Neural Networks 137 (2021) 119–126

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

The Kolmogorov–Arnold representation theorem revisited☆

Johannes Schmidt-Hieber

University of Twente and Leiden University, Drienerlolaan 5, 7522 NB Enschede, The Netherlands

ARTICLE INFO

Article history:
Received 31 July 2020
Revised and accepted 21 January 2021
Available online 29 January 2021

Keywords:
Kolmogorov–Arnold representation theorem
Function approximation
Deep ReLU networks
Space-filling curves

ABSTRACT

There is a longstanding debate whether the Kolmogorov–Arnold representation theorem can explain the use of more than one hidden layer in neural networks. The Kolmogorov–Arnold representation decomposes a multivariate function into an interior and an outer function and therefore has indeed a similar structure as a neural network with two hidden layers. But there are distinctive differences. One of the main obstacles is that the outer function depends on the represented function and can be wildly varying even if the represented function is smooth. We derive modifications of the Kolmogorov–Arnold representation that transfer smoothness properties of the represented function to the outer function and can be well approximated by ReLU networks. It appears that instead of two hidden layers, a more natural interpretation of the Kolmogorov–Arnold representation is that of a deep neural network where most of the layers are required to approximate the interior function.

1. Introduction

Why are additional hidden layers in a neural network helpful? The Kolmogorov–Arnold representation (KA representation in the following) seems to offer an answer to this question as it shows that every continuous function can be represented by a specific network with two hidden layers (Hecht-Nielsen, 1987). But this interpretation has been highly disputed. Articles discussing the connection between both concepts have titles such as "Representation properties of networks: Kolmogorov's theorem is irrelevant" (Girosi & Poggio, 1989) and "Kolmogorov's theorem is relevant" (Kurkova, 1991).

The original version of the KA representation theorem states that for any continuous function $f : [0,1]^d \to \mathbb{R}$, there exist univariate continuous functions g_q, $\psi_{p,q}$ such that

$$f(x_1,\ldots,x_d) = \sum_{q=0}^{2d} g_q\Big(\sum_{p=1}^{d} \psi_{p,q}(x_p)\Big). \tag{1.1}$$

This means that the $(2d+1)(d+1)$ univariate functions g_q and $\psi_{p,q}$ are enough for an exact representation of a d-variate function. Kolmogorov published the result in 1957 disproving the statement of Hilbert's 13th problem that is concerned with the solution of algebraic equations. The earliest proposals in the literature introducing multiple layers in neural networks date back to the sixties and the link between KA representation and multilayer neural networks occurred much later.

A ridge function is a function of the form $f(\mathbf{x}) = \sum_{p=1}^{m} g_p(\mathbf{w}_p^\top \mathbf{x})$, with vectors $\mathbf{w}_p \in \mathbb{R}^d$ and univariate functions g_p. The structure of the KA representation can therefore be viewed as the composition of two ridge functions. There exists no exact representation of continuous functions by ridge functions and matching upper and lower bounds for the best approximations are known (Gordon, Maiorov, Meyer, & Reisner, 2002; Maiorov, 1999; Maiorov, Meir, & Ratsaby, 1999). The composition structure is thus essential for the KA representation. A two-hidden-layer feedforward neural network with activation function σ, hidden layers of width m_1 and m_2, and one output unit can be written in the form

$$f(\mathbf{x}) = \sum_{q=1}^{m_1} d_q\, \sigma\Big(\sum_{p=1}^{m_2} b_{pq}\sigma(\mathbf{w}_p^\top \mathbf{x} + a_p) + c_q\Big),$$

with parameters $\mathbf{w}_p \in \mathbb{R}^d$, $a_p, b_{pq}, c_q, d_q \in \mathbb{R}$.

Because of the similarity between the KA representation and neural networks, the argument above suggests that additional hidden layers can lead to unexpected features of neural network functions.

There are several reasons why the Kolmogorov–Arnold representation theorem has been initially declared as irrelevant for neural networks in Girosi and Poggio (1989). The original proof of the KA representation in Kolmogorov (1957) and some later versions are non-constructive providing very little insight on how the function representation works. Although the $\psi_{p,q}$ are

☆ The research has been supported by the Dutch STAR network and a Vidi grant from the Dutch science organization (NWO), The Netherlands. This work was done while the author was visiting the Simons Institute for the Theory of Computing. The constructive comments and suggestions shared by the associate editor and the three referees resulted in a significantly improved version of the article. The author wants to thank Matus Telgarsky for helpful remarks and pointing to the article Siegelmann and Sontag (1994).

E-mail addresses: a.j.schmidt-hieber@utwente.nl, schmidthieberaj@math.leidenuniv.nl.

https://doi.org/10.1016/j.neunet.2021.01.020

continuous, they are still rough functions sharing similarities with the Cantor function. Meanwhile more refined KA representation theorems have been derived strengthening the connection to neural networks (Braun & Griebel, 2009; Sprecher, 1965, 1996, 1997). Maiorov and Pinkus (1999) showed that the KA representation can essentially be rewritten in the form of a two-hidden-layer neural network for a non-computable activation function σ, see the literature review in Section 4 for more details. The following KA representation is much more explicit and practical.

Theorem 1 (*Theorem 2.14 in Braun, 2009*)**.** *Fix $d \geq 2$. There are real numbers a, b_p, c_q and a continuous and monotone function $\psi : \mathbb{R} \to \mathbb{R}$, such that for any continuous function $f : [0,1]^d \to \mathbb{R}$, there exists a continuous function $g : \mathbb{R} \to \mathbb{R}$ with*

$$f(x_1, \ldots, x_d) = \sum_{q=0}^{2d} g\Big(\sum_{p=1}^{d} b_p \psi(x_p + qa) + c_q\Big).$$

This representation is based on translations of one inner function ψ and one outer function g. The inner function ψ is independent of f. The dependence on q in the first layer comes through the shifts qa. The right hand side can be realized by a neural network with two hidden layers. The first hidden layer has d units and activation function ψ and the second hidden layer consists of $2d+1$ units with activation function g.

For a given $0 < \beta \leq 1$, we will assume that the represented function f is β-smooth, which here means that there exists a constant C, such that $|f(\mathbf{x}) - f(\mathbf{y})| \leq C\|\mathbf{x} - \mathbf{y}\|_\infty^\beta$ for all $\mathbf{x}, \mathbf{y} \in [0,1]^d$. Let $m > 0$ be arbitrary. To approximate a β-smooth function up to an error $m^{-\beta}$, it is well-known that standard approximation schemes need at least of the order of m^d parameters. This means that any efficient neural network construction mimicking the KA representation and approximating β-smooth functions up to error $m^{-\beta}$ should have at most of the order of m^d many network parameters.

Starting from the KA representation, the objective of the article is to derive a deep ReLU network construction that is optimal in terms of number of parameters. For that reason, we first present novel versions of the KA representation that are easy to prove and also allow to transfer smoothness from the multivariate function to the outer function. In Section 3 the link is made to deep ReLU networks.

The efficiency of the approximating neural network is also the main difference to the related work (Kurkova, 1992; Montanelli & Yang, 2020). Based on sigmoidal activation functions, the proof of Theorem 2 in Kurkova (1992) proposes a neural network construction based on the KA representation with two hidden layers and $dm(m+1)$ and $m^2(m+1)^d$ hidden units to achieve approximation error of the order of $m^{-\beta}$. This means that more than m^{4+d} network weights are necessary, which is sub-optimal in view of the argument above. The very recent work (Montanelli & Yang, 2020) uses a modern version of the KA representation that guarantees some smoothness of the interior function. Combined with the general result on function approximation by deep ReLU networks in Yarotsky (2018), a rate is derived that depends on the smoothness of the outer function via the function class $K_C([0,1]^d; \mathbb{R})$, see p. 4 in Montanelli and Yang (2020) for a definition. The non-trivial dependence of the outer function on the represented function f makes it difficult to derive explicit expressions for the approximation rate if f is β-smooth. Moreover, as the KA representation only guarantees low regularity of the interior function, it remains unclear whether optimal approximation rates can be obtained.

