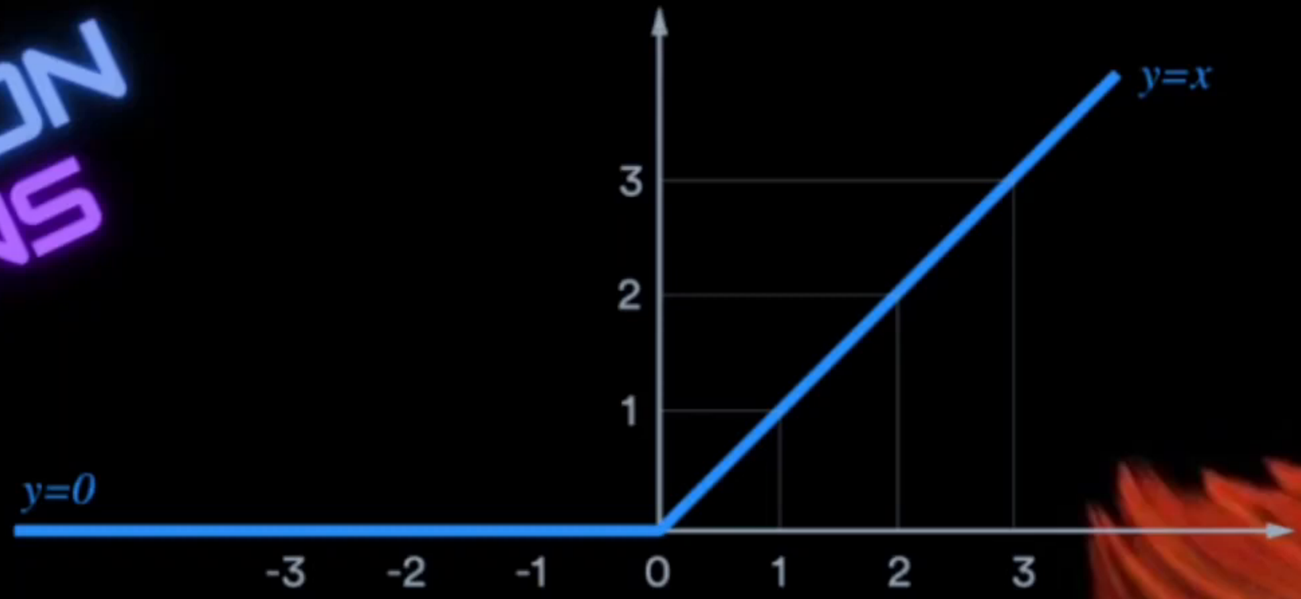
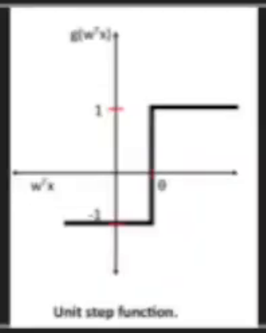
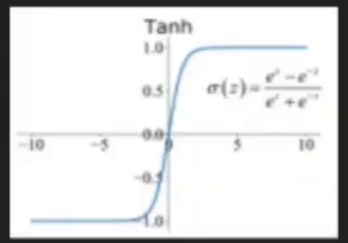
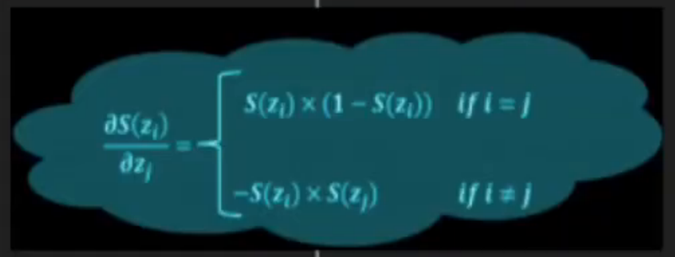
ACTIVATION FUNCTIONS



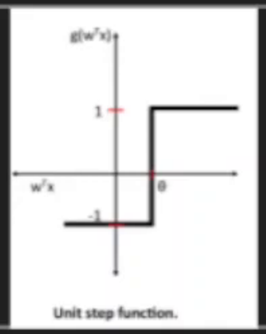
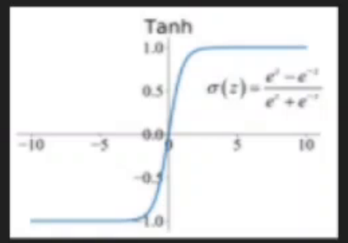
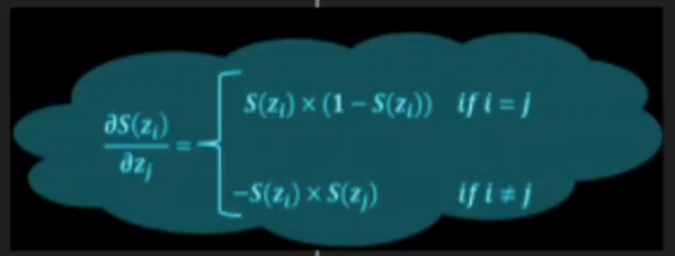

