11-1-2021

Why Residual Neural Networks

Sofia Holguin
The University of Texas at El Paso, seholguin2@miners.utep.edu

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

Comments:
Technical Report: UTEP-CS-21-98

Why Residual Neural Networks

Sofia Holguin and Vladik Kreinovich

Abstract In the traditional neural networks, the outputs of each layer serve as inputs to the next layer. It is known that in many cases, it is beneficial to also allow outputs from pre-previous etc. layers as inputs. Such networks are known as residual. In this paper, we provide a possible theoretical explanation for the empirical success of residual neural networks.

1 Formulation of the Problem

What are neural networks: a brief reminder. Lately, neural networks have shown to be the most efficient machine learning tools; see, e.g., [1]. The basic computations unit of a neural network is a *neuron*. It transforms inputs $x_1, \ldots, x_n$ into a value

$$s(a_0 + a_1 \cdot x_1 + \ldots + a_n \cdot x_n) \tag{1}$$

for some constants a_i. Here $s(x)$ is a nonlinear function known as an *activation function*. In a neural network:

- some neurons process the inputs,
- some neurons process the results of other neurons.

Usually, neurons form *layers*:

- neurons from layer 1 process inputs,
- neurons of layer 2 process the results of neurons of layer 1, etc.

Sofia Holguin and Vladik Kreinovich
Department of Computer Science, University of Texas at El Paso
500 W. University, El Paso, TX 79968, USA
e-mail: seholguin2@miners.utep.edu, vladik@utep.edu

In the last layer, usually, we simply compute a linear combination of the signals from the previous layer.

What are residual neural networks. The main idea behind residual neural networks is that each neuron at layer i can use, as inputs:

- not only the outputs of the previous $(i-1)$-st layer,
- but also outputs from the layers before it: $(i-2)$-nd, etc.

Residual neural networks are efficient, but why? Empirically, residual neural networks are often more efficient than the traditional ones; see, e.g., [**?**]. In this paper, we provide a possible theoretical explanation for this efficiency.

2 Our Explanation

Our model. In real life applications, most dependencies are smooth. Functions describing many smooth dependencies can be expanded in Taylor series. In this case, the sum of the first few terms in these Taylor series provides a good approximation to the resulting dependence. This is how most special functions like exp, sin, etc. are usually computed. For example, the exponential function is usually computed as

$$\exp(x) \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!}. \tag{2}$$

The simplest nonlinear approximation is when we take into account only constant, linear, and quadratic terms in the general Taylor expansion. Then, we consider expressions of the type

$$f(x_1, \ldots, x_n) = a_0 + \sum_{i=1}^{n} a_i \cdot x_i + \sum_{i,j=1}^{n} a_{ij} \cdot x_i \cdot x_j. \tag{3}$$

This approximation is what we will consider in our model, both:

- in the description of the function that we want to approximate and
- in description of the activation function.

In both cases, we will ignore cubic and higher order terms, and assume that all these functions are quadratic.

It is sufficient to consider neurons with activation function $s(x) = x^2$. First, we show that in this approximation, we can replace each neuron by a neuron with $s(x) = x^2$. This can be done at the expense of changing the coefficients in the corresponding linear terms $a_0 + a_1 \cdot x_1 + \ldots$

Indeed, any nonlinear quadratic function of one variable $s(x) = a \cdot x^2 + b \cdot x + c$, with $a \neq 0$, can be represented as

$$s(x) = a \cdot \left(x + \frac{b}{2a}\right)^2 + \left(c - \frac{b^2}{4a}\right). \tag{4}$$

Thus, the output

$$y = s(a_0 + a_1 \cdot x_1 + \ldots + a_n \cdot x_n) \tag{5}$$

of this neuron can be computed by the simple quadratic neuron $s(x) = x^2$ as

$$y = a \cdot \left(\left(a_0 + \frac{b}{2a}\right) + a_1 \cdot x_1 + \ldots + a_n \cdot x_n\right)^2 + \left(c - \frac{b^2}{4a}\right). \tag{6}$$

Vice versa, for each nonlinear quadratic expression $s(x) = a \cdot x^2 + b \cdot x + c$, from the formula (4), we conclude that

$$s\left(x - \frac{b}{2a}\right) = a \cdot x^2 + \left(c - \frac{b^2}{4a}\right), \tag{7}$$

thus

$$a \cdot x^2 = s\left(x - \frac{b}{2a}\right) - \left(c - \frac{b^2}{4a}\right), \tag{8}$$

and

$$x^2 = \frac{1}{a} \cdot s\left(x - \frac{b}{2a}\right) - \frac{1}{a} \cdot \left(c - \frac{b^2}{4a}\right). \tag{9}$$

Thus, the output

$$y = (a_0 + a_1 \cdot x_1 + \ldots + a_n \cdot x_n)^2 \tag{9}$$

of the simple quadratic neuron can be computed by the neuron with activation function $s(x)$ as

$$y = \frac{1}{a} \cdot s\left(\left(a_0 - \frac{b}{2a}\right) + a_1 \cdot x_1 + \ldots + a_n \cdot x_n\right) - \frac{1}{a} \cdot \left(c - \frac{b^2}{4a}\right). \tag{10}$$

Because of this equivalence, in the following text, we will consider the simplest quadratic neuron, with activation function $s(x) = x^2$.

In this approximation, one nonlinear layer is sufficient. A general quadratic expression is a linear combination of terms x_i^2, $x_i \cdot x_j$, x_i, and 1. Each of these terms can be computed by a single layer; indeed:

- Each term x_i^2 can be obtained by a single quadratic neuron.
- Each term $x_i \cdot x_j$ can be obtained as

$$\frac{(x_i + x_j)^2 - (x_i - x_j)^2}{4}. \tag{11}$$

- Each term x_i can be obtained as

$$\frac{(x_i + 1)^2 - (x_i - 1)^2}{4}. \tag{12}$$

So one nonlinear layer is sufficient to represent any quadratic expression.

How many neurons we need. Let us denote by k the rank of the matrix a_{ij}. We can use new coordinates $z_1, \ldots, z_n$ in which coordinate axes are proportional to eigenvectors. Then, the given quadratic expression takes the form

$$c_0 + \sum_{i=1}^{n} c_i \cdot z_i + \sum_{i=1}^{k} c_{ii} \cdot z_i^2. \tag{13}$$

When $k < n$, then:

- traditional neural network needs at least $k+1$ neurons, since otherwise it cannot cover terms proportional to z_{k+1}, z_{k+2}, etc., but
- with residual neural network, the above formulas enables us to use only k nonlinear neurons.

This explains why residual neural networks are more efficient.

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 thankful to all the participants of the 26th Annual UTEP/NMSU Workshop on Mathematics, Computer Science, and Computational Science (El Paso, Texas, November 5, 2021) for valuable discussions.

References

1. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, Massachusetts, 2016.