International Journal of Automation and Computing 14(5), October 2017, 503-519
DOI: 10.1007/s11633-017-1054-2

Why and When Can Deep-but Not Shallow-networks Avoid the Curse of Dimensionality: A Review

Tomaso Poggio[1] Hrushikesh Mhaskar[2, 3] Lorenzo Rosasco[1] Brando Miranda[1] Qianli Liao[1]

[1]Center for Brains, Minds, and Machines, McGovern Institute for Brain Research,
Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[2]Department of Mathematics, California Institute of Technology, Pasadena, CA 91125, USA
[3]Institute of Mathematical Sciences, Claremont Graduate University, Claremont, CA 91711, USA

Abstract: The paper reviews and extends an emerging body of theoretical results on deep learning including the conditions under which it can be exponentially better than shallow learning. A class of deep convolutional networks represent an important special case of these conditions, though weight sharing is not the main reason for their exponential advantage. Implications of a few key theorems are discussed, together with new results, open problems and conjectures.

Keywords: Machine learning, neural networks, deep and shallow networks, convolutional neural networks, function approximation, deep learning.

1 A theory of deep learning

1.1 Introduction

There are at three main sets of theory questions about deep neural networks. The first set of questions is about the power of the architecture – Which classes of functions can it approximate and learn well? The second set of questions is about the learning process: Why is stochastic gradient descent (SGD) so unreasonably efficient, at least in appearance? The third, more important question is about generalization. Overparametrization may explain why minima are easy to find during training but then why does overfitting seems to be less of a problem than for classical shallow networks? Is this because deep networks are very efficient algorithms for hierarchical vector quantization?

In this paper, we focus especially on the first set of questions, summarizing several theorems that have appeared online in 2015[1−3] and 2016[4, 5]. We then describe additional results as well as a few conjectures and open questions. The main message is that deep networks have the theoretical guarantee, which shallow networks do not have, that they can avoid the curse of dimensionality for an important class of problems, corresponding to compositional functions, i.e., functions of functions. An especially interesting subset of such compositional functions are hierarchically local compositional functions where all the constituent functions are local in the sense of bounded small dimensionality. The deep networks that can approximate them without the curse of dimensionality are of the deep convolutional type (though weight sharing is not necessary).

Implications of the theorems likely to be relevant in practice are:

1) Certain deep convolutional architectures have a theoretical guarantee that they can be much better than one layer architectures such as kernel machines.

2) The problems for which certain deep networks are guaranteed to avoid the curse of dimensionality (see for a nice review[6]) correspond to input-output mappings that are compositional. The most interesting set of problems consists of compositional functions composed of a hierarchy of constituent functions that are local: An example is $f(x_1, \cdots, x_8) = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8)))$. The compositional function f requires only "local" computations (here with just dimension 2) in each of its constituent functions h.

3) The key aspect of convolutional networks that can give them an exponential advantage is not weight sharing but locality at each level of the hierarchy.

2 Previous theoretical work

Deep learning references start with Hinton′s backpropagation and with Lecun′s convolutional networks (see for a nice review[7]). Of course, multilayer convolutional networks have been around at least as far back as the optical processing era of the 1970s. The Neocognitron[8] was a convolutional neural network that was trained to recognize characters. The property of compositionality was a main motivation for hierarchical models of visual cortex such as HMAX which can be regarded as a pyramid of AND and OR layers[9], that is a sequence of conjunctions and disjunctions. Several papers in the 1980s focused on the approximation power and learning properties of

Review
Special Issue on Human-inspired Computing
Manuscript received November 3, 2016; accepted December 12, 2017; published online March 14, 2017
This work was supported by the Center for Brains, Minds and Machines (CBMM), NSF STC award CCF (No. 1231216), and ARO (No. W911NF-15-1-0385).
Recommended by Associate Editor Hong Qiao
© The Author(s) 2017

one-hidden layer networks (called shallow networks here). Very little appeared on multilayer networks[10−12], mainly because one hidden layer nets performed empirically as well as deeper networks. On the theory side, a review by Pinkus in 1999[13] concludes that "$\cdots$ there seems to be reason to conjecture that the two hidden layer model may be significantly more promising than the single hidden layer model$\cdots$". A version of the questions about the importance of hierarchies was asked in [14] as follows: "A comparison with real brains offers another, and probably related, challenge to learning theory. The "learning algorithms" we have described in this paper correspond to one-layer architectures. Are hierarchical architectures with more layers justifiable in terms of learning theory? It seems that the learning theory of the type we have outlined does not offer any general argument in favor of hierarchical learning machines for regression or classification. This is somewhat of a puzzle since the organization of cortex – for instance visual cortex – is strongly hierarchical. At the same time, hierarchical learning systems show superior performance in several engineering applications." Because of the great empirical success of deep learning over the last three years, several papers addressing the question of why hierarchies have appeared. Sum-Product networks, which are equivalent to polynomial networks (see [15, 16]), are a simple case of a hierarchy that was analyzed[17] but did not provide particularly useful insights. Montufar et al.[18] showed that the number of linear regions that can be synthesized by a deep network with rectified linear unit (ReLU) nonlinearities is much larger than by a shallow network. However, they did not study the conditions under which this property yields better learning performance. In fact, we will show later that the power of a deep network cannot be exploited in general but for certain specific classes of functions. Relevant to the present review is the work on hierarchical quadratic networks[16], together with function approximation results[13,19]. Also relevant is the conjecture by Cohen et al.[20] on a connection between deep learning networks and the hierarchical Tucker representations of tensors. In fact, our theorems describe formally the class of functions for which the conjecture holds. This paper describes and extends results presented in [4, 21–24] which derive new upper bounds for the approximation by deep networks of certain important classes of functions which avoid the curse of dimensionality. The upper bound for the approximation by shallow networks of general functions was well known to be exponential. It seems natural to assume that, since there is no general way for shallow networks to exploit a compositional prior, lower bounds for the approximation by shallow networks of compositional functions should also be exponential. In fact, examples of specific functions that cannot be represented efficiently by shallow networks have been given very recently by [25, 26]. We provide in Theorem 5 another example of a class of compositional functions for which there is a gap between shallow and deep networks.

3 Function approximation by deep networks

In this section, we state theorems about the approximation properties of shallow and deep networks.

3.1 Degree of approximation

The general paradigm is as follows. We are interested in determining how complex a network ought to be to theoretically guarantee approximation of an unknown target function f up to a given accuracy $\epsilon > 0$. To measure the accuracy, we need a norm $\|\cdot\|$ on some normed linear space $\mathbb{X}$. As we will see the norm used in the results of this paper is the sup norm in keeping with the standard choice in approximation theory. Notice, however, that from the point of view of machine learning, the relevant norm is the L_2 norm. In this sense, several of our results are stronger than needed. On the other hand, our main results on compositionality require the sup norm in order to be independent from the unknown distribution of the input data. This is important for machine learning.

Let V_N be the set of all networks of a given kind with complexity N which we take here to be the total number of units in the network (e.g., all shallow networks with N units in the hidden layer). It is assumed that the class of networks with a higher complexity include those with a lower complexity; i.e., $V_N \subseteq V_{N+1}$. The degree of approximation is defined by

$$\mathrm{dist}(f, V_N) = \inf_{P \in V_N} \|f - P\|. \tag{1}$$

For example, if $\mathrm{dist}(f, V_N) = \mathcal{O}(N^{-\gamma})$ for some $\gamma > 0$, then a network with complexity $N = \mathcal{O}(\epsilon^{-\frac{1}{\gamma}})$ will be sufficient to guarantee an approximation with accuracy at least ϵ. Since f is unknown, in order to obtain theoretically proved upper bounds, we need to make some assumptions on the class of functions from which the unknown target function is chosen. This apriori information is codified by the statement that $f \in W$ for some subspace $W \subseteq \mathbb{X}$. This subspace is usually a smoothness class characterized by a smoothness parameter m. Here, it will be generalized to a smoothness and compositional class, characterized by the parameters m and d ($d = 2$ in the example of Fig. 1, is in general the size of the kernel in a convolutional network).

3.2 Shallow and deep networks

This section characterizes conditions under which deep networks are "better" than shallow network in approximating functions. Thus we compare shallow (one-hidden layer) networks with deep networks as shown in Fig. 1. Both types of networks use the same small set of operations – dot products, linear combinations, a fixed nonlinear function of one variable, possibly convolution and pooling. Each node in the networks we consider usually corresponds to a node in the graph of the function to be approximated, as shown in Fig. 1. In particular each node in the network contains a certain number of units. A unit is a neuron which computes

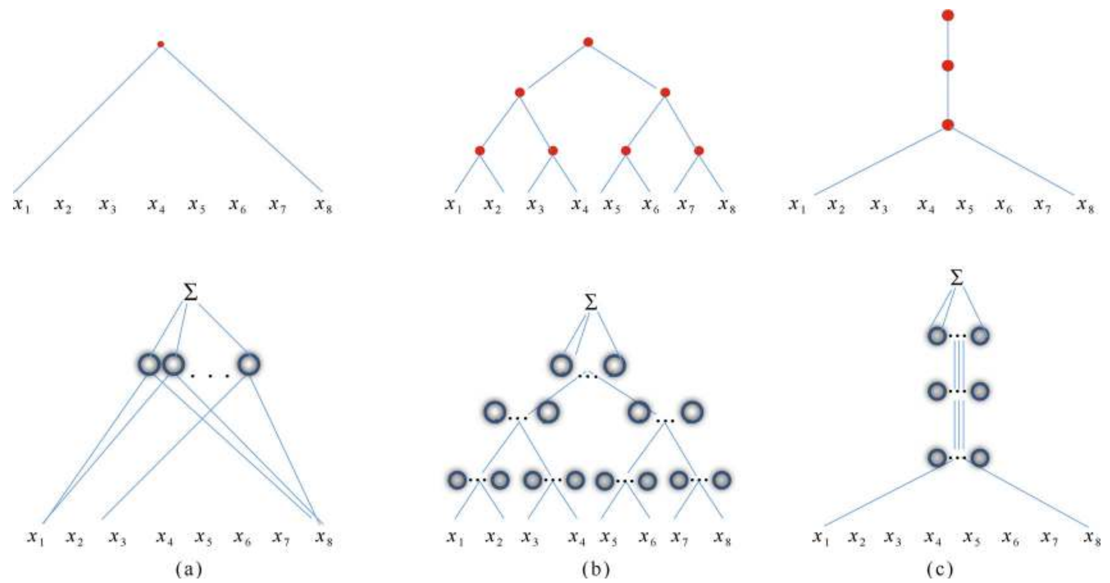

Fig. 1 The top graphs are associated to functions, each of the bottom diagrams depicts the network approximating the function above. (a) shows a shallow universal network in 8 variables and N units approximates a generic function of 8 variables $f(x_1, \cdots, x_8)$; (b) shows a binary tree hierarchical network at the bottom in $n = 8$ variables, which approximates well functions of the form $f(x_1, \cdots, x_8) = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8)))$ as represented by the binary graph above. In the approximating network each of the $n - 1$ nodes in the graph of the function corresponds to a set of $Q = \frac{N}{n-1}$ ReLU units computing the ridge function $\sum_{i=1}^{Q} a_i(\langle \boldsymbol{v}_i \boldsymbol{x}\rangle + t_i)_+$, with $\boldsymbol{v}_i, \boldsymbol{x} \in \mathbf{R}^2$, $a_i, t_i \in \mathbf{R}$. Each term in the ridge function corresponds to a unit in the node (this is somewhat different from todays deep networks, see text and note in Section 7). In a binary tree with n inputs, there are $\log_2 n$ levels and a total of $n - 1$ nodes. Similar to the shallow network, a hierarchical network is universal, i.e., it can approximate any continuous function, the text proves that it can approximate a compositional functions exponentially better than a shallow network. No invariance – that is weight sharing – is assumed here. Notice that the key property that makes convolutional deep nets exponentially better than shallow for compositional functions is the locality of the constituent functions – that is their low dimensionality. Weight sharing corresponds to all constituent functions at one level to be the same ($h_{11} = h_{12}$, etc.); (c) shows a different mechanism that can be exploited by the deep network at the bottom to reduce the curse of dimensionality in the compositional function at the top: leveraging different degrees of smoothness of the constituent functions, see Theorem 6 in the text. Notice that in (c) the input dimensionality must be ≥ 2 in order for deep nets to have an advantage over shallow nets. The simplest examples of functions to be considered for (a), (b) and (c) are functions that are polynomials with a structure corresponding to the graph at the top.