Although the deep ReLU network proposed in Shen, Yang, and Zhang (2020) is not motivated by the KA representation or space-filling curves, the network construction is quite similar. Section 4 contains a more detailed comparison with this and other related approaches.

2. New versions of the KA representation

The starting point of our work is the apparent connection between the KA representation and space-filling curves (Gorchakov & Mozolenko, 2019; Sprecher & Draghici, 2002). A space-filling curve γ is a surjective map $[0,1] \to [0,1]^d$. This means that it hits every point in $[0,1]^d$ and thus "fills" the cube $[0,1]^d$. Known constructions are based on iterative procedures producing fractal-type shapes. If γ^{-1} exists, we could then rewrite any function $f : [0,1]^d \to \mathbb{R}$ in the form

$$f = \underbrace{(f \circ \gamma)}_{=:g} \circ \gamma^{-1}. \tag{2.1}$$

This would decompose the function f into a function $\gamma^{-1} : \mathbb{R}^d \to [0,1]$ that can be chosen to be independent of f and a univariate function $g = f \circ \gamma : [0,1] \to \mathbb{R}$ containing all the information of the d-variate function f. Compared to the KA representation, there are two differences. Firstly, the interior function γ^{-1} is d-variate and not univariate. Secondly, by Netto's theorem (Kupers), a continuous surjective map $[0,1] \to [0,1]^2$ cannot be injective and γ^{-1} does not exist. The argument above can therefore not be made precise for arbitrary dimension d and a continuous space-filling curve γ.

To illustrate our approach, we first derive a simple KA representation based on (2.1) and with γ^{-1} an additive function. The identity avoids the continuity of the functions ψ and g, which is the major technical obstacle in the proof of the KA representation. The proof does moreover not require that the represented function f is continuous.

Lemma 1. *Fix integers $d, B \geq 2$. There exists a monotone function $\psi : [0,1] \to \mathbb{R}$ such that for any function $f : [0,1]^d \to \mathbb{R}$, we can find a function $g : \mathbb{R} \to \mathbb{R}$ with*

$$f(x_1, \ldots, x_d) = g\Big(\sum_{p=1}^{d} B^{-p} \psi(x_p)\Big). \tag{2.2}$$

Proof. The B-adic representation of a number is not unique. For the decimal representation, 1 is for instance the same as $1 = 0.999\ldots$ To avoid any problems that this may cause, we select for each real number $x \in [0,1]$ one B-adic representation $x = \sum_{j\geq 1} B^{-j} a_j^x$ with $a_j^x \in \{0, \ldots, B-1\}$. Throughout the following, it is often convenient to rewrite x in its B-adic expansion. Set

$$x = \sum_{j=1}^{\infty} \frac{a_j^x}{B^j} =: [0.a_1^x a_2^x a_3^x \ldots]_B$$

and define the function

$$\psi(x) = \sum_{j=1}^{\infty} \frac{a_j^x}{B^{d(j-1)}}.$$

The function ψ is monotone and maps x to a number with B-adic representation

$$[a_1^x . \underbrace{0 \ldots\ldots 0}_{(d-1)\text{-times}} a_2^x \underbrace{0 \ldots\ldots 0}_{(d-1)\text{-times}} a_3^x 0 \ldots\ldots]_B$$

inserting always $d-1$ zeros between the original B-adic digits of x. Multiplication by B^{-p} shifts moreover the digits by p places to the right. From that we obtain the B-adic representation

$$\Psi(x_1, \ldots, x_d) := \sum_{p=1}^{d} B^{-p} \psi(x_p) = \big[0.a_1^{x_1} a_1^{x_2} \ldots a_1^{x_d} a_2^{x_1} \ldots\big]_B \tag{2.3}$$

Because we can recover $x_1, \ldots, x_d$ from $\Psi(x_1, \ldots, x_d)$, the map Ψ is invertible. Denote the inverse by Ψ^{-1}. We can now define $g = f \circ \Psi^{-1}$ and this proves the result. □

The proof provides some insights regarding the structure of the KA representation. Although one might find the construction of $\Psi : [0,1]^d \to [0,1]$ in the proof very artificial, a substantial amount of neighborhood information persists under Ψ. Indeed, points that are close are often mapped to nearby values. If for instance $\mathbf{x}_1, \mathbf{x}_2 \in [0,1]^d$ are two points coinciding in all components up to the kth B-adic digit, then, $\Psi(\mathbf{x}_1)$ and $\Psi(\mathbf{x}_2)$ coincide up to the kd-th B-adic digit. In this sense, the KA representation can be viewed as a two step procedure, where the first step Ψ does some extreme dimension reduction. Compared to low-dimensional random embeddings which by the Johnson–Lindenstrauss lemma nearly preserve the Euclidean distances among points, there seems, however, to be no good general characterization of how the interior function changes distances.

The function Ψ is discontinuous at all points with finite B-adic representation. The map Ψ defines moreover an order relation on $[0,1]^d$ via $\mathbf{x} < \mathbf{y} :\Leftrightarrow \Psi(\mathbf{x}) < \Psi(\mathbf{y})$. For $B = d = 2$, the inverse map Ψ^{-1} is often called the Morton order and coincides, up to a rotation of 90 degrees, with the z-curve in the theory of space-filling curves (Bader, 2013, Section 7.2).

If f is a piecewise constant function on a dyadic grid, the outer function g is also piecewise constant. As a negative result, we show that for this representation, smoothness of f does not translate into smoothness on g.

Lemma 2. *Let k be a positive integer. Consider representation (2.2) for $B = 2$ and let g be as in the proof of Lemma 1.*

(i) If $f : \mathbb{R}^d \to \mathbb{R}$ is piecewise constant on the 2^{kd} hypercubes $\times_{j=1}^d (\ell_j 2^{-k}, (\ell_j+1)2^{-k})$, with $\ell_1, \dots, \ell_d \in \{0, 2^k - 1\}$, then g is a piecewise constant function on the intervals $(\ell 2^{-kd}, (\ell+1)2^{-kd})$, $\ell = 0, \dots, 2^{kd} - 1$.

(ii) If $f(x) = x$, then g is discontinuous.

Proof. *(i)* If $x \in (\ell 2^{-kd}, (\ell+1)2^{-kd})$, we can write $x = \Delta + \ell 2^{-kd}$ with $0 < \Delta < 2^{-kd}$. There exist thus $\ell_1, \dots, \ell_d \in \{0, 2^k - 1\}$, such that $\Psi^{-1}(x) = \Psi^{-1}(\Delta) + (\ell_1 2^{-k}, \dots, \ell_d 2^{-k})$. Since $\Psi^{-1}(\Delta) \in (0, 2^{-k}) \times \cdots \times (0, 2^{-k})$, the result follows from $g = f \circ \Psi^{-1}$.
(ii) If f is the identity, $g = f \circ \Psi^{-1} = \Psi^{-1}$. For $x \uparrow 1/2$, we find that $\Psi^{-1}(x) \to (1/2, 1, 1, \dots, 1)$ and for $x \downarrow 1/2$, $\Psi^{-1}(x) \to (1/2, 0, 0, \dots, 0)$. Even stronger, every point with finite binary representation is a point of discontinuity. □

The discontinuity of the space-filling map Ψ^{-1} causes g to be more irregular than f. Many constructions of space-filling curves are known but to obtain a representation of KA type Ψ needs to be an additive function. The additivity condition rules out most of the canonical choices, such as for instance the Hilbert curve. Below, we use for Ψ^{-1} the Lebesgue curve and show that this then leads to a representation that allows to transfer smoothness properties of f to smoothness properties on g and therefore overcomes the shortcomings of the representation in (2.2). In contrast to the earlier result, g is now a function that maps from the Cantor set, in the following denoted by $\mathcal{C}$, to the real numbers.

