BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 66, No. 6, 2018
DOI: 10.24425/bpas.2018.125924

INVITED PAPER

Theory I: Deep networks and the curse of dimensionality

T. POGGIO* and Q. LIAO

Center for Brains, Minds, and Machines, McGovern Institute for Brain Research,
Massachusetts Institute of Technology, Cambridge, MA, 02139

Abstract. We review recent work characterizing the classes of functions for which deep learning can be exponentially better than shallow learning. Deep convolutional networks are a special case of these conditions, though weight sharing is not the main reason for their exponential advantage.

Key words: deep and shallow networks, convolutional neural networks, function approximation, deep learning.

1. A theory of deep learning

1.1. Introduction[1]**.** There are three main 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: what is the landscape of the empirical risk? The third question is about generalization. Why there is no apparent overfitting despite overparametrization?

Here we focus on the first set of questions, reviewing previouswork [2–6]). 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 a subset of compositional functions, that is functions of functions, that we call hierarchically local compositional functions where all the constituent functions are local, in the sense that they have bounded, small dimensionality. The deep networks that can approximate them without the curse of dimensionality are of the deep convolutional type, but in general without weight sharing.

The most relevant implications of the above results 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 [7]) correspond to input-output mappings that are compositional with a hierarchy of constituent functions that are local: an example is $f(x_1, \cdots, x_8) = h_3\big(h_{21}\big(h_{11}(x_1, x_2), h_{12}(x_3, x_4)\big), h_{22}\big(h_{13}(x_5, x_6), h_{14}(x_7, x_8)\big)\big)$. 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 review [8]). Of course, multilayer convolutional networks have been around at least as far back as the optical processing era of the 1970s. The Neocognitron [9] 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 [10], that is a sequence of conjunctions and disjunctions. Several papers in the 1980s focused on the approximation power and learning properties of one-hidden layer networks (called shallow networks here). Very little appeared on multilayer networks, (but see [11–13]), mainly because one hidden layer nets performed empirically as well as deeper networks. On the theory side, a review by Pinkus in 1999 [14] concludes that "*...there seems to be reason to conjecture that the two hidden layer model may be significantly more promising than the single hidden layer model...*". A version of the questions about the importance of hierarchies was asked in [15] 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

[1] The material of this review is based on previous publications, in particular [1].

*e-mail: tp@csail.mit.edu

Manuscript

networks, which are equivalent to polynomial networks (see [16, 17]), are a simple case of a hierarchy that was analyzed [18] but did not provide particularly useful insights. Montufar and Bengio [19] showed that the number of linear regions that can be synthesized by a deep network with ReLU nonlinearities is much larger than by a shallow network. The meaning of this result in terms of approximation theory and of our own results is at the moment an open question[2]. Relevant to the present review is the work on hierarchical quadratic networks [17], together with function approximation results [14, 20]. Also relevant is the conjecture by Shashua (see [21]) on a connection between deep learning networks and the hierarchical Tucker representations of tensors. In fact, our theorems, that characterize the functions represented well by deep convolutional networks, may also lead to a characterization of the class of functions which can be represented well by a hierarchical Tucker representation.

It was already well known that the upper bound for the approximation of general functions by shallow networks is exponential. It is then 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 recently by Telgarsky [22] and by Shamirc [23]. We provide in Theorem 5 an older 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 $\varepsilon > 0$. 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, which is important for machine learning.

Let V_N be the be 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

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

For example, if $(f, V_N) = \mathcal{O}(N^{-\gamma})$ for some $\gamma > 0$, then a network with complexity $N = \mathcal{O}\left(\varepsilon^{-\frac{1}{\gamma}}\right)$ 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 a priori 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; in general is 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 the Figure. In particular each node in the network contains a certain number of units. A unit is a neuron which computes

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

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

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

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 Equation 3). Our approach and basic results apply rather directly to more complex networks (see third note in Section 6).

The logic of our theorems is as follows.

- Both shallow (a) and deep (b) networks are universal, that is they can approximate arbitrarily well any continuous

[2] We conjecture that the result may be similar to other examples in Section 4.2. It says that among the class of functions that are piecewise linear, there exist functions that can be synthesized by deep networks with a certain number of units but require a much large number of units to be synthesized by shallow networks.

