

# Rectifier (neural networks)

In the context of [artificial neural networks](#), the **rectifier** is an [activation function](#) defined as:

$$f(x) = \max(0, x)$$

,

where  $x$  is the input to a neuron. This is also known as a [ramp function](#) and is analogous to [half-wave rectification](#) in electrical engineering. This [activation function](#) was first introduced to a dynamical network by Hahnloser et al. in a 2000 paper in Nature<sup>[1]</sup> with strong [biological](#) motivations and mathematical justifications.<sup>[2]</sup> It has been used in [convolutional networks](#)<sup>[3]</sup> more effectively than the widely used [logistic sigmoid](#) (which is inspired by [probability theory](#); see [logistic regression](#)) and its more practical<sup>[4]</sup> counterpart, the [hyperbolic tangent](#). The rectifier is, as of 2015, the most popular activation function for [deep neural networks](#).<sup>[5]</sup>

A unit employing the rectifier is also called a **rectified linear unit (ReLU)**.<sup>[6]</sup>

A smooth approximation to the rectifier is the [analytic function](#)

which is called the **softplus** function.<sup>[7]</sup> The derivative of softplus is

$$f'(x) = e^x / (e^x + 1) = 1 / (1 + e^{-x})$$

, i.e. the [logistic function](#).

Rectified linear units find applications in [computer vision](#)<sup>[3]</sup> and [speech recognition](#)<sup>[8][9]</sup> using [deep neural nets](#).

## Variants

### Noisy ReLUs

Rectified linear units can be extended to include [Gaussian noise](#), making them noisy ReLUs, giving<sup>[6]</sup>

$$f(x) = \max(0, x + Y)$$

, with

Noisy ReLUs have been used with some success in [restricted Boltzmann machines](#) for computer vision tasks.<sup>[6]</sup>

## Leaky ReLUs

Leaky ReLUs allow a small, non-zero gradient when the unit is not active.  
[9]

Parametric ReLUs take this idea further by making the coefficient of leakage into a parameter that is learned along with the other neural network parameters.<sup>[10]</sup>

Note that for

, this is equivalent to

and thus has a relation to "maxout" networks.<sup>[10]</sup>

## ELUs

Exponential linear units try to make the mean activations closer to zero which speeds up learning. It has been shown that ELUs obtain higher classification accuracy than ReLUs.<sup>[11]</sup>

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ a(e^x - 1) & \text{otherwise} \end{cases}$$

$a$  is a hyper-parameter to be tuned and

is a constraint.

# Advantages

- Biological plausibility: One-sided, compared to the [antisymmetry](#) of [tanh](#).
- Sparse activation: For example, in a randomly initialized network, only about 50% of hidden units are activated (having a non-zero output).
- Efficient gradient propagation: No [vanishing or exploding gradient](#) problems.
- Efficient computation: Only comparison, addition and multiplication.
- Scale-invariant:

$$\max(0, ax) = a \max(0, x)$$

.

For the first time in 2011,<sup>[3]</sup> the use of the rectifier as a non-linearity has been shown to enable training deep [supervised](#) neural networks without requiring [unsupervised](#) pre-training. Rectified linear units, compared to [sigmoid function](#) or similar activation functions, allow for faster and effective training of deep neural architectures on large and complex datasets.

## Potential problems

- Non-differentiable at zero: however it is differentiable anywhere else, including points arbitrarily close to (but not equal to) zero.
- Non-zero centered
- Unbounded
- Dying ReLU problem: ReLU neurons can sometimes be pushed into states in which they become inactive for essentially all inputs. In this state, no gradients flow backward through the neuron, and so the neuron becomes stuck in a perpetually inactive state and "dies." In some cases, large numbers of neurons in a network can become stuck in dead states, effectively decreasing the model capacity. This problem typically arises when the learning rate is set too high. It may be mitigated by using Leaky ReLUs instead.


## See also

- [Softmax function](#)
- [Sigmoid function](#)
- [Tobit model](#)
- [Batch Normalization](#)

## References

1. ^ R Hahnloser, R. Sarpeshkar, M A Mahowald, R. J. Douglas, H.S. Seung (2000). *Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit*. *Nature*. **405**. pp. 947–951.
2. ^ R Hahnloser, H.S. Seung (2001). *Permitted and Forbidden Sets in Symmetric Threshold-Linear Networks*. *NIPS 2001*.
3. ^ <sup>a</sup> <sup>b</sup> <sup>c</sup> Xavier Glorot, Antoine Bordes and [Yoshua Bengio](#) (2011). [Deep sparse rectifier neural networks](#) (PDF). *AISTATS*.
4. ^ [Yann LeCun](#), [Leon Bottou](#), Genevieve B. Orr and [Klaus-Robert Müller](#) (1998). ["Efficient BackProp"](#) (PDF). In G. Orr and K. Müller. *Neural Networks: Tricks of the Trade*. Springer.
5. ^ LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey (2015). "Deep learning". *Nature*. **521** (7553): 436–444.  
[Bibcode:2015Natur.521..436L](#). [PMID 26017442](#).  
[doi:10.1038/nature14539](#).
6. ^ <sup>a</sup> <sup>b</sup> <sup>c</sup> Vinod Nair and [Geoffrey Hinton](#) (2010). [Rectified linear units improve restricted Boltzmann machines](#) (PDF). *ICML*.
7. ^ C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, R. Garcia, NIPS'2000, (2001),[Incorporating Second-Order Functional Knowledge for Better Option Pricing](#).
8. ^ László Tóth (2013). [Phone Recognition with Deep Sparse Rectifier Neural Networks](#) (PDF). *ICASSP*.
9. ^ <sup>a</sup> <sup>b</sup> Andrew L. Maas, Awni Y. Hannun, Andrew Y. Ng (2014). [Rectifier Nonlinearities Improve Neural Network Acoustic Models](#)
10. ^ <sup>a</sup> <sup>b</sup> He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on Image Net Classification". [arXiv:1502.01852](#)<sup>3</sup>

[[cs.CV](#)].

11. ^ Clevert, Djork-Arné; Unterthiner, Thomas; Hochreiter, Sepp (2015). "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". [arXiv:1511.07289](#)  [[cs.LG](#)].