# A review of Deep vs Shallow approximation capabilities Statistical Learning

**Riccardo Cimboli**
**Olga Lazareva**
**Simone Totaro**

July 2017

# 1 Introduction

## 1.1 Main idea

Artificial neural network and more specifically deep neural network achieved remarkable results in different areas of applied machine learning like vision, speech and perception. Despite the great results, it's still unclear how they actually work. Within this project we were mainly focused on the set of questions about the power of the architecture – which classes of functions can NN approximate and learn better? The main message is that the approximation of functions with a compositional structure – can be achieved with the same degree of accuracy by deep and shallow networks but that the number of parameters are much smaller for the deep networks than for the shallow with equivalent approximation accuracy. We were interested in determining how complex the network ought to be to theoretically guarantee that it would approximate an unknown target function $f$ up to a given accuracy $\epsilon > 0$.

## 1.2 Two types of neural networks we are working with

A shallow neural network is a neural network with one hidden layer of the following form:

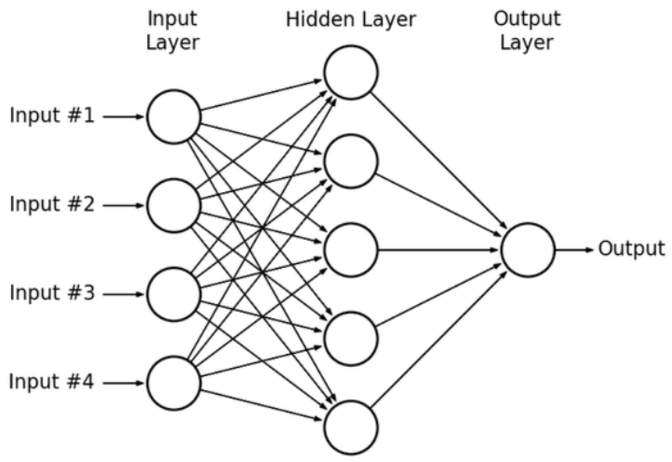

*Figure 1: Shallow neural network*

Any shallow network with $n$ input units can be represented in the following form $x \mapsto \sum_{k=1}^{N} \alpha_k \sigma(\langle w_k, x\rangle + b_k)$, where $w_k \in \mathbb{R}^n$ are the weights for input parameters,$b_k \in \mathbb{R}$ is bias,$a_k \in \mathbb{R}$ coefficient and $\sigma$ is an activation function. Thus, we have $N(n+2)$ number of trainable parameters here.
As a deep neural network we consider a neural network represented as a binary tree.
In a binary tree with $n$ inputs, there are $log_2 n$ levels and a total of $n-1$ nodes. Each of the nodes in a binary tree computes the ridge function $\sum_{k=1}^{N} \alpha_k \sigma(\langle w_k, x\rangle + b_k)$ . Similar to the shallow network,

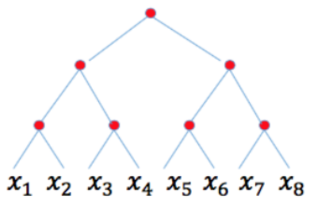

*Figure 2: Example of a binary tree*

a hierarchical network is universal, that is, it can approximate any continuous function; in fact, what we are trying to prove is that it can approximate a compositional functions exponentially better than a shallow network. The key property that makes deep nets exponentially better than shallow for compositional functions is the locality of the constituent functions – that is their low dimensionality. In case of binary tree NN we have $N(n-1)(n+2)$ number of trainable parameters.

### 1.3 Compositional functions

We focused on an important class of problems, corresponding to compositional functions, that is functions of functions. This particular class is of great interest when dealing with real life problems: for instance, every image we may consider can be mathematically expressed as a composition of functions so that if we want to approximate his structure whe should take this into consideration. An especially interesting subset of such compositional functions are hierarchically local compositional functions where all the constituent functions are local in the sense of bounded small dimensionality. The compositional function which would fit the example of deep NN above would have the following structure: $f(x_1, ...x_8) = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8)))$. We assume that the shallow networks do not have any structural information on the function to be learned (here its compositional structure), because they cannot represent it directly and cannot exploit the advantage of a smaller number of parameters. In any case, in the context of approximation theory, we will exhibit and cite lower bounds of approximation by shallow networks for the class of compositional functions. Deep networks on the other hand do represent compositionality in their architecture and can be adapted to the details of such prior information.

## 2 Theoretical background

Within this section we are going to show the different degree of efficiency reach by both shallow and deep neural networks, also we are going to clarify through a theorem what are the main properties that lead one of them to be better than the other in representing functions from the class we are considering (compositional functions). Though, before diving into theorems and main results, it would be helpful to clarify the notation we are going to use and to explore a little bit the space we are going to work with.

