# On the Complexity of Neural Network Classifiers: A Comparison Between Shallow and Deep Architectures

Monica Bianchini and Franco Scarselli

***Abstract*— Recently, researchers in the artificial neural network field have focused their attention on connectionist models composed by several hidden layers. In fact, experimental results and heuristic considerations suggest that deep architectures are more suitable than shallow ones for modern applications, facing very complex problems, e.g., vision and human language understanding. However, the actual theoretical results supporting such a claim are still few and incomplete. In this paper, we propose a new approach to study how the depth of feedforward neural networks impacts on their ability in implementing high complexity functions. First, a new measure based on topological concepts is introduced, aimed at evaluating the complexity of the function implemented by a neural network, used for classification purposes. Then, deep and shallow neural architectures with common sigmoidal activation functions are compared, by deriving upper and lower bounds on their complexity, and studying how the complexity depends on the number of hidden units and the used activation function. The obtained results seem to support the idea that deep networks actually implements functions of higher complexity, so that they are able, with the same number of resources, to address more difficult problems.**

***Index Terms*— Betti numbers, deep neural networks, function approximation, topological complexity, Vapnik–Chervonenkis dimension (VC-dim).**

## I. Introduction

IN RECENT years, there has been a substantial growth of interest in feedforward neural networks with many layers [1]–[3]. The main idea underlying this research is that, even if common machine learning models often exploit only shallow architectures, deep architectures[1] are required to solve complex AI problems as, e.g., image analysis or language understanding [6]. Such a claim is supported by numerous evidences coming from different research fields. For example, in neuroscience, it is well known that the neocortex of the mammalian brain is organized in a complex layered structure of neurons [7], [8]. In machine learning practice, complex applications are commonly approached by passing the data through a sequence of stages, e.g., preprocessing, prediction and postprocessing. In the specific field of artificial neural networks, researchers have proposed and experimented several models exploiting multilayer architectures, including convolutional neural networks [9], cascade correlation networks [10], deep neural networks [4], [5], layered graph neural networks [11], and networks with adaptive activations [12].

In addition, even recurrent/recursive models [13], [14] and graph neural networks [15], which cannot be strictly defined deep architectures, actually implement the unfolding mechanism during learning, which, in practice, produces networks that are as deep as the data they have to model. All the above mentioned architectures have found in recent years a wide application to challenging problems in pattern recognition, such as, for example, in natural language processing [16], chemistry [17], image processing [18], [19], and web mining [20], [21].

Nevertheless, from a theoretical point of view, the advantages of deep architectures are not yet completely understood. The existing results are limited to neural networks with boolean inputs and units, implementing logical gates or threshold activation functions, and to sum-product networks [22], [23]. In the first case, it has been proved that the implementation of some maps, for instance, the parity function, requires a different number of units according to whether a shallow or a deep architecture is exploited. However, such results deal only with specific cases and do not provide a general tool for studying the computational capabilities of deep neural networks. For sum-product networks, instead, deep architectures were proven to be more efficient in representing real-valued polynomials. Anyway, there seems to be no way to directly extend those results to common neural networks, with sigmoidal activation functions.

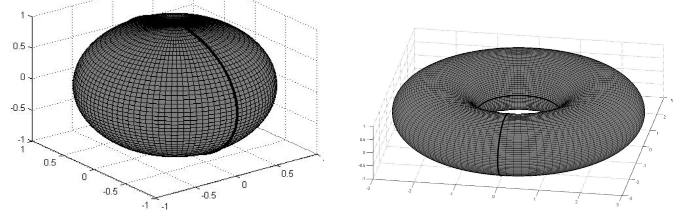
Intuitively, an important actual limitation is that no measure is available to evaluate the complexity of the functions implemented by neural networks. In fact, the claim that deep networks can solve more complex problems can be reformulated as *deep neural networks can implement functions with higher complexity than shallow ones, when using the same number of resources*. Anyway, to the best of our knowledge, the concept of high complexity function has not been formally defined yet in the artificial neural network field. Actually, the complexity measures commonly used are not useful for our purposes, since they deal with different concepts,

Manuscript received May 6, 2013; revised October 12, 2013; accepted November 21, 2013. Date of publication January 2, 2014; date of current version July 14, 2014.

The authors are with the University of Siena, Siena 53100, Italy (e-mail: monica@dii.unisi.it; franco@dii.unisi.it).

Digital Object Identifier 10.1109/TNNLS.2013.2293637

[1] It is worth noting that, in this paper, the terms "deep architectures" and "deep networks" are completely interchangeable and are used to address the general class of feedforward networks with many hidden layers, whereas in [4] and [5] "deep network" identifies a specific model in the above class.



(a) (b)

Fig. 1. (a) 3-D sphere. (b) Torus.

such as the architectural complexity (number of parameters, units, and layers) or the generalization capability of a model, based on the Vapnik–Chervonenkis dimension (VC-dim).[2] Here, we want to evaluate the complexity of the implemented function, independently from other characteristics of the network.

In this paper, we propose to exploit some concepts coming from topology theory in order to design a measure of function complexity that can be applied to neural networks, used in the classification framework. In addition, we will employ such a measure to compare deep and shallow networks, proving that the former have some theoretical advantages over the latter. More precisely, let $f_{\mathcal{N}} : \mathbb{R}^n \rightarrow \mathbb{R}$ be the function implemented by a feedforward neural network $\mathcal{N}$, with $n$ inputs and a single output. We will measure the complexity of the function $f_{\mathcal{N}}$ by the topological complexity of the set $S_{\mathcal{N}} = \{x \in \mathbb{R}^n | \, f_{\mathcal{N}}(x) \geq 0\}$. Such an approach is natural when the network $\mathcal{N}$ is used for classification since, in this case, $S_{\mathcal{N}}$ is just the set of inputs scored with a nonnegative response, i.e., the set of patterns belonging to the positive class.

We will also exploit the concept of Betti numbers [25], in order to measure the complexity of the set $S_{\mathcal{N}}$. In algebraic topology, Betti numbers are used to distinguish spaces with different topological properties. More precisely, for any subset $S \subset \mathbb{R}^n$, there exist $n$ Betti numbers,[3] which, in the following, will be denoted by $b_i(S)$, $0 \leq i \leq n-1$. Intuitively, the first Betti number $b_0(S)$ is the number of connected components of the set $S$, while the $i$th Betti number $b_i(S)$ counts the number of $(i+1)$-dimensional holes in $S$. For example, consider the four subsets $S_{BI}, S_B, S_D$, and $S_I$ in $\mathbb{R}^2$, representing the character strings $BI$, $B$, $D$, and $I$, respectively. All the subsets are constituted by a single connected component except for $S_{BI}$, which has two components. Thus, $b_0(S_B) = b_0(S_D) = b_0(S_I) = 1$ and $b_0(S_{BI}) = 2$. In addition, $b_1(S_{BI}) = b_1(S_B) = 2$, $b_1(S_D) = 1$, and $b_1(S_I) = 0$, because $S_{BI}$ and $S_B$ contain two (2-D) holes, $S_D$ contains one hole, and $S_I$ has no hole, respectively. For an example, in a larger dimensional space, let us compare a 3-D sphere, $S_\pi$, and a torus, $S_\tau$ (Fig. 1). We can observe that both of them have a single connected component and contain a 3-D hole (i.e., $b_0(S_\pi) = b_2(S_\pi) = b_0(S_\tau) = b_2(S_\tau) = 1$). On the other hand, the sphere contains a single 2-D hole, defined by any circle on its surface, whereas the torus has two 2-D holes, the one in the center and the one in the middle of the "tube" (i.e., $b_1(S_\pi) = 1$, $b_1(S_\tau) = 2$).

TABLE I

UPPER AND LOWER BOUNDS ON THE GROWTH OF $B(S_{\mathcal{N}})$ FOR NETWORKS WITH $h$ HIDDEN UNITS, $n$ INPUTS, AND $l$ HIDDEN LAYERS. THE BOUND IN THE FIRST ROW IS A WELL-KNOWN RESULT AVAILABLE IN [26]

| **Inputs** | **Layers** | **Activation function** | **Bound** |
|---|---|---|---|
| | | Upper bounds | |
| $n$ | 3 | threshold | $O(h^n)$ |
| $n$ | 3 | arctan | $O((n+h)^{n+2})$ |
| $n$ | 3 | polynomial, degree $r$ | $\frac{1}{2}(2+r)(1+r)^{n-1}$ |
| 1 | 3 | arctan | $h$ |
| $n$ | any | arctan | $2^{h(2h-1)}O((nl+n)^{n+2h})$ |
| $n$ | any | tanh | $2^{(h(h-1))/2}O((nl+n)^{n+h})$ |
| $n$ | any | polynomial, degree $r$ | $\frac{1}{2}(2+r^l)(1+r^l)^{n-1}$ |
| | | Lower bounds | |
| $n$ | 3 | any sigmoid | $(\frac{h-1}{n})^n$ |
| $n$ | any | any sigmoid | $2^{l-1}$ |
| $n$ | any | polynomial, deg. $r \geq 2$ | $2^{l-1}$ |

Thus, Betti numbers capture a topological notion of complexity that can be used to compare subspaces of $\mathbb{R}^n$. In the examples, we can assert that $S_{BI}$ is more complex than $S_B$, because the former is made by two components, $S_B$ is more complex than $S_D$, because $S_B$ has more holes than $S_D$, and the torus is more complex than the sphere, because it has two 2-D holes instead of one.

In this paper, we will evaluate the Betti numbers $b_i(S_{\mathcal{N}})$ of the regions $S_{\mathcal{N}}$, positively classified by feedforward neural networks, in order to understand how they are affected by the network architecture, i.e., based on being $\mathcal{N}$ deep or not. Several theorems will be derived, providing both upper and lower bounds on the sum of the Betti numbers, $B(S_{\mathcal{N}}) = \sum_i b_i(S_{\mathcal{N}})$, varying the architecture and the activation functions exploited by the network. More specifically, considering a particular class of networks, the existence of a lower bound $L$ implies that for at least one network $\mathcal{N}$, belonging to the considered class, $B(S_{\mathcal{N}}) \geq L$ holds, while an upper bound $U$ is such if $B(S_{\mathcal{N}}) \leq U$ for all the networks in the class.

Table I summarizes the obtained results. Here, $h$ represents the number of hidden neurons, $n$ is the number of inputs, and $l$ is the number of layers in the network. In addition, the common big $O$ notation is used to describe the limit behavior of $B(S_{\mathcal{N}})$ with respect to $h$. Thus, $B(S_{\mathcal{N}}) \in O(f(h))$ means that $\lim_{h\rightarrow\infty} B(S_{\mathcal{N}})/f(h) < \infty$ holds and, similarly, $B(S_{\mathcal{N}}) \in \Omega(f(h))$ means that $\lim_{h\rightarrow\infty} f(h)/B(S_{\mathcal{N}}) < \infty$ holds. It is worth mentioning that all the proposed results hold also if $l$ is replaced by $h$ in the upper bounds, and by $2h$, in the lower bounds.[4]

Interestingly, a general result seems to emerge from our analysis, summarized by the following two propositions.

[2] A probabilistic upper bound on the test error of a classification model can be predicted using the VC–dimension [24].

[3] Formally, $b_i(S)$ is defined for any $i \geq 0$, but $b_i(S) = 0$ for $i \geq n$.

[4] Indeed, $h \geq l$ holds, whereas, for lower bounds, theorems are proved by providing an example of a network with two hidden units per layer.