$$(\langle x, w\rangle + b)_+ \tag{2}$$

where w is the vector of weights on the vector input x. Both t and the real number b are parameters tuned by learning. We assume here that each node in the networks computes the linear combination of r such units

$$\sum_{i=1}^{r} c_i(\langle x, t_i\rangle + b_i)_+. \tag{3}$$

Notice that for our main example of a deep network corresponding to a binary tree graph, the resulting architecture is an idealized version of the plethora of deep convolutional neural networks described in the literature. In particular, it has only one output at the top unlike most of the deep architectures with many channels and many top-level outputs. Correspondingly, each node computes a single value instead of multiple channels, using the combination of several units (see (3)). Our approach and basic results apply rather directly to more complex networks (see the third note in Section 7). A careful analysis and comparison with simulations will be described in the future work.

The logic of our theorems is as follows:

1) Both shallow (a) and deep (b) networks are universal, i.e., they can approximate arbitrarily well any continuous function of n variables on a compact domain. The result for shallow networks is classical. Since shallow networks can be viewed as a special case of deep networks, it is clear that for any continuous function of n variables, there exists also a deep network that approximates the function arbitrarily well on a compact domain.

2) We consider a special class of functions of n variables on a compact domain that is a hierarchical composition of local functions such as

$$f(x_1, \cdots, x_8) = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8))). \tag{4}$$

The structure of the function in (4) is represented by a graph of the binary tree type. This is the simplest example of compositional functions, reflecting dimensionality $d = 2$ for the constituent functions h. In general, d is arbitrary but fixed and independent of the dimensionality n of the compositional function f. In our results, we will often think of

n increasing while d is fixed. In Section 4, we will consider the more general compositional case.

3) The approximation of functions with a compositional structure can be achieved with the same degree of accuracy by deep and shallow networks but that the number of parameters are much smaller for the deep networks than for the shallow network with equivalent approximation accuracy. It is intuitive that a hierarchical network matching the structure of a compositional function should be "better" at approximating it than a generic shallow network but universality of shallow networks asks for non-obvious characterization of "better". Our result makes clear that the intuition is indeed correct.

In the perspective of machine learning, we assume that the shallow networks do not have any structural information on the function to be learned (here its compositional structure), because they cannot represent it directly and cannot exploit the advantage of a smaller number of parameters. In any case, in the context of approximation theory, we will exhibit and cite lower bounds of approximation by shallow networks for the class of compositional functions. Deep networks with standard architectures on the other hand do represent compositionality in their architecture and can be adapted to the details of such prior information.

We approximate functions of n variables of the form of (4) With networks in which the activation nonlinearity is a smoothed version of the so called ReLU, originally called ramp by Breiman and given by $\sigma(x) = |x|_+ = \max(0, x)$. The architecture of the deep networks reflects (4) with each node h_i being a ridge function, comprising one or more neurons.

Let $I^n = [-1, 1]^n$, $\mathbb{X} = C(I^n)$ be the space of all continuous functions on I^n, with $\|f\| = \max_{x \in I^n} |f(x)|$. Let $\mathcal{S}_{N,n}$ denote the class of all shallow networks with N units of the form

$$x \mapsto \sum_{k=1}^{N} a_k \sigma(\langle w_k, x\rangle + b_k)$$

where $w_k \in \mathbf{R}^n$, $b_k, a_k \in \mathbf{R}$. The number of trainable parameters here is $(n + 2)N$. Let $m \geq 1$ be an integer, and W_m^n be the set of all functions of n variables with continuous partial derivatives of orders up to $m < \infty$ such that $\|f\| + \sum_{1 \leq |k|_1 \leq m} \|\mathcal{D}^k f\| \leq 1$, where $\mathcal{D}^k$ denotes the partial derivative indicated by the multi-integer $k \geq 1$, and $|k|_1$ is the sum of the components of k.

For the hierarchical binary tree network, the analogous spaces are defined by considering the compact set $W_m^{n,2}$ to be the class of all compositional functions f of n variables with a binary tree architecture and constituent functions h in W_m^2. We define the corresponding class of deep networks $\mathcal{D}_{N,2}$ to be the set of all deep networks with a binary tree architecture, where each of the constituent nodes is in $\mathcal{S}_{M,2}$, where $N = |V|M$, V is the set of non–leaf vertices of the tree. We note that in the case when n is an integer power of 2, the total number of parameters involved in a deep network in $\mathcal{D}_{N,2}$, i.e., weights and biases, is $4N$.

Two observations are critical to understand the meaning of our results:

1) Compositional functions of n variables are a subset of functions of n variables, i.e., $W_m^n \supseteq W_m^{n,2}$. Deep networks can exploit in their architecture the special structure of compositional functions, whereas shallow networks are blind to it. Thus, from the point of view of shallow networks, functions in $W_m^{n,2}$ are just functions in W_m^n, this is not the case for deep networks.

2) The deep network does not need to have exactly the same compositional architecture as the compositional function to be approximated. It is sufficient that the acyclic graph representing the structure of the function is a subgraph of the graph representing the structure of the deep network. The degree of approximation estimates depend on the graph associated with the network and are thus an upper bound on what could be achieved by a network exactly matched to the function architecture.

Theorems 1 and 2 estimate the degree of approximation for shallow and deep networks.

3.3 Shallow networks

Theorem 1 is about shallow networks.

Theorem 1. Let $\sigma : \mathbf{R} \to \mathbf{R}$ be infinitely differentiable, and not a polynomial. For $f \in W_m^n$, the complexity of shallow networks that provide accuracy at least ϵ is

$$N = \mathcal{O}(\epsilon^{-\frac{n}{m}}) \text{ and is the best possible.} \tag{5}$$

In Theorem 2.1 of [27], the theorem is stated under the condition that σ is infinitely differentiable, and there exists $b \in \mathbf{R}$ such that $\sigma^{(k)}(b) \neq 0$ for any integer $k \geq 0$. It is proved in [28] that the second condition is equivalent to σ not being a polynomial. The proof in [27] relies on the fact that under these conditions on σ, the algebraic polynomials in n variables of (total or coordinatewise) degree $< q$ are in the uniform closure of the span of $\mathcal{O}(q^n)$ functions of the form $\boldsymbol{x} \mapsto \sigma(\langle w, x\rangle + b)$ (see the Appendix of [29], Section "Neural networks: polynomial viewpoint"). The estimate itself is an upper bound on the degree of approximation by such polynomials. Since it is based on the approximation of the polynomial space contained in the ridge functions implemented by shallow networks, one may ask whether it could be improved by using a different approach. The answer relies on the concept of nonlinear n-width of the compact set W_m^n[4, 30]. The n-width results imply that the estimate in Theorem 1 is the best possible among all reasonable[30] methods of approximating arbitrary functions in W_m^n. The estimate of Theorem 1 is the best possible if the only a priori information we are allowed to assume is that the target function belongs to $f \in W_m^n$. The exponential dependence on the dimension n of the number $\epsilon^{-\frac{n}{m}}$ of parameters needed to obtain an accuracy $\mathcal{O}(\epsilon)$ is known as the curse of dimensionality. Note that the constants involved in $\mathcal{O}$ in the theorems will depend upon the norms of the derivatives of f as well as σ.

A simple but useful corollary follows from the proof of Theorem 1 about polynomials (which are a smaller space than spaces of Sobolev functions). Let us denote with P_k^n

the linear space of polynomials of degree at most k in n variables.

Corollary 1. Let $\sigma : \mathbf{R} \to \mathbf{R}$ be infinitely differentiable, and not a polynomial. Every $f \in P_k^n$ can be realized with an arbitrary accuracy by shallow network with r units, $r = \binom{n+k}{k} \approx k^n$.

3.4 Deep hierarchically local networks

Theorem 2 is about deep networks with smooth activations and is recent (preliminary versions appeared in [2–4]). We formulate it in the binary tree case for simplicity but it extends immediately to functions that are compositions of constituent functions of a fixed number of variables d instead than of $d = 2$ variables as in the statement of the theorem (in convolutional networks d corresponds to the size of the kernel).

Theorem 2. For $f \in W_m^{n,2}$, consider a deep network with the same compositional architecture and with an activation function $\sigma : \mathbf{R} \to \mathbf{R}$ which is infinitely differentiable, and not a polynomial. The complexity of the network to provide approximation with accuracy at least ϵ is

$$N = \mathcal{O}((n-1)\epsilon^{-\frac{2}{m}}). \tag{6}$$

Proof. To prove Theorem 2, we observe that each of the constituent functions being in W_m^2, (1) applied with $n = 2$ implies that each of these functions can be approximated from $\mathcal{S}_{N,2}$ up to accuracy $\epsilon = cN^{-\frac{m}{2}}$. Our assumption that $f \in W_m^{N,2}$ implies that each of these constituent functions is Lipschitz continuous. Hence, it is easy to deduce that, e.g., if P, P_1, P_2 are approximations to the constituent functions h, h_1, h_2, respectively within an accuracy of ϵ, then since $\|h - P\| \leq \epsilon$, $\|h_1 - P_1\| \leq \epsilon$ and $\|h_2 - P_2\| \leq \epsilon$, then $\|h(h_1, h_2) - P(P_1, P_2)\| = \|h(h_1, h_2) - h(P_1, P_2) + h(P_1, P_2) - P(P_1, P_2)\| \leq \|h(h_1, h_2) - h(P_1, P_2)\| + \|h(P_1, P_2) - P(P_1, P_2)\| \leq c\epsilon$ by Minkowski inequality. Thus,

$$\|h(h_1, h_2) - P(P_1, P_2)\| \leq c\epsilon$$

for some constant $c > 0$ independent of the functions involved. This, together with the fact that there are $(n-1)$ nodes, leads to (6). □

Also in this case the proof provides the following corollary about the subset T_k^n of the space P_k^n which consists of compositional polynomials with a binary tree graph and constituent polynomial functions of degree k (in 2 variables).

Corollary 2. Let $\sigma : \mathbf{R} \to \mathbf{R}$ be infinitely differentiable, and not a polynomial. Let $n = 2^l$. Then, $f \in T_k^n$ can be realized by a deep network with a binary tree graph and a total of r units with $r = (n-1)\binom{2+k}{2} \approx (n-1)k^2$.

