CBMM Memo No. 041 December 29, 2015

I-theory on depth vs width: hierarchical function composition

by

Tomaso Poggio with Fabio Anselmi and Lorenzo Rosasco

Abstract

Deep learning networks with convolution, pooling and subsampling are a special case of hierarchical architectures, which can be represented by trees (such as binary trees). Hierarchical as well as shallow networks can approximate functions of several variables, in particular those that are compositions of low dimensional functions. We show that the power of a deep network architecture with respect to a shallow network is rather independent of the specific nonlinear operations in the network and depends instead on the the behavior of the VC-dimension. A shallow network can approximate compositional functions with the same error of a deep network but at the cost of a VC-dimension that is exponential instead than quadratic in the dimensionality of the function. To complete the argument we argue that there exist visual computations that are intrinsically compositional. In particular, we prove that recognition invariant to translation cannot be computed by shallow networks in the presence of clutter. Finally, a general framework that includes the compositional case is sketched. The key condition that allows tall, thin networks to be nicer that short, fat networks is that the target input-output function must be sparse in a certain technical sense.

This work was supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF - 1231216.

I-theory on depth vs width: hierarchical function composition

Tomaso Poggio with Fabio Anselmi and Lorenzo Rosasco

December 29, 2015

Abstract

Deep learning networks with convolution, pooling and subsampling are a special case of hierarchical architectures, which can be represented by trees (such as binary trees). Hierarchical as well as shallow networks can approximate functions of several variables, in particular those that are compositions of low dimensional functions. We show that the power of a deep network architecture with respect to a shallow network is rather independent of the specific nonlinear operations in the network and depends instead on the the behavior of the VC-dimension. A shallow network can approximate compositional functions with the same error of a deep network but at the cost of a VC-dimension that is exponential instead than quadratic in the dimensionality of the function. To complete the argument we argue that there exist visual computations that are intrinsically compositional. In particular, we prove that recognition invariant to translation cannot be computed by shallow networks in the presence of clutter. Finally, a general framework that includes the compositional case is sketched. The key condition that allows tall, thin networks to be nicer that short, fat networks is that the target input-output function must be sparse in a certain technical sense.

1 Introduction

We attempt to organize a few classic recent observations on approximation properties of hierarchical networks in a coherent framework that extend i-theory [1] to hierarchical architectures similar to Deep Learning Networks (DLN) and to supervised learning. In particular, we connect computational hierarchies to properties of image statistics ultimately dependent on the existence of objects – local clusters of surfaces with similar physical properties – and their distribution in images.

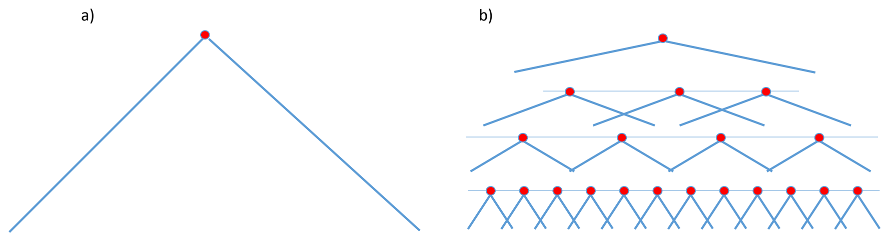

Figure 1: *a) A shallow universal network (here the $\bigwedge$ denotes a one-hidden layer network). The "image" is at the bottom. b) A deep hierarchical network.*

2 Summary of argument

The main goal of this paper is to answer the question: why deep networks instead of shallow networks (with just one-hidden layer)? The claim here is that hierarchical networks are a more efficient approximation of the computations that need to be performed on images. The argument of the paper consider the shallow and deep networks of Figure 1. Both types of networks use the same small set of operations – dot products, linear combinations, a fixed nonlinear function of one variable – combining the motifs of Figure 2 a,b,c. Each $\bigwedge$ combines over at least two inputs, usually more. The logic is as follows.

- Both shallow (a) and deep (b) networks of Figure 1 are *universal*, that is they can approximate arbitrarily well any continuous function of d variables on a bounded 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 of the deep networks is in general much smaller than the VC-dimension of the equivalent shallow network (the VC-dimension provides bounds for sample complexity). 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 shows that the VC dimension is the appropriate measure to consider.

- Why are compositional functions important? We discuss how some basic visual recognition tasks require compositional functions. More in general, and less formally, it can be argued that symmetry properties of image statistics require hierarchies such as Figure 1. In particular, hierarchical functions are effectively required 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. In addition, they are more tolerant to small deformations induced by the compositional structure of objects and parts.
- There are additional less fundamental, reasons for the outperformance of parts of current very deep networks (trained on residuals) that do not involve compositions of lower dimensional functions. We introduce and discuss the conjecture that, in analogy with quadratic networks, composition of ridge function approximations may have a higher rate of convergence that their additive combination.

3 Previous work

There are several papers addressing the question of why hierarchies. Needless to say none of them provides a full answer to the question. Interesting work that is not directly relevant for the results of this paper is [2] for an estimation of number of linear regions that a network can in principle produce and [3] for discussing Sum-Product networks, which are equivalent to the polynomial networks of [4] and [5]. The latter paper, together with the function approximation results reviewed in [6], are directly relevant to this paper. We will discuss with more details the previous work in an eventual journal version of this paper.

4 Network components

We consider in this paper both shallow and deep network architectures, as shown in Figure 1. As we mentioned, both types of networks use the same small set of operations – dot products, linear combinations, a fixed nonlinear function of one variable. We describe now in detail the operations implemented in our architectures and mention the differences with current DCLNs.
Standard nodes A node consists of a neuron which computes

$$|\langle x, t\rangle + b|_+, \tag{1}$$

where t is the vector of weights on the incoming input x. Both t and the real number b are parameters tuned by learning. Notice that the linear combination of r such nodes (at the next layer) $\sum_{i=1}^{r} c_i|(\langle x, t_i\rangle + b_i)|_+$ corresponds to ridge approximation which is universal.
A convolution node is a neuron that computes

$$|\langle x, g_i t\rangle + b|_+, \tag{2}$$

where $g_i t$ is the vector t transformed by $g_i \in G$.
Pooling according to i-theory corresponds to a node averaging several neurons according to

$$\sum_i |\langle x, g_i t\rangle + b_n|_+, n = 1, \cdots, N \tag{3}$$

A max-pooling node combines several convolution units according to

$$max_i |\langle x, g_i t\rangle + b|_+. \tag{4}$$

Notice that replacing the ramp nonlinearity – or the equivalent absolute value – with a square one obtains from the nodes above the corresponding nodes of a quadratic network [7].

Remark Combining several of the above units can give different types of nodes. For instance we may assume as the basic node a set of K neurons sharing the same weights t but with different bias terms b providing

$$\sum_{k=1}^{K} c_k |\langle x, t\rangle + b_k|_+. \tag{5}$$

A convolution node now computes $\sum_k c_k |(\langle x, g_i t\rangle + b_k)|_+$, whereas a max-pooling node computes $max_i \sum_k c_k |(\langle x, g_i t\rangle + b_k)|_+$.

Notice that replacing the ramp nonlinearity above, or the equivalent absolute value, with a square yields quadratic splines. The absolute value or the ramp give piecewise linear splines. In both cases the splines are free-knots splines. Sometime we assume "spline" nodes – for instance when we use Kolmogorov theorem in [7]. Of course in a multilayer network spline nodes can be synthesized from standard nodes by appropriate choices of the parameters and a sufficient number of neurons. In practice, however, this assumption may not be realizable by the learning procedure.

