7-1-2022

Why Rectified Power (RePU) Activation Functions Are Efficient in Deep Learning: A Theoretical Explanation

Laxman Bokati
The University of Texas at El Paso, lbokati@miners.utep.edu

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

Joseph Baca
The University of Texas at El Paso, jabaca3@miners.utep.edu

Natasha Rovelli
The University of Texas at El Paso, nrovelli@miners.utep.edu

Comments:
Technical Report: UTEP-CS-22-90

Why Rectified Power (RePU) Activation Functions Are Efficient in Deep Learning: A Theoretical Explanation

Laxman Bokati, Vladik Kreinovich, Joseph Baca, and Natasha Rovelli

Abstract At present, the most efficient machine learning techniques is deep learning, with neurons using Rectified Linear (ReLU) activation function $s(z) = \max(0, z)$, in many cases, the use of Rectified Power (RePU) activation functions $(s(z))^p$ – for some p – leads to better results. In this paper, we explain these results by proving that RePU functions (or their "leaky" versions) are optimal with respect that all reasonable optimality criteria.

1 Formulation of the Problem

Need for machine learning. In many practical situation, we need to transform the input x into the desired output y. For example, we want to predict tomorrow's weather y based on the information x about the weather today and in several past days, or we may want to find the species y of an animal based on its photo x.

In many cases, we do not know an algorithm transforming x into y, but we have several (K) past cases $k = 1, 2, \ldots, K$, in which we know both the inputs $x^{(k)}$ and the corresponding output $y^{(k)}$. Based on this information, we need to design an algorithm $y = f(x)$ that, given an input x, produces a reasonable estimate for the desired output y.

In mathematics, reconstructing a function based on its few values is known as interpolation/extrapolation. In computer science, such a reconstruction is known as *machine learning*.

Laxman Bokati
Computational Science Program, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA, e-mail: lbokati@miners.utep.edu

Vladik Kreinovich, Joseph Baca, and Natasha Rovelli
Department of Computer Science, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA
e-mail: vladik@utep.edu, jabaca3@miners.utep.edu, nrovelli@miners.utep.edu

Deep learning: a brief reminder. At present, the most efficient machine learning technique is *deep learning*; see, e.g., [6]. In this technique, the final computation is performed by a network of *neurons*. Each neuron takes several inputs $i_1, \ldots, i_n$ and transforms them into an output

$$o = s(w_1 \cdot i_1 + \ldots + w_n \cdot i_n + w_0),$$

where:

- w_i are real numbers – that need to be adjusted – and
- $s(z)$ is a continuous non-linear function known as the *activation function*.

Since the usual training of a neural network requires computing derivatives, the activation function should be differentiable – at least everywhere except maybe a few points.

In a neural network, first, the input x to the problem are fed into several neurons, then outputs of these neurons became inputs for other neurons, etc. – and the output of the final neurons is returned as the desired answer y.

Which activation functions should we use. Most successful neural networks use:

- either so-called *Rectified Linear* (ReLU) activation function $s(z) = \max(0, z)$
- or its "Leaky" version $s(z) = z$ for $z > 0$ and $s(z) = a \cdot z$ for $z < 0$, for some value $a \neq 1$ for which $|a| \leq 1$.

Interestingly, several papers have shown that in situations in which the actual dependence $y = f(x)$ is sufficiently smooth, we can get better results if we use a different activation function $s(z) = z^p$ for $z > 0$ and $s(z) = 0$ for $z \leq 0$, for some parameter p. This activation function is known as *Rectified Power* (RePU) activation function; see, e.g., [1, 3, 4, 5, 7, 8, 9, 11, 12]. In some cases, a "Leaky" version of RePU – when $s(z) = z^p$ for $z > 0$ and $s(z) = a \cdot |z|^p$ for $z < 0$ – works even better.

Resulting challenge and what we do in this paper. Several of the above-mentioned papers prove that RePU leads to an almost optimal approximation to smooth functions. However, it is remains unclear whether RePU is only activation function with this property – and whether some other activation function would lead to even better approximations.

