Title: Rectified linear unit
URL: https://en.wikipedia.org/wiki/Rectified_linear_unit
PageID: 37862937
Categories: Category:Artificial neural networks

Supervised learning
Unsupervised learning
Semi-supervised learning
Self-supervised learning
Reinforcement learning
Meta-learning
Online learning
Batch learning
Curriculum learning
Rule-based learning
Neuro-symbolic AI
Neuromorphic engineering
Quantum machine learning
Classification
Generative modeling
Regression
Clustering
Dimensionality reduction
Density estimation
Anomaly detection
Data cleaning
AutoML
Association rules
Semantic analysis
Structured prediction
Feature engineering
Feature learning
Learning to rank
Grammar induction
Ontology learning
Multimodal learning

Apprenticeship learning

Decision trees

Ensembles Bagging Boosting Random forest

Bagging

Boosting

Random forest

k -NN

Linear regression

Naive Bayes

Artificial neural networks

Logistic regression

Perceptron

Relevance vector machine (RVM)

Support vector machine (SVM)

BIRCH

CURE

Hierarchical

k -means

Fuzzy

Expectation–maximization (EM)

DBSCAN

OPTICS

Mean shift

Factor analysis

CCA

ICA

LDA

NMF

PCA

PGD

t-SNE

SDL

Graphical models Bayes net Conditional random field Hidden Markov

Bayes net

Conditional random field

Hidden Markov

RANSAC

k -NN

Local outlier factor

Isolation forest

Autoencoder

Deep learning

Feedforward neural network

Recurrent neural network LSTM GRU ESN reservoir computing

LSTM

GRU

ESN

reservoir computing

Boltzmann machine Restricted

Restricted

GAN

Diffusion model

SOM

Convolutional neural network U-Net LeNet AlexNet DeepDream

U-Net

LeNet

AlexNet

DeepDream

Neural field Neural radiance field Physics-informed neural networks

Neural radiance field

Physics-informed neural networks

Transformer Vision

Vision

Mamba

Spiking neural network

Memtransistor

Electrochemical RAM (ECRAM)

Q-learning

Policy gradient

SARSA

Temporal difference (TD)

Multi-agent Self-play

Self-play

Active learning

Crowdsourcing

Human-in-the-loop

Mechanistic interpretability

RLHF

Coefficient of determination

Confusion matrix

Learning curve

ROC curve

Kernel machines

Bias–variance tradeoff

Computational learning theory

Empirical risk minimization

Occam learning

PAC learning

Statistical learning

VC theory

Topological deep learning

AAAI

ECML PKDD

NeurIPS

ICML

ICLR

IJCAI

ML

JMLR

Glossary of artificial intelligence

List of datasets for machine-learning research List of datasets in computer vision and image processing

List of datasets in computer vision and image processing

Outline of machine learning

v

t

e

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 non-negative part of its argument, i.e., the ramp function :

where x {\displaystyle x} is the input to a neuron . This is analogous to half-wave rectification in electrical engineering .

ReLU is one of the most popular activation functions for artificial neural networks, [3] and finds application in computer vision [4] and speech recognition [5] [6] using deep neural nets and computational neuroscience . [7] [8]

History

The ReLU was first used by Alston Householder in 1941 as a mathematical abstraction of biological neural networks. [9]

Kunihiko Fukushima in 1969 used ReLU in the context of visual feature extraction in hierarchical neural networks. [10] [11] Thirty years later, Hahnloser et al. argued that ReLU approximates the biological relationship between neural firing rates and input current, in addition to enabling recurrent neural network dynamics to stabilise under weaker criteria. [12] [13]

Prior to 2010, most activation functions used were the logistic sigmoid (which is inspired by probability theory ; see logistic regression) and its more numerically efficient [14] counterpart, the hyperbolic tangent . Around 2010, the use of ReLU became common again.

