6-2015

# Why Deep Neural Networks: A Possible Theoretical Explanation

Chitta Baral
*Arizona State University*, baral@asu.edu

Olac Fuentes
ofuentes@utep.edu

Vladik Kreinovich
*University of Texas at El Paso*, vladik@utep.edu



Comments:
Technical Report: UTEP-CS-15-55
To appear in: Martine Ceberio and Vladik Kreinovich (eds.), *Constraint Programming and Decision Making: Theory and Applications*, Springer Verlag, Berlin, Heidelberg.

# Why Deep Neural Networks: A Possible Theoretical Explanation

Chitta Baral[1], Olac Fuentes[2], and Vladik Kreinovich[2]
[1]Department of Computer Science and Engineering
Arizona State University
Tempe, Arizona 85287-5406, USA
baral@asu.edu
[2]Department of Computer Science
University of Texas at El Paso
500 W. University, El Paso, Texas 79968, USA
ofuentes@utep.edu, vladik@utep.edu

**Abstract**

In the past, the most widely used neural networks were 3-layer ones. These networks were preferred, since one of the main advantages of the biological neural networks – which motivated the use of neural networks in computing – is their parallelism, and 3-layer networks provide the largest degree of parallelism. Recently, however, it was empirically shown that, in spite of this argument, multi-layer ("deep") neural networks leads to a much more efficient machine learning. In this paper, we provide a possible theoretical explanation for the somewhat surprising empirical success of deep networks.

## 1 Formulation of the Problem

**Why neural networks.** In spite of all the progress in computer-based recognition algorithms, we humans still perform many recognition tasks much faster (and often much more reliably) than computer programs. And we perform faster in spite of the fact that the fastest of our brain's data processing units – neurons – has reaction time $\approx$10 msec, while computer components operate in nanoseconds. The explanation lies largely in the fact that in the human brain, billion of neurons operate in parallel.

Thus, a natural idea is to speed up computer-based data processing, by simulating the way biological neurons operate. The resulting data processing techniques are known as *artificial neural networks*, or simply *neural networks*, for short; see, e.g., [1].

**Traditionally, neural networks used the smallest possible number of**

**layers.** When we have neurons working in parallel, the computation time is proportional to the number of layers that the signal passes through:

- in each layer, all the processing is done in parallel,
- so, data processing in each layer takes the same time, no matter how many neurons we use.

Most widely used neurons perform two types of operations: a linear combination of inputs $y = w_0 + \sum_{i=1}^{n} w_i \cdot x_i$ and a non-linear transformation $y = s(x)$ for some non-linear *activation function* $s(x)$. Activations functions are usually assumed to be smooth (at least three times differentiable). The most widely used activation function is the sigmoid function $s(x) = \dfrac{1}{1+\exp(-x)}$.

It is known that, for the sigmoid activation function, already 3-layer neurons are universal approximators; see, e.g., [1]. To be more precise, the following class of functions can approximate any continuous function $f(x_1, \ldots, x_n)$ on a given box $[\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_n, \overline{x}_n]$ with a given accuracy $\varepsilon$:

$$y = \sum_{k=1}^{K} W_k \cdot y_k - W_0, \tag{1}$$

where

$$y_k = s(z_k) \tag{2}$$

and

$$z_k = \sum_{i=1}^{n} w_{ki} \cdot x_i - w_{k0}. \tag{3}$$

In such neurons:

- the original signals $x_i$ pass through the first linear layer in which all the values $z_k$ are computed;
- then the second (non-linear) layer computes all the values $y_k$, and
- finally, the third (linear) layer computes the resulting value $y$.

It is also known that 2-layer neural networks do not have the universal approximation property. As a result, 3-layer networks used to be most frequently used.

**Recent successes of deep networks: a mystery.** Recently, it was empirically shown that in many cases, it is beneficial to use "deep" neural networks, i.e., neural networks with a large number of layers; see, e.g., [2, 3, 5, 6, 7]. What is still not clear is why this works better than the more traditional (and seemingly better) 3-layer network.