Theorem 2. *For fixed dimension $d \geq 2$, there exists a monotone function $\phi : [0,1] \to \mathcal{C}$ (the Cantor set) such that for any function $f : [0,1]^d \to \mathbb{R}$, we can find a function $g : \mathcal{C} \to \mathbb{R}$ such that*

(i)

$$f(x_1, \dots, x_d) = g\Big(3 \sum_{p=1}^d 3^{-p} \phi(x_p)\Big); \tag{2.4}$$

(ii) if $f : [0,1]^d \to \mathbb{R}$ is continuous, then also $g : \mathcal{C} \to \mathbb{R}$ is continuous;

(iii) if there exist $\beta \leq 1$ and a constant Q, such that $|f(\mathbf{x}) - f(\mathbf{y})| \leq Q|\mathbf{x} - \mathbf{y}|_\infty^\beta$, for all $\mathbf{x}, \mathbf{y} \in [0,1]^d$, then,

$$|g(x) - g(y)| \leq 2^\beta Q |x - y|^{\frac{\beta \log 2}{d \log 3}}, \quad \text{for all } x, y \in \mathcal{C};$$

(iv) if $f(\mathbf{x}) = \mathbf{x}$, then, there exist sequences $(x_k)_k, (y_k)_k \subset \mathcal{C}$ with $\lim_k x_k = \lim_k y_k$ and

$$\big|g(x_k) - g(y_k)\big|_\infty = \Big(\frac{|x_k - y_k|}{2}\Big)^{\frac{\log 2}{d \log 3}}.$$

Proof. The construction of the interior function is similar as in the proof of Lemma 1. We associate with each $x \in [0,1]$ one binary representation $x = [0.a_1^x a_2^x \dots]_2$ and define

$$\phi(x) := \sum_{j=1}^\infty \frac{2a_j^x}{3^{1+d(j-1)}} = \big[0.(2a_1^x)\underbrace{0 \dots \dots 0}_{(d-1)\text{-times}}(2a_2^x)\underbrace{0 \dots \dots 0}_{(d-1)\text{-times}}\big]_3. \tag{2.5}$$

The function ϕ multiplies the binary digits by two (thus, only the values 0 and 2 are possible) and then expresses the digits in a ternary expansion adding $d-1$ zeros between each two digits. By construction, the Cantor set consists of all $y \in [0,1]$ that only have 0 and 2 as digits in the ternary expansion. This shows that $\phi : [0,1] \to \mathcal{C}$. Define now

$$\begin{aligned} \Phi(x_1, \dots, x_d) &:= 3 \sum_{p=1}^d 3^{-p} \phi(x_p) \\ &= \big[0.(2a_1^{x_1})(2a_1^{x_2}) \dots (2a_1^{x_d})(2a_2^{x_1}) \dots\big]_3 \end{aligned} \tag{2.6}$$

where the right hand side is written in the ternary system. Because we can recover the binary representation of $x_1, \dots, x_d$, the map Φ is invertible. Since $2a_\ell^{x_r} \in \{0, 2\}$ for all $\ell \geq 1$ and $r \in \{1, \dots, d\}$, the image of Φ is contained in the Cantor set. We can now define the inverse by $\Phi^{-1} : \mathcal{C} \to \mathbb{R}$ and set $g = f \circ \Phi^{-1} : \mathcal{C} \to \mathbb{R}$, proving *(i)*.

In the next step of the proof, we show that

$$\big|\Phi^{-1}(x) - \Phi^{-1}(y)\big|_\infty \leq 2|x - y|^{\log 2/(d \log 3)}, \quad \text{for all } x, y \in \mathcal{C}, \tag{2.7}$$

For that we extend the proof in Bader (2013, p. 98). Observe that Φ^{-1} maps $x = [0.x_1 x_2 x_3 \dots]_3$ to the vector $([0.(x_1/2)(x_{d+1}/2) \dots]_2, \dots, [0.(x_d/2)(x_{2d}/2) \dots]_2)^\top \in [0,1]^d$. Given arbitrary $x, y \in \mathcal{C}$, $k^* = k^*(x,y)$ denotes the integer k for which $3^{-(k+1)d} \leq |x - y| < 3^{-kd}$. Suppose that the first $k^* d$ ternary digits of x and y are not all the same and denote by J the position of the first digit of x that is not the same as y. Since only the digits 0 and 2 are possible, the difference between x and y can be lower bounded by $|x - y| \geq 2 \cdot 3^{-J} - 3^{-J}$, where the term -3^{-J} accounts for the effect of the later digits. Thus $|x - y| \geq 3^{-J}$ and this is a contradiction with $|x - y| < 3^{-k^* d}$ and $J \leq k^* d$. Thus, the first $k^* d$ ternary digits of x and y coincide. Using the explicit form of Φ^{-1} this also implies that $\Phi^{-1}(x)$ and $\Phi^{-1}(y)$ coincide in the first k^* binary digits in each component. This means that $|\Phi^{-1}(x) - \Phi^{-1}(y)|_\infty \leq 2^{-k^*}$ and together with the definition of k^*, we find

$$\begin{aligned} \big|\Phi^{-1}(x) - \Phi^{-1}(y)\big|_\infty &\leq 2 \cdot 2^{-(k^*+1)} = 2\big(3^{-(k^*+1)d}\big)^{\frac{\log 2}{d \log 3}} \\ &\leq 2|x - y|^{\frac{\log 2}{d \log 3}} \end{aligned} \tag{2.8}$$

proving (2.7), since $x, y \in \mathcal{C}$ were arbitrary. Using again that $g = f \circ \Phi^{-1}$, *(ii)* and *(iii)* follow.

To prove *(iv)*, take $x_k = 0$ and $y_k = 2/3^{kd}$. Then, $\Phi^{-1}(x_k) = (0, \dots, 0)^\top$ and $\Phi^{-1}(y_k) = (0, \dots, 0, 2^{-k})^\top$. Rewriting this yields $|g(x_k) - g(y_k)|_\infty = |\Phi^{-1}(x_k) - \Phi^{-1}(y_k)|_\infty = (|x_k - y_k|/2)^{\frac{\log 2}{d \log 3}}$ for all $k \geq 1$. □

Thus, by restricting to the Cantor set, one can overcome the limitations of Netto's theorem mentioned at the beginning of the section. In fact by construction and (2.8), $\gamma = \Phi^{-1}$ is a surjective, invertible and continuous space-filling curve. The previous theorem is in a sense more extreme than the KA representation as the univariate interior function maps to a set of Hausdorff dimension $\log 2/\log 3 < 1$. Siegelmann and Sontag (1994) use a similar construction to prove embeddings of the function spaces generated by circuits into neural network function classes.

Representation (2.4) has the advantage that smoothness imposed on f translates into smoothness properties on g. The reason is that the function Φ associates to each $\mathbf{x} \in [0,1]^d$ one value in the Cantor set such that values that are far from each other in $[0,1]^d$ are not mapped to nearby values in the Cantor set. Based on this value in the Cantor set, the outer function g reconstructs the function value $f(\mathbf{x})$. Since the distance of the values in $[0,1]^d$ is linked to the distance of the values in the Cantor set, local variability in the function f does not lead to arbitrarily large fluctuations of the outer function g. Therefore smoothness imposed on f translates into smoothness properties on g.

