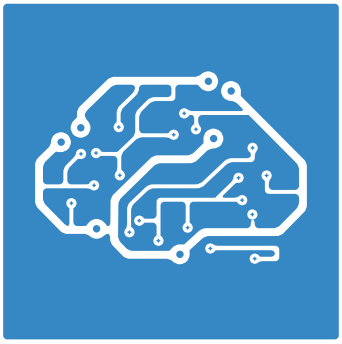

Center for Brains, Minds & Machines

CBMM Memo No. 45 **March 24, 2016**

Learning Real and Boolean Functions: When Is Deep Better Than Shallow

by

Hrushikesh Mhaskar[1], **Qianli Liao**[2], **Tomaso Poggio**[2]
[1] Department of Mathematics, California Institute of Technology, Pasadena,
Institute of Mathematical Sciences, Claremont Graduate University, CA, 91125
[2] Center for Brains, Minds, and Machines, McGovern Institute for Brain Research,
Massachusetts Institute of Technology, Cambridge, MA, 02139

Abstract: We describe computational tasks – especially in vision – that correspond to compositional/hierarchical functions. While the universal approximation property holds both for hierarchical and shallow networks, we prove that deep (hierarchical) networks can approximate the class of *compositional functions* with the same accuracy as shallow networks but with exponentially lower VC-dimension as well as the number of training parameters. This leads to the question of approximation by *sparse* polynomials (in the number of independent parameters) and, as a consequence, by deep networks. We also discuss connections between our results and learnability of sparse Boolean functions, settling an old conjecture by Bengio. [1]

This work was supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF 1231216. HNM was supported in part by ARO Grant W911NF-15-1-0385.

[1]This version (March 24th, 2016) is an update of the original version submitted on March 5th, 2016.

1 INTRODUCTION

The main goal of this paper is to begin to answer the question: why should deep networks be better than shallow networks? Our claim here is that hierarchical networks are a more efficient approximation of the computations that need to be performed on images – and possibly other sensory signals. The argument of the paper compares shallow (one-hidden layer) networks with deep networks (see for example Figure 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. The logic of the paper is as follows.

- Both shallow (a) and deep (b) networks are *universal*, that is they can approximate arbitrarily well any continuous function of d variables on a compact domain.
- We show that the approximation of functions with a *compositional structure* – such as $f(x_1, \cdots, x_d) = h_1(h_2 \cdots (h_j(h_{i1}(x_1, x_2), h_{i2}(x_3, x_4)), \cdots))$ – can be achieved with the same degree of accuracy by deep and shallow networks but that the VC-dimension and the fat-shattering dimension 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 makes the statement less than obvious. Our result makes clear that the intuition is indeed correct and provides quantitative bounds.
- Why are compositional functions important? We show that some basic visual recognition tasks do in fact require compositional functions. More in general, and less formally, it can be argued that symmetry properties of image statistics require hierarchies such as the bottom of Figure 1. In particular, we argue that hierarchical functions are effectively dictated by the statistics of natural images consisting of several overlapping objects, that is objects in clutter, rather than a single object against an homeogeneous background.
- Finally we discuss the relation between compositionality and sparsity. We sketch new results that lead to interesting connections between the learning of Boolean functions and the learning of functions of real variables.

2 PREVIOUS WORK

The success of Deep Learning in the present landscape of machine learning poses again an old theory question: why are multi-layer networks better than one-hidden-layer networks? Under which conditions? The question is relevant in several related fields from machine learning to function approximation and has appeared many times before.

A personal (TP) version of this question starts with an old paper on nonlinear associative memories which described under which conditions higher and higher degree operators should be learned from data to improve the performance of linear regression. The idea that compositionality is important in networks for learning and requires several layers in a network was the subject of a chapter in an early paper on (mostly RBF) networks for regularization(Poggio and Girosi, 1989). Most Deep Learning references these days start with Hinton's backpropagation and with Lecun's convolutional networks (see for a nice review (LeCun et al., 2015)). Of course, multilayer convolutional networks have been around at least as far back as the optical processing era of the 70s. Fukushima's Neocognitron(Fukushima, 1980) was a convolutional neural network that was trained to recognize characters. The HMAX model of visual cortex(Riesenhuber and Poggio, 1999a) was described as a series of AND and OR layers to represent hierarchies of disjunctions of conjunctions.

A version of the questions about why hierarchies was asked in (Poggio and Smale, 2003) as follow: *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. ...The theoretical issues surrounding hierarchical systems of this type are wide open, and likely to be of paramount importance for the next major development of efficient classifiers in several application domains...*

More specific, intriguing work (Montufar et al., 2014) provided an estimation of the number of linear regions that a network with ReLU nonlinearities can in principle synthesize but leaves open the question of whether they can be used for learning. Sum-Product networks, which are equivalent to polynomial networks (see (B. Moore and Poggio, 1998; Livni et al., 2013), are a simple case of a hierarchy that can be analyzed (Delalleau and Bengio, 2011). Work on hierarchical quadratic networks (Livni et al., 2013), together with function approximation results (Pinkus, 1999; Mhaskar, 1993b), is most relevant to the approach here. This paper is a short, updated version of material that appeared in (Poggio et al., 2015b) and especially in (Poggio et al., 2015a).

3 MAIN RESULTS

3.1 COMPOSITIONAL FUNCTIONS

It is natural to conjecture that hierarchical compositions of functions such as

$$f(x_1, \cdots, x_8) = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), \\ h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8))) \quad (1)$$

are approximated more efficiently by deep than by shallow networks.