# RELU ACTIVATION


Programming

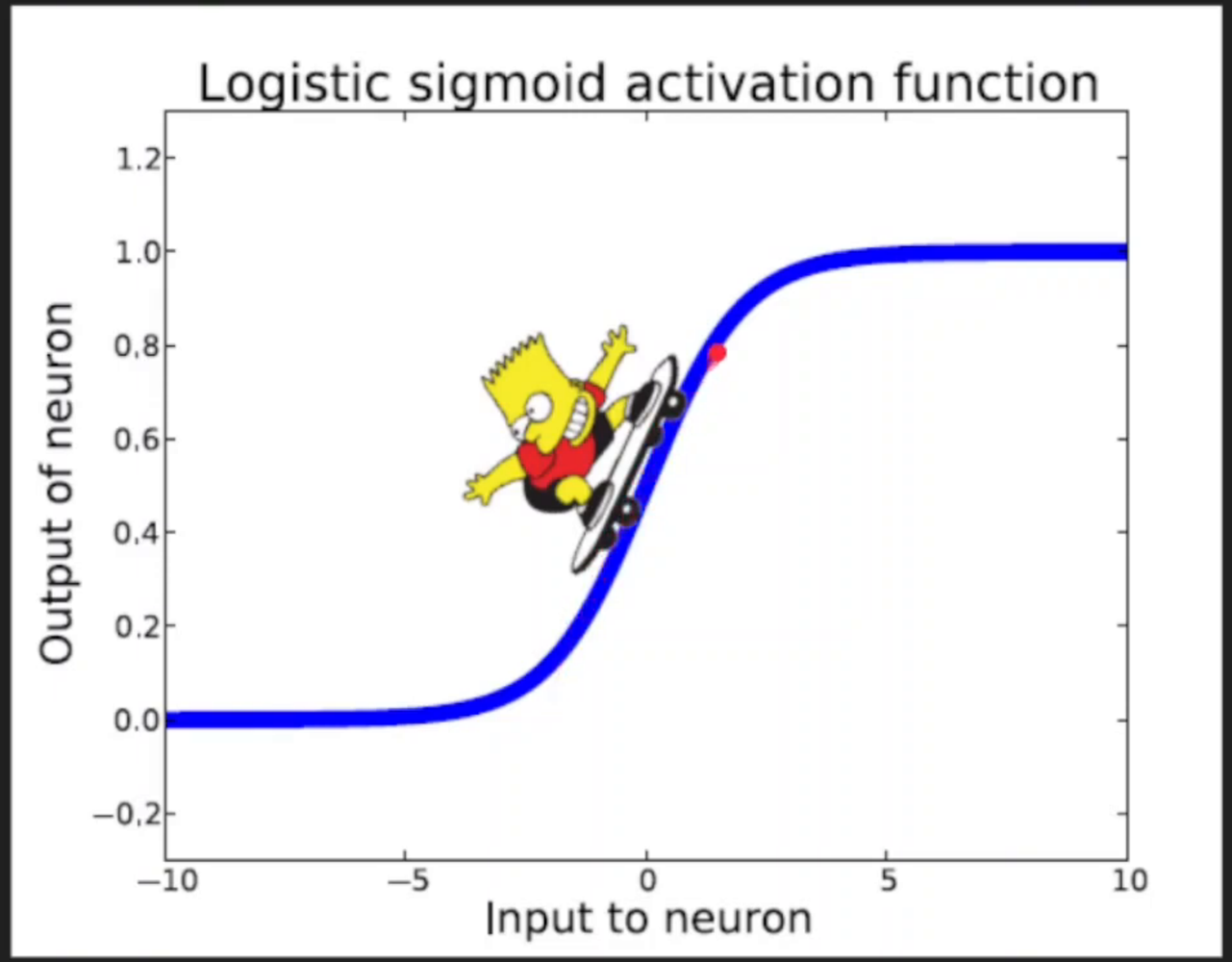
# Recap

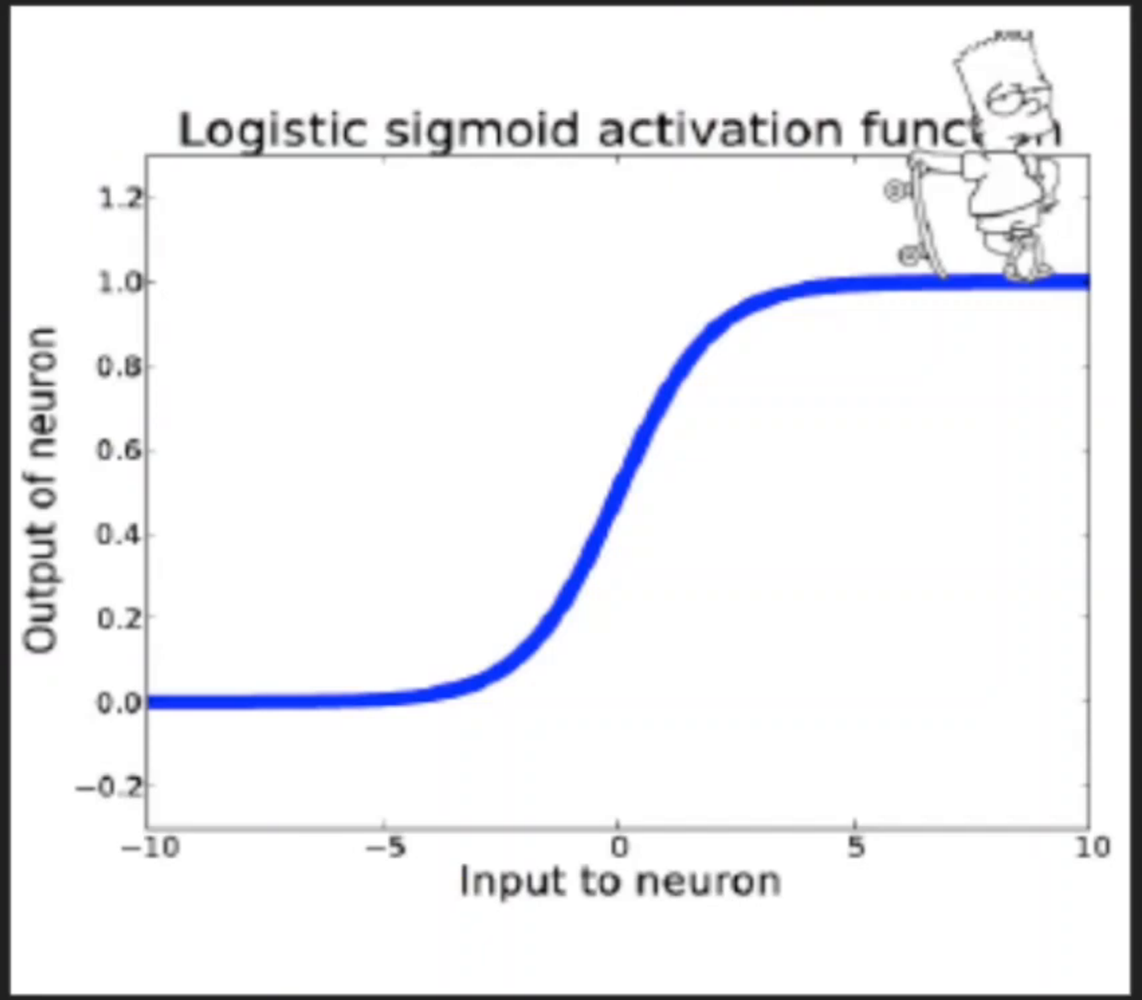
| | | | | |
|---|---|---|---|---|
| **Step** | Binary step<br>$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$ | **f'(x)=0** |  | ***Not used much**<br>***Binary** |
| **Sigmoid** | $S(x) = \frac{1}{1+e^{-x}}$ | **f'(x) = f(x)* (1-f(x))** |  | *** Binary**<br>*** Preferred** |
| **Tanh** | Tanh Function<br>$a = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$ | **f'(x) = 1- f(x)^2** |  | *** Binary**<br>*** Better than Sigmoid**<br>*** Hidden layers** |
| **Softmax** | $\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ |  | *** Multi-class**<br>*** Output layers** |

# Recap

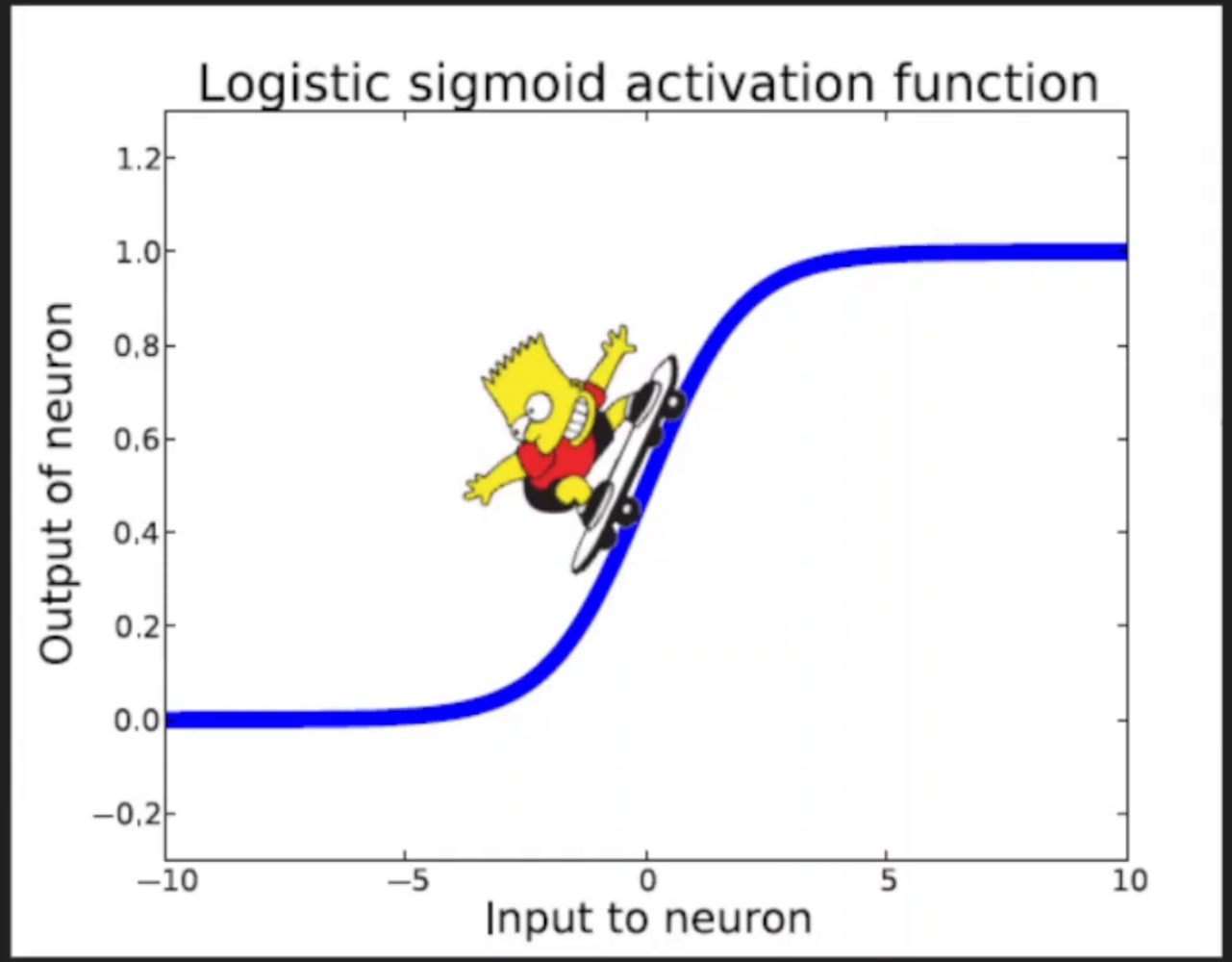
| | | | | |
|---|---|---|---|---|
| **Step** | Binary step<br>$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$ | **f'(x)=0** |  | ***Not used much**<br>***Binary** |
| **Sigmoid** | $S(x) = \frac{1}{1+e^{-x}}$ | **f'(x) = f(x)* (1-f(x))** |  | *** Binary**<br>*** Preferred** |
| **Tanh** | Tanh Function<br>$a = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$ | **f'(x) = 1-f(x)^2** |  | *** Binary**<br>*** Better than Sigmoid**<br>*** Hidden layers** |
| **Softmax** | $\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ |  | *** Multi-class**<br>*** Output layers** |