A natural question is whether we gain or lose something if instead of approximating f directly, we use (2.4) and approximate g. Recall that the approximation rate should be $m^{-\beta}$ if m^d is the number of free parameters of the approximating function, β the smoothness and d the dimension. Since g is by (iii) α-smooth with $\alpha = \beta \log 2/(d \log 3)$ and is defined on a set with Hausdorff dimension $d^* = \log 2/\log 3$, we see that there is no loss in terms of approximation rates since $\beta/d = \alpha/d^*$. Thus, we can reduce multivariate function approximation to univariate function approximation on the Cantor set. This, however, only holds for $\beta \leq 1$. Indeed, the last statement of the previous theorem means that for the smooth function $f(x) = x$, the outer function g is not more than $\beta \log 2/(d \log 3)$-smooth, implying that for higher order smoothness, there seems to be a discrepancy between the multivariate and univariate function approximation.

The only direct drawback of (2.4) compared to the traditional KA representation is that the interior function ϕ is discontinuous. We will see in Section 3 that ϕ can, however, be well approximated by a deep neural network.

It is also of interest to study the function class containing all f that are generated by the representation in (2.4) for β-smooth outer function g. Observe that if $g(x) = x$, then f coincides with the interior function which is discontinuous. This shows that for $\beta \leq 1$, the class of all f of the form (2.4) with g a $\beta \log 2/(d \log 3)$-smooth function on the Cantor set $\mathcal{C}$ is strictly larger than the class of β-smooth functions. Interestingly, the function class with Lipschitz continuous outer function g contains all functions that are piecewise constant on a dyadic partition of $[0,1]^d$.

Lemma 3. *Consider representation (2.4) and let k be a positive integer. If $f : [0,1]^d \to \mathbb{R}$ is piecewise constant on the 2^{kd} hypercubes $\times_{j=1}^d [\ell_j 2^{-k}, (\ell_j+1)2^{-k})$, with $\ell_1, \ldots, \ell_d \in \{0, 2^k - 1\}$, then g is a Lipschitz function with Lipschitz constant bounded by $2\|f\|_\infty 3^{kd}$.*

Proof. Let ϕ and Φ be the same as in the proof of Theorem 2. For any vector $\mathbf{a} = (a_1, \ldots, a_{kd}) \in \{0,2\}^{kd}$ define $I(\mathbf{a}) = \{[0.a_1 \ldots a_{kd} b_1 b_2 \ldots]_3 : b_1, b_2, \ldots \in \{0,2\}\}$. There exist integers $\ell_1, \ldots, \ell_d \in \{0, \ldots, 2^k - 1\}$ such that $\Phi^{-1}(I(\mathbf{a})) \subseteq \times_{j=1}^d [\ell_j 2^{-k}, (\ell_j+1)2^{-k})$. Since f is constant on these dyadic hypercubes, $g(I(\mathbf{a})) = (f \circ \Phi^{-1})(I(\mathbf{a})) = \text{const}$. If $\mathbf{a}, \widetilde{\mathbf{a}} \in \{0,2\}^{kd}$ and $\mathbf{a} \neq \widetilde{\mathbf{a}}$, then, arguing as in the proof of Theorem 2, we find that $|x-y| \geq 3^{-kd}$ whenever $x \in I(\mathbf{a})$ and $y \in I(\widetilde{\mathbf{a}})$. Therefore, we have $|g(x) - g(y)| = 0$ if $x, y \in I(\mathbf{a})$ and $|g(x) - g(y)| \leq 2\|g\|_\infty \leq 2\|f\|_\infty \leq 2\|f\|_\infty 3^{kd}|x-y|$ if $x \in I(\mathbf{a})$ and $y \in I(\widetilde{\mathbf{a}})$. Since $\mathbf{a}, \widetilde{\mathbf{a}}$ were arbitrary, the result follows. □

It is important to realize that the space-filling curves and fractal shapes occur because of the exact identity. It is natural to wonder whether the KA representation leads to an interesting approximation theory. For that, one wants to truncate the number of digits in (2.5), hence reducing the complexity of the interior function. We obtain an approximation bound that only depends on the dimension d through the smoothness of the outer function g.

Lemma 4. *Let $d \geq 2$ and suppose K is a positive integer. For $x = [0.a_1^x a_2^x \ldots]_2$, define $\phi_K(x) := \sum_{j=1}^K 2a_j^x 3^{-1-d(j-1)}$. If there exist $\beta \leq 1$ and a constant Q, such that $|f(\mathbf{x}) - f(\mathbf{y})| \leq Q|\mathbf{x}-\mathbf{y}|_\infty^\beta$, for all $\mathbf{x}, \mathbf{y} \in [0,1]^d$, then, we can find a univariate function g, such that $|g(x) - g(y)| \leq 2^\beta Q |x-y|^{\frac{\beta \log 2}{d \log 3}}$, for all $x, y \in \mathcal{C}$, and*

$$\Big|f(\mathbf{x}) - g\Big(3\sum_{p=1}^d 3^{-p}\phi_K(x_p)\Big)\Big| \leq 2Q2^{-\beta K}, \quad \text{for all } \mathbf{x} = (x_1, \ldots, x_d)^\top \in [0,1]^d.$$

Moreover, $\|f\|_{L^\infty([0,1]^d)} = \|g\|_{L^\infty(\mathcal{C})}$.

Proof. From the geometric sum formula, $\sum_{q=0}^\infty 3^{-q} = 3/2$. Let ϕ and g be as in Theorem 2 and ϕ_K as defined in the statement of the lemma. Since $\beta \leq 1$, we then have that $|g(x) - g(y)| \leq 2Q|x-y|^{\frac{\beta \log 2}{d \log 3}}$, for all $x, y \in \mathcal{C}$. Moreover, (2.6) shows that $3\sum_{p=1}^d 3^{-p}\phi(x_p)$ and $3\sum_{p=1}^d 3^{-p}\phi_K(x_p)$ are both in the Cantor set $\mathcal{C}$ and have the same first Kd ternary digits. Thus, using (2.4), we find

$$\begin{aligned}
&\Big|f(\mathbf{x}) - g\Big(3\sum_{p=1}^d 3^{-p}\phi_K(x_p)\Big)\Big| \\
&= \Big|g\Big(3\sum_{p=1}^d 3^{-p}\phi(x_p)\Big) - g\Big(3\sum_{p=1}^d 3^{-p}\phi_K(x_p)\Big)\Big| \\
&\leq 2Q\Big|3\sum_{p=1}^d 3^{-p}\big(\phi(x_p) - \phi_K(x_p)\big)\Big|^{\frac{\beta \log 2}{d \log 3}} \\
&\leq 2Q\Big|2\sum_{q=Kd+1}^\infty 3^{-q}\Big|^{\frac{\beta \log 2}{d \log 3}} \\
&\leq 2Q\Big|2 \cdot 3^{-dK-1}\sum_{q=0}^\infty 3^{-q}\Big|^{\frac{\beta \log 2}{d \log 3}} \\
&\leq 2Q3^{-K\frac{\beta \log 2}{\log 3}}.
\end{aligned}$$

Finally, $\|f\|_{L^\infty([0,1]^d)} = \|g\|_{L^\infty(\mathcal{C})}$ follows as an immediate consequence of the function representation (2.4). □