5 Why hierarchies

5.1 Depth is not needed for universality

We consider classic shallow and deep networks, see Figures 1, defined by a set of basic operations (*motifs*), see Figure 2.

We interpret the networks in Figure 1 as two ways of approximating a function of d variables. The shallow network approximates $f(x_1, \cdots, x_d)$ in one hidden layer with nodes of the type $\sum_k c_k h(\langle x, w^i\rangle + b_{k,i})$, where the h are nonlinear functions such as polynomial functions or linear rectifiers $h(z) = |z|_+$. The network on the right approximates a function such as $f(x_1, \cdots, x_8)$ using a linear combination of terms such as

$$h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{11}(x_5, x_6), h_{12}(x_7, x_8))) \tag{6}$$

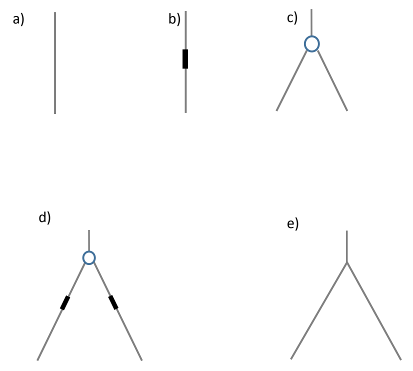

Figure 2: *Basic motifs of the DLNs networks described in the paper. We denote x as the input vector to a module; each component $x_{i,q}$ of the vector x represents the value of channel q at position i. Templates (called filters) are also vectors: the q-th template t^q is a vector with components over positions and channels. In pooling operations $g_i t_q$ denotes the transformation of template t^q (in the case of the translation group g_i shifts t_q to location i). The basic operation is the dot product $\langle x, gt^q \rangle$ involving spatial and channel components. a) A 1×1 "convolutional" module performs dot products $\langle x, gt^q \rangle$ between the input vector x and Q filter t^q; the scalar products are here with template vectors that are defined at a single spatial position. b) The dot product is followed by a nonlinearity (indicated by the black rectangle) which is typically a ramp. The overall operation performed is $|\langle x, t^q \rangle + b^q|_+, q = 1, \cdots, Q$ or $\sum_k c_k |\langle x, t^q \rangle + b^k|_+, q = 1, \cdots, Q$. c) A pooling operation denoted by the node (open circle) computes $\sum_g$ or max_g over its inputs in the pooling range (which includes at least two inputs as in the figure). These motifs can composed in various ways to create hierarchical networks that are variations of the binary tree of the Figure 1. For instance d) shows a typical max-pooling module performing $h_q(x) = max_{g \in G} |\langle x, gt^q \rangle + b^q|_+$. In the language of i-theory an open circle represents a set of complex cells pooling the outputs of the associated simple cells – typically many of them within each $\bigwedge$. Each simple cell computes a nonlinear transformation of the dot product between a patch of the image in the receptive field of the complex cell and a template. There are several different types of simple cells (different t^q per each complex cell). The $\bigwedge$ represents the receptive field of the complex cell – the part of the (neural) image visible to the module (for translations this is also the pooling range). Subsampling after pooling is usual. e) A "pooling" module with subsampling in which different input channels at different positions are combined linearly into a set of channels by dot products with weight vectors. In all the insets in this figure nodes consist of several neurons.*

where terms such as $h_{11}(x_1, x_2)$ correspond to the $\bigwedge$ pooling module (here we consider the extreme case of a module with just two inputs x_1 and x_2). Our main example of a compositional function is the binary tree of the Figure 1 but other types of tree can be analyzed in a very similar way. The main property of the two types of networks – shallow and deep – is that they can both approximate arbitrarily well continuous functions on a compact domain. The following result summarizes old work (see [6] for a review) and recent observations in [7].

Theorem 1. *Shallow, one-hidden layer networks with a nonlinearity which is not a polynomial are universal. Deep networks with a nonlinearity that could also be a polynomial are universal.*

Thus *universality* of the networks is *robust with respect to a broad spectrum of nonlinearities.*

Examples: *Examples of universal one-hidden layer network are Gaussian RBF or HBF [7], polynomial nonlinearities such as squares [5] and additive, piecewise linear splines [7]. The latter correspond to Deep Learning Networks with linear rectifiers.*

Approximation properties can be related to the notion of *connectivity* of a network. Connectivity is a key property in network computations. Local processing may be a key constraint also in neuroscience. One of the natural measures of connectivity that can be introduced is the *order* of a node defined as *the number of its distinct inputs.* The *order of a network is then the maximum order among its nodes.* The term order dates back to the Perceptron book ([8], see also [9]). From the previous observations on Figure 1 b), it follows that a *hierarchical network of order at least* 2 *can be universal.*

Remarks.

- The case of quadratic polynomial approximation considered in [4, 5] can be extended to the case of quadratic splines (spline interpolation is usually better than polynomial interpolation because it yields similar results to interpolating with higher degree polynomials while avoiding instability). I recall that linear piecewise splines have the form $\sum_j c_{i,j} |\langle x, w_i \rangle - b_i^j|$ whereas quadratic splines are $\sum_j c_{i,j} (\langle x, w_i \rangle - b_i^j)^2$. The output of a second layer still consists of hyperplanes in the case of linear splines and of quadratic surfaces in the case of quadratic splines.
- Each node of the network of Figure 1 can approximate a HBF unit with inputs x_1, x_2 using a fixed number of units, such as linear rectifiers. Similarly, a binary tree with additive spline nodes can approximate a universal one hidden layer Gaussian HBF. The same is true for the corresponding graphic notation of Figure 2.

- An attractive property from the point of view of neuroscience is the robustness of the approximation results with respect to the choice of nonlinearities (linear rectifiers, sigmoids, Gaussians etc.) and pooling (to yield either moments, probability density functions, or signatures that are equivalent to them).
- The Hierarchical Tucker (HT) factorization, which maps a tensor into a binary tree, describe function approximation by tensor products [10, 11].

5.1.1 Complexities of depth

Depth beyond one hidden layer is not required for universal approximation. Can then depth provide low complexity approximation of functions? A widespread belief is that good approximation in shallow networks requires a huge number of units while deep networks can achieve it with much fewer units. As a general statement, irrespectively of the specific elements and architecture of the networks, this is not true. The situation is as follows.

