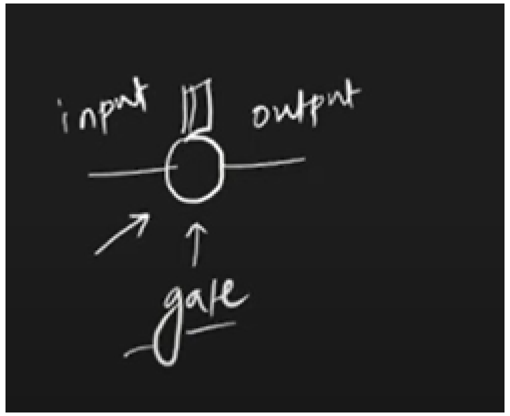
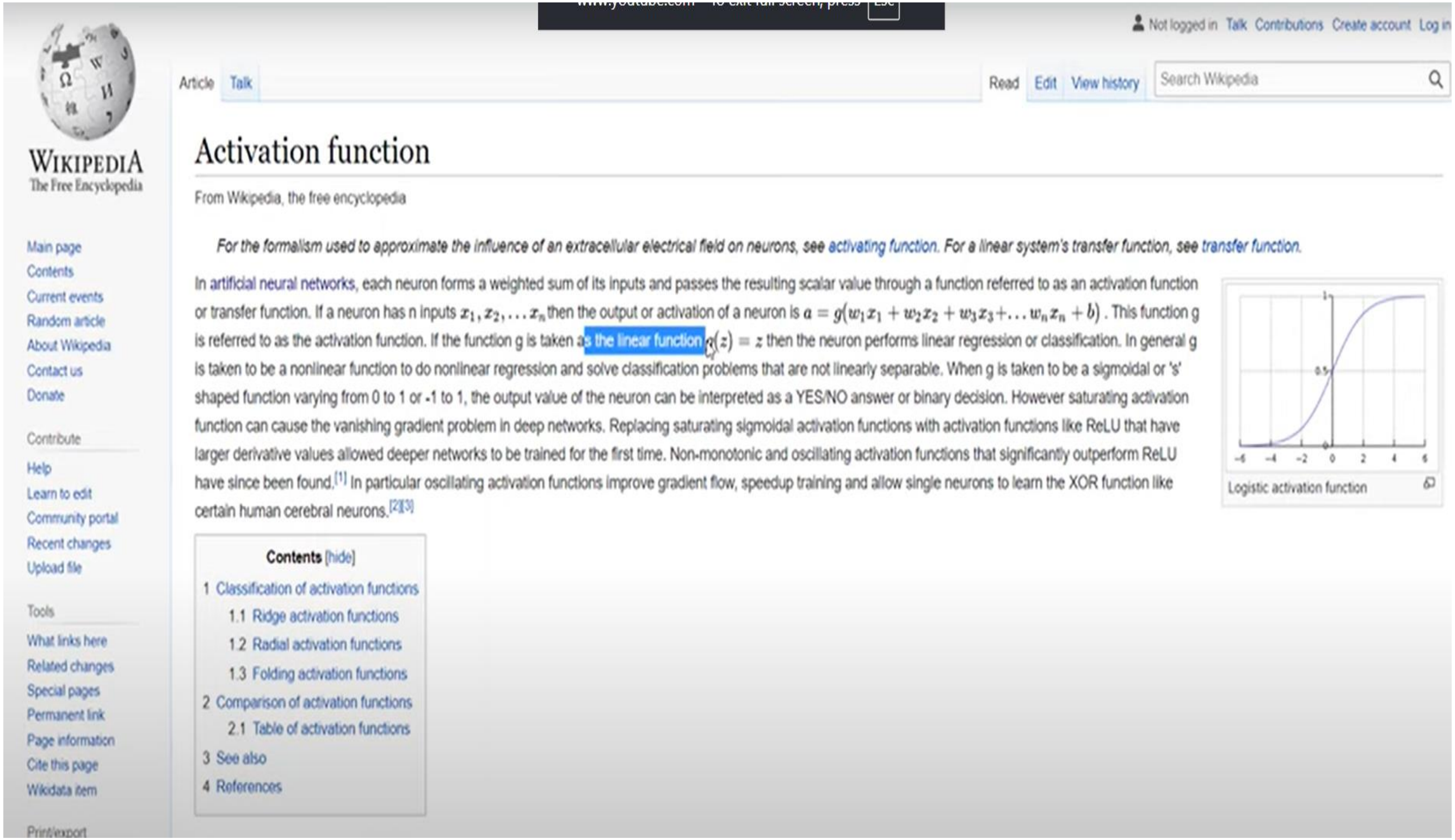
Activation Function

- In artificial neural networks, each neuron forms a weighted sum of its inputs and passes the resulting scalar value through a function referred to as an activation function or transfer function. If a neuron has n inputs then the output or activation of a neuron is

$$a = g(w_1x_1 + w_2x_2 + w_3x_3 + \ldots w_nx_n + b)$$

- This function g is referred to as the activation function."



