\}$, where we choose $A \in \{10, 100, 1000\}$ and $B \in \{20, 200, 2000\}$ via splitting of the sample.

4.4 Is an adaptive choice of the weights bounds A and B during initialization useful?

We have identified the parameters A and B of the uniform distributions for the initialization of the weights as possible smoothing parameters of our neural network estimate. In this subsection we investigate whether it is useful to choose these parameters in data-dependent way using splitting of the sample (cf., e.g., Chapter 7 in Györfi et al. (2002)). Here the given data is divided into the training data consisting of the first n_l data points, and the testing data consisting of the $n_t = n - n_l$ remaining data points (e.g., with $n_l \approx n/2$ or $n_l \approx \frac{2}{3} \cdot n$). Then a finite set $\mathcal{P}$ of possible values for (A, B) is selected, for each value of (A, B) of this set the estimate

$$m_{n,(A,B)}(\cdot) = m_{n_l,(A,B)}(\cdot, \mathcal{D}_{n_l})$$

is computed using this value of (A, B) and only the training data, and finally that value $(\hat{A}, \hat{B}) \in \mathcal{P}$ is selected for which the empirical L_2 risk on the testing data is minimal. Thus, we compute

$$(\hat{A}, \hat{B}) = \arg \min_{(A,B)\in\mathcal{P}} \frac{1}{n_t} \sum_{i=n_l+1}^{n} |Y_i - m_{n_l,(A,B)}(X_i)|^2$$

and use as estimate

$$m_n(\cdot) = m_{n_l,(\hat{A},\hat{B})}(\cdot, \mathcal{D}_{n_l}).$$

We compute this estimate 25-times with $n = 100$, $n_{train} = 80$, $n_{test} = 20$, $K \in \{100, 200, 400, 800, 1600\}$, $L = 4$, $r = 8$ and choose the stepsize and the number of gradient descent steps adaptively as in the previous section and choose the parameters A and B adaptively from the sets $A \in \{10, 100, 1000\}$ and $B \in \{20, 200, 2000\}$ via splitting of the sample. The results are reported in Table 4.

Plots of typical estimates which we get for the different values of K are shown in Figure 4.

The results show that for small values of K this adaptive estimate yields a smaller error than the non-adaptive estimate. For large values of K the error of the estimate is approximately the same as for the other estimates, although it is based mainly on only 80% of the data.

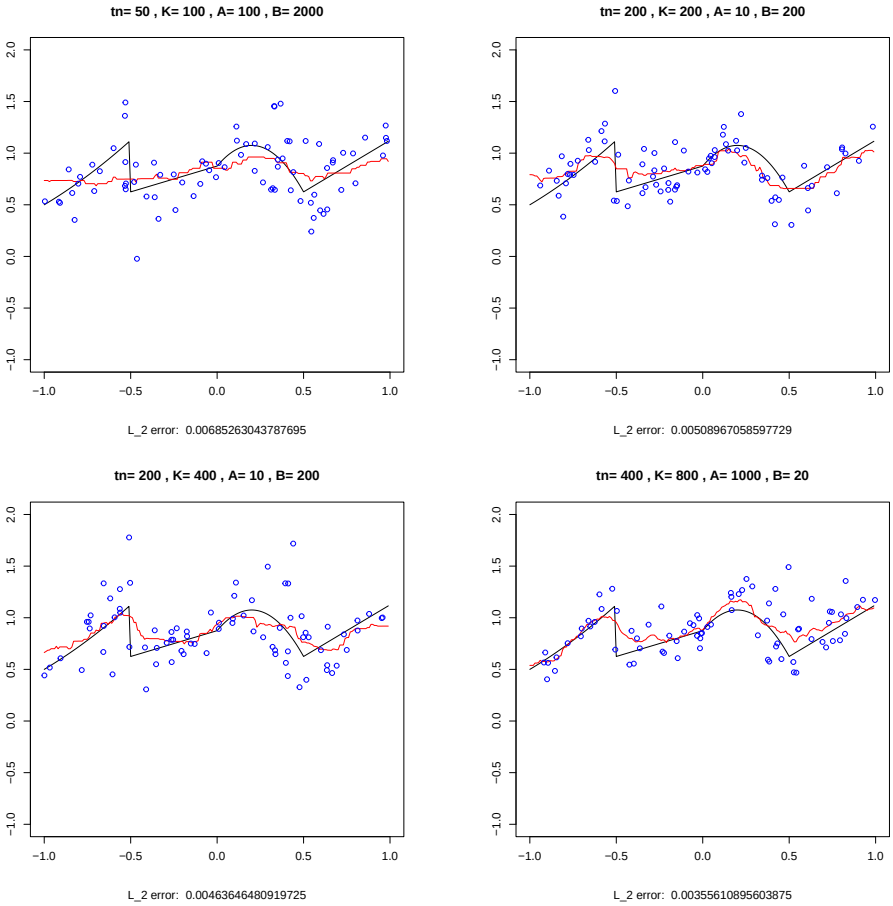

Figure 4: Estimate applied to a sample of size $n = 100$, with parameters $K \in \{100, 200, 400, 800\}$, $L = 4$, $r = 8$, adaptively chosen values for λ and t_n, and values of $A \in \{10, 100, 1000\}$ and $B \in \{20, 200, 2000\}$ chosen via splitting of the sample with $n_{train} = 80$ and $n_{test} = 20$.

4.5 How good is the estimate?

In order to see how good our newly introduced neural network regression estimate is compared with other known estimates, we apply to our data also standard neural network estimates with 2, 4 and 6 hidden layers and a data-dependent chosen number of hidden neurons, and a smoothing spline estimate. For the neural network estimates $nnfc2$, $nnfc4$ and $nnfc6$ with 2, 4 and 6 hidden layers, resp., the number $r \in \{10, 25, 50, 100, 200\}$ of hidden neurons and the number $t_n \in \{500, 1000, 2000\}$ of gradient descent steps is chosen data-dependent using splitting of the sample with $n_{train} = 80$ and $n_{test} = 20$. The estimate uses the logistic squasher as activation function and the initialization of the weights is done as before, i.e., all outer weights are initialized by zero and the weights between the hidden layers and the weights at the input layer are uniformly distributed on

Estimate	median L_2 error (IQR)
$nnfc2$	0.0080 (0.0030)
$nnfc4$	0.0099 (0.0047)
$nnfc6$	0.0100 (0.0059)
$smooth-spline$	0.0038 (0.0026)

Table 5: Median L_2 errors in 25 simulations of the three different standard neural network estimates and the smoothing spline estimate.

the intervals $[-20, 20]$ and $[-1000, 1000]$, respectively. The estimates are implemented in Python using the package *tensorflow* with gradient descent as implemented in this package using the ADAM rule for the data-dependent choice of the stepsize. The smoothing spline estimate $smooth - spline$ is applied as as implemented in R by the procedure *Tps()* from the library *fields*. The smoothing parameter of this estimate is chosen data dependent by generalized cross validation as implemented in *Tps()*. We apply each of these estimates 25 times to independent data sets of sample size $n = 100$. The results are reported in Table 5. Plots of typical estimates which we get for the different estimates are shown in Figure 5.

We see that the median L_2 errors of the neural network estimates in Table 5 are substantially larger than the median L_2 error of the smoothing spline estimate. In contrast, the newly proposed deep neural network estimates of this paper achieve for $K \geq 800$ a performance which is as good or even slightly better than this smoothing spline estimate.

This shows that our theoretical approach to deep learning improves in our example deep neural network estimates drastically such that they become comparable good as a standard estimate in an univariate regression problem. Of course, in this case the standard estimate is much easier to compute, however the potential of this result is that by modifying deep neural networks in the multivariate case in the same way (which requires an extension of the currently available theory for deep neural network estimates learned by gradient descent) might lead to an improvement of the deep neural network estimates in a case where standard estimates do not outperform them (because their results in high-dimensional settings are not as good as standard deep neural network estimates as is shown, e.g., in the simulations in Bauer and Kohler (2019)).