3. Deep ReLU networks and the KA representation

This section studies the construction of deep ReLU networks imitating the KA approximation in Lemma 4. A deep/multilayer feedforward neural network is a function $\mathbf{x} \mapsto f(\mathbf{x})$ that can be represented by an acyclic graph with vertices arranged in a finite number of layers. The first layer is called the input layer, the last layer is the output layer and the layers in between are called hidden layers. We say that a deep network has architecture $(L, (p_0, \ldots, p_{L+1}))$, if the number of hidden layers is L, and p_0, p_j and p_{L+1} are the number of vertices in the input layer, jth hidden layer and output layer, respectively. The input layer of vertices represents the input $\mathbf{x}$. For all other layers, each vertex stands for an operation of the form $\mathbf{y} \mapsto \sigma(\mathbf{a}^\top \mathbf{y} + b)$ with $\mathbf{y}$ the output (viewed as vector) of the previous layer, $\mathbf{a}$ a weight vector, b a

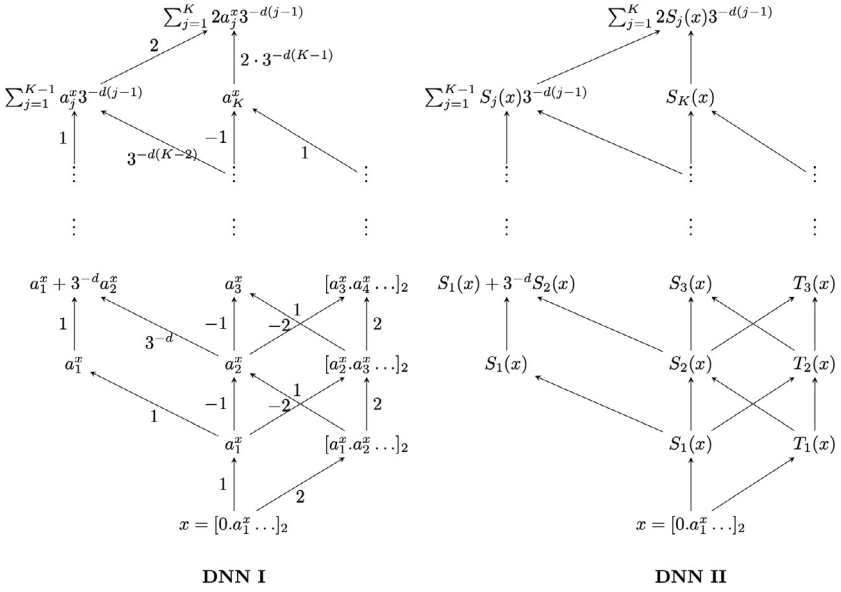

Fig. 1. (Left) A deep neural network with K hidden layers and width three computing the function $x = [0.a_1^x a_2^x \ldots]_2 \mapsto 3\phi_K(x) = \sum_{j=1}^K 2a_j^x 3^{-d(j-1)}$ exactly. In each hidden layer the linear activation function is applied to the left and right unit. The units in the middle use the threshold activation function $\sigma(x) = \mathbf{1}(x \geq 1/2)$. (Right) A deep ReLU network approximating the function ϕ_K. For the definitions of S_r and T_r see the proof of Theorem 3.

shift parameter and σ the activation function. Each vertex has its own set of parameters $(\mathbf{a}, b)$ and also the activation function does not need to be the same for all vertices. If for all vertices in the hidden layers the ReLU activation function $\sigma(x) = \max(x, 0)$ is used and $L > 1$, the network is called a deep ReLU network. As common for regression problems, the activation function in the output layer will be the identity.

Approximation properties of deep neural networks for composed functions are studied in Bauer and Kohler (2019), Fan, Wang, Xie, and Yang (2020), Horowitz and Mammen (2007), Kohler and Krzyżak (2017), Mhaskar and Poggio (2016), Nakada and Imaizumi (2020), Poggio, Mhaskar, Rosasco, Miranda, and Liao (2017) and Schmidt-Hieber (2019, 2020). These approaches do, however, not lead to straightforward constructions of ReLU networks exploiting the specific structure of the KA approximation

$$f(\mathbf{x}) \approx g\Big(3\sum_{p=1}^{d} 3^{-p}\phi_K(x_p)\Big). \tag{3.1}$$

in Lemma 4. To find such a construction, recall that the classical neural network interpretation of the KA representation associates the interior function with the activation function in the first layer (Hecht-Nielsen, 1987). Here, we argue that the interior function can be efficiently approximated by a deep ReLU network. The role of the hidden layers is to retrieve the next bit in the binary representation of the input. Interestingly, some of the proposed network constructions to approximate β-smooth functions use a similar idea without making the link to the KA representation, see Section 4 for more details.

Fig. 1 gives the construction of a network computing $x = [0.a_1^x a_2^x \ldots]_2 \mapsto 3\phi_K(x) = \sum_{j=1}^K 2a_j^x 3^{-d(j-1)}$ combining units with linear activation function $\sigma(x) = x$ and threshold activation function $\sigma(x) = \mathbf{1}(x \geq 1/2)$. The main idea is that for $x = [0.a_1^x a_2^x \ldots]_2$, we can extract the first bit using $a_1^x = \mathbf{1}(x \geq 1/2) = \sigma(x)$ and then define $2x - 2\sigma(x) = 2(x - a_1^x) = [0.a_2^x a_3^x \ldots]_2$. Iterating the procedure allows us to extract a_2^x and consequently any further binary digit of x. The deep neural network DNN I in Fig. 1 has K hidden layers and network width three. The left units in the hidden layer successively build the output value; the units in the middle extract the next bit in the binary representation and the units on the right compute the remainder of the input after bit extraction. To learn the bit extraction algorithm, deep networks lead obviously to much more efficient representations compared to shallow networks.

Constructing d networks computing $\phi_K(x_p)$ for each $x_1, \ldots, x_d$ and combining them yields a network with $K + 1$ hidden layers and network width $3d$, computing the interior function $(x_1, \ldots, x_d) \mapsto 3\sum_{p=1}^d 3^{-p}\phi_K(x_p)$ in (3.1). The overall number of non-zero parameters is of the order Kd. To approximate a β-smooth function f by a neural network via the KA approximation (3.1), the interior step makes the approximating network deep but uses only very few parameters compared to the approximation of the univariate function g.

A close inspection of the network DNN I in Fig. 1 shows that all linear activation functions get non-negative input and can therefore be replaced by the ReLU activation function without changing the outcome. The threshold activation functions $\sigma(x) = \mathbf{1}(x \geq 1/2)$ can be arbitrarily well approximated by the linear combination of two ReLU units via $\varepsilon^{-1}(x - (1-\varepsilon)/2)_+ - \varepsilon^{-1}(x - (1+\varepsilon)/2)_+ \approx \mathbf{1}(x \geq 1/2)$ for $\varepsilon \downarrow 0$. If one accepts potentially huge network parameters, the network DNN I in Fig. 1 can therefore be approximated by a deep ReLU network with K hidden layers and network width four. Consequently, also the construction in (3.1) can be arbitrarily well approximated by deep ReLU networks. It is moreover possible to reduce the size of the network parameters by inserting additional hidden layers in the neural network, see for instance Proposition A.3 in Elbrächter, Perekrestenko, Grohs, and Bölcskei (2019).

Throughout the following we write $\|f\|_p := \|f\|_{L^p([0,1]^d)}$.