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. Deep networks with standard architectures on the other hand *do represent* compositionality and can be adapted to the details of such prior information.

In addition, both shallow and deep representations may or may not reflect invariance to group transformations of the inputs of the function (Soatto, 2011; Anselmi et al., 2015). Invariance is expected to decrease the complexity of the network, for instance its VC-dimension. Since we are interested in the comparison of shallow vs deep architectures, here we consider the generic case of networks (and functions) for which invariance is not assumed.

We approximate functions of d variables of the form of Equation 1 functions with networks in which the activation nonlinearity is a so called ReLU, originally called *ramp* by Breiman and given by $\sigma(x) = |x|_+$ where $|x|_+ = max(0, x)$. The architecture of the deep networks reflects Equation 1 with each node h_i being a ridge function (in particular it may be an additive piecewise linear spline in the generalized sense of (Girosi et al., 1995), see Figure 1). As such, each node contains a certain number of units.

In the following we will estimate the number of units and layers required by shallow and hierarchical networks to approximate a function with a desired accuracy; we will then estimate the VC-dimension associated with the resulting network (assuming its output is used for binary classification). A direct connection between regression and binary classification is provided by the following observation due to (Livni et al., 2013). Combining Theorems 11.13 and 14.1 from (Anthony and Bartlett, 2002), it is possible to show that the fat-shattering dimension is upper-bounded by the VC-dimension of a slightly larger class of networks, which have an additional real input and an additional output node computing a linear threshold function in $\mathbb{R}^2$. Such a class of networks has a similar VC-dimension to the original class, hence the fat-shattering dimension can also be bounded.

3.2 VC BOUNDS

To compute the relevant VC bounds we use a well-known result (Anthony and Bartlett, 2002):

Theorem 1. *Any binary function class in a Euclidean space, which is parameterized by at most n parameters and can be computed by at most p floating point operations, has VC-dimension at most $O(np)$.*

This result provides immediately VC bounds for shallow and deep networks with d input variables. Examples for $d = 8$ are shown in Figure 1. We assume that the node in the shallow network consists of N units computing $\sum_{i=1}^{N} c_i \sigma(\langle \mathbf{w}_i, \mathbf{x} \rangle + b_i)$, with $\mathbf{w}_i, \mathbf{x} \in \mathbb{R}^d$ and $c_i, b_i \in \mathbb{R}$. Thus each unit has $d + 2$ parameters for a total of $O(dN)$ parameters. Each unit involves d multiplications, $d + 1$ additions and 1 rectification for a total of $O(dN)$ operations. Similarly each of the $d - 1$ nodes of the deep network consists of n units computing $\sum_{i=1}^{n} a_i |\langle \mathbf{v}_i, \mathbf{x} \rangle + t_i|$, with $\mathbf{v}_i, \mathbf{x} \in \mathbb{R}^2$, $a_i, t_i \in \mathbb{R}$. We assume that $d = 2^\ell$ where ℓ is the number of layers. Each unit has 4 parameters and requires ≤ 8 operations for a total of $4n(d-1)$ parameters and $8n(d-1)$ operations. With this notation we obtain

Proposition 1. *The VC-dimension of the shallow network with N units is $O(d^2N^2)$; the VC-dimension of the binary tree network with $n(d-1)$ units is bounded by $O(n^2d^2)$.*

It is *important to emphasize* here that state-of-art Deep Learning Neural Networks (DLNNs), with their small kernel size and many layers, are quite similar to the binary tree architecture (notice that the number of units decreases at higher and higher levels and that each unit receives only neighboring inputs from the layer below), independently of any additional (see remark in Figure 1) convolutional architecture. Visual cortex has a similar compositional architecture with receptive fields becoming larger and larger in higher and higher areas corresponding to layers here. An example is the old HMAX model (Riesenhuber and Poggio, 1999b) which took into account information about physiology of the ventral stream: it has an architecture of the binary tree type. Taking it all together, it is not suprising that DLNNs trained on Imagenet share properties with neurons at various stages of visual cortex: the results of this paper suggest that the reason is the very similar binary-tree-like architecture.

3.3 DEGREE OF APPROXIMATION

We now describe the complexity of the shallow and deep networks, measured by the number of trainable parameters required in order to obtain uniform approximation to a target function up to an accuracy of $\epsilon > 0$. Let $I^d = [-1, 1]^d$ be the unit cube in the Euclidean space $\mathbb{R}^d$, and $\|\cdot\|$ denote