It is important to emphasize that the assumptions on σ in the theorems are not satisfied by the ReLU function $x \mapsto x_+$, but they are satisfied by smoothing the function in an arbitrarily small interval around the origin. This suggests that the result of Theorem 2 should be valid also for the non-smooth ReLU. Section 4 provides formal results. Stronger results than the theorems of this section[5] hold for networks where each unit evaluates a Gaussian non–linearity; i.e., Gaussian networks of the form

$$G(x) = \sum_{k=1}^{N} a_k \mathrm{e}^{-|x-w_k|^2}, \qquad x \in \mathbf{R}^d \tag{7}$$

where the approximation is on the entire Euclidean space.

In summary, when the only a priori assumption on the target function is about the number of derivatives, then to guarantee an accuracy of ϵ, we need a shallow network with $\mathcal{O}(\epsilon^{-\frac{n}{m}})$ trainable parameters. If we assume a hierarchical structure on the target function as in Theorem 2, then the corresponding deep network yields a guaranteed accuracy of ϵ with $\mathcal{O}(\epsilon^{-\frac{2}{m}})$ trainable parameters. Note that Theorem 2 applies to all f with a compositional architecture given by a graph which correspond to, or is a subgraph of, the graph associated with the deep network – in this case the graph corresponding to $W_m^{n,d}$. Theorem 2 leads naturally to the notion of effective dimensionality that we formalize in the next section.

Definition 1. The effective dimension of a class W of functions (for a given norm) is said to be d if for every $\epsilon > 0$, any function in W can be recovered within an accuracy of ϵ (as measured by the norm) using an appropriate network (either shallow or deep) with ϵ^{-d} parameters.

Thus, the effective dimension for the class W_m^n is $\frac{n}{m}$, that of $W_m^{n,2}$ is $\frac{2}{m}$.

4 General compositionality results: functions composed by a hierarchy of functions with bounded effective dimensionality

The main class of functions we considered in previous papers consists of functions as in Fig. 1 (b) that we called compositional functions. The term "compositionality" was used with the meaning it has in language and vision, where higher level concepts are composed of a small number of lower level ones, objects are composed of parts, sentences are composed of words and words are composed of syllables. Notice that this meaning of compositionality is narrower than the mathematical meaning of composition of functions. The compositional functions we have described in previous papers may be more precisely called functions composed of hierarchically local functions.

Here we generalize formally our previous results to the broader class of compositional functions (beyond the hierarchical locality of Figs. 1 (b), 1 (c) and 2) by restating formally a few comments of previous papers. Let begin with one of the previous examples. Consider

$$Q(x, y) = (Ax^2y^2 + Bx^2y + Cxy^2 + Dx^2 + 2Exy + Fy^2 + 2Gx + 2Hy + I)^{2^{10}}.$$

Since Q is nominally a polynomial of coordinatewise degree 2^{11}, Lemma 3.2 of [27] shows that a shallow network with

$2^{11}+1$ units is able to approximate Q arbitrarily well on I^2. However, because of the hierarchical structure of Q, Lemma 3.2 of [27] shows also that a hierarchical network with 9 units can approximate the quadratic expression, and 10 further layers, each with 3 units can approximate the successive powers. Thus, a hierarchical network with 11 layers and 39 units can approximate Q arbitrarily well. We note that even if Q is nominally of degree 2^{11}, each of the monomial coefficients in Q is a function of only 9 variables, $A, \cdots, I$.

A different example is

$$Q(x,y) = |x^2 - y^2|. \tag{8}$$

This is obviously a Lipschitz continuous function of 2 variables. The effective dimension of this class is 2, and hence, a shallow network would require at least $c\epsilon^{-2}$ parameters to approximate it within ϵ. However, the effective dimension of the class of univariate Lipschitz continuous functions is 1. Hence, if we take into account the fact that Q is a composition of a polynomial of degree 2 in 2 variables and the univariate Lipschitz continuous function $t \mapsto |t|$, then it is easy to see that the same approximation can be achieved by using a two layered network with $\mathcal{O}(\epsilon^{-1})$ parameters.

To formulate our most general result that includes the examples above as well as the constraint of hierarchical locality, we first define formally a compositional function in terms of a directed acyclic graph. Let $\mathcal{G}$ be a directed acyclic graph (DAG), with the set of nodes V. A $\mathcal{G}$–function is defined as follows. Each of the source node obtains an input from $\mathbf{R}$. Each in-edge of every other node represents an input real variable, and the node itself represents a function of these input real variables, called a constituent function. The out-edges fan out the result of this evaluation. We assume that there is only one sink node, whose output is the $\mathcal{G}$-function. Thus, ignoring the compositionality of this function, it is a function of n variables, where n is the number of source nodes in $\mathcal{G}$.

Theorem 3. Let $\mathcal{G}$ be a DAG, n be the number of source nodes, and for each $v \in V$, let d_v be the number of in-edges of v. Let $f : \mathbf{R}^n \mapsto \mathbf{R}$ be a compositional $\mathcal{G}$-function, where each of the constituent function is in $W^{d_v}_{m_v}$. Consider shallow and deep networks with infinitely smooth activation function as in Theorem 1. Then deep networks – with an associated graph that corresponds to the graph of f – avoid the curse of dimensionality in approximating f for increasing n, whereas shallow networks cannot directly avoid the curse. In particular, the complexity of the best approximating shallow network is exponential in n

$$N_s = \mathcal{O}(\epsilon^{-\frac{n}{m}}) \tag{9}$$

where $m = \min_{v \in V} m_v$, while the complexity of the deep network is

$$N_d = \mathcal{O}(\sum_{v \in V} \epsilon^{-\frac{d_v}{m_v}}). \tag{10}$$

Following Definition 1, we call $\frac{d_v}{m_v}$ the effective dimension of function v. Then, deep networks can avoid the curse of dimensionality if the constituent functions of a compositional function have a small effective dimension, i.e., have fixed, "small" dimensionality or fixed, "small" roughness. A different interpretation of Theorem 3 is the following.

Proposition 1. If a family of functions $f : \mathbf{R}^n \mapsto \mathbf{R}$ of smoothness m has an effective dimension $< \frac{n}{m}$, then the functions are compositional in a manner consistent with the estimates in Theorem 3.

Notice that the functions included in Theorem 3 are functions that are either local or the composition of simpler functions or both. Fig. 2 shows some examples in addition to the examples at the top of Fig. 1.

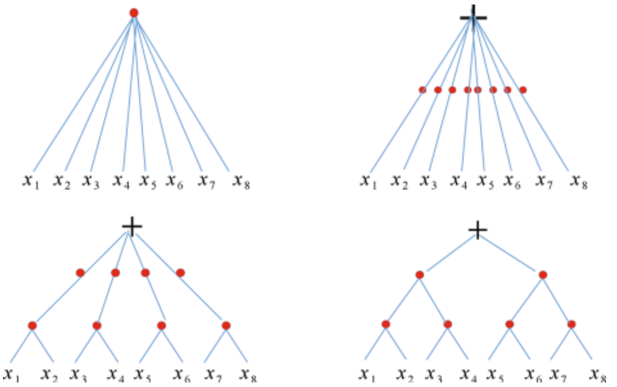
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
+
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
+
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
+
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8

Fig. 2 The figure shows the graphs of functions that may have small effective dimensionality, depending on the number of units per node required for good approximation.

As before, there is a simple corollary for polynomial functions:

Corollary 3. Let $\sigma : \mathbf{R} \to \mathbf{R}$ be infinitely differentiable, and not a polynomial. Let $S^n_k \in P^n_k$ be the family of compositional polynomial with a total number of monomials which is non-exponential, e.g., it is $\mathcal{O}(k^n)$. Then, $f \in S^n_k$ can be represented by a deep network with a a total of r units which is at most polynomial in n.

Notice that polynomials in S^n_k are sparse with a number of terms which is not exponential in n, i.e., it is not $\mathcal{O}(k^n)$ but linear in n (that is $\mathcal{O}(nk)$) or at most polynomial in n.

4.1 Approximation results for shallow and deep networks with (non-smooth) ReLUs

The results we described so far use smooth activation functions. We already mentioned why relaxing the smoothness assumption should not change our results in a fundamental way. While studies on the properties of neural networks with smooth activation abound, the results on non-smooth activation functions are much more sparse. Here we briefly recall some of them.

In the case of shallow networks, the condition of a smooth activation function can be relaxed to prove density (see Proposition 3.7 of [13]):

Proposition 2. Let $\sigma =: \mathbf{R} \to \mathbf{R}$ be in $\mathcal{C}^0$, and not a polynomial. Then shallow networks are dense in $\mathcal{C}^0$.

In particular, ridge functions using ReLUs of the form $\sum_{i=1}^{r} c_i(\langle w_i, x\rangle + b_i)_+$, with $w_i, x \in \mathbf{R}^n$, $c_i, b_i \in \mathbf{R}$ are dense in $\mathcal{C}$.

Networks with non-smooth activation functions are expected to do relatively poorly in approximating smooth functions such as polynomials in the sup norm. "Good" degree of approximation rates (modulo a constant) have been proved in the L_2 norm. Define B the unit ball in $\mathbf{R}^n$. Call $C^m(B^n)$ the set of all continuous functions with continuous derivative up to degree m defined on the unit ball. We define the Sobolev space W_p^m as the completion of $C^m(B^n)$ with respect to the Sobolev norm p (see page 168 of [13] for detail). We define the space $B_p^m = \{f\colon f \in W_p^m, \|f\|_{m,p} \leq 1\}$ and the approximation error $E(B_2^m; H; L_2) = \inf_{g \in H} \|f - g\|_{L_2}$. It is shown in Corollary 6.10 in [13] that

Proposition 3. For $M^r\colon f(x) = \sum_{i=1}^{r} c_i(\langle w_i, x\rangle + b_i)_+$, it holds $E(B_2^m; M_r; L_2) \leq C r^{-\frac{m}{n}}$ for $m = 1, \cdots, \frac{n+3}{2}$.

These approximation results with respect to the L^2 norm cannot be applied to derive bounds for compositional networks. Indeed, in the latter case, as we remarked already, estimates in the uniform norm are needed to control the propagation of the errors from one layer to the next, see Theorem 2. Results in this direction are given in [31], and more recently in [32] and [5] (see Theorem 3.1). In particular, using a result in [32] and following the proof strategy of Theorem 2, it is possible to derive the following results on the approximation of Lipshitz continuous functions with deep and shallow ReLU networks that mimics our Theorem 2.

Theorem 4. Let f be a L-Lipshitz continuous function of n variables. Then, the complexity of a network which is a linear combination of ReLU providing an approximation with accuracy at least ϵ is

$$N_s = \mathcal{O}\left(\left(\frac{\epsilon}{L}\right)^{-n}\right)$$

where that of a deep compositional architecture is

$$N_d = \mathcal{O}\left(\left(n-1)(\frac{\epsilon}{L}\right)^{-2}\right).$$

Our general Theorem 3 can be extended in a similar way. Theorem 4 is an example of how the analysis of smooth activation functions can be adapted to ReLU. Indeed, it shows how deep compositional networks with standard ReLUs can avoid the curse of dimensionality. In the above results, the regularity of the function class is quantified by the magnitude of Lipshitz constant. Whether the latter is best notion of smoothness for ReLU based networks, and if the above estimates can be improved, are interesting questions that we defer to a future work. A result that is more intuitive and may reflect what networks actually do is described in the Appendix of [29] (Section "Non-smooth ReLUs: how deep nets may work in reality"). Though the construction described there provides approximation in the L_2 norm but not in the sup norm, this is not a problem under any discretization of real number required for computer simulations (see the Appendix of [29]).