5 Proofs

5.1 An auxiliary result for the proof of Theorem 1

In the proof of Theorem 1 we will use the following lemma in order to analyze the gradient descent.

Lemma 1 *Let $d, J_n \in \mathbb{N}$, and for $\mathbf{w} \in \mathbb{R}^{J_n}$ let $f_{\mathbf{w}} : \mathbb{R}^d \to \mathbb{R}$ be a (deep) neural network with weight vector $\mathbf{w}$. Assume that for each $x \in \mathbb{R}^d$*

$$\mathbf{w} \mapsto f_{\mathbf{w}}(x)$$

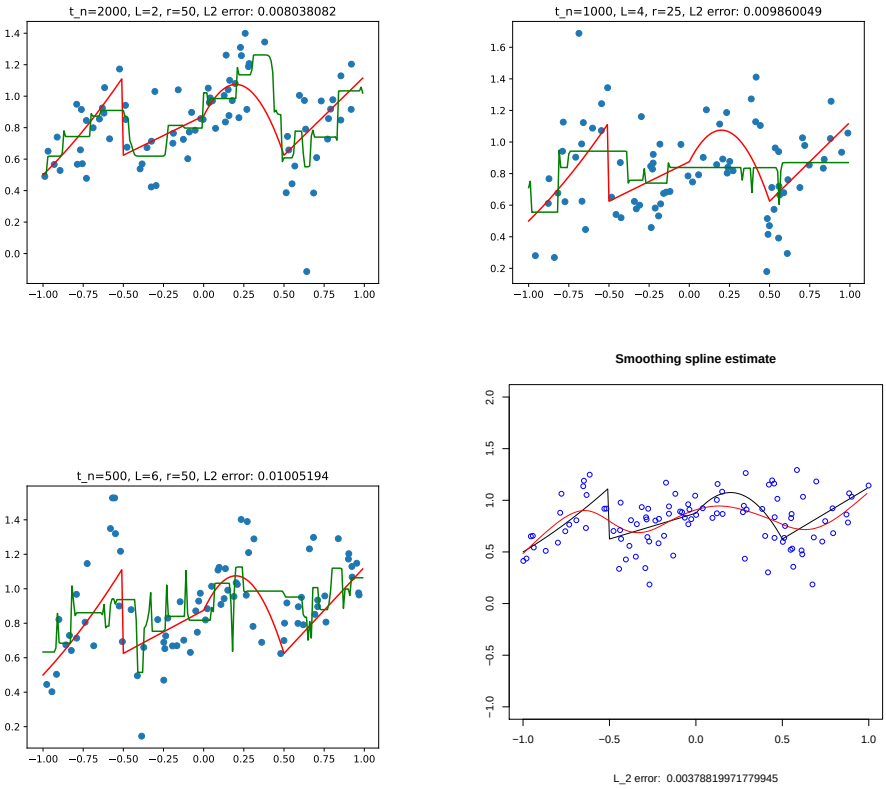

Figure 5: Standard neural network estimates with $L = 2$, $L = 4$ and $L = 6$ hidden layers and a smoothing spline estimate applied each time to a sample of size $n = 100$.

is a continuously differentiable function on $\mathbb{R}^{J_n}$. *Let*

$$F_n(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} |Y_i - f_{\mathbf{w}}(X_i)|^2$$

be the empirical L_2 *risk of* $f_{\mathbf{w}}$, *and use gradient descent in order to minimize* $F_n(\mathbf{w})$. *To do this, choose a starting weight vector* $\mathbf{w}^{(0)} \in \mathbb{R}^{J_n}$, *choose* $\delta_n \geq 0$ *and let*

$$A \subset \left\{ \mathbf{w} \in \mathbb{R}^{J_n} \, : \, \|\mathbf{w} - \mathbf{w}^{(0)}\| \leq \delta_n \right\}$$

be a closed and convex set of weight vectors. Choose a stepsize $\lambda_n > 0$ *and a number of gradient descent steps* $t_n \in \mathbb{N}$ *and compute*

$$\mathbf{w}^{(t+1)} = Proj_A \left(\mathbf{w}^{(t)} - \lambda_n \cdot \nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)}) \right)$$

for $t = 0, \ldots, t_n - 1$.

Let $C_n \geq 0$, $\beta_n \geq 1$ and assume

$$\sum_{j=1}^{J_n} \left| \frac{\partial}{\partial w^{(j)}} f_{\mathbf{w}_1}(x) - \frac{\partial}{\partial w^{(j)}} f_{\mathbf{w}_2}(x) \right|^2 \leq C_n^2 \cdot \|\mathbf{w}_1 - \mathbf{w}_2\|^2 \tag{25}$$

for all $\mathbf{w}_1, \mathbf{w}_2 \in A$, $x \in \{X_1, \ldots, X_n\}$,

$$|Y_i| \leq \beta_n \quad (i = 1, \ldots, n) \tag{26}$$

and

$$C_n \cdot \delta_n^2 \leq 1. \tag{27}$$

Let $\mathbf{w}^* \in A$ and assume

$$|f_{\mathbf{w}^*}(x)| \leq \beta_n \quad (x \in \{X_1, \ldots, X_n\}). \tag{28}$$

Then

$$\begin{aligned} \frac{1}{t_n} \sum_{t=0}^{t_n-1} F_n(\mathbf{w}^{(t)}) \quad &\leq \quad F_n(\mathbf{w}^*) + \frac{\|\mathbf{w}^* - \mathbf{w}^{(0)}\|^2}{2 \cdot \lambda_n \cdot t_n} + 3 \cdot \beta_n \cdot C_n \cdot \frac{1}{t_n} \sum_{t=0}^{t_n-1} \|\mathbf{w}^* - \mathbf{w}^{(t)}\|^2 \\ &\quad + \frac{1}{2} \cdot \lambda_n \cdot \frac{1}{t_n} \sum_{t=0}^{t_n-1} \|\nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)})\|^2. \end{aligned}$$

Proof. The result follows by a straightforward modification of the proof of Lemma 1 in Kohler (2024). For the sake of completeness we nevertheless present a more or less complete proof here.

The basic idea of the proof is to analyze the gradient descent by relating it to the gradient descent of the linear Taylor polynomial of $f_{\mathbf{w}}$. To do this, we define for $\mathbf{w}_0, \mathbf{w} \in \mathbb{R}^{J_n}$ the linear Taylor polynomial of $f_{\mathbf{w}}(x)$ around $\mathbf{w}_0$ by

$$f_{lin,\mathbf{w}_0,\mathbf{w}}(x) = f_{\mathbf{w}_0}(x) + \sum_{j=1}^{J_n} \frac{\partial f_{\mathbf{w}_0}(x)}{\partial \mathbf{w}^{(j)}} \cdot (\mathbf{w}^{(j)} - \mathbf{w}_0^{(j)})$$

and introduce the empirical L_2 risk of this linear approximation of $f_{\mathbf{w}}$ by

$$F_{n,lin,\mathbf{w}_0}(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} |Y_i - f_{lin,\mathbf{w}_0,\mathbf{w}}(X_i)|^2.$$

Then $F_{n,lin,\mathbf{w}_0}(\mathbf{w})$ is as a function of $\mathbf{w}$ a convex function (cf. Kohler (2024), proof of Lemma 1).