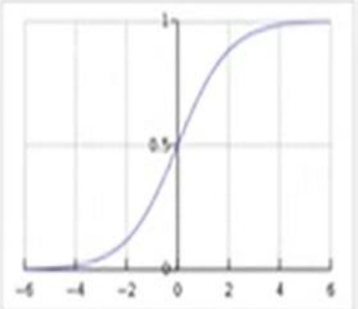


Activation function

From Wikipedia, the free encyclopedia

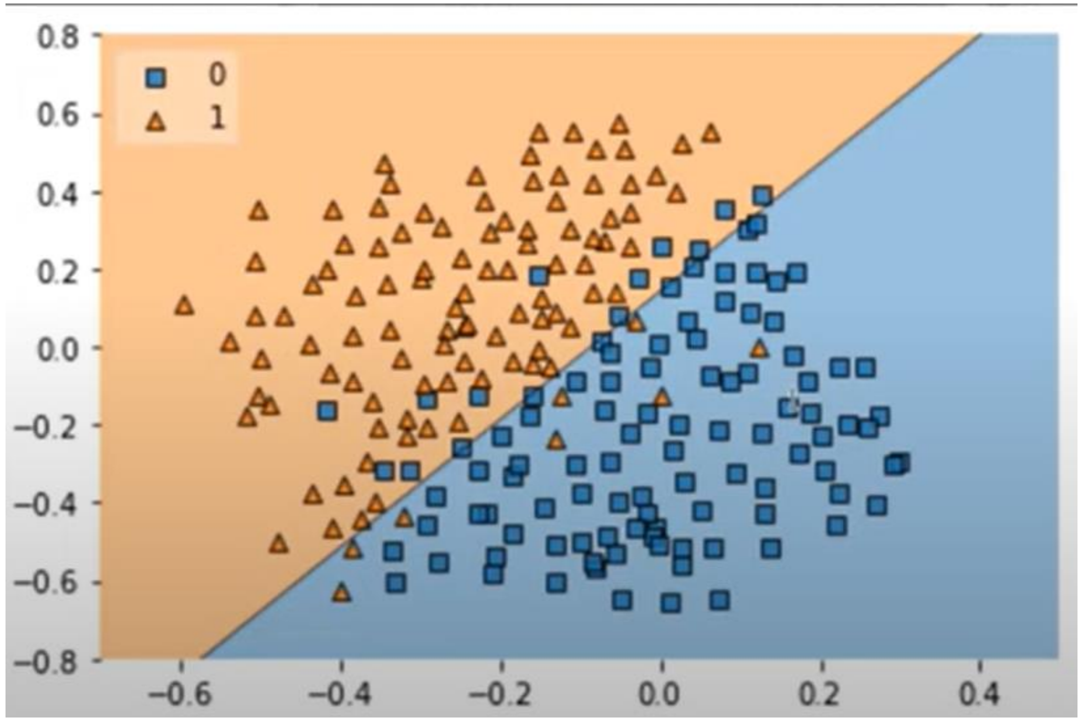
For the formalism used to approximate the influence of an extracellular electrical field on neurons, see activating function. For a linear system's transfer function, see transfer function.

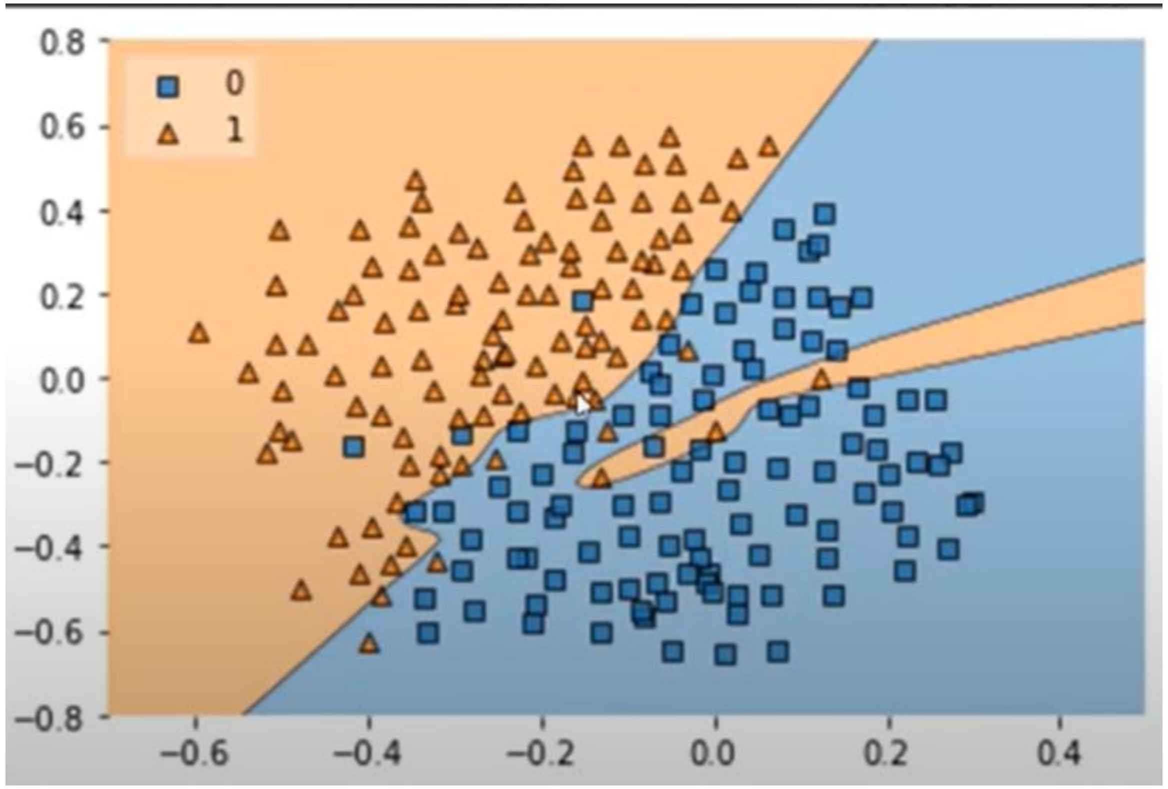
In artificial neural networks, each neuron forms a weighted sum of its inputs and passes the resulting scalar value through a function referred to as an activation function or transfer function. If a neuron has n inputs $x_1, x_2, \ldots x_n$ then the output or activation of a neuron is $a = g(w_1x_1 + w_2x_2 + w_3x_3 + \ldots w_nx_n + b)$. This function g is referred to as the activation function. If the function g is taken as the linear function $g(z) = z$ then the neuron performs linear regression or classification. In general g is taken to be a nonlinear function to do nonlinear regression and solve classification problems that are not linearly separable. When g is taken to be a sigmoidal or 's' shaped function varying from 0 to 1 or -1 to 1, the output value of the neuron can be interpreted as a YES/NO answer or binary decision. However saturating activation function can cause the vanishing gradient problem in deep networks. Replacing saturating sigmoidal activation functions with activation functions like ReLU that have larger derivative values allowed deeper networks to be trained for the first time. Non-monotonic and oscillating activation functions that significantly outperform ReLU have since been found.[1] In particular oscillating activation functions improve gradient flow, speedup training and allow single neurons to learn the XOR function like certain human cerebral neurons.[2][3]