Jarrett et al. (2009) noted that rectification by either absolute or ReLU (which they called "positive part") was critical for object recognition in convolutional neural networks (CNNs), specifically because it allows average pooling without neighboring filter outputs cancelling each other out. They hypothesized that the use of sigmoid or tanh was responsible for poor performance in previous CNNs. [15]

Nair and Hinton (2010) made a theoretical argument that the softplus activation function should be used, in that the softplus function numerically approximates the sum of an exponential number of linear models that share parameters. They then proposed ReLU as a good approximation to it. Specifically, they began by considering a single binary neuron in a Boltzmann machine that takes x {\displaystyle x} as input, and produces 1 as output with probability σ (x) = 1 1 + e − x {\displaystyle \sigma (x)={\frac {1}{1+e^{-x}}}} . They then considered extending its range of output by making infinitely many copies of it X 1 , X 2 , X 3 , … {\displaystyle X_{1},X_{2},X_{3},\dots } , that all take the same input, offset by an amount 0.5 , 1.5 , 2.5 , … {\displaystyle 0.5,1.5,2.5,\dots } , then their outputs are added together as ∑ i = 1 ∞ X i {\displaystyle \sum _{i=1}^{\infty }X_{i}} . They then demonstrated that ∑ i = 1 ∞ X i {\displaystyle \sum _{i=1}^{\infty }X_{i}} is approximately equal to N (log ■ (1 + e x) , σ (x)) {\displaystyle {\mathcal {N}}(\log(1+e^{x}),\sigma (x))} , which is also approximately equal to ReLU ■ (N (x , σ (x))) {\displaystyle \operatorname {ReLU} ({\mathcal {N}}(x,\sigma (x)))} , where N {\displaystyle {\mathcal {N}}} stands for the gaussian distribution .

They also argued for another reason for using ReLU: that it allows "intensity equivariance" in image recognition. That is, multiplying input image by a constant k {\displaystyle k} multiplies the output also. In contrast, this is false for other activation functions like sigmoid or tanh. They found that ReLU activation allowed good empirical performance in restricted Boltzmann machines . [16]

Glorot et al (2011) argued that ReLU has the following advantages over sigmoid or tanh:

ReLU is more similar to biological neurons' responses in their main operating regime.

ReLU avoids vanishing gradients.

ReLU is cheaper to compute.

ReLU creates sparse representation naturally, because many hidden units output exactly zero for a given input.

They also found empirically that deep networks trained with ReLU can achieve strong performance without unsupervised pre-training, especially on large, purely supervised tasks. [4]

Advantages

Advantages of ReLU include:

Sparse activation: for example, in a randomly initialized network, only about 50% of hidden units are activated (i.e. have a non-zero output).

Better gradient propagation: fewer vanishing gradient problems compared to sigmoidal activation functions that saturate in both directions. [4]

Efficiency: only requires comparison and addition.

Scale-invariant (homogeneous , or "intensity equivariance" [16]):

Potential problems

Possible downsides can include:

Non-differentiability at zero (however, it is differentiable anywhere else, and the value of the derivative at zero can be chosen to be 0 or 1 arbitrarily).

Not zero-centered: ReLU outputs are always non-negative. This can make it harder for the network to learn during backpropagation, because gradient updates tend to push weights in one direction (positive or negative). Batch normalization can help address this. [citation needed]

ReLU is unbounded.

Redundancy of the parametrization: Because ReLU is scale-invariant, the network computes the exact same function by scaling the weights and biases in front of a ReLU activation by k {\displaystyle k} , and the weights after by 1 / k {\displaystyle 1/k} . [4]

Dying ReLU: 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 (it "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 and potentially even halting the learning process. This problem typically arises when the learning rate is set too high. It may be mitigated by using "leaky" ReLU instead, where a small positive slope is assigned for x < 0 {\displaystyle x<0} . However, depending on the task, performance may be reduced.

Variants

Piecewise-linear variants

Leaky ReLU (2014) allows a small, positive gradient when the unit is inactive, [6] helping to mitigate the vanishing gradient problem. This gradient is defined by a parameter α {\displaystyle \alpha } , typically set to 0.01–0.3. [17] [18]

The same function can also be expressed without the piecewise notation as:

Parametric ReLU (PReLU, 2016) takes this idea further by making α {\displaystyle \alpha } a learnable parameter along with the other network parameters. [19]

Note that for α ≤ 1 {\displaystyle \alpha \leq 1} , this is equivalent to

and thus has a relation to "maxout" networks. [19]

Concatenated ReLU (CReLU, 2016) preserves positive and negative phase information by returning two values: [20]

Smooth variants

Softplus

A smooth approximation to the rectifier is the analytic function

which is called the softplus (2000) [21] [4] or SmoothReLU function. [22] For large negative x {\displaystyle x} it is roughly ln ■ 1 {\displaystyle \ln 1} , so just above 0, while for large positive x {\displaystyle x} it is roughly ln ■ (e x) {\displaystyle \ln(e^{x})} , so just above x {\displaystyle x} .

