Towards a regularity theory for ReLU networks -- chain rule and global error estimates

J. Berner and D. Elbrächter and P. Grohs and A. Jentzen

Research Report No. 2019-26
May 2019

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

Towards a regularity theory for ReLU networks – chain rule and global error estimates

Julius Berner*, Dennis Elbrächter*, Philipp Grohs‡, Arnulf Jentzen§
*Faculty of Mathematics, University of Vienna
Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
‡Faculty of Mathematics and Research Platform DataScience@UniVienna, University of Vienna
Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
§Department of Mathematics, ETH Zürich
Rämistrasse 101, 8092 Zürich, Switzerland

***Abstract*—Although for neural networks with locally Lipschitz continuous activation functions the classical derivative exists almost everywhere, the standard chain rule is in general not applicable. We will consider a way of introducing a derivative for neural networks that admits a chain rule, which is both rigorous and easy to work with. In addition we will present a method of converting approximation results on bounded domains to global (pointwise) estimates. This can be used to extend known neural network approximation theory to include the study of regularity properties. Of particular interest is the application to neural networks with ReLU activation function, where it contributes to the understanding of the success of deep learning methods for high-dimensional partial differential equations.**

I. Introduction

It has been observed that deep neural networks exhibit the remarkable capability of overcoming the curse of dimensionality in a number of different scenarios. In particular, for certain types of high-dimensional partial differential equations (PDEs) there are promising empirical observations [1], [2], [3], [4], [5], [6], [7] backed by theoretical results for both the approximation error [8], [9], [10], [11] as well as the generalization error [12]. In this context it becomes relevant to not only show how well a given function of interest can be approximated by neural networks but also to extend the study to the derivative of this function. A number of recent publications [13], [14], [15] have investigated the required size of a network which is sufficient to approximate certain interesting (classes of) functions within a given accuracy. This is achieved, first, by considering the approximation of basic functions by very simple networks and, subsequently, by combining those networks in order to approximate more difficult structures. To extend this approach to include the regularity of the approximation, one requires some kind of chain rule for the composition of neural networks. For neural networks with differentiable activation function the standard chain rule is sufficient. It, however, fails when considering neural networks with an activation function, which is not everywhere differentiable. Although locally Lipschitz continuous functions are w.r.t the Lebesgue measure almost everywhere (a.e.) differentiable, the standard chain rule is not applicable, as, in general, it does not hold even in an 'almost everywhere' sense. We will introduce derivatives of neural networks in a way that admits a chain rule which is both rigorous as well as easy to work with. Chain rules for functions which are not everywhere differentiable have been considered in a more general setting in e.g. [16], [17]. We employ the specific structure of neural networks to get stronger results using simpler arguments. In particular it allows for a stability result, i.e. Lemma III.3, the application of which will be discussed in Section V. We would also like to mention a very recent work [18] about approximation in Sobolev norms, where they deal with the issue by using a general bound for the Sobolev norm of the composition of functions from the Sobolev space $W^{1,\infty}$. Note however that this approach leads to a certain factor depending on the dimensions of the domains of the functions, which can be avoided with our method. For ease of exposition, we formulate our results for neural networks with the ReLU activation function. We, however, consider in Section IV how such a chain rule can be obtained for any activation function which is locally Lipschitz continuous (with at most countably many points at which it is not differentiable). In Section V we briefly sketch how the results from Section III can be utilized to get approximation results for certain classes of functions. Subsequently, in Section VI, we present a general method of deriving global error estimates from such approximation results, which are naturally obtained for bounded domains. Ultimately, we discuss how our results can be used to extend known theory, enabling the further study of the approximation of PDE solutions by neural networks.

II. Setting

As in [14], we consider a neural network Φ to be a finite sequence of matrix-vector pairs, i.e.