*Proposition 1:* For network architectures with a single hidden layer, the sum of the Betti numbers, $B(S_{\mathcal{N}})$, grows at most polynomially with respect to the number of the hidden units $h$, i.e., $B(S_{\mathcal{N}}) \in O(h^n)$, where $n$ is the input dimension.

*Proposition 2:* For deep networks, $B(S_{\mathcal{N}})$ can grow exponentially in the number of the hidden units, i.e., $B(S_{\mathcal{N}}) \in \Omega(2^h)$.

More precisely, the above propositions have been proved for networks exploiting some specific activation functions, namely, the inverse tangent, $\arctan(a) = \tan^{-1}(a)$, the hyperbolic tangent,[5] $\tanh(a) = (e^a - e^{-a})/(e^a + e^{-a})$, and polynomials. In fact, by simplifying the bounds in Table I, it can be easily seen that both the claims are proved for polynomial activation functions of degree $r$: $B(S_{\mathcal{N}}) \in O(r^n)$, for networks with a unique hidden layer, and it ranges between $\Omega(2^h)$ and $O(r^{2hn})$ for deep networks. In addition, both claims are also demonstrated for networks using the arctangent; the sum of the Betti numbers ranges between $\Omega((h/n)^n)$ and $O((n+h)^{n+2})$ in shallow networks, and between $\Omega(2^h)$ and $O(4^{h^2}(nh)^{n+2h})$ in deep networks. On the other hand, Proposition 2 holds for generic sigmoid activation functions (i.e., monotone increasing, functions having left and right limits); in this case, the lower bound is $\Omega(2^h)$.

For the sake of clarity, it is worth pointing out that we do not know whether the two propositions hold for all the commonly used activation functions, and that the presented results are nonconclusive and leave open interesting avenues of investigation for future work. For instance, Proposition 1 has not been proved for networks with the hyperbolic tangent function. Interestingly, we will clarify that the extension of such results may require some significant advancements in algebraic topology, since they are related to some important and still unsolved problems.

Finally, from a practical point of view, both propositions suggest that deep networks have the capability of implementing more complex sets than shallow ones. This is particularly interesting if compared with current results on VC-dim of multilayer perceptrons. Actually, the available bounds on VC-dim with respect to the number of network parameters[6] do not depend on whether deep or shallow architectures are considered. Since the VC-dim is related to generalization, the two results together seem to suggest that deep networks allow to largely increase the complexity of the classifiers implemented by the network without significantly affecting its generalization behavior. This paper includes also an intuitive and preliminary discussion on the peculiarities that an application domain should have to take advantage of deep networks.

This paper is organized as follows. In Section II, the notation is introduced. Section III collects the main results. In Section IV, such results are discussed and compared with the state-of-the-art achievements. Finally, conclusions are drawn in Section V.

## II. Notation

### A. Multilayer Perceptron Networks

In this paper, we consider multilayer perceptron networks with ridge activation functions and a unique output. Formally, since it is assumed that the network is composed of neurons (units) organized in layers, each neuron is uniquely identified by a pair $(k, l)$, where $l$ is the layer it belongs to, and $k$ is the sequential position it occupies within the layer. In addition, it is also assumed that the layers are fully connected, i.e., there is a link from any neuron of a layer to any neuron of the next layer, but no link exists between units of the same layer or among nonconsecutive layers (in other words, there are no shortcut connections). The network parameters collect the weights assigned to the links and the biases related to each unit; in particular, with $w^l_{k,i} \in \mathbb{R}$ we will denote the weight of the connection from neuron $(i, l-1)$ to neuron $(k, l)$, whereas $b^l_k \in \mathbb{R}$ will represent the bias of the $(k, l)$ unit.

The network implements a function by activating each neuron, starting from the first layer, which actually "passes" the input, to the last layer, which returns the calculated function, $f_{\mathcal{N}}(x) : \mathbb{R}^n \to \mathbb{R}$. The activation of each neuron produces a real value, $o_{k,l}$, called output. More precisely, the outputs of the neurons in the first layer, i.e., layer 0, are set to the components of the input $x \in \mathbb{R}^n$, i.e., $o_{k,0} = x_k$, $1 \le k \le n$. For the following layers, $l > 0$, the output is computed based on the outputs of the units in the previous layer $l-1$, as

$$o^l_k = \sigma\left(\sum_{i=1}^{h_{l-1}} w^l_{k,i} o^{l-1}_i + b^l_k\right)$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is the activation function, and $h_0 = n$.

We assume here that the activation function of the unique neuron in the output (last) layer is the identity function, i.e., $\sigma(a) = a$.[7] On the other hand, for the activation function of the hidden layers, four common cases will be studied: 1) the inverse tangent function, i.e., $\sigma(y) = \arctan(y) = \tan^{-1}(y)$; 2) the hyperbolic tangent function, i.e., $\sigma(y) = \tanh(y) = (e^y - e^{-y})/(e^y + e^{-y})$; 3) polynomial activation functions, i.e., $\sigma(y) = p(y)$, where $p$ is a polynomial; and 4) sigmoid functions, i.e., $\sigma$ is a limited monotone increasing function, for which there exist $l_\sigma, r_\sigma \in \mathbb{R}$ such that $\lim_{y \to -\infty} \sigma(y) = l_\sigma$ and $\lim_{y \to \infty} \sigma(y) = r_\sigma$.

Finally, the network output is given by the output of the neuron in the last layer. Thus, the function $f_{\mathcal{N}}$, implemented by a three-layer network $\mathcal{N}$ (one hidden layer), with $n$ inputs and $h_1$ hidden units, may be expressed in the form

$$f_{\mathcal{N}}(x) = b^2_1 + \sum_{i_1=1}^{h_1} w^2_{1,i_1} \sigma\left(b^1_{i_1} + \sum_{i_0=1}^{n} w^1_{i_1,i_0} x_{i_0}\right). \quad (1)$$

On the other hand, a four-layer network with, respectively, $h_1$ and $h_2$ units in the first and in the second hidden layer,

[5] It is worth mentioning that all the presented results for the hyperbolic tangent apply also to the logistic sigmoid $\mathrm{logsig(a)} = 1/(1+\mathrm{e}^{-\mathrm{a}})$, which can be obtained by translations and dilations from $\tanh(\cdot)$.

[6] Current bounds range from $O(p \log p)$ to $O(p^4)$, according to the used activation function, being $p$ the number of the network parameters.

[7] In common applications, output neurons with linear or sigmoidal activation functions are usually employed. In this paper, the analysis is carried out only for the former case, but most of the presented results can easily be extended to the latter.

implements the function

$$f_{\mathcal{N}}(x) = b_1^3 + \sum_{i_2=1}^{h_2} w_{1,i_2}^3 \times \sigma\left(b_{i_2}^2 + \sum_{i_1=1}^{h_1} w_{i_2,i_1}^2 \sigma\left(b_{i_1}^1 + \sum_{i_0=1}^{n} w_{i_1,i_0}^1 x_{i_0}\right)\right)$$

and so on.

### *B. Betti Numbers and Pfaffian Functions*

Formally, the $k$th Betti number, $b_k(S)$, is defined as the rank of the $k$th homology group of the space $S$. The reader is referred to text books on algebraic topology for a wider introduction to homology [25], [27]. Here, it suffices to say that the homology theory realizes a link between topological and algebraic concepts, and provides a theoretical framework that allows to compare objects according to their forms.

Homology theory starts from the observation that two topological spaces can be distinguished looking at their holes, and provides a formal mathematical tool for determining different types of holes. Actually, homology is also defined to be based on cycles. In fact, a $k$-dimensional compact hole can be identified by its $(k-1)$-dimensional cyclic border. For example, the border of the 3-D hole inside a sphere is the surface of the sphere itself; a torus has two 2-D holes, corresponding to the cyclic paths around the inside (of the "donut") and around the tube, respectively (Fig. 1). The $k$th homology group of a space $S$, intuitively, collects a coding of all the different types of $(k+1)$-dimensional holes/cycles contained in $S$. Thus, the $k$th Betti number provides a sort of signature to recognize similar objects, by counting the number of $(k+1)$-dimensional holes.

The homology theory plays a fundamental role in many disciplines, including, for instance, physics [28], image processing [29], and sensor networks [30].

A sequence $(f_1, \ldots, f_\ell)$ of real analytic functions in $\mathbb{R}^n$ is a Pfaffian chain on the connected component $\mathcal{U} \subseteq \mathbb{R}^n$, if there exists a set of polynomials $P_{i,j}$ such that for each $x \in \mathcal{U}$

$$\frac{\partial f_i}{\partial x_j} = P_{i,j}(x, f_1(x), \ldots, f_i(x)),\ 1 \le i \le \ell,\ 1 \le j \le n \quad (2)$$

holds. A Pfaffian function in the chain $(f_1, \ldots, f_\ell)$ is a map $f$, expressed as a polynomial $P$ in $x$ and $f_i(x), i = 1, \ldots, \ell$

$$f(x) = P(x, f_1(x), \ldots, f_\ell(x)).$$

The complexity (or the format) of the Pfaffian function $f$ is defined by a triplet $(\alpha, \beta, \ell)$, where $\alpha$ is the degree of the chain, i.e., the maximum degree of the polynomials $P_{i,j}$, $\beta$ is the degree of the Pfaffian function, i.e., the degree of $P$, and $\ell$ is the length of the chain.

The class of Pfaffian maps is very wide and includes most of the functions commonly used in engineering and computer science applications. Actually, most of the elementary functions are Pfaffian, e.g., the exponential, the trigonometric functions,[8] the polynomials, and all the algebraic functions. In addition, the compositions and the inverses of Pfaffian functions are Pfaffian. Therefore, the natural logarithm, $\ln(x)$, is Pfaffian, being the inverse of the exponential function $e^x$, and the logistic sigmoid $\operatorname{logsig}(x) = \frac{1}{(1+e^{-x})}$ is Pfaffian, since it is an algebraic function of Pfaffian functions.

## III. Theoretical Result

In this section, we collect some theoretical results on function complexity, which can be applied to neural networks used in the classification framework, in order to compare deep and shallow architectures. Actually, in Sections III-A and B, results on upper and lower bounds, evaluated for different activation functions (threshold, polynomials, and sigmoidal), are, respectively, reported. For the theorems' proofs, the reader is referred to the Appendix, that also contains a discussion on possible extensions to the presented results.

### *A. Upper Bounds*

The first theorem we propose provides an upper bound on the sum of the Betti numbers of the positive set realized by three-layer networks with arctan(·) activation function.

*Theorem 1:* Let $\mathcal{N}$ be a three-layer neural network with $n$ inputs and $h$ hidden neurons, with arctangent activation in the hidden units. Then, for the set $S_{\mathcal{N}} = \{x \in \mathbb{R}^n | f_{\mathcal{N}}(x) > 0\}$, $B(S_{\mathcal{N}}) \in O((n+h)^{n+2})$ holds.

Thus, $B(S_{\mathcal{N}})$ grows at most polynomially in the number of hidden neurons. Interestingly, a similar result can be obtained for analogous shallow architectures having a single hidden layer of polynomial units.