This function can be approximated as:

By making the change of variables x = y ln ■ (2) {\displaystyle x=y\ln(2)} , this is equivalent to

A sharpness parameter k {\displaystyle k} may be included:

The derivative of softplus is the logistic function . This in turn can be viewed as 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:

The LogSumExp function is

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 (2015) smoothly allow negative values. This is an attempt 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. [23]

In these formulas, α {\displaystyle \alpha } is a hyperparameter to be tuned with the constraint α ≥ 0 {\displaystyle \alpha \geq 0} .

Given the same interpretation of α {\displaystyle \alpha } , ELU can be viewed as a smoothed version of a shifted ReLU (SReLU), which has the form f (x) = max (– α , x) {\displaystyle f(x)=\max(-\alpha ,x)} .

Gaussian-error linear unit (GELU)

GELU (2016) is a smooth approximation to the rectifier:

where Φ (x) = P (X ■ x) {\displaystyle \Phi (x)=P(X\leqslant 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. It has a "bump" with negative derivative to the left of x < 0. It serves as the default activation for many transformer models such as BERT . [24]

SiLU

The SiLU (sigmoid linear unit) or swish function [25] is another smooth approximation which uses the sigmoid (logistic) function , first introduced in the 2016 GELU paper: [24]

It is cheaper to calculate than GELU. It also has a "bump".

Mish

The mish function (2019) can also be used as a smooth approximation of the rectifier. [25] It is defined as

where tanh ■ (x) {\displaystyle \tanh(x)} is the hyperbolic tangent , and softplus ■ (x) {\displaystyle \operatorname {softplus} (x)} is the softplus function.

Mish was obtained by experimenting with functions similar to Swish (SiLU, see above). It is non-monotonic (has a "bump") like Swish. The main new feature is that it exhibits a "self-regularizing" behavior attributed to a term in its first derivative. [25] [26]

Squareplus

Squareplus (2021) [27] is the function

where b ≥ 0 {\displaystyle b\geq 0} is a hyperparameter that determines the "size" of the curved region near x = 0 {\displaystyle x=0} . (For example, letting b = 0 {\displaystyle b=0} yields ReLU, and letting b = 4 {\displaystyle b=4} yields the metallic mean function.)

Squareplus shares many properties with softplus: It is monotonic , strictly positive , approaches 0 as x → – ∞ {\displaystyle x\to -\infty } , approaches the identity as x → + ∞ {\displaystyle x\to +\infty } , and is C ∞ {\displaystyle C^{\infty }} smooth . However, squareplus can be computed using only algebraic functions , making it well-suited for settings where computational resources or instruction sets are limited. Additionally, squareplus requires no special consideration to ensure numerical stability when x {\displaystyle x} is large.

DELU

ExtendeD Exponential Linear Unit (DELU, 2023) is an activation function which is smoother within the neighborhood of zero and sharper for bigger values, allowing better allocation of neurons in the learning process for higher performance. Thanks to its unique design, it has been shown that DELU may obtain higher classification accuracy than ReLU and ELU. [28]

In these formulas, a {\displaystyle a} , b {\displaystyle b} and x c {\displaystyle x_{c}} are hyperparameter values which could be set as default constraints a = 1 {\displaystyle a=1} , b = 2 {\displaystyle b=2} and x c = 1.25643 {\displaystyle x_{c}=1.25643} , as done in the original work.