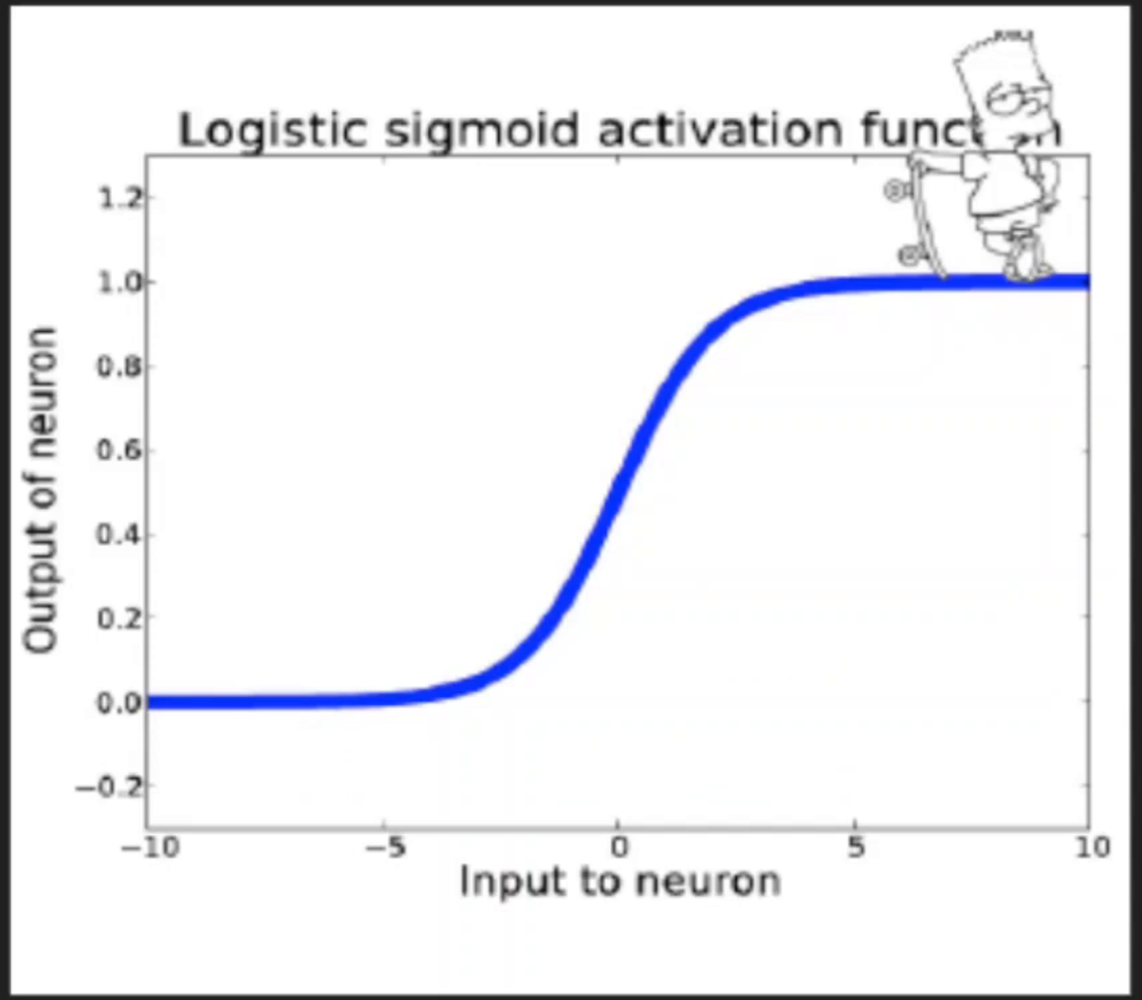
# Vanishing Gradient



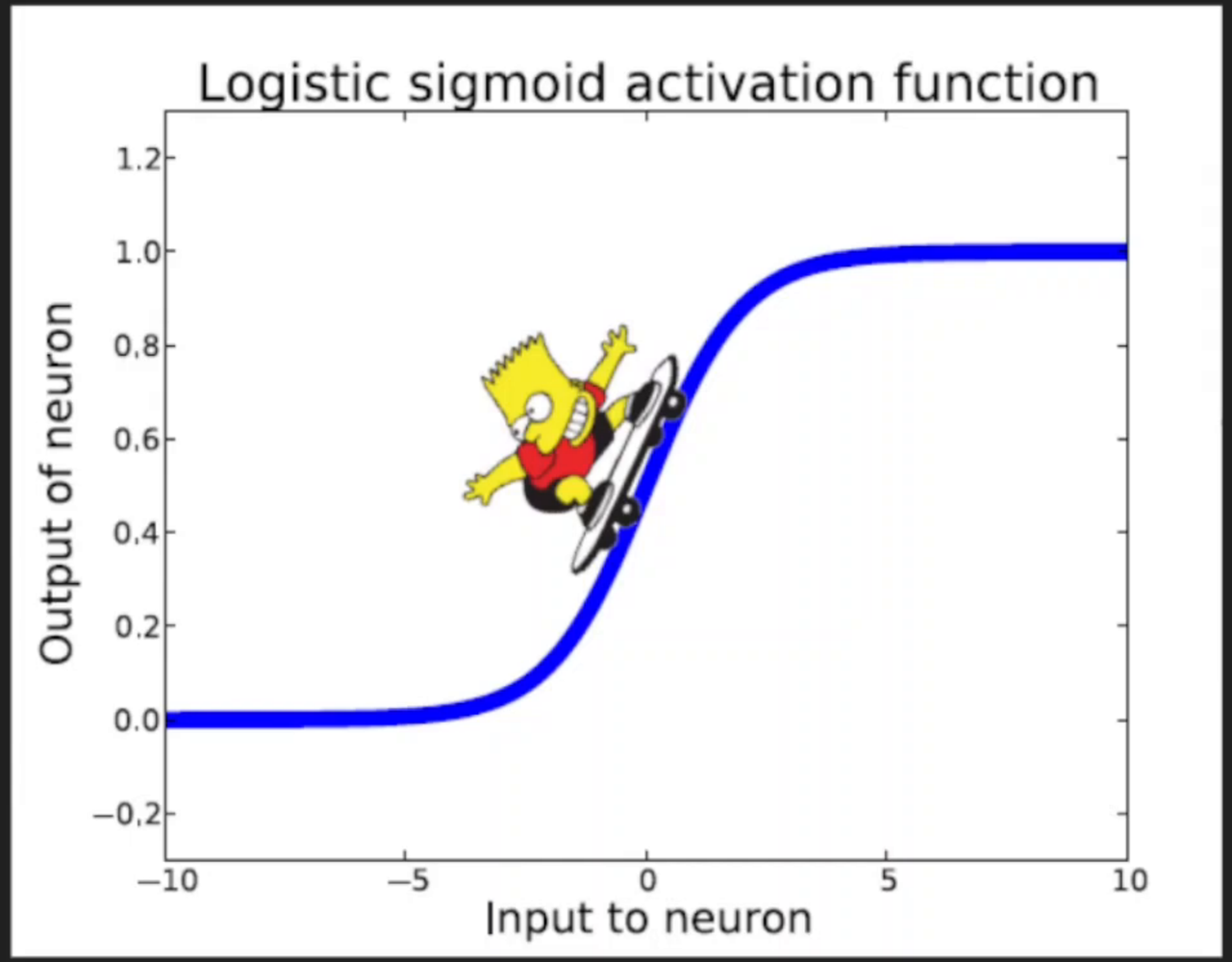



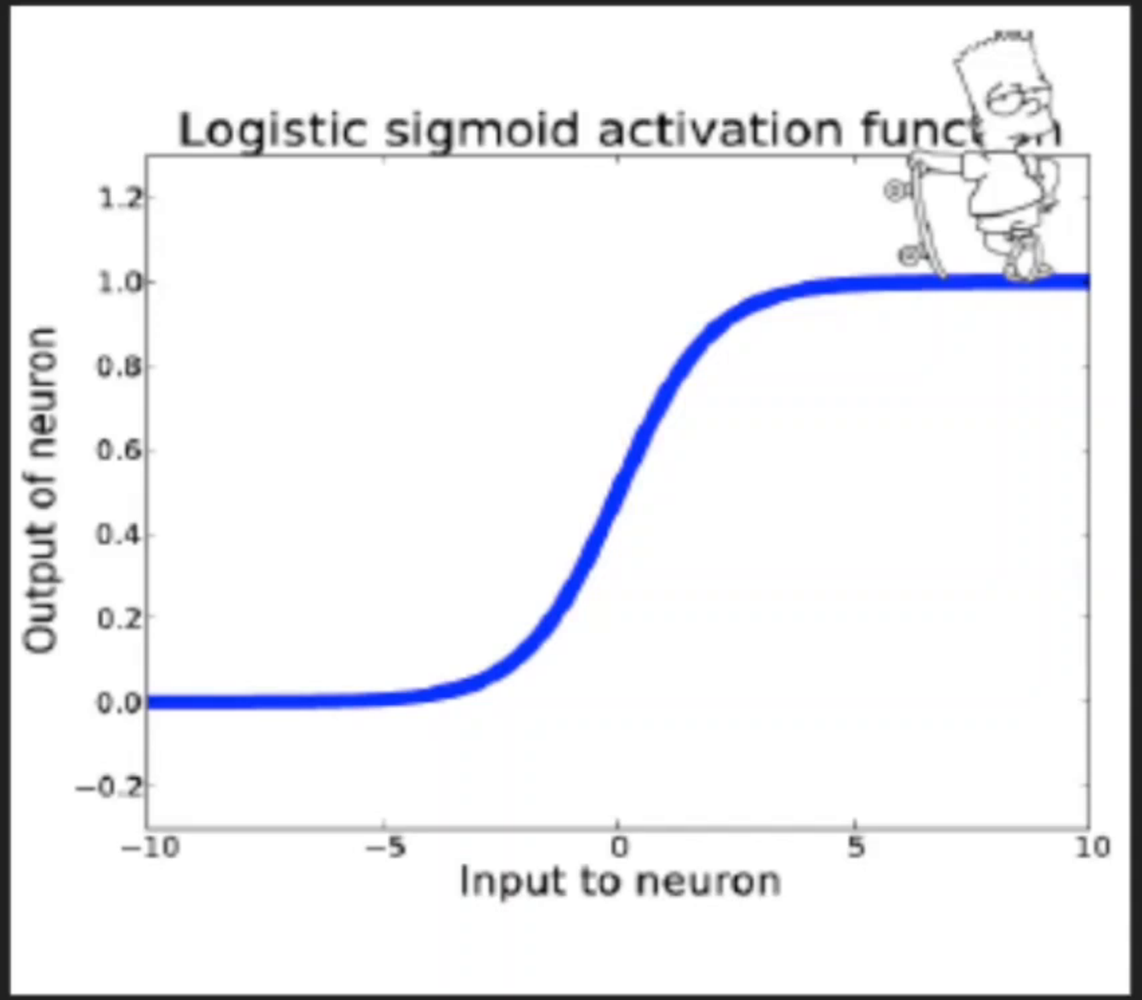
# Vanishing Gradient



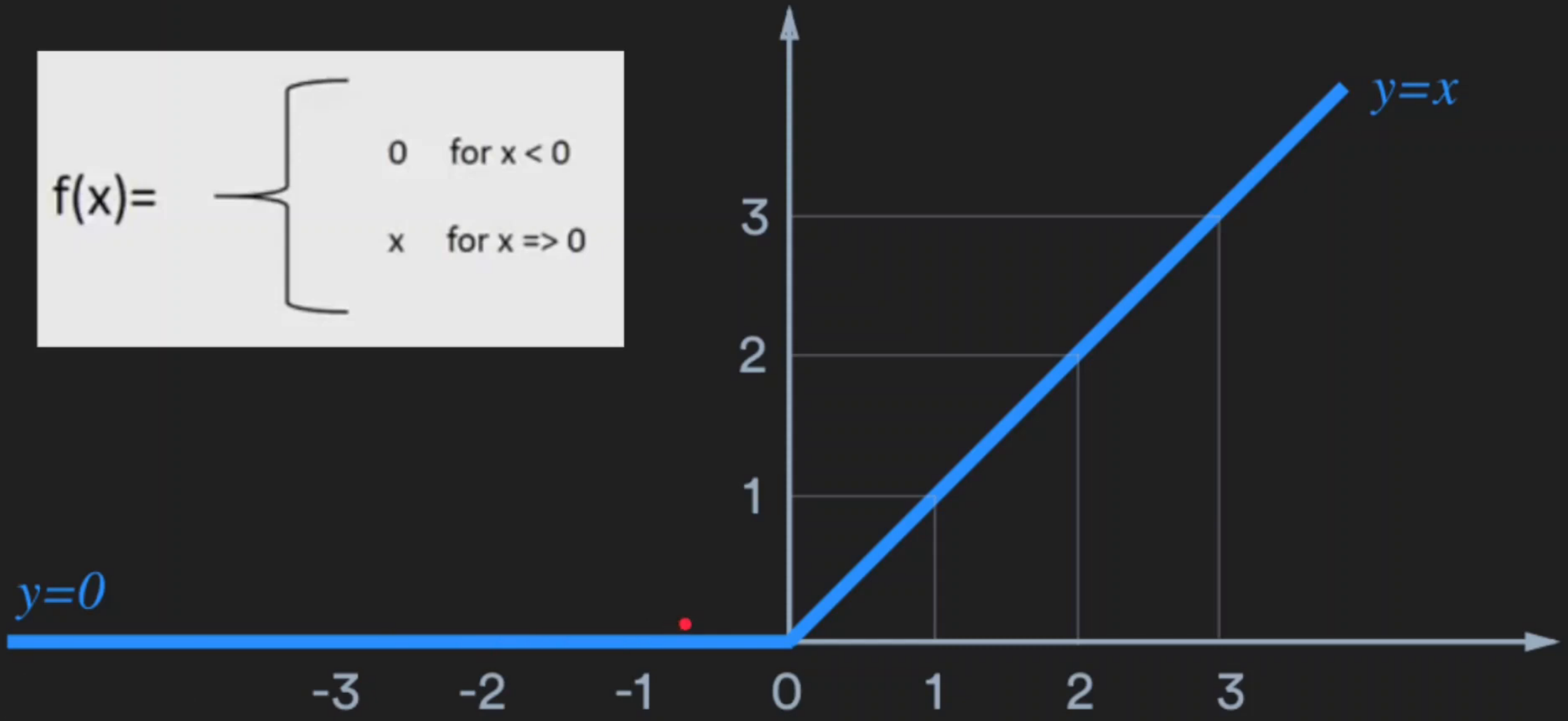