Fig. 3–6 provide a sanity check and empirical support for our main results and for the claims in the introduction.

4.2 Lower bounds and gaps

So far we have shown that there are deep networks – for instance of the convolutional type – that can avoid the curse of dimensionality if the functions they are learning are blessed with compositionality. There are no similar guarantee for shallow networks: for shallow networks approximating generic continuous functions the lower and the upper bound are both exponential[13]. From the point of view of machine learning, it is obvious that shallow networks, unlike deep ones, cannot exploit in their architecture the reduced number of parameters associated with priors corresponding to compositional functions. In past papers we listed a few examples, some of which are also valid lower bounds from the point of view of approximation theory:

1) The polynomial considered earlier

$$Q(x_1, x_2, x_3, x_4) = (Q_1(Q_2(x_1, x_2), Q_3(x_3, x_4)))^{1\,024}$$

which can be approximated by deep networks with a smaller number of parameters than shallow networks is based on polynomial approximation of functions of the type $g(g(g()))$. Here, however, a formal proof of the impossibility of good approximation by shallow networks is not available. For a lower bound, we need at one case of a compositional function which cannot be approximated by shallow networks with a non-exponential degree of approximation.

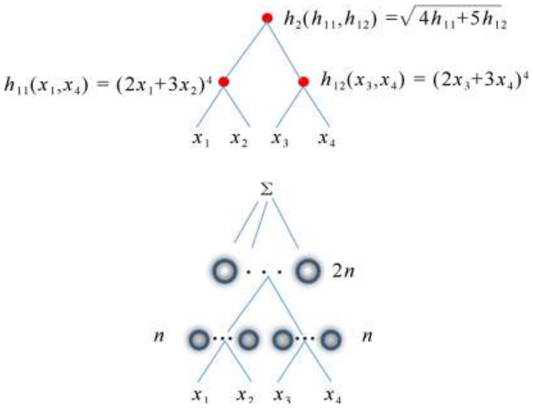
$h_2(h_{11}, h_{12}) = \sqrt{4h_{11} + 5h_{12}}$
$h_{11}(x_1, x_4) = (2x_1 + 3x_2)^4$
$h_{12}(x_3, x_4) = (2x_3 + 3x_4)^4$
x_1 x_2 x_3 x_4
Σ
$2n$
n
n
x_1 x_2 x_3 x_4

Fig. 3 The figure shows on the top the graph of the function to be approximated, while the bottom part of the figure shows a deep neural network with the same graph structure. The left and right node inf the first layer has each n units giving a total of $2n$ units in the first layer. The second layer has a total of $2n$ units. The first layer has a convolution of size n to mirror the structure of the function to be learned. The compositional function we approximate has the form $f(x_1, x_2, x_3, x_4) = h_2(h_{11}(x_1, x_2), h_{12}(x_3, x_4))$ with h_{11}, h_{12} and h_2 as indicated in the figure.

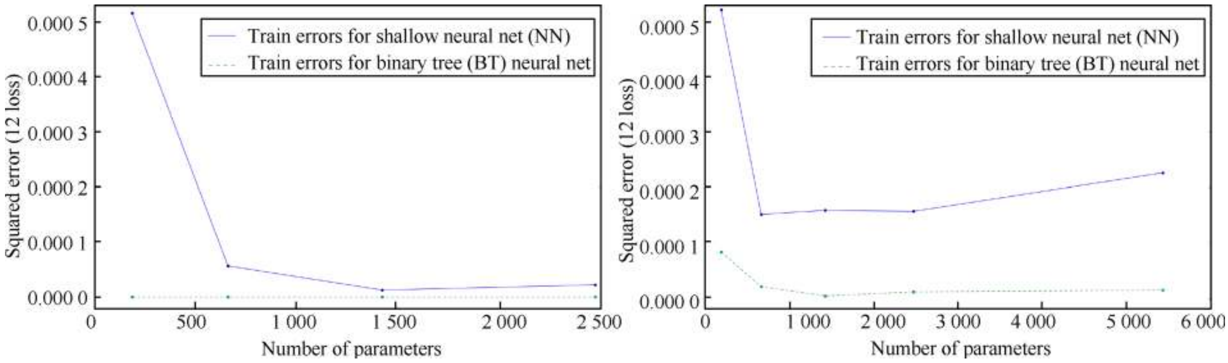

Fig. 4 An empirical comparison of shallow versus 2-layers binary tree networks in the approximation of compositional functions. The loss function is the standard mean square error (MSE). There are several units per node of the tree. In our setup here the network with an associated binary tree graph was set up so that each layer had the same number of units and shared parameters. The number of units for the shallow and binary tree neural networks had the same number of parameters. On the left, the function is composed of a single ReLU per node and is approximated by a network using ReLU activations. On the right, the compositional function is $f(x_1, x_2, x_3, x_4) = h_2(h_{11}(x_1, x_2), h_{12}(x_3, x_4))$ and is approximated by a network with a smooth ReLU activation (also called softplus). The functions h_1, h_2, h_3 are as described in Fig. 3. In order to be close to the function approximation case, a large data set of 60 K training examples was used for both training sets. We used for SGD the Adam[33] optimizer. In order to get the best possible solution, we ran 200 independent hyper parameter searches using random search[34] and reported the one with the lowest training error. The hyper parameters search was over the step size, the decay rate, frequency of decay and the mini-batch size. The exponential decay hyper parameters for Adam were fixed to the recommended values according to the original paper[33]. The implementations were based on TensorFlow[35].

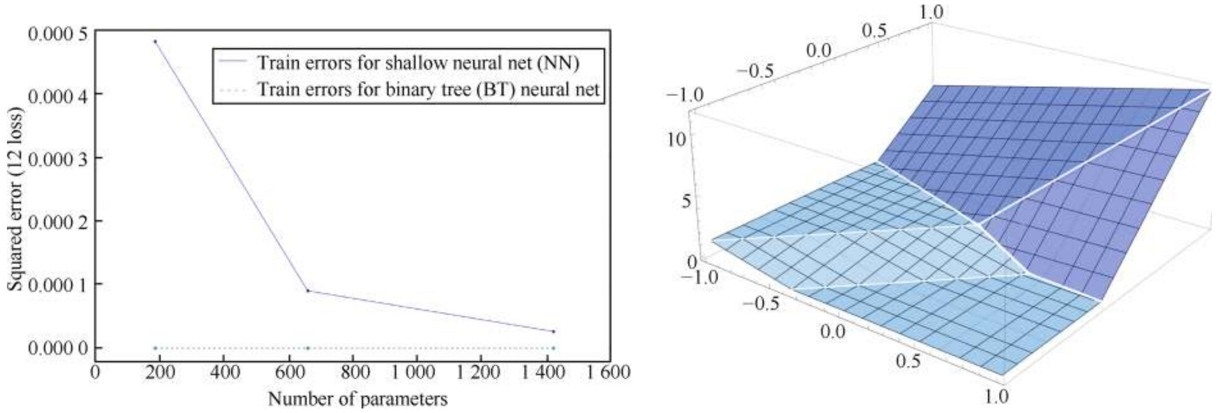

Fig. 5 Another comparison of shallow versus 2-layers binary tree networks in the learning of compositional functions. The set up of the experiment was the same as in the one in Fig. 4 except that the compositional function had two ReLU units per node instead of only one. The right part of the figure shows a cross section of the function $f(x_1, x_2, 0.5, 0.25)$ in a bounded interval $x_1 \in [-1, 1], x_2 \in [-1, 1]$. The shape of the function is piece wise linear as it is always the case for ReLUs networks.

2) Such an example, for which a proof of the lower bound exists since a few decades, consider a function which is a linear combination of n tensor product Chui–Wang spline wavelets, where each wavelet is a tensor product cubic spline. It is shown in [11, 12] that is impossible to implement such a function using a shallow neural network with a sigmoidal activation function using $\mathcal{O}(n)$ neurons, but a deep network with the activation function $(|x|_+)^2$ can do so. In this case, as we mentioned, there is a formal proof of a gap between deep and shallow networks. Similarly, Eldan and Shamir[36] show other cases with separations that are exponential in the input dimension.

3) As we mentioned earlier, Telgarsky proves an exponential gap between certain functions produced by deep networks and their approximation by shallow networks. The theorem[25] can be summarized as saying that a certain family of classification problems with real-valued inputs cannot be approximated well by shallow networks with fewer than exponentially many nodes whereas a deep network achieves zero error. This corresponds to high-frequency, sparse trigonometric polynomials in our case. His upper bound can be proved directly from our Theorem 2 by considering the real-valued polynomials $x_1x_2\cdots x_d$ defined on the cube $[-1, 1]^d$ which is obviously a compositional function with a binary tree graph.