*Theorem 2:* Let $p$ be a polynomial of degree $r$. Let $\mathcal{N}$ be a three-layer neural network with $n$ inputs, and $h$ hidden neurons, having $p$ as activation function. Then, for the set $S_{\mathcal{N}} = \{x \in \mathbb{R}^n | f_{\mathcal{N}}(x) > 0\}$, it holds that

$$B(S_{\mathcal{N}}) \le \frac{1}{2}(2+r)(1+r)^{n-1}.$$

In addition, when the network has only one input and the activation function is the arctangent, the problem of defining a bound on $B(S_{\mathcal{N}})$ becomes simpler. In fact, if $n = 1$, then $S_{\mathcal{N}}$ is a set of intervals in $\mathbb{R}$ and the only nonnull Betti number is the first one, i.e., $b_0(S_{\mathcal{N}})$, which counts the number of intervals where $f_{\mathcal{N}}(x)$ is nonnegative. In this case, as proved in the following theorem, the bound on $b_0(S_{\mathcal{N}})$ is linear.

*Theorem 3:* Let $\mathcal{N}$ be a three-layer neural network with one input and $h$ hidden neurons. If the activation function used in the hidden units is the arctangent, then it holds that

$$b_0(S_{\mathcal{N}}) \le h.$$

Finally, we derive some upper bounds for deep networks, which can have any number of hidden layers.

*Theorem 4:* Let $\mathcal{N}$ be a network with $n$ inputs, one output, $l$ hidden layers, and $h$ hidden units with activation $\sigma$. If $\sigma$ is a Pfaffian function such that the polynomials $P_{i,j}$, defining its chain, do not depend on the input variable $x$ (see Eq.(2)), $\sigma$ has complexity $(\alpha, \beta, \ell)$, and $n \le h\ell$, then

$$B(S_{\mathcal{N}}) \le 2^{(h\ell(h\ell-1))/2} \times O\Big((n((\alpha+\beta-1+\alpha\beta)l + \beta(\alpha+1)))^{n+h\ell}\Big)$$

[8] More precisely, sin(·) and cos(·) are Pfaffian within a period. For instance, sin(·) and cos(·) are Pfaffian in $[0, 2\pi]$.

holds. In addition, if $\sigma = \arctan(\cdot)$ and $n \leq 2h$, then

$$B(S_{\mathcal{N}}) \leq 2^{h(2h-1)} O\left((nl+n)^{n+2h}\right),$$

and if $\sigma = \tanh(\cdot)$ and $n \leq h$

$$B(S_{\mathcal{N}}) \leq 2^{(h(h-1))/2} O\left((nl+n)^{n+h}\right).$$

It is worth noting that the hypothesis $n \leq h\ell$ ($n \leq 2h$, $n \leq h$) is not restrictive for our purposes, since we are interested in the asymptotic behavior of deep architectures, with the number of hidden layers (and, therefore, the number of hidden units) tending to infinity. Similarly, the assumption on $P_{i,j}$, defining the chain of $\sigma$, is satisfied by common activation functions and represents just a technical hypothesis added to simplify the proof of Theorem 4 (see the Appendix, for more details).

Finally, for deep networks with polynomial activation functions, the upper bound of Theorem 4 can be improved as follows.

*Corollary 1:* Let $f_{\mathcal{N}}(x)$ be the function implemented by a neural network with $n$ inputs, one output, and $l$ hidden layers, composed by neurons with a polynomial activation function of degree $r$. Then, it holds that

$$B(S_{\mathcal{N}}) \leq \frac{1}{2}(2+r^l)(1+r^l)^{n-1}.$$

### B. Lower Bounds

In the following, we present some results on lower bounds of $B(S_{\mathcal{N}})$, for both deep and shallow networks. In particular, we study the growth of $B(S_{\mathcal{N}})$ when the number of layers is progressively increased. Actually, the presented results concern $b_0(S_{\mathcal{N}})$. However, since $B(S_{\mathcal{N}}) > b_0(S_{\mathcal{N}})$, they also hold for $B(S_{\mathcal{N}})$.

In the next theorem, we prove that for networks of growing depth, $B(S_{\mathcal{N}})$ can actually become exponential with respect to the number of the hidden neurons. In fact, Theorem 5 provides an exponential lower bound on $b_0(S_{\mathcal{N}_l})$, for deep networks with hidden neurons having sigmoid activation.

*Theorem 5:* For any positive integers $l, n$, and any sigmoid activation function, there exists a multilayer network $\mathcal{N}_l$ with $n$ inputs, a unique output, $l$ hidden layers, and two neurons per layer, such that

$$b_0(S_{\mathcal{N}_l}) \geq 2^{l-1} + 1.$$

Interestingly, Theorem 5 can be extended to networks with nonlinear continuously differentiable activation functions.

*Theorem 6:* Let $\sigma \in C^3$ be a nonlinear function in an open interval $U \subseteq I\!R$. Then, for any positive integers $l, n$, there exists a multilayer network $\mathcal{N}_l$, with $n$ inputs, a unique output, $l$ hidden layers, and two neurons in each layer, with activation $\sigma$, such that

$$b_0(S_{\mathcal{N}_l}) \geq 2^{l-1} + 1.$$

Notice that even if both Theorems 5 and 6 can be applied to the common activation functions $\tanh(\cdot)$ and $\arctan(\cdot)$, they deal with slightly different classes of functions. In particular, Theorem 5 copes with general sigmoidal activation functions, including, e.g., piecewise continuous functions, whereas Theorem 6 deals with nonlinear $C^3$ activations, a class that includes all the polynomials with degree larger than one.

On the other hand, the obtained results are not useful for networks with a fixed number of layers, since, in this case, the bound becomes a constant. In the following, a lower bound for the complexity of three-layer networks is devised, varying the number of hidden neurons. More precisely, in the next theorem, we show that $b_0(S_{\mathcal{N}})$ can grow as $((h-1)/2n)^n$ for three-layer networks with sigmoid activation functions. Interestingly, the lower bound is $\Omega(h^n)$, and it is optimal for networks with $\arctan(\cdot)$ function, since it equals the upper bound devised in Theorem 1. It is also optimal for networks with threshold activation (see [26], [31]).

*Theorem 7:* For any positive integers $n, m$, and any sigmoidal activation function $\sigma$, there exists a three-layer network $\mathcal{N}$, with $n$ inputs and $h = n(2m-1)$ hidden units, for which

$$b_0(S_{\mathcal{N}}) \geq \left(\frac{h-1}{2n}\right)^n.$$

It is worth mentioning that a tighter lower bound $b_0(S_{\mathcal{N}}) \geq (h/n)^n$ can be obtained by assuming that the activation function is nonpolynomial and analytic at some point.[9] (see the theorem's proof, for more details).

Instead, Theorem 7 cannot be extended to three-layer networks with polynomial activations, since they are not universal approximators. Actually, it can easily be seen that, in this case, $f_{\mathcal{N}}$ is a polynomial having at most the degree of the activation function, independently of the number of hidden units. In fact, also the upper bound in Theorem 2 does not depend on the number of hidden neurons, but only on the degree of the activation function and on the input space dimension.

## IV. Related Literature and Discussion

This paper belongs to the wide class of studies whose intent has been that of characterizing the peculiarities of the functions implemented by artificial neural networks. In the early 90s, the capacity of multilayer networks—with sigmoidal, RBF, and threshold activation functions—of approximating any continuous map up to any degree of precision was established [32]–[35]. Such seminal studies have been subsequently extended in several ways, considering networks with very generic neurons (e.g., with analytic activation functions), expanding the class of approximable maps (e.g., to integrable functions [36]), or studying nonstatic networks (e.g., recurrent, recursive, and graph neural networks [37]–[39]). Unfortunately, those results do not allow to distinguish between deep and shallow architectures, since they are based on common properties instead of on distinguishing features.

On the other hand, researchers in the field of logic networks have actually pursued the goal of defining the effect of the network depth on the amount of resources required to implement a given function. In this ambit, it has been shown that there exist boolean functions, whose realization requires a polynomial number of logic gates (AND, OR, and NOT) using

[9] A function is analytic at a given point if its Taylor series, computed with respect to such a point, converges.

a network with $l$ layers, whereas an exponential number of gates is needed for a smaller number of layers [40], [41]. A well-known function of this class is the parity function,[10] which requires an exponential number of gates (in the number of inputs), if the network has two layers, whereas it can be implemented by a linear number of logic gates, if a logarithmic number of layers (in the input dimension) are employed. It is worth mentioning that it has been also proved that most of the boolean functions require an exponential number of gates to be computed with a two-layer network [42].

In addition, in [22], deep and shallow sum-product networks[11] were compared, using two classes of functions. Their implementation by deep sum-product networks requires a linear number of neurons with respect to the input dimension $n$ and the network depth $l$, whereas a two-layer network needs at least $O(2^{\sqrt{n}})$ and $O((n-1)^l)$ neurons, respectively.

Finally, similar results were obtained for weighed threshold circuits.[12] For example, it has been proved that there are monotone functions[13] $f_k$ that can be computed with depth $k$ and a linear number of logic (AND, OR) gates, but they require an exponential number of gates to be computed by a weighed threshold circuit with depth $k-1$ [44].

The above mentioned results, however, cannot be applied to common Backpropagation neural networks. In this sense, as far as we know, the theorems reported in this paper are the first attempt to compare deep and shallow architectures, which exploit the common $\tanh(\cdot)$ and $\arctan(\cdot)$ activation functions. Interestingly, the presented theorems are also related to those on VC-dim, both from a mathematical and a practical point of view. An exhaustive explanation of such mathematical relationship would require the introduction of tools that are out of the scope of this paper. Here, it suffices to say that some of the results obtained for the VC-dim can be derived using the same theory, on Betti numbers and Pfaffian functions, that we will use in our proofs (see [45]). On the other hand, in order to clarify practical relationships, it is worth pointing out that the VC-dim provides a way of measuring the generalization capability of a classifier. Current available results show that the VC-dim is $O(p^2)$, for networks with $\arctan(\cdot)$ or $\tanh(\cdot)$ activation, where $p$ is the number of parameters [45]–[47]. In addition, for generic sigmoidal activation functions,[14] a lower bound $\Omega(p^2)$ has been devised, so that the upper bound $O(p^2)$ for $\arctan(\cdot)$ and $\tanh(\cdot)$ is optimal. Thus, the VC-dim is polynomial with respect to the number of parameters and, as a consequence, it is polynomial also in the number of hidden units. These bounds do not depend on the number of layers in the network, thus suggesting that, in practice, the depth of a network has a larger impact on the complexity of the implemented function than on its generalization capability. In other words, using the same amount of resources, deep networks are able to face more complex applications without loosing their generalization capability.

Let us now discuss the limits of the presented theorems. Further limits will be discussed in Subsection I of the Appendix. First, as mentioned in Section I, the claim that the sum of the Betti numbers related to sets implemented by common multilayer networks grows exponentially in the number of neurons if the network is deep, whereas it grows polynomially for a three-layer architecture, has been proved only for some activation functions. The most evident limitation of our results lies in the lack of a tight and polynomial upper bound for three-layer networks with hyperbolic tangent activation (Table I). Such a limitation may appear surprising, since a bound has been derived for the $\arctan(\cdot)$ function, which is, intuitively, very similar to $\tanh(\cdot)$. Actually, the difference between $\arctan(\cdot)$ and $\tanh(\cdot)$ is more technical than fundamental. In the Appendix, it will be explained how the problem of devising a tighter bound for three-layer networks with $\tanh(\cdot)$ activation is strictly related to a well known and unsolved problem on the number of the solutions of polynomial systems. Also in that context, it will be deepen how the general goal of improving the bounds reported in Table I is related to unsolved challenging problems in mathematics as well.