Because of $f_{lin,\mathbf{w}_0,\mathbf{w}_0}(x) = f_{\mathbf{w}_0}(x)$ and $\nabla_{\mathbf{w}} f_{lin,\mathbf{w}_0,\mathbf{w}_0}(x) = \nabla_w f_{\mathbf{w}_0}(x)$ we have

$$F_{n,lin,\mathbf{w}^{(t)}}(\mathbf{w}^{(t)}) = F_n(\mathbf{w}^{(t)}) \quad \text{and} \quad \nabla_w F_{n,lin,\mathbf{w}^{(t)}}(\mathbf{w}^{(t)}) = \nabla_w F_n(\mathbf{w}^{(t)}),$$

hence $\mathbf{w}^{(t+1)}$ is computed from $\mathbf{w}^{(t)}$ by one gradient descent step

$$\mathbf{w}^{(t+1)} = Proj_A \left(\mathbf{w}^{(t)} - \lambda_n \cdot \nabla_{\mathbf{w}} F_{n,lin,\mathbf{w}^{(t)}}(\mathbf{w}^{(t)}) \right)$$

applied to the convex function $F_{n,lin,\mathbf{w}^{(t)}}(\mathbf{w})$. This will enable us to use techniques for the analysis of the gradient descent for convex functions in order to analyze the gradient descent applied to the nonconvex function $F_n(\mathbf{w})$.

In order to do this we observe

$$\begin{aligned}
&\frac{1}{t_n}\sum_{t=0}^{t_n-1} F_n(\mathbf{w}^{(t)}) - F_n(\mathbf{w}^*) \\
&= \frac{1}{t_n}\sum_{t=0}^{t_n-1} (F_n(\mathbf{w}^{(t)}) - F_n(\mathbf{w}^*)) \\
&= \frac{1}{t_n}\sum_{t=0}^{t_n-1} (F_{n,lin,\mathbf{w}^{(t)}}(\mathbf{w}^{(t)}) - F_{n,lin,\mathbf{w}^{(t)}}(\mathbf{w}^*)) + \frac{1}{t_n}\sum_{t=0}^{t_n-1} (F_{n,lin,\mathbf{w}^{(t)}}(\mathbf{w}^*) - F_n(\mathbf{w}^*)) \\
&=: T_{1,n} + T_{2,n}.
\end{aligned}$$

It is shown in the proof of Lemma 1 in Kohler (2024) that assumption (25) implies

$$|f_{\mathbf{w}}(x) - f_{lin,\mathbf{w}_0,\mathbf{w}}(x)| \leq \frac{1}{2}\cdot C_n \cdot \|\mathbf{w}-\mathbf{w}_0\|^2$$

for all $x \in \{X_1,\ldots,X_n\}$ and all $\mathbf{w}_0, \mathbf{w} \in A$.

Using (26)–(28) we can conclude

$$\begin{aligned}
&|F_n(\mathbf{w}^*) - F_{n,lin,\mathbf{w}^{(t)}}(\mathbf{w}^*)| \\
&\leq \frac{1}{n}\sum_{i=1}^{n} |Y_i - f_{\mathbf{w}^*}(X_i) + Y_i - f_{lin,\mathbf{w}^{(t)},\mathbf{w}^*}(X_i)| \cdot |f_{\mathbf{w}^*}(X_i) - f_{lin,\mathbf{w}^{(t)},\mathbf{w}^*}(X_i)| \\
&\leq \frac{1}{n}\sum_{i=1}^{n} (4\cdot\beta_n + \frac{1}{2}\cdot C_n \cdot \|\mathbf{w}^* - \mathbf{w}^{(t)}\|^2) \cdot \frac{1}{2}\cdot C_n \cdot \|\mathbf{w}^* - \mathbf{w}^{(t)}\|^2 \\
&\leq \frac{1}{n}\sum_{i=1}^{n} (4\cdot\beta_n + \frac{1}{2}\cdot C_n \cdot 4\delta_n^2) \cdot \frac{1}{2}\cdot C_n \|\mathbf{w}^* - \mathbf{w}^{(t)}\|^2 \\
&\leq 3\cdot\beta_n\cdot C_n \cdot \|\mathbf{w}^* - \mathbf{w}^{(t)}\|^2.
\end{aligned}$$

This proves

$$T_{2,n} = \frac{1}{t_n}\sum_{t=0}^{t_n-1} (F_{n,lin,\mathbf{w}^{(t)}}(\mathbf{w}^*) - F_n(\mathbf{w}^*)) \leq 3\cdot\beta_n\cdot C_n \cdot \frac{1}{t_n}\sum_{t=0}^{t_n-1} \|\mathbf{w}^* - \mathbf{w}^{(t)}\|^2,$$

hence it suffices to show

$$T_{1,n} \leq \frac{\|\mathbf{w}^* - \mathbf{w}^{(0)}\|^2}{2\cdot\lambda_n\cdot t_n} + \frac{1}{2}\cdot\lambda_n\cdot\frac{1}{t_n}\sum_{t=0}^{t_n-1} \|\nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)})\|^2. \tag{29}$$

The convexity of $F_{n,lin,\mathbf{w}^{(t)}}(\mathbf{w})$ together with $\mathbf{w}^* \in A$ implies

$$F_{n,lin,\mathbf{w}^{(t)}}(\mathbf{w}^{(t)}) - F_{n,lin,\mathbf{w}^{(t)}}(\mathbf{w}^*)$$

$$
\begin{aligned}
&\leq < \nabla_{\mathbf{w}} F_{n,lin,\mathbf{w}^{(t)}}(\mathbf{w}^{(t)}), \mathbf{w}^{(t)} - \mathbf{w}^* > \\
&= < \nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)}), \mathbf{w}^{(t)} - \mathbf{w}^* > \\
&= \frac{1}{2 \cdot \lambda_n} \cdot 2 \cdot < \lambda_n \cdot \nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)}), \mathbf{w}^{(t)} - \mathbf{w}^* > \\
&= \frac{1}{2 \cdot \lambda_n} \cdot \left(\|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2 - \|\mathbf{w}^{(t)} - \mathbf{w}^* - \lambda_n \cdot \nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)})\|^2 + \|\lambda_n \cdot \nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)})\|^2 \right) \\
&= \frac{1}{2 \cdot \lambda_n} \cdot \left(\|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2 - \|\mathbf{w}^{(t)} - \lambda_n \cdot \nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)}) - \mathbf{w}^*\|^2 \right) + \frac{1}{2} \cdot \lambda_n \cdot \|\nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)})\|^2 \\
&\leq \frac{1}{2 \cdot \lambda_n} \cdot \left(\|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2 - \|Proj_A \left(\mathbf{w}^{(t)} - \lambda_n \cdot \nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)}) \right) - \mathbf{w}^*\|^2 \right) \\
&\quad + \frac{1}{2} \cdot \lambda_n \cdot \|\nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)})\|^2 \\
&= \frac{1}{2 \cdot \lambda_n} \cdot \left(\|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2 - \|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|^2 \right) + \frac{1}{2} \cdot \lambda_n \cdot \|\nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)})\|^2.
\end{aligned}
$$

This implies

$$
\begin{aligned}
T_{1,n} &\leq \frac{1}{t_n} \sum_{t=0}^{t_n - 1} \left(\frac{1}{2 \cdot \lambda_n} \cdot \left(\|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2 - \|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|^2 \right) + \frac{1}{2} \cdot \lambda_n \cdot \|\nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)})\|^2 \right) \\
&\leq \frac{\|\mathbf{w}^{(0)} - \mathbf{w}^*\|^2}{2 \cdot \lambda_n \cdot t_n} + \frac{1}{2} \cdot \frac{1}{t_n} \sum_{t=0}^{t_n - 1} \lambda_n \cdot \|\nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)})\|^2,
\end{aligned}
$$

which proves (29). □

5.2 Proof of Theorem 1

We mimick the proof of Theorem 1 in Kohler (2024).

W.l.o.g. we assume throughout the proof that n is sufficiently large and that $\|m\|_\infty \leq \beta_n$ holds. Let E_n be the event that

$$
\max_{i=1,\ldots,n} |Y_i| \leq \sqrt{\beta_n}
$$

holds.