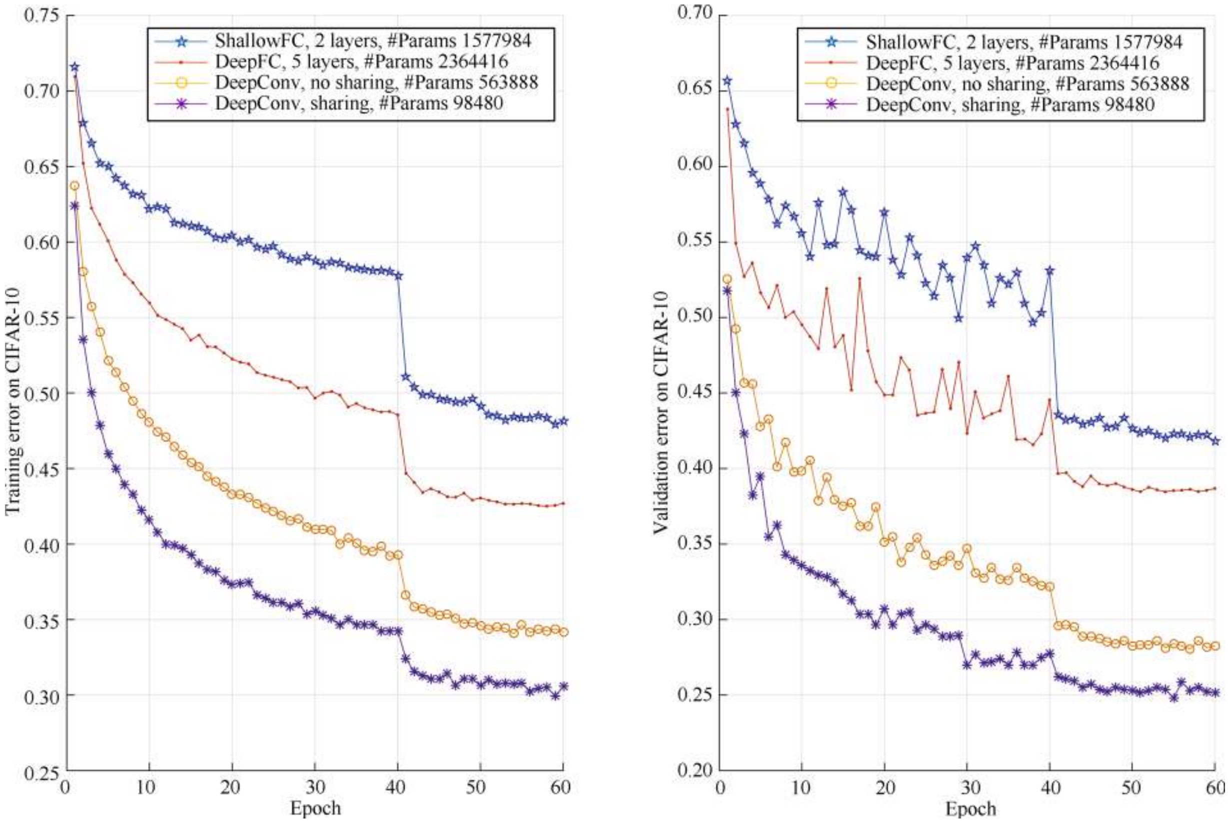

Fig. 6 We show that the main advantage of deep convolutional networks (convNets) comes from "hierarchical locality" instead of weight sharing. We train two 5-layer ConvNets with and without weight sharing on CIFAR-10. ConvNet without weight sharing has different filter parameters at each spatial location. There are 4 convolutional layers (filter size 3×3, stride 2) in each network. The number of feature maps (i.e., channels) are 16, 32, 64 and 128 respectively. There is an additional fully-connected layer as a classifier. The performances of a 2-layer and 5-layer fully-connected networks are also shown for comparison. Each hidden layer of the fully-connected network has 512 units. The models are all trained for 60 epochs with cross-entropy loss and standard shift and mirror flip data augmentation (during training). The training errors are higher than those of validation because of data augmentation. The learning rates are 0.1 for epoch 1 to 40, 0.01 for epoch 41 to 50 and 0.001 for rest epochs. The number of parameters for each model are indicated in the legends. Models with hierarchical locality significantly outperform shallow and hierarchical non-local networks.

4) We exhibit here another example of a compositional function that can be approximated well by deep networks but not by shallow networks.

Let $n \geq 2$ be an integer, $B \subset \mathbf{R}^n$ be the unit ball of $\mathbf{R}^n$. We consider the class W of all compositional functions $f = f_2 \circ f_1$, where $f_1 : \mathbf{R}^n \to \mathbf{R}$, and $\sum_{|k| \leq 4} \|D^k f_1\|_\infty \leq 1$, $f_2 : \mathbf{R} \to \mathbf{R}$ and $\|D^4 f_2\|_\infty \leq 1$. We consider

$$\Delta(\mathcal{A}_N) := \sup_{f \in W} \inf_{P \in \mathcal{A}_N} \|f - P\|_{\infty, B}$$

where $\mathcal{A}_N$ is either the class $\mathcal{S}_N$ of all shallow networks with N units or $\mathcal{D}_N$ of deep networks with two layers, the first with n inputs, and the next with one input. In both cases, the activation function is a C^∞ function $\sigma : \mathbf{R} \to \mathbf{R}$ that is not a polynomial.

Theorem 5. There exist constants $c_1, c_2 > 0$ such that for $N \geq c_1$,

$$\Delta(\mathcal{S}_N) \geq c_2. \tag{11}$$

In contrast, there exists $c_3 > 0$ such that

$$\Delta(\mathcal{D}_N) \leq c_3 N^{-\frac{4}{n}}. \tag{12}$$

The constants c_1, c_2, c_3 may depend upon n.

Proof. The estimate (12) follows from the estimates already given for deep networks. In the remainder of this proof, c will denote a generic positive constant depending upon n alone, but its value may be different at different occurrences. To prove (11), we use Lemma 3.2 in [12]. Let ϕ be a C^∞ function supported on $[0, 1]$, and we consider $f_N(x) = \phi(|4^N x|^2)$. Note that $\|f_N\|_1 \geq C$, with C independent of N. Then, it is clear that each $f_N \in W$, and $\|f_N\|_1 \geq c$. Clearly,

$$\Delta(\mathcal{S}_N) \geq c \inf_{P \in \mathcal{S}_N} \int_B |f_N(x) - P(x)| \mathrm{d}x. \tag{13}$$

We choose $P^*(x) = \sum_{k=1}^N \sigma(\langle w_k^*, x\rangle + b_k^*)$ such that

$$\inf_{P\in\mathcal{S}_N} \int_B |f_N(x) - P(x)|\mathrm{d}x \geq \\ \frac{1}{2}\int_B |f_N(x) - P^*(x)|\mathrm{d}x. \qquad (14)$$

Since f_N is supported on $\{x \in \mathbf{R}^n : |x| \leq 4^{-N}\}$, we may use Lemma 3.2 in [12] with $g_k^*(t) = \sigma(t + b_k^*)$ to conclude that

$$\int_B |f_N(x) - P(x)|\mathrm{d}x \geq \\ \inf_{g_k\in L^1_{loc},\ w_k\in\mathbf{R}^n,\ a_k\in\mathbf{R}} \int_B |f_N(x) - \sum a_k g_k(\langle w_k, x\rangle)|\mathrm{d}x \geq c.$$

Together with (13) and (14), this implies (11).

So by now plenty of examples of lower bounds exist showing a gap between shallow and deep networks. A particularly interesting case is the product function, that is the monomial $f(x_1, \cdots, x_n) = x_1 x_2 \cdots x_n$ which is, from our point of view, the prototypical compositional functions. Keeping in mind the issue of lower bounds, the question here has to do with the minimum integer $r(n)$ such that the function f is in the closure of the span of $\sigma(\langle w_k, x\rangle + b_k)$, with $k = 1, \cdots, r(n)$, and w_k, b_k ranging over their whole domains. Such a proof has been published for the case of smooth ReLUs, using unusual group techniques and is sketched in the Appendix of [37]:

Proposition 4. For shallow networks approximating the product monomial $f(x_1, \cdots, x_n) = x_1 x_2 \cdots x_n$, the minimum integer $r(n)$ is $r(n) = \mathcal{O}(2^n)$.

Notice, in support of the claim, that assuming that a shallow network with (non-smooth) ReLUs has a lower bound of $r(q) = \mathcal{O}(q)$ will lead to a contradiction with Håstad theorem by restricting x_i from $x_i \in (-1, 1)$ to $x_i \in \{-1, +1\}$. Håstad theorem[38] establishes the inapproximability of the parity function by shallow circuits of non-exponential size. In fact, Håstad′s theorem can be used to prove Proposition 4 by approximating the product function using the binary representation of x_i. This requires combining a number of products of Boolean variables: Håstad result applies to each of the products. The upper bound for approximation by deep networks of ReLUs can be obtained following the arguments of Proposition 7 in the Appendix of [29] (Section "Non-smooth ReLUs: how deep nets may work in reality").

4.3 Messy graphs and densely connected deep networks

As mentioned already, the approximating deep network does not need to exactly match the architecture of the compositional function as long as the graph or tree associated with the function is contained in the graph associated with the network. This is of course good news: The compositionality prior embedded in the architecture of the network does not reflect exactly the graph of a new function to be learned. We have shown that for a given class of compositional functions characterized by an associated graph, there exists a deep network that approximates such a function better than a shallow network. The same network approximates well functions characterized by subgraphs of the original class.

The proofs of our theorems show that linear combinations of compositional functions are universal in the sense that they can approximate any function and that deep networks with a number of units that increases exponentially with layers can approximate any function. Notice that deep compositional networks can interpolate if they are overparametrized with respect to the data, even if the data reflect a non-compositional function (see Proposition 8 in the Appendix of [29], Section "Optimization of compositional functions and Bezout theorem").

As an aside, note that the simplest compositional function – addition – is trivial in the sense that it offers no approximation advantage to deep networks. The key function is multiplication which is for us the prototypical compositional functions. As a consequence, polynomial functions are compositional – they are linear combinations of monomials which are compositional.

As we mentioned earlier, networks corresponding to graphs that include the graph of the function to be learned can exploit compositionality. The bounds, however, will depend on the number of parameters r in the network used and not the parameters $r*$ ($r* < r$) of the optimal deep network with a graph exactly matched to the graph of the function to be learned. As an aside, the price to be paid in using a non-optimal prior depend on the learning algorithm. For instance, under sparsity constraints, it may be possible to pay a smaller price than r (but higher than r^*).

In this sense, some of the densely connected deep networks used in practice – which contain sparse graphs possibly relevant for the function to be learned and which are still "smaller" than the exponential number of units required to represent a generic function of n variables – may be capable in some cases of exploiting an underlying compositionality structure without paying an exorbitant price in terms of required complexity.

5 Connections with the theory of Boolean functions

The approach followed in our main theorems suggest the following considerations (see the Appendix of [29], Section "Boolean Functions" for a brief introduction). The structure of a deep network is reflected in polynomials that are best approximated by it – for instance generic polynomials or sparse polynomials (in the coefficients) in d variables of order k. The tree structure of the nodes of a deep network reflects the structure of a specific sparse polynomial. Generic polynomial of degree k in d variables are difficult to learn because the number of terms, trainable parameters and associated VC-dimension are all exponential in d. On the other hand, functions approximated well by sparse polynomials can be learned efficiently by deep networks with a tree structure that matches the polynomial. We recall that in a similar way several properties of certain Boolean functions can be "read out" from the terms of their Fourier expansion corresponding to "large" coefficients, that is from a polynomial that approximates well the function.

Classical results[38] about the depth-breadth tradeoff in circuits design show that deep circuits are more efficient in representing certain Boolean functions than shallow circuits. Håstad proved that highly-variable functions (in the sense of having high frequencies in their Fourier spectrum), in particular the parity function cannot even be decently approximated by small constant depth circuits[39]. A closely related result follow immediately from Theorem 2 since functions of real variables of the form $x_1x_2\cdots x_d$ have the compositional form of the binary tree (for d even). Restricting the values of the variables to $-1,+1$ yields an upper bound:

Proposition 5. The family of parity functions $x_1x_2\cdots x_d$ with $x_i \in \{-1,+1\}$ and $i = 1,\cdots, x_d$ can be represented with exponentially fewer units by a deep than a shallow network.

Notice that Håstad′s results on Boolean functions have been often quoted in support of the claim that deep neural networks can represent functions that shallow networks cannot. For instance, Bengio and LeCun[40] write "We claim that most functions that can be represented compactly by deep architectures cannot be represented by a compact shallow architecture."

Finally, we want to mention a few other observations on Boolean functions that show an interesting connection with our approach. It is known that within Boolean functions the AC^0 class of polynomial size constant depth circuits is characterized by Fourier transforms where most of the power spectrum is in the low order coefficients. Such functions can be approximated well by a polynomial of low degree and can be learned well by considering only such coefficients. There are two algorithms[41] that allow learning of certain Boolean function classes:

1) the low order algorithm that approximates functions by considering their low order Fourier coefficients

2) the sparse algorithm which learns a function by approximating its significant coefficients.

Decision lists and decision trees can be learned by the first algorithm. Functions with small L_1 norm can be approximated well by the second algorithm. Boolean circuits expressing DNFs can be approximated by the first one but even better by the second. In fact, in many cases a function can be approximated by a small set of coefficients but these coefficients do not correspond to low-order terms. All these cases are consistent with the notes about sparse functions in Section 7.