*Comment.* To be more precise, there are qualitative explanations for this empirical phenomena, but they have not been transformed into a precise result:

- One qualitative explanation is that if we have few neurons on each layer, then we have fewer combinations of weights on each layer, so it is easier to try all such combinations.
- Another qualitative explanation is that when we have several neurons on the same layer, there is a potential duplication of information – since we can have two identical neurons – but neurons on different layers do not lead to duplication.

# 2 Why Deep Neural Networks: Our Explanation

**The universal approximation property of 3-layer networks depends on the choice of the activation function: reminder.** In principle, different activation functions $s(x)$ are used in neural networks. The most important requirement is that the function $s(x)$ should be non-linear: otherwise, we will only be able to represent linear functions.

The universal approximation result for 3-layer networks was originally proved for the sigmoid activation function. A similar result is true for many other activations functions, but it is not true for many other non-linear functions. For example, if we use a non-linear function $s(x) = x^2$, then the network (1)–(3) is only able to compute quadratic functions – and thus, it will not have the universal approximation property.

**Similar results.** Similarly, if we select $s(x)$ to be any polynomial $s(x) = a_0 \cdot x^d + a_1 \cdot x^{d-1} + \ldots + a_{d-1} \cdot x + a_d$, then every function computed by a network (1)–(3) is a polynomial of degree $\le d$, and thus, the corresponding network does not have the universal approximation property either.

A similar negative result holds even if, instead of a 3-layer network, we allow multi-layer networks with the possibility of $\ell$ non-linear layers. Indeed:

- the function computed by the network is a composition of functions corresponding to each layer; and
- the composition $P_1(P_2(x))$ of two polynomials $P_1(x)$ and $P_2(x)$ of degrees $d_1$ and $d_2$ has a degree $d_1 \cdot d_2$.

Thus, if the activation function is a polynomial of degree $d$, and we allow $\ell$ non-linear layers, then each function computed such a network is a polynomial of degree $\le D \stackrel{\text{def}}{=} d^\ell$. Thus, such networks are not universal approximators.

**Why this is important.** Since we are mostly using the sigmoid activation function $s(x)$, why does it matter that something is wrong with other functions $s(x)$? At first glance, the above negative results only emphasize the importance of using the sigmoid activation function.

In the ideal world, yes. However, in reality, no matter how we implement the activation function, whether we implement it in hardware or in software, we

cannot implement is exactly. We can implement the activation function with a certain accuracy. As a result, in a real neural network, instead of the desired sigmoid activation function $s(x)$, we actually have an approximate function $\widetilde{s}(x) \approx s(x)$.

And here lies a problem. It is known (see, e.g., [1]) that an arbitrary continuous function on a box can be approximated, within any given accuracy, by a polynomial. This is not just a theoretical possibility: when a computer computes a standard non-linear function, be it $\sin(x)$ or $\exp(x)$, the most widely use computational algorithms actually compute the sum of the first few terms in the Taylor expansion of the desired function – i.e., actually compute a polynomial, the activation function for which the universal approximation property is lost.

**Conclusion: the usual formulation of the universal approximation property is not fully adequate.** The usual formulation of the universal approximation property assumes that we can implement the exact activation function. In practice, however, we can only implement some approximation to the ideal activation function – and, if we take that into account, that the universal approximation property may be lost.

To study real-life neural networks, it is therefore desirable to come up with a more adequate formulation, that takes into account the fact that an activation function can only be implemented with a certain accuracy. Let us formulate this in precise terms.