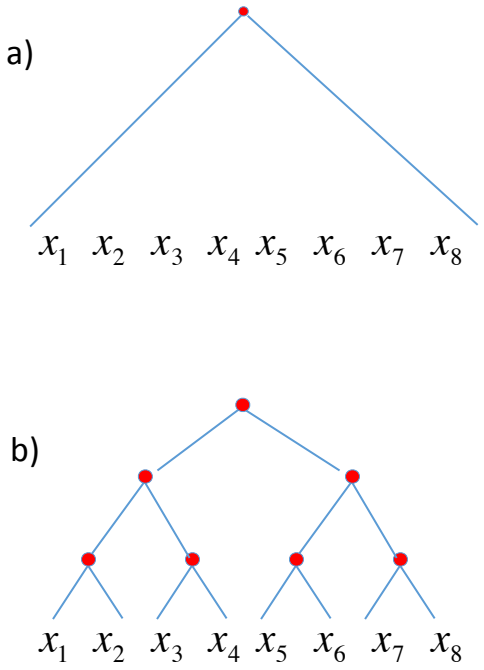

Figure 1: a) A shallow universal network in 8 variables and N units which can approximate a generic function $f(x_1, \cdots, x_8)$. b) A binary tree hierarchical network in 8 variables, which approximates well functions of the form $f(x_1, \cdots, x_8) = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8)))$. Each of the nodes in b) consists of n ReLU units and computes the ridge function (Pinkus, 1999) $\sum_{i=1}^{n} a_i |\langle \mathbf{v}_i, \mathbf{x} \rangle + t_i|_+$, with $\mathbf{v}_i, \mathbf{x} \in \mathbb{R}^2$, $a_i, t_i \in \mathbb{R}$. Each term corresponds to called a "channel'. Similar to the shallow network a hierarchical network as in b) can approximate a generic function; the text proves how it approximates a compositional functions better than a shallow network. No invariance is assumed here. Additional properties of position and scale invariance of the compositional function would imply $h_{11} = h_{12} = h_{13} = h_{14}$ and $h_{21} = h_{22}$ and $h_{11} \propto h_{21} \propto h_3$. These constraints (the first one corresponds to weight sharing in convolutional networks) further reduce the VC-dimension of the network. Notice that state-of-art DLNNs with their small kernel size and many layers are quite similar to the binary tree architecture, which is itself very similar to hierarchical models of visual cortex.

the uniform norm:

$$\|f\| = \max_{\mathbf{x} \in I^d} |f(\mathbf{x})|, \qquad f \in C(I^d).$$

If $V_n \subset C(I^d)$, where the parameter n indicates the complexity of V_n, the degree of approximation of $f \in C(I^d)$ from V is defined by

$$\mathsf{dist}(f, V_n) = \inf_{P \in V_n} \|f - P\|. \tag{2}$$

A central problem in approximation theory is to study the interdependence of the rate at which $\mathsf{dist}(f, V_n) \to 0$ as $n \to \infty$ and the "smoothness" of f. It is customary in approximation theory to codify this smoothness information in terms of membership in a compact set K of functions. The worst case error in approximating a target function given the **only** a priori information that $f \in K$ is therefore

$$\mathsf{worst}(V_n, K) = \sup\{\mathsf{dist}(f, V_n) : f \in K\}. \tag{3}$$

If $V_n \subseteq V_{n+1}$ for all n, and $\mathcal{V}$ is the union of V_n's, then the complexity of approximation from $\mathcal{V}$ of functions in K is given for $\epsilon > 0$ by

$$\mathsf{complexity}(\mathcal{V}, K, \epsilon) = \min\{n \geq 0 : \mathsf{worst}(V_n, K) \leq \epsilon\}. \tag{4}$$

Typically, the index n in the notation V_n is proportional to the number of trainable parameters in describing the elements of V_n. The quantity $\mathsf{complexity}(\mathcal{V}, K, \epsilon)$ thus represents the minimum number of parameters required to approximate an **arbitrary** function $f \in K$ by elements of $\mathcal{V}$ in order to **guarantee** that the error in approximation is $\leq \epsilon$.

Let $\sigma : \mathbb{R} \to \mathbb{R}$ be the activation function computed by each unit in the network. The number of trainable parameters in the set $\mathcal{S}_n$ of all shallow networks of the form

$$\sum_{i=1}^{n} a_i \sigma(\langle \mathbf{w}_i, \mathbf{x} \rangle + b_i) \tag{5}$$

is $(d + 2)n$, and $\mathsf{dist}(f, \mathcal{S}_n)$ is the best error than can be expected in approximating f by such networks. We define W to be the set of all $f : I^d \to \mathbb{R}$ which have a continuous gradient, such that $\|f\|_W = \|f\| + \sum_{k=1}^{d} \|D_k f\| \leq 1$. We are interested in estimating $\mathsf{complexity}(\mathcal{S}, W, \epsilon)$ (see (6) below).

The key idea here is that the closure of the space spanned by r linear combination of dilations and shifts of a univariate C^∞ function σ which is not a polynomial contains the space of univariate polynomials Π_{r-1} of degree $r - 1$ (see (Mhaskar, 1996, Proposition 2.2), (Pinkus, 1999, Corollary 3.6)). In fact, the following proposition is proved in (Mhaskar, 1996), (see also (Pinkus, 1999)):

Proposition 2. *Let $N \geq 1$, $d \geq 2$ be integers, and σ be a univariate C^∞ function which is not a polynomial. Then the closure of $\mathcal{S}_{N^d}$ contains the space of all polynomials in d variables with coordinatewise degree $< N$.*

Using this fact, it is observed in (Mhaskar, 1996, Theorem 2.1) that when σ is as in Proposition 2, the following estimate on the complexity of approximation by shallow networks $\mathcal{S} = \bigcup_{n=1}^{\infty} \mathcal{S}_n$ holds:

$$\mathsf{complexity}(\mathcal{V}, W, \epsilon) = \mathcal{O}(\epsilon^{-d}). \tag{6}$$