In addition, we have considered only networks with Pfaffian activation functions. Though this is a general feature for the activations commonly used in Backpropagation networks, such an assumption is fundamental and cannot be easily removed, because it ensures that the Betti numbers of the set implemented by the network are finite. Actually, in [48], it has been shown that the VC-dim is infinite for networks with one input, and two hiddens, with activation function $\sigma(a) = \pi^{-1} \arctan(a) + (\alpha(1+a^2))^{-1}\cos(a) + 1/2$, where $\alpha > 2\pi$. In fact, the presence of the sinusoidal component allows to construct a single input network $\mathcal{N}$ for which $f_{\mathcal{N}}$ has an infinite number of roots. Obviously, in this case, also $B(S_n)$ is infinite. However, such a degenerate behavior is not possible when the activation function is Pfaffian, since a system of $n$ Pfaffian equations in $n$ variables has a finite number of solutions [49].

Also, one may wonder whether there are alternatives to the sum of the Betti numbers, $B(S_{\mathcal{N}}) = \sum_i b_i(S_{\mathcal{N}})$, for measuring the complexity of the regions realized by a classifier. Actually, it is easy to see that other measures exist and we must admit that our choice of using $B(S_{\mathcal{N}})$ is due both to the availability of mathematical tools suitable to derive a set of useful bounds and to the fact that it provides a reasonable way of distinguishing topological complexity details. Actually, $b_0(S_{\mathcal{N}})$ (the number of connected regions of $S_{\mathcal{N}}$) is a possible alternative to $B(S_{\mathcal{N}})$, even if it captures less information on the set $S_{\mathcal{N}}$,[15] while we are not aware of any tighter bound that

[10] The parity function, $f : \{0,1\}^n \to \{0,1\}$, is defined as $f(x) = 1$, if $\sum_{i=1}^n x_i$ is odd, and $f(x) = 0$, if $\sum_{i=1}^n x_i$ is even.

[11] A sum-product network consists of neurons that either compute the product or a weighed sum of their inputs.

[12] Thresholds circuits are multilayer networks with threshold activation functions, i.e., $\sigma(a) = 1$, if $a \geq 0$, and $\sigma(a) = 0$, if $a < 0$. Interestingly, threshold circuits are more powerful than logic networks and allow to implement the parity function by a two–layer network with $n$ threshold gates [43].

[13] See [44] for a definition of monotone function, in this case.

[14] More precisely, this result holds if the activation function $\sigma$ has left and right limits and it is differentiable at some point $a$, where also $\sigma'(a) \neq 0$ holds.

[15] For example, using $B(S_{\mathcal{N}})$, we can recognize that the region representing the character $B$ is more complex than the region $D$, which, in turn, is more complex than the region $I$, whereas all the regions have the same complexity according to $b_0(S_{\mathcal{N}})$.

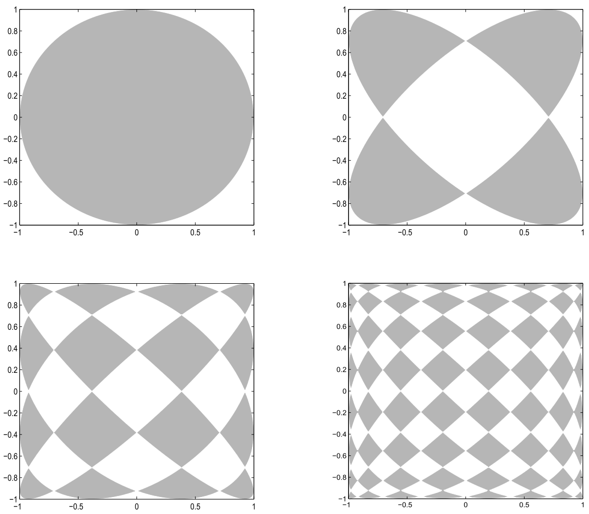


Fig. 2. Plots of the set $S_f$, for $f = g$ (top left), $f = g \circ t$ (top right), $f = g \circ t_2$ (bottom left), and $f = g \circ t_3$ (bottom right), where $g = 1 - \|x\|^2$, $t(x) = [1 - 2x_1^2, 1 - 2x_2^2]$, $t_2 = t \circ t$, and $t_3 = t \circ t \circ t$ hold.

can be derived using $b_0$. On the other hand, it is not difficult to design a measure that captures the fact that the character M looks more complex than the character I, a difference which is not recognized by $B(S_{\mathcal{N}})$, but then it becomes challenging to derive useful bounds using such a measure.

Finally, it may help to provide an intuitive explanation of the possible advantages of deep architectures. First, notice that the $k$th hidden layer of a feedforward network realizes a function that correlates the outputs of the neurons in layer $k - 1$ with the inputs of the neurons in layer $k + 1$. Such a correlation can be represented as a map, so that the global function implemented by a deep network results in a composition of several maps, whose number depends on the number of its hidden layers. On the other hand, the function composition mechanism intuitively allows to replicate the same behavior on different regions of the input space. Without providing a formal definition of such a statement, let us illustrate this concept with an example. Consider the composition $f = g \circ t$, with $g : D \to I\!R$, $t : D \to D$, defined on the domain $D \subseteq I\!R^n$. In addition, let us assume that there exist $m$ sets, $A_1, \ldots, A_m \subseteq D$, such that $t(A_i) = D$, $1 \leq i \leq m$. We can observe that the number of connected regions $b_0(S_f)$ of the set $S_f = \{x | f(x) \geq 0\}$ is at least $m$ times $b_0(S_g)$, where $S_g = \{x | g(x) \geq 0\}$. In fact, intuitively, $f$ behaves on each $A_i$ as $g$ behaves on the whole domain $D$. In addition, this argument can be extended to the case when $g$ is composed with $t$ several times, i.e., $f = g \circ t_k$, where $t_k = t \circ t \circ \ldots \circ t$ for $k$ occurrences of $t$. In this case, the ratio $b_0(S_f)/b_0(S_g)$ is at least $m^k$. For example, Fig. 2 shows the set $S_{g \circ t}$, when $g = 1 - \|x\|^2$, $t(x) = [1 - 2x_1^2, 1 - 2x_2^2]$, and $D = \{x | -1 \leq x_1 \leq 1, -1 \leq x_2 \leq 1\}$. Since $t$ maps four distinct regions to $D$, then $S_g$ is composed of a single connected region, $S_{g \circ t}$ of four connected regions, $S_{g \circ t_2}$ of 16 regions, $S_{g \circ t_3}$ of 64 regions, and so on.

Therefore, we can intuitively conclude that deep networks are able to realize, with few resources, functions that replicate a certain behavior in different regions of the input space. Obviously, here the concept of replicating a behavior must be interpreted in a very broad sense, since the hidden layers of a deep network can approximate any function.

## V. Conclusions

In this paper, we proposed a new measure, based on topological concepts, to evaluate the complexity of functions implemented by neural networks. The measure has been used for comparing deep and shallow feedforward neural networks. It has been shown that, with the same number of hidden units, deep architectures can realize maps with a higher complexity with respect to shallow ones. The above statement holds for networks with both arctangent and polynomial activation functions. This paper analyzes the limits of the obtained results and describes technical difficulties, which prevented us from extending them to other commonly used activation functions. An informal discussion on the practical differences between deep and shallow networks is also included.

Interestingly, the proposed complexity measure provides a tool that allows us to study connectionist models from a new perspective. Actually, it is a future matter of research the application of the proposed measure to more complex architectures, such as convolutional neural networks [9], recurrent neural networks [13], [50], and neural networks for graphs [14], [15]. Hopefully, this analysis will help in comparing the peculiarities of different models and in disclosing the role of their architectural parameters.

## Appendix

In this appendix, we collect the proofs of the presented theorems along with some comments on future research, aimed at improving the given bounds. Such proofs exploit an important property of Pfaffian functions, according to which the solutions of systems of equalities and inequalities based on Pfaffian functions define sets whose Betti numbers are finite, and can be estimated by the format of the functions themselves. Therefore, let us start this section by introducing some theoretical properties of Pfaffian varieties.

Formally, a Pfaffian variety is a set $\mathcal{V} \subset I\!R^n$ defined by a system of equations based on Pfaffian functions $g_1, \ldots, g_t$, i.e., $\mathcal{V} = \{x \in \mathcal{U} | g_1(x) = 0, \ldots, g_t(x) = 0\}$, where $\mathcal{U} \subset I\!R^n$ is the connected component on which $g_1, \ldots, g_t$ are defined. A semi-Pfaffian variety $S$ on $\mathcal{V}$ is a subset of $\mathcal{V}$ defined by a set of sign conditions (inequalities or equalities) based on Pfaffian functions $f_1, \ldots, f_s$, i.e., $S = \{x \in \mathcal{U} | f_1(x)\mathcal{R}_1 0, \ldots, f_s(x)\mathcal{R}_s 0\}$, where, for each $i$, $\mathcal{R}_i$ represents a partial order relation in $\{<, >, \leq, \geq, =\}$. For our purposes, we will use the following theorem, due to Zell [51].

*Theorem 8:* Let $S$ be a compact semi-Pfaffian variety in $\mathcal{U} \subset I\!R^n$, given on a compact Pfaffian variety $\mathcal{V}$, of dimension $n'$, defined by $s$ sign conditions on Pfaffian functions. If all the functions defining $S$ have complexity at most $(\alpha, \beta, \ell)$, then the sum of the Betti numbers satisfies

$$B(S) \in s^{n'} 2^{(\ell(\ell-1))/2} O\left((n\beta + \min(n, \ell)\alpha)^{n+\ell}\right). \qquad (3)$$

Notice that the constraint on the compactness of $\mathcal{U}$ and $\mathcal{V}$ can actually be removed without affecting the bound, as clarified in [52]. In addition, in this paper, Theorem 8 will be used with $\mathcal{U} = I\!R^n$ being the input domain of the network,[16] and with $S$ being the positive set defined by the network, i.e., $S = S_{\mathcal{N}} = \{x \mid f_{\mathcal{N}}(x) \geq 0\}$. In this case, the term $s^{n'}$ in (3) can be ignored, since $s = 1$.

Finally, it is worth noting that a bound is available also for the sum of the Betti numbers of semialgebraic sets, i.e., sets defined by inequalities involving polynomials [53].

*Theorem 9:* Let $S$ be the set defined by $s$ inequalities, $q_1(x) \geq 0, \ldots, q_s(x) \geq 0$, where the $q_i$ are polynomials in $n$ real variables. Let $d$ be the sum of the degrees of the $q_i$. Then

$$B(S) \leq \frac{1}{2}(2+d)(1+d)^{n-1} \tag{4}$$

holds.

In the following, the proofs of the theorems are reported, together with some lemmas that are functional to such proofs.

### A. Proof of Theorem 1

The next lemma defines the exact format of the function implemented by a three-layer network with arctangent activation.

*Lemma 1:* Let $\mathcal{N}$ be a three-layer neural network, having one output and $h$ hidden neurons, with arctangent activation. Then, the function $f_{\mathcal{N}}(x)$, implemented by $\mathcal{N}$, is Pfaffian, with complexity $(2h, 2, 2)$.