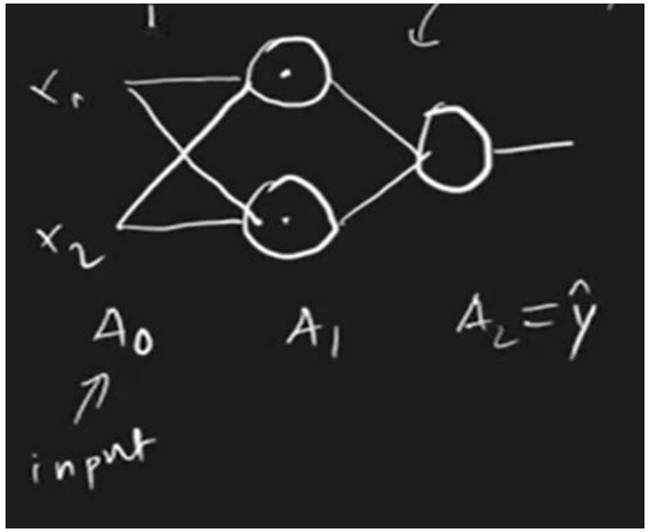
Logistic activation function

Contents [hide]


0
1
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4


0
1
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4

Why activation function needed?



Ideal activation function

- Non linear

Ideal activation function

- Differentiable

Ideal activation function

- Computationaly inexpensive

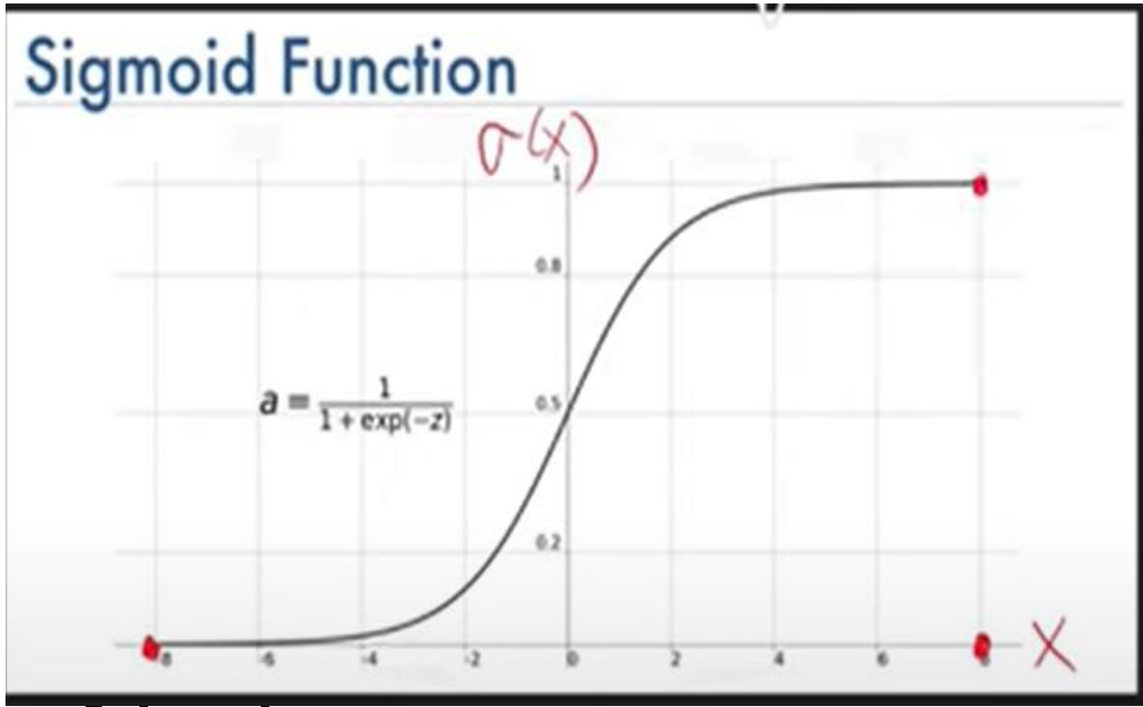
Ideal activation function

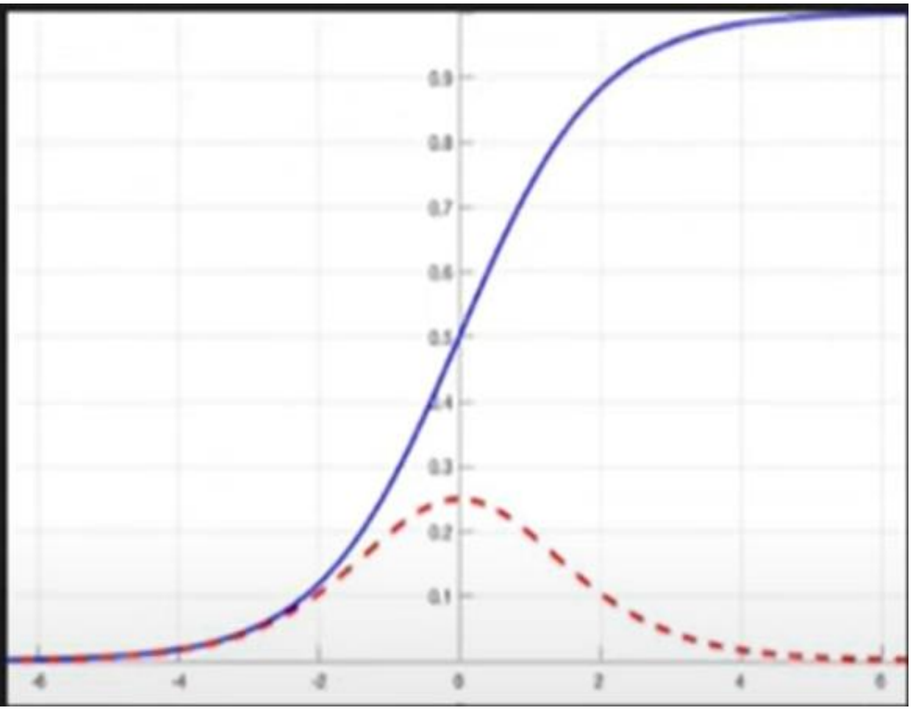
- Zero-centerd (Normalized)