Is this the best? To investigate this question, let $M_n : K \to \mathbb{R}^n$ be a continuous mapping (parameter selection), and $A_n : \mathbb{R}^n \to C(I^d)$ be any mapping (recovery algorithm). Then an approximation to f is given by $A_n(M_n(f))$, where the continuity of M_n means that the selection of parameters is robust with respect to perturbations in f. Analogous to the way the complexity was defined in (4) using (3), one can define the nonlinear n–width of a compact set K by (cf. (DeVore et al., 1989))

$$d_n(K) = \inf_{M_n, A_n} \sup_{f \in K} \|f - A_n(M_n(f))\|, \tag{7}$$

and the *curse for* K by

$$\mathsf{curse}(K, \epsilon) = \min\{n \geq 1 : d_n(K) \leq \epsilon\}. \tag{8}$$

While the complexity in (4) depends upon the choice of approximants $\mathcal{V}$, the curse depends only on K, and represents the best that can be achieved by **any** continuous parameter selection and recovery processes. Neither of them addresses the question of how to accomplish the approximation. It is shown in (DeVore et al., 1989) that $\mathsf{curse}(W, \epsilon) \geq c\epsilon^{-d}$. So, the estimate (6) is the best possible among **all** reasonable methods of approximating arbitrary functions in W, although by itself, the estimate (6) is blind to the process by which the approximation is accomplished; in particular, this process is not required to be robust.

We note that both the curse and the complexity depends upon the norm in which the approximation is desired and the class K to which the target function in known to belong. In general, shallow networks do a better approximation, for example, with both the curse and complexity of the order ϵ^{-2}, independent of the input dimension d, if the class K consists of functions which are inherently shallow networks in some abstract sense, i.e., where the target function has the form (5), except that the sum expression is replaced by an integral with respect to some measure (e.g., (Barron, 1993; Kurková and Sanguineti, 2001; Kurková and Sanguineti, 2002; Mhaskar, 2004)).

For the hierarchical binary tree network, the analogue of these results is obtained by considering the compact set W_H to be the class of all functions f which have the same structure (e.g., (1)), where each of the constituent functions h are in W (applied with only 2 variables). We define the corresponding class of deep networks $\mathcal{D}_n$ to be set of all functions with the same structure, where each of the functions h is in $\mathcal{S}_n$, and $\mathcal{D} = \bigcup_{n=1}^{\infty} \mathcal{D}_n$. Since each of the constituent function is a function in W of two variables, (6) applied with $d = 2$ implies that each of these functions can be approximated from $\mathcal{S}_{\epsilon^{-2}}$ up to accuracy ϵ. Our assumption that $f \in W_H$ implies that there is a "good propagation" of the error in an approximation of f from $\mathcal{D}_{\epsilon^{-2}}$. Thus, for functions in W_H, we have

$$\mathsf{complexity}(\mathcal{D}, K, \epsilon) = \mathcal{O}(\epsilon^{-2}). \tag{9}$$

We note that the number of parameters involved in an element of $\mathcal{D}_{\epsilon^{-2}}$ is $\leq 6(d-1)\epsilon^{-2}$. Thus, (9) is a substantial improvement over (6) in terms of the number of trainable parameters required to achieve the accuracy ϵ, provided the target function has the hierarchical structure.

Remarks

- Given the estimates in this section, a very coarse bound on the VC-dimension of the binary classifier induced by the deep network in $\mathcal{D}_{\epsilon^{-2}}$ is $O(d^2\epsilon^{-4})$, whereas the VC-dimension of the shallow network in $\mathcal{S}_{\epsilon^{-2}}$ is $O(d^2\epsilon^{-2d})$.
- The above estimates can be easily extended from the special case of a binary tree to similar types of trees, and even directed acyclic graphs.
- The assumption $f \in W$ (respectively, $f \in W_H$) may be guaranteed in actual implementations by commonly used normalization operations on the weights of a deep network.
- The ReLU activation function satisfies in practice the assumptions of Proposition 2 (see Proposition 3.7 (Pinkus, 1999)).
- Since the bounds (9) and (6) apply under different a priori assumptions on the target functions, it is misleading to conclude that deep networks always yield better approximations in terms of the number of parameters involved for **every** target function. From the point of view of approximation theory as applied to arbitrary smooth functions, an interesting advantage of deep networks (with more than one hidden layer) is that they provide optimal local approximation analogous to spline approximation, while shallow networks are inherently unable to do so (Mhaskar, 1993a; Mhaskar, 1993b; Chui et al., 1994; Chui et al., 1996).
- The fact that hierarchical functions are approximated more efficiently by deep than by shallow networks has been shown to be true in an important special

case. For the hierarchical quadratic networks described in (Livni et al., 2013) (see section 4) there a VC-dimension bound is much lower than for the corresponding shallow network.

- In a sense *hierarchical deep networks can avoid the curse of dimensionality for compositional functions* with respect to shallow networks because each module in the hierarchy has bounded dimensionality, which is equal to 2 in the binary tree case. As we mention elsewhere, the VC-dimension can be further reduced by invariances such as translation invariance (corresponding to weight sharing).

4 WHY DOES VISION REQUIRE COMPOSITIONAL FUNCTIONS?

We saw that, though both deep hierarchies and shallow hierarchies are universal, deep hierarchies are approximated inefficiently by shallow ones. The final step in the argument is to show that deep hierarchical functions represent critical computations for vision.

The general point is that hierarchies – and weight sharing – reflect symmetries in the physical world that manifest themselves through the image statistics. Assume for instance that a computational hierarchy such as

$$h_l(\cdots h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), \\ h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8))\cdots))) \quad (10)$$

is given. Then shift invariance of the image statistics could be reflected in the following property: the local node "processors" should satisfy $h_{21} = h_{22}$ and $h_{11} = h_{12} = h_{13} = h_{14}$ since there is no reason for them to be different across images. In a similar way h_3 and h_{21} should be "scaled" versions of each other because of scale invariance. Similar invariances of image statistics – for instance to rotation – can be similarly use to constrain the local processes h.