*Proof:* Since $d(\arctan(a))/da = 1/(1+a^2)$

$$\frac{\partial f_{\mathcal{N}}(x)}{\partial x_k} = \sum_{i_1=1}^{h} \frac{w^2_{1,i_1} w^1_{i_1,k}}{1 + \left(\sum\limits_{i_0=1}^{n} w_{i_1,i_0} x_{i_0} + b^1_{i_1}\right)^2} = \frac{Q(x)}{T(x)}$$

where $Q$ and $T$ are polynomials in $x_1, \ldots, x_n$, of degree smaller than $2(h-1)$ and $2h$, respectively.[17]
Let us now consider the sequence $(f_1, f_2)$, with $f_1(x) = Q(x)$ and $f_2(x) = 1/T(x)$, which is a Pfaffian chain, with degree $\alpha = 2h$ and length $\ell = 2$. Actually, $f_1$ is straightforwardly Pfaffian, since it is a polynomial, whereas for $f_2$ the following equation holds:

$$\frac{\partial f_2(x)}{\partial x_j} = -\frac{T_j(x)}{T(x)^2} = -T_j(x) \cdot (f_2(x))^2$$

being $T_j$ the partial derivative of $T$ with respect to $x_j$.
Finally, since $\partial f_{\mathcal{N}}(x)/\partial x_k$ can be computed as a polynomial in $f_1(x)$ and $f_2(x)$, namely $\partial f_{\mathcal{N}}(x)/\partial x_k = f_1(x) \cdot f_2(x)$, then $f_{\mathcal{N}}$ is a Pfaffian function with complexity $(2h, 2, 2)$. ■

*Proof of Theorem 1:* Theorem 1 is a direct consequence of Theorem 8, which provides a bound for Pfaffian functions, and Lemma 1. ■

### B. Proof of Theorem 2

By Eq. (1), it can be easily verified that if the activation function of a three-layer network is a polynomial of degree $r$, also the function $f_{\mathcal{N}}(x)$, implemented by the network, is a polynomial with the same degree. Thus, Theorem 9 can be applied to $f_{\mathcal{N}}(x)$, with $d = r$, proving the thesis. ■

### C. Proof of Theorem 3

The first Betti number $b_0(S_{\mathcal{N}})$ equals the number of intervals in $I\!R$ where $f_{\mathcal{N}}(x)$ is nonnegative. Thus, $b_0(S_{\mathcal{N}})$ can be computed from the number of the roots of $f_{\mathcal{N}}$ and, in turn, according to the intermediate value theorem, from the number of extreme points (maxima and minima) of $f_{\mathcal{N}}$. More precisely, $f_{\mathcal{N}}(x)$ has at most $r+1$ roots, which define $r+2$ intervals where $f_{\mathcal{N}}(x)$ has constant sign. Since, the number of intervals where $f_{\mathcal{N}}(x)$ is nonnegative is at most a half of the total number of intervals, then

$$b_0(S_{\mathcal{N}}) \leq \left\lceil \frac{r+2}{2} \right\rceil \tag{5}$$

holds, where $\lceil \cdot \rceil$ is the ceil operator. On the other hand, from Lemma 1, $\partial f_{\mathcal{N}}(x)/\partial x = Q(x)/T(x)$, where $Q$ has degree at most $r = 2(h-1)$. Thus, $f_{\mathcal{N}}(x)$ has at most $2(h-1)$ extreme points, which, along with Eq. (5), proves the thesis. ■

### D. Proof of Theorem 4

The strategy used to prove Theorem 4 is similar to that adopted for Theorem 1 and requires to first compute the bounds on the complexity of the Pfaffian function implemented by a multilayer perceptron.

*Lemma 2:* Let $\sigma : I\!R \to I\!R$ be a function for which there exist a Pfaffian chain $c = (\sigma_1, \ldots, \sigma_\ell)$ and $\ell + 1$ polynomials, $Q$ and $P_i$, $1 \leq i \leq \ell$, of degree $\beta$ and $\alpha$, respectively, such that

$$\frac{d\sigma_i(a)}{da} = P_i(a, \sigma_1(a), \ldots, \sigma_i(a)), \quad 1 \leq i \leq \ell \tag{6}$$

$$\sigma(a) = Q(\sigma_1(a), \ldots, \sigma_\ell(a)). \tag{7}$$

Let $f_{\mathcal{N}}(x)$ be the function implemented by a neural network with $n$ inputs, one output, $l \geq 1$ hidden layers, and $h$ hidden units with activation $\sigma$. Then, $f_{\mathcal{N}}(x)$ is Pfaffian with complexity bounded by $(\bar{\alpha}, \beta, h\ell)$, where $\bar{\alpha} = (\alpha + \beta - 1 + \alpha\beta)l + \alpha\beta$, in the general case, and $\bar{\alpha} = (\alpha + \beta - 1)l + \alpha$, if, $\forall i$, $P_i$ does not depend directly on $a$, i.e., $P_i = P_i(\sigma_1(a), \ldots, \sigma_i(a))$.

*Proof:* To prove the lemma, we first show that the activation level

$$a^d_i = \sum_{j=1}^{h_{d-1}} w^d_{i,j} o^{d-1}_j + b^d_i \tag{8}$$

of the $i$th neuron in layer $d$, is a Pfaffian function in the chain $s^d$, constructed by applying all the $\sigma_i$, $1 \leq i \leq \ell$, on all the activation levels $a^k_j$ (of neuron $j$ in layer $k$) up to layer $d-1$

$$\begin{aligned} s^d = \big(&\sigma_1(a^1_1), \sigma_2(a^1_1), \ldots, \sigma_\ell(a^1_1), \ldots \\ &\ldots, \sigma_1(a^{d-1}_{h_{d-1}}), \sigma_2(a^{d-1}_{h_{d-1}}), \ldots, \sigma_\ell(a^{d-1}_{h_{d-1}})\big). \end{aligned}$$

[16] Formally, $I\!R^n$ is the special variety $U = \{x \mid 0 = 0\}$.

[17] Actually, $Q = \sum_{t=1}^{h} N_t(x) \prod_{j \neq t} D_j(x)$ and $T = \prod_{j=1}^{h} D_j(x)$, where $N_t(x) = w^2_{1,t} w^1_{t,k}$, $D_j(x) = 1 + (\sum_{i_0=1}^{n} w_{j,i_0} x_{i_0} + b^1_j)^2$ hold.

Notice that the above hypothesis straightforwardly implies that $f_{\mathcal{N}}(x)$ is Pfaffian and that its length is $h\ell$, since the output of the network is just the activation level of the unique output neuron in layer $l+1$, i.e., $f_{\mathcal{N}}(x) = a_1^{l+1}$.

In fact, for $d \geq 2$, from Eqs. (7) and (8) it follows:

$$a_i^d = \sum_{j=1}^{h_{d-1}} w_{i,j}^d \, Q\left(\sigma_1\left(a_j^{d-1}\right), \dots, \sigma_\ell\left(a_j^{d-1}\right)\right) + b_i^d \tag{9}$$

so that $a_i^d$ is a polynomial of degree $\beta$ in the chain $s^d$.

Thus, it remains to prove that $s^d$ is a Pfaffian chain, i.e., the derivative of each function in $s^d$ can be defined as a polynomial in the previous functions of the chain and in the input $x$. Actually, for each $i, j, d, k$

$$\begin{aligned}\frac{\partial \sigma_i(a_j^d)}{\partial x_k} &= \left.\frac{d\sigma_i(a)}{da}\right|_{a=a_j^d} \frac{\partial a_j^d}{\partial x_k}\\ &= P_i\left(a_j^d, \sigma_1\left(a_j^d\right), \dots, \sigma_i\left(a_j^d\right)\right) \frac{\partial a_j^d}{\partial x_k}\end{aligned} \tag{10}$$

holds, due to the chain rule. The first term in the right part of (10) is a polynomial with respect to the elements of the chain and the input. In fact, based on the previous discussion on $a_j^d$ and on the hypothesis that $P_i$ is a polynomial of degree $\alpha$ in its variables, it follows that $P_i(a_j^d, \sigma_1(a_j^d), \dots, \sigma_i(a_j^d))$ has degree $\beta\alpha$, in the general case, whereas it has degree $\alpha$ if $P_i$ does not depend directly on $a_j^d$. Let us now expand the term $\partial a_j^d / \partial x_k$, by iteratively applying the chain rule

$$\begin{aligned}\frac{\partial a_j^d}{\partial x_k} &= b_j^d + \sum_{r=1}^{h_{d-1}} w_{j,r}^d \frac{\partial o_r^{d-1}}{\partial x_k}\\ &= b_j^d + \sum_{r=1}^{h_{d-1}} w_{j,r}^d \left.\frac{d\sigma(a)}{da}\right|_{a=a_r^{d-1}} \frac{\partial a_r^{d-1}}{\partial x_k}\\ &= b_j^d + \sum_{r=1}^{h_{d-1}} w_{j,r}^d \left.\frac{d\sigma(a)}{da}\right|_{a=a_r^{d-1}}\\ &\quad \times \left(b_r^{d-1} + \sum_{s=1}^{h_{d-2}} w_{r,s}^{d-1} \left.\frac{d\sigma(a)}{da}\right|_{a=a_s^{d-2}} \frac{\partial a_s^{d-2}}{\partial x_k}\right).\\ &= \cdots\end{aligned}$$

Therefore, $\partial a_j^d / \partial x_k$ can be considered as a polynomial in the derivatives $d\sigma(a)/da|_{a=a_{r_t}^t}$, $1 \leq t \leq d-1$, where $r_1, \dots, r_{d-1}$ is a sequence of neuron indexes. The highest degree terms of such a polynomial are in the form of $\prod_{t=1}^{d-1} d\sigma(a)/da|_{a=a_{r_t}^t}$. In addition

$$\begin{aligned}\left.\frac{d\sigma(a)}{da}\right|_{a=a_j^d} &= \left.\frac{dQ(\sigma_1(a), \dots, \sigma_\ell(a))}{da}\right|_{a=a_j^d}\\ &= \left.\left(\sum_{k=1}^{\ell} \frac{\partial Q(y_1, \dots, y_\ell)}{\partial y_k} P_i(a, y_1, \dots, y_i)\right)\right|_{\substack{1 \leq i \leq \ell\\ y_i = \sigma_i(a_j^d)\\ a = a_j^d}}\end{aligned}$$

holds, so that $d\sigma(a)/da|_{a=a_j^d}$ is polynomial with respect to the chain $s^{d+1}$ and the activation level $a_j^d$, which, in turn, is a polynomial in the chain $s^d$. More precisely, the degree of $d\sigma(a)/da|_{a=a_j^d}$ is $\alpha+\beta-1$, if, $\forall i$, $P_i$ does not depend directly on $a$, and it is $\alpha+\beta-1+\alpha\beta$, in the general case. Collecting the results for $\partial a_j^d/\partial x_k$ and $d\sigma(a)/da|_{a=a_j^d}$, it follows that $\partial a_j^d/\partial x_k$ is a polynomial (with respect to the chain $s_j^d$ and $x$) whose degree is $(\alpha+\beta-1+\alpha\beta)(d-1)$, in the general case, and $(\alpha+\beta-1)(d-1)$, if, $\forall i$, $P_i$ does not depend directly on $a$.