In the *first step of the proof* we show that on E_n the conditions (17)–(19) hold (provided we replace the constant c_9 in (17)–(19) by a larger constant, which we will denote again by c_9).

To show this it suffices to show that in case

$$
\hat{t}_n \geq n \cdot (\log n)^{c_8} \cdot K_n^3
$$

conditions (17)–(19) are satisfied. Observe that in this case we have

$$
n \cdot (\log n)^{c_8} \cdot K_n^3 \leq \hat{t}_n \leq 2 \cdot n \cdot (\log n)^{c_8} \cdot K_n^3,
$$

which implies

$$\lambda_n \cdot t_n = \frac{1}{\hat{t}_n} \cdot \min\left\{\hat{t}_n, \lceil (\log n)^{c_8} \cdot K_n^3 \rceil\right\} \geq \frac{1}{2 \cdot n \cdot (\log n)^{c_8} \cdot K_n^3} \cdot \lceil (\log n)^{c_8} \cdot K_n^3 \rceil \geq \frac{1}{2 \cdot n}$$

and

$$\lambda_n \cdot t_n = \frac{1}{\hat{t}_n} \cdot \min\left\{\hat{t}_n, \lceil (\log n)^{c_8} \cdot K_n^3 \rceil\right\} \leq \frac{\lceil (\log n)^{c_8} \cdot K_n^3 \rceil}{n \cdot (\log n)^{c_8} \cdot K_n^3} \leq \frac{2}{n}.$$

On E_n we have

$$F_n(\mathbf{w}^{(0)}) = \frac{1}{n} \sum_{i=1}^{n} |Y_i - 0|^2 \leq \beta_n,$$

hence

$$\sqrt{8 \cdot \frac{t_n}{\hat{t}_n} \cdot \max\left\{F_n(\mathbf{w}^{(0)}), 1\right\}} \leq 4 \cdot \frac{\sqrt{\beta_n}}{\sqrt{n}} \leq 1$$

holds.

From this, $c_8 > 2L$ and the initial choice of $\mathbf{w}^{(0)}$ we can conclude from Lemma 3 in Kohler (2024) (which we apply with $\gamma_n^* = 1$ and $B_n = c_{13} \cdot \log n + 1$) that

$$\|\mathbf{w} - \mathbf{w}^{(0)}\| \leq \sqrt{2 \cdot \frac{t_n}{\hat{t}_n} \cdot \max\{F_n(\mathbf{w}^{(0)}), 1\}}$$

implies

$$\|\nabla_{\mathbf{w}} F_n(\mathbf{w})\| \leq c_{18} \cdot (\log n)^L \cdot K_n^{3/2} \leq \sqrt{2 \cdot t_n \cdot \hat{t}_n \cdot \max\{F_n(\mathbf{w}^{(0)}), 1\}},$$

and by Lemma 5 in Kohler (2024) we see that

$$\|\mathbf{w}_1 - \mathbf{w}^{(0)}\| \leq \sqrt{8 \cdot \frac{t_n}{\hat{t}_n} \cdot \max\{F_n(\mathbf{w}^{(0)}), 1\}}$$

and

$$\|\mathbf{w}_2 - \mathbf{w}^{(0)}\| \leq \sqrt{8 \cdot \frac{t_n}{\hat{t}_n} \cdot \max\{F_n(\mathbf{w}^{(0)}), 1\}}$$

imply

$$\|\nabla_{\mathbf{w}} F_n(\mathbf{w}_1) - \nabla_{\mathbf{w}} F_n(\mathbf{w}_2)\| \leq c_{19} \cdot K_n^{3/2} \cdot (\log n)^{2L} \cdot \|\mathbf{w}_1 - \mathbf{w}_2\| \leq \hat{t}_n \cdot \|\mathbf{w}_1 - \mathbf{w}_2\|.$$

Hence the assumptions of Lemma 4 in Kohler (2024) are satisfied, and from this lemma we immediately get

$$\|\mathbf{w}^{(t)} - \mathbf{w}^{(0)}\| \leq \sqrt{2 \cdot \frac{t_n}{\hat{t}_n} \cdot \max\{F_n(\mathbf{w}^{(0)}), 1\}} \leq \sqrt{\frac{4 \cdot \beta_n}{n}} \quad (t = 1, \ldots, t_n)$$

and

$$F_n(\mathbf{w}^{(t)}) \leq F_n(\mathbf{w}^{(t-1)}) \quad (t = 1, \ldots, t_n),$$

which implies (18) and (19). Furthermore, another application of Lemma 3 in Kohler (2024) yields

$$\frac{1}{t_n} \cdot \sum_{t=0}^{t_n-1} \lambda_n \cdot \left\| \nabla_{\mathbf{w}} F_n(\mathbf{w}^{(t)}) \right\|^2 \leq \frac{1}{\hat{t}_n} \cdot c_{20} \cdot (\log n)^{2L} \cdot K_n^3 \leq \frac{c_9}{n}$$

(where we have used $c_8 > 2L$), which completes the first step of the proof.

In the *second step of the proof* we decompose the L_2 error in a sum of several terms. To do this we set $m_{\beta_n}(x) = \mathbf{E}\{T_{\beta_n} Y | X = x\}$ and observe

$$\begin{aligned}
&\int |m_n(x) - m(x)|^2 \mathbf{P}_X(dx) \\
&= \left(\mathbf{E}\left\{|m_n(X) - Y|^2 | \mathcal{D}_n\right\} - \mathbf{E}\{|m(X) - Y|^2\}\right) \cdot 1_{E_n} + \int |m_n(x) - m(x)|^2 \mathbf{P}_X(dx) \cdot 1_{E_n^c} \\
&= \Big[\mathbf{E}\left\{|m_n(X) - Y|^2 | \mathcal{D}_n\right\} - \mathbf{E}\{|m(X) - Y|^2\} \\
&\qquad\qquad - \left(\mathbf{E}\left\{|m_n(X) - T_{\beta_n} Y|^2 | \mathcal{D}_n\right\} - \mathbf{E}\{|m_{\beta_n}(X) - T_{\beta_n} Y|^2\}\right) \Big] \cdot 1_{E_n} \\
&\quad + \Big[\mathbf{E}\left\{|m_n(X) - T_{\beta_n} Y|^2 | \mathcal{D}_n\right\} - \mathbf{E}\{|m_{\beta_n}(X) - T_{\beta_n} Y|^2\} \\
&\qquad\qquad - 2 \cdot \frac{1}{n} \sum_{i=1}^n \left(|m_n(X_i) - T_{\beta_n} Y_i|^2 - |m_{\beta_n}(X_i) - T_{\beta_n} Y_i|^2\right) \Big] \cdot 1_{E_n} \\
&\quad + \Big[2 \cdot \frac{1}{n} \sum_{i=1}^n |m_n(X_i) - T_{\beta_n} Y_i|^2 - 2 \cdot \frac{1}{n} \sum_{i=1}^n |m_{\beta_n}(X_i) - T_{\beta_n} Y_i|^2 \\
&\qquad\qquad - \left(2 \cdot \frac{1}{n} \sum_{i=1}^n |m_n(X_i) - Y_i|^2 - 2 \cdot \frac{1}{n} \sum_{i=1}^n |m(X_i) - Y_i|^2\right) \Big] \cdot 1_{E_n} \\
&\quad + \Big[2 \cdot \frac{1}{n} \sum_{i=1}^n |m_n(X_i) - Y_i|^2 - 2 \cdot \frac{1}{n} \sum_{i=1}^n |m(X_i) - Y_i|^2\Big] \cdot 1_{E_n} \\
&\quad + \int |m_n(x) - m(x)|^2 \mathbf{P}_X(dx) \cdot 1_{E_n^c} \\
&=: \sum_{j=1}^5 T_{j,n}.
\end{aligned}$$

In the remainder of the proof we bound

$$\mathbf{E} T_{j,n}$$

for $j \in \{1, \ldots, 5\}$.