It is natural to ask whether the hierarchy itself – for simplicity the idealized binary tree of the Figure 1– follows from a specific symmetry in the world and which one. A possible answer to this question follows from the fact that in natural images the target object is usually among several other objects at a range of scales and position, that is the target object is embedded in clutter. From the physical point of view, this is equivalent to the observation that there are several localized clusters of surfaces with similar properties (objects). These basic aspects of the physical world are reflected in properties of the statistics of images: *locality, shift invariance and scale invariance*. In particular, locality reflects clustering of similar surfaces in the world – the closer to each other pixels are in the image, the more likely they are to be correlated. Thus nearby patches are likely to be correlated (because of locality), and so are neighboring (because of shift invariance) image regions of increasing size (because of scale invariance). Ruderman's pioneering work (Ruderman, 1997) concludes that this set of properties is *equivalent to the statement that natural images consist of many object patches that may partly occlude each other* (object patches are image patches which have similar properties because they are induced by local groups of surfaces with similar properties). We argue that Ruderman's conclusion implies

- the property of selfsimilarity of image statistics, which in part reflects the compositionality of objects and parts: parts are themselves objects, that is selfsimilar clusters of similar surfaces in the physical world.
- the pervasive presence of clutter: in an image target objects are typically embedded among many objects at different scales and positions.

The first property – *compositionality* – was a main motivation for hierarchical architectures such as Fukushima's and later imitations of it such as HMAX which can be seen to be similar to a pyramid of AND and OR layers (Riesenhuber and Poggio, 1999b), that is a sequence of conjunctions and disjunctions. The presence of clutter, together with the need of position and scale invariance, implies that many visual computations, that is functions evaluated on images, should have a compositional structure. According to these arguments, compositional functions should be important for vision tasks. Notice that there is a subtle but important distinction between a) translation and scale invariance in object recognition and b) shift invariance and scale invariance of the statistics of natural images. Of course, the reason one wants shift and scale invariance in object recognition is because objects can appear at any position and scale in images, which is exactly why the statistics of images is position and scale invariant. In a related vein, it is possible to learn invariances, such as shift and scale invariance, from generic visual experience, because they are reflected in image statistics.

4.1 SANITY CHECK: RECOGNITION IN CLUTTER

The arguments of the previous section are suggestive. Here we provide a more formal argument which shows a significant advantage of hierarchical functions relative to shallow ones. The advantage derives from the locality property of objects and concerns an important computation: *recognition in clutter*. Figure 2 summarizes the skeleton of the argument: the recognition of object A suffers from interference from clutter (B) in a shallow network.

The point can be formalized by starting from an (obvious) result (Anselmi et al., 2015) showing that pooling over a

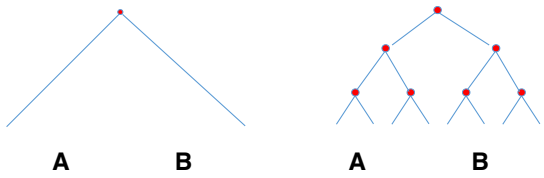

Figure 2: Two objects (A and B) are shown to a shallow (a) and a deep (b) network, both with invariance to translation obtained via convolution and pooling within each of the receptive fields. In the case of the deep network each of the two objects can be processed in the lower layers without interference by the other object. There is a key prediction here for the architecture of the ventral stream: there should be bypass connections – direct or indirect – from lower layers to the output layer to allow clutter-robust recognition at the cost of reduced invariance. This is not possible for the shallow network.

convolution is a group average of dot products of the image with transformations of templates by group elements. This group average is invariant and can be unique. By adapting the proof there (Anselmi et al., 2015), it is easy to show (as we are doing in a longer version of this paper), that invariance and potential uniquenss of pooling *do not hold when clutter* – defined as image patches that may change from one presentation of the object to the next presentation – *is present within the pooling regions*.

As shown in Figure 2, in the case of the deep network each of the two objects can be processed in the lower layers without interference from the other object. Thus *hierarchies, unlike shallow architectures, correspond to visual computations that are more robust to clutter.*

5 SPARSE FUNCTIONS

In this section we argue that the case of tree-like compositional functions is a special case of a more general formulation.

For the the intuition underlying our claim consider a node in the last layer of a deep network with two layers. The outputs of the layer are combined to provide the one-dimensional output of the network. The node in the last layer consists of m units, each evaluating an activation function σ as described in Section 3.3. The node receives inputs from the layer below which approximates an element of $\Pi_{k,d}$, the linear space of multivariate algebraic polynomials of coordinatewise degree at most $k-1$ in d variables. Suppose now that the node we are considering has the same inputs for each of its m units. The node then approximates a polynomial in $\Pi_{m,1}$, the linear space of (univariate) algebraic polynomials of degree at most $m-1$. The output of the node – which is the output of the network in our example – $\sum_{j=1}^{m} a_j \sigma\left(\left\langle \mathbf{v}_j, \sum_{i=1}^{k^d} c_i \sigma(\langle \mathbf{w}_i, \mathbf{x}\rangle + b_i)\right\rangle + t_j\right)$ thus approximates polynomials of coordinatewise degree $(k-1)(m-1)$ in $\mathbf{x}$ which are sparse in the sense that they may have much fewer independent coefficients than a generic polynomial in $\mathbf{x}$ of the same degree.