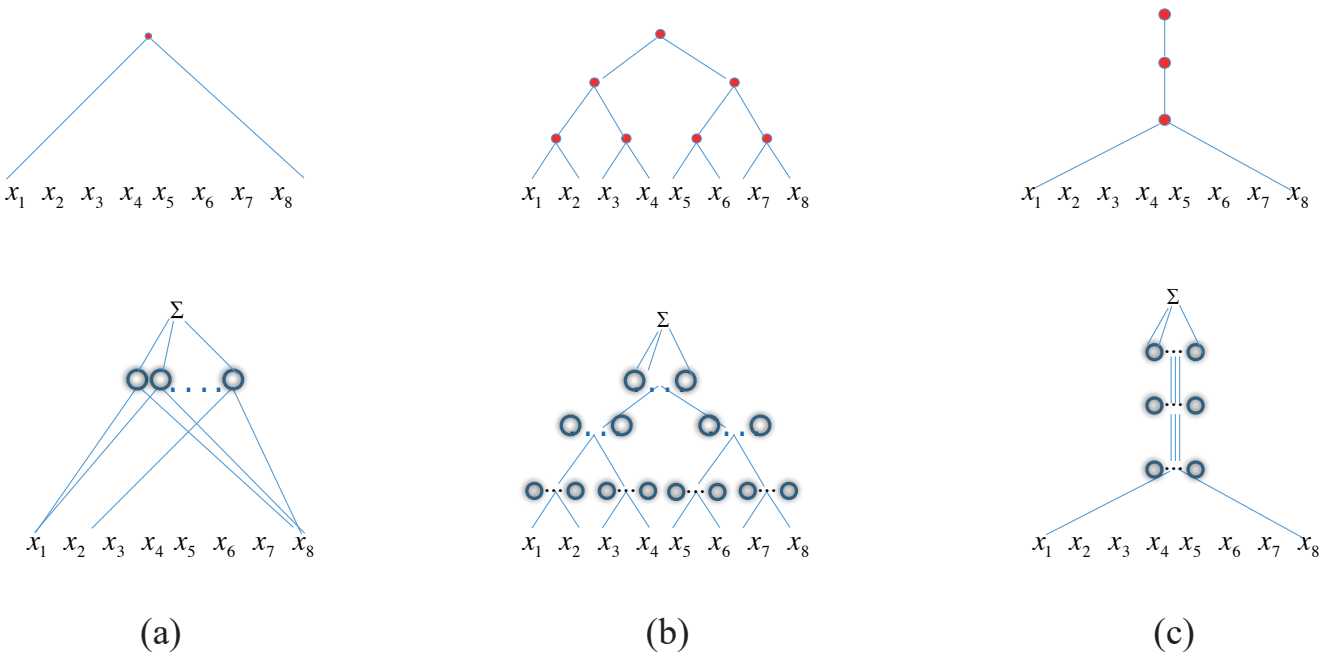

Fig. 1. The top graphs are associated to *functions*; each of the bottom diagrams depicts the ideal *network* approximating the function above. In (a) a shallow universal network in 8 variables and N units approximates a generic function of 8 variables $f(x_1, \cdots, x_8)$. Inset b) shows a binary tree hierarchical network at the bottom in $n = 8$ variables, which approximates well functions of the form $f(x_1, \cdots, x_8) = h_3\big(h_{21}\big(h_{11}(x_1, x_2), h_{12}(x_3, x_4)\big), h_{22}\big(h_{13}(x_5, x_6), h_{14}(x_7, x_8)\big)\big)$ 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 $\sum_{i=1}^{Q} a_i(\langle \mathbf{v}_i, \mathbf{x}\rangle + t_i)_+$, with $\mathbf{v}_i, \mathbf{x} \in \mathbb{R}^2$, $a_i, t_i \in \mathbb{R}$. Each term in the ridge function corresponds to a unit in the node (this is somewhat different from todays deep networks, but equivalent to them, see text and note in 6). 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, that is, 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 networks 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.). Inset 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 polynomials with a structure corresponding to the graph at the top

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 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.

- We consider a special class of functions of n variables on a compact domain that are a hierarchical compositions of local functions such as

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

The structure of the function in equation 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.

- 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 perspetive 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 Equation (4) with networks in which the activation nonlinearity is a smoothed version of the so called ReLU, given by $\sigma(x) = x_+ = max(0, x)$. The architecture of the deep networks reflects Equation (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 $\mathscr{S}_{N,n}$ 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 $w_k \in \mathbb{R}^n$, b_k, $a_k \in \mathbb{R}$. The number of trainable parameters here is $(n + 2)N \sim n$. 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 $\|f\| + \sum_{1 \leq |\mathbf{k}|_1 \leq m} \left\| D^{\mathbf{k}} f \right\| \leq 1$, where D^{k} denotes the partial derivative indicated by the multi-integer $\mathbf{k} \geq 1$, and $|\mathbf{k}|_1$ is the sum of the components of $\mathbf{k}$.

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 $\mathscr{D}_{N,2}$ to be the set of all deep networks with a binary tree architecture, where each of the constituent nodes is in $\mathscr{S}_{M,2}$, where $N = |V|M$, V being 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 $\mathscr{D}_{N,2}$ – that is, weights and biases, is $4N$.

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

- compositional functions of n variables are a subset of functions of n variables, that is $W_m^n \supseteq W_m^{n,2}$. 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.
- 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.

The following two theorems estimate the degree of approximation for shallow and deep networks.

3.3. Shallow networks. The first Theorem is about shallow networks.

Theorem 1. Let $\sigma : \mathbb{R} \to \mathbb{R}$ 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

$$N = \mathscr{O}\left(\varepsilon^{-n/m}\right) \textit{ and is the best possible}. \tag{5}$$

Notes. In [24, Theorem 2.1], the theorem is stated under the condition that σ is infinitely differentiable, and there exists $b \in \mathbb{R}$ such that $\sigma^{(k)}(b) \neq 0$ for any integer $k \geq 0$. It is proved in [25] that the second condition is equivalent to σ not being a polynomial. The proof in [25] 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 $\mathscr{O}(q^n)$ functions of the form $\mathbf{x} \mapsto \sigma(\langle w, x \rangle + b)$. 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 (cf. [5, 26]). The n-width results imply that the estimate in Theorem (1) is the best possible among all reasonable [26] 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 $\varepsilon^{-n/m}$ of parameters needed to obtain an accuracy $\mathscr{O}(\varepsilon)$ is known as the curse of dimensionality. Note that the constants involved in $\mathscr{O}$ 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. Then

Corollary 1. Let $\sigma : \mathbb{R} \to \mathbb{R}$ 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. Our second and main Theorem is about deep networks with smooth activations (preliminary versions appeared in [3–5]). 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.