- One-hidden layer networks corresponding to additive linear spline approximate a function of d variables f on the unit cube as $f(x) = \sum_i^d h_i(x^i)$, where x^i is the i-th component of the vector x and $h_i(x^i) = \sum_j c_{ij}|x^i - b_i^j|$ are not universal (not even for general $h_i(x^i)$) [7].
- One-hidden layer networks corresponding to ridge approximations $f(x) = \sum_i^r h_i(\langle w^i, x\rangle)$ are universal approximators. Similarly to the case of polynomial approximation there is a lower bound in their degree of approximation: the number of units r increases with the inverse of the error, that is $r \geq C\epsilon^{-\frac{d-1}{m}}$, where m is related to the smoothness of the function to be approximated [6].
- Two-hidden layers networks can approximate any continuous function on a bounded domain *with any given error* ϵ using a fixed total number of units, that is $r = 8d + 4$; the nonlinearity σ, however, depends on ϵ (but not on f) and is very complex; more precisely the following theorem holds:

Theorem 2 (Theorem 7.1 in [6]). *There exists an activation function σ which is $\mathcal{C}^\infty$, strictly increasing with the property that, for any $f \in \mathcal{C}[0,1]^d$, $x \in [0,1]^d$, and $\varepsilon > 0$ there exists constants $d_i, c_{i,j}, w_{i,j}, b_{i,j}$ and γ_i such that:*

$$|f(x) - \sum_{i=1}^{4d+3} d_i \sigma(\sum_{j=1}^{2d+1} c_{i,j} |\sigma(\langle w^{i,j}, x\rangle + b_{i,j}) + \gamma_i|) \leq \varepsilon. \quad (7)$$

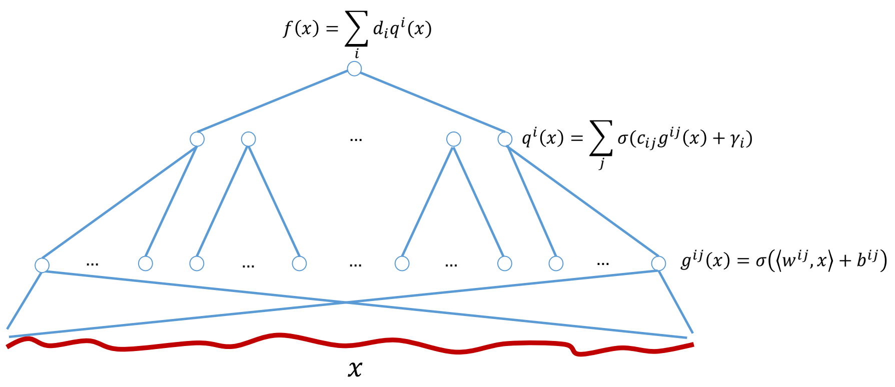
$f(x) = \sum_i d_i q^i(x)$
$q^i(x) = \sum_j \sigma(c_{ij} g^{ij}(x) + \gamma_i)$
$g^{ij}(x) = \sigma(\langle w^{ij}, x\rangle + b^{ij})$
x

Figure 3: *A network representing Pinkus' approximation based on Kolmogorov theorem.*

The corresponding network is shown in Figure 3. As remarked by Pinkus [6] the situation above suggests that the number of units is not a good measure of complexity. It seems natural that in terms of learning theory the VC-dimension (for binary output) is a more relevant quantity. The VC-dimension of multilayer networks can be characterized in terms of number of computations. A simple result is the following [12]: *any binary function class in a euclidean space, which is parameterized by at most n parameters and such that each function can be specified using at most t addition, multiplication, and comparison operations, has VC-dimension at most $O(nt)$.*

The above results suggests two observations, noting that a ramp function requires a simple comparison whereas Pinkus' σ requires a very large number of operations:

Conjecture 1. *The VC dimension of a network of units with Pinkus nonlinearity σ is growing to infinite with decreasing ϵ.*

Theorem 3. *The VC dimension of a network of r units, each with a ramp nonlinearity (linear rectifier) performing $|\langle x, w\rangle - b|$ with $d+1$ parameters (d is the dimensionality of w) has VC dimension $O(r^2(d+2)^2)$.*

Remark. As mentioned in [12] the characterization of the VC-dimension of a deep networks in terms of its operations has interesting consequences. If we consider a model of computing that only allows the standard arithmetic operations and comparisons with real numbers, then any class consisting of functions that are specified by a finite number of parameters can be computed in finite time on such a computer has finite VC-dimension. Furthermore, if we consider a sequence of function classes computed in this way, with increasing dimension d of the input domain ($X = R^d$), and with the number of parameters and the computation time growing only polynomially with d, then the VC-dimension also grows only polynomially. It follows that any class containing functions computable in time polynomial in the input dimension d has VC-dimension polynomial in d. This implies that the sample complexity grows only polynomially with the input dimension.

5.2 Deep networks can approximate compositional functions with lower VC-dimension than shallow ones

It is natural, but not obvious, to think that hierarchical compositions of functions such as

$$f(x_1, \cdots, x_8) = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8))) \tag{8}$$

are approximated more efficiently by deep than by shallow networks. We spell out here the rules of the game, in our comparison of shallow vs deep networks, even if they are obvious from the context. In general, we assume that the shallow networks do not have any structural information on the function to be learned (for instance its compositional structure). Deep networks on the other hand contain such prior information in their hierarchy. As it is natural to expect, this may lead to a decrease in the sample complexity.

Both shallow and deep representations can be invariant to group transformations of the input as shown in i-theory [1] and invariance is expected to decrease their complexity, that is the VC-dimension. Since we are interested in the comparison of shallow vs deep architectures, here we consider the generic case - no invariance is assumed (for simplicity).

The assumption that hierarchical functions are approximated more efficiently by deep than by shallow networks is known to be true in an important special case. For the hierarchical quadratic networks described in [5] (see section 4 there and [7]) a VC-dimension bound is much lower than for the corresponding shallow network. To fix ideas consider a function of d variables. Suppose that the function has the compositional structure of a binary tree. In this case $l = \log_2 d$ and the number v of the nodes $g($ that is $\bigwedge)$ is $v = d - 1$.

In the case of a binary tree as in Figure 4 with quadratic nonlinearities and $d - 1$ units, the VC-dimension according to Bartlett bound is $O(5(d-1)^2)$.

On the other hand, the VC-dimension of shallow network, with hidden layer units each corresponding to monomials, is $O(\binom{d+\Delta}{\Delta})$, where $\Delta = 2^l = d$ is the degree of the polynomial, d is the dimensionality of the input. The VC-dimension $O(\binom{2d}{d}) \approx \frac{4^d}{\sqrt{\pi d}}$ grows much more quickly with d. This suggests that a deep network can indeed generalize better than directly learning high-degree polynomials from a dictionary of all monomials. For the square loss, instead of binary classification, our estimate of the VC bound provides a similar upper bound for the V_γ dimension [12] .

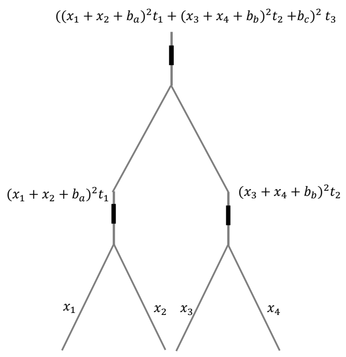
$((x_1 + x_2 + b_a)^2 t_1 + (x_3 + x_4 + b_b)^2 t_2 + b_c)^2\, t_3$
$(x_1 + x_2 + b_a)^2 t_1$
$(x_3 + x_4 + b_b)^2 t_2$
x_1
x_2
x_3
x_4

Figure 4: *A sum-of-squares binary tree network in four variables.*

We would like to show that similar results hold for networks in which the quadratic nonlinearity is replaced by the absolute value or, equivalently [7], by ramps. Consider again functions of d variables of the form of Equation 8 where the nodes h are ridge functions (in particular they may be additive piecewise linear splines). As in the quadratic case, let us assume that this is the class of functions that performs the desired computation.

To compute the associated VC bound the following lemma is useful.