We have already implied in Section 3.3 that a hierarchical network can approximate a high degree polynomial P in the input variables $x_1, \cdots, x_d$, that can be written as a hierarchical composition of lower degree polynomials. For example, let

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

Since Q is nominally a polynomial of coordinatewise degree 2^{11}, Proposition 2 shows that a shallow network with $2^{11}+1$ units is able to approximate Q arbitrarily well on I^d. However, because of the hierarchical structure of Q, Proposition 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 similar, simpler example was tested using standard DLNN software and is shown in Figure 3.

Similar considerations apply to trigonometric polynomials and thus to Fourier approximations of functions. In this context, the lowest level layer can be regarded as approximating a trigonometric polynomial of a low degree, e.g., see (Mhaskar and Micchelli, 1995), and the layers after that can continue to be regarded as effectively approximating algebraic polynomials of high degree as above. In this way the network can be shown to approximate trigonometric polynomials containing high frequency terms. Notice that adding units to a shallow network with a single layer can also increase the highest frequency in the Fourier approximation but much more slowly than adding the same number of units to create an additional layer. The tradeoff is that now the Fourier terms that are contained in the function space approximated by the network are not anymore guaranteed to be a generic polynomial but just a sparse polynomial and have fewer trainable free parameters.

Definition 1. *We formally say that a polynomial (algebraic or trigonometric) is sparse if each of the coefficients in the polynomial with respect to some basis is a function of a small number of parameters.*

It is clear – and we will provide formal proofs in another paper – that this notion generalizes the hierarchical network

polynomials as well as the notion of sparsity in compressive sensing, where sparsity is defined in terms of the number of non–zero coefficients in some expansion.

With this definition we state results from previous sections in the following way.

Compositional functions can be represented by sparse algebraic or trigonometric polynomials; such sparse polynomials can be approximated by hierarchical networks with a matching architecture better than by shallow networks.

Following the discussion of this section and the example of Figure 3, we conjecture that general sparse functions, and not only compositional ones, may be approximated better by hierarchical networks. It will be interesting to explore the sense in which functions admitting sparse polynomial approximations may be *generic* (see section 4 for an analysis similar in this spirit). It is equally intriguing to speculate whether in practice (Poggio and Smale, 2003) *only sparse functions may be learnable*.

6 BOOLEAN FUNCTIONS

Our results sketched in the previous section are interesting not only in themselves but also because they suggest several connections to similar properties of Boolean functions. In fact our results seem to generalize properties already known for Boolean functions which are of course a special case of functions of real variables. We first recall some definitions followed by a few observations.

One of the most important and versatile tools for theoretical computer scientists for the study of functions of n Boolean variables, their related circuit design and several associated learning problems, is the Fourier transform over the Abelian group $\mathcal{Z}_2^n$. This is known as Fourier analysis over the Boolean cube $\{-1,1\}^n$. The Fourier expansion of a Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ or even a real-valued Boolean function $f : \{-1,1\}^n \to [-1,1]$ is its representation as a real polynomial, which is multilinear because of the Boolean nature of its variables. Thus for Boolean functions their Fourier representation is identical to their polynomial representation. In the following we will use the two terms interchangeably. Unlike functions of real variables, the full finite Fourier expansion is exact instead of an approximation and there is no need to distingush between trigonometric and real polynomials. Most of the properties of standard harmonic analysis are otherwise preserved, including Parseval theorem. The terms in the expansion correspond to the various monomials; the low order ones are parity functions over small subsets of the variables and correspond to low degrees and low frequencies in the case of polynomial and Fourier approximations, respectively, for functions of real variables.

Section 5 suggests the following approach to characterize which functions are best learned by which type of network – for instance shallow or deep. The structure of the 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 (Hastad, 1987) 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 (Linial et al., 1993)). These 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 (Bengio and LeCun, 2007) write *"We claim that most functions that can be represented compactly by deep architectures cannot be represented by a compact shallow architecture"."*. Until now this conjecture lacked a formal proof. It seems now that the results summarized in this paper and especially the ones announced in section 5 settle the issue, justifying the original conjecture and providing a general approach connecting results on Boolean functions with current real valued neural networks. Of course, we do not imply that the capacity of deep networks is exponentially larger than the capacity of shallow networks. As pointed out by Shalev-Shwartz, this is clearly not true, since the VC dimension of a network depends on the number of nodes and parameters and not on the depth. We remark that a nice theorem was recently published (Telgarsky, 2015), showing 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 is a special case of our results and corresponds to high-frequency, sparse trigonometric polynomials.

Finally, we want to speculate about a series of observations on Boolean functions that may show an interesting use of our approach using the approximating polynomials and networks for studying the learning of general functions. It is known that within Boolean functions the AC^0 class of