## 2.1 Notation

Consider the hypercube $[-1,1]^q$ and let $\mathbf{x}$ be a vector $(\mathbf{x_1}, ..., \mathbf{x_n}) \in \mathbb{R}^n$, $\mathbb{X} = C\left(I^n\right)$ be the space of all continuous functions on $I^n$ with $\|f\| = \max_{x \in I^n} \|f(x)\|_{L_2}$. We can define

$$\mathbf{dist}(f, \mathbb{V}) = \inf_{P \in \mathbb{V}} \|f - P\|, \tag{1}$$

is the *degree of approximation* reached by our class $\mathbb{V}$ of approximators $P$.

Let $m$ be an integer such that $m \geq 1$, we define $W_m^n$ as the set of all functions with continuous partial derivatives of orders up to $m$ such that $\|f\| + \sum_{1 \leq |k_1| \leq m} \left\|D^k f\right\| \leq 1$, where $D^k$ are the partial derivatives, $\mathbf{k} \geq 1$ is a multi-integer and $|\mathbf{k}|_1$ is the sum of components of $\mathbf{k}$. Note that we are actually dealing with a Sobolev space and the condition imposed is bounding the function in the hypercube previously mentioned.
For the sake of this study, we are going to consider the class of compositional function and, more precisely, we will make use of the binary tree structure, which has the form:

$$f\left(x_1, ..., x_n\right) = h_l\left(...h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8)))...\right) \tag{2}$$

for which the correspondent to the previous space $W_m^n$ is $W_m^{2,n}$) which is the space of all functions $f$ with the structure (2) where each constituent function $h$ is in $W_m^n$.
Let's consider, in the end, the corresponding class of deep neural network $D_{N,2}$ that is the set of all functions with the same structure as previously stated, where each constituent function is in $S_{N,n}$, the class of shallow neural network with $N$ units of the form

$$x \to \sum_{k=1}^{N} \alpha_k \sigma(\langle w_k, x\rangle + b_k) \tag{3}$$

where $w_k \in \mathbb{R}^n$ and are vectors containing the weights, $b_k$ is the bias and $a_k \in \mathbb{R}$.
Having explained the notation we are going to use, we can provide the following theorems.

## 2.2 Main theorems

**Theorem 1.** *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be infinitely differentiable, and not polynomial. For $f \in W_{m,n}$ the complexity of shallow network that provide accuracy at least $\epsilon$ is*

$$N = O(\epsilon^{-n/m}) \tag{4}$$

*and is the best possible.*

In order to prove that the result is the best possible, we should go back to the notion of nonlinear widths. Let's say $\mathbb{X}$ is a normed linear space and let's take a compact subset of it $W$. Let $M_n : W \to \mathbb{R}^n$ be a continuous mapping (parameter selection) and $A_n : \mathbb{R}^n \to \mathbb{X}$ be any mapping (recovery algorithm); then we can approximate the function $f$ with $A_n(M_n(f))$. The *nonlinear n-widht* of the compact set W is defined by

$$d_n(W) = inf_{M_n, A_n} sup_{f \in W} \|f, A_n(M_n(f))\|. \tag{5}$$

It was shown by DeVore, Howard and Micchelli in their "*Manuscripta mathematica*" that when we deal with a Sobolev space it is possible to state that we have a lower bound for the $n - width$ which is actually $d(W_{m,n}) \geq cN^{-m/n}$ so, if we try to "reverse" it and go back to to complexity, it is shown that the result in (4) is the best possible.

There is an useful corollary about polynomials which follows from (4), since polynomials are smaller space than spaces of Sobolev functions. Let's denote with $P_k^n$ the linear space of polynomials of degree at most $k$ in $n$ variables. Then:
**Corollary 1.** *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be infinitely differentiable, and not a polynomial. Every $f \in P_k^n$ can be realized with an arbitrary accuracy by shallow network with $k^n$ units (roughly).*

The second main theorem is about deep neural network and, for simplicity, will be formulated for the binary tree case, but it is extendable to all compositional functions.
**Theorem 2.** *For $f \in W_{m,n}^2$ consider a deep network with the same compositional architecture and with an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ which is infinitely differentiable, and not a polynomial. The complexity of the network to provide approximation with accuracy at least $\epsilon$ is*

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

**Proof.** Let's take for instance $f$ with constituent functions $h, h_1, h_2$ and their approximators $P, P_1, P_2$ with an accuracy of at most $\epsilon$. We can show then:

$$\begin{gathered} \|h(h_1, h_2) - P(P_1, P_2\| = \\ \|h(h_1, h_2) - h(P_1, P_2) + h(P_1, P_2) - P(P_1, P_2\| \leq \\ \|h(h_1, h_2) - h(P_1, P_2)\| + \|h(P_1, P_2) - P(P_1, P_2\| \leq c\epsilon \end{gathered}$$

by Minkowski inequality. If we consider the situation described in (13), it follows from result (4) that each node is in $S_{N,2}$ and so, as its complexity is equal to $O(\epsilon^{-2/m})$ and considering that we have $(n-1)$ nodes, this leads to our last theorem.