In the *third step of the proof* we show

$$\mathbf{E} T_{j,n} \leq c_{21} \cdot \frac{\log n}{n} \quad \text{for } j \in \{1, 3\}.$$

This follows as in the proof of Lemma 1 in Bauer and Kohler (2019).

In the *fourth step of the proof* we show

$$\mathbf{E}T_{5,n} \leq c_{22} \cdot \frac{(\log n)^2}{n}.$$

The definition of m_n implies $\int |m_n(x) - m(x)|^2 \mathbf{P}_X(dx) \leq 4 \cdot c_{12}^2 \cdot (\log n)^2$, hence

$$\begin{aligned} \mathbf{P}(E_n^c) &\leq \mathbf{P}\{\max_{i=1,\ldots,n} |Y_i| > \sqrt{\beta_n}\} \leq n \cdot \mathbf{P}\{|Y| > \sqrt{\beta_n}\} \\ &\leq n \cdot \frac{\mathbf{E}\{\exp(c_7 \cdot Y^2)}{\exp(c_7 \cdot \beta_n)} \leq \frac{c_{23}}{n^2} \end{aligned} \tag{30}$$

where the last inequality holds because of (22) and $c_7 \cdot c_{12} \geq 3$, implies the assertion.

Let $\epsilon > 0$ be arbitrary. In the *fifth step of the proof* we show

$$\mathbf{E}T_{2,n} \leq c_{24} \cdot \frac{A_n^d \cdot B_n^{(L-1)\cdot d}}{n^{1-\epsilon}}.$$

Let $\mathcal{W}_n$ be the set of all weight vectors $(w_{i,j,k}^{(l)})_{i,j,k,l}$ which satisfy

$$|w_{k,1,1}^{(L)}| \leq c_{25} \quad (k = 1, \ldots, K_n),$$

$$|w_{k,i,j}^{(l)}| \leq c_{26} \cdot B_n \quad (l = 1, \ldots, L-1)$$

and

$$|w_{k,i,j}^{(0)}| \leq c_{27} \cdot A_n.$$

By the first step of the proof we know that on E_n condition (19) holds. From this and the initial choice of $\mathbf{w}^{(0)}$ we can conclude that on E_n we have

$$\mathbf{w}^{(t_n)} \in \mathcal{W}_n.$$

Hence, for any $u > 0$ we get

$$\begin{aligned} &\mathbf{P}\{T_{2,n} > u\} \\ &\leq \mathbf{P}\Bigg\{\exists f \in \mathcal{F}_n : \mathbf{E}\left(\left|\frac{f(X)}{\beta_n} - \frac{T_{\beta_n}Y}{\beta_n}\right|^2\right) - \mathbf{E}\left(\left|\frac{m_{\beta_n}(X)}{\beta_n} - \frac{T_{\beta_n}Y}{\beta_n}\right|^2\right) \\ &\qquad -\frac{1}{n}\sum_{i=1}^n \left(\left|\frac{f(X_i)}{\beta_n} - \frac{T_{\beta_n}Y_i}{\beta_n}\right|^2 - \left|\frac{m_{\beta_n}(X_i)}{\beta_n} - \frac{T_{\beta_n}Y_i}{\beta_n}\right|^2\right) \Bigg\} \\ &\qquad > \frac{1}{2} \cdot \left(\frac{u}{\beta_n^2} + \mathbf{E}\left(\left|\frac{f(X)}{\beta_n} - \frac{T_{\beta_n}Y}{\beta_n}\right|^2\right) - \mathbf{E}\left(\left|\frac{m_{\beta_n}(X)}{\beta_n} - \frac{T_{\beta_n}Y}{\beta_n}\right|^2\right)\right) \Bigg\}, \end{aligned}$$

where

$$\mathcal{F}_n = \{T_{\beta_n} f_{\mathbf{w}} \quad : \quad \mathbf{w} \in \mathcal{W}_n\}.$$

By Lemma 12 in Kohler (2024) we get

$$\begin{aligned}&\mathcal{N}_1\left(\delta, \left\{\frac{1}{\beta_n} \cdot f : f \in \mathcal{F}_n\right\}, x_1^n\right) \leq \mathcal{N}_1\left(\delta \cdot \beta_n, \mathcal{F}_n, x_1^n\right) \\ &\leq \left(\frac{c_{28}}{\delta}\right)^{c_{29} \cdot A_n^d \cdot B_n^{(L-1)\cdot d} \cdot \left(\frac{K_n \cdot c_{30}}{\beta_n \cdot \delta}\right)^{d/k} + c_{31}}.\end{aligned}$$

By choosing k large enough we get for $\delta > 1/n^2$

$$\mathcal{N}_1\left(\delta, \left\{\frac{1}{\beta_n} \cdot f : f \in \mathcal{F}_n\right\}, x_1^n\right) \leq c_{32} \cdot n^{c_{33} \cdot A_n^d \cdot B_n^{(L-1)\cdot d} \cdot n^{\epsilon/2}}.$$

This together with Theorem 11.4 in Györfi et al. (2002) leads for $u \geq 1/n$ to

$$\mathbf{P}\{T_{2,n} > u\} \leq 14 \cdot c_{32} \cdot n^{c_{33} \cdot A_n^d \cdot B_n^{(L-1)\cdot d} \cdot n^{\epsilon/2}} \cdot \exp\left(-\frac{n}{5136 \cdot \beta_n^2} \cdot u\right).$$

For $\epsilon_n \geq 1/n$ we can conclude

$$\begin{aligned}\mathbf{E}\{T_{2,n}\} &\leq \epsilon_n + \int_{\epsilon_n}^{\infty} \mathbf{P}\{T_{2,n} > u\}\, du \\ &\leq \epsilon_n + 14 \cdot c_{32} \cdot n^{c_{33} \cdot A_n^d \cdot B_n^{(L-1)\cdot d} \cdot n^{\epsilon/2}} \cdot \exp\left(-\frac{n}{5136 \cdot \beta_n^2} \cdot \epsilon_n\right) \cdot \frac{5136 \cdot \beta_n^2}{n}.\end{aligned}$$

Setting

$$\epsilon_n = \frac{5136 \cdot \beta_n^2}{n} \cdot c_{33} \cdot A_n^d \cdot B_n^{(L-1)\cdot d} \cdot n^{\epsilon/2} \cdot \log n = \frac{5136 \cdot \beta_n^2}{n} \cdot \log\left(n^{c_{33} \cdot A_n^d \cdot B_n^{(L-1)\cdot d} \cdot n^{\epsilon/2}}\right)$$

yields the assertion of the fourth step of the proof.