See also

Softmax function

Sigmoid function

Tobit model

Layer (deep learning)

References

v

t

e

History timeline

timeline

Companies

Projects

Parameter Hyperparameter

Hyperparameter

Loss functions

Regression Bias–variance tradeoff Double descent Overfitting

Bias–variance tradeoff

Double descent

Overfitting

Clustering

Gradient descent SGD Quasi-Newton method Conjugate gradient method

SGD

Quasi-Newton method

Conjugate gradient method

Backpropagation

Attention

Convolution

Normalization Batchnorm

Batchnorm

Activation Softmax Sigmoid Rectifier

Softmax

Sigmoid

Rectifier

Gating

Weight initialization

Regularization
Datasets Augmentation
Augmentation
Prompt engineering
Reinforcement learning Q-learning SARSA Imitation Policy gradient
Q-learning
SARSA
Imitation
Policy gradient
Diffusion
Latent diffusion model
Autoregression
Adversary
RAG
Uncanny valley
RLHF
Self-supervised learning
Reflection
Recursive self-improvement
Hallucination
Word embedding
Vibe coding
Machine learning In-context learning
In-context learning
Artificial neural network Deep learning
Deep learning
Language model Large language model NMT
Large language model
NMT
Reasoning language model
Model Context Protocol
Intelligent agent
Artificial human companion
Humanity's Last Exam
Artificial general intelligence (AGI)
AlexNet
WaveNet
Human image synthesis

HWR

OCR

Computer vision

Speech synthesis 15.ai ElevenLabs

15.ai

ElevenLabs

Speech recognition Whisper

Whisper

Facial recognition

AlphaFold

Text-to-image models Aurora DALL-E Firefly Flux Ideogram Imagen Midjourney Recraft Stable Diffusion

Aurora

DALL-E

Firefly

Flux

Ideogram

Imagen

Midjourney

Recraft

Stable Diffusion

Text-to-video models Dream Machine Runway Gen Hailuo AI Kling Sora Veo

Dream Machine

Runway Gen

Hailuo AI

Kling

Sora

Veo

Music generation Riffusion Suno AI Udio

Riffusion

Suno AI

Udio

Word2vec

Seq2seq

GloVe

BERT

T5

Llama

Chinchilla AI

PaLM

GPT 1 2 3 J ChatGPT 4 4o o1 o3 4.5 4.1 o4-mini 5

1

2

3

J

ChatGPT

4

4o

o1

o3

4.5

4.1

o4-mini

5

Claude

Gemini Gemini (language model) Gemma

Gemini (language model)

Gemma

Grok

LaMDA

BLOOM

DBRX

Project Debater

IBM Watson

IBM Watsonx

Granite

PanGu-Σ

DeepSeek

Qwen

AlphaGo

AlphaZero

OpenAI Five

Self-driving car

MuZero

Action selection AutoGPT

AutoGPT

Robot control
Alan Turing
Warren Sturgis McCulloch
Walter Pitts
John von Neumann
Claude Shannon
Shun'ichi Amari
Kunihiko Fukushima
Takeo Kanade
Marvin Minsky
John McCarthy
Nathaniel Rochester
Allen Newell
Cliff Shaw
Herbert A. Simon
Oliver Selfridge
Frank Rosenblatt
Bernard Widrow
Joseph Weizenbaum
Seymour Papert
Seppo Linnainmaa
Paul Werbos
Geoffrey Hinton
John Hopfield
Jürgen Schmidhuber
Yann LeCun
Yoshua Bengio
Lotfi A. Zadeh
Stephen Grossberg
Alex Graves
James Goodnight
Andrew Ng
Fei-Fei Li
Alex Krizhevsky
Ilya Sutskever
Oriol Vinyals
Quoc V. Le
Ian Goodfellow

Demis Hassabis
David Silver
Andrej Karpathy
Ashish Vaswani
Noam Shazeer
Aidan Gomez
John Schulman
Mustafa Suleyman
Jan Leike
Daniel Kokotajlo
François Chollet
Neural Turing machine
Differentiable neural computer
Transformer Vision transformer (ViT)
Vision transformer (ViT)
Recurrent neural network (RNN)
Long short-term memory (LSTM)
Gated recurrent unit (GRU)
Echo state network
Multilayer perceptron (MLP)
Convolutional neural network (CNN)
Residual neural network (RNN)
Highway network
Mamba
Autoencoder
Variational autoencoder (VAE)
Generative adversarial network (GAN)
Graph neural network (GNN)
Category