Finally, we can estimate the degree of $\partial\sigma_i(a_j^d)/\partial x_k$ in (10), by the bounds on the degrees of $P_i(a_j^d, \sigma_1(a_j^d), \dots, \sigma_i(a_j^d))$ and $\partial a_j^d/\partial x_k$. It follows that the degree of $\partial\sigma_i(a_j^d)/\partial x_k$ is at most $(\alpha+\beta-1+\alpha\beta)(d-1)+\alpha\beta$, in the general case, and $(\alpha+\beta-1)(d-1)+\alpha$, if $P_i$ does not depend directly on $a$. The bound on $\bar{\alpha}$ follows, observing that the unique output neuron belongs to the $(l+1)$th layer in a network with $l$ hidden layers. ■

*Proof of Theorem 4:* The proof is a direct consequence of Theorem 8 and Lemma 2. In fact, replacing the format given by Lemma 2 into the bound defined in (3), it follows that:

$$\begin{aligned}B(S_{\mathcal{N}}) &\leq 2^{(h\ell(h\ell-1))/2} O\big((n\beta + \min(n, h\ell)\\ &\qquad \times ((\alpha+\beta-1+\alpha\beta)l+\alpha\beta))^{n+h\ell}\big)\\ &\leq 2^{(h\ell(h\ell-1))/2}\\ &\qquad \times O\big((n((\alpha+\beta-1+\alpha\beta)l+\beta(\alpha+1)))^{n+h\ell}\big).\end{aligned}$$

On the other hand, the arctan function is Pfaffian in the chain $c = (\sigma_1, \sigma_2)$, where $\sigma_1(a) = (1+a^2)^{-1}$ and $\sigma_2$ are the arctan itself. In fact, we have

$$\begin{aligned}\frac{d\sigma_1(a)}{da} &= -2a(\sigma_1(a))^2\\ \frac{d\sigma_2(a)}{da} &= \sigma_1(a)\end{aligned}$$

so that the format of arctan is $(3, 1, 2)$. Thus, if the network exploits the arctan function, the format of $f_{\mathcal{N}}(x)$ is $(6l+3, 1, 2h)$. In addition, by Theorem 8

$$\begin{aligned}B(S_{\mathcal{N}}) &\leq 2^{(2h(2h-1))/2} O\big((n + (6l+3)\cdot\min(n, 2h))^{n+2h}\big)\\ &\leq 2^{h(2h-1)} O\big((6nl+4n)^{n+2h}\big)\\ &= 2^{h(2h-1)} O\big((nl+n)^{n+2h}\big).\end{aligned}$$

Finally, tanh is Pfaffian in the chain $c = (\sigma_1)$, where $\sigma_1$ is tanh itself. Actually, we have

$$\frac{d\tanh(a)}{da} = 1 - (\tanh(a))^2.$$

Thus, the format of tanh is $(2, 1, 1)$ and $P_1$ does not depend directly on $a$. Then, according to Lemma 2, the complexity of the function implemented by the network with tanh activation function is $(2l+2, 1, h)$. Then, by Theorem 8

$$\begin{aligned}B(S_{\mathcal{N}}) &\leq 2^{(h(h-1))/2} O\big((n + (2l+2)\cdot\min(n, h))^{n+h}\big)\\ &\leq 2^{(h(h-1))/2} O\big((2nl+3n)^{n+h}\big)\\ &= 2^{(h(h-1))/2} O\big((nl+n)^{n+h}\big).\end{aligned}$$ ■

Interestingly, from the proofs of Lemma 2 and Theorem 4 it is clear that an upper bound can also be derived for generic Pfaffian functions (without the hypothesis on the polynomials defining the chain).

### *E. Proof of Corollary 1*

Since a neural network with $l$ hidden layers and a polynomial activation of degree $r$ implements a polynomial map of degree $r^l$, the proofs straightforwardly follows from (4). ■

### *F. Proof of Theorem 5*

Intuitively, the proof of this theorem is based on the observation that the map implemented by a layered network is the composition of a set of functions, each of which is realized by a single layer. Hence, the proof is carried out in two steps: 1) a function $g$ is described such that its repeated composition produces a map $f = g \circ g \circ \ldots g$, for which $B(\{x \in I\!R^n | f(x) \geq 0\})$ grows exponentially in the number of repetitions of $g$ and 2) then, it is proved that each $g$ can be realized by a single layer in a multilayer network.

*Lemma 3:* Let $g : I\!R \to I\!R$ be a continuous function for which $g(-1) = 1$, $g(0) = -1$, $g(1) = 1$ hold. Let $g_k$ be the function obtained by the composition of $k$ copies of $g$, i.e., $g_1 = g$ and, for $k > 1$, $g_k = g \circ g_{k-1}$. Then, the set $S = \{x \in I\!R | f(x) \geq 0\}$ contains at least $2^{k-1}+1$ disconnected intervals.

*Proof:* To prove the lemma, the following assertion is demonstrated by induction on $k$. Notice that for each $k$, there exist $2^k + 1$ points, $y_1, \ldots, y_{2^k+1}$, in $[-1, 1]$ such that:

1) $y_1, \ldots, y_{2^k+1}$ are arranged in an ascending order, i.e., $y_i < y_{i+1}$, for $1 \leq i \leq 2^k$;
2) $g_k$ is alternatively $-1$ or 1 on such points, i.e., $g_k(y_i) = 1$, if $i$ is odd, and $g_k(y_i) = -1$, if $i$ is even.

Actually, the above statement directly implies the thesis of the lemma since, for each odd $i_1$, there exists an interval which contains $y_{i_1}$ and where $g_k$ is positive and, vice versa, for each even $i_2$ there exists an interval which contains $y_{i_2}$ and where $g_k$ is negative. Thus, the intervals containing the odd indexes, which are disconnected because they are separated by those with an even index, are the intervals required by the lemma. In addition, they are exactly $2^{k-1} + 1$.

The induction hypothesis is straightforwardly proved for $k = 1$, since $g_1 = g$ and $y_1 = -1$, $y_2 = 0$, and $y_3 = 1$. Suppose now that the induction hypothesis holds for $k$ and prove that it holds also for $k+1$. For this purpose, let us consider two consecutive points $y_i, y_{i+1}$, for any odd index $i$, and let us study the behavior of $g_{k+1}$ on $[y_i, y_{i+1}]$. Notice that on the border points of the interval, $g_{k+1}$ is equal to 1, because $g_{k+1}(y_i) = g(g_k(y_i)) = g(1) = 1$ and $g_{k+1}(y_{i+1}) = g(g_k(y_{i+1})) = g(-1) = 1$ hold. In addition, the behavior of $g_{k+1}$ on $[y_i, y_{i+1}]$ resembles the behavior of $g$ in $[-1, 1]$, since the image of $g_k$ on the interval $[y_i, y_{i+1}]$ is $[-1, 1]$. Therefore, there must exist at least another point $z_i \in (y_i, y_{i+1})$, where $g_{k+1}(z_i) = -1$ holds. Finally, the set of points of the induction hypothesis is represented, for $k+1$, by $\left\{y_i, \ldots, y_{2^k+1}\right\} \cup \left(\cup_{i=1}^{2^k} \{z_i\}\right)$, which ends the proof. ■

*Proof of Theorem 5:* Let us consider networks with a unique input neuron. The extension to $n$ inputs is straightforward, supposing null connection weights between inputs $2, \ldots, n$ to the hidden neurons. The proof consists of two steps.

*Step 1:* It is proved that there exists a network $\mathcal{N}$, with one hidden layer, which implements the function $g$ defined in Lemma 3.

*Step 2:* Based on $\mathcal{N}$, a network $\mathcal{N}_l$ is constructed, which implements the composition $g_l = g \circ g \circ \ldots g$.

To prove (1), notice that a network with one hidden layer, containing two hidden units, computes the function

$$g(x) = w_1\sigma(v_1x + b_1) + w_2\sigma(v_2x + b_2) + c$$

where, for simplicity, $v_1$, $v_2$ are the input-to-hidden weights, $w_1$, $w_2$ are the hidden-to-output weights, and $b_1$, $b_2$, and $c$ are the biases of the hidden units and the output unit, respectively. Let us set the weights so that the hypothesis of Lemma 3 holds. Actually, the hypothesis consists of three equalities $g(-1) = 1$, $g(0) = -1$, and $g(1) = 1$, which can be rewritten as

$$A \begin{bmatrix} w_1 \\ w_2 \\ c \end{bmatrix} = B \tag{11}$$

where

$$A = \begin{bmatrix} \sigma(-v_1 + b_1) & \sigma(-v_2 + b_2) & 1 \\ \sigma(b_1) & \sigma(b_2) & 1 \\ \sigma(v_1 + b_1) & \sigma(v_2 + b_2) & 1 \end{bmatrix}, \quad B = [1, -1, 1].$$

Equation (11) is a linear system in the unknowns $[w_1, w_2, c]$, which can be uniquely solved if $A$ is a full rank matrix. On the other hand, it can be easily seen that there exist values of $v_1, v_2, b_1, b_2$ for which the determinant of $A$, $det(A)$, is nonnull. In fact, let us set $v_1 = v_2 = \alpha$, $b_1 = \alpha$, and $b_2 = -\alpha$, for some real value $\alpha$. In this case

$$\begin{aligned}\det(A) = {} & \sigma(0) \cdot \sigma(-\alpha) + \sigma(-2\alpha) \cdot \sigma(2\alpha) + \sigma(\alpha) \cdot \sigma(0) \\ & -\sigma(2\alpha) \cdot \sigma(-\alpha) - (\sigma(0))^2 - \sigma(\alpha) \cdot \sigma(-2\alpha).\end{aligned}$$

In addition, let $r_\sigma = \lim_{\alpha\to\infty} \sigma(\alpha)$ and $l_\sigma = \lim_{\alpha\to-\infty} \sigma(\alpha)$ be the right and the left limit of $\sigma$, respectively. Then

$$\begin{aligned}\lim_{\alpha\to\infty} \det(A) &= l_\sigma\sigma(0) + r_\sigma\sigma(0) - (\sigma(0))^2 - l_\sigma r_\sigma \\ &= (\sigma(0) - l_\sigma)(r_\sigma - \sigma(0)) > 0\end{aligned}$$

where the last inequality holds because $\sigma$ is a monotonic increasing function. Thus, there must exist values of $\alpha$ for which $\det(A) > 0$, which completes the proof of Step 1. On the other hand, let $\mathcal{N}$ be the network that implements $g$ and, for each $l$, let us construct a new network $\mathcal{M}_l$ by concatenating $l$ copies of $\mathcal{N}$. More precisely, the copies are merged so that the output unit of the $i$th copy is just the input unit of the $(i+1)$th copy. Thus, by definition, $\mathcal{M}_l$ implements the function $g_l$ of Lemma 3. In addition, $\mathcal{M}_l$ contains $2l - 1$ hidden layers, a half of which (exactly $l - 1$) exploit the linear activation function used in the input and output neurons of $\mathcal{N}$, whereas the remaining layers exploit the sigmoidal function used in the hidden neurons of $\mathcal{N}$. Interestingly, the layers with the linear activation function (the even layers) can be removed and replaced by direct connections between the layers with sigmoidal activation function (odd layers), without any change to the function implemented by the network. Actually, the output of a neuron on an odd layer $l$, $l \geq 3$, can be represented as a function of the outputs of the units in layer $l - 2$. Since layer $l-1$ adopts a linear activation function, using the notation of (1), we derive

$$\begin{aligned} o_k^l &= \sigma(w_{k,1}^l o_1^{l-1} + w_{k,2}^l o_2^{l-1} + b_k^l) \\ &= \sigma\big(w_{k,1}^l(w_{1,1}^{l-1} o_1^{l-2} + w_{1,2}^l o_2^{l-2} + b_1^{l-1}) \\ &\quad + w_{k,2}^l(w_{2,1}^{l-1} o_1^{l-2} + w_{2,2}^l o_2^{l-2} + b_2^{l-1}) + b_k^l\big) \\ &= \sigma\big((w_{k,1}^l w_{1,1}^{l-1} + w_{k,2}^l w_{2,1}^{l-1}) o_1^{l-2} \\ &\quad + (w_{k,1}^l w_{1,2}^{l-1} + w_{k,2}^l w_{2,2}^{l-1}) o_2^{l-2} \\ &\quad + (w_{k,1}^l b_1^{l-1} + w_{k,2}^l b_2^{l-1} + b_k^l)\big). \end{aligned}$$