In the *sixth step of the proof* we show

$$\mathbf{E}\{T_{4,n}\} \leq c_{34} \cdot \left(\mathbf{E}\left\{\inf_{\mathbf{w}:\|\mathbf{w}-\mathbf{w}^{(0)}\| \leq \frac{1}{n}} \int |f_{\mathbf{w}}(x) - m(x)|^2 \mathbf{P}_X(dx)\right\} + \frac{(\log n)^{2L+2}}{n}\right).$$

Using

$$|T_{\beta_n} z - y| \leq |z - y| \quad \text{for } |y| \leq \beta_n$$

we get

$$\begin{aligned}&T_{4,n}/2 \\ &= \Big[\frac{1}{n}\sum_{i=1}^n |m_n(X_i) - Y_i|^2 - \frac{1}{n}\sum_{i=1}^n |m(X_i) - Y_i|^2\Big] \cdot 1_{E_n} \\ &\leq \Big[\frac{1}{n}\sum_{i=1}^n |f_{\mathbf{w}^{(t_n)}}(X_i) - Y_i|^2 - \frac{1}{n}\sum_{i=1}^n |m(X_i) - Y_i|^2\Big] \cdot 1_{E_n}\end{aligned}$$

$$\leq \Big[F_n(\mathbf{w}^{(t_n)}) - \frac{1}{n}\sum_{i=1}^{n}|m(X_i)-Y_i|^2\Big]\cdot 1_{E_n}.$$

By the first step of the proof we know that on E_n

$$\mathbf{w}^{(t)} \in A = \left\{\mathbf{w}\in\mathbb{R}^{J_n} \,:\, \|\mathbf{w}-\mathbf{w}^{(0)}\| \leq \frac{c_{35}\cdot\sqrt{\log n}}{\sqrt{n}}\right\}$$

holds for $t = 1,\ldots,t_n$. If $\mathbf{w}_1$ and $\mathbf{w}_2$ satisfy

$$\|\mathbf{w}_i - \mathbf{w}^{(0)}\| \leq \frac{c_{36}\cdot\sqrt{\log n}}{\sqrt{n}} \quad (i\in\{1,2\}),$$

then the initialization of $\mathbf{w}^{(0)}$ implies

$$|(\mathbf{w}_i)^{(L)}_{k,1,1}| \leq c_{37} \quad \text{and} \quad |(\mathbf{w}_i)^{(l)}_{k,i,j}| \leq c_{38}\cdot\log n \quad (l=1,\ldots,L-1)$$

for $i\in\{1,2\}$, and by Lemma 2 in Kohler (2024) we can conclude

$$\sum_{j=1}^{J_n}\left|\frac{\partial}{\partial w^{(j)}}f_{\mathbf{w}_1}(x) - \frac{\partial}{\partial w^{(j)}}f_{\mathbf{w}_2}(x)\right|^2 \leq c_{39}\cdot(\log n)^{4L}\cdot\|\mathbf{w}_1-\mathbf{w}_2\|^2$$

for $x\in supp(X)$. Application of Lemma 1 with A defined as above and $C_n = c_{40}\cdot(\log n)^{2L}$ yields because of $\lambda_n\cdot t_n \geq 1/2n$ (which follows from the first step of the proof)

$$\begin{aligned}
&T_{4,n}/2\\
&\leq \Big[\frac{1}{n}\sum_{i=1}^{n}|f_{\mathbf{w}^*}(X_i)-Y_i|^2 + c_{41}\cdot\frac{(\log n)^{2L+2}}{n} - \frac{1}{n}\sum_{i=1}^{n}|m(X_i)-Y_i|^2\Big]\cdot 1_{E_n}\\
&\leq \frac{1}{n}\sum_{i=1}^{n}|f_{\mathbf{w}^*}(X_i)-Y_i|^2 - \frac{1}{n}\sum_{i=1}^{n}|m(X_i)-Y_i|^2 + c_{41}\cdot\frac{(\log n)^{2L+2}}{n}\\
&\quad + \frac{1}{n}\sum_{i=1}^{n}|m(X_i)-Y_i|^2\cdot 1_{E_n^c}
\end{aligned}$$

for any $\mathbf{w}^*$ with $\|\mathbf{w}^*-\mathbf{w}^{(0)}\|\leq 1/n$. Hence using (30) we can conclude

$$\mathbf{E}\{T_{4,n}/2|\mathbf{w}^{(0)}\} \leq \int |f_{\mathbf{w}^*}(x)-m(x)|^2\mathbf{P}_X(dx) + c_{42}\cdot\frac{(\log n)^{2L+2}}{n}$$

for any $\mathbf{w}^*$ with $\|\mathbf{w}^*-\mathbf{w}^{(0)}\|\leq 1/n$, which implies

$$\mathbf{E}\{T_{4,n}/2|\mathbf{w}^{(0)}\} \leq \inf_{\mathbf{w}:\|\mathbf{w}-\mathbf{w}^{(0)}\|\leq 1/n}\int |f_{\mathbf{w}}(x)-m(x)|^2\mathbf{P}_X(dx) + c_{42}\cdot\frac{(\log n)^{2L+2}}{n}$$

and

$$\mathbf{E}\{T_{4,n}/2\} \leq \mathbf{E}\left\{\inf_{\mathbf{w}:\|\mathbf{w}-\mathbf{w}^{(0)}\|\leq 1/n}\int |f_{\mathbf{w}}(x)-m(x)|^2\mathbf{P}_X(dx)\right\} + c_{42}\cdot\frac{(\log n)^{2L+2}}{n}.$$

□

5.3 An auxiliary result for the proof of Corollary 1

In the proof of Corollary 1 we will need the following result concerning the approximation of (p, C)–smooth functions by neural networks with bounded weights.

Lemma 2 *Let $d \in \mathbb{N}$, $p = q + \beta$ where $\beta \in (0, 1]$ and $q \in \mathbb{N}_0$, $C > 0$, $A \geq 1$ and $A_n, B_n, \gamma_n^* \geq 1$. For $L, r, K \in \mathbb{N}$ let $\mathcal{F}$ be the set of all networks $f_{\mathbf{w}}$ defined by (12)-(14) with K_n replaced by r, where the weight vector satisfies*

$$|w_{i,j}^{(0)}| \leq A_n, \quad |w_{i,j}^{(l)}| \leq B_n \quad \text{and} \quad |w_{i,j}^{(L)}| \leq \gamma_n^*$$

for all $l \in \{1, \ldots, L-1\}$ and all i, j, and set

$$\mathcal{H} = \left\{ \sum_{k=1}^{K^d} f_k \quad : \quad f_k \in \mathcal{F} \quad (k = 1, \ldots, K) \right\}.$$

Let $L, r \in \mathbb{N}$ with

$$L \geq \lceil \log_2(q + d) \rceil \quad \text{and} \quad r \geq 2 \cdot (2p + d) \cdot (q + d),$$

and set

$$A_n = A \cdot K \cdot \log K, \quad B_n = c_{43} \quad \text{and} \quad \gamma_n^* = c_{44} \cdot K^{q+d}.$$

Assume $K \geq c_{45}$ for $c_{45} > 0$ sufficiently large. Then there exists for any (p, C)–smooth $f : \mathbb{R}^d \to \mathbb{R}$ a neural network $h \in \mathcal{H}$ such that

$$\sup_{x \in [-A, A)^d} |f(x) - h(x)| \leq \frac{c_{46}}{K^p}.$$

Proof. See Theorem 3 in Kohler (2024). □

5.4 Proof of Corollary 1

In the proof we will use arguments from the proof of Theorem 1 in Kohler (2024).

W.l.o.g. we assume throughout the proof that n is sufficiently large and that $\|m\|_\infty \leq \beta_n$ holds. Let $A > 0$ with $supp(X) \subseteq [-A, A]^d$. Set

$$\tilde{K}_n = \left\lceil c_{47} \cdot n^{\frac{d}{2p+d}} \right\rceil$$

and

$$N_n = \left\lceil c_{48} \cdot n^{4 + \frac{d}{2p+d}} \right\rceil$$

and let $\mathbf{w}^*$ be a weight vector of a neural networks where the results of $N_n \cdot \tilde{K}_n \cdot r$ in parallel computed neural networks with L hidden layers and r neurons per layer are computed such that the corresponding network

$$f_{\mathbf{w}^*}(x) = \sum_{k=1}^{N_n \cdot \tilde{K}_n \cdot r} (\mathbf{w}^*)_{k,1,1}^{(L)} \cdot f_{\mathbf{w}^*,k,1}^{(L)}(x)$$

satisfies

$$\sup_{x\in[-A,A]^d} |f_{\mathbf{w}^*}(x) - m(x)| \le \frac{c_{49}}{\tilde{K}_n^{p/d}} \tag{31}$$

and

$$|(\mathbf{w}^*)_{k,1,1}^{(L)}| \le \frac{c_{50}\cdot \tilde{K}_n^{(q+d)/d}}{N_n} \quad (k = 1,\dots,N_n\cdot\tilde{K}_n\cdot r).$$

Note that such a network exists according to Lemma 2 if we repeat in the outer sum of the function space $\mathcal{H}$ each of the f_k's in Lemma 2 N_n–times with outer weights divided by N_n. Set

$$\epsilon_n = \frac{c_{51}}{n\cdot\sqrt{N_n\cdot\tilde{K}_n}} \ge \frac{c_{52}}{n^4}.$$

Let E_n be the event that the weight vector $\mathbf{w}^{(0)}$ satisfies

$$|(\mathbf{w}^{(0)})_{j_s,k,i}^{(l)} - (\mathbf{w}^*)_{s,k,i}^{(l)}| \le \epsilon_n \quad \text{for all } l\in\{0,\dots,L-1\}, s\in\{1,\dots,N_n\cdot\tilde{K}_n\cdot r\}$$

for some pairwise distinct $j_1,\dots,j_{N_n\cdot\tilde{K}_n\cdot r}\in\{1,\dots,K_n\}$.

