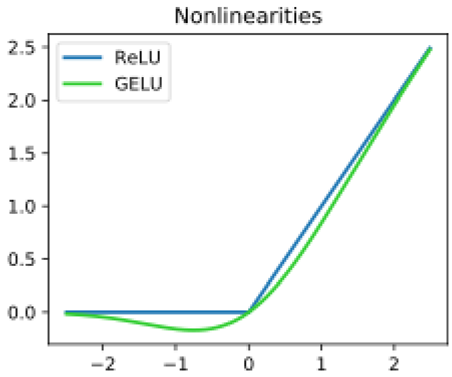
Wikipedia

# Rectifier (neural networks)

In the context of artificial neural networks, the **rectifier** or **ReLU (Rectified Linear Unit) activation function**[1][2] is an activation function defined as the positive part of its argument:

$$f(x) = x^{+} = \max(0, x)$$

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.



Plot of the ReLU rectifier (blue) and GELU (green) functions near *x* = 0

This activation function started showing up in the context of visual feature extraction in hierarchical neural networks starting in the late 1960s.[3][4] It was later argued that it has strong biological motivations and mathematical justifications.[5][6] In 2011 it was found to enable better training of deeper networks,[7] compared to the widely used activation functions prior to 2011, e.g., the logistic sigmoid (which is inspired by probability theory; see logistic regression) and its more practical[8] counterpart, the hyperbolic tangent. The rectifier is, as of 2017, the most popular activation function for deep neural networks.[9]

Rectified linear units find applications in computer vision[7] and speech recognition[10][11] using deep neural nets and computational neuroscience.[12][13][14]

## Contents

## Advantages

- Sparse activation: For example, in a randomly initialized network, only about 50% of hidden units are activated (have a non-zero output).
- Better gradient propagation: Fewer vanishing gradient problems compared to sigmoidal activation functions that saturate in both directions.[7]
- Efficient computation: Only comparison, addition and multiplication.
- Scale-invariant: $\max(0, ax) = a\max(0, x)$ for $a \geq 0$.

Rectifying activation functions were used to separate specific excitation and unspecific inhibition in the neural abstraction pyramid, which was trained in a supervised way to learn several computer vision tasks.[15] In 2011,[7] 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 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, and the value of the derivative at zero can be arbitrarily chosen to be 0 or 1.
- Not zero-centered.
- Unbounded.
- Dying ReLU problem: ReLU (Rectified Linear Unit) 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". This is a form of the vanishing gradient problem. 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, which assign a small positive slope for *x* < 0; however, the performance is reduced.

# Variants

## Piecewise-Linear Variants

### Leaky ReLU

Leaky ReLUs allow a small, positive gradient when the unit is not active.[11]

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ 0.01x & \text{otherwise.} \end{cases}$$

### Parametric ReLU

Parametric ReLUs (PReLUs) take this idea further by making the coefficient of leakage into a parameter that is learned along with the other neural-network parameters.[16]

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ ax & \text{otherwise.} \end{cases}$$

Note that for $a \leq 1$, this is equivalent to

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

and thus has a relation to "maxout" networks.[16]

## Other Non-linear variants

### Gaussian Error Linear Unit (GELU)

GELU is a smooth approximation to the rectifier. It has a non-monotonic "bump" when $x < 0$, and it serves as the default activation for models such as BERT.[17]

$$f(x) = x \cdot \Phi(x),$$

where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution.

This activation function is illustrated in the figure at the start of this article.

### SiLU

The SiLU (Sigmoid Linear Unit) or swish function[18] is another smooth approximation first coined in the GELU paper. [17]

$$f(x) = x \cdot \operatorname{sigmoid}(x)$$

where $\operatorname{sigmoid}(x)$ is the sigmoid function.

### Softplus

A smooth approximation to the rectifier is the analytic function

$$f(x) = \ln(1 + e^x),$$

which is called the *softplus*[19][7] or *SmoothReLU* function.[20] For large negative $x$ it is roughly $ln(1)$ so just above 0, while for large positive $x$ it is roughly $ln(e^x)$ so just above $x$.

A sharpness parameter $k$ may be included:

$$f(x) = \frac{\ln\left(1 + e^{kx}\right)}{k}$$

The derivative of softplus is the logistic function. Starting from the parametric version,

$$f'(x) = \frac{e^{kx}}{1 + e^{kx}} = \frac{1}{1 + e^{-kx}}$$

The logistic sigmoid function is a smooth approximation of the derivative of the rectifier, the Heaviside step function.