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

$$N = \mathscr{O}\left((n-1)\varepsilon^{-2/m}\right). \tag{6}$$

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 $\mathscr{S}_{M,2}$ up to accuracy $\varepsilon = cN^{-m/2}$. 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, for example, 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 $\|h - P\| \leq \varepsilon$, $\|h_1 - P_1\| \leq \varepsilon$ and $\|h_2 - P_2\| \leq \varepsilon$, then $\|h(h_1, h_2) - P(P_1, P_2)\| = \|h(h_1, h_2) - h(P_1, P_2) + h(P_1, P_2) - P(P_1, P_2)\| \leq \|h(h_1, h_2) - h(P_1, P_2)\| + \|h(P_1, P_2) - P(P_1, P_2)\| \leq c\varepsilon$ by Minkowski inequality. Thus

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

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 $\sigma : \mathbb{R} \to \mathbb{R}$ 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. Empirical results suggest that the Theorem should be valid also for the non-smooth ReLU. Section 4.1 provides formal results. Stronger results than thevtheorems of this Section (see [6]) hold for networks where each unit evaluates a Gaussian non–linearity; i.e., Gaussian networks of the form

$$G(x) = \sum_{k=1}^{N} a_k \exp\left(-|x - w_k|^2\right), \quad x \in \mathbb{R}^d \tag{7}$$

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* anvaccuracy of ε, we need a shallow network with $\mathscr{O}(\varepsilon^{-n/m})$ trainable parameters. If we assume, however, a hierarchical structure on the target function as in Theorem 2, then the corresponding deep network yields a guaranteed accuracy of ε with $\mathscr{O}(\varepsilon^{-2/m})$ 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 $\varepsilon >$, 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 n/m, that of $W_m^{n,2}$ is $2/m$.

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. 1b 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 Fig. 1b to Fig. 1c and Fig. 2) by restating formally a few comments of previous papers. Let us begin with one of the previous examples. Consider

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

Since Q is nominally a polynomial of coordinatewise degree 2^{11}, [24, Lemma 3.2] 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, [24, Lemma 3.2] 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

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

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 $c\varepsilon^{-2}$ 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 $\mathscr{O}\left(\varepsilon^{-1}\right)$ 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 $\mathscr{G}$ be a directed acyclic graph (DAG), with the set of nodes V. A $\mathscr{G}$–function is defined as follows. Each of the source node obtains an input from $\mathbb{R}$. 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 $f : \mathbb{R}^n \mapsto \mathbb{R}$ be a compositional $\mathcal{G}$-function, where each of the constitutent function is in $W_{m_v}^{d_v}$. 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

$$N_s = \mathcal{O}\left(\varepsilon^{-\frac{n}{m}}\right), \tag{9}$$

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

$$N_d = \mathcal{O}\left(\sum_{v \in V} \varepsilon^{-d_v/m_v}\right). \tag{10}$$

Following definition 1 we call 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 $f : \mathbb{R}^n \mapsto \mathbb{R}$ of smoothness m has an effective dimension $< n/m$, then the functions are compositional in a manner consistent with the estimates in Theorem 3.

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

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

Corollary 3. Let $\sigma : R \to R$ be infinitely differentiable, and not a polynomial. With the set up as in Theorem 3, let f be DAG polynomial; i.e., a DAG function, each of whose constituent functions is a polynomial of degree k. Then f can be represented by a deep network with $\mathcal{O}\left(|V_N|k^d\right)$ units, where $|V_N|$ is the number of non-leaf vertices, and d is the maximal indegree of the nodes.

For example, if $\mathcal{G}$ is a full binary tree with 2^n leaves, then the nominal degree of the $\mathcal{G}$ polynomial as in Corollary 3 is k^{k^n}, and therefore requires a shallow network with $\mathcal{O}\left(nk^2\right)$ units, while a deep network requires only $\mathcal{O}\left(nk^2\right)$ units.

Notice that polynomials in S_k^n are *sparse* with a number of terms which is not exponential in n, that is it is not $O\left(k^n\right)$ but linear in n $\left(\text{that is } O(nk)\right)$ 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 [14], Proposition 3.7):

Proposition 2. Let $\sigma =: \mathbb{R} \to \mathbb{R}$ 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 $w_i, x \in \mathbb{R}^n$, $c_i, b_i \in \mathbb{R}$ 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 $\mathcal{B}$ the unit ball in $\mathbb{R}^n$. Call $C^m\left(B^n\right)$ 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 $C^m\left(B^n\right)$ with respect to the Sobolev norm p (see for details [14] page 168). We define the space $\mathcal{B}_p^m = \{f : f \in W_p^m, \|f\|_{m,p} \leq 1\}$ and the approximation error $E\left(B_2^m; H; L_2\right) = \inf_{g \in H} \|f - g\|_{L_2}$. It is shown in [14, Corollary 6.10] that

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

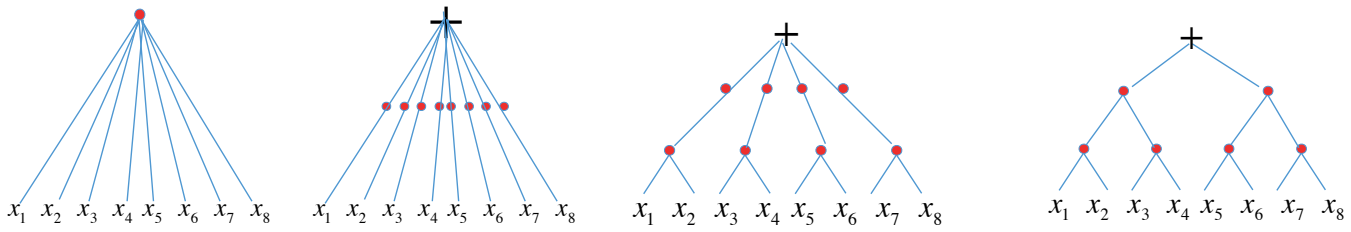

Fig. 2. The figure shows the graphs of functions that may have small effective dimensionality, depending on the number of units per node required for good approximation

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 [27], and more recently in [28] and [6] (see Theorem 3.1). In particular, using a result in [28] 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

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

wheres that of a deep compositional architecture is

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

The 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 the 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. An informal result that is more intuitive and may reflect what networks actually do is described in the Appendix of [1].