In this paper, we analyze this problem from the theoretical viewpoint. Specifically, we prove that, for all reasonable optimality criteria, RePU and Leaky RePU are the only optimal activation functions. To prove this result, we first explain the importance of scale-invariance – in Section 2, and then, in Section 3, prove an auxiliary result – that RePU and Leaky RePU are the only scale-invariant activation functions. In Section 4, we use this auxiliary result to prove the main result about optimality of RePU and Leaky RePU.

2 Importance of Scale-Invariance

Artificial neural networks vs. biological ones. Artificial neural networks largely simulate networks of biological neurons in the brain. In the brain, both the input and the output signals are physical signals, i.e., values of the corresponding physical quantities. For example, z can be a frequency of pulses, or the amplitude, etc.

Computer-based neurons simulate how these signals are processed in the brain. To perform such simulations, we describe both input and output signals by their numerical values.

Importance of scale-invariance. It is important to emphasize that the numerical value of a physical quantity depends on the choice of a measuring unit. For example, we can measure frequency as the number of pulses per second or as number of pulses per minute, and we get different numerical values describing the same frequency: 1 pulse per second is equivalent to 60 pulses per minute. In general, if we change a measuring unit to the one which is $\lambda > 0$ times smaller, then all the numerical results will be multiplied by λ. This is similar to the fact that if we, e.g., replace meters with centimeters – a 100 times smaller measuring unit – then all numerical values of lengths get multiplied by 100, so 1.7 m becomes $1.7 \cdot 100 = 170$ cm.

There is no preferred measuring unit, so it makes sense to require that the transformation described by the activation function $o = s(z)$ should not change if we simply change the measuring unit for the input signal. To be more precise, if we have the relation $o = s(z)$ in the original units, then we should have a similar relation $O = s(Z)$ in the new units, and for each change of units for measuring z, there should be an appropriate selection of the corresponding unit for measuring o. This property is known as *scale-invariance.* Let us describe this property in precise terms.

Definition 1. *We say that a function $o = s(z)$ is* scale-invariant *if for each real number $\lambda > 0$ there exists a value $\mu > 0$ (depending on λ) for which, once we have $o = s(z)$, then we should also have $O = s(Z)$, where $O = \mu \cdot o$ and $Z = \lambda \cdot z$.*

3 First Auxiliary Result: RePU and Leaky RePU Are, In Effect, the Only Scale-Invariant Activation Functions

Discussion. If we simply change the unit for the output to the one which is $c > 0$ time smaller, then, from the purely mathematical viewpoint, we get a new activation function $c \cdot s(z)$. But, of course, from the physical viewpoint, everything remains the same. In this sense, the activation functions $s(z)$ are $c \cdot s(z)$ are equivalent. Similarly, we get physically the same – but mathematically different – function if we change the direction of z, from z to $-z$.

Let us describe this equivalence in precise terms.

Definition 2. *We say that two activation functions $s(z)$ and $S(z)$ are* equivalent *if there exist a constants $c \neq 0$ for which:*

- *either we have $S(z) = c \cdot s(z)$ for all z,*
- *or we have $S(z) = c \cdot s(-z)$ for all z.*

Now, we can formulate our first result.

Definition 3. *By a* Leaky RePU*, we mean a function $s(z) = z^p$ for $z \geq 0$ and $s(z) = a \cdot |z|^p$ for some $p \geq 0$ and for some $a \neq 1$ for which $|a| \leq 1$.*

Proposition 1. *Every continuous non-linear scale-invariant function is equivalent to a Leaky RePU.*

Proof. Scale-invariance means that $o = s(z)$ implies that $O = s(Z)$, i.e., that $\mu(\lambda) \cdot o = s(\lambda \cdot z)$. Substituting $o = s(z)$ into this equality, we get

$$\mu(\lambda) \cdot s(z) = s(\lambda \cdot z). \tag{1}$$

It is known – see, e.g., [2] – that for $z > 0$, every continuous solution to this functional equation has the form $s(z) = A_+ \cdot z^{p_+}$ for some $p_+ > 0$. In this case, $\mu(\lambda) = \lambda^{p_+}$. Similarly, for $z < 0$, we get $s(z) = A_- \cdot z^{p_-}$ for some $p_- > 0$. In this case, $\mu(\lambda) = \lambda^{p_-}$.