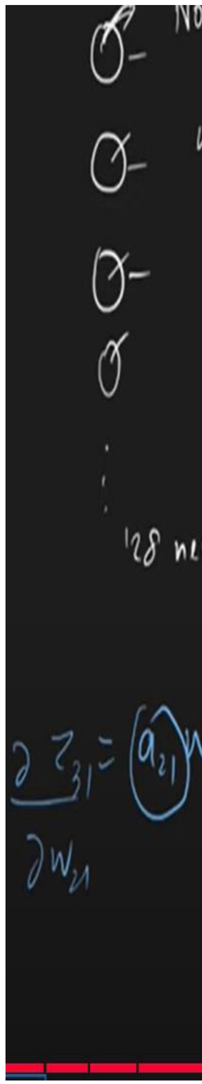
Ideal activation function

- None Saturating
  - Vanishing gardient

Sigmoid Activation Function

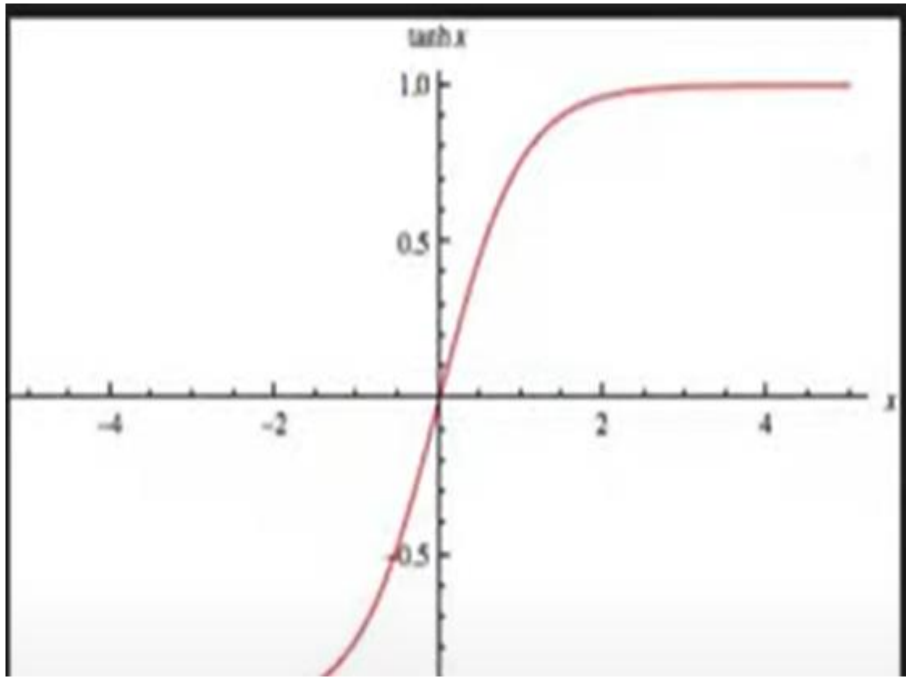


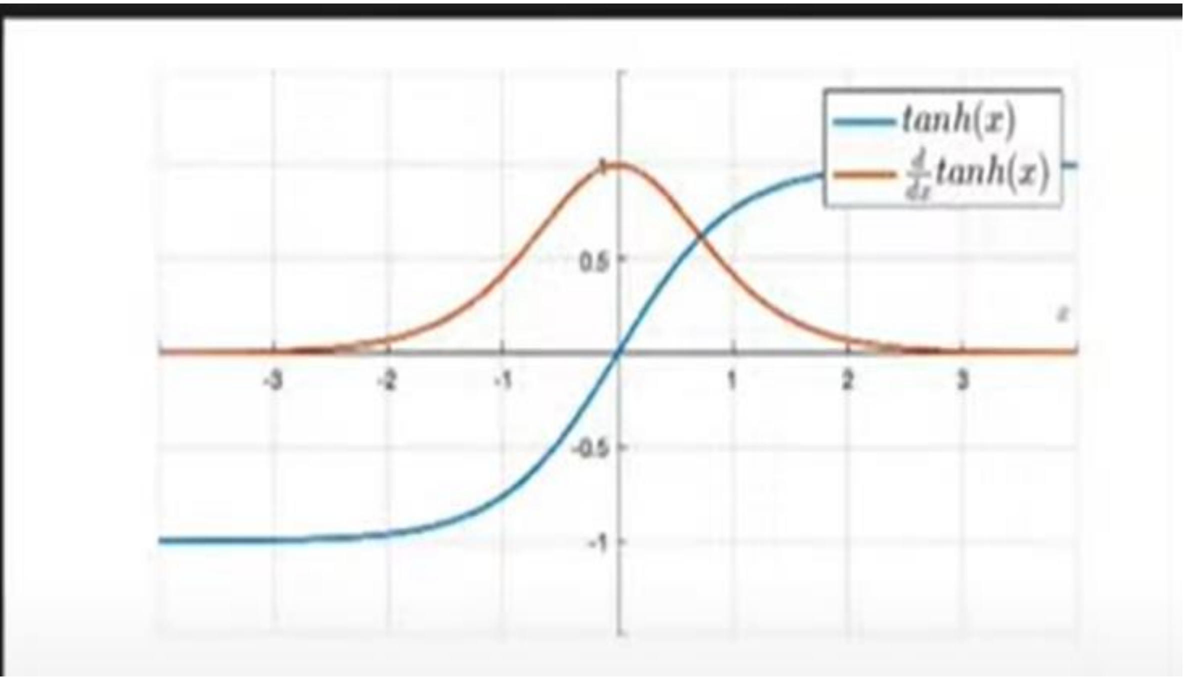




- **Advantage**
- [0,1], non linear function , differentiable
- **Disadvantage**
- Saturating function, Not used in hidden layers, used in output layer
- None zero centerd→ slow convergence
- Computationaly expensive

Tanh Activation Function



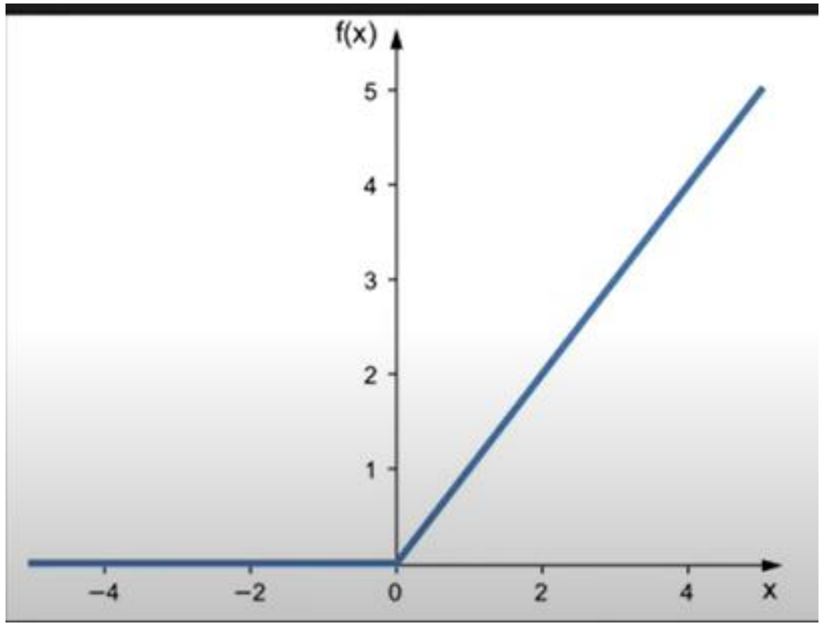


$$f(x) = \frac{(e^{x} - e^{-x})}{e^{x} + e^{-x}}$$

$$f'(x) = (1 - \tanh^{2}(x))$$

- **Advantages:**
  - Non Linear
  - Differentiable
  - Zreo-centered
- **Disadvantages:**
  - Saturating function $\rightarrow$ vanishing Gradient problem
  - Computationally Expensive$\rightarrow$ slow

Relu Activation Function



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

$\underline{f(x) = x}$

- Advantage:
  - Non Linear
  - Non saturated in +ve region
  - Computationally Inexpensive
  - Convergence → faster