Thus, the output $o_k^l$ does not change if the $l$th layer is connected directly to the $(l-2)$th layer and the new weights of the neurons are $\bar{w}_{k,1}^l = w_{k,1}^l w_{1,1}^{l-1} + w_{k,2}^l w_{2,1}^{l-1}$, $\bar{w}_{k,2}^l = w_{k,1}^l w_{1,2}^{l-1} + w_{k,2}^l w_{2,2}^{l-1}$, and $\bar{b}_k^l = w_{k,1}^l b_1^{l-1} + w_{k,2}^l b_2^{l-1} + b_k^l$. Hence, the network $\mathcal{N}_l$, obtained by removing the linear layers from $\mathcal{M}_l$, satisfies the hypothesis. ■

### G. Proof of Theorem 6

In this proof, we adopt the same reasoning as that used for Theorem 5. Actually, it suffices to prove that if $\sigma \in C^3$ in $U$, then there exist $v_1, v_2, b_1, b_2$ such that $\det(A)$ in (11)

$$\begin{aligned} det(A) = &\ \sigma(-v_1 + b_1) \cdot \sigma(b_2) + \sigma(v_1 + b_1) \cdot \sigma(-v_2 + b_2) \\ &+ \sigma(b_1) \cdot \sigma(v_2 + b_2) - \sigma(v_1 + b_1) \cdot \sigma(b_2) \\ &- \sigma(-v_1 + b_1) \cdot \sigma(v_2 + b_2) - \sigma(b_1) \cdot \sigma(-v_2 + b_2) \end{aligned}$$

is nonnull.

Let $b_1, b_2 \in U$ and suppose that $\sigma'(b_1) \neq 0$ and $\sigma''(b_2) \neq 0$, where $\sigma', \sigma''$ denote the first and the second derivative of $\sigma$, respectively. The existence of $b_1, b_2$ with such a property is ensured by the hypothesis on the nonlinearity of $\sigma$. Under these conditions, we can apply the Taylor's theorem to obtain

$$\begin{aligned} \sigma(-v_1 + b_1) &= \sigma(b_1) - v_1\sigma'(b_1) + \frac{1}{2}v_1^2\sigma''(b_1) + O\big(v_1^3\big) \\ \sigma(-v_2 + b_2) &= \sigma(b_2) - v_2\sigma'(b_2) + \frac{1}{2}v_2^2\sigma''(b_2) + O\big(v_2^3\big) \\ \sigma(v_1 + b_1) &= \sigma(b_1) + v_1\sigma'(b_1) + \frac{1}{2}v_1^2\sigma''(b_1) + O\big(v_1^3\big) \\ \sigma(v_2 + b_2) &= \sigma(b_2) + v_2\sigma'(b_2) + \frac{1}{2}v_2^2\sigma''(b_2) + O\big(v_2^3\big). \end{aligned}$$

Then, by inserting these expansions in $\det(A)$ and by simple algebra, it follows that:

$$\begin{aligned} \det(A) = &\ v_1 v_2^2 \sigma'(b_1)\sigma''(b_2) + \\ &+ v_1^2 v_2 \sigma''(b_1)\sigma'(b_2) + O(v_1^3) + O(v_2^3) \end{aligned}$$

which is positive and nonnull if $v_1, v_2$ are sufficiently close to 0 and $v_1, v_2$ have the same sign of $\sigma'(b_1)\sigma''(b_2)$ and $\sigma''(b_1)\sigma'(b_2)$, respectively.[18] ■

### H. Proof of Theorem 7

To prove the theorem, the following lemma is required: it provides a lower bound for the complexity of three-layer networks, varying the number of their hidden neurons.

*Lemma 4:* Let $n$ be a positive integer and $\mathcal{N}'$ be a three-layer neural network with one input, one output, and $h'$ hidden units, having activation function $\sigma$. Suppose that $f_{\mathcal{N}'}(x) \leq 1$, for any $x \in I\!R$, and that there exist $2m-1$ consecutive disjoint intervals $U_1, \ldots, U_{2m-1} \in I\!R$ such that, for any $i$, $1 \leq i \leq 2m-1$:

1) if $i$ is even, $f_{\mathcal{N}'}(x) < -n$ holds for any $x \in U_i$;
2) if $i$ is odd, $f_{\mathcal{N}'}(x) > 0$ holds for any $x \in U_i$.

Then, there exists a network $\mathcal{N}$, with $n$ inputs, one output, and $h = nh'$ hidden units, having activation function $\sigma$, such that

$$b_0(S_{\mathcal{N}}) \geq m^n$$

holds.

*Proof:* Let us consider the network $\mathcal{N}_k'$ obtained by the introduction into $\mathcal{N}$ of $n-1$ input neurons, where the $k$th input represents the original input neuron of $\mathcal{N}'$, whereas all the other inputs are disconnected from the rest of the network. It can easily be seen that the function computed by such a network is $f_{\mathcal{N}_k'}(x) = f_{\mathcal{N}'}(x_k)$, where $x_k$ is the $k$th component of $x$.

Therefore, the network $\mathcal{N}$ can be constructed by merging all the networks $\mathcal{N}_k'$, $1 \leq k \leq n$, where also the corresponding inputs and outputs are merged, so that only $n$ inputs and one output are contained in $\mathcal{N}$; in addition, the sets of the hiddens of $\mathcal{N}_k'$ are joined into a single set of $nh'$ hidden units. In this way, $\mathcal{N}$ computes the function

$$f_{\mathcal{N}}(x) = \sum_{k=1}^{n} f_{\mathcal{N}_k'}(x) = \sum_{k=1}^{n} f_{\mathcal{N}'}(x_k).$$

For any sequence $i_1, \ldots, i_n$ of positive numbers not larger than $2m-1$, let $H_{i_1,\ldots,i_n} = \{x | x_i \in U_i, \ 1 \leq i \leq n\}$ hold. If the sequence contains only odd numbers and $x \in H_{i_1,\ldots,i_n}$ then,

$$f_{\mathcal{N}}(x) = \sum_{k=1}^{n} f_{\mathcal{N}'}(x_k) > 0$$

holds, because, by hypothesis, $f_{\mathcal{N}'}(x_k) > 0$ for any $x_k \in U_i$ with odd $i$. If, on the other hand, the sequence contains at least one even integer, without loss of generality, let us assume that such an integer is the first. Then

$$\begin{aligned} f_{\mathcal{N}}(x) = \sum_{k=1}^{n} f_{\mathcal{N}'}(x_k) = f_{\mathcal{N}'}(x_1) + \sum_{k=2}^{n} f_{\mathcal{N}'}(x_k) \\ < -n + n - 1 < 0 \end{aligned}$$

because, by hypothesis, $f_{\mathcal{N}'}(x_k) \leq 1$ for any $x_k \in I\!R$ and $f_{\mathcal{N}'}(x_k) < -n$ for any $x_k \in U_i$ with even index.

Summing up, the function $f_{\mathcal{N}}$ is positive in the $m^n$ regions $H_{i_1,\ldots,i_n}$ that can be defined choosing the indexes $i_1, \ldots, i_n$ among the odd integers in $[1, 2m-1]$. On the other hand, $f_{\mathcal{N}}$ is negative in the regions $H_{i_1,\ldots,i_n}$, where at least an $i_j$ is even. Since, each region $H_{i_1,\ldots,i_n}$ of the former kind is surrounded

[18] By hypothesis, $\sigma'(b_1)\sigma''(b_2)$ is nonnull. On the other hand, $\sigma''(b_1)\sigma'(b_2)$ may be zero: in such a case, $v_2$ can be either positive or negative.

by regions of the latter kind, it follows that $S_{\mathcal{N}}$ contains at least $m^n$ disconnected sets, i.e., $b_0(S_{\mathcal{N}}) \geq m^n$ holds. ■

*Proof of Theorem 7:* The proof consists of defining a three-layer network, with sigmoidal activation functions, which satisfies the hypothesis of Lemma 4. More precisely, the chosen network has $2m - 1$ hidden units and approximates a staircase[19] function on a set of $2m$ open intervals $U_i \subseteq [i, i+1]$, $1 \leq i \leq 2m$. The staircase function is discontinuous on the points $i$, $1 \leq i \leq 2m$; it is constant and equal to $1/2$ on the intervals $U_i$ for which $i$ is odd, and to $-n-1$ on the intervals for which $i$ is even. Notice that, provided that the above network can be defined, the thesis directly follows from Lemma 4, observing that $m = (h-1)/(2n)$ holds. Actually, such network can be constructed using the same reasoning adopted for proving that neural networks with sigmoidal activation functions are universal approximators for continuous functions on $I\!\!R$ [36]. In fact, we can first observe that any staircase function with $2m - 1$ steps can be approximated by a three-layer neural network with $2m - 1$ hidden nodes, exploiting the Heaviside (step) activation function, and then that the Heaviside activation function can be approximated by a sigmoid function.[20] ■

It is worth mentioning that a tighter lower bound can be obtained by using the fact that the properties of function $f_{\mathcal{N}'}$ in Lemma 4 can be satisfied also by a polynomial of degree $2m - 1$. In addition, a polynomial of degree $2m - 1$ can be approximated on compact intervals by a three-layer network with $m$ hidden units and any nonpolynomial and analytic[21] activation function [36]. Therefore, by adding the hypothesis that the activation function is analytic, the lower bound $b_0(S_{\mathcal{N}}) \geq (h/n)^n$ is obtained.

### *I. Further Comments on the Limits of the Presented Results*

A discussion on Zell's bound (3), which has been used in Theorems 1, 3, and 4, may be of help to understand some peculiarities and limits of the presented results.

First of all, Zell's bound (3) is exponential in the number of inputs $n$ and in the length $\ell$ of the considered Pfaffian function. In Lemma 2, it is proved that the length of the function implemented by a network with a Pfaffian activation depends linearly on the number of its hidden units. Thus, in deep networks, the derived upper bounds are exponential in the number of hidden units and inputs, and, obviously, also with respect to the network depth. On the other hand, the function implemented by three-layer networks with arctan$(\cdot)$ activation has the peculiar property of having a length that does not depend on the number of inputs (see Lemma 1). Such a peculiarity allowed us to derive a polynomial upper bound for such a class of networks. The fact that the peculiarity is not shared with networks having tanh$(\cdot)$ activation explains why a polynomial bound was not derived in this case.

A careful inspection of the presented results suggests that they are prone to be improved, both because most of the upper bounds are not close to the corresponding lower bounds, and because the difference between the upper bounds for networks with arctan$(\cdot)$ and tanh$(\cdot)$ activations are counterintuitive. However, (3) is probably not tight for general Pfaffian functions, and a completely different technique may be required to improve our upper bounds. Actually, Zell's Theorem is based on Khovanskiĭ bounds [49] on the number of the solutions of systems of Pfaffian equations, and it is well known that those bounds can be significantly larger than the actual ones.