# Vanishing Gradient



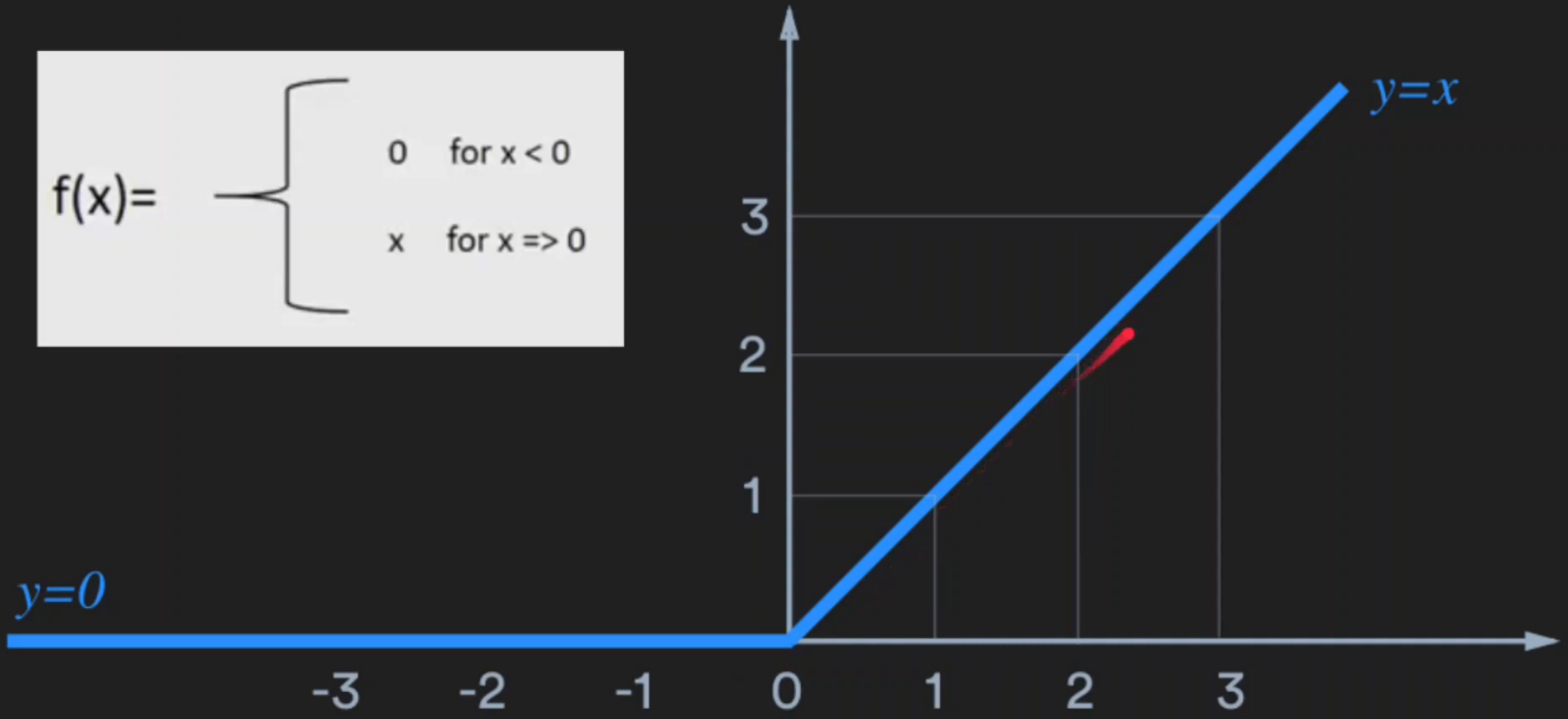
# ReLU

$$f(x)=\begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \Rightarrow 0 \end{cases}$$