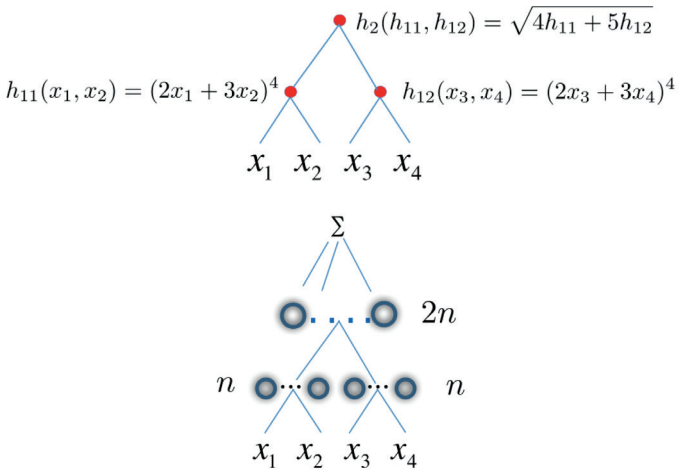

Fig. 3. The figure shows on the top the graph of the function to be approximated, while the bottom part of the figure shows a deep neural network with the same graph structure. The left and right node inf the first layer has each n units giving a total of $2n$ units in the first layer. The second layer 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

Figures 3–6 provide a sanity check and empirical support for our main results and for the claims in the introduction. Further details can be found in the original paper [1].

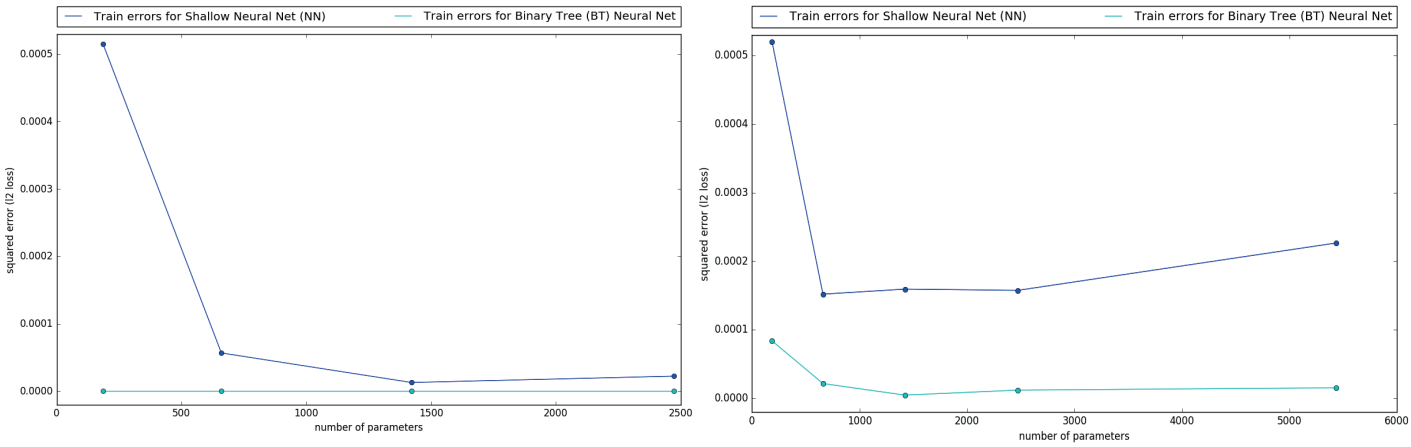

Fig. 4. An empirical comparison of shallow vs 2-layers binary tree networks in the approximation of compositional functions. The loss function is the standard mean square error (MSE). There are several units per node of the tree. In our setup here the network with an associated binary tree graph was 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 [29] optimizer. In order to get the best possible solution we ran 200 independent hyper parameter searches using random search [30] 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 [29]. The implementations were based on TensorFlow [31]