Similarly to (4), it follows a corollary. Let's consider $T_k^q$ a subset of $P_k^q$ which consist of compositional polynomial with a binary tree graph structure and constituent polynomial functions of degree k; then:
**Corollary 2.** *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be infinitely differentiable, and not a polynomial. Then $f \in T_k^q$ can be realized by a deep network with a binary tree graph and a total of $(q-1)k^2$.*

## 3 Complexity

### 3.1 VC-dimensions

A VC dimension in a neural network is a measure of capacity of the neural network. A capacity is the cardinality (number of elements in a set) of the largest set of elements that can be shattered. Shattering refers to the process of dissecting two classes of elements in such a way that the neural network would always classify the labels and the elements accurately. That means all data with

the same or similar characteristics would lie in their respective classes or groups. Since we can't determine VC-dimention of NN explicitly, we are using bound of VC dimension for both class of networks to see how complicated they ought to be.
**Theorem 3.** *The VC-dimension of the shallow network with N units is bounded by* $(n+2)N^2$ *, the VC-dimension of the binary tree network with N units is bounded by* $(n+2)(n-1)N^2$.

## 3.2 Effective dimensions

The effective dimension of a class $W$ of functions (for a given norm) is said to be $d'$ if for every $\epsilon > 0$, any function in can be recovered within an accuracy of $\epsilon$ (as measured by the norm) using an appropriate network (either shallow or deep) with $\epsilon^- d$ parameters.
Thus, as we wrote above the complexity of the class of functions $W_m^n$ is equal to $O(\epsilon^{-\frac{n}{m}})$ hence the effective dimension for this kind of networks is $\frac{n}{m}$ .
For the class of functions $W_m^{n,2}$ (keeping in mind that $W_m^{n,2} \subset W_m^n$) complexity is equal to $O(\epsilon^{-\frac{2}{m}})$, hence the effective number of dimensions for the class $W_m^{n,2}$ is $\frac{2}{m}$.
It's important to emphasize the following theorem:
**Theorem 4.** *If a family of functions* $f : \mathbb{R}^n \mapsto \mathbf{R}$*a of smoothness* $m$ *has an effective dimension* $< N/r$*, then the functions are compositional and can be approximated with corresponding deep NN avoiding curse of dimensionality, whereas shallow networks cannot directly avoid the curse.*
The key idea we'd like to emphasize with the concept of effective dimensions is that certain deep networks can avoid the curse of dimensionality because of the function compositionality via a small effective dimension.

## 3.3 The curse of dimentionality

It was shown within this report that results obtained in the two cases (shallow and deep network) are quite different. But why can we say that one is "better", or at least more efficient, than the other? It's all connected to the dimension of the space.
Note that, in both cases, things seem to work quite well in low dimensional set-up, but if we consider an increasing number of dimensions things start to get much worse in the shallow network with respect to a deep network representing a binary tree. The reason is simple and clear: having a fixed degree of approximation $\epsilon$, while the complexity of a shallow network that provides the desired $\epsilon$ is exponential in $n$ (dimensionality), the structure approximated by the deep neural network allows us to consider each constituent function at a time, approximated by a shallow network with $n = 2$, so that in the end the complexity is just linear in n, hence it is much more advantageous to use them in order to avoid problems deriving from high-dimensionality.

# 4 Experiment

In order to evalulate the theoretical results got from Theorem 1 and Theorem 2, we construct the following experiments. Let $P_n^k$ be a polynomial in $n$ variables of degree $k$. Let $Dn_{N,l}$ be a feed-forward neural network with $N$ neuron per layer, and $l$ be the number of layer. The general training procedure adopted in both of the experiment is as follow: for each $P_n^k$ where $n, k \in \{2, 3, 4\}$ we generated a dataset made of m = 60.000 samples, which is given to network respectively $l \in 1, 2, 3$

in randomly sampled mini-batches of 3.000, for 20 epochs, which minimize $\frac{1}{m}\sum_{m=1}^{M}(\hat{y}-y)^2$ using Adam Optimizer with fixed learning rate (lr = 0.001). We repeat this process for 5 trial, and pick the minimum loss reached by the network, for each $P_n^k$.

## 4.1 Given N, what is the best $\epsilon$?

In the first experiment our goal is to evaluate the impact of the increasing complexity of the true function namely $O(n^k)$, on the $D_n\{N,l\}$ with $N \in 24, 48, 72$, arranged over the number of layers. It's easy to see that as the function increase, i.e. as it's degree increases, the shallow network, substantially underperform deep neural network, and the speed at which the error decreases is almost flat. Leading to the fact that to reach an approximation of $\epsilon$ the number of units $N$ should increase exponentially.