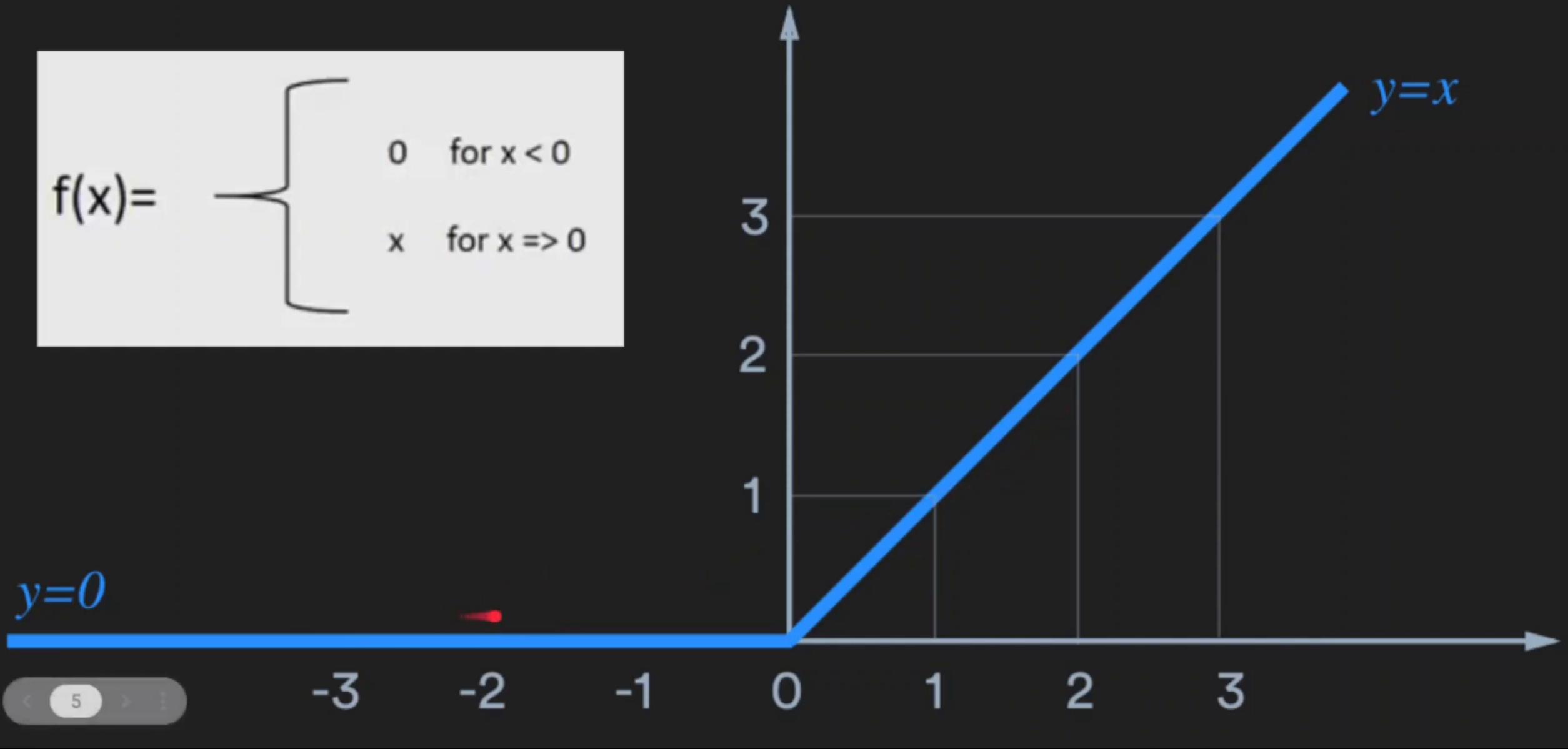
# ReLU

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \Rightarrow 0 \end{cases}$$



# ReLU

$$f(x)=\begin{cases}0 & \text{for } x < 0\\ x & \text{for } x \Rightarrow 0\end{cases}$$



# ReLU Layer

**Filter 1 Feature Map**

| | | | |
|---|---|---|---|
| 9 | 3 | 5 | -8 |
| -6 | 2 | -3 | 1 |
| 1 | 3 | 4 | 1 |
| 3 | -4 | 5 | 1 |

| | | | |
|---|---|---|---|
| 9 | 3 | 5 | 0 |
| 0 | 2 | 0 | 1 |
| 1 | 3 | 4 | 1 |
| 3 | 0 | 5 | 1 |