6 Generalization bounds

Our estimate of the number of units and parameters needed for a deep network to approximate compositional functions with an error ϵ_G allow the use of one of several available bounds for the generalization error of the network to derive sample complexity bounds. As an example consider Theorem 16.2 in [42] which provides the following sample bound for a generalization error ϵ_G with probability at least $1-\delta$ in a network in which the W parameters (weights and biases) which are supposed to minimize the empirical error (the theorem is stated in the standard ERM setup) are expressed in terms of k bits:

$$M(\epsilon_G,\delta) \leq \frac{2}{\epsilon_G^2}\left(kW\log 2 + \log\left(\frac{2}{\delta}\right)\right). \tag{15}$$

This suggests the following comparison between shallow and deep compositional (here binary tree-like networks). Assume a network size that ensures the same approximation error ϵ .

Then in order to achieve the same generalization error ϵ_G, the sample size $M_{shallow}$ of the shallow network must be much larger than the sample size M_{deep} of the deep network:

$$\frac{M_{deep}}{M_{shallow}} \approx \epsilon^n. \tag{16}$$

This implies that for largish n, there is a (large) range of training set sizes between M_{deep} and $M_{shallow}$ for which deep networks will not overfit (corresponding to small ϵ_G) but shallow networks will (for dimensionality $n \approx 10^4$ and $\epsilon \approx 0.1$, (16) yields $M_{shallow} \approx 10^{10^4} M_{deep}$).

A similar comparison is derived if one considers the best possible expected error obtained by a deep and a shallow network. Such an error is obtained finding the architecture with the best trade-off between the approximation and the estimation error. The latter is essentially of the same order as the generalization bound implied by (15), and is essentially the same for deep and shallow networks, i.e.,

$$\frac{rn}{\sqrt{M}} \tag{17}$$

where we denoted by M the number of samples. For shallow networks, the number of parameters corresponds to r units of n dimensional vectors (plus off-sets), whereas for deep compositional networks the number of parameters corresponds to r units of 2 dimensional vectors (plus off-sets) in each of the $n-1$ units. Using our previous results on degree of approximation, the number of units giving the best approximation/estimation trade-off is

$$r_{shallow} \approx \left(\frac{\sqrt{M}}{n}\right)^{\frac{n}{m+n}} \quad \text{and} \quad r_{deep} \approx \left(\sqrt{M}\right)^{\frac{2}{m+2}} \tag{18}$$

for shallow and deep networks, respectively. The corresponding (excess) expected errors $\mathcal{E}$ are

$$\mathcal{E}_{shallow} \approx \left(\frac{n}{\sqrt{M}}\right)^{\frac{m}{m+n}} \tag{19}$$

for shallow networks and

$$\mathcal{E}_{deep} \approx \left(\frac{1}{\sqrt{M}}\right)^{\frac{m}{m+2}} \tag{20}$$

for deep networks. For the expected error, as for the generalization error, deep networks appear to achieve an exponential gain. The above observations hold under the assumption that the optimization process during training finds the optimum parameters values for both deep and shallow networks. Taking into account optimization, e.g., by stochastic gradient descent, requires considering a further error term, but we expect that the overall conclusions about generalization properties for deep versus shallow networks should still hold true.

Notice that independently of considerations of generalization, deep compositional networks are expected to be very efficient memories – in the spirit of hierarchical vector quantization – for associative memories reflecting compositional rules (see the Appendix of [29], Section "Vector quantization and hierarchical vector quantization" and [43]). Notice that the advantage with respect to shallow networks from the point of view of memory capacity can be exponential (as in the example after (16) showing $M_{shallow} \approx 10^{10^4} M_{deep}$).

6.1 Generalization in multi-class deep networks

Most of the successful neural networks exploit compositionality for better generalization in an additional important way[44]. Suppose that the mappings to be learned in a family of classification tasks (for instance classification of different object classes in Imagenet) may be approximated by compositional functions such as $f(x_1, \cdots, x_n) = h_l(\cdots(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8) \cdots)) \cdots)$, where h_l depends on the task (for instance to which object class) but all the other constituent functions h are common across the tasks. Under such an assumption, multi-task learning, that is training simultaneously for the different tasks, forces the deep network to "find" common constituent functions. Multi-task learning has theoretical advantages that depends on compositionality: The sample complexity of the problem can be significantly lower[45]. The Maurer′s approach is in fact to consider the overall function as the composition of a preprocessor function common to all task followed by a task-specific function. As a consequence, the generalization error, defined as the difference between expected and empirical error, averaged across the T tasks, is bounded with probability at least $1 - \delta$ (in the case of finite hypothesis spaces) by

$$\frac{1}{\sqrt{2M}}\sqrt{\ln|\mathcal{H}| + \frac{\ln|\mathcal{G}| + \ln(\frac{1}{\delta})}{T}} \tag{21}$$

where M is the size of the training set, $\mathcal{H}$ is the hypothesis space of the common classifier and $\mathcal{G}$ is the hypothesis space of the system of constituent functions, common across tasks.

The improvement in generalization error because of the multitask structure can be in the order of the square root of the number of tasks (in the case of Imagenet with its 1 000 object classes, the generalization error may therefore decrease by a factor ≈ 30). It is important to emphasize the dual advantage here of compositionality, which 1) reduces generalization error by decreasing the complexity of the hypothesis space $\mathcal{G}$ of compositional functions relative the space of non-compositional functions and 2) exploits the multi task structure, that replaces $\ln|\mathcal{G}|$ with $\frac{\ln|\mathcal{G}|}{T}$.

We conjecture that the good generalization exhibited by deep convolutional networks in multi-class tasks such as CiFAR and Imagenet are due to three factors:

1) SGD has a regularizing effect.

2) The task is compositional.

3) The task is multiclass.

7 Notes on a theory of compositional computation

The key property of the theory of compositional functions sketched here is that certain deep networks can learn them avoiding the curse of dimensionality because of the blessing of compositionality via a small effective dimension.

We state here several comments and conjectures.

1. General comments

1) Assumptions of the compositionality type may have more direct practical implications and be at least as effective as assumptions about function smoothness in countering the curse of dimensionality in learning and approximation.

2) The estimates on the n-width imply that there is some function in either W_m^n (Theorem 1) or $W_m^{n,2}$ (Theorem 2) for which the approximation cannot be better than that suggested by the theorems.

3) The main question that may be asked about the relevance of the theoretical results of this paper and networks used in practice has to do with the many "channels" used in the latter and with our assumption that each node in the networks computes a scalar function – the linear combination of r units (3). The following obvious but interesting extension of Theorem 1 to vector-valued functions says that the number of hidden units required for a given accuracy in each component of the function is the same as in the scalar case considered in our theorems (of course the number of weights is larger):

Corollary 4. Let $\sigma : \mathbf{R} \to \mathbf{R}$ be infinitely differentiable, and not a polynomial. For a vector-valued function $f : \mathbf{R}^n \to \mathbf{R}^q$ with components $f_i \in W_m^n,\ i = 1, \cdots, q$, the number of hidden units in shallow networks with n inputs, q outputs that provide accuracy at least ϵ in each of the components of f is

$$N = \mathcal{O}(\epsilon^{-\frac{n}{m}}). \tag{22}$$

The demonstration follows the proof of Theorem 1, see also the Section "Neural networks: polynomial viewpoint" in the Appendix of [29]. It amounts to realizing that the hidden units (or linear combinations of them) can be equivalent to the monomials of a generic polynomial of degree k in n variables that can be used by a different set of coefficients for each of f_i. This argument of course does not mean that during learning this is what happens; it provides one way to synthesize the approximation and an associated upper bound. The corollary above leads to a simple argument that generalizes our binary tree results to standard, multi-channel deep convolutional networks by introducing a set of virtual linear units as outputs of one layer and inputs of the next one.

4) We have used polynomials (see the Appendix of [29], Section "Non-smooth ReLUs: how deep nets may work in reality") to prove results about complexity of approximation in the case of neural networks. Neural network learning with SGD may or may not synthesize polynomial, depending on the smoothness of the activation function and on the target. This is not a problem for theoretically establishing upper bounds on the degree of convergence because results using the framework on nonlinear width guarantee the "polynomial" bounds are optimal.

5) Both shallow and deep representations may or may not reflect invariance to group transformations of the inputs of the function[22, 46]. Invariance – also called weight sharing – decreases the complexity of the network. Since we are interested in the comparison of shallow versus deep architectures, we have considered the generic case of networks (and functions) for which invariance is not assumed. In fact, the key advantage of deep versus shallow network – as shown by the proof of the theorem – is the associated hierarchical locality (the constituent functions in each node are local that is have a small dimensionality) and not invariance (which designates shared weights that is nodes at the same level sharing the same function). One may then ask about the relation of these results with i-theory[47]. The original core of i-theory describes how pooling can provide either shallow or deep networks with invariance and selectivity properties. Invariance of course helps but not exponentially as hierarchical locality does.

6) There are several properties that follow from the theory here which are attractive from the point of view of neuroscience. A main one is the robustness of the results with respect to the choice of nonlinearities (linear rectifiers, sigmoids, Gaussians, etc.) and pooling.

7) In a machine learning context, minimization over a training set of a loss function such as the square loss yields an empirical approximation of the regression function $p(y/x)$. Our hypothesis of compositionality becomes an hypothesis about the structure of the conditional probability function.

2. Spline approximations, Boolean functions and tensors

1) Consider again the case of Section 4 "General compositionality results" in the Appendix of [29] of a multivariate function $f : [0,1]^d \rightarrow \mathbf{R}$. Suppose to discretize it by a set of piecewise constant splines and their tensor products. Each coordinate is effectively replaced by n Boolean variables. This results in a d-dimensional table with $N = n^d$ entries. This in turn corresponds to a boolean function $f : \{0,1\}^N \rightarrow \mathbf{R}$. Here, the assumption of compositionality corresponds to compressibility of a d-dimensional table in terms of a hierarchy of $(d-1)$ two-dimensional tables. Instead of n^d entries there are $(d-1)n^2$ entries. This has in turn obvious connections with hierarchical vector quantization (HVQ), discussed in the Appendix of [29], Section "Vector quantization and hierarchical vector quantization".

2) As the Appendix of [29], Section "Non-smooth ReLUs: how deep nets may work in reality" shows, every function f can be approximated by an epsilon-close binary function f_B. Binarization of $f : \mathbf{R}^n \rightarrow \mathbf{R}$ is done by using k partitions for each variable x_i and indicator functions. Thus, $f \mapsto f_B : \{0,1\}^{kn} \rightarrow \mathbf{R}$ and $sup|f - f_B| \leq \epsilon$, with ϵ depending on k and bounded Df.

3) f_B can be written as a polynomial (a Walsh decomposition) $f_B \approx p_B$. It is always possible to associate a p_b to any f, given ϵ.

4) The binarization argument suggests a direct way to connect results on function approximation by neural nets with older results on Boolean functions. The latter are special cases of the former results.

5) One can think about tensors in terms of d-dimensional tables. The framework of hierarchical decompositions of tensors – in particular the hierarchical tucker format – is closely connected to our notion of compositionality. Interestingly, the hierarchical tucker decomposition has been the subject of recent papers on deep learning[20]. This work, as well more classical papers[48], does not characterize directly the class of functions for which these decompositions are effective. Notice that tensor decompositions assume that the sum of polynomial functions of order d is sparse (see equation at the top of page 2 030 of [48]). Our results provide a rigorous grounding for the tensor work related to deep learning. There is obviously a wealth of interesting connections with approximation theory that should be explored.