Theorem 3. *Let $p \in [1, \infty)$. If there exist $\beta \leq 1$ and a constant Q, such that $|f(\mathbf{x}) - f(\mathbf{y})| \leq Q|\mathbf{x} - \mathbf{y}|_\infty^\beta$, for all $\mathbf{x}, \mathbf{y} \in [0,1]^d$, then, there exists a deep ReLU network $\widetilde{f}$ with $2K + 3$ hidden layers, network architecture $(2K+3, (d, 4d, \ldots, 4d, d, 1, 2^{Kd}+1, 1))$ and all network weights bounded in absolute value by $2(Kd \vee \|f\|_\infty)2^{K(d\vee(p\beta))}$, such that*

$$\big\|f - \widetilde{f}\big\|_p \leq 2\big(Q + \|f\|_\infty\big)2^{-\beta K}.$$

Proof. The proof consists of four parts. In part (A) we construct a ReLU network mimicking the approximand constructed

in Lemma 4. For that we first build a ReLU network with architecture $(2K, (1, 4, \ldots, 4, 1))$ imitating the function $x = [0.a_1^x a_2^x \ldots]_2 \mapsto 3\phi_K(x) = \sum_{j=1}^K 2a_j^x 3^{-d(j-1)}$. In part (B), it is shown that the ReLU network approximation coincides with the function $3\phi_K$ on a subset of $[0,1]^d$ with Lebesgue measure $\geq 1 - 2^{-K\beta p}$. In part (C), we construct a neural network approximation for the outer function g in Lemma 4. The approximation error is controlled in part (D).

(A): Let r be the largest integer such that $2^r \leq 2Kd2^{K\beta p}$ and set $S_1(x) := 2^r(x - 1/2 + 2^{-r-1})_+ - 2^r(x - 1/2 - 2^{-r-1})_+$ and $T_1(x) := 2x$. Given $S_j(x)$, $T_j(x)$, we can then define

$$T_{j+1}(x) := \big(2T_j(x) - 2S_j(x)\big)_+, \quad S_{j+1}(x) := S_1\big(T_j(x) - S_j(x)\big). \tag{3.2}$$

There exists a ReLU network with architecture $(1, (1, 2, 1))$ and all network weights bounded in absolute value by 2^r computing the function $x \mapsto S_1(x)$. Similarly, there exists a ReLU network with architecture $(1, (2, 2, 1))$ computing $(S_j(x), T_j(x)) \mapsto S_{j+1}(x) = S_1(T_j(x) - S_j(x))$. Since $S_1(x) \geq 0$, we have that $(S_j(x))_+ = S_j(x)$ and $T_j(x) = (T_j(x))_+$. Because of that, we can now concatenate these networks as illustrated in Fig. 1 to construct a deep ReLU network computing $x \mapsto \sum_{j=1}^K 2S_j(x)3^{-d(j-1)}$. Recall that computing $S_{j+1}(x)$ from $(S_j(x), T_j(x))$ requires an extra layer with two nodes that is not shown in Fig. 1. Thus, any arrow, except for the ones pointing to the output, adds one additional hidden layer to the ReLU network. The overall number of hidden layers is thus $2K$. Because of the two additional nodes in the non-displayed hidden layers, the width in all hidden layers is four and thus the overall architecture of this deep ReLU network is $(2K, (1, 4, \ldots, 4, 1))$. By checking all edges, it can be seen that all network weights are bounded by $2^r \leq 2Kd2^{K\beta p}$.

(B): Recall that $x = [0.a_1^x a_2^x \ldots]_2$. We now show that on a large subset of the unit interval, it holds that $S_j(x) = a_j^x$ and $T_j(x) = [a_j^x.a_{j+1}^x a_{j+2}^x \ldots]_2$ for all $j = 1, \ldots, K$ and therefore also $\sum_{j=1}^K 2S_j(x)3^{-d(j-1)} = \sum_{j=1}^K 2a_j^x 3^{-d(j-1)} = 3\phi_K(x)$.

We have that $S_1(x) = \mathbf{1}(x > 1/2)$, whenever $|x - 1/2| \geq 2^{-r-1}$. Set $A_{j,r} := \{x : |[0.a_j^x a_{j+1}^x \ldots]_2 - 1/2| \geq 2^{-r-1}\}$. If $x \in A_{j,r}$, then, $S_1([0.a_j^x a_{j+1}^x \ldots]_2) = a_j^x$. Thus, if $S_j(x) = a_j^x$, $T_j(x) = [a_j^x.a_{j+1}^x a_{j+2}^x \ldots]_2$, and $x \in A_{j+1,r}$, then, (3.2) implies $S_{j+1}(x) = a_{j+1}^x$ and $T_{j+1}(x) = [a_{j+1}^x.a_{j+2}^x a_{j+3}^x \ldots]_2$. Hence, the deep ReLU network constructed in part (A) computes the function $[0,1] \ni x \mapsto 3\phi_K(x)$ exactly on the set $\cap_{j=1}^K A_{j,r}$.

For fixed $a_1^x, \ldots, a_{j-1}^x \in \{0,1\}$, the set $\{x : |[0.a_j^x a_{j+1}^x \ldots]_2 - 1/2| < 2^{-r-1}\}$ is an interval of length 2^{-r-j+1}. As there are 2^{j-1} possibilities to choose $a_1^x, \ldots, a_{j-1}^x \in \{0,1\}$, the complement $A_{j,r}^c = \{x \in [0,1] : x \notin A_{j,r}\}$ can be written as the union of 2^{j-1} subintervals of length 2^{-r-j+1}. The Lebesgue measure of $A_{j,r}^c$ is therefore bounded by 2^{-r}. Since $Kd2^{K\beta p} \leq 2^r$, we find that $(\cap_{j=1}^K A_{j,r})^c$ has Lebesgue measure bounded by $K2^{-r} \leq 2^{-K\beta p}/d$. This completes the proof for part (B).

(C): Now we construct a shallow ReLU network interpolating the outer function g in Lemma 4 at the $2^{Kd}+1$ points $\{\sum_{j=1}^{Kd} 2t_j 3^{-j} : (t_1, \ldots, t_{Kd}) \in \{0,1\}^{Kd}\} \cup \{1\}$. Denote these points by $0 =: s_0 < s_1 < \cdots < s_{2^{Kd}-1} < s_{2^{Kd}} := 1$. For any $x \in [0,1]$,

$$\begin{aligned}\widetilde{g}(x) &:= g(s_0) + \sum_{j=1}^{2^{Kd}} \frac{g(s_j) - g(s_{j-1})}{s_j - s_{j-1}}\big((x - s_{j-1})_+ - (x - s_j)_+\big) \\ &= g(s_0)(x+1)_+ + \Big(\frac{g(s_1) - g(s_0)}{s_1 - s_0} - g(s_0)\Big)(x)_+ \\ &\quad + \sum_{j=1}^{2^{Kd}-1} \Big(\frac{g(s_{j+1}) - g(s_j)}{s_{j+1} - s_j} - \frac{g(s_j) - g(s_{j-1})}{s_j - s_{j-1}}\Big)(x - s_j)_+.\end{aligned}$$

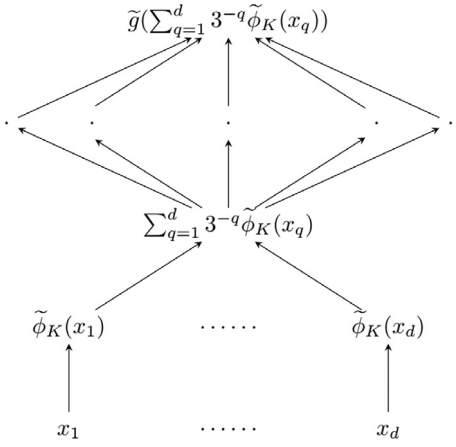

Fig. 2. Construction of the deep ReLU network in part (D) of the proof for Theorem 3.