Lemma 1. *A binary tree network composed of K $\bigwedge$ motifs (each computing $\sum^r g_i(\langle x, w^i\rangle)$ has total VC-dimension bounded by $K^2 V_{\bigwedge}$, where $V_{\bigwedge} = 2r^2$ is the VC-dimension of the binary $\bigwedge$ motif computed according to Bartlett's bound.*

Suppose we want to approximate the function computed by a deep network by using a shallow network. Its VC dimension will depend on the number of parameters and units in the network as $VC_{total} \geq (2r)(r)VC$. The number r depends on the error ϵ according to the following upper bound given in Theorem 6.2 in [6] [1]

$$\epsilon \leq C r^{-\frac{m}{d-1}} = C r^{-\frac{1}{d-1}} \tag{9}$$

Thus $r \geq (\frac{C}{\epsilon})^{d-1}$ which grows quickly with increasing dimension d and accuracy $(\frac{1}{\epsilon})$. We have thus shown

Theorem 4. *Deep networks can implement functions in a specific class of compositional functions with a much lower VC dimension (and therefore sample complexity) than a shallow network approximating the same functions. The VC*

[1]The bound is obtained taking $m = 1$ in Theorem 6.2 in [6], since $m = 1$ corresponds to piecewise linear splines).

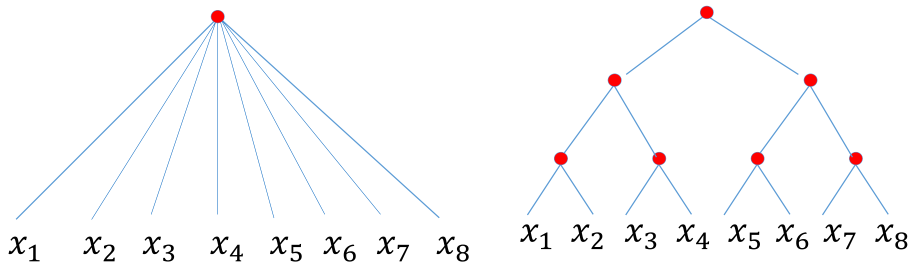
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8

Figure 5: *a) A shallow universal network in 8 variables. b) A binary tree hierarchical network in 8 variables. The first one can approximate a generic function $f(x_1, \cdots, x_8)$; the second represents functions of the specific compositional form $f(x_1, \cdots, x_8) = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8)))$. Since both representation can be invariant to group transformations of the input we consider the generic case - no invariance is assumed here - since we are interested in the comparison of shallow vs deep architecture. If additional properties of position and scale invariance of the probability distribution of images hold then 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 weigh sharing in DCLNNs) which are usually valid for vision further reduce substantially the VC-dimension of the network.*

dimension of the approximating shallow network grows as $O((\frac{1}{\epsilon})^{d-1})$ where ϵ is the desired approximation error and d is the dimensionality of the function domain.

The theorem reflects the fact that the units composing a hierarchical networks have a smaller number of inputs (two in the binary tree case) that the global units of the shallow network. We conjecture therefore that *the order (in the perceptron sense defined earlier) of a hierarchical network is the key parameter in determining its sample complexity*. As a consequence, increasing depth to achieve lower order may be a useful design principle.

Another way to compare shallow with deep networks is to assume that there is an unknown compositional function that performs the desired computation. It is then possible to learn from examples an empirical approximation of it it with a hierarchical network and compare the latter with the approximation obtained from a shallow network. We set the complexity of both network in such a way that they obtain the same degree of approximation. We also use the same nonlinear units (such as ramps) in both networks. Our goal here is to show that the VC dimension of the hierarchical network is usually much lower than the VC dimension of the "equivalent" shallow one. A sketch of the argument is as follows.

Proof sketch Consider again a function of d variables with the compositional structure of a binary tree. As before, $l = \log_2 d$ and the number v of the nodes (that is $\bigwedge$) is $v = d - 1$. In the example in the figure $d = 8$, $l = 3$, $v = 7$.

Let us now denote with B^d the unit ball in $\mathbb{R}^d$. We consider a specific case that applies to our case of piecewise linear splines approximation (the nonlinearity used – see h below – is in C^1). In particular we assume that f is defined on B^d and belongs to the Sobolev space $W^1_\infty(B^d)$ with $||f||_{1,\infty} = max_{0 \le k \le 1} ||D^k f||_\infty \le 1$. This implies that $||f||_{1,2} \le 1$. Under these assumptions, ridge approximants of the form

$$\sum_{i=1}^{d} h_i(\langle w^i, x \rangle), \tag{10}$$

with $h_i(z) = c_i |z - b_i|_+$, are dense in $L_\infty(B^n)$.

We want to estimate the number r needed to guarantee a maximum error E in approximating $f(x_1, \cdots, x_d)$ with the shallow network represented by Equation 10. Note that the shallow network contains r operations, r parameter vectors $w^i \in R^d$, and r offset parameters b_i

We use Theorem 6.2 in [6]. We consider the upper bound of the approximation error ϵ obtained by ridge functions of the form $\sum_{i=1}^{r} h_i(\langle w^i, x \rangle)$. The upper bound provides a minimum required number $r_{Shallow}$ as $\epsilon \le C r_{Shallow}^{-\frac{1}{d-1}}$, where C is independent of f and r.

Let us now consider the deep (binary tree) network. Our assumptions above guarantee that $||D^1 f||_2 \le 1$. Thus there is "good propagation" of the error in the compositional representation of f in terms of the h functions. In particular, the total L_2 error of the deep network with a binary tree structure can be estimated (because in our case $E^2 = \sum_j (D_j f)^2 (\Delta z)^2)$ as $(d-1)^{\frac{1}{2}} E_\bigwedge$, where $E_\bigwedge$ is the error of each of the $\bigwedge$ modules, the total number of which is $d - 1$.

Let us set $r_\bigwedge$ the number of units within each $\bigwedge$. Then using again Theorem 6.2 in [6] for each $\bigwedge$ module, one obtains $E_\bigwedge \le C(r_\bigwedge)^{-1}$ and thus $E_{deep} \le C(d-1)^{\frac{1}{2}} (r_\bigwedge)^{-1}$. Set $r_{Shallow}$ at a value sufficient to attain no more than the same error of the deep network yielding $C r_{Shallow}^{-\frac{1}{d-1}} \le C(d-1)^{\frac{1}{2}} (r_\bigwedge)^{-1}$. The equation finally yields

$$r_{Shallow} \ge \frac{(r_\bigwedge)^{d-1}}{(d-1)^{\frac{d-1}{2}}} \tag{11}$$

In the special case of $d = 2$ the shallow and the deep network achieve the same error. Otherwise the shallow network requires an number of units which increases exponentially with d *if* the number of units in each $\bigwedge$ module in the depp network is larger than $\sqrt{d-1}$. Notice that the number of units in each $\bigwedge$ in the deep network has to increase with d (e.g. with the depth of the network, since $d = 2^l$) in order to maintain the same overall error.

The previous lemma provides bounds on the VC-dimension of deep and shallow networks. A $\bigwedge$ unit in the binary tree has a bound of order $O(12)$, whereas the bound for a $\bigwedge$ unit in the shallow network is $O((d+2)^2)$.