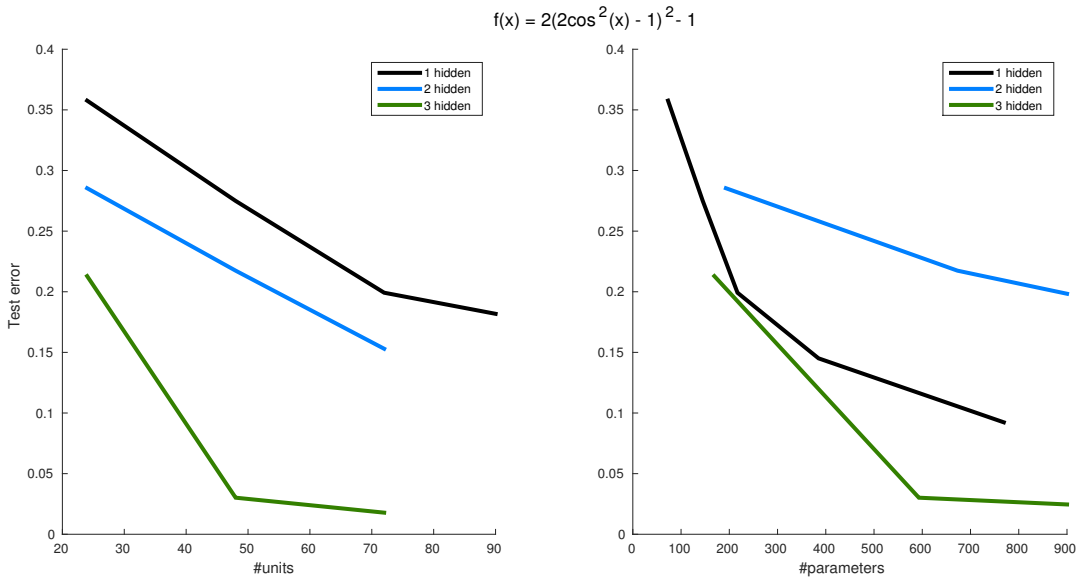

Figure 3: A sparse trigonometric function $f(x) = \cos^4 x$ (shown on the top of the figure) with one input variable is learned in a regression set-up using standard deep networks with 1, 2 or 3 hidden layers. In the 1 hidden layer setting, 24, 48, 72, 128 and 256 hidden units were tried. With 2 hidden layers, 12, 24 and 36 units per layer were tried. With 3 hidden layers, 8, 16 and 24 units per layer were tried. Each of the above settings was repeated 5 times, reporting the lowest test error. Mean squared error (MSE) was used as the objective function; the y axes in the above figures are the square root of the testing MSE. For the experiments with 2 and 3 hidden layers, batch normalization (Ioffe and Szegedy, 2015) was used between every two hidden layers. 60k training and 60k testing samples were drawn from a uniform distribution over $[-2\pi, 2\pi]$. The training process consisted of 2000 passes through the entire training data with mini batches of size 3000. Stochastic gradient descent with momentum 0.9 and learning rate 0.0001 was used. Implementations were based on MatConvNet (Vedaldi and Lenc, 2015). Same data points are plotted in 2 sub-figures with x axes being number of units and parameters, respectively. Note that with the input being 1-D, the number of parameters of a shallow network scales slowly with respect to the number of units, giving a shallow network some advantages in the right sub-figure. Although not shown here, the training errors are very similar to those of testing. The advantage of deep networks is expected to increase with increasing dimensionality of the function as implied by section Sparse Functions. As noted in the text, even in this simple case the solution found by SGD are almost certain to be suboptimal. Thus the figure cannot be taken as fully reflecting the theoretical results of this paper.

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. In general, two algorithms (Mansour, 1994) seems to 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 algorithm 1. Functions with small L_1 norm can be approximated well by algorithm 2. Boolean circuits expressing DNFs can be approximated by 1 but even better by 2. In fact, in many cases most of the coefficients of the low terms may still be negligeable and furthermore it may the case that 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 description we have in section 5. For general functions they may suggest the following. Many functions can be learned efficiently in terms of their low order coefficients and thus by shallow networks. This corresponds to using Tikhonov regularization that effectively cuts out high frequencies. Other functions must be learned in terms of their sparse coefficients by a deep network with an appropriate architecture. This is more similar to L_1 regularization. The sparsity approach which corresponds to deep networks includes the shallow Tikhonov approach and thus is more general and preferrable at least as long as computational and sample complexity issues are not taken into account.

7 DISCUSSION

There are two basic questions about Deep Neural Networks. The first question is about the power of the architecture – which classes of functions can it approximate how well? In this paper we focus on the approximation properties of a polynomial in the input vector $\mathbf{x}$ which is in the span of the network parameters. The second question, which we do not address here, is about learning the unknown coefficients from the data: do multiple solutions exist? How "many"? Why is SGD so unreasonably efficient, at least in appearance? Are good minima easier to find with deep than shallow networks? This last point is in practice very important. Notice that empirical tests like, Figure 3, typically mix answers to the two questions.

In this paper we address only the first question. We have introduced compositional functions and show that they are *sparse* in the sense that they can be approximated by sparse polynomials. Sparse polynomials do not need to be compositional in the specific sense of Figure 1 (consider the one-dimensional case). Deep networks can represent functions that have high frequencies in their *sparse* Fourier representation with fewer units than shallow networks (thus with lower VC-dimension). An unexpected outcome of our comparison of shallow vs. deep networks is that *sparsity of Fourier coefficients* is a a more general constraint for learning than Tikhonov-type smoothing priors. The latter is usually equivalent to *cutting high order Fourier coefficients*. The two priors can be regarded as two different implementations of sparsity, one more general than the other. Both reduce the number of terms, that is trainable parameters, in the approximating trigonometric polynomial – that is the number of Fourier coefficients.

In summary, we have derived new connections between a body of apparently disparate mathematical concepts that apply to learning of real-valued as well as of Boolean functions. They include compositionality, degree of approximating polynomials, sparsity, VC-dimension, Fourier analysis of Boolean functions and their role in determining when deep networks have a lower sample complexity than shallow networks. One of the conclusions is that standard Tikhonov regularization in RKHS should be replaced, when possible, by compositionality or sparsity via deep networks.