$$\Phi = ((A_k, b_k))_{k=1}^L, \tag{1}$$

where $A_k \in \mathbb{R}^{N_k \times N_{k-1}}$ and $b_k \in \mathbb{R}^{N_k}$ for some depth $L \in \mathbb{N}$ and layer dimensions $N_0, N_1, \ldots, N_L \in \mathbb{N}$. The realization of the neural network Φ is the function $\mathcal{R}\Phi\colon \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ given by

$$\mathcal{R}\Phi = W_L \circ \mathrm{ReLU} \circ W_{L-1} \circ \ldots \circ \mathrm{ReLU} \circ W_1, \tag{2}$$

where $W_k(x) = A_k x + b_k$ for every $x \in \mathbb{R}^{N_k}$ and where

$$\mathrm{ReLU}(x) := (\max\{0, x_1\}, \ldots, \max\{0, x_N\}) \tag{3}$$

for every $x \in \mathbb{R}^N$. We distinguish between a neural network and its realization, since Φ uniquely induces $\mathcal{R}\Phi$, while in general there can be multiple non-trivially different neural networks with the same realization. The representation of a neural network as a structured set of weights as in (1) allows the introduction of notions of network sizes. While there are slight differences between various publications, commonly considered quantities are the depth (i.e. number of affine transformations), the connectivity (i.e. number of non-zero entries of the A_k and b_k), and the weight bound (i.e. maximum of the absolute values of the entries of the A_k and b_k). In [15] it has been shown that these three quantities determine the length of a bit string which is sufficient to encode the network with a prescribed quantization error. In the following let

$$\Phi = ((A_k, b_k))_{k=1}^{L}, \quad \Psi = ((\tilde{A}_k, \tilde{b}_k))_{k=1}^{\tilde{L}} \tag{4}$$

be neural networks with matching dimensions in the sense that $\mathcal{R}\Phi\colon \mathbb{R}^d \to \mathbb{R}^m$ and $\mathcal{R}\Psi\colon \mathbb{R}^m \to \mathbb{R}^n$. We then define their composition as

$$\begin{aligned}\Psi \odot \Phi :=& \\ &\big(((A_k, b_k))_{k=1}^{L-1}, (\tilde{A}_1 A_L, \tilde{A}_1 b_L + \tilde{b}_1), ((\tilde{A}_k, \tilde{b}_k))_{k=2}^{\tilde{L}}\big).\end{aligned} \tag{5}$$

Direct computation shows

$$\mathcal{R}(\Psi \odot \Phi) = \mathcal{R}\Psi \circ \mathcal{R}\Phi. \tag{6}$$

Note that the realization $\mathcal{R}\Phi$ of a neural network Φ is continuous piecewise linear (CPL) as a composition of CPL functions. Consequently, it is Lipschitz continuous and the realization $\mathcal{R}\Phi$ is almost everywhere differentiable by Rademacher's theorem. In particular all three functions in (6) are a.e. differentiable. This, however, is not sufficient to get the derivative of $\mathcal{R}(\Psi \odot \Phi)$ from the derivatives of $\mathcal{R}\Psi$ and $\mathcal{R}\Phi$ by use of the classical chain rule. Consider the very simple counterexample of $u(x) := \operatorname{ReLU}(x)$ and $v(x) := 0$ and formally apply the chain rule, i.e.

$$(D(u \circ v))(x) = (Du)(v(x)) \cdot (Dv)(x). \tag{7}$$

Even though $(Du)(y)$ is well-defined for every $y \in \mathbb{R}\backslash\{0\}$, the expression $(Du)(v(x))$ is defined for no $x \in \mathbb{R}$. In general this problem occurs when the inner function maps a set of positive measure into a set where the derivative of the outer function does not exist. Now in this case, one can directly see that setting $(Du)(0)$ to any arbitrary value would cause (7) to provide the correct result since $(Dv)(x) = 0$.

III. ReLU network derivative

We proceed by defining the derivative of an arbitrary neural network in a way such that it not only coincides a.e. with the derivative of the realization, but also admits a chain rule. To this end let $H\colon \mathbb{R}^N \to \mathbb{R}^{N \times N}$ be the function given by

$$H(x) := \operatorname{diag}(\mathbb{1}_{(0,\infty)}(x_1), \ldots, \mathbb{1}_{(0,\infty)}(x_N)) \tag{8}$$

for every $x = (x_1, \ldots, x_N) \in \mathbb{R}^N$ and let $\mathcal{R}_K\Phi := \mathcal{R}((A_k, b_k))_{k=1}^{K}$. We then define the neural network derivative of Φ as the function $\mathcal{D}\Phi\colon \mathbb{R}^{N_0} \to \mathbb{R}^{N_L \times N_0}$ given by

$$\mathcal{D}\Phi := A_L \cdot H(\mathcal{R}_{L-1}\Phi) \cdot A_{L-1} \cdot \ldots \cdot H(\mathcal{R}_1\Phi) \cdot A_1. \tag{9}$$

Note that this definition is motivated by formally applying the chain rule with the convention that the derivative of $\max\{0, \cdot\}$ is zero at the origin. Now we need to verify that this is justified.