Disadvantabe

Not completely differentiable

X<0 →0

X>0 → 1

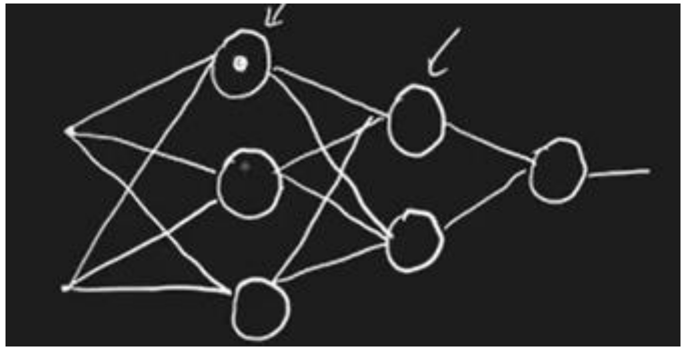
Not zero centered → batch normalization

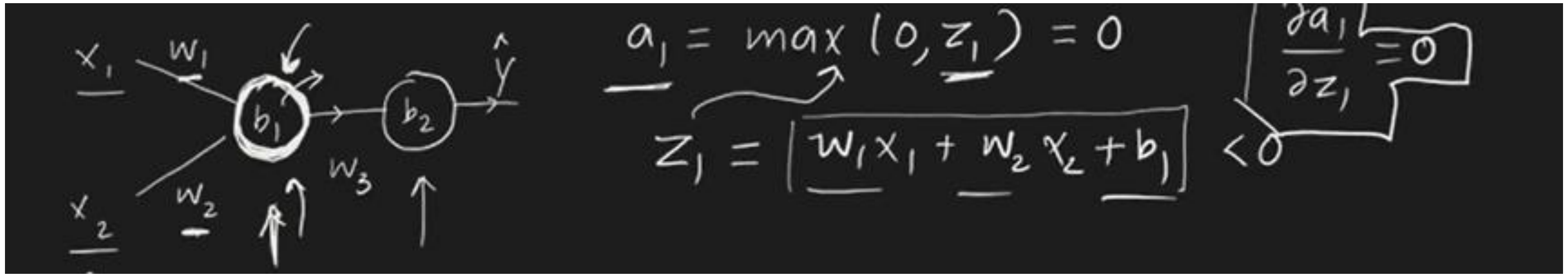
Dying relu problem

Best choice for hidden Layers

Dying Relu Problem

- Relu → vanishing Gradient problem→ ouput Of some neuron become zero, whatever the Input is → Dead Neuron

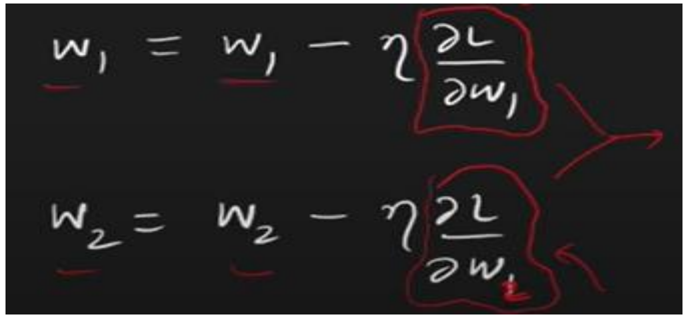




$$a_1 = \max(0, z_1) = 0$$

$$z_1 = \boxed{w_1 x_1 + w_2 x_2 + b_1} < 0$$

$$\boxed{\frac{\partial a_1}{\partial z_1} = 0}$$



$$w_1 = w_1 - \eta \boxed{\frac{\partial L}{\partial w_1}}$$

$$w_2 = w_2 - \eta \boxed{\frac{\partial L}{\partial w_2}}$$

$$\begin{tikzcd}
 & & w_3 & & & w_1 \\
L \arrow[r] & \hat{y} \arrow[ur] \arrow[r] \arrow[dr] & b_2 & & & \\
 & & a_1 \arrow[r] & z_1 \arrow[uurr] \arrow[rr] \arrow[drr] \arrow[ddl] \arrow[dd] & & w_2 \\
 & & & & & x_1 \\
 & & b_1 & x_2 & &
\end{tikzcd}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_1}{\partial w_1}$$

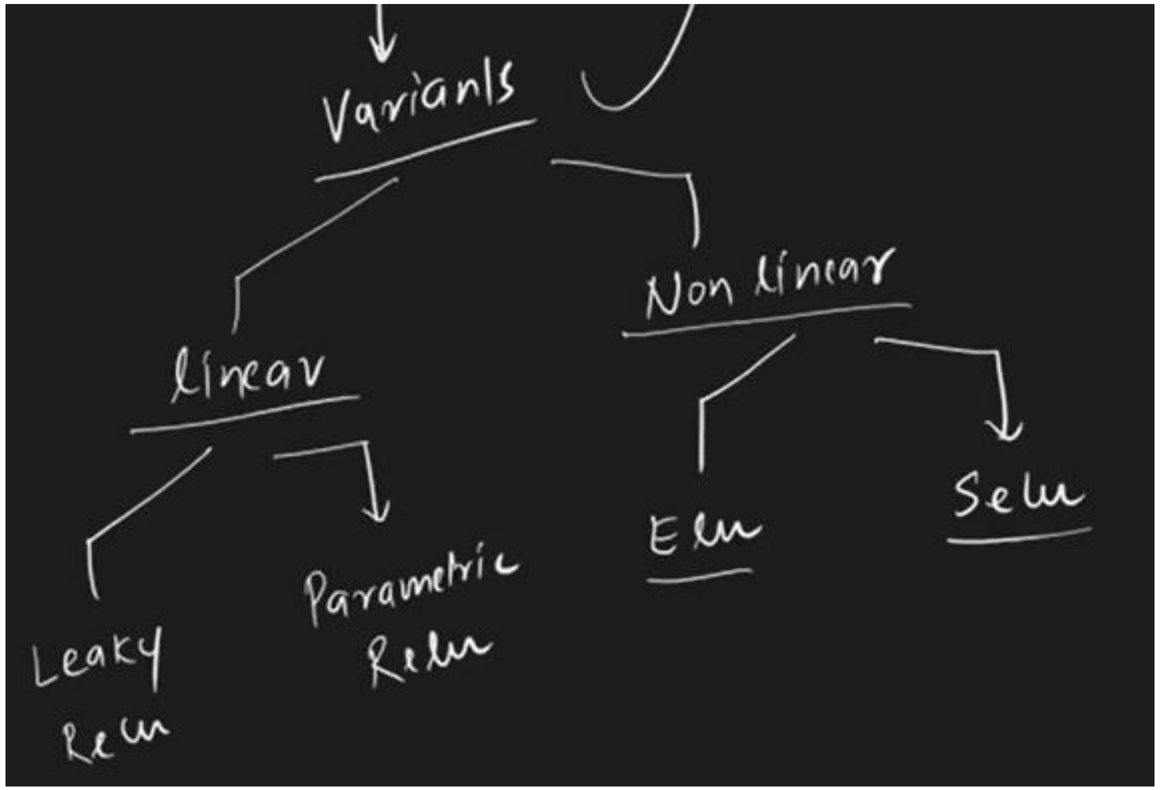
$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_1}{\partial w_2}$$