Acknowledgment

This work was supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF 1231216. HNM was supported in part by ARO Grant W911NF-15-1-0385.

References

Anselmi, F., Leibo, J. Z., Rosasco, L., Mutch, J., Tacchetti, A., and Poggio, T. (2015). Unsupervised learning of invariant representations. *Theoretical Computer Science*.

Anthony, M. and Bartlett, P. (2002). *Neural Network Learning - Theoretical Foundations*. Cambridge University Press.

B. Moore, B. and Poggio, T. (1998). Representations properties of multilayer feedforward networks. *Abstracts of the First annual INNS meeting*, 320:502.

Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *Information Theory, IEEE Transactions on*, 39(3):930–945.

Bengio, Y. and LeCun, Y. (2007). Scaling learning algorithms towards ai. In Bottou, L., Chapelle, O., and DeCoste, D.and Weston, J., editors, *Large-Scale Kernel Machines*. MIT Press.

Chui, C. K., Li, X., and Mhaskar, H. N. (1994). Neural networks for localized approximation. *Mathematics of Computation*, 63(208):607–623.

Chui, C. K., Li, X., and Mhaskar, H. N. (1996). Limitations of the approximation capabilities of neural networks with one hidden layer. *Advances in Computational Mathematics*, 5(1):233–243.

Delalleau, O. and Bengio, Y. (2011). 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.*, pages 666–674.

DeVore, R. A., Howard, R., and Micchelli, C. A. (1989). Optimal nonlinear approximation. *Manuscripta mathematica*, 63(4):469–478.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202.

Girosi, F., Jones, M., and Poggio, T. (1995). Regularization theory and neural networks architectures. *Neural Computation*, 7:219–269.

Hastad, J. T. (1987). *Computational Limitations for Small Depth Circuits*. MIT Press.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Kurková, V. and Sanguineti, M. (2001). Bounds on rates of variable basis and neural network approximation. *IEEE Transactions on Information Theory*, 47(6):2659–2665.

Kurková, V. and Sanguineti, M. (2002). Comparison of worst case errors in linear and neural network approximation. *IEEE Transactions on Information Theory*, 48(1):264–275.

LeCun, Y., Bengio, Y., and G., H. (2015). Deep learning. *Nature*, pages 436–444.

Linial, N., Y., M., and N., N. (1993). Constant depth circuits, fourier transform, and learnability. *Journal of the ACM*, 40(3):607620.

Livni, R., Shalev-Shwartz, S., and Shamir, O. (2013). A provably efficient algorithm for training deep networks. *CoRR*, abs/1304.7045.

Mansour, Y. (1994). Learning boolean functions via the fourier transform. In Roychowdhury, V., Siu, K., and Orlitsky, A., editors, *Theoretical Advances in Neural Computation and Learning*, pages 391–424. Springer US.

Mhaskar, H. N. (1993a). Approximation properties of a multilayered feedforward artificial neural network. *Advances in Computational Mathematics*, 1(1):61–80.

Mhaskar, H. N. (1993b). Neural networks for localized approximation of real functions. In *Neural Networks for Processing [1993] III. Proceedings of the 1993 IEEE-SP Workshop*, pages 190–196. IEEE.

Mhaskar, H. N. (1996). Neural networks for optimal approximation of smotth and analytic functions. *Neural Computation*, 8:164–177.

Mhaskar, H. N. (2004). On the tractability of multivariate integration and approximation by neural networks. *Journal of Complexity*, 20(4):561–590.

Mhaskar, H. N. and Micchelli, C. A. (1995). Degree of approximation by neural and translation networks with a single hidden layer. *Advances in Applied Mathematics*, 16(2):151–183.

Montufar, G. F.and Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. *Advances in Neural Information Processing Systems*, 27:2924–2932.

Pinkus, A. (1999). Approximation theory of the mlp model in neural networks. *Acta Numerica*, pages 143–195.

Poggio, T., Anselmi, F., and Rosasco, L. (2015a). I-theory on depth vs width: hierarchical function composition. *CBMM memo 041*.

Poggio, T. and Girosi, F. (1989). A theory of networks for approximation and learning. *Laboratory, Massachusetts Institute of Technology*, A.I. memo n1140.

Poggio, T., Rosaco, L., Shashua, A., Cohen, N., and Anselmi, F. (2015b). Notes on hierarchical splines, dclns and i-theory. *CBMM memo 037*.

Poggio, T. and Smale, S. (2003). The mathematics of learning: Dealing with data. *Notices of the American Mathematical Society (AMS)*, 50(5):537–544.

Riesenhuber, M. and Poggio, T. (1999a). Hierarchical models of object recognition in cortex. *Nat. Neurosci.*, 2(11):1019–1025.

Riesenhuber, M. and Poggio, T. (1999b). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025.

Ruderman, D. (1997). Origins of scaling in natural images. *Vision Res.*, pages 3385 – 3398.

Soatto, S. (2011). Steps Towards a Theory of Visual Information: Active Perception, Signal-to-Symbol Conversion and the Interplay Between Sensing and Control. *arXiv:1110.2053*, pages 0–151.

Telgarsky, M. (2015). Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101v2 [cs.LG] 29 Sep 2015*.

Vedaldi, A. and Lenc, K. (2015). Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*, pages 689–692. ACM.