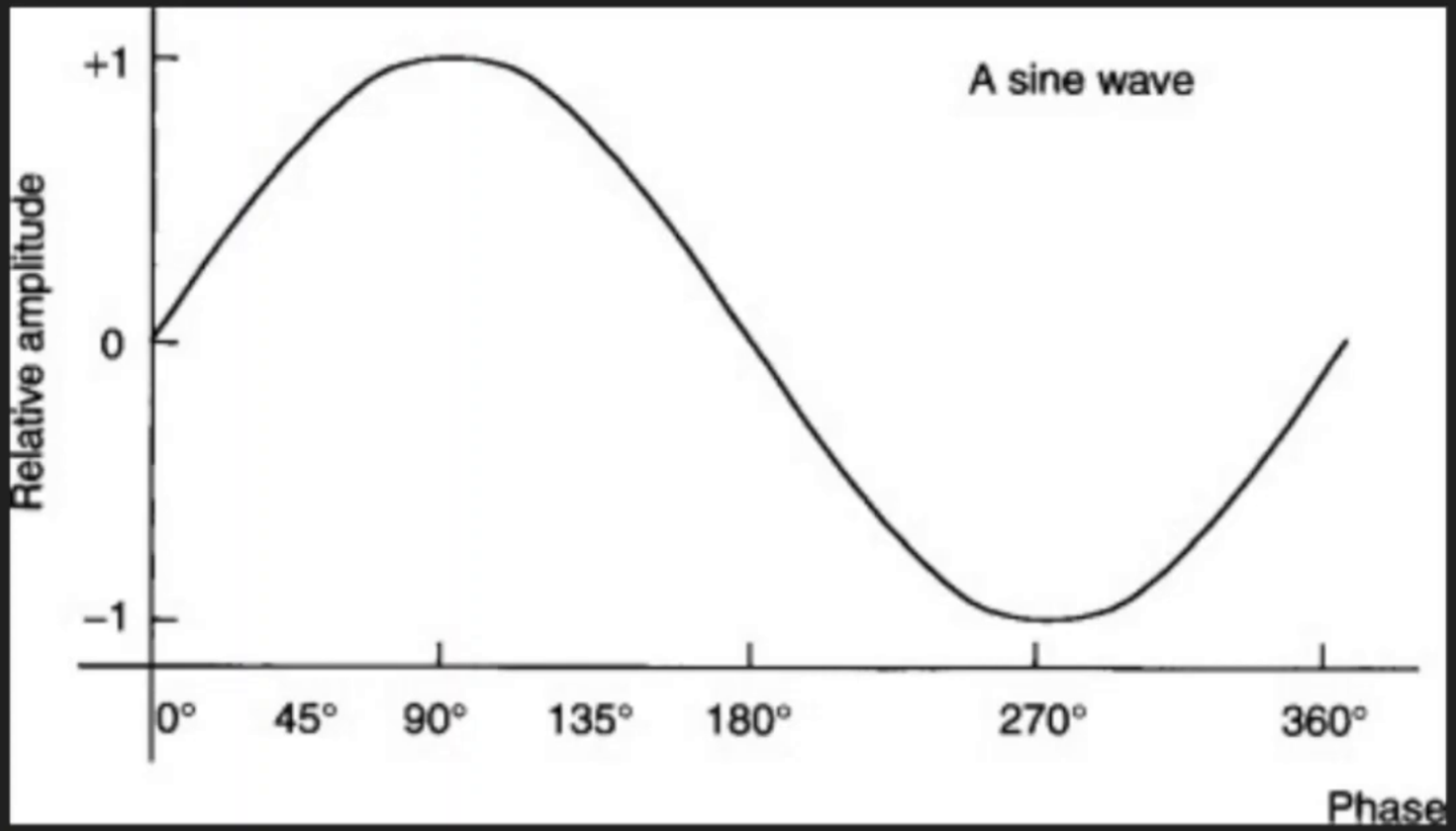
# ReLU Layer

**Filter 1 Feature Map**

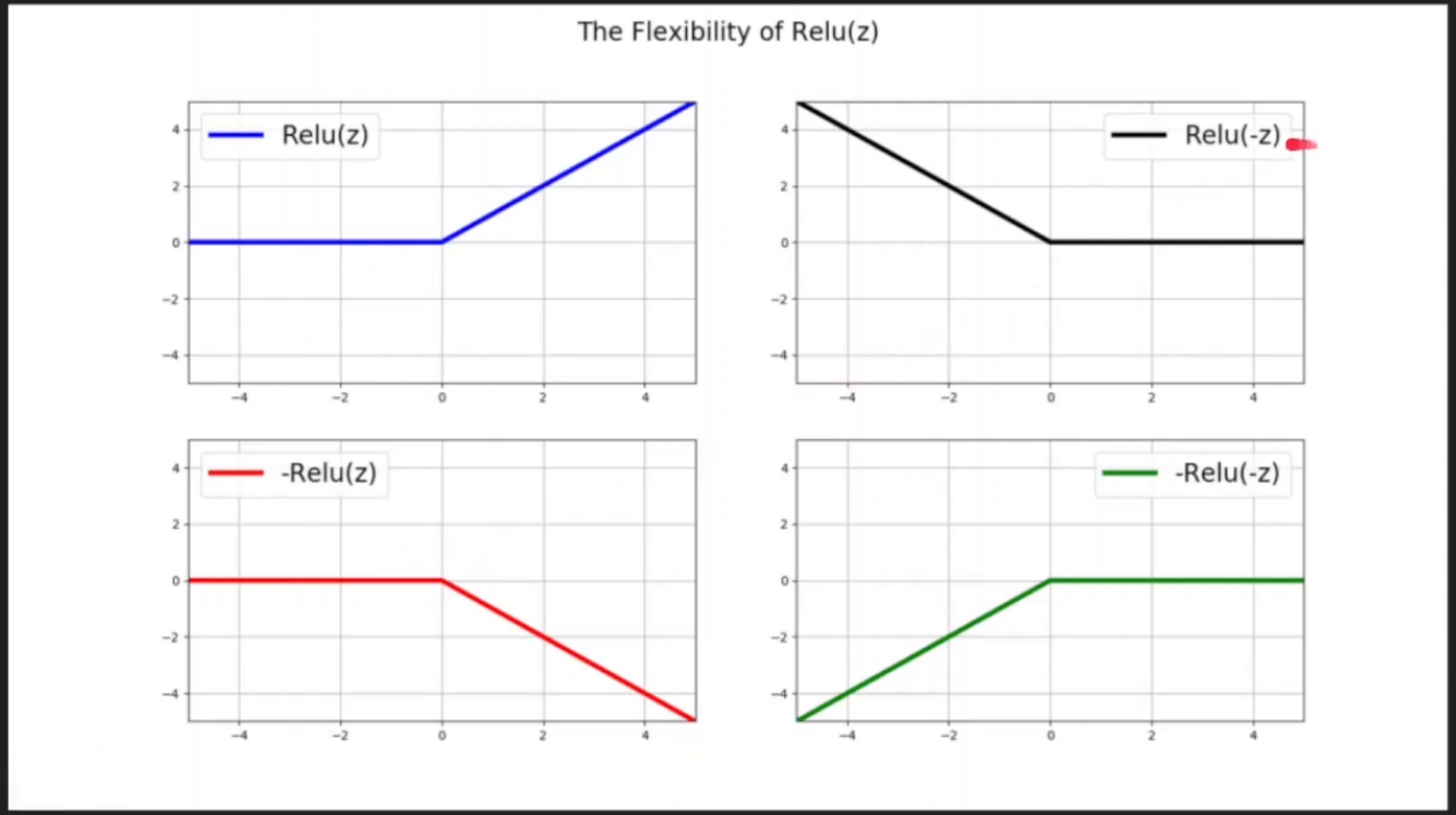
| | | | |
|---|---|---|---|
| 9 | 3 | 5 | -8 |
| -6 | 2 | -3 | 1 |
| 1 | 3 | 4 | 1 |
| 3 | -4 | 5 | 1 |

| | | | |
|---|---|---|---|
| 9 | 3 | 5 | 0 |
| 0 | 2 | 0 | 1 |
| 1 | 3 | 4 | 1 |
| 3 | 0 | 5 | 1 |