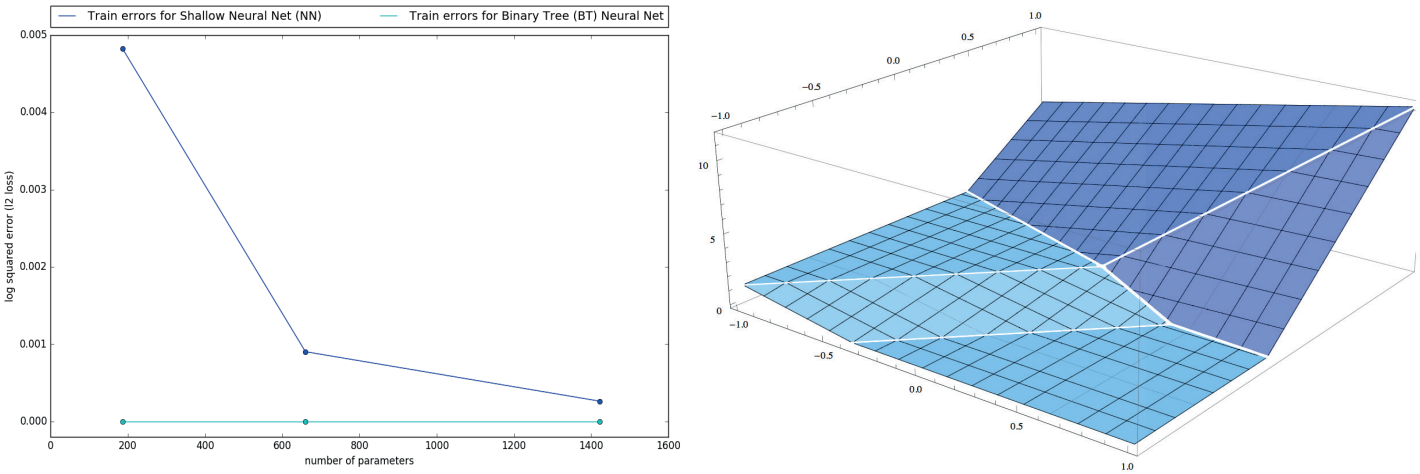

Fig. 5. Another comparison of shallow vs 2-layers binary tree networks in the learning of compositional functions. The set up of the experiment was the same as in the one in Fig 4 except that the compositional function had two ReLU units per node instead of only one. The right part of the figure shows 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 piecewise linear as it is always the case for ReLUs networks

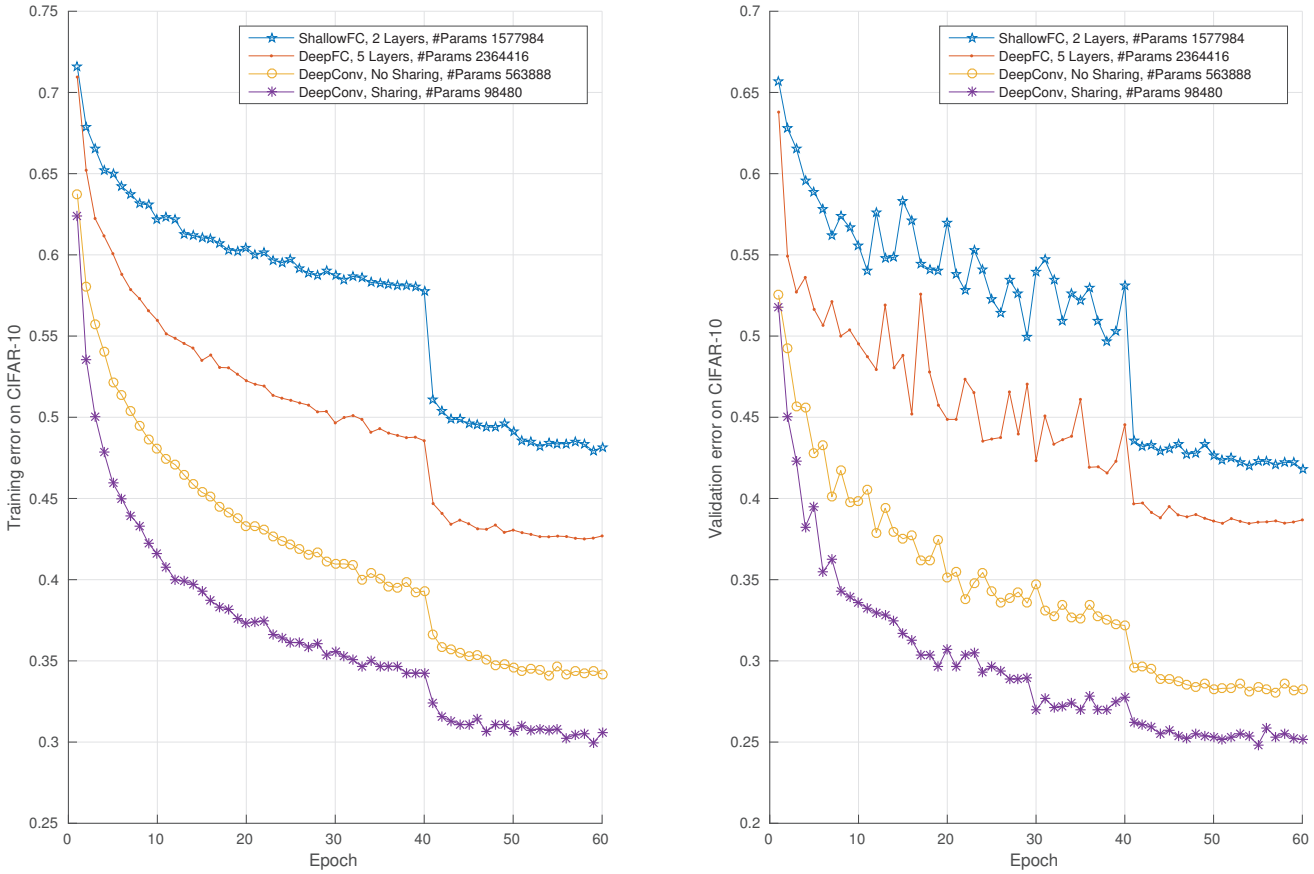

Fig. 6. We show that the main advantage of deep Convolutional Networks (ConvNets) comes from "hierarchical locality" instead of weight sharing. We train two 5-layer ConvNets with and without weight sharing on CIFAR-10. ConvNet without weight sharing has different filter parameters at each spatial location. 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

4.2. 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 [14]. 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:

- The polynomial considered earlier

$$Q(x_1, x_2, x_3, x_4) = \big(Q_1(Q_2(x_1, x_2), Q_3(x_3, x_4)\big)^{1024},$$

  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 $g\big(g\big(g(\ )\big)\big)$. Here, however, a formal proof of the impossibility of good approximation by shallow networks is not available. For a lower bound we need at least one example of a compositional function which cannot be approximated by shallow networks with a non-exponential degree of approximation.
- 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 [12, 13] 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 [32] show other cases with separations that are exponential in the input dimension.
- As we mentioned earlier, Telgarsky proves an exponential gap between certain functions produced by deep networks and their approximation by shallow networks. The Theorem [22] 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. His upper bound can be proved directly from our main theorem by considering a different function – the real-valued polynomial $x_1x_2 \ldots x_d$ defined on the cube $(-1, 1)^d$ which can be seen as a compositional function with a binary tree graph.
- We exhibit here another example to illustrate a limitation of shallow networks in approximating a compositional function. Let $n \geq 2$ be an integer, $B \subset \mathbb{R}^n$ be the unit ball of $\mathbb{R}^n$. We consider the class W of all compositional functions $f = f_2 \circ f_1$, where $f_1 : \mathbb{R}^n \to \mathbb{R}$, and $\sum_{|\mathbf{k}| \leq 4} \|D^{\mathbf{k}} f_1\|_\infty \leq 1, f_2 : \mathbb{R} \to \mathbb{R}$ and $\|D^4 f_2\|_\infty \leq 1$. We consider

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

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

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

$$\Delta(\mathscr{S}_N) \geq \left\lceil 2^{-N/(n-1)} \right\rceil, \tag{11}$$

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

$$\Delta(\mathscr{D}_N) \geq c_3 N^{-4/n}. \tag{12}$$

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. To prove (11), we use Lemma 3.2 in [13]. Let ϕ be a C^∞ function supported on $[0, 1]$, and we consider $f_N(x) = \phi\big(|4^N x|^2\big)$. We may clearly choose ϕ so that $\|f_N\|_\infty = 1$. Then it is clear that each $f_N \in W$. Clearly,

$$\Delta(\mathscr{S}_N) \geq \inf_{P \in \mathscr{S}_N} \max_{x \in B} \big|f_N(x) - P(x)\big|. \tag{13}$$

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

$$\begin{aligned} &\inf_{P \in \mathscr{S}_N} \max_{x \in B} \big|f_N(x) - P(x)\big| \geq \\ &\geq (1/2) \max_{x \in B} \big|f_N(x) - P^*(x)\big|. \end{aligned} \tag{14}$$

Since f_N is supported on $\big\{x \in \mathbb{R}^n : |x| \leq 4^{-N}\big\}$, we may imitate the proof of Lemma 3.2 in [13] with $g_k^*(t) = \sigma(t + b_k^*)$. Let $x_0 \in B$ be such that (without loss of generality) $f_N(x_0) = \max_{x \in B} |f_N(x)|$, and μ_0 be the Dirac measure supported at x_0. We group $\{w_k^*\}$ in $m = \lceil N/(n-1) \rceil$ disjoint groups of $n-1$ vectors each. For each group, we take vectors $\{v_\ell\}$ such that v_ℓ is orthogonal to the w_k^*'s in group ℓ. The argument in the proof of Lemma 3.2 in [13] can be modified to get a measure μ with total variation 2^m such that

$$\int_B f_N(x) d\mu(x) = \|f_N\|_\infty, \quad \int_B g_k^*(x) d\mu(x) = 0, \ \ k = 1, \cdots, N.$$

It is easy to deduce from here as in [13] using the duality principle that

$$\max_{x \in B} \big|f_N(x) - P^*(x)\big| \geq c 2^{-m}.$$

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 $\sigma\big(\langle w_k, x\rangle b_k\big)$, with $k = 1, \cdots, r(n)$, and w_k, b_k ranging over their whole domains. Such a result has been claimed for the case of smooth ReLUs [33].

4.3. 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. We have shown that for a given class of compositional functions characterized by an associated graph there exist 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; deep networks with a number of units that increases exponentially with layers can approximate any function.

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. Polynomial functions are indeed linear combinations of monomials which are compositional. However, their compositional structure does not confer any advantage in terms of approximation, because of the exponential number of compositional terms.

As we mentioned earlier, networks corresponding to graphs that include the graph of the function to be learned can exploit compositionality. The relevant number of parameters to be optimized, however, is the number of parameters r in the network and not the number of parameters $r*$ $(r* < r)$ of the optimal deep network with a graph exactly matched to the graph of the function to be learned. In Theory III we will show that overparametrization of deep networks for classification does not need to pay any overfitting price if the data sets are nice.

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 exhorbitant 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. 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 [34, 35] about the depth-breadth tradeoff in circuits design show that deep circuits are more efficient in representing certain Boolean functions than shallow circuits. Hastad 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 (see also [36]).

Notice that Hastad'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 [37] write "*We claim that most functions that can be represented compactly by deep architectures cannot be represented by a compact shallow architecture*". It is important however to observe that circuits composed of RELU's have different properties than Hastad circuits. In particular, the boolean parity function can be represented efficiently by a shallow circuit of RELUs.

Finally, we want to mention a few other observations on Boolean functions that suggest interesting connections with our results. 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 [38] that allow learning of certain Boolean function classes:

1. the low order algorithm that approximates functions by considering their low order Fourier coefficients and
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 6.

6. 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.

6.1. Compositionality, smoothness and curse of dimensionality. Properties of the compositionality type may have a more significant impact than smoothness properties in countering the curse of dimensionality in practical cases of learning and approximation.

6.2. Extension to vector nodes and functions. The main question that may be asked about the relevance of the theoretical re-

sults 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 (Equation 3). The following 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 previous theorems (of course the number of weigths is larger):

Corollary 4. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be infinitely differentiable, and not a polynomial. For a vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^q$ 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

$$N = \mathcal{O}\left(\varepsilon^{-n/m}\right). \tag{15}$$

The demonstration 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 the f_i. This argument of course does not mean that during learning this is what happens; it provides one way to perform 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 ouputs of one layer and inputs of the next one. This in turn leads to the following prediction: for consistent approximation accuracy across the layers, the rank of the weights matrices between units in successive layers should be in the order of the number of the dimensionality in the first layer (inputs and outputs have to be defined wrt support of the convolution kernel).