Theorem 5. *Consider a function f on B^d belonging to the Sobolev space $W^1_\infty(B^d)$ with $||f||_{1,\infty} = max_{0\leq k\leq 1}||D^k f||_\infty \leq 1$. Suppose that*

- *the function f has a binary tree compositional structure in terms of lower dimensional functions $h(x_1, x_2)$ and*
- *a deep network with the same binary tree architecture has a sufficient number $r_\bigwedge$ of terms in each of its $d-1$ $\bigwedge$ nodes to approximate f within ϵ.*

Then

1. *a shallow network with at least the same approximation power needs at least $r_{Shallow} \geq \frac{(r_\bigwedge)^{d-1}}{(d-1)^{\frac{d-1}{2}}}$ units.*
2. *a bound on the VC-dimension of the binary classifier induced by the deep network is $O((d-1)^2 r^2_\bigwedge 12)$, whereas the VC-dimension of the shallow network is $O((d+2)^2 \frac{r_\bigwedge^{2(d-1)}}{(d-1)^{d-1}})$.*

Setting $r_\bigwedge \approx Kd$ shows that the VC-dimension of the shallow network increases exponentially (as $\approx d^d K^{2d}$) with the dimension d whereas the VC-dimension of the deep network increases only polynomially (as $\approx d^4 K^2$).
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 is further strongly reduced by invariances of the network such as translation invariance (corresponding to weight sharing).

Remarks

- The result above provides the following explanation of the observed convergence of deep networks (when they converge) with simultaneous decrease of the training *and* the test error. The reason is that the network architecture matches almost exactly the target function (error is close to zero) *and* it can be learned from data because of the relatively small sample complexity.
- Notice that the assumption $||f||_{1,\infty} \leq 1$ may be guaranteed by normalization operations or bounds on the weights of a deep network.
- The proposition above can be extended to any d-dimensional function $f(x_1, \cdots, x_d)$ in $W^m_p(B^d)$ satisfying $||f||_{m,p} \leq 1$. The proposition was proved for binary trees using linear splines but it is valid more in general.

- As mentioned in [5], combining Theorems 11.13 and 14.1 from [12], it is known that for a class of networks such as those we are learning, 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 our original class, hence we can effectively bound the fat-shattering dimension as well.

5.3 Why vision requires 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

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

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 5– follows from a specific symmetry in the world and which one. We suggest that the answer to this question follows from the fact that in natural images the target object is usually among several other objects at a range od 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)[2]

This 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 [13] concludes that

[2]Usually objects contain smaller clusters of similar surfaces in a selfsimilar fractal way. One of Mandelbrot examples described the coast of Britain as consisting of large features each one containing smaller similar features and so on. As a reminder, a self-similar object is similar to a part of itself (i.e. the whole is similar to one or more of the parts). Many objects in the real world, are statistically self-similar: parts of them show the same statistical properties at many scales.

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 HMAX which can be seen to be similar to a pyramid of AND and OR layers [14], that is a sequence of conjunctions and disjunctions. As we will see in the next section 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. Multilayer networks such as the binary tree of Figure 1 are the representations of compositional functions which lead to the lowest sample complexity. According to these intuitions, deep hierarchical networks should be important for vision tasks.

Remarks

- 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. The point is that one can have sets of patterns which do not have the statistics of natural images and process them in a scale and shift-invariant way. 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, using i-theory it is possible to learn invariances , such as shift and scale invariance, from visual experience because they are reflected in image statistics.

5.3.1 Formal sanity check: recognition in clutter

The arguments of the previous section are suggestive but unfortunately there is no fully equivalent formal result (as yet). Here I provide a formal argument, somewhat in the spirit of the recognition-in-context theorems of the *Perceptron* book. While they are still far from a full formalization of the previous sections, they shows a significant advantage of hierarchical architecture relative to shallow ones. The advantage derives from the locality property of objects and concerns an important computation: *recognition in clutter*. Figure 6 summarizes the skeleton of the argument: the recognition of object A suffers from interference from clutter (B) in the case of a shallow network. The proof is quite simple but we need to recall the definitions in [1].

Let $\mathcal{I} = \mathbb{R}^d$, and $\mathcal{P}(\mathcal{I})$ the space of probability measures on $\mathcal{I}$. In the following we assume $\mathcal{G}$ to be Abelian and compact and the corresponding Haar measure to be normalized. Note that, a group defines a natural equivalence relation $I \sim I'$, corresponding to point lying on the same *orbit*. The first step in our reasoning is the following definition.

Definition 1. *For all $I \in \mathcal{I}$, define the random variable*

$$Z_I : (\mathcal{G}, dg) \to \mathcal{I}, \quad Z_I(g) = gI, \quad \forall g \in \mathcal{G},$$

with law

$$\rho_I(A) = \int_{Z_I^{-1}(A)} dg,$$

for all measurable sets $A \subset \mathcal{I}$. Let

$$P : \mathcal{I} \to \mathcal{P}(\mathcal{I}), \quad P(I) = \rho_I, \quad \forall I \in \mathcal{I}.$$

The map P associates to each point a corresponding probability distribution. Then, we have the following result.

Theorem 6. *$P(I) \neq P(I')$ unless $I \sim I'$.*

The above result is a restatement of a result in [1]. The point is that, since having clutter in I' implies that $I \sim I'$ does not hold, then the theorem says that the distribution cannot be the same: therefore no invariance and no uniqueness. In other words, the proof of uniqueness and invariance of pooling (see [1]) *does 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 region.*

As shown in Figure 6, in the case of the deep network each of the two objects can be recognized in the lower layers without interference from the other object. Thus we have

Proposition 1. *Hierarchies allows for visual computations in the lower layers that are more robust to clutter than shallow architectures.*

5.3.2 Another sanity check: representing objects and parts

In addition to the issues of sample complexity and connectivity, compositional functions capture the hierarchical organization of the visual world where scenes are composed of objects which are themselves composed of parts. Objects can move in a scene relative to each other without changing their identity and often

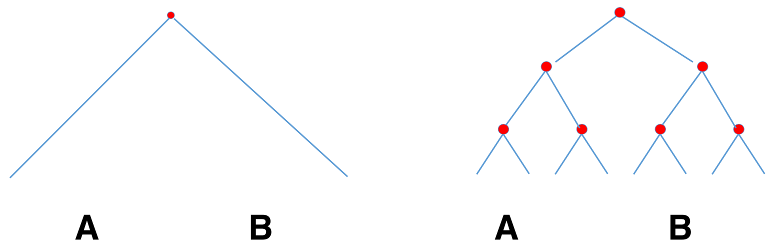

Figure 6: *Two objects (A and B) are shown to a shallow (a) and a deep (b) network. In the case of the deep network each of the two objects can be recognized in the lower layers without interference by the other object. This is not possible for the shallow network. There is a key prediction here for the architecture of the ventral stream: there should be bypass connections – direct or indirect – from lower layers to the output layer.*

changing the scene only in a minor way; the same is often true for parts within an object. In a hierarchical network global and local signatures from all levels of the hierarchy provide access memory and enable the categorization and identification of patches of the image corresponding to objects and their parts.

Each $\bigwedge$ module provides uniqueness and invariance to affine transformations within its pooling range. A deep network also provides invariance over affine transformations but in addition it is capable of providing invariance over non-global deformations. Consider for instance the two face drawings of Figure 7. The following holds:

Property *There exist local image deformations such as in Figure 7 to which a two layer hierarchical network can be invariant.*

Thus, in addition to the desired properties of invariance and selectivity, these architectures match the hierarchical structure of the visual world and the need to retrieve items from memory at various levels of size and complexity. The property of compositionality is related to the efficacy of hierarchical architectures vs. one layer architectures in dealing with the problem of partial occlusion and the more difficult problem of clutter in object recognition.