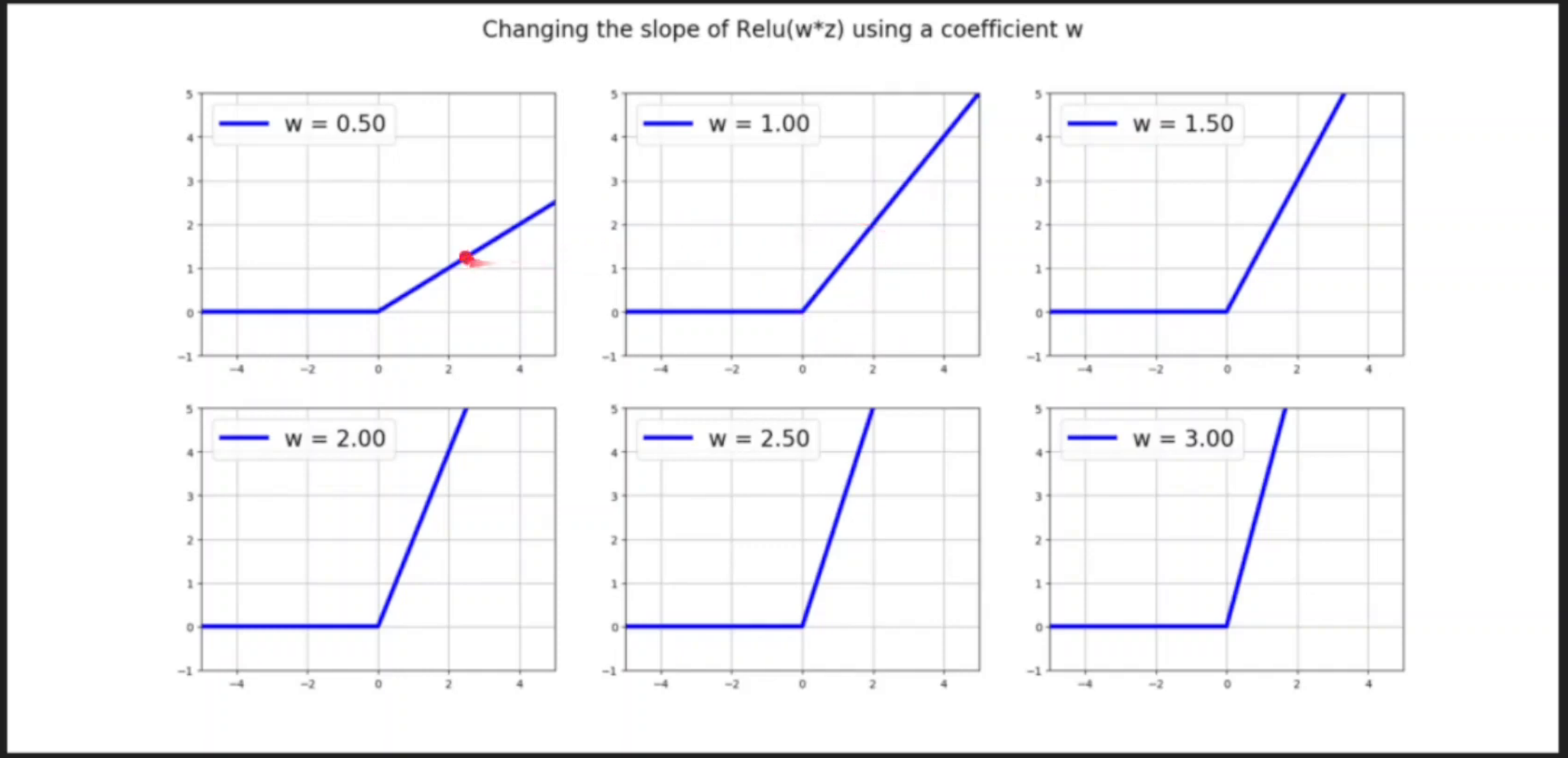
# Fit the Sine wave



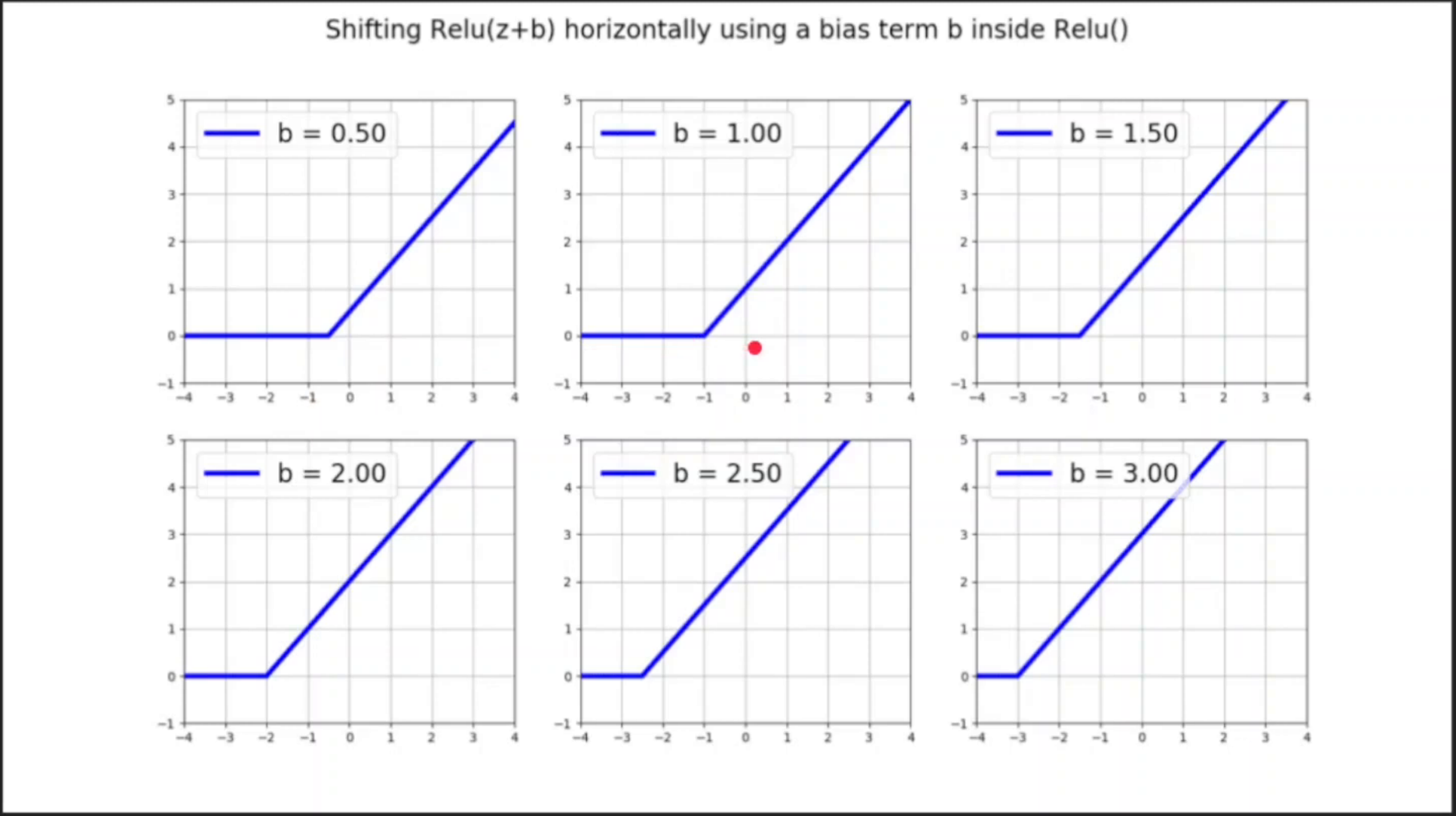
# ReLU function variations - Negative inputs