**Definition 1.** *By a* $\ell$-layer neural network with activation function $s(x)$ and $n$ inputs*, we mean a (marked) ordered graph whose vertices (called* neurons*) are divided into* $\ell + 1$ *subsets (called* layers*) 0, 1, ..., $\ell$, in such a way that:*

- *the 0-th (*input*) layer has exactly $n$ neurons marked $x_1, \ldots, x_n$;*
- *the last ($\ell$-th) layer has exactly one neuron marked $y$;*
- *an edge from a neuron in the $i$-the layer can only go to a neuron in a $j$-th layer, with $j > i$;*
- *some neurons who have only one incoming edge are marked by $s$; each such neuron applies the activation function $s(z)$ to the output $z$ of the incoming neuron;*
- *for each neuron that is not marked by $s$, each edge going into this neuron is marked by a real number $w_i$ and the neuron itself is marked by a number $w_0$; this neuron computes the value $\sum_i w_i \cdot y_i - w_0$, where $y_i$ are the outputs of the incoming neurons.*

*The markings enable us to compute, layer-by-layer, from Layer 1 to Layer $\ell$, the output of each neuron, until we reach the output of the neuron in the final layer; its output is called the* result of applying the neural network to the inputs $x_1, \ldots, x_n$.

**Definition 2.** *Let $\delta > 0$ be a real number. We say that functions $s(x)$ and $\widetilde{s}(x)$ defined on an interval $[-X, X]$ are* $\delta$-close *if $|\widetilde{s}(x) - s(x)| \le \delta$ for all $x \in [-X, X]$.*

**Definition 3.** *Let $s(x)$ be a given smooth activation function. We say that a class of neural networks with this activation function has the* realistic universal approximation property *if:*

- *for every continuous functions $f(x_1, \ldots, x_n)$ on a box $[\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_n, \overline{x}_n]$,*
- *for every two real numbers $\delta > 0$ and $\varepsilon > 0$, and*
- *for every smooth function $\widetilde{s}(x)$ which is $\delta$-close to $s(x)$,*

*there exists a neural network from this class for which,*

- *when we use the activation function $\widetilde{s}(x)$,*
- *for all inputs from the given box, the result of applying this neural network is $\varepsilon$-close to $f(x_1, \ldots, x_n)$.*

**Proposition 1.** *For any $\ell$ and for any activation function $s(x)$, the class of all $\ell$-layer neural networks does not have the realistic universal approximation property.*

**Proof** follows directly from the fact that we can approximate any function $s(x)$ by polynomials $\widetilde{s}(x)$, and, and we have shown, $\ell$-layer neural networks that use a polynomial activation function do not have the universal approximation property.

**Proposition 2.** *For any nonlinear activation function $s(x)$, the class of all neural networks has the realistic universal approximation property.*

*Comment.* The proof of this result is, in effect, contained in [4], where it is shown that if we do not limit the number of layers, then any non-linear activation function has the universal approximation property.

**Conclusion.** In the idealized case, when we assume that we can implement the activation function exactly, 3-layer networks have the universal approximation property. However, in a more realistic setting, if we take into account that we can only implement the activation function approximately, neither 3-layer networks not network with any fixed number of layer have the corresponding realistic universal approximation property. Thus, to adequately approximate different dependencies, we have to consider networks with many layers – i.e., deep networks. Thus, this theoretical result explains the need for deep neural networks.

# References


[1] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2006.

[2] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why Does Unsupervised Pre-training Help Deep Learning?," *The Journal of Machine Learning Research*, 2010, Vol. 11, pp. 625–660.

[3] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets", *Neural Computation*, 2006, Vol. 18, pp. 1527–1554.

[4] V. Kreinovich, "Arbitrary nonlinearity is sufficient to represent all functions by neural networks: a theorem," *Neural Networks*, 1991, Vol. 4, pp. 381–383.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., "Human-level control through deep reinforcement learning", *Nature*, 2015, Vol. 518, No. 7540, pp. 529–533.

[7] J. Ngiam, A. Khosla, M. Kim, H. Lee, and A. Y. Ng, "Multimodal Deep Learning", *Proceedings of the 28-th International Conference on Machine Learning ICML'2011*, Bellevue, Washington, USA, June 28 – July 2, 2011, pp. 265–272.