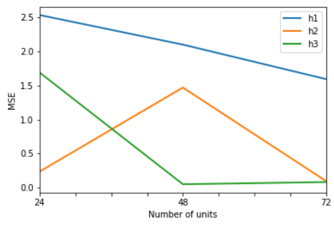

*Figure 3: Approximating $P_4^2$*

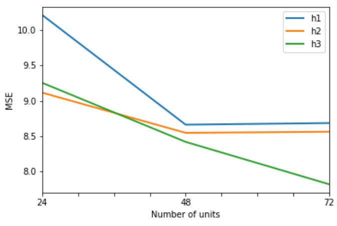

*Figure 4: Approximating $P_4^3$*

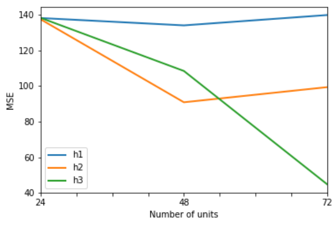

*Figure 5: Approximating $P_4^4$*

## 4.2 Given $\epsilon$, how many N do I need?

In the second experiment the idea is to pick an $\epsilon = 0.05$ and evaluate the number of hidden units needed for each architecture to reach the given accuracy or less, as the complexity of $P_n^k$ increases. For computational purposes the N scales in power of 2, and we stop the training after using $2^9$ units. As expected the shallow network uses exponentially many more units with respect to composed architectures. The results are summarized in the following tables. The first table represent the best accuracy reached by each architecture, if NaN, means that the network never reached the requested accuracy even with $2^9$ units. The second table report the number of units used by each architecture.

| | $\mathbf{S_N}$ | $\mathbf{D_{N,2}}$ | $\mathbf{D_{N,3}}$ |
|---|---|---|---|
| $P_2^2$ | 0.037 | 0.0293 | 0.002 |
| $P_2^4$ | NaN | 0.0282 | 0.024 |
| $P_3^2$ | 0.031 | 0.0386 | 0.006 |
| $P_3^4$ | NaN | NaN | 0.039 |
| $P_4^2$ | 0.011 | 0.0241 | 0.022 |

| | $\mathbf{S_N}$ | $\mathbf{D_{N,2}}$ | $\mathbf{D_{N,3}}$ |
|---|---|---|---|
| $P_2^2$ | 1536 | 48 | 32 |
| $P_2^4$ | 1536 | 768 | 64 |
| $P_3^2$ | 1536 | 12 | 32 |
| $P_3^4$ | 1536 | 1536 | 256 |
| $P_4^2$ | 1536 | 96 | 32 |

Also, we'd like to provide the report with a table that contains number of trainable parameters (Table 3) and a table with corresponding number of VC-dimensions (Table4)

| | $\mathbf{S_N}$ | $\mathbf{D_{N,2}}$ | $\mathbf{D_{N,3}}$ |
|---|---|---|---|
| $P_2^2$ | 4608 | 48 | 32 |
| $P_2^4$ | 4608 | 768 | 64 |
| $P_3^2$ | 6144 | 24 | 64 |
| $P_3^4$ | 6144 | 3072 | 512 |
| $P_4^2$ | 7680 | 288 | 96 |

| | $\mathbf{S_N}$ | $\mathbf{D_{N,2}}$ | $\mathbf{D_{N,3}}$ |
|---|---|---|---|
| $P_2^2$ | 63700992 | 9216 | 4096 |
| $P_2^4$ | 63700992 | 2359296 | 16384 |
| $P_3^2$ | 150994944 | 5760 | 40960 |
| $P_3^4$ | 150994944 | 94371840 | 2621440 |
| $P_4^2$ | 294912000 | 1492992 | 165888 |

## 5 Conclusion and Discussion

As a brief summary, we reviewed the main theoretical results discussed in ”Learning functions: Why deep is better than shallow”, from Mhaskar, Liao and Poggio and also ”Why and When Can Deep – but Not Shallow – Networks Avoid the Curse of Dimensionality: a Review” from Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, Qianli Liao. The result is that if the target function of the form of compositional structure then deep neural network can achieve a given level accuracy with exponentially less paramater than shallow networks. One of the hypothesis states from the author is that deep neural network can implicitly represent the compositional structure of the true function and hence avoid the curse of dimensionality. We then experiment with polynomials of different degrees and dimensions to evaluate the error achieved by fixed different architecture, and the other way around, that is fixing the required accuracy and letting the complexity varies up to an upper bound base of 2. The experimental results showed that deep neural networks perform better than shallow networks in approximating compositional functions as the complexity of the underlying function increases.