Theorem III.1. *It holds for almost every $x \in \mathbb{R}^d$ that*

$$(\mathcal{D}\Phi)(x) = (D(\mathcal{R}\Phi))(x). \tag{10}$$

Proof. Let $v\colon \mathbb{R}^d \to \mathbb{R}^N$ be a locally Lipschitz continuous function, define $w := \operatorname{ReLU} \circ v$, and

$$L_i := \{x \in \mathbb{R}^d\colon w_i(x) = 0\} = \{x \in \mathbb{R}^d\colon v_i(x) \leq 0\}. \tag{11}$$

We now use an observation about differentiability on level sets (see e.g. [19, Thm 3.3(i)]), which states that

$$(Dw_i)(x) = 0 \quad \text{for almost every } x \in L_i. \tag{12}$$

As $w_i(x) = v_i(x)$ for every $x \in \mathbb{R}^d \backslash L_i$, we get a.e.

$$Dw_i = \mathbb{1}_{\mathbb{R}^d \backslash L_i} \cdot Dv_i = \mathbb{1}_{(0,\infty)}(v_i) \cdot Dv_i \tag{13}$$

and consequently

$$D(\operatorname{ReLU} \circ v) = H(v) \cdot Dv. \tag{14}$$

The claim follows by induction over the layers $K = 1, \ldots, L$ of Φ, using (14) with $v = \mathcal{R}_K\Phi$ for the induction step. □

Note that even for convex $\mathcal{R}\Phi$ the values of $\mathcal{D}\Phi$ on the nullset do not necessarily lie in the respective subdifferentials of $\mathcal{R}\Phi$, as can be seen in Figure 1. Although Theorem III.1 holds regardless of which value is chosen for the derivative of $\max\{0, \cdot\}$ at the origin, no choice will guarantee that all values of $\mathcal{D}\Phi$ lie in the respective subdifferentials of $\mathcal{R}\Phi$. Here we have set the derivative at the origin to zero, following the convention of software implementations for deep learning applications, e.g. TensorFlow and PyTorch. Using (5) and (9) one can verify by direct computation that $\mathcal{D}$ obeys the chain rule.

Corollary III.2. *It holds for every $x \in \mathbb{R}^d$ that*

$$(\mathcal{D}(\Psi \odot \Phi))(x) = (\mathcal{D}\Psi)(\mathcal{R}\Phi(x)) \cdot (\mathcal{D}\Phi)(x). \tag{15}$$

Note that (15) is well-defined as $\mathcal{D}\Psi$ exists everywhere, although it only coincides with $D(\mathcal{R}\Psi)$ almost everywhere. Theorem III.1 however guarantees that we still have a.e.

$$\mathcal{D}(\Psi \odot \Phi) = D(\mathcal{R}(\Psi \odot \Phi)) = D(\mathcal{R}\Psi \circ \mathcal{R}\Phi). \tag{16}$$

Next we provide a technical result dealing with the stability of our chain rule, which will prove to be useful in Section V.

Lemma III.3. *It holds for almost every $x \in \mathbb{R}^d$ that*

$$\lim_{y \to \mathcal{R}\Phi(x)} \big[(\mathcal{D}\Psi)(y) - (\mathcal{D}\Psi)(\mathcal{R}\Phi(x))\big] \cdot (\mathcal{D}\Phi)(x) = 0. \tag{17}$$

Proof. We first show for every locally Lipschitz continuous function $u\colon \mathbb{R}^m \to \mathbb{R}^N$ and for almost every $x \in \mathbb{R}^d$ that

$$\lim_{y \to \mathcal{R}\Phi(x)} [H(u(y)) - H(u(\mathcal{R}\Phi(x)))] \cdot (D(u \circ \mathcal{R}\Phi))(x) = 0. \tag{18}$$