6.3. Invariances. Both shallow and deep representations may or may not reflect invariance to group transformations of the inputs of the function ([39, 40]). Invariance – also called weight sharing – decreases the complexity of the network. Since we are interested in the comparison of shallow vs 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 vs. 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 [41]. 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.

6.4. Neuroscience. 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.

6.5. Spline approximations and Boolean functions.

- Consider again the case of a multivariate function $f : [0, 1]^d \to \mathbb{R}$. 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 $f : \{0, 1\}^N \to \mathbb{R}$. Here, the assumption of compositionality corresponds to compressibility of a d-dimensional table in terms of a hierarchy of $d - 1$ 2-dimensional tables. Instead of n^d entries there are $(d - 1)n^2$ entries.
- Every function f can be approximated by an epsilon-close binary function f_B. Binarization of $f : \mathbb{R}^n \to \mathbb{R}$ is done by using k partitions for each variable x_i and indicator functions. Thus $f \mapsto f_B : \{0, 1\}^{kn} \to \mathbb{R}$ and $sup|f - f_B| \leq \varepsilon$, with ε depending on k and bounded Df.
- 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.

6.6. Tensors. One can think of tensors as d-dimensional tables representing or approximating a d-dimensional function. 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 (for instance see [21]). This work, as well more classical papers [42], does not characterize directly the class of functions for which these decompositions are effective. From recent discussions with Hrushikesh Mhaskar and Or Sharir, it seems clear that tensors with a HT decomposition of low complexity are a subset of our compositional functions. In particular, the following (informal) statement holds.

Proposition 4.

1. There exist functions which can be approximated well – that is, avoiding the curse of dimensionality – by deep nets, that do not have a low complexity HT decomposition;
2. all tensors with a low complexity HT decomposition can be approximated well by deep nets.

Notice that tensor decompositions assume that the sum of polynomial functions of order d is sparse (see eq. at top of page 2030 of [42]). Our results may provide a rigorous grounding for the tensor work related to deep learning.

6.7. Theory of computation, locality and compositionality:

- From the computer science point of view, feedforward multilayer networks are equivalent to finite state machines running for a finite number of time steps [43, 44]. 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.

In other words, all computable functions (by a Turing machine) are recursive, that is composed of a small set of primitive operations. In this broad sense all computable functions are compositional (composed from elementary functions). Conversely a Turing machine can be written as a compositional function $y = f^{(d)}(x, p)$ where $f : Z^n \times P^m \mapsto Z^h \times P^k$, P being parameters that are inputs and outputs of f. If t is bounded we have a finite state machine, otherwise a Turing machine, in terms of elementary functions. As mentioned above, each layer in a deep network correspond to one time step in a Turing machine. In a sense, this is sequential compositionality, as in the example of Fig. 1c.

- Hierarchically local compositionality can be related to the notion of *local connectivity* of a network. 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 ([45], see also [44]), where it was used for a very specific type of shallow networks. In this case many interesting visual computations have low order (e.g. recognition of isolated figures), since 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. For a deep networkthe situation is different: effective high order at the top can be achieved using units with low order. The network architecture of Fig. 1b 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).
- 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.
- The idea of functions that are compositions of "simpler" functions extends in a natural way to recurrent computations and recursive functions. For instance $h\left(f^{(t)}g\left((x)\right)\right)$ represents t iterations of the algorithm f (h and g match input and output dimensions to f).

7. Why are compositional functions so common?

Let us provide a couple of simple examples of compositional functions. Addition is compositional but the degree of approximation does not improve by decomposing addition in different layers of a 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. So compositionality is not enough: we need certain sublasses of compositional functions (such as the hierarchically local functions we described) in order to avoid the curse of dimensionality.

It is not clear, of course, why problems encountered in practice should involve this class of functions. Though we and others have argued that the explanation may be in either the physics or the neuroscience of the brain, these arguments [1] are not rigorous. Our conjecture at present is that compositionality is imposed by the wiring of our cortex and is reflected in language and the common problems we worry about. Thus compositionality of several – but not all – computations on images many reflect the way we describe and think about them. More details about these arguments as well as numerical experiments and other topics related to this paper can be found in the original contribution [1].

Acknowledgment. This work was supported in part by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF – 1231216 and in part by C-BRIC, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, the National Science Foundation, Intel Corporation, and the DoD Vannevar Bush Fellowship. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the DGX-1 used for this research.

References

[1] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, "Theory i: Why and when can deep networks avoid the curse of dimensionality?," tech. rep., MIT Center for Brains, Minds and Machines, 2016.

[2] F. Anselmi, L. Rosasco, C. Tan, and T. Poggio, "Deep convolutional network are hierarchical kernel machines," *Center for Brains, Minds and Machines (CBMM) Memo No. 35, also in arXiv*, 2015.

[3] T. Poggio, L. Rosasco, A. Shashua, N. Cohen, and F. Anselmi, "Notes on hierarchical splines, dclns and i-theory," tech. rep., MIT Computer Science and Artificial Intelligence Laboratory, 2015.

[4] T. Poggio, F. Anselmi, and L. Rosasco, "I-theory on depth vs width: hierarchical function composition," *CBMM memo 041*, 2015.

[5] H. Mhaskar, Q. Liao, and T. Poggio, "Learning real and boolean functions: When is deep better than shallow?," *Center for Brains, Minds and Machines (CBMM) Memo No. 45, also in arXiv*, 2016.

[6] H. Mhaskar and T. Poggio, "Deep versus shallow networks: an approximation theory perspective," *Center for Brains, Minds and Machines (CBMM) Memo No. 54, also in arXiv*, 2016.