$$z_1 = w_1 x_1 + \overset{\downarrow}{w_2} x_2 + b_1 < 0$$

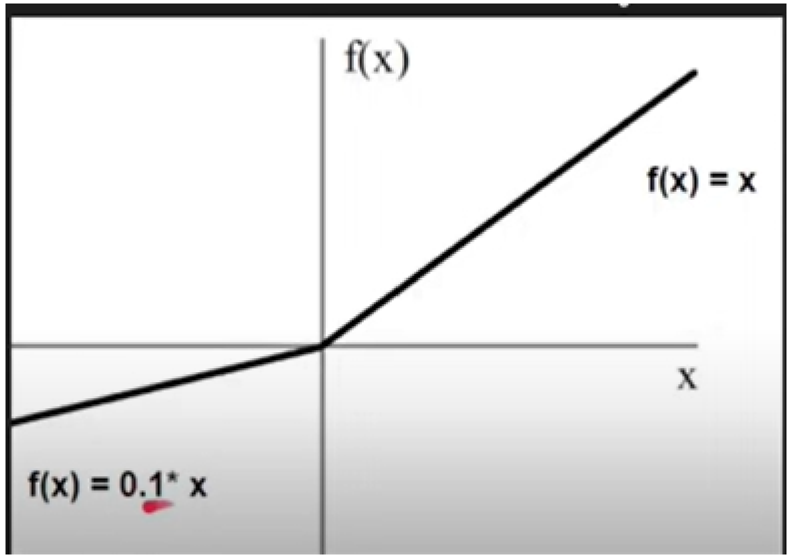
- Reason of become negative
  - Learning Rate become very high
  - High negative bias

Solutions:

- Set Low learning rate
- Set +ve value of bias$\rightarrow$ 0.01
- Don’t use Relu -$\rightarrow$ use its variants
-


Varianls
linear
Non linear
Leaky Relu
Parametric Relu
Elu
Selu

Leaky Relu



$$f(z) = \max(0.01z, z)$$

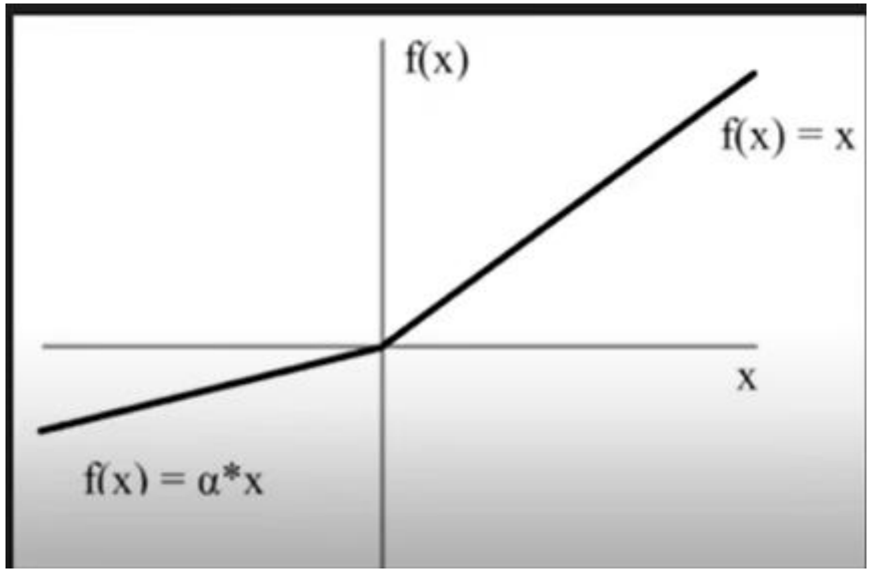
$$f'(z) \quad \begin{cases} z \geq 0 \rightarrow \textcircled{1} \\ z < 0 \rightarrow \boxed{0.01} \end{cases}$$