In the *first step of the proof* we show

$$\mathbf{P}(E_n^c) \le c_{53}\cdot n^6\cdot\exp(-n^{0.5}). \tag{32}$$

To do this, we consider a sequential choice of the weights of K_n fully connected neural networks. The probability that the weights in the first of these networks differ in all components at most by ϵ_n from $((\mathbf{w}^*)_{1,i,j}^{(l)})_{i,j,l:l<L}$ is for large n bounded from below by

$$\begin{aligned}&\left(\frac{c_{52}}{2\cdot c_{54}\cdot(\log n)\cdot n^4}\right)^{r\cdot(r+1)\cdot(L-1)}\cdot\left(\frac{c_{52}}{2\cdot c_{55}\cdot(\log n)\cdot n^{1/(2p+d)}\cdot n^4}\right)^{r\cdot(d+1)}\\ &\ge n^{-r\cdot(r+1)\cdot(L-1)\cdot4-r\cdot4\cdot(d+1)-r\cdot(d+1)/(2p+d)-0.5}.\end{aligned}$$

Hence probability that none of the first $n^{r\cdot(r+1)\cdot(L-1)\cdot4+r\cdot4\cdot(d+1)+r\cdot(d+1)/(2p+d)+1}$ neural networks satisfies this condition is for large n bounded above by

$$\begin{aligned}&(1-n^{-r\cdot(r+1)\cdot(L-1)\cdot4-r\cdot4\cdot(d+1)-r\cdot(d+1)/(2p+d)-0.5})^{n^{r\cdot(r+1)\cdot(L-1)\cdot4+r\cdot4\cdot(d+1)+r\cdot(d+1)/(2p+d)+1}}\\ &\le\left(\exp\left(-n^{-r\cdot(r+1)\cdot(L-1)\cdot4-r\cdot4\cdot(d+1)-r\cdot(d+1)/(2p+d)-0.5}\right)\right)^{n^{r\cdot(r+1)\cdot(L-1)\cdot4+r\cdot4\cdot(d+1)+r\cdot(d+1)/(2p+d)+1}}\\ &=\exp(-n^{0.5}).\end{aligned}$$

Since we have $K_n \ge n^{r\cdot(r+1)\cdot(L-1)\cdot4+r\cdot4\cdot(d+1)+r\cdot(d+1)/(2p+d)+1}\cdot N_n\cdot\tilde{K}_n\cdot r$ for n large we can successively use the same construction for all of $N_n\cdot\tilde{K}_n\cdot r$ weights and we can conclude: The probability that there exists $k\in\{1,\dots,N_n\cdot\tilde{K}_n\cdot r\}$ such that none of the K_n weight vectors of the fully connected neural network differs by at most ϵ_n from $((\mathbf{w}^*)_{k,i,j}^{(l)})_{i,j,l:l<L}$ is for large n bounded from above by

$$N_n\cdot\tilde{K}_n\cdot r\cdot\exp(-n^{0.5}) \le c_{56}\cdot n^6\cdot\exp(-n^{0.5}),$$

which implies the assertion of the first step of the proof.

In the *second step of the proof* we show

$$\mathbf{E}\left\{\inf_{\mathbf{w}:\|\mathbf{w}-\mathbf{w}^{(0)}\|\le\frac{1}{n}}\int|f_{\mathbf{w}}(x)-m(x)|^2\mathbf{P}_X(dx)\right\}\le c_{57}\cdot n^{-\frac{2p}{2p+d}}. \tag{33}$$

On E_n we have

$$\begin{aligned}\|\mathbf{w}^*-\mathbf{w}^{(0)}\|^2 &\le \sum_{k=1}^{N_n\cdot\tilde{K}_n\cdot r}|(\mathbf{w}^*)_{k,1,1}^{(L)}|^2+N_n\cdot\tilde{K}_n\cdot r\cdot L\cdot(r+d)^2\cdot\epsilon_n^2\\ &\le \frac{c_{50}^2\cdot\tilde{K}_n^{1+2\cdot\frac{p+d}{d}}}{N_n}+\frac{c_{51}^2\cdot r\cdot L\cdot(r+d)^2}{n^2}\\ &\le \frac{1}{n^2},\end{aligned}$$

provided n is sufficiently large. This implies

$$\begin{aligned}&\mathbf{E}\left\{\inf_{\mathbf{w}:\|\mathbf{w}-\mathbf{w}^{(0)}\|\le\frac{1}{n}}\int|f_{\mathbf{w}}(x)-m(x)|^2\mathbf{P}_X(dx)\right\}\\ &\le\mathbf{E}\left\{\int|f_{\mathbf{w}^*}(x)-m(x)|^2\mathbf{P}_X(dx)\cdot 1_{E_n}\right\}\\ &\quad+\mathbf{E}\left\{\inf_{\mathbf{w}:\|\mathbf{w}-\mathbf{w}^{(0)}\|\le\frac{1}{n}}\int|f_{\mathbf{w}}(x)-m(x)|^2\mathbf{P}_X(dx)\cdot 1_{E_n^c}\right\}\\ &\le\int|f_{\mathbf{w}^*}(x)-m(x)|^2\mathbf{P}_X(dx)+\int|0-m(x)|^2\mathbf{P}_X(dx)\cdot\mathbf{P}\{E_n^c\}\\ &\le\int|f_{\mathbf{w}^*}(x)-m(x)|^2\mathbf{P}_X(dx)+\frac{c_{58}}{n},\end{aligned}$$

where the second last inequality followed from $f_{\mathbf{w}^{(0)}}(x)=0$ for all $x\in\mathbb{R}^d$. Application of (31) yields the assertion.

In the *third step of the proof* we show the assertion.

Application of Theorem 1 with ϵ replaced by $\epsilon/2$ together with the result of the second step of the proof yields

$$\begin{aligned}&\mathbf{E}\int|m_n(x)-m(x)|^2\mathbf{P}_X(dx)\\ &\le c_{59}\cdot\left(\mathbf{E}\left\{\inf_{\mathbf{w}:\|\mathbf{w}-\mathbf{w}^{(0)}\|\le\frac{1}{n}}\int|f_{\mathbf{w}}(x)-m(x)|^2\mathbf{P}_X(dx)\right\}+\frac{A_n^d\cdot B_n^{(L-1)\cdot d}}{n^{1-\epsilon/2}}\right)\\ &\le c_{60}\cdot\left(n^{-\frac{2p}{2p+d}}+\frac{A_n^d\cdot B_n^{(L-1)\cdot d}}{n^{1-\epsilon/2}}\right)\\ &\le c_{61}\cdot\left(n^{-\frac{2p}{2p+d}}+\frac{n^{\frac{d}{2p+d}}\cdot(\log n)^{L\cdot d}}{n^{1-\epsilon/2}}\right)\end{aligned}$$

$$\leq c_{62} \cdot n^{-\frac{2p}{2p+d}+\epsilon}.$$

□

References

[1] Abbe, E., Boix-Adsera, E. and Misiakiewicz, T. (2023). SGD learning on neural networks: leap complexity and saddle-to-saddle dynamics. In: *The Thirty Sixth Annual Conference on Learning Theory*, pp. 2552-2623. PMLR.