[7] D. L. Donoho, "High-dimensional data analysis: The curses and blessings of dimensionality," in *AMS CONFERENCE ON MATH CHALLENGES OF THE 21ST CENTURY*, 2000.

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, pp. 436–444, 2015.

[9] K. Fukushima, "Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.
[10] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neuroscience*, vol. 2, pp. 1019–1025, Nov. 1999.
[11] H. Mhaskar, "Approximation properties of a multilayered feedforward artificial neural network," *Advances in Computational Mathematics*, pp. 61–80, 1993.
[12] C. Chui, X. Li, and H. Mhaskar, "Neural networks for localized approximation," *Mathematics of Computation*, vol. 63, no. 208, pp. 607–623, 1994.
[13] C. K. Chui, X. Li, and 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.
[14] A. Pinkus, "Approximation theory of the mlp model in neural networks," *Acta Numerica*, vol. 8, pp. 143–195, 1999.
[15] T. Poggio and S. Smale, "The mathematics of learning: Dealing with data," *Notices of the American Mathematical Society (AMS)*, vol. 50, no. 5, pp. 537–544, 2003.
[16] B.B. Moore and T. Poggio, "Representations properties of multilayer feedforward networks," *Abstracts of the First annual INNS meeting*, vol. 320, p. 502, 1998.
[17] R. Livni, S. Shalev-Shwartz, and O. Shamir, "A provably efficient algorithm for training deep networks," *CoRR*, vol. abs/1304.7045, 2013.
[18] O. Delalleau and Y. Bengio, "Shallow vs. deep sum-product networks," in *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12–14 December 2011, Granada, Spain.*, pp. 666–674, 2011.
[19] R. Montufar, G.F. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," *Advances in Neural Information Processing Systems*, vol. 27, pp. 2924–2932, 2014.
[20] H.N. Mhaskar, "Neural networks for localized approximation of real functions," in *Neural Networks for Processing [1993] III. Proceedings of the 1993 IEEE-SP Workshop*, pp. 190–196, IEEE, 1993.
[21] N. Cohen, O. Sharir, and A. Shashua, "On the expressive power of deep learning: a tensor analysis," *CoRR*, vol. abs/1509.0500, 2015.
[22] M. Telgarsky, "Representation benefits of deep feedforward networks," *arXiv preprint arXiv:1509.08101v2 [cs.LG] 29 Sep 2015*, 2015.
[23] I. Safran and O. Shamir, "Depth separation in relu networks for approximating smooth non-linear functions," *arXiv:1610.09887v1*, 2016.
[24] H.N. Mhaskar, "Neural networks for optimal approximation of smooth and analytic functions," *Neural Computation*, vol. 8, no. 1, pp. 164–177, 1996.
[25] E. Corominas and F.S. Balaguer, "Condiciones para que una funcion infinitamente derivable sea un polinomio," *Revista matemática hispanoamericana*, vol. 14, no. 1, pp. 26–43, 1954.
[26] R.A. DeVore, R. Howard, and C.A. Micchelli, "Optimal nonlinear approximation," *Manuscripta mathematica*, vol. 63, no. 4, pp. 469–478, 1989.
[27] H.N. Mhaskar, "On the tractability of multivariate integration and approximation by neural networks," *J. Complex.*, vol. 20, pp. 561–590, Aug. 2004.
[28] F. Bach, "Breaking the curse of dimensionality with convex neural networks," *arXiv:1412.8690*, 2014.
[29] D.P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.
[30] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.
[31] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. 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, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.
[32] R. Eldan and O. Shamir, "The power of depth for feedforward neural networks," *arXiv preprint arXiv:1512.03965v4*, 2016.
[33] M. Lin and H. Tegmark, "Why does deep and cheap learning work so well?," *arXiv:1608.08225*, pp. 1–14, 2016.
[34] J.T. Hastad, *Computational Limitations for Small Depth Circuits*. MIT Press, 1987.
[35] M. Furst, J. Saxe, and M. Sipser, "Parity, circuits, and the polynomial-time hierarchy," *Math. Systems Theory*, vol. 17, pp. 13–27, 1984.
[36] N. Linial, M. Y., and N. N., "Constant depth circuits, fourier transform, and learnability," *Journal of the ACM*, vol. 40, no. 3, p. 607–620, 1993.
[37] Y. Bengio and Y. LeCun, "Scaling learning algorithms towards ai," in *Large-Scale Kernel Machines* (L. Bottou, O. Chapelle, and J. DeCoste, D. Weston, eds.), MIT Press, 2007.
[38] Y. Mansour, "Learning boolean functions via the fourier transform," in *Theoretical Advances in Neural Computation and Learning* (V. Roychowdhury, K. Siu, and A. Orlitsky, eds.), pp. 391–424, Springer US, 1994.
[39] 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*, pp. 0–151, 2011.
[40] F. Anselmi, J.Z. Leibo, L. Rosasco, J. Mutch, A. Tacchetti, and T. Poggio, "Unsupervised learning of invariant representations," *Theoretical Computer Science*, 2015.
[41] F. Anselmi and T. Poggio, *Visual Cortex and Deep Networks*. MIT Press, 2016.
[42] L. Grasedyck, "Hierarchical Singular Value Decomposition of Tensors," *SIAM J. Matrix Anal. Appl.*, no. 31,4, pp. 2029–2054, 2010.
[43] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge eBooks, 2014.
[44] T. Poggio and W. Reichardt, "On the representation of multi-input systems: Computational properties of polynomial algorithms.," *Biological Cybernetics*, 37, 3, 167–186., 1980.
[45] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. Cambridge MA: The MIT Press, ISBN 0–262–63022–2, 1972.