Figure 7: *Two line drawings of a face. The drawing on the right is a non affine transformation of the face on the left. A hierarchical network in which separate $\bigwedge$ modules "see" the two eyes may be invariant to the non-affine transformation.*

6 A general theoretical framework: depth vs width

A very recent paper indirectly suggests that our explanation of the role of depth, though correct, may not represent the full story *for current* DCNNs. In [15] an extremely deep architecture is improved, in terms of training and testing error, by adding layers which are not compositional and do not reduce dimensionality. In this network with bypass, identity connections, each layer attempts to fit the *error* between the output of the previous layers and the target. The elegant residual design is not important for the argument below; what is important is that provides stability to training of very deep architectures.

In any case, let us consider the residual network for concreteness. We draw an analogy with the quadratic networks of Figure 4 – which add layers that increase the degree of the polynomial represented by the network. Training on the residuals, as [15] do, would be equivalent to set the new higher degree polynomials to fit the residual of the set of previous layers. In the case of quadratic networks it is clear that additional layers increase the approximation power of the network by synthesizing higher degree polynomials. This *cannot* be achieved at all by increasing the number of units in the previous layer (it could be achieved by adding a polynomial of appropriate degree computed *ex novo* on the input x at a high cost in terms of units). Consider, on the other hand, ridge functions built on the absolute value nonlinearity (or the ramp): the approximating space of one layer can be expanded simply by adding units (see Proposition below). However, this may not be as effective as composing with an additional layer (e.g. as in [15]). The key observation is the following. Consider the approximation of a one-dimensional function by a network. The remark below about $\mathcal{N}_r$ says that $\mathcal{N}_{r+1}$ contains π_k, the linear space of univariate algebraic polynomials of degree at most r. The following statement (that we will prove in a paper in preparation) follows:

Theorem 7 (Depth-width Theorem)**.** *Consider adding depth to K versus adding width: $H \circ K$ has degree MK if H has degree M and K has degree N, instead of $M + K$.*

Of course a *generic* polynomial of degree N will require a similar order of magnitude in number of units for both shallow and deep implementations. If the function however is *p-sparse* in the sense of being well approximated by a sparse polynomial of high degree, then the deep implementation will use fewer units and have a lower VC-dimension bound.

A similar reasoning applies to functions of d variables. We recall that the number of monomials in a polynomial in d variables with total degree $\leq N$ is $\binom{d+N}{d}$ and can be written as a linear combination of the same number of terms of the form $(\langle w, x\rangle + b)^N$. Mhaskar ([16], see also [6]) proves that the space

$$\mathcal{M}_r(\phi) = \left\{ \sum_{i=1}^{r} c_i \phi(\langle w^i, x\rangle - b_i) \right\}, \tag{13}$$

with $r = N^d$ contains P_N the linear space of d-variate algebraic polynomials of degree at most N.

Similar results hold for spline approximation where a lemma of critical importance is Lemma 3.6 [17] which is trivially satisfied by $\phi(x) = (|x|_+)^m$: the activation function now commonly used is the basis for B-splines of order m. In practice this means that a tensor product B-spline can be implemented using a neural network with units arranged in multiple layers.

An examination of the proofs leads to the following conjecture (see forthcoming paper), stated here informally. Let us first give a definition: a function of d variables is p_ϵ-sparse if it can be approximated within ϵ by a polynomial of degree N which is not generic.

Conjecture 2. *We conjecture that functions that can be well approximated by a deep network with lower VC-dimension than by a shallow network must be p_ϵ-sparse .*

A simple case of the above property consists of polynomials corresponding to a tree where each of the Q nodes represents products and powers of terms of the form $(\langle w, x\rangle + b)$ with $Q \ll \binom{d+N}{d}$. A similar more extreme case are polynomials that are sparse in the number of monomials. Still another case is the function composition case studied in most of this paper.

Remarks

- The elegant approach of [15] is an interesting twist on the how to train a deep architecture. Usually the learning process optimizes the operators

K_i in

$$f(x) = K_n \circ \cdots \circ K_2 \circ K_1 x \tag{14}$$

which we write in a formal short-hand notation as (the various K are intended to be of the same type but with different parameters) as $f(x) = K^n x$.

An alternative is to learn the H_i in the equivalent network proposed by [15] where $K = (I + H)$

$$f(x) = (I + H)^n x = (I + H_n) \circ \cdots \circ (I + H_1)x. \tag{15}$$

In this network it is clear that layer H_n is an additive correction to the operator approximated by the previous $n-1$ layers; learning it corresponds to minimizing the residual $f(x) - K^{n-1}x$.

- As described in section 4 node j in a network may compute

$$f_j(x) = \sum_{k=1}^{K} c_{j,k} | \langle x, t_j \rangle + b_{j,k} |_+ . \tag{16}$$

whereas a standard neuron in DCNNs computes the above with $K = 1$ that is

$$f_j(x) = c_j | \langle x, t_j \rangle + b_j |_+ . \tag{17}$$

As we mentioned, it is illuminating to consider the two cases when the nonlinearity is square. The two situations are: *the polynomial case*

$$f^j(x) = c^j | \langle t_i, x \rangle |^2, \tag{18}$$

and the *spline case*

$$f_j(x) = \sum_{k=1}^{K} c_{j,k} | \langle x, t_j \rangle + b_{j,k} |^2 \tag{19}$$

Typically splines have better numerical properties than polynomials, but they *both have the same rate of approximation.*

- Properties of the iteration scheme of [15] may be characterized following the arguments of Theorem 4.5 in [18]. Similar properties are likely to hold.

- Notice that the set of shifted and dilated ridge functions has the following property. Consider for $c_i, b_i, \lambda_i \in \mathbb{R}$ the space of univariate functions

$$\mathcal{N}_r(\phi) = \left\{ \sum_{i=1}^{r} c_i \phi(\lambda_i x - b_i) \right\}. \tag{20}$$

  The following (Proposition 3.6 in [6]) holds

  Proposition 2. *If ϕ is not a polynomial, the closure of $\mathcal{N}$ contains the linear space of algebraic polynomial of degree at most $r - 1$.*

  Notice that $\mathcal{N}_r + \mathcal{N}_s = \mathcal{N}_{r+s}$.

- Interesting properties, that we will discuss in a sequel to this paper, include the following ones:
  - Mhaskar's "local approximation" in deep vs shallow networks;
  - Mhaskar's suggestion that neural networks may combine advantages of spline and polynomial approximation;
  - the "supervised PCA" stage between layers is similar to the (unsupervised) PCA step performed within the HT tensor decomposition and is similar to a suggestion by Shalev-Schwartz for his quadratic networks;
  - the network may not mimic the exact hierarchy or sparsity of the input-output mapping as long as its architecture allows the learning process to converge to it by finding appropriate weights;
  - p-sparsity may be generic but it is not clear under which assumptions.

7 Remarks on computational symmetries, image statistics, physics, hierarchies

- Networks that with architecture matching the compositional structure of a specific computation, are optimal ("the power is the hierarchy."). Therefore one expects that somewhat different types of trees may be required for different tasks.