Since the function $\mu(\lambda)$ must be the same for positive and negative z, we thus have $p_+ = p_-$. Let us denote the common value of this exponent by $p = p_+ = p_-$. If $A_- > A_+$, then, after the transformation $z \to -z$, we will have $A_- \leq A_+$. Thus, for $c = A_+$ and $a = A_-/A_+$, we get the desired equivalence. The proposition is proven.

Comment. The general proof from [2] is somewhat too complicated to be reproduced here, but since we are interested in differentiable activations functions, we can get a simpler proof of this result. Indeed, since the function $s(z)$ is differentiable, the function

$$\mu(\lambda) = \frac{s(\lambda \cdot z)}{s(z)}$$

is differentiable as well – as the ratio of two differentiable functions. Thus, all the functions in the equality (1) are differentiable. So, we can differentiate both sides of this equality by λ. As a result, we get

$$\mu'(\lambda) \cdot s(z) = z \cdot s'(\lambda \cdot z).$$

In particular, for $\lambda = 1$, we get $p \cdot s(z) = z \cdot s'(z)$, where we denoted $p \overset{\text{def}}{=} \mu'(1)$. Thus, we have

$$p \cdot s = z \cdot \frac{ds}{dz}.$$

In this equality, we can separate the variables if we divide both parts by s and z and multiply both parts by dz, then we get

$$p \cdot \frac{dz}{z} = \frac{ds}{s}.$$

Integrating both parts, we get $p \cdot \ln(z) + C = \ln(s)$, where C is an integration constant. By applying the function $\exp(x)$ to both sides, and taking into account that $\exp(\ln(s)) = s$ and that

$$\exp(p \cdot \ln(z) + C) = \exp(p \cdot \ln(z)) \cdot \exp(C) = (\exp(\ln(z))^p \cdot \exp(C) = \exp(C) \cdot z^p,$$

we get the desired expression $s(z) = \text{const} \cdot z^p$.

4 Main Result: RePU and Leaky RePu Are Optimal Activation Functions

What do we want to select. Two equivalent activation functions represent, in effect, the same neurons, and the same neural networks. Thus, what we to select is not a single activation function, but an equivalence class – i.e., the set of all the activation functions which are equivalent to the given one.

Discussion: what do we mean by optimal? We can have many different optimality criteria. Usually, we select some objective function, and we say that an alternative is optimal if it has the largest possible value of this objective function. However, this is not the only possible formulation of optimality. For example, if it turns out that there are several activation functions whose use leads to the same success rate in determining an animal's specie from a photo, we can select, among these activation functions, the one for which the training time is the shortest. In this case, the optimality criterion is more complex than simply comparing the two value of the objective function: we also need to take into account training times. If this change does not lead to a unique choice, this means that our criterion is not final, we can use the current non-uniqueness to optimize something else.

In general, in all such complex optimality criteria, what we have is an ability to compare two alternative a and b and say that some are better (we will denote it by $a > b$) and some are of the same quality (we will denote it by $a \sim b$). Of course, these comparisons must be consistent: if a is better than b and b is better than c, then we should have a better than c. Thus, we arrive at the following definition.

Definition 3. [10] *Let A be a set; its elements will be called* alternatives*. By an* optimality criterion*, we mean the pair $(>, \sim)$ of binary relations on this set that satisfy the following properties for all $a, b, c \in A$:*

- *if $a > b$ and $b > c$, then $a > c$;*
- *if $a > b$ and $b \sim c$, then $a > c$;*
- *if $a \sim b$ and $b > c$, then $a > c$;*
- *if $a \sim b$ and $b \sim c$, then $a \sim c$;*
- *if $a \sim b$, then $b \sim a$;*
- *$a \sim a$;*
- *if $a > b$ and $b > c$, then $a > c$;*
- *if $a > b$, then we cannot have $a \sim b$.*

Definition 4. *We say that an alternative a is* optimal *with respect to the criterion $(>, \sim)$ if for every $b \in A$, we have either $a < b$ or $a \sim b$.*

Definition 5. *We say that the optimality criterion is* final *if there exists one and only one optimal alternative.*

Finally, since there is no preferred measuring unit for signals, it makes sense to require that the comparison between equivalence classes of activation functions should not change if we simply change the measuring unit for the input signal.

Definition 5. *For each function $s(z)$ and for each $\lambda > 0$, by a λ*-rescaling *$T_\lambda(s)$, we mean a function $s(\lambda \cdot z)$.*

One can easily check that if two functions are equivalent, then their λ-rescalings are also equivalent.