[2] Allen-Zhu, Z., Li, Y. and Song, Z. (2019). A convergence theory for deep learning via over-parameterization. *Proceedings of the 36th International Conference on Machine Learning (PMLR 2019)*, Long Beach, California, **97**, pp. 242-252.

[3] Arous, B. G. ,Gheissari, R. and Jagannath, A. (2021). Online stochastic gradient descent on non-convex losses from high-dimensional inference. *Journal of Machine Learning Research* **22**, pp. 1–51.

[4] Ba, J., Erdogdu, M. A., Suzuki,T. , Wu, D. and Zhang, T. (2020). Generalization of two-layer neural networks: An asymptotic viewpoint. In: *International conference on learning representations*, 2020.

[5] Bartlett, P., Harvey, N., Liaw, C. and Mehrabian, A. (2019). Nearly-tight VC-dimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research* **20**, pp. 1-17.

[6] Bauer, B. and Kohler, M. (2019). On deep learning as a remedy for the curse of dimensionality in nonparametric regression. *Annals of Statistics* **4**, pp. 2261–2285.

[7] Bhattacharya, S., Fan, J. and Mukherjee, D. (2024). Deep neural networks for nonparametric interaction models with diverging dimension. *Annals of Statistics* **52**, pp. 2738–2766.

[8] Bietti, A., Bruna, J., Sanford, C. and Song, M. J. (2022). Learning single-index models with shallow neural networks. *Advances in Neural Information Processing Systems* **35**, pp. 9768–9783.

[9] Braun, A., Kohler, M., Langer, S. and Walk, H. (2023). Convergence rates for shallow neural networks learned by gradient descent. *Bernoulli* **30**, pp. 475-502.

[10] Chizat, L. and Bach, F. (2018). On the global convergence of gradient descent for over-parameterized models using optimal transport. Preprint, *arXiv: 1805.09545.*

[11] Drews, S. and Kohler, M. (2023). Analysis of the expected L_2 error of an over-parametrized deep neural network estimate learned by gradient descent without regularization. Preprint.

[12] Drews, S. and Kohler, M. (2024). On the universal consistency of an over-parametrized deep neural network estimate learned by gradient descent. *Annals of the Instititute of Statistical Mathematics* **70**, pp. 361-391.

[13] Du, S., Lee, J., Li, H., Wang, L. and Zhai, X. (2019). Gradient descent finds global minima of deep neural networks. International Conference on Machine Learning, Preprint, *arXiv: 1811.03804.*

[14] Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning.* MIT Press, Cambridge.

[15] Golowich, N., Rakhlin, A. and Shamir, O. (2019). Size-Independent sample complexity of neural networks. Preprint, *arXiv: 1712.06541.*

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

[17] Hanin, B. and Nica, M. (2019). Finite depth and width corrections to the neural tangent kernel. *arXiv: 1909.05989.*

[18] Jacot, A., Gabriel, F. and Hongler, C. (2020). Neural tangent kernel: convergence and generalization in neural networks. *arXiv: 1806.07572v4.*

[19] Kawaguchi, K and Huang, J. (2019). Gradient descent finds global minima for generalizable deep neural networks of practical sizes. *57th IEEE Annual Allerton Conference on Communication, Control, and Computing*, Allerton, IL, pp. 92-99.

[20] Kim, Y. (2014). Convolutional neural networks for sentence classification. Preprint, *arXiv: 1408.5882.*

[21] Kohler, M. (2024). On the rate of convergence of an over-parametrized deep neural network regression estimate learned by gradient descent. *arXiv: 2504.03405.*

[22] Kohler, M. and Krzyżak, A. (2017). Nonparametric regression based on hierarchical interaction models. *IEEE Transaction on Information Theory* **63**, pp. 1620-1630.

[23] Kohler, M. and Krzyżak, A. (2021). Over-parametrized deep neural networks minimizing the empirical risk do not generalize well. *Bernoulli* **27**, pp. 2564-2597.

[24] Kohler, M. and Krzyżak, A. (2022). Analysis of the rate of convergence of an over-parametrized deep neural network estimate learned by gradient descent. Preprint, *arXiv: 2210.01443.*

[25] Kohler, M. and Krzyżak, A. (2023). On the rate of convergence of an over-parametrized deep neural network regression estimate with ReLU activation function learned by gradient descent. Preprint.

[26] Kohler, M. and Langer, S. (2021). On the rate of convergence of fully connected deep neural network regression estimates using ReLU activation functions. *Annals of Statistics* **49**, pp. 2231-2249.

[27] Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira et al. (Eds.), *Advances In Neural Information Processing Systems* Red Hook, NY: Curran. **25**, pp. 1097-1105.

[28] Kutyniok, G. (2020). Discussion of "Nonparametric regression using deep neural networks with ReLU activation function". *Annals of Statistics* **48**, pp. 1902–1905.

[29] Langer, S. (2021). Approximating smooth functions by deep neural networks with sigmoid activation function. *Journal of Multivariate Analysis* **182**.

[30] Liang, T., Rakhlin, A. and Sridharan, K. (2015). Learning with square loss: localization through offset Rademacher complexity. Preprint, *arXiv: 1502.06134*.

[31] Lin, S. and Zhang, J. (2019). Generalization bounds for convolutional neural networks. Preprint, *arXiv: 1910.01487*.

[32] Lu, J., Shen, Z., Yang, H. and Zhang, S. (2020). Deep network approximation for smooth functions. *arxiv: 2001.03040*.

[33] Mei, S., Montanari, A. and Nguyen, P.-M. (2018). A mean field view of the landscape of two-layer neural networks. In *Proceedings of the National Academy of Sciences*, **115**, pp. E7665-E7671.

[34] Nguyen, P.-M. and Pham, H. T. (2020). A rigorous framework for the mean field limit of multilayer neural networks. Preprint, *arXiv: 2001.1144*.

[35] Nitanda, A. and Suzuki, T. (2021). Optimal rates for averaged stochastic gradient descent under neural tangent kernel regime. *arXiv: 2006.12297*.

[36] Schmidt-Hieber, J. (2020). Nonparametric regression using deep neural networks with ReLU activation function (with discussion). *Annals of Statistics* **48**, pp. 1875–1897.

[37] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Huber, T., et al. (2017). Mastering the game of go without human knowledge. *Nature* **550**, pp. 354-359.

[38] Stone, C. J. (1982). Optimal global rates of convergence for nonparametric regression. *Annals of Statistics*, **10**, pp. 1040-1053.

[39] Wang, M. and Ma, C. (2022). Generalization error bounds for deep neural network trained by SGD. Preprint, *arXiv: 2206.03299v1*.

[40] Wu, Y., Schuster, M., Chen, Z., Le, Q., Norouzi, M., Macherey, W., Krikum, M., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. Preprint, *arXiv: 1609.08144*.

[41] Yarotsky, D. (2018). Optimal approximation of continuous functions by very deep ReLU networks. Preprint, *arXiv: 1802.03620*.

[42] Yarotsky, D. and Zhevnerchuk, A. (2019). The phase diagram of approximation rates for deep neural networks. Preprint, *arXiv: 1906.09477*.

[43] Zou, D., Cao, Y., Zhou, D. and Gu, Q. (2018). Stochastic gradient descent optimizes over-parameterized deep ReLU networks. Preprint, *arXiv: 1811.08888*.