- It is the combination of localization, scale invariance and shift invariance –rather than any one of the three symmetries – which is key in implying convolutional-type hierarchies

- In the *Perceptron* book many interesting visual computations have low order (e.g. recognition of isolated figures). The message is that they can be implemented in a single layer by units that have a small number of inputs. More complex visual computations require inputs from the full visual field. A hierarchical network can achieve effective high order at the top using units with low order.

- The network architecture of Figure 5 b) has low order: each node in the intermediate layers is connected to just 2 other nodes, rather than (say) all nodes in the previous layer (notice that the connections in the trees of the figures may reflect linear combinations of the input units).

- 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 the ventral stream.

- On the other hand, a low-order hierarchy may be the computational consequence of the compositionality of the visual world and the related effectiveness of learning first common parts or features and more complex objects later in the hierarchy. This is particulary important when a deep network is trained on a large multiclass problem: since simple parts are likely to be common across many different classes the first layers of the network can leverage a large set of examples independently of the label.

- What is the relation of the arguments in this note with i-theory? This note provide a logic and formal results to explain why hierarchical networks are better than shallow ones for visual computations. The original core of i-theory describes how pooling can provide either shallow or deep networks with invariance and selectivity properties. Such properties are embedded in the $\bigwedge$ modules of the theory (which usually have many inputs). Older i-theory papers discuss the question of why hierarchies along lines qualitatively similar to this paper. Our results here, shown for the case of $\bigwedge$ with two inputs, are equally valid for the usual case of $\bigwedge$ with many inputs and invariance arising from pooling.

- Theorem 4 is closely related to the compression argument of [7], see Appendices.

- There is a distinction in learning templates *and their transformations* on one hand and on the other hand learning the other parameters. The issue is that the weight vectors $w_i = g_i t$ and $w_j = g_j t$, corresponding to two transformations of the same template within the same $\bigwedge$, can be updated but only under the constraint $w_i = g_h w_j$ (for some h). This is what "weight sharing" in DLNs means (when g is an element of the translation group). It seems difficult to extend the "weight sharing" idea to situations in which the group is not the translation group or, even worse, is not known. In particular, as discussed in [7], a biological learning rule that enforces weight-sharing across the visual field seems implausible. In addition to the two possibilities we listed there we want to suggest a third more attractive conjecture: *images that change continuously drive unsupervised learning of the simple cells receptive fields; all other parameters are learned only in a supervised way, for instance by SGD.* In this way, the simple cells weights would be automatically shared; they would however be independent of supervision in learning.

- *A model of universal neural network computation.* In the traditional "digital" model of computation, universality means the capability of running any program that is computable by a Turing machine. Concepts such as polynomial vs non-polynomial complexity emerge from this point of view. For neural networks the model of computations that seems to take form now is closely related to function approximation. Here universality may be defined in terms of universal approximation of a function of d variables (the inputs). The appropriate definitions of complexity are the classical ones for Machine Learning (VC dimension and related quantities). An analysis of the complexity of the specific class of hierarchical functions of d variables suggests that *order* and *depth* of its tree representation play a key role in determining its VC dimension. Notice in particular that the VC dimension of a network corresponding to a binary tree is small and becomes very small under the conditions of shift-invariance and scaling properties of the h functions corresponding to the nodes (there is effectively only one h with its small number of parameters because of the low dimensionality (2) plus scaling parameters which should be in the order of the number of layers).

Acknowledgment

This work was supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF 1231216.

References

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

[2] R. Montufar, G. F.and 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.

[3] 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.

[4] B. B. Moore and T. Poggio, "Representations properties of multilayer feedforward networks," *Abstracts of the First annual INNS meeting*, vol. 320, p. 502, 1998.

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

[6] A. Pinkus, "Approximation theory of the mlp model in neural networks," *Acta Numerica*, pp. 143–195, 1999.

[7] 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.

[8] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry.* Cambridge MA: The MIT Press, ISBN 0-262-63022-2, 1972.

[9] T. Poggio and W. Reichardt, "On the representation of multi-input systems: Computational properties of polynomial algorithms.," *Biological Cybernetics, 37, 3, 167-186.*, 1980.

[10] H. W. and S. Kuhn, "A new scheme for the tensor representation," *J. Fourier Anal. Appl.*, vol. 15, no. 5, pp. 706–722, 2009.

[11] N. Cohen and A. Shashua, "Simnets: A generalization of convolutional networks," *CoRR*, vol. abs/1410.0781, 2014.

[12] M. Anthony and P. Bartlett, *Neural Network Learning - Theoretical Foundations.* Cambridge University Press, 2002.

[13] D. Ruderman, "Origins of scaling in natural images," *Vision Res.*, pp. 3385 – 3398, 1997.

[14] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neuroscience*, vol. 2, pp. 1019–1025, Nov. 1999.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385v1 [cs.CV] 10 Dec 2015*, 2015.

[16] H. Mhaskar, "Neural networks for optimal approximation of smotth and analytic functions," *Neural Computation*, vol. 8, pp. 164–177, 1996.

[17] H. Mhaskar, "Approximation properties of a multilayered feedforward artificial neural network," *Advances in Computational Mathematics*, vol. 1, pp. 61–80, 1993.

[18] T. Poggio, "On optimal nonlinear associative recall," *Biological Cybernetics*, vol. 19, pp. 201–209, 1975.

[19] T. Poggio and F. Girosi, "Regularization algorithms for learning that are equivalent to multilayer networks," *Science*, vol. 247, pp. 978–982, 1990.

[20] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Computation*, vol. 7, pp. 219–269, 1995.

[21] T. Poggio and F. Girosi, "A theory of networks for approximation and learning," *Laboratory, Massachusetts Institute of Technology*, vol. A.I. memo n1140, 1989.

[22] J. Mihalik, “Hierarchical vector quantization. of images in transform domain.,” *ELEKTROTECHN. CA5, 43, NO. 3. 92,94.*, 1992.

[23] F. Girosi and T. Poggio, “Representation properties of networks: Kolmogorov’s theorem is irrelevant,” *Neural Computation*, vol. 1, no. 4, pp. 465–469, 1989.

[24] F. Girosi and T. Poggio, “Networks and the best approximation property,” *Biological Cybernetics*, vol. 63, pp. 169–176, 1990.

[25] T. Poggio, J. Mutch, and L. Isik, “Computational role of eccentricity dependent cortical magnification.,” *CBMM Memo No. 017. arXiv:1406.1770v1 . CBMM Funded.*, 2014.

[26] D. Marr, T. Poggio, and E. Hildreth, “Smallest channel in early human vision,” *J. Opt. Soc. Am.*, vol. 70, no. 7, 1980.

1 Appendices

1.1 Compression via depth

An argument that should be equivalent or closely related to our theorem 4 shows that hierarchies can achieve higher compression in terms of bits tuned by learning than shallow architectures. The key condition for compression to be possible is that the inputs allow for reusable “code vectors”. The argument was developed for HBF networks in [7]. It applies to the hierarchical linear splines networks discussed in this paper because of the universality of the underlying motifs. We reproduce the section of [7] below.

We consider a HBF module with Gaussian-like radial kernel and “movable” centers (we assume standard Euclidean distance, see Poggio, Girosi,[19]; Jones, Girosi and Poggio,[20]). The presence of pooling does not change the result.