Definition 6. *We say that the optimality criterion on the set S of all equivalence classes of continuous non-linear functions is* scale-invariant *if for every two classes e_1 and e_2 and for every $\lambda > 0$, the following two conditions are satisfied:*

- *if $e_1 > e_2$, then $T_\lambda(e_1) > T_\lambda(e_2)$; and*
- *if $e_1 \sim e_2$, then $T_\lambda(e_1) \sim T_\lambda(e_2)$.*

Proposition 2. *For every final scale-invariant optimality criterion on the set S of all equivalence classes of continuous non-linear functions, every function from the optimal class is equivalent to a Leaky RePU.*

Proof. Let e_{opt} be the optimal class. This means that for every other class e, we have either $e_{\text{opt}} > e$ or $e_{\text{opt}} \sim e$. This is true for every class e, in particular, this is true for the class $T_{1/\lambda}(e)$. Thus, for every e, we have either $e_{\text{opt}} > T_{1/\lambda}(e)$ or $e_{\text{opt}} \sim T_{1/\lambda}(e)$. So, by scale-invariance, we have:

- either $T_\lambda(e_{\text{opt}}) > T_\lambda(T_{1/\lambda}(e)) = e$
- or $T_\lambda(e_{\text{opt}}) \sim T_\lambda(T_{1/\lambda}(e)) = e$.

Thus, by definition of an optimal alternative, the class $T_\lambda(e_{\text{opt}})$ is optimal. However, since the optimality criterion is final, there is only one optimal class, so $T_\lambda(e_{\text{opt}}) = e_{\text{opt}}$.

Thus, the class e_{opt} is scale-invariant. Hence, by Proposition 1, every function from the optimal class is equivalent to Leaky RePU. The proposition is proven.

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), and by the AT&T Fellowship in Information Technology.

It was also supported by the program of the development of the Scientific-Educational Mathematical Center of Volga Federal District No. 075-02-2020-1478, and by a grant from the Hungarian National Research, Development and Innovation Office (NRDI).

References

1. A. Abdeljawad and P. Grosh, *Integral Representations of Shallow Neural Network with Rectified Power Unit Activation Function*, arXiv:2112.11157v1, 2021.
2. J. Aczél and J. Dhombres, *Functional Equations in Several Variables*, Cambridge University Press, 2008.
3. M. Ali and A. Nouy, "Approximation of smoothness classes by deep rectifier networks", *SIAM Journal of Numerical Analysis*, 2021, Vol. 59, No. 6, pp. 3032–3051.
4. C. K. Chui, X. Li, and H. N. Mhaskar, "Neural networks for localized approximation", *Mathemarics of Computation*, 1994, Vol. 63, No. 208, pp. 607–623.
5. C. K. Chui and H. N. Mhaskar, "Deep nets for local manifold learning", *Frontiers in Applied Mathematics and Statistics*, 2018, Vol. 4, Paper 00012.
6. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, Massachusetts, 2016.
7. R. Gribonval, G. Kutyniok, M. Nielsen, and F. Voigtlaender, "Approximation spaces of deep neural networks", *Constructive Approximation*, 2022, Vol. 55, pp. 259–367.
8. B. Li, S. Tang, and H. Yu, "Better approximations of high dimensional smooth functions by deep neural networks with rectified power units", *Communications in Computational Physics*, 2020, Vol. 27, No. 2, pp. 379–411.
9. H. N. Mhaskar, "Approximation properties of a multilayered feedforward artificial neural network", *Advances in Computational Mathematics*, 1993, Vol. 1, No. 1, pp. 61–80.
10. H. T. Nguyen and V. Kreinovich, *Applications of Continuous Mathematics to Computer Science*, Kluwer, Dordrecht, 1997.
11. J. A. A. Opschoor, Ch. Schwab, and J. Zech, *Exponential ReLU DNN expression of holomorphic maps in high dimension*, Swiss Federal Institute of Technology ETH Zürich, Seminar on Applied Mathematics, Technical Report 35, 2019.
12. E. Weinan and Y. Bing, "The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems", *Communications in Mathemaics and Statistics*, 2018, Vol. 6, No. 1, pp. 1–12.