The function $\widetilde{g}(x)$ can therefore be represented on $[0,1]$ by a shallow ReLU network with $2^{Kd}+1$ units in the hidden layer. Moreover, $\widetilde{g}(s_j) = g(s_j)$ for all $j = 0, \ldots, 2^{Kd}$. Finally, we bound the size of the network weights. We have $s_{j+1} - s_j \geq 3^{-Kd}$. By Lemma 4, $\|f\|_\infty = \|g\|_{L^\infty(\mathcal{C})}$. Since $0 \leq s_j \leq 1$ and for any positive a, $a(x - s_j)_+ = \sqrt{a}(\sqrt{a}x - \sqrt{a}s_j)_+$, we conclude that all network weights can be chosen to be smaller than $2\|f\|_\infty 2^{Kd}$.

(D): Fig. 2 shows how the neural networks $\widetilde{\phi}_K$ and $\widetilde{g}$ can be combined into a deep ReLU network with architecture $(2K + 3, (d, 4d, \ldots, 4d, d, 1, 2^{Kd} + 1, 1))$ and all network weights bounded in absolute value by $\max(2\|f\|_\infty 2^{Kd}, 2Kd2^{K\beta p})$ computing the function $\widetilde{f}(x_1, \ldots, x_d) := \widetilde{g}(3\sum_{q=1}^d 3^{-q}\widetilde{\phi}_K(x_q))$. Since $3\sum_{q=1}^d 3^{-q}\phi_K(x_q) \in \{s_0, \ldots, s_{2^{Kd}}\}$, the interpolation property $\widetilde{g}(s_j) = g(s_j)$ implies that $\widetilde{g}(3\sum_{q=1}^d 3^{-q}\phi_K(x_q)) = g(3\sum_{q=1}^d 3^{-q}\phi_K(x_q))$. Together with (B), we conclude that

$$\begin{aligned}\widetilde{f}(x_1, \ldots, x_d) &= \widetilde{g}\Big(3\sum_{q=1}^d 3^{-q}\widetilde{\phi}_K(x_q)\Big) \\ &= g\Big(3\sum_{q=1}^d 3^{-q}\phi_K(x_q)\Big), \text{ if } x_1, \ldots, x_d \in \bigcap_{j=1}^K A_{j,r}.\end{aligned}$$

As shown in Lemma 4, $\|f\|_\infty = \|g\|_{L^\infty(\mathcal{C})}$. Since $\widetilde{g}$ is a piecewise linear interpolation of g, we also have $\|\widetilde{g}\|_{L^\infty([0,1])} \leq \|f\|_\infty$. As shown in (B), the Lebesgue measure of $(\cap_{j=1}^K A_{j,r})^c$ is bounded by $2^{-K\beta p}/d$. Decomposing the integral and using the approximation bound in Lemma 4,

$$\begin{aligned}\|f - \widetilde{f}\|_p^p &\leq \int_{\forall i: x_i \in \cap_{j=1}^K A_{j,r}} \Big|f(\mathbf{x}) - g\Big(3\sum_{q=1}^d 3^{-q}\phi_K(x_p)\Big)\Big|^p d\mathbf{x} \\ &\quad + \int_{\exists i: x_i \notin \cap_{j=1}^K A_{j,r}} 2^p\|f\|_\infty^p \, d\mathbf{x} \\ &\leq 2^p Q^p 2^{-\beta K p} + 2^p\|f\|_\infty^p 2^{-K\beta p} \\ &\leq 2^p\big(Q + \|f\|_\infty\big)^p 2^{-K\beta p},\end{aligned}$$

using for the last inequality that $a^p + b^p \leq (a+b)^p$ for all $p \geq 1$ and all $a, b \geq 0$. □

Recall that for a function class with m^d parameters, the expected optimal approximation rate for a β-smooth function in d dimensions is $m^{-\beta}$. The previous theorem leads to the rate $2^{-K\beta}$ using of the order of 2^{Kd} network parameters. This coincides thus with the expected rate. In contrast to several other constructions, no network sparsity is required to recover the rate. It is unclear

whether the construction can be generalized to higher order smoothness or anisotropic smoothness.

The function approximation in Lemma 4 is quite similar to tree-based methods in statistical learning. CART or MARS, for instance, selects a partition of the input space by making successive splits along different directions and then fits a piecewise constant (or piecewise linear) function on the selected partition (Hastie, Tibshirani, & Friedman, 2009, Section 9.2). The KA approximation is also piecewise constant and the interior function assigns a unique value to each set in the dyadic partition. Enlarging K refines the partition. The deep ReLU network constructed in the proof of Theorem 3 imitates the KA approximation and also relies on a dyadic partition of the input space. By changing the network parameters in the first layers, the unit cube $[0,1]^d$ can be split into more general subsets and similar function systems as the ones underlying MARS or CART can be generated using deep ReLU networks, see also Eckle and Schmidt-Hieber (2019) and Kohler, Krzyzak, and Langer (2019).

As typical for neural network constructions that decompose function approximation into a localization and a local approximation step, the deep ReLU network in Theorem 3 only depends on the represented function f via the weights in the last hidden layer. As a consequence, one could use this deep ReLU network construction to initialize stochastic gradient descent. For that, it is natural to sample the weights in the output layer from a given distribution and assign all other network parameters to the corresponding value in the network construction. A comparison with standard network initializations will be addressed in future work.

The fact that in the proposed network construction only the output layer depends on the represented function matches also with the observation that in deep learning a considerable amount of information about the represented function is decoded in the last layer. This is exploited in pre-training where a trained deep network from a different classification problem is taken and only the output layer is learned by the new dataset, see for instance Zeiler and Fergus (2014). The fact that pre-training works show that deep networks build rather generic function systems in the first layers. For real datasets, the learned parameters in the first hidden layers still exhibit some dependence on the underlying problem and transfer learning updating all weights based on the new data outperforms pre-training (He, Girshick, & Dollar, 2019).

4. Related literature

The section is intended to provide a brief overview of related approaches.

Shen et al. (2020) propose a similar deep ReLU network construction without making a link to the KA representation or space-filling curves. The similarity between both approaches can be best seen in their Fig. 5 or the outline of the proof for Theorem 2.1 in Section 3.2. Indeed, in a first step, the input space is partitioned into smaller hypercubes that are enumerated. The first hidden layers map the input to the index of the hypercube. This localization step is closely related to the action of the interior function used here. The last hidden layers of the deep ReLU network perform a piecewise linear approximation and this is essentially the same as the implementation of the outer function in the modified KA representation in this paper. To ensure good smoothness properties, Shen et al. (2020) also include gaps in the indexing, that fulfill a similar role as the gaps in the Cantor set here. Lu, Shen, Yang, and Zhang (2020) combine the approach with local Taylor expansions and achieves optimal approximation rates for functions that are smoother than Lipschitz.

Another direction is to search for activation function with good representation property based on the modified KA representation in Maiorov and Pinkus (1999). Their Theorem 4 states that one can find a real analytic, strictly increasing, and sigmoidal activation function σ, such that for any continuous function $f : [0,1]^d \to \mathbb{R}$ and any $\varepsilon > 0$, there exist parameters $\mathbf{w}_{pq} \in \mathbb{R}^d$, $a_{pq}, b_{pq}, c_q, d_q \in \mathbb{R}$, satisfying

$$\sup_{\mathbf{x}\in[0,1]^d} \left| f(\mathbf{x}) - \sum_{q=1}^{6d+3} d_q\, \sigma\Big(\sum_{p=1}^{3d} b_{pq}\sigma(\mathbf{w}_{pq}^\top \mathbf{x} + a_{pq}) + c_q\Big)\right| < \varepsilon.$$

This removes the dependence of the outer activation function in the KA representation on the represented function f. The main issue is that despite its smoothness properties, the activation function σ is not computable and it is unclear how to transfer the result to popular activation functions such as the ReLU. One step in this direction has been done in the recent work Guliyev and Ismailov (2018) proving that one can design computable activation functions with complexity increasing as $\varepsilon \downarrow 0$. Shen, Yang, and Zhang (2020) show that for a neural network with three hidden layers and three explicit and relatively simple but non-differentiable activation functions one can achieve extremely fast approximation rates.