Notice that the notion of separation rank of a tensor is very closely related to the effective r in (32) of the Appendix of [29] (Section "Neural networks: polynomial viewpoint").

3. Sparsity

1) We suggest to define binary sparsity of f, in terms of the sparsity of the Boolean function p_B. Binary sparsity implies that an approximation to f can be learned by non-exponential deep networks via binarization. Notice that if the function f is compositional, the associated Boolean functions f_B is sparse. The converse is not true.

2) In may situations, Tikhonov regularization corresponds to cutting high order Fourier coefficients. Sparsity of the coefficients subsumes Tikhonov regularization in the case of a Fourier representation. Notice that as an effect, the number of Fourier coefficients is reduced, that is trainable parameters, in the approximating trigonometric polynomial. Sparsity of Fourier coefficients is a general constraint for learning Boolean functions.

3) Sparsity in a specific basis. A set of functions may be defined to be sparse in a specific basis when the number of parameters necessary for its ϵ-approximation increases less than exponentially with the dimensionality. An open question is the appropriate definition of sparsity. The notion of sparsity we suggest here is the effective r in (32) of the

Appendix of [29] (Section "Neural networks: polynomial viewpoint"). For a general function $r \approx k^n$, we may define sparse functions those for which $r \ll k^n$ in

$$f(x) \approx P_k^*(x) = \sum_{i=1}^{r} p_i(\langle w_i, x\rangle) \tag{23}$$

where P^* is a specific polynomial that approximates $f(x)$ within the desired ϵ. Notice that the polynomial P_k^* can be a sum of monomials or a sum of, for instance, orthogonal polynomials with a total of r parameters. In general, sparsity depends on the basis and one needs to know the basis and the type of sparsity to exploit it in learning, for instance with a deep network with appropriate activation functions and architecture.

There are function classes that are sparse in every bases. Examples are compositional functions described by a binary tree graph.

4. Theory of computation, locality and compositionality

1) From the computer science point of view, feedforward multilayer networks are equivalent to finite state machines running for a finite number of time steps[49, 50]. This result holds for almost any fixed nonlinearity in each layer. Feedforward networks are equivalent to cascades without loops (with a finite number of stages) and all other forms of loop free cascades (i.e., McCulloch-Pitts nets without loops, perceptrons, analog perceptrons, linear threshold machines). Finite state machines, cascades with loops, and difference equation systems which are Turing equivalent, are more powerful than multilayer architectures with a finite number of layers. The latter networks, however, are practically universal computers, since every machine we can build can be approximated as closely as we like by defining sufficiently many stages or a sufficiently complex single stage. Recurrent networks as well as differential equations are Turing universal.

2) Approximation properties can be related to the notion of connectivity of a network. Connectivity is a key property in network computations. Local processing may be a key constraint also in neuroscience. One of the natural measures of connectivity that can be introduced is the order of a node defined as the number of its distinct inputs. The order of a network is then the maximum order among its nodes. The term order dates back to the Perceptron book[50, 51]. From the previous observations, it follows that a hierarchical network of order at least 2 can be universal. In the Perceptron book, many interesting visual computations have low order (e.g., recognition of isolated figures). The message is that they can be implemented in a single layer by units that have a small number of inputs. More complex visual computations require inputs from the full visual field. A hierarchical network can achieve effective high order at the top using units with low order. The network architecture of Fig. 1 (b) has low order: Each node in the intermediate layers is connected to just 2 other nodes, rather than (say) all nodes in the previous layer (notice that the connections in the trees of the figures may reflect linear combinations of the input units).

3) Low order may be a key constraint for cortex. If it captures what is possible in terms of connectivity between neurons, it may determine by itself the hierarchical architecture of cortex which in turn may impose compositionality to language and speech.

4) The idea of functions that are compositions of "simpler" functions extends in a natural way to recurrent computations and recursive functions. For instance, $h(f^{(t)}(g(x)))$ represents t iterations of the algorithm f (h and g match input and output dimensions to f).

8 Why are compositional functions so common or important?

First, let us formalize the requirements on the algorithms of local compositionality is to define scalable computations as a subclass of nonlinear discrete operators, mapping vectors from $\mathbf{R}^n$ into $\mathbf{R}^d$ (for simplicity we put in the following $d = 1$). Informally, we call an algorithm $K_n : \mathbf{R}^n \mapsto \mathbf{R}$ scalable if it maintains the same "form" when the input vectors increase in dimensionality, i.e., the same kind of computation takes place when the size of the input vector changes. This motivates the following construction. Consider a "layer" operator $H_{2m} : \mathbf{R}^{2m} \mapsto \mathbf{R}^{2m-2}$ for $m \geq 1$ with a special structure that we call "shift invariance".

Definition 2. For integer $m \geq 2$, an operator H_{2m} is shift-invariant if $H_{2m} = H'_m \oplus H''_m$ where $\mathbf{R}^{2m} = \mathbf{R}^m \oplus \mathbf{R}^m$, $H' = H''$ and $H' : \mathbf{R}^m \mapsto \mathbf{R}^{m-1}$. An operator $K_{2M} : \mathbf{R}^{2M} \to \mathbf{R}$ is called scalable and shift invariant if $K_{2M} = H_2 \circ \cdots \circ H_{2M}$ where each H_{2k}, $1 \leq k \leq M$, is shift invariant.

We observe that scalable shift-invariant operators $K : \mathbf{R}^{2m} \mapsto \mathbf{R}$ have the structure $K = H_2 \circ H_4 \circ H_6 \circ \cdots \circ H_{2m}$, with $H_4 = H'_2 \oplus H'_2$, $H_6 = H''_2 \oplus H''_2 \oplus H''_2$, etc.

Thus, the structure of a shift-invariant, scalable operator consists of several layers. Each layer consists of identical blocks. Each block is an operator $H : \mathbf{R}^2 \mapsto \mathbf{R}$: see Fig. 7. We note also that an alternative weaker constraint on H_{2m} in Definition 2, instead of shift invariance, is mirror symmetry, i.e., $H'' = R \circ H'$, where R is a reflection. Obviously, shift-invariant scalable operator are equivalent to shift-invariant compositional functions. Obviously the definition can be changed in several of its details. For instance for two-dimensional images, the blocks could be operators $H : \mathbf{R}^5 \to \mathbf{R}$ mapping a neighborhood around each pixel into a real number.

The final step in the argument uses the results of previous sections to claim that a nonlinear node with two inputs and enough units can approximate arbitrarily well each of the H_2 blocks. This leads to conclude that deep convolutional neural networks are natural approximators of scalable, shift-invariant operators.

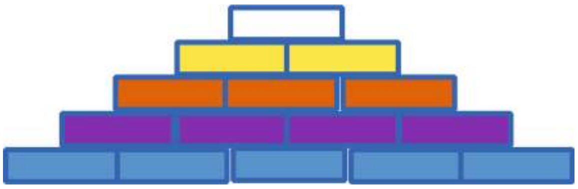

Fig. 7 A shift-invariant, scalable operator. Processing is from the bottom (input) to the top (output). Each layer consists of identical blocks, each block has two inputs and one output; each block is an operator $H_2 : \mathbf{R}^2 \mapsto \mathbf{R}$. The step of combining two inputs to a block into one output corresponds to coarse-graining of a Ising model.

Let us provide a couple of very simple examples of compositional functions. Addition is compositional but degree of approximation does not improve by decomposing addition in different layers of network; all linear operators are compositional with no advantage for deep networks; multiplication as well as the AND operation (for Boolean variables) is the prototypical compositional function that provides an advantage to deep networks.

This line of arguments defines a class of algorithms that is universal and can be optimal for a large set of problems. It does not however explain why problems encountered in practice should match this class of algorithms. Though we and others have argued that the explanation may be in either the physics or the neuroscience of the brain, these arguments (see the Appendix of [29], Section "Does physics or neuroscience imply compositionality?") are not (yet) rigorous. Our conjecture is that compositionality is imposed by the wiring of our cortex and is reflected in language. Thus, compositionality of many computations on images many reflect the way we describe and think about them.

Acknowledgement

The authors thank O. Shamir for useful emails that prompted us to clarify our results in the context of lower bounds.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

[1] F. Anselmi, L. Rosasco, C. Tan, T. Poggio. Deep Convolutional Networks are Hierarchical Kernel Machines, Center for Brains, Minds and Machines (CBMM) Memo No. 035, The Center for Brains, Minds and Machines, USA, 2015.

[2] T. Poggio, L. Rosasco, A. Shashua, N. Cohen, F. Anselmi. Notes on Hierarchical Splines, DCLNs and i-theory, Center for Brains, Minds and Machines (CBMM) Memo No. 037, The Center for Brains, Minds and Machines, USA, 2015.

[3] T. Poggio, F. Anselmi, L. Rosasco. I-theory on Depth vs Width: Hierarchical Function Composition, Center for Brains, Minds and Machines (CBMM) Memo No. 041, The Center for Brains, Minds and Machines, USA, 2015.

[4] H. Mhaskar, Q. L. Liao, T. Poggio. Learning Real and Boolean Functions: When is Deep Better than Shallow, Center for Brains, Minds and Machines (CBMM) Memo No. 045, The Center for Brains, Minds and Machines, USA, 2016.

[5] H. N. Mhaskar, T. Poggio. Deep Vs. Shallow Networks: An Approximation Theory Perspective, Center for Brains, Minds and Machines (CBMM) Memo No. 054, The Center for Brains, Minds and Machines, USA, 2016.

[6] D. L. Donoho. High-dimensional data analysis: The curses and blessings of dimensionality. *Lecture – Math Challenges of Century*, vol. 13, pp. 178–183, 2000.

[7] Y. LeCun, Y. Bengio, G. Hinton. Deep learning. *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[8] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[9] M. Riesenhuber, T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, vol. 2, no. 11, pp. 1019–1025, 1999.

[10] H. N. Mhaskar. Approximation properties of a multilayered feedforward artificial neural network. *Advances in Computational Mathematics*, vol. 1, no. 1, pp. 61–80, 1993.

[11] C. K. Chui, X. Li, H. Mhaskar. Neural networks for localized approximation. *Mathematics of Computation*, vol. 63, no. 208, pp. 607–623, 1994.

[12] C. K. Chui, X. Li, H. N. Mhaskar. Limitations of the approximation capabilities of neural networks with one hidden layer. *Advances in Computational Mathematics*, vol. 5, no. 1, pp. 233–243, 1996.

[13] A. Pinkus. Approximation theory of the MLP model in neural networks. *Acta Numerica*, vol. 8, pp. 143–195, 1999.

[14] T. Poggio, S. Smale. The mathematics of learning: Dealing with data. *Notices of the American Mathematical Society*, vol. 50, no. 5, pp. 537–544, 2003.

[15] B. Moore, T. Poggio. Representation properties of multilayer feedforward networks. *Neural Networks*, vol. 1, no. S1, pp. 203, 1998.

[16] R. Livni, S. Shalev-Shwartz, O. Shamir. A provably efficient algorithm for training deep networks. CoRR, abs/1304.7045, 2013.

[17] O. Delalleau, Y. Bengio. Shallow vs. deep sum-product networks. In *Proceedings of Advances in Neural Information Processing Systems 24*, NIPS, Granada, Spain, pp. 666–674, 2011.