First, notice that one-hidden-layer HBF can be much more efficient in storage (e.g. bits used for all the centers) than classical RBF because of the smaller number of centers (HBFs are similar to a multidimensional free-knots spline whereas RBFs correspond to classical spline). The next step in the argument is the observation that a network of radial Gaussian-like units become in the limit of $\sigma \to 0$ a look-up table with entries corresponding to the centers. The network can be described in terms of *soft Vector Quantization* (VQ) (see section 6.3 in Poggio and Girosi, [21]). Notice that hierarchical VQ (dubbed HVQ) can be even more efficient than VQ in terms of storage requirements (see e.g. [22]). This suggests that a hierarchy of HBF layers may be similar (depending on which weights are determined by learning) to HVQ. Note that *compression is achieved when parts can be reused in higher level layers*, as shown in the following example.

Example Consider the case of kernels that are in the limit delta-like functions (such as Gaussian with very small variance). Suppose that there are four possible quantizations of the input x: x_1, x_2, x_3, x_4. One hidden layer would consist of four units $\delta(x - x_i), i = 1, \cdots, 4$. But suppose that the vectors x_1, x_2, x_3, x_4 can be decomposed in terms of two smaller parts or features x' and x", e.g. $x_1 = x' \oplus x$", $x_2 = x' \oplus x'$, $x_3 = x$" $\oplus x$" and $x_4 = x$" $\oplus x'$. Then a two layer network could have two types of units in the first layer $\delta(x-x')$ and $\delta(x-x$"); in the second layer four units will detect the conjunctions of x' and x" corresponding to x_1, x_2, x_3, x_4. The memory requirements will go from $4N$ to $2N/2 + 8$ where N is the length of the quantized vectors; the latter is much smaller for large N. Memory compression for HVQ vs VQ – that is for multilayer networks vs one-layer networks – increases with the number of (reusable) parts. Thus for problems that are *compositional*, such as text and probably images, hierarchical architectures of HBF modules minimize memory requirements (but do not increase representational power).

In summary, classical theorems (see references in [23, 24] show that one hidden layer networks can approximate arbitrarily well rather general classes of functions. A possible advantage of multilayer vs one-layer networks that emerges from the above analysis is memory efficiency which can be critical for large data sets and is related to generalization rates.

1.2 Scale invariance and self-similarity

Consider a multiresolution representation of an image such as in Figure 8. Suppose that a non-fractal object – such as a square – activates the bottom two layers. If the object is moved further away the activation will translate to higher layers but remain the same. Suppose to repeat the test with a fractal object – such as a Koch curve. In this case the activation of all layers will be the same and it will not change if the curve is moved further or closer. Self-similarity is rare but statistical self-similarity is everywhere.

2 Convergence of an iterative learning algorithm

Theorem 4.2 in [18] may be adapted to prove convergence of an iteration method that optimizes the best correction at layer m until the last layer and then iterates.

The steps are: I) The optimum approximation of zero degree and the sequences of optimal corrections up to the layer n are calculated. II) The optimum corrections to the result of step are computed for all degrees, starting again from the zero order degree. III)

The iteration results in a series of layers ($i = 1, 2, \cdots, n$) and in the associated mean square errors. The iteration algorithm outlined here (adapted from Katzenelson et al., 1964) gives at each step a meaningful result. Convergence of the iteration algorithm, as well as the uniqueness of the optimum estimator up to an estimator which does not affect the mean square error, are proved in the next theorem.

Theorem 4.2 *The iterative algorithm a) has a limit n-th order estimator; b) the limit estimator is the optimal n-th order estimator in the least square sense; c) the limit estimator is unique up to an estimator of the same degree which does not affect the mean square error.*

2.1 Pooling over scale and position layerwise

The argument relating hierarchies and compositionality of objects and parts is about today's deep convolutional networks. In this case the inputs are at a single scale in different positions. The primate visual system presents a somewhat different input organization.

According to the arguments of [25] the templates in the first stage of the ventral stream are Gabor-like functions with a multiresolution structure (in x, y, s), see Figure 8. In a shallow network pooling over that set of transformed templates would give uniform invariance to all scale transformations of a pattern over the range (s_0, s_{max}); invariance to shifts will be at least within $(-x_0, x_0)$, depending on scale. Note that under scale and translation a single Gabor template originates a set of Gabor wavelets (a tight frame).

An image may activate all or part of the Gabor filters within the inverted truncated pyramid. We assume that the Gabor filter and its transforms under translation and scaling are roughly bandpass and the sampling interval at one scale over x is s, implying half overlap of the filters in x.

For the scale axis a possible sampling pattern follows the estimate of Marr et al.[26] with 5 "frequency channels" having diameter $2s$ = 1'20", 3.1', 6.2', 11.7', 21'. Filter channels as described above are supported by sampling by photoreceptors that starts in the center of the fovea at the Shannon rate with an exagonal lattice with a separation of 30" which increases with increasing eccentricity.

In this "truncated pyramid" model, the Gabor templates correspond to simple cells. Templates across scales may all be in V1 as assumed more or less explicitly by [25]. Here we propose that it may be possible they are distributed between V1, V2 and possibly V4, with small scales in V1 and larger in V2 and V4. We also assume that the pooling is local in x, y, s, possibly in a 2 by 2 by 2 cube.

Because this input architecture is eccentricity dependent and because of the estimates of its dimensions, there is limited shift invariance in the ventral stream while scale invariance is significant. This would imply that in this architecture, differently from the current deep learning networks, the representation of objects by parts will mostly unfold across units at different scales *and* positions rather than only at different positions as in the case of DLNs.

Remarks

- The inverse pyramid model with parameters inferred from data implies

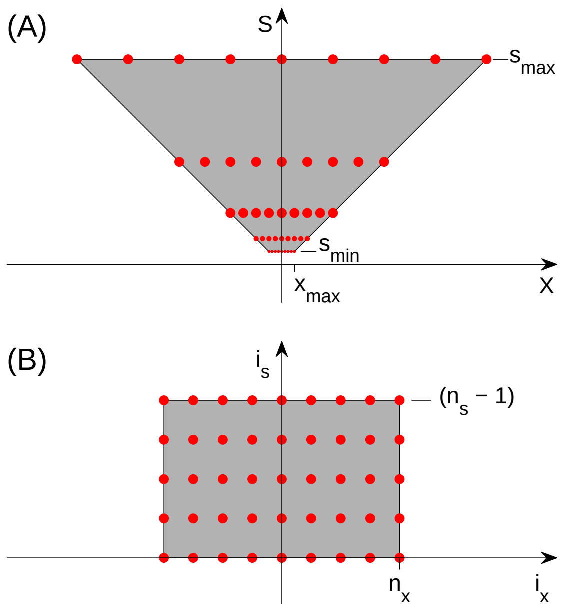

Figure 8: *Figure A shows a foveola of $\approx 26'$ and a total of ≈ 40 units (20 on each side of the center of the fovea). It also shows sampling intervals at the coarsest scale in the fovea (assumed to be around $2s = 21'$ which would span $\approx \pm 6°$. Note that the size of a letter on the ophthalmologist's screen for 20/20 vision is around $5'$. Since the inverted truncated pyramid (A) has the same number of sample points at every scale, it maps onto a square array (B).*

that processing happens on multiple "small" images at different resolutions: these are suggestive of Ullman's MIRCs. Notice that performing recognition on fragments provides, among several other properties, robustness to clutter.

- It is important to assume that information at the (complex cells) nodes of each layer is available to the output. Our argument about clutter in the main text relies on this.