If $u_i(\mathcal{R}\Phi(x)) \neq 0$ we have

$$\lim_{y \to \mathcal{R}\Phi(x)} \mathbb{1}_{(0,\infty)}(u_i(y)) = \mathbb{1}_{(0,\infty)}(u_i(\mathcal{R}\Phi(x))) \tag{19}$$

as u_i is continuous and $\mathbb{1}_{(0,\infty)}$ is continuous on $\mathbb{R}\backslash\{0\}$. Furthermore, [19, Thm 3.3(i)] implies that

$$(D(u_i \circ \mathcal{R}\Phi))(x) = 0 \tag{20}$$

for almost every $x \in \mathbb{R}^d$ with $u_i(\mathcal{R}\Phi(x)) = 0$. Since a finite union of nullsets is again a nullset, this proves the claim (18). The lemma follows by induction over the layers $K = 1, \dots, \tilde{L}$ of Ψ and applying (18) with $u = \mathcal{R}_K\Psi$. □

IV. General Activation Functions

As mentioned in the introduction, it is possible to replace the ReLU activation function in (2) by some locally Lipschitz continuous, component-wise applied function $\varrho\colon \mathbb{R} \to \mathbb{R}$ with an at most countably large set S of points where ϱ is not differentiable. Specifically, one can define the neural network derivative (with activation function ϱ) as in (9) with $\mathbb{1}_{(0,\infty)}(x_i)$ in (8) replaced by

$$(\bar{D}\varrho)(x_i) := \begin{cases} 0, & x_i \in S \\ (D\varrho)(x_i), & \text{else} \end{cases}. \tag{21}$$

The chain rule can, again, be checked by direct computation and it is straightforward to adapt Theorem III.1 to this more general setting by considering the level sets

$$\{x \in \mathbb{R}^d \colon w_i(x) = s\}, \quad s \in S. \tag{22}$$

If additionally $\bar{D}\varrho$ is continuous on $\mathbb{R} \setminus S$, the proof of Lemma III.3 translates without any modifications.

V. Utilization in Approximation Theory

These results can now be employed to bound the L^∞-norm of $\mathcal{D}(\Psi \circ \Phi) - D(u \circ v)$, given corresponding estimates for the approximation of u and v by Ψ and Φ, respectively. Here, one has to take some care when bounding the term

$$\left\| \left[\mathcal{D}\Psi \circ \mathcal{R}\Phi - Du \circ \mathcal{R}\Phi\right] \mathcal{D}\Phi \right\|_{L^\infty} \tag{23}$$

by

$$\|\mathcal{D}\Psi - Du\|_{L^\infty} \|\mathcal{D}\Phi\|_{L^\infty}. \tag{24}$$

Again it can happen that $\mathcal{R}\Phi$ maps a set of positive measure into a nullset where the estimate for the approximation of Du by $\mathcal{D}\Psi$ in the *essential* supremum norm is not valid. However, using the stability result in Lemma III.3 one can for almost every $x \in \mathbb{R}^d$ shift to a sufficiently close point $y \approx \mathcal{R}\Phi(x)$ where the estimate holds. In [13] Yarotsky explicitly constructs networks whose realization is a linear interpolation[1] of the squaring function (see Fig. 1 for illustration), which directly gives an estimate on the approximation rate for the derivatives. These simple networks can then be combined to get networks approximating multiplication, polynomials and eventually, by means of e.g. local Taylor approximation, functions f whose first $n \geq 1$ (weak) derivatives are bounded. This leads to estimates of the form

$$\|f - \mathcal{R}\Phi_{\varepsilon,B}\|_{L^\infty(I_B)} \leq \varepsilon, \tag{25}$$