[18] G. F. Montufar, R. Pascanu, K. Cho, Y. Bengio. On the number of linear regions of deep neural networks. In *Proceedings of Advances in Neural Information Processing Systems 27*, NIPS, Denver, USA, pp. 2924–2932, 2014.

[19] H. N. Mhaskar. Neural networks for localized approximation of real functions. In *Proceedings of IEEE-SP Workshop on Neural Networks for Processing III*, pp. 190–196, IEEE, Linthicum Heights, USA, 1993.

[20] N. Cohen, O. Sharir, A. Shashua. On the expressive power of deep learning: A tensor analysis. arXiv:1509.0500v1, 2015.

[21] F. Anselmi, J. Z. Leibo, L. Rosasco, J. Mutch, A. Tacchetti, T. Poggio. Unsupervised Learning of Invariant Representations With Low Sample Complexity: The Magic of Sensory Cortex or A New Framework for Machine Learning? Center for Brains, Minds and Machines (CBMM) Memo No. 001, The Center for Brains, Minds and Machines, USA, 2014.

[22] F. Anselmi, J. Z. Leibo, L. Rosasco, J. Mutch, A. Tacchetti, T. Poggio. Unsupervised learning of invariant representations. *Theoretical Computer Science*, vol. 633, pp. 112–121, 2016.

[23] T. Poggio, L. Rosaco, A. Shashua, N. Cohen, F. Anselmi. Notes on Hierarchical Splines, DCLNs and i-theory, Center for Brains, Minds and Machines (CBMM) Memo No. 037. The Center for Brains, Minds and Machines, 2015.

[24] Q. L. Liao, T. Poggio. Bridging the Gaps between Residual Learning, Recurrent Neural Networks and Visual Cortex, Center for Brains, Minds and Machines (CBMM) Memo No. 047, The Center for Brains, Minds and Machines, 2016.

[25] M. Telgarsky. Representation benefits of deep feedforward networks. arXiv:1509.08101v2, 2015.

[26] I. Safran, O. Shamir. Depth separation in ReLU networks for approximating smooth non-linear functions. arXiv:1610.09887v1, 2016.

[27] H. N. Mhaskar. Neural networks for optimal approximation of smooth and analytic functions. *Neural Computation*, vol. 8, no. 1, pp. 164–177, 1996.

[28] E. Corominas, F. S. Balaguer. Conditions for an infinitely differentiable function to be a polynomial. *Revista Matemática Hispanoamericana* vol. 14, no. 1–2, pp. 26–43, 1954. (in Spanish)

[29] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, Q. L. Liao. Why and when can deep–but not shallow–networks avoid the curse of dimensionality: A review. arXiv:1611.00740v3, 2016.

[30] R. A. DeVore, R. Howard C. A. Micchelli. Optimal nonlinear approximation. *Manuscripta Mathematica*, vol. 63, no. 4, pp. 469–478, 1989.

[31] H. N. Mhaskar. On the tractability of multivariate integration and approximation by neural networks. *Journal of Complexity*, vol. 20, no. 4, pp. 561–590, 2004.

[32] F. Bach. Breaking the curse of dimensionality with convex neural networks. arXiv:1412.8690, 2014.

[33] D. Kingma, J. Ba. Adam: A method for stochastic optimization. arXiv:1412.6980, 2014.

[34] J. Bergstra, Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[35] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. F. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Q. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Q. Zheng. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. arXiv:1603.04467, 2016.

[36] R. Eldan, O. Shamir. The power of depth for feedforward neural networks. arXiv:1512.03965v4, 2016.

[37] H. W. Lin, M. Tegmark. Why does deep and cheap learning work so well? arXiv:1608.08225, 2016.

[38] J. T. Håstad. *Computational Limitations for Small Depth Circuits*, Cambridge, MA, USA: MIT Press, 1987.

[39] N. Linial, Y. Mansour, N. Nisan. Constant depth circuits, Fourier transform, and learnability. *Journal of the ACM*, vol. 40, no. 3, pp. 607–620, 1993.

[40] Y. Bengio, Y. LeCun. Scaling learning algorithms towards AI. *Large-Scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste, J. Weston, Eds., Cambridge, MA, USA: MIT Press, 2007.

[41] Y. Mansour. Learning Boolean functions *via* the Fourier transform. *Theoretical Advances in Neural Computation and Learning*, V. Roychowdhury, K. Y. Siu, A. Orlitsky, Eds., pp. 391–424, US: Springer, 1994.

[42] M. Anthony, P. Bartlett. *Neural Network Learning: Theoretical Foundations*, Cambridge, UK: Cambridge University Press, 2002.

[43] F. Anselmi, L. Rosasco, C. Tan, T. Poggio. Deep Convolutional Networks are Hierarchical Kernel Machines, Center for Brains, Minds and Machines (CBMM) Memo No. 035, The Center for Brains, Minds and Machines, USA, 2015.

[44] B. M. Lake, R. Salakhutdinov, J. B. Tenenabum. Human-level concept learning through probabilistic program induction. *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[45] A. Maurer. Bounds for linear multi-task learning. *Journal of Machine Learning Research*, vol. 7, no. 1, pp. 117–139, 2016.

[46] S. Soatto. Steps towards a theory of visual information: Active perception, signal-to-symbol conversion and the interplay between sensing and control. arXiv:1110.2053, 2011.

[47] T. A. Poggio, F. Anselmi. *Visual Cortex and Deep Networks: Learning Invariant Representations*, Cambridge, MA, UK: MIT Press, 2016.

[48] L. Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM Journal on Matrix Analysis and Applications*, no. 31, no. 4, pp. 2029–2054, 2010.

[49] S. Shalev-Shwartz, S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*, Cambridge, UK: Cambridge University Press, 2014.

[50] T. Poggio, W. Reichardt. On the representation of multi-input systems: Computational properties of polynomial algorithms. *Biological Cybernetics*, vol. 37, no. 3, 167-186, 1980.

[51] M. L. Minsky, S. A. Papert. *Perceptrons: An Introduction to Computational Geometry*, Cambridge MA, UK: The MIT Press, 1972.

Tomaso Poggio received the Ph. D. degree in theoretical physics from University of Genoa, Italy in 1971. He is the Eugene McDermott Professor at Department of Brain and Cognitive Sciences, the director of Center for Brains, Minds and Machines, the member of the Computer Science and Artificial Intelligence Laboratory at Massachusetts Institute of Technology (MIT), USA. Since 2000, he is a member of the faculty of the McGovern Institute for Brain Research. He was a Wissenschaftlicher assistant in Max Planck Institut für Biologische Kybernetik, Tüebingen, Germany from 1972 until 1981 when he became an associate professor at MIT. He is an honorary member of the Neuroscience Research Program, a member of the American Academy of Arts and Sciences and a Founding Fellow of AAAI. He received several awards such as the Otto-Hahn-Medaille Award of the Max-Planck-Society, the Max Planck Research Award (with M. Fahle), from the Alexander von Humboldt Foundation, the MIT 50K Entrepreneurship Competition Award, the Laurea Honoris Causa from the University of Pavia in 2000 (Volta Bicentennial), the 2003 Gabor Award, the 2009 Okawa prize, the American Association for the Advancement of Science (AAAS) Fellowship (2009) and the Swartz Prize for Theoretical and Computational Neuroscience in 2014. He is one of the most cited computational neuroscientists (with a h-index greater than 100-based on GoogleScholar).

E-mail: tp@ai.mit.edu (Corresponding author)
ORCID iD: 0000-0002-3944-0455

Hrushikesh Mhaskar did his undergraduate studies in Institute of Science, Nagpur, and received the first M. Sc. degree in mathematics from the Indian Institute of Technology, India in 1976. He received the Ph. D. degree in mathematics and M. Sc. degree in computer science from the Ohio State University, USA in 1980. He then joined Cal. State L. A., and was promoted to full professor in 1990. After retirement in 2012, he is now a visiting associate at California Institute of Technology, Research Professor at Claremont Graduate University, and occasionally served as a consultant for Qualcomm. He has published more than 135 refereed articles in the area of approximation theory, potential theory, neural networks, wavelet analysis, and data processing. His book *Weighted Polynomial Approximation* was published in 1997 by World Scientific, and the book with Dr. D. V. Pai, *Fundamentals of Approximation Theory* was published by Narosa Publishers, CRC, and Alpha Science in 2000. He serves on the editorial boards of *Journal of Approximation Theory, Applied and Computational Harmonic Analysis*, and *Jaen Journal of Approximation*. In addition, he was a co-editor of a special issue of "Advances in Computational Mathematics on Mathematical Aspects of Neural Networks", two volumes of *Journal of Approximation Theory*, dedicated to the memory of G. G. Lorentz, as well as two edited collections of research articles: *Wavelet Analysis and Applications*, Narosa Publishers, 2001, and *Frontiers in Interpolation and Approximation*, Chapman and Hall/CRC, 2006. He has held visiting positions, as well as given several invited lectures throughout North America, Europe, and Asia. He was awarded the Humboldt Fellowship for research in Germany four times. He was John von Neumann distinguished professor at Technical University of Munich in 2011. He is listed in Outstanding Young Men of America (1985) and Who′s Who in America′s Teachers (1994). His research was supported by the National Science Foundation and the U. S. Army Research Office, the Air Force Office of Scientific Research, the National Security Agency, and the Research and Development Laboratories.

E-mail: hmhaska@calstatela.edu

Lorenzo Rosasco received the Ph. D. degree from the University of Genova, Italy in 2006, where he worked under the supervision of Alessandro Verri and Ernesto De Vito in the Statistical Learning and Image Processing Research Unit (SLIPGURU). He is an assistant professor at the University of Genova, Italy. He is also affiliated with the Massachusetts Institute of Technology (MIT), USA, where is a visiting professor, and with the Istituto Italiano di Tecnologia (IIT), Italy where he is an external collaborator. He is leading the efforts to establish the Laboratory for Computational and Statistical Learning (LCSL), born from a collaborative agreement between IIT and MIT. During his Ph. D. degree period, he has been visiting student at the Toyota Technological Institute at Chicago, USA (working with Steve Smale) and at the Center for Biological and Computational Learning (CBCL) at MIT– working with Tomaso Poggio. Between 2006 and 2009, he was a postdoctoral fellow at CBCL working with Tomaso Poggio.

His research interests include theory and algorithms for machine learning. He has developed and analyzed methods to learn from small as well as large samples of high dimensional data, using analytical and probabilistic tools, within a multidisciplinary approach drawing concepts and techniques primarily from computer science but also from statistics, engineering and applied mathematics.

E-mail: lrosasco@mit.edu

Brando Miranda received the B. Sc. degree in electrical engineering and computer science (EECS) and the M. Eng. degree (supervised by Professor Tomaso Poggio) in machine learning from Massachusetts Institute of Technology (MIT), USA in 2014 and 2016, respectively.

His research interests include machine learning, statistics, neural networks, theories in deep learning and applied mathematics.

E-mail: brando90@mit.edu

Qianli Liao is a Ph. D. degree candidate in electrical engineering and computer science (EECS) at Massachusetts Institute of Technology (MIT), USA, supervised by Professor Tomaso Poggio.

His research interests include machine learning, optimization, neural networks, theoretical deep learning, computer vision, visual object/face recognition, biologically-plausible and brain-inspired learning.

E-mail: lql@mit.edu