The fact that deep networks can do bit encoding and decoding efficiently has been used previously in Bartlett, Harvey, Liaw, and Mehrabian (2019) to prove (nearly) sharp bounds for the VC dimension of deep ReLU networks and also in Yarotsky (2018) and Yarotsky and Zhevnerchuk (2019) for a different construction to obtain approximation rates of very deep networks with fixed with. There are, however, several distinctive differences. These works employ bit encoding to compress several function values in one number (see for instance Section 5.2.1 in Yarotsky, 2018), while we apply bit extraction to the input vector. In our approach the bit extraction leads to a localization of the input space and the function values only enters in the last hidden layer. It can be checked that in our construction the weight assignment is continuous, that is, small changes in the represented function will lead to small changes in the network weights. On the contrary, bit encoding in the function values results in discontinuous weight assignment and it is known that this is unavoidable for efficient function approximation based on very deep ReLU networks (Yarotsky, 2018).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Bader, M. (2013). *Texts in computational science and engineering*: *Vol. 9, Space-filling curves* (p. xiv+278). Heidelberg: Springer.

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

Bauer, B., & Kohler, M. (2019). On deep learning as a remedy for the curse of dimensionality in nonparametric regression. *The Annals of Statistics*, *47*(4), 2261–2285.

Braun, J. (2009). *An application of Kolmogorov's superposition theorem to function reconstruction in higher dimensions* (Ph.D. thesis), Universität Bonn.

Braun, J., & Griebel, M. (2009). On a constructive proof of Kolmogorov's superposition theorem. *Constructive Approximation*, *30*(3), 653–675.

Eckle, K., & Schmidt-Hieber, J. (2019). A comparison of deep networks with ReLU activation function and linear spline-type methods. *Neural Networks*, *110*, 232–242.

Elbrächter, D., Perekrestenko, D., Grohs, P., & Bölcskei, H. (2019). Deep neural network approximation theory. arXiv e-prints, arXiv:1901.02220.

Fan, J., Wang, Z., Xie, Y., & Yang, Z. (2020). A theoretical analysis of deep Q-learning. In A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, & M. Zeilinger (Eds.), *Proceedings of machine learning research*: *Vol. 120*, *Proceedings of the 2nd conference on learning for dynamics and control* (pp. 486–489). PMLR.

Girosi, F., & Poggio, T. (1989). Representation properties of networks: Kolmogorov's theorem is irrelevant. *Neural Computation*, *1*(4), 465–469.

Gorchakov, A., & Mozolenko, V. (2019). Analysis of approaches to the universal approximation of a continuous function using Kolmogorov's superposition. In *2019 international conference on engineering and telecommunication* (pp. 1–4). http://dx.doi.org/10.1109/EnT47717.2019.9030591.

Gordon, Y., Maiorov, V., Meyer, M., & Reisner, S. (2002). On the best approximation by ridge functions in the uniform norm. *Constructive Approximation*, *18*(1), 61–85.

Guliyev, N. J., & Ismailov, V. E. (2018). Approximation capability of two hidden layer feedforward neural networks with fixed weights. *Neurocomputing*, *316*, 262–269.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *Springer series in statistics*, *The elements of statistical learning* (2nd ed.). (p. xxii+745). New York: Springer.

He, K., Girshick, R., & Dollar, P. (2019). Rethinking ImageNet pre-training, In *The IEEE international conference on computer vision*.

Hecht-Nielsen, R. (1987). Kolmogorov's mapping neural network existence theorem. *Vol. III*, In *Proceedings of the IEEE first international conference on neural networks* (pp. 11–13). Piscataway, NJ: IEEE.

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

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

Kohler, M., Krzyzak, A., & Langer, S. (2019). Estimation of a function of low local dimensionality by deep neural networks. arXiv e-prints, arXiv:1908.11140.

Kolmogorov, A. N. (1957). On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Doklady Akademii Nauk SSSR*, *114*, 953–956.

Kupers, A. On space-filling curves and the Hahn-Mazurkiewicz theorem. Unpublished manuscript.

Kurkova, V. (1991). Kolmogorov's theorem is relevant. *Neural Computation*, *3*(4), 617–622.

Kurkova, V. (1992). Kolmogorov's theorem and multilayer neural networks. *Neural Networks*, *5*(3), 501–506.

Lu, J., Shen, Z., Yang, H., & Zhang, S. (2020). Deep network approximation for smooth functions. arXiv e-prints, arXiv:2001.03040.

Maiorov, V. E. (1999). On best approximation by ridge functions. *Journal of Approximation Theory*, *99*(1), 68–94.

Maiorov, V., Meir, R., & Ratsaby, J. (1999). On the approximation of functional classes equipped with a uniform measure using ridge functions. *Journal of Approximation Theory*, *99*(1), 95–111.

Maiorov, V., & Pinkus, A. (1999). Lower bounds for approximation by MLP neural networks. *Neurocomputing*, *25*(1), 81–91.

Mhaskar, H. N., & Poggio, T. (2016). Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, *14*(06), 829–848.

Montanelli, H., & Yang, H. (2020). Error bounds for deep ReLU networks using the Kolmogorov–Arnold superposition theorem. *Neural Networks*, *129*, 1–6.

Nakada, R., & Imaizumi, M. (2020). Adaptive approximation and generalization of deep neural network with intrinsic dimensionality. *Journal of Machine Learning Research*, *21*(174), 1–38.

Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., & Liao, Q. (2017). Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *International Journal of Automation and Computing*, *14*(5), 503–519.

Schmidt-Hieber, J. (2019). Deep relu network approximation of functions on a manifold. arXiv e-prints, arXiv:1908.00695.

Schmidt-Hieber, J. (2020). Nonparametric regression using deep neural networks with ReLU activation function. *The Annals of Statistics*, *48*(4), 1875–1897.

Shen, Z., Yang, H., & Zhang, S. (2020). Deep network approximation characterized by number of neurons. *Communications in Computational Physics*, *28*(5), 1768–1811.

Shen, Z., Yang, H., & Zhang, S. (2020). Neural network approximation: Three hidden layers are enough. arXiv e-prints, arXiv:2010.14075.

Siegelmann, H. T., & Sontag, E. D. (1994). Analog computation via neural networks. *Theoretical Computer Science*, *131*(2), 331–360.

Sprecher, D. A. (1965). On the structure of continuous functions of several variables. *Transactions of the American Mathematical Society*, *115*, 340–355.

Sprecher, D. A. (1996). A numerical implementation of Kolmogorov's superpositions. *Neural Networks*, *9*(5), 765–772.

Sprecher, D. A. (1997). A numerical implementation of Kolmogorov's superpositions II. *Neural Networks*, *10*(3), 447–457.

Sprecher, D. A., & Draghici, S. (2002). Space-filling curves and Kolmogorov superposition-based neural networks. *Neural Networks*, *15*(1), 57–67.

Yarotsky, D. (2018). Optimal approximation of continuous functions by very deep ReLU networks. In S. Bubeck, V. Perchet, & P. Rigollet (Eds.), *Proceedings of machine learning research*: *Vol. 75*, *Proceedings of the 31st conference on learning theory* (pp. 639–649).

Yarotsky, D., & Zhevnerchuk, A. (2019). The phase diagram of approximation rates for deep neural networks. arXiv e-prints, arXiv:1906.09477.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer vision* (pp. 818–833). Cham: Springer International Publishing.