The multivariable generalization of single-variable softplus is the LogSumExp with the first argument set to zero:

$$\operatorname{LSE_0}^+(x_1, \dots, x_n) := \operatorname{LSE}(0, x_1, \dots, x_n) = \log(1 + e^{x_1} + \cdots + e^{x_n}).$$

The LogSumExp function is

$$\operatorname{LSE}(x_1, \dots, x_n) = \log(e^{x_1} + \cdots + e^{x_n}),$$

and its gradient is the softmax; the softmax with the first argument set to zero is the multivariable generalization of the logistic function. Both LogSumExp and softmax are used in machine learning.

### ELU

Exponential linear units try to make the mean activations closer to zero, which speeds up learning. It has been shown that ELUs can obtain higher classification accuracy than ReLUs.[21]

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

where $a$ is a hyper-parameter to be tuned, and $a \geq 0$ is a constraint.

The ELU can be viewed as a smoothed version of a shifted ReLU (SReLU), which has the form $f(x) = \max(-a, x)$ given the same interpretation of $a$.

### Mish

The mish function could also be used as a smooth approximation of the rectifier.[18] It is defined as

$$f(x) = x \tanh(\operatorname{softplus}(x))$$

where $\tanh(x)$ is the hyperbolic tangent and $\operatorname{softplus}(\mathrm{x})$ is the softplus function.

Mish is non-monotonic and self-gated.[22] It was inspired by Swish, itself a variant of ReLU.[22]

## See also

- Softmax function
- Sigmoid function
- Tobit model
- Layer (deep learning)

## References

1. Brownlee, Jason (8 January 2019). "A Gentle Introduction to the Rectified Linear Unit (ReLU)" (https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/). *Machine Learning Mastery*. Retrieved 8 April 2021.
2. Liu, Danqing (30 November 2017). "A Practical Guide to ReLU" (https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7). *Medium*. Retrieved 8 April 2021.
3. Fukushima, K. (1969). "Visual feature extraction by a multilayered network of analog threshold elements". *IEEE Transactions on Systems Science and Cybernetics*. **5** (4): 322–333. doi:10.1109/TSSC.1969.300225 (https://doi.org/10.1109%2FTSSC.1969.300225).
4. Fukushima, K.; Miyake, S. (1982). "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". *In Competition and Cooperation in Neural Nets*. Lecture Notes in Biomathematics. Springer. **45**: 267–285. doi:10.1007/978-3-642-46466-9_18 (https://doi.org/10.1007%2F978-3-642-46466-9_18). ISBN 978-3-540-11574-8.
5. Hahnloser, R.; Sarpeshkar, R.; Mahowald, M. A.; Douglas, R. J.; Seung, H. S. (2000). "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit". *Nature*. **405** (6789): 947–951. Bibcode:2000Natur.405..947H (https://ui.adsabs.harvard.edu/abs/2000Natur.405..947H). doi:10.1038/35016072 (https://doi.org/10.1038%2F35016072). PMID 10879535 (https://pubmed.ncbi.nlm.nih.gov/10879535). S2CID 4399014 (https://api.semanticscholar.org/CorpusID:4399014).
6. Hahnloser, R.; Seung, H. S. (2001). *Permitted and Forbidden Sets in Symmetric Threshold-Linear Networks*. NIPS 2001.