An example can help in explaining the importance of Khovanskiĭ's theory and its relationship to the problem we faced in this paper. Let us consider a system of $n$ equations of polynomials in $n$ variables, $x = [x_1, \ldots, x_n]$

$$p_1(x) = 0, \ldots, p_n(x) = 0 \tag{12}$$

where the polynomial $p_i(x)$ contains $m_i$ monomials, i.e., $p_i(x) = \sum_{j=1}^{m_i} t_{i,j}$ and $t_{i,j} = a_{i,j} \prod_{k=1}^{n} x_i^{v_{i,j,k}}$, with parameters $a_{i,j}$ and exponents $v_{i,j,k}$ both belonging to $I\!\!R$. An important problem in mathematics consists in defining a bound on the number of nondegenerated solutions of such a system with respect to the total number of monomials, $m = \sum_i m_i$. When $n = 1$, a tight bound is provided by the Descartes' rule of signs, which states that the number of roots is at most equal to the number of sign changes between consecutive monomials, when the monomials are sorted by their exponents. In the more general case $n > 1$, no tight bound is known. Actually, Khovanskiĭ's theory [49] provides a bound, $2^{(m^2-3m+2)/2}(n+1)^{n-1}$, which is exponential with respect to $m$, but some authors claim that the actual bound might be polynomial. Unfortunately, despite the long time efforts, such a claim has not been proved, nor a counterexample has been found, yet (see [54]).

On the other hand, let us consider the function $f(x) = -\sum_i (p_i(x))^2$; the problem of counting the nondegenerated solutions of the system (12) can be transformed into the problem of computing the number of the connected components of the set $S = \{x | f(x) = 0\}$, i.e., the first Betti number $b_0(S)$ of $S$. In addition, notice that, by the variable substitution $y_i = e^{x_i}$, each term $t_{i,j} = a_{i,j} e^{\sum_k v_{i,j,k} x_i}$ is equal to the output of a hidden neuron in a three-layer network, where the activation function is the exponential $e^x$. By simple algebra, it can be easily shown that, more generally, $f(x) = -\sum_i (p_i(x))^2$ is the output of a three-layer network with exponential activation functions, and where the number of hidden neurons is $h = \sum_i (2m_i + m_i^2)$. Thus, the problem of bounding the solutions of (12) can be reduced to the problem of bounding the number of connected components of the set realized by a three-layer network with a Pfaffian activation function. As a consequence, if we could find a tighter bound for $B(S_{\mathcal{N}})$ for networks with generic Pfaffian activation, then we could also find a solution to another theoretical problem that has been open for a long time.

Obviously, the above discussion provides only an informal viewpoint to understand the difficulty of the

[19] A staircase function is a piecewise constant function with discontinuity points.

[20] More precisely, in this way we obtain a network $\mathcal{N}$ that approximates, up to any degree of precision, the target staircase function, $t : I\!\!R \to I\!\!R$, on the whole $I\!\!R$ except on the discontinuity points. The construction is sufficient for our goal, since the discontinuity points are outside the intervals $U_i$.

[21] A function is analytic if its Taylor series converges. A neural network is a universal approximator, if the activation function of the hidden neurons is not polynomial and analytic in a neighborhood of some point. The functions tanh$(\cdot)$ and arctan$(\cdot)$ satisfy such a property.

considered problem. In fact, the functions implemented by neural networks belong to a subclass of the Pfaffian functions, so that tighter bounds could be more easily obtainable in this case.

## References

[1] Y. Bengio, Y. LeCun, R. Salakhutdinov, and H. Larochelle, *Proc. Deep Learn. Workshop, Found. Future Directions NIPS*, 2007.

[2] H. Lee, M. Ranzato, Y. Bengio, G. Hinton, Y. LeCun, and A. Ng, *Proc. Deep Learn. Unsupervised Feature Learn. Workshop NIPS*, 2010.

[3] K. Yu, R. Salakhutdinov, Y. LeCun, G. E. Hinton, and Y. Bengio, *Proc. Workshop Learn. Feature Hierarchies ICML*, 2009.

[4] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer–wise training of deep networks," in *Proc. Adv. NIPS*, 2007, pp. 153–160.

[5] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[6] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[7] T. S. Lee, D. Mumford, R. Romero, and V. A. F. Lamme, "The role of the primary visual cortex in higher level vision," *Vis. Res.*, vol. 38, nos. 15–16, pp. 2429–2454, 1998.

[8] T. Serre, G. Kreiman, M. Kouh, C. Cadieu, U. Knoblich, and T. Poggio, "A quantitative theory of immediate visual recognition," *Progr. Brain Res.*, vol. 165, pp. 33–56, Feb. 2007.

[9] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*, M. Arbib, Ed. Cambridge, MA, USA: MIT Press, 1995, pp. 255–258.

[10] S. E. Fahlman and C. Lebiere, "The cascade–correlation learning architecture," in *Advances in Neural Information Processing Systems 2*. San Mateo, CA, USA: Morgan Kaufmann, 1990, pp. 524–532.

[11] N. Bandinelli, M. Bianchini, and F. Scarselli, "Learning long–term dependencies using layered graph neural networks," in *Proc. IJCNN*. 2010, pp. 1–8.

[12] I. Castelli and E. Trentin, "Supervised and unsupervised co–training of adaptive activation functions in neural nets," in *Proc. 1st IAPR Workshop PSL*, 2011, pp. 52–61.

[13] J. L. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, no. 2, pp. 179–211, 1990.

[14] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 768–786, Sep. 1998.

[15] F. Scarselli, M. Gori, A.-C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans.Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[16] C. L. Giles, S. Lawrence, and S. Fong, "Natural language grammatical inference with recurrent neural networks," *IEEE Trans. Knowl. Data Eng.*, vol. 12, no. 1, pp. 126–140, Jan./Feb. 2000.

[17] H.-T. Su, "Identification of chemical processes using recurrent neural networks," in *Proc. Amer. Control Conf.*, 1991, pp. 2311–2319.

[18] M. Bianchini, M. Maggini, L. Sarti, and F. Scarselli, "Recursive neural networks for processing graphs with labelled edges: Theory and applications," *Neural Netw.*, vol. 18, no. 8, pp. 1040–1050, 2005.

[19] M. Bianchini, M. Maggini, L. Sarti, and F. Scarselli, "Recursive neural networks learn to localize faces," *Pattern Recognit. Lett.*, vol. 26, no. 12, pp. 1885–1895, 2005.

[20] A. Pucci, M. Gori, M. Hagenbuchner, F. Scarselli, and A.-C. Tsoi, "Investigation into the application of graph neural networks to large–scale recommender systems," *Syst. Sci.*, vol. 32, no. 4, pp. 17–26, 2006.

[21] A.-C. Tsoi, M. Hagenbuchner, and F. Scarselli, "Computing customized page ranks," *ACM Trans. Internet Technol.*, vol. 6, no. 4, pp. 381–414, 2006.

[22] Y. Bengio and O. Delalleau, "Shallow vs. deep sum–products networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 24. 2011, pp. 666–674.

[23] O. Delalleau and Y. Bengio, "On the expressive power of deep architectures," in *Proc. 22nd Int. Conf. Algorithmic Learn. Theory*, 2011, pp. 18–36.

[24] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Learnability and the Vapnik–Chervonenkis dimension," *J. ACM*, vol. 36, no. 4, pp. 929–965, 1989.

[25] G. E. Bredon, *Topology and Geometry, Graduate Texts in Mathematics*. New York, NY, USA: Springer-Verlag, 1993.

[26] R. Rojas, *Neural Networks: A Systematic Introduction*. New York, NY, USA: Springer-Verlag, 1996.

[27] A. Hatcher, *Algebraic Topology*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[28] M. Nakahara, *Geometry, Topology and Physics, (Graduate Student Series in Physics)*, 2nd ed. New York, NY, USA: Taylor & Francis, 2003.

[29] T. Kaczynski, K. Mischaikow, and M. Mrozek, *Computational Homology*, vol. 157. New York, NY, USA: Springer-Verlag, 2004.

[30] R. Ghrist and A. Muhammad, "Coverage and hole–detection in sensor networks via homology," in *Proc. 4th Int. Symp. Inf. Process. Sensor Netw.*, 2005, pp. 254–260.

[31] S. Zimmerman, "Slicing space," *College Math. J.*, vol. 32, no. 2, pp. 126–128, 2001.

[32] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989.

[33] K. I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Netw.*, vol. 2, no. 3, pp. 183–192, 1989.

[34] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, 1991.

[35] J. Park and I. W. Sandberg, "Universal approximation using radial–basis–function networks," *Neural Comput.*, vol. 3, no. 2, pp. 246–257, 1991.

[36] F. Scarselli and A.-C. Tsoi, "Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results," *Neural Netw.*, vol. 11, no. 1, pp. 15–37, 1998.

[37] K. I. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Netw.*, vol. 6, no. 6, pp. 801–806, 1993.

[38] B. Hammer, "Approximation capabilities of folding networks," in *Proc. 7th ESANN*, 1999, pp. 33–38.

[39] F. Scarselli, M. Gori, A.-C. Tsoi, M. Hagenbuchner, and G. Monfardini, "Computational capabilities of graph neural networks," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 81–102, Jan. 2009.

[40] J. Håstad, "Almost optimal lower bounds for small depth circuits," in *Proc. 18th ACM Symp. Theory Comput.*, 1986, pp. 6–20.

[41] A. C. C. Yao, "Separating the polynomial–time hierarchy by oracles," in *Proc. 26th Annu. Symp. Found. Comput. Sci.*, 1985, pp. 1–10.

[42] I. Wegener, *The Complexity of Boolean Functions*. New York, NY, USA: Wiley, 1987.

[43] H.-K. Fung and L. K. Li, "Minimal feedforward parity networks using threshold gates," *Neural Comput.*, vol. 13, no. 2, pp. 319–326, 2001.

[44] J. Håstad and M. Goldmann, "On the power of small–depth threshold circuits," *Comput. Complex.*, vol. 1, no. 2, pp. 113–129, 1991.

[45] M. Karpinski, "Polynomial bounds of VC dimension of sigmoidal and general Pfaffian neural networks," *J. Comput. Syst. Sci.*, vol. 54, no. 1, pp. 169–176, 1997.

[46] P. L. Bartlett and W. Maass, "Vapnik–Chervonenkis dimension of neural nets," in *The Handbook of Brain Theory and Neural Networks*, 2nd ed. M. Arbib, Ed. Cambridge, MA, USA: MIT Press, 2003, pp. 1188–1192.

[47] E. D. Sontag, "VC dimension of neural networks," in *Neural Networks and Machine Learning*, C. M. Bishop, Ed. New York, NY, USA: Springer-Verlag, 1998, pp. 69–95.

[48] E. D. Sontag, "Feedforward nets for interpolation and classification," *J. Comput. Syst. Sci.*, vol. 45, no. 1, pp. 20–48, 1992.

[49] A. G. Khovanskiĭ, *Fewnomials*. vol. 88. Providence, RI, USA: AMS, 1991.

[50] B. A. Pearlmutter, "Learning state space trajectories in recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 263–269, 1989.

[51] T. Zell, "Betti numbers of semi–Pfaffian sets," *J. Pure Appl. Algebra*, vol. 139, no. 1, pp. 323–338, 1999.

[52] T. Zell, "Quantitative study of semi–Pfaffian sets," Ph.D. dissertation, School of Math., Georgia Inst. Technol., Atlanta, GA, USA, 2004.

[53] J. Milnor, "On the Betti numbers of real varieties," *Proc. Amer. Math. Soc.*, vol. 15, no. 2, pp. 275–280, 1964.

[54] F. Sottile, *Real Solutions to Equations from Geometry*, vol. 57, Providence, RI, USA: AMS, 2011.

Authors' photographs and biographies not available at the time of publication.