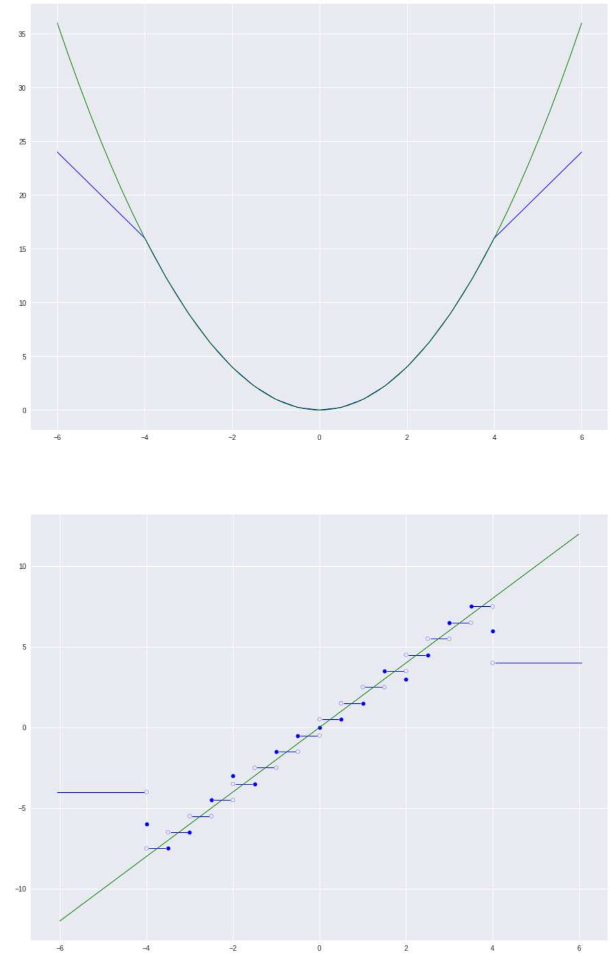

Fig. 1. Approximation of the function $x \mapsto x^2$ and its derivative on the interval $[-4, 4]$ by a neural network Φ with depth 6, connectivity 52 and weight bound 4. Note that not all values of $\mathcal{D}\Phi$ at the points of non-differentiablity of $\mathcal{R}\Phi$ lie between the values at either side, i.e. in the subdifferential.

with $I_B = [-B, B]^d$, including estimates for the scaling of the size of the network $\Phi_{\varepsilon,B}$ w.r.t. B and ε. As these constructions are based on composing simpler functions with known estimates one can now employ Theorem III.1 and Corollary III.2 to show that the derivatives of those networks also approximate the derivative of the function, i.e.

$$\|Df - \mathcal{D}\Phi_{\varepsilon,B}\|_{L^\infty(I_B)} \leq c\,\varepsilon^r. \tag{26}$$

Such constructive approaches can further be found in [8], in [14] for β-cartoon-like functions, in [20] for $(\boldsymbol{b}, \varepsilon)$-holomorphic maps, and in [15] for high-frequent sinusoidal functions.

VI. Global Error Estimates

The error estimates above are usually only sensible for bounded domains, as the realization of a neural network is always CPL with a finite number of pieces. We briefly discuss a general way of transforming them into global pointwise error estimates, which can be useful in the context of PDEs (see e.g.

[1] The interpolation points are uniformly distributed over the domain of approximation and their number grows exponentially with the size of the networks.

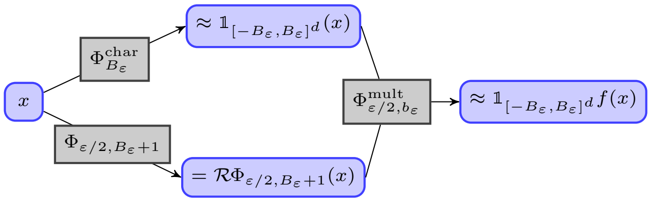
x
$\Phi^{\mathrm{char}}_{B_\varepsilon}$
$\approx \mathbb{1}_{[-B_\varepsilon,B_\varepsilon]^d}(x)$
$\Phi_{\varepsilon/2,B_\varepsilon+1}$
$= \mathcal{R}\Phi_{\varepsilon/2,B_\varepsilon+1}(x)$
$\Phi^{\mathrm{mult}}_{\varepsilon/2,b_\varepsilon}$
$\approx \mathbb{1}_{[-B_\varepsilon,B_\varepsilon]^d} f(x)$

Fig. 2. The neural networks Φ_ε approximating f globally.

[9], [10]). In the following assume that we have a function f with an at most polynomially growing derivative, i.e.

$$\|(Df)(x)\|_2 \le \boldsymbol{c}(1 + \|x\|_2^\kappa). \tag{27}$$

Denote by Φ_B^{char} a neural network which represents the d-dimensional approximate characteristic function of I_B, i.e. $\mathcal{R}\Phi_B^{\mathrm{char}}(x) \in [0,1]$ and

$$\begin{aligned} \mathcal{R}\Phi_B^{\mathrm{char}}(x) &= 1, \quad x \in I_B, \\ \mathcal{R}\Phi_B^{\mathrm{char}}(x) &= 0, \quad x \notin I_{B+1}. \end{aligned} \tag{28}$$

See [15, Proof of Thm. VIII.3] for such a construction. Further let $\Phi^{\mathrm{mult}}_{\varepsilon,b}$ be the neural network approximating the multiplication function on $[-b,b]^2$ with error ε (see e.g. [20, Prop. 3.1]).