Non saturated

Easily computed

- No dying
- Close to zero

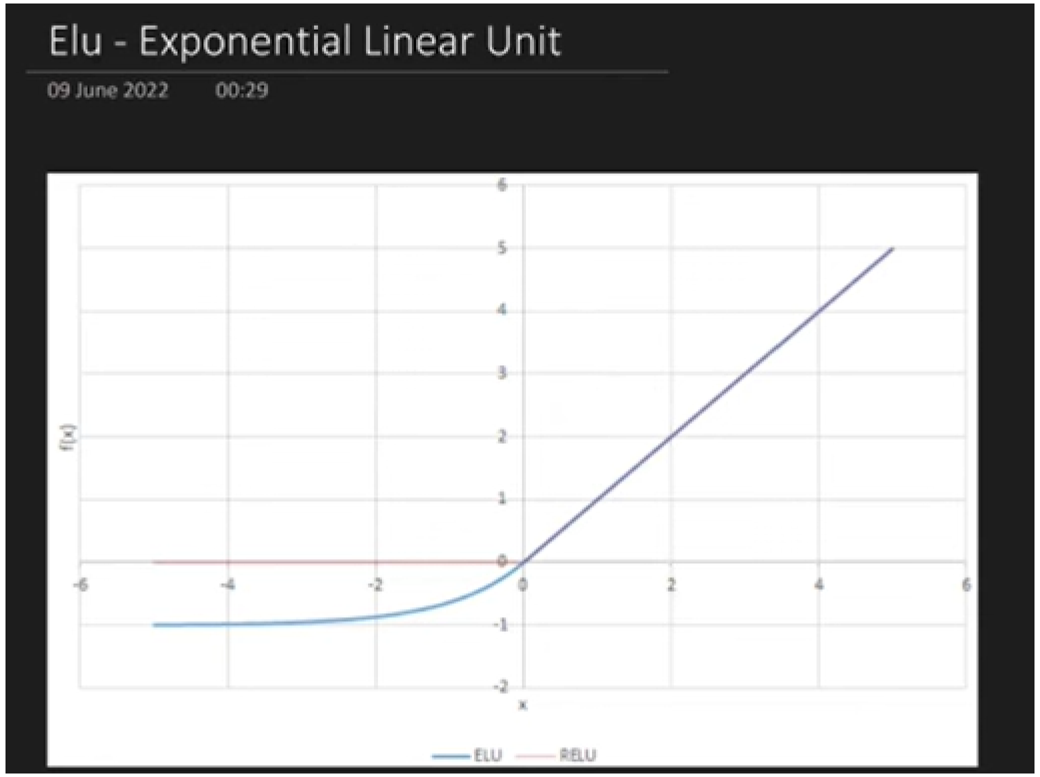
Parmetric Relu



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

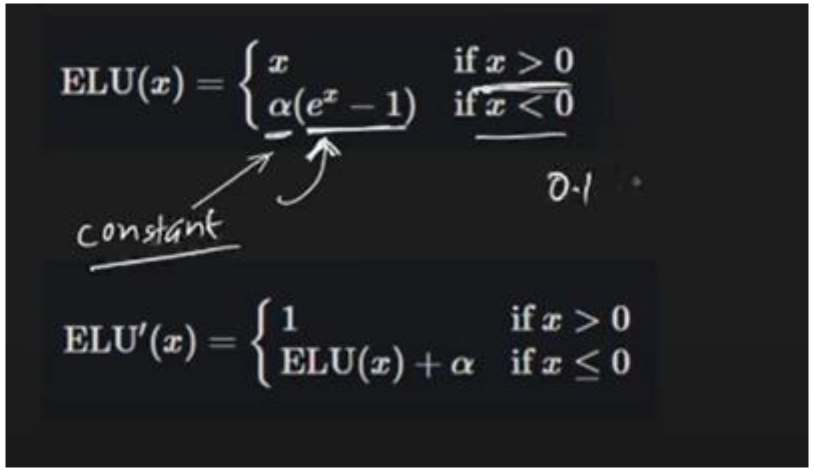
$\nearrow$ $0.01x$

a is traninable parameter



Elu - Exponential Linear Unit

09 June 2022 00:29



$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \underline{\alpha}\underline{(e^x - 1)} & \underline{\text{if } x < 0} \end{cases}$$

constant

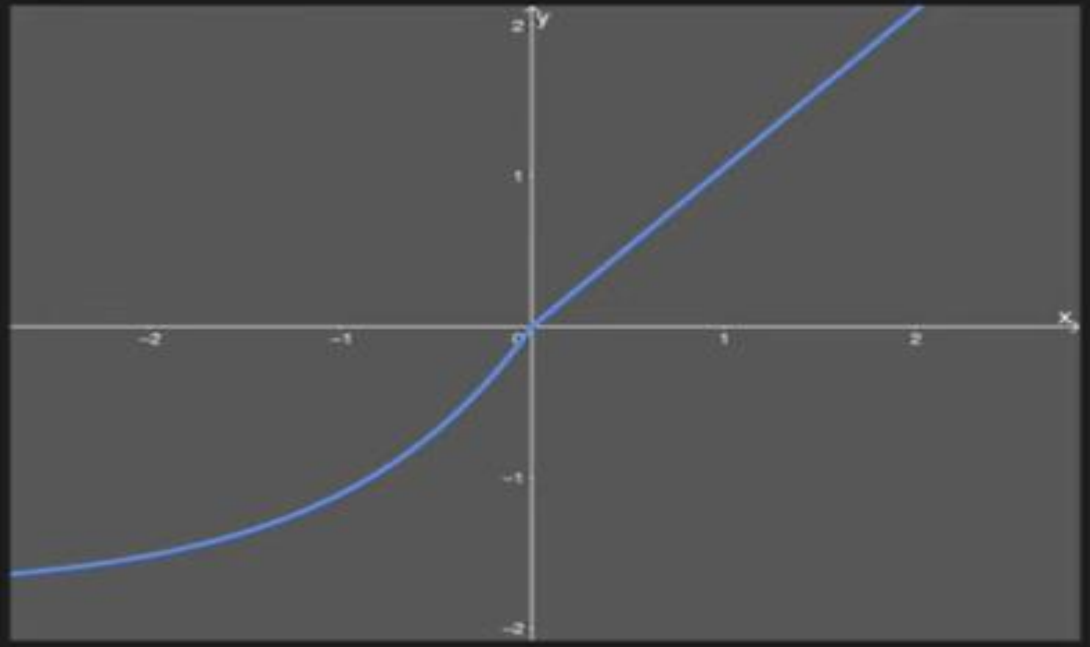
0.1

$$\mathrm{ELU}'(x) = \begin{cases} 1 & \text{if } x > 0 \\ \mathrm{ELU}(x) + \alpha & \text{if } x \leq 0 \end{cases}$$

- Close to zero center – faster
- Always continues and differentiable
- Computationally inexpensive

Selu - Scaled Exponential Linear Unit

09 June 2022 00:29



$$\mathrm{SELU}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^{x} - \alpha & \text{if } x \leq 0 \end{cases}$$

$$\alpha \approx 1.6732632423543772848170429916717$$

$$\lambda \approx 1.0507009873554804934193349852946$$

$$\mathrm{SELU}'(x) = \lambda \begin{cases} 1 & \text{if } x > 0 \\ \alpha e^{x} & \text{if } x \leq 0 \end{cases}$$

- Self normalize
- Mean → zero standard deviation =1