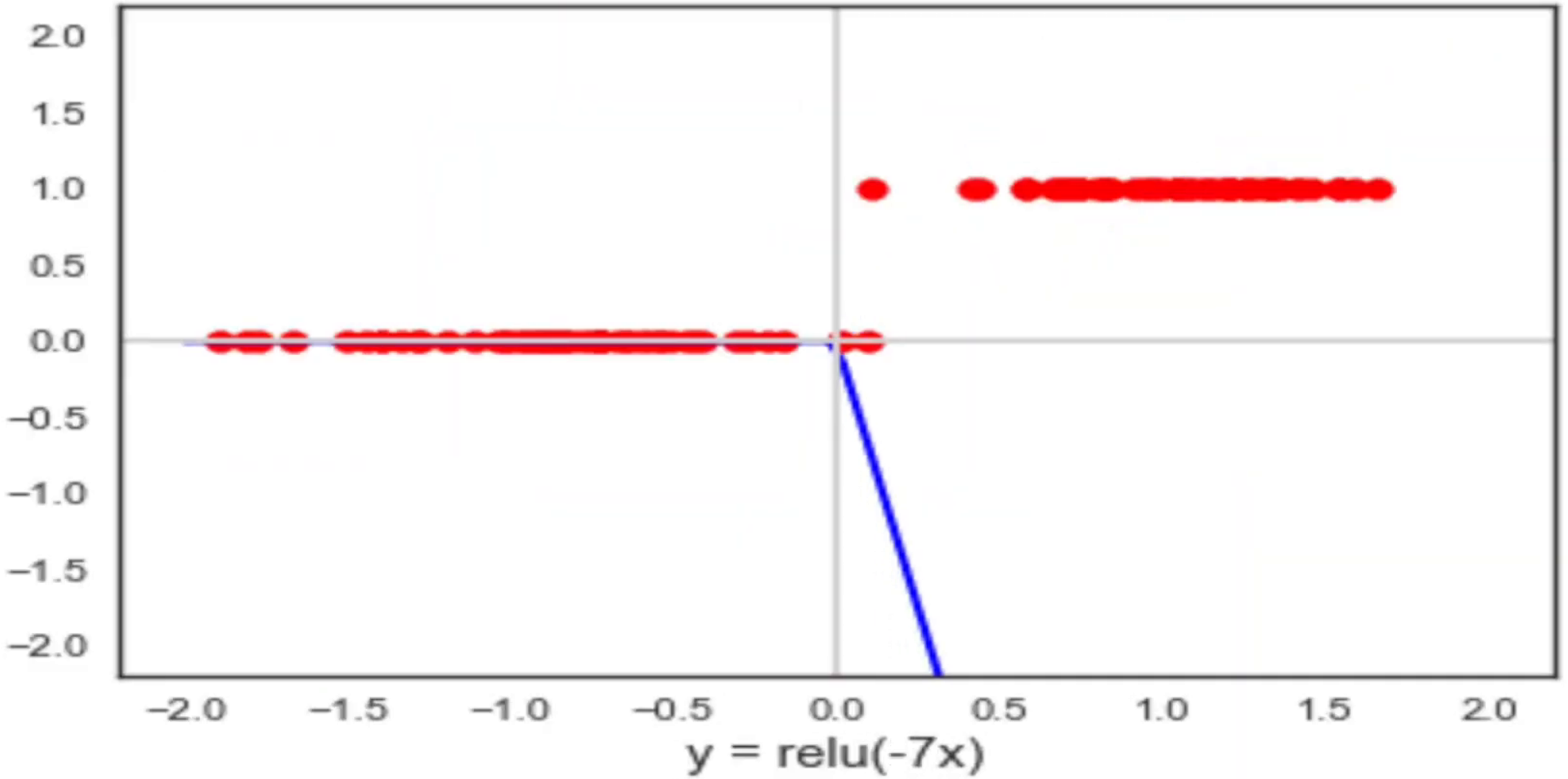
# Changing Slope



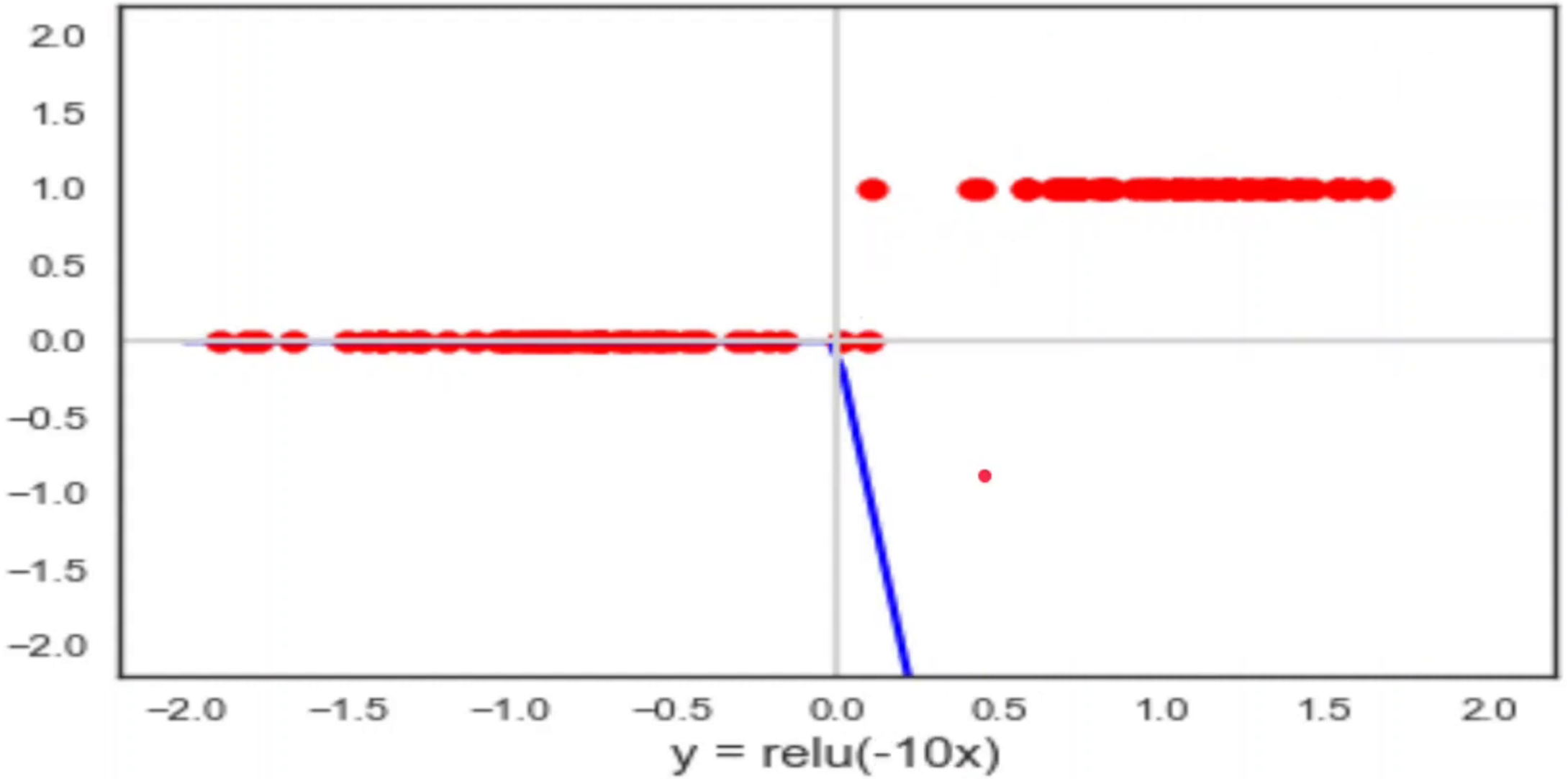
# Changing Bias - Horizontal



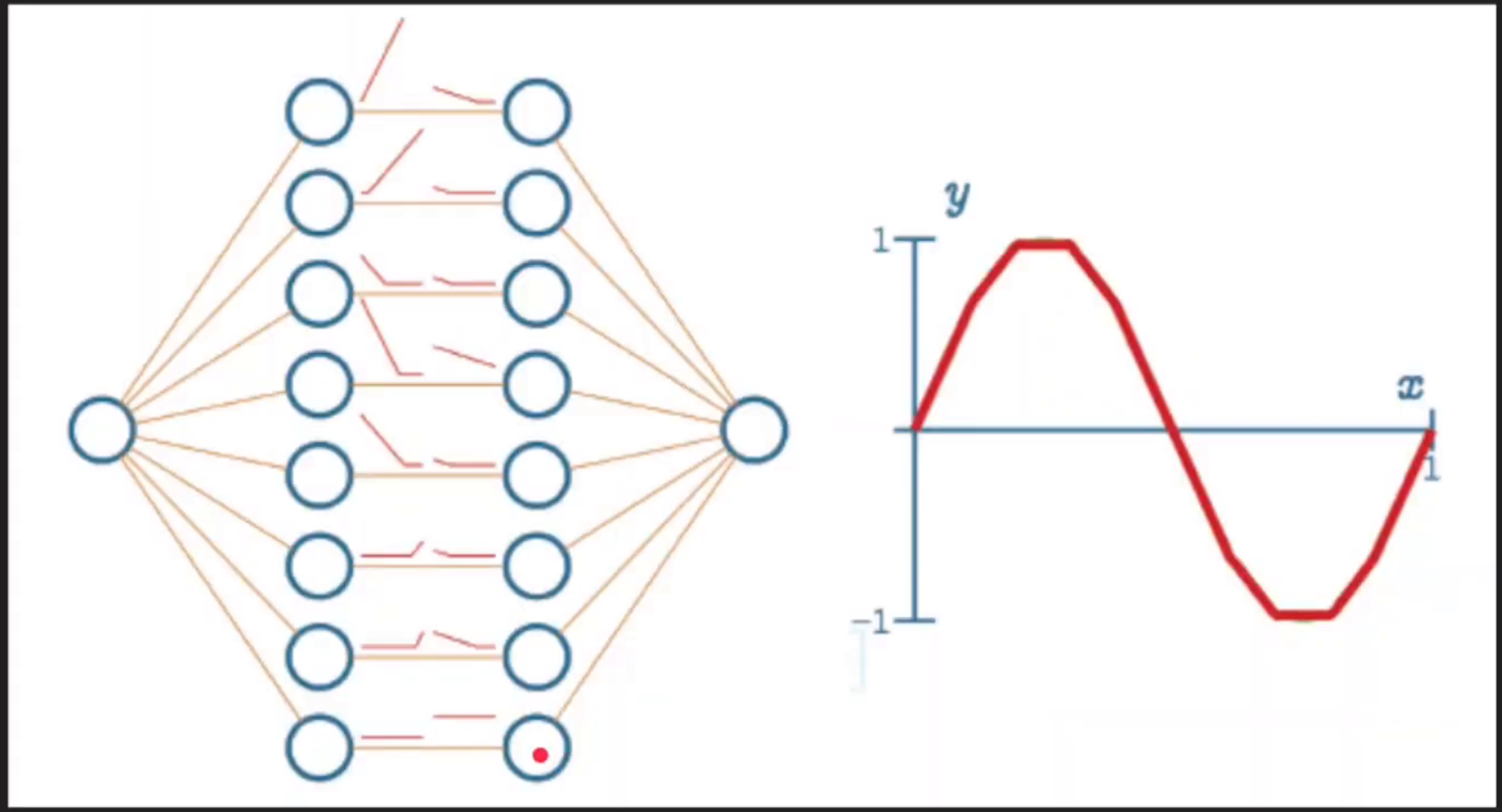
# ReLU in Action



# ReLU in Action

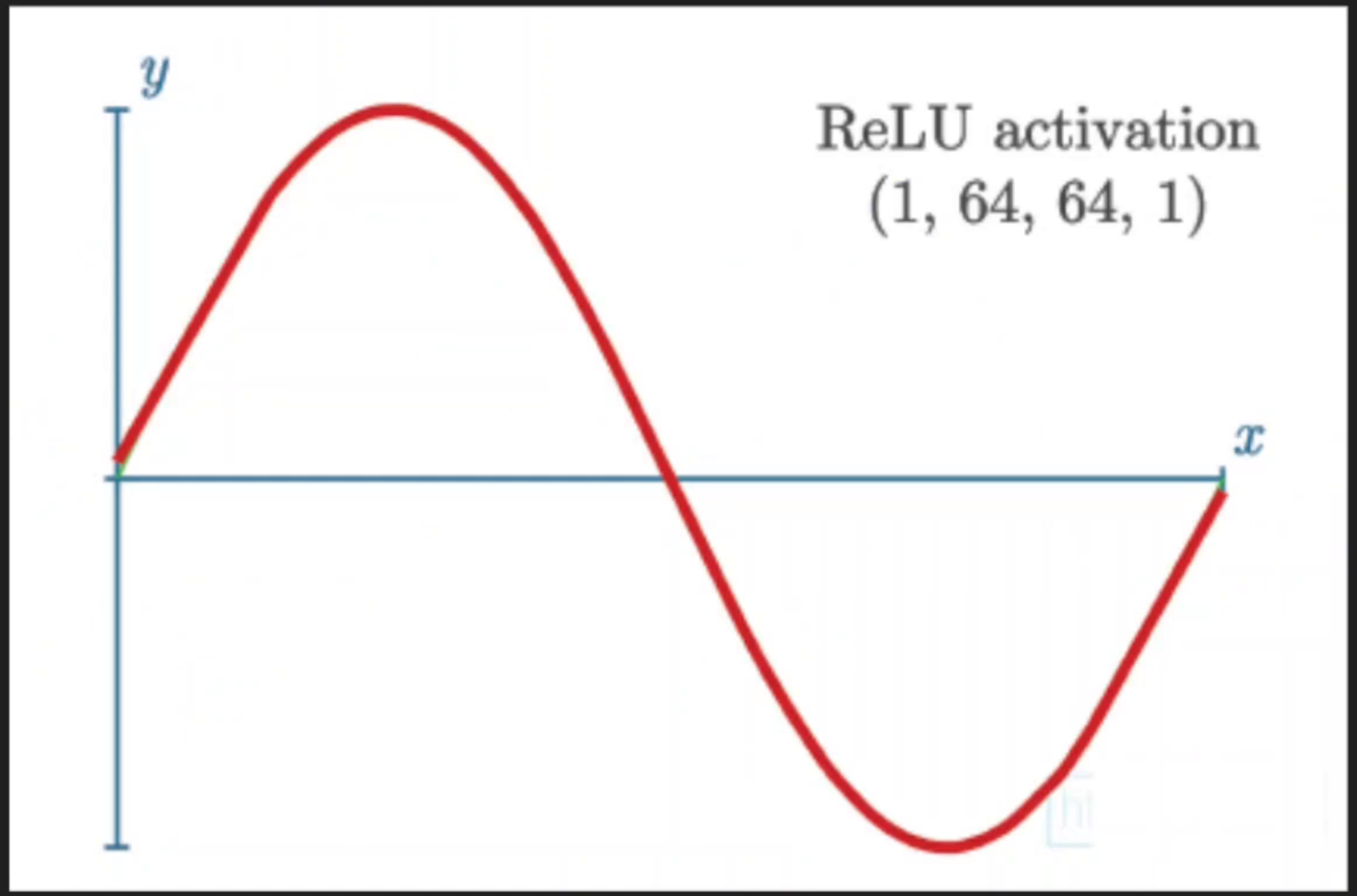


# Sine wave Approximation