Now we define the global approximation networks Φ_ε as the composition of $\Phi^{\mathrm{mult}}_{\varepsilon/2,b_\varepsilon}$ with the parallelization of $\Phi^{\mathrm{char}}_{B_\varepsilon}$ and $\Phi_{\varepsilon/2,B_\varepsilon+1}$ for suitable

$$B_\varepsilon \in \mathcal{O}(\varepsilon^{-1}) \quad \text{and} \quad b_\varepsilon \in \mathcal{O}(\varepsilon^{-\kappa-1}). \tag{29}$$

See Figure 2 for an illustration and e.g. [14, Def. 2.7] for a formal definition of parallelization. Considering the errors on I_B, $I_{B+1}\backslash I_B$ and $\mathbb{R}^d\backslash I_{B+1}$ leads to global estimates, i.e. for every $x \in \mathbb{R}^d$

$$|f(x) - \mathcal{R}\Phi_\varepsilon(x)| \le \varepsilon(1 + \|x\|_2^{\kappa+2}) \tag{30}$$

and, by use of the chain rule III.2, for almost every $x \in \mathbb{R}^d$

$$\|(Df)(x) - (\mathcal{D}\Phi_\varepsilon)(x)\|_2 \le C\varepsilon^r(1 + \|x\|_2^{\kappa+2}). \tag{31}$$

Due to the logarithmic size scaling of the multiplication network, the size of Φ_ε can be bounded by the size of $\Phi_{\varepsilon/2,B_\varepsilon+1}$ plus an additional term in $\mathcal{O}(d + \kappa \log \varepsilon^{-1})$.

VII. Application to PDEs

Analyzing the regularity properties of neural networks was motivated by the recent successful application of deep learning methods to PDEs [2], [3], [4], [5], [6], [7], [11]. Initiated by empirical experiments [1] it has been proven that neural networks are capable of overcoming the curse of dimensionality for solving so-called Kolmogorov PDEs [12]. More precisely, the solution to the empirical risk minimization problem over a class of neural networks approximates the solution of the PDE up to error ε with high probability and with size of the networks and number of samples scaling only polynomially in the dimension d and ε^{-1}. The above requires a suitable learning problem and a sufficiently good approximation of the solution function by neural networks. For Kolmogorov PDEs, this boils down to calculating global Lipschitz coefficients and error estimates for neural networks approximating the initial condition and coefficient functions (see e.g. [9], [10]). Employing estimates of the form (26) one can bound the derivative on I_B, i.e.

$$L_B := \|\mathcal{D}\Phi_{\varepsilon,B}\|_{L^\infty(I_B)} \le \|Df\|_{L^\infty(I_B)} + c\varepsilon^r. \tag{32}$$

Using mollification and the mean value theorem we can establish local Lipschitz estimates, i.e. for all $x, y \in (-B,B)^d$ that

$$|\mathcal{R}\Phi_{\varepsilon,B}(x) - \mathcal{R}\Phi_{\varepsilon,B}(y)| \le L_B\|x - y\|_2, \tag{33}$$

and corresponding linear growth bounds

$$|\mathcal{R}\Phi_{\varepsilon,B}(x)| \le \big(|\mathcal{R}\Phi_{\varepsilon,B}(0)| + L_B\big)(1 + \|x\|_2). \tag{34}$$

Similarly, one can use (31) to obtain estimates of the form

$$|\mathcal{R}\Phi_\varepsilon(x) - \mathcal{R}\Phi_\varepsilon(y)| \le \boldsymbol{C}(1 + \|x\|_2^{\kappa+2} + \|y\|_2^{\kappa+2})\|x - y\|_2 \tag{35}$$

for all $x, y \in \mathbb{R}^d$ (which are demanded in [10, Theorem 1.1]). Moreover, note that the capability to produce approximation results which include error estimates for the derivative is of significant independent interest. Various numerical methods (for instance Galerkin methods) rely on bounding the error in some Sobolev norm $\|\cdot\|_{W^{1,p}}$, which requires estimates of the derivative differences. We believe that the possibility to obtain regularity estimates significantly contributes to the mathematical theory of neural networks and allows for further advances in the numerical approximation of high dimensional partial differential equations.