7. Xavier Glorot, Antoine Bordes and Yoshua Bengio (2011). *Deep sparse rectifier neural networks* (http://jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf) (PDF). AISTATS. "Rectifier and softplus activation functions. The second one is a smooth version of the first."
8. Yann LeCun, Leon Bottou, Genevieve B. Orr and Klaus-Robert Müller (1998). "Efficient BackProp" (http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf) (PDF). In G. Orr; K. Müller (eds.). *Neural Networks: Tricks of the Trade*. Springer.
9. Ramachandran, Prajit; Barret, Zoph; Quoc, V. Le (October 16, 2017). "Searching for Activation Functions". arXiv:1710.05941 (https://arxiv.org/abs/1710.05941) [cs.NE (https://arxiv.org/archive/cs.NE)].
10. László Tóth (2013). *Phone Recognition with Deep Sparse Rectifier Neural Networks* (http://www.inf.u-szeged.hu/~tothl/pubs/ICASSP2013.pdf) (PDF). ICASSP.
11. Andrew L. Maas, Awni Y. Hannun, Andrew Y. Ng (2014). Rectifier Nonlinearities Improve Neural Network Acoustic Models (https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf).
12. Hansel, D.; van Vreeswijk, C. (2002). "How noise contributes to contrast invariance of orientation tuning in cat visual cortex" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6757721). *J. Neurosci.* **22** (12): 5118–5128. doi:10.1523/JNEUROSCI.22-12-05118.2002 (https://doi.org/10.1523%2FJNEUROSCI.22-12-05118.2002). PMC 6757721 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6757721). PMID 12077207 (https://pubmed.ncbi.nlm.nih.gov/12077207).
13. Kadmon, Jonathan; Sompolinsky, Haim (2015-11-19). "Transition to Chaos in Random Neuronal Networks". *Physical Review X*. **5** (4): 041030. arXiv:1508.06486 (https://arxiv.org/abs/1508.06486). Bibcode:2015PhRvX...5d1030K (https://ui.adsabs.harvard.edu/abs/2015PhRvX...5d1030K). doi:10.1103/PhysRevX.5.041030 (https://doi.org/10.1103%2FPhysRevX.5.041030). S2CID 7813832 (https://api.semanticscholar.org/CorpusID:7813832).
14. Engelken, Rainer; Wolf, Fred; Abbott, L. F. (2020-06-03). "Lyapunov spectra of chaotic recurrent neural networks". arXiv:2006.02427 (https://arxiv.org/abs/2006.02427) [nlin.CD (https://arxiv.org/archive/nlin.CD)].
15. Behnke, Sven (2003). *Hierarchical Neural Networks for Image Interpretation* (https://www.researchgate.net/publication/220688219). Lecture Notes in Computer Science. Vol. 2766. Springer. doi:10.1007/b11963 (https://doi.org/10.1007%2Fb11963). ISBN 978-3-540-40722-5. S2CID 1304548 (https://api.semanticscholar.org/CorpusID:1304548).
16. He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on Image *Net* Classification". arXiv:1502.01852 (https://arxiv.org/abs/1502.01852) [cs.CV (https://arxiv.org/archive/cs.CV)].
17. Hendrycks, Dan; Gimpel, Kevin (2016). "Gaussian Error Linear Units (GELUs)". arXiv:1606.08415 (https://arxiv.org/abs/1606.08415) [cs.LG (https://arxiv.org/archive/cs.LG)].
18. Diganta Misra (23 Aug 2019), *Mish: A Self Regularized Non-Monotonic Activation Function* (https://www.bmvc2020-conference.com/assets/papers/0928.pdf) (PDF), arXiv:1908.08681v1 (https://arxiv.org/abs/1908.08681v1), retrieved 26 March 2022
19. Dugas, Charles; Bengio, Yoshua; Bélisle, François; Nadeau, Claude; Garcia, René (2000-01-01). "Incorporating second-order functional knowledge for better option pricing" (http://papers.nips.cc/paper/1920-incorporating-second-order-functional-knowledge-for-better-option-pricing.pdf) (PDF). *Proceedings of the 13th International Conference on Neural Information Processing Systems (NIPS'00)*. MIT Press: 451–457. "Since the sigmoid *h* has a positive first derivative, its primitive, which we call softplus, is convex."
20. "Smooth Rectifier Linear Unit (SmoothReLU) Forward Layer" (https://software.intel.com/sites/products/documentation/doclib/daal/daal-user-and-reference-guides/daal_prog_guide/GUID-FAC73B9B-A597-4F7D-A5C4-46707E4A92A0.htm). *Developer Guide for Intel Data Analytics Acceleration Library*. 2017. Retrieved 2018-12-04.
21. Clevert, Djork-Arné; Unterthiner, Thomas; Hochreiter, Sepp (2015). "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". arXiv:1511.07289 (https://arxiv.org/abs/1511.07289) [cs.LG (https://arxiv.org/archive/cs.LG)].

22. Shaw, Sweta (2020-05-10). "Activation Functions Compared with Experiments" (https://wandb.ai/shweta/Activation%20Functions/reports/Activation-Functions-Compared-with-Experiments--VmlldzoxMDQwOTQ). *W&B*. Retrieved 2022-07-11.

**This page was last edited on 29 July 2022, at 08:41 (UTC).**