Why Deep Neural Networks: Yet Another Explanation

Ricardo Lozano
The University of Texas at El Paso, ralozano3@miners.utep.edu

Ivan Montoya Sanchez
The University of Texas at El Paso, iamontoyasa@miners.utep.edu

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

Why Deep Neural Networks: Yet Another Explanation

Ricardo Lozano, Ivan Montoya Sanchez, and Vladik Kreinovich

Abstract One of the main motivations for using artificial neural networks was to speed up computations. From this viewpoint, the ideal configuration is when we have a single nonlinear layer: this configuration is computationally the fastest, and it already have the desired universal approximation property. However, the last decades have shown that for many problems, deep neural networks, with several nonlinear layers, are much more effective. How can we explain this puzzling fact? In this paper, we provide a possible explanation for this phenomena: that the universal approximation property is only true in the idealized setting, when we assume that all computations are exact. In reality, computations are never absolutely exact. It turns out that if take this non-exactness into account, then one-nonlinear-layer networks no longer have the universal approximation property, several nonlinear layers are needed – and several layers is exactly what deep networks are about.

1 Formulation of the Problem

Neural networks: a brief reminder. A natural way to perform data processing is to simulate how data is processed in a brain.

In a brain, signals are processed by special cells called neurons. In the first approximation, a neuron transforms the inputs $x_1,\ldots,x_n$ into a value

$$y_k(x_1,\ldots,x_n)=s\left(\sum_{i=1}^n w_{ki}\cdot x_i - w_{k0}\right).$$

Ricardo Lozano, Ivan Montoya Sanchez, and Vladik Kreinovich
Vladik Kreinovich
Department of Computer Science, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA,
e-mail: ralozano3@miners.utep.edu, iamontoyasa@miners.utep.edu, vladik@utep.edu

Here, $s(z)$ is a nonlinear continuous function known as an *activation function*.

Output signals become inputs for other neurons, etc. This network of neurons is called a *neural network*.

This is exactly what is implemented in artificial neural networks; see, e.g., [1, 2].

Why neural networks? When a computer program recognizes a face, it performs millions of elementary computational operations. The results still come our fast, since a computer performs billions of operations per second.

A human can also recognize a face in a fraction of a second. The big difference is that it takes milliseconds for a biological neuron to perform any operation.

The reason why such slow devices return the results fast is that the brain is heavily parallel. When we look, millions of neurons simultaneously process different parts of the image.

This time-saving parallelism was one of the main original motivations for neural computing.

(Artificial) neural networks: a brief history. As we have just mentioned, one of the main objectives of a neural network was to speed up computations.

In such a heavily parallel system, the processing time is proportional to the number of computational stages. So, it is desirable to use the smallest possible number of stages.

It turns out that already one nonlinear stage is sufficient, when:

- first, signals go into nonlinear neurons, and
- then a linear combination of neurons' results is returned:

$$y(x_1,\ldots,x_n)=\sum_{k=1}^{K} W_k\cdot y_k(x_1,\ldots,x_n)-W_0.$$

To be more precise, it has been proven that such one-nonlinear-layer networks have the following universal approximation property:

- for every continuous function $f(x_1,\ldots,x_n)$ on a bounded domain ($|x_i|\le B$ for all i) and for every desired accuracy $\varepsilon>0$,
- there exist weights w_{ki} and W_k for which the resulting value $y(x_1,\ldots,x_n)$ is ε-close to $f(x_1,\ldots,x_n)$ for all x_i.

The proof follows, e.g., from Fourier transform. Indeed, Newton's prism experiment showed that every light can be decomposed into pure colors. In mathematical terms, a color is pure when the signal depends on time as a sinusoid. Thus, every continuous function can be represented as a linear combination of sinusoids. This representation is what is known as Fourier transform of a function.

This is the desired result for $s(z)=\sin(z)$. So:

- one nonlinear stage is the fastest, and
- with a single such stage, we can, in principle, represent any dependence.

Thus, for several decades, people mostly used one-nonlinear-layer neural networks.

Puzzling development. Surprisingly, last decades have shown that often, multi-stage ("deep") neural networks are much more efficient; see, e.g., [2]. Many machine learning problems:

- could not solved with traditional ("shallow") networks, but
- were successfully solved by deep ones.

A natural question is: why are deep networks effective?

2 Towards a Possible Explanation

Idea. In the above analysis, we implicitly assumed that all computations are exact. In practice, computational devices are never absolutely accurate – e.g., due to rounding.

For example, in a usual binary computer we cannot even represent 1/3 exactly. As a result, the computed value $3 \cdot (1.0/3)$ is slightly different from 1.

So, we can only implement an activation function approximately, with some accuracy $\delta > 0$. Thus, all we know that the actual neurons apply some function $\widetilde{s}(z)$ which is δ-close to $s(z)$.

How this affects the universal approximation property: we need more than one nonlinear layer. It is known that every continuous function on a bounded domain can be approximated, with any given accuracy, by a polynomial.

This is, by the way, how all special functions like $\exp(x)$, $\sin(x)$, etc. are computed in a computer: they are computed by computing the corresponding approximating polynomial.

So, it is possible that neurons use a polynomial function $\widetilde{s}(z)$. Let d be the order (= degree) of this polynomial, i.e., the largest overall degree of its terms. Then, in a traditional neural network, all the outputs $y_k(x_1,\ldots,x_n)$ are polynomials of order d. Thus, their linear combination $y(x_1,\ldots,x_n)$ is also a polynomial of order d.

It is known that polynomials of fixed order are *not* universal approximators. So, we need more than one non-linear stage (layer).

What if we use two nonlinear layers? What if we have two non-linear stages, so that the outputs y_k of the first later serve as inputs to the second one?

In this case, the resulting function is obtained by applying a polynomial of degree d to polynomials of degree d. This results in a polynomial of degree d^2. Thus, it is also not a universal approximator.

What is use a fixed number of layers? For any fixed number L of layers, we will get polynomials of degree bounded by d^L.

Conclusion. Thus, in this realistic setting:

- to get the universal approximation property,
- we cannot limit the number of layers.

And this is exactly what deep networks are about.

Acknowledgments

This work was supported in part by the National Science Foundation grants:

- 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), and
- HRD-1834620 and HRD-2034030 (CAHSI Includes).

It was also supported:

- by the AT&T Fellowship in Information Technology, and
- by the program of the development of the Scientific-Educational Mathematical Center of Volga Federal District No. 075-02-2020-1478.

The authors are also thankful to all the participants of the 27th Joint NMSU/UTEP Workshop on Mathematics, Computer Science, and Computational Sciences, Las Cruces, New Mexico, April 2, 2022, for valuable suggestions.

References

1. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2006.
2. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, Massachusetts, 2016.