VIII. Relation to Backpropagation in Training

The approach discussed here could further be applied to the training of neural networks by (stochastic) gradient descent. Note, however, that this is a slightly different setting. From the approximation theory perspective we were interested in the derivative of $x \mapsto \mathcal{R}\Phi(x)$, while in training one requires the derivative of $\Phi \mapsto \mathcal{R}\Phi(x^*)$ for some fixed sample x^*. In particular this function is no longer CPL but rather continuous piecewise polynomial. While this would necessitate some technical modifications, we believe that it should be possible to employ the method used here in order to show that the gradient of $\Phi \mapsto \mathcal{R}\Phi(x^*)$ coincides a.e. with what is computed by backpropagation using the convention of setting the derivative of $\max\{0,\cdot\}$ to zero at the origin (as well as similar conventions for e.g. max-pooling).

Acknowledgment

The research of JB and DE was supported by the Austrian Science Fund (FWF) under grants I3403-N32 and P 30148.

References

[1] C. Beck, S. Becker, P. Grohs, N. Jaafari, and A. Jentzen, "Solving stochastic differential equations and Kolmogorov equations by means of deep learning," *arXiv:1806.00421*, 2018.

[2] W. E, J. Han, and A. Jentzen, "Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations," *Communications in Mathematics and Statistics*, vol. 5, no. 4, pp. 349–380, 2017.

[3] J. Han, A. Jentzen, and W. E, "Solving high-dimensional partial differential equations using deep learning," *arXiv:1707.02568*, 2017.

[4] J. Sirignano and K. Spiliopoulos, "DGM: A deep learning algorithm for solving partial differential equations," *arXiv:1708.07469*, 2017.

[5] M. Fujii, A. Takahashi, and M. Takahashi, "Asymptotic Expansion as Prior Knowledge in Deep Learning Method for high dimensional BSDEs," *arXiv:1710.07030*, 2017.

[6] Y. Khoo, J. Lu, and L. Ying, "Solving parametric PDE problems with artificial neural networks," *arXiv:1707.03351*, 2017.

[7] W. E and B. Yu, "The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems," *arXiv:1710.00211*, 2017.

[8] D. Elbrächter, P. Grohs, A. Jentzen, and C. Schwab, "DNN Expression Rate Analysis of high-dimensional PDEs: Application to Option Pricing," *arXiv:1809.07669*, 2018.

[9] P. Grohs, F. Hornung, A. Jentzen, and P. von Wurstemberger, "A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations," *arXiv:1809.02362*, 2018.

[10] A. Jentzen, D. Salimova, and T. Welti, "A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients," *arxiv:1809.07321*, 2018.

[11] M. Hutzenthaler, A. Jentzen, T. Kruse, and T. A. Nguyen, "A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations," *arXiv:1707.02568*, 2019.

[12] J. Berner, P. Grohs, and A. Jentzen, "Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations," *arXiv:1809.03062*, 2018.

[13] D. Yarotsky, "Optimal approximation of continuous functions by very deep ReLU networks," *arXiv:1802.03620*, 2018.

[14] P. Petersen and F. Voigtlaender, "Optimal approximation of piecewise smooth functions using deep ReLU neural networks," *arXiv:1709.05289*, 2017.

[15] P. Grohs, D. Perekrestenko, D. Elbrächter, and H. Bölcskei, "Deep Neural Network Approximation Theory," *arxiv:1901.02220*, 2019.

[16] F. Murat and C. Trombetti, "A chain rule formula for the composition of a vector-valued function by a piecewise smooth function," *Bollettino dell'Unione Matematica Italiana*, vol. 6, no. 3, pp. 581–595, 2003.

[17] L. Ambrosio and G. Dal Maso, "A general chain rule for distributional derivatives," *Proceedings of the American Mathematical Society*, vol. 108, no. 3, pp. 691–702, 1990.

[18] I. Gühring, G. Kutyniok, and P. Petersen, "Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms," *arXiv:1902.07896*, 2019.

[19] L. C. Evans and R. F. Gariepy, *Measure Theory and Fine Properties of Functions, Revised Edition*, ser. Textbooks in Mathematics. CRC Press, 2015.

[20] C. Schwab and J. Zech, "Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in UQ," *Analysis and Applications, Singapore*, vol. 17, no. 1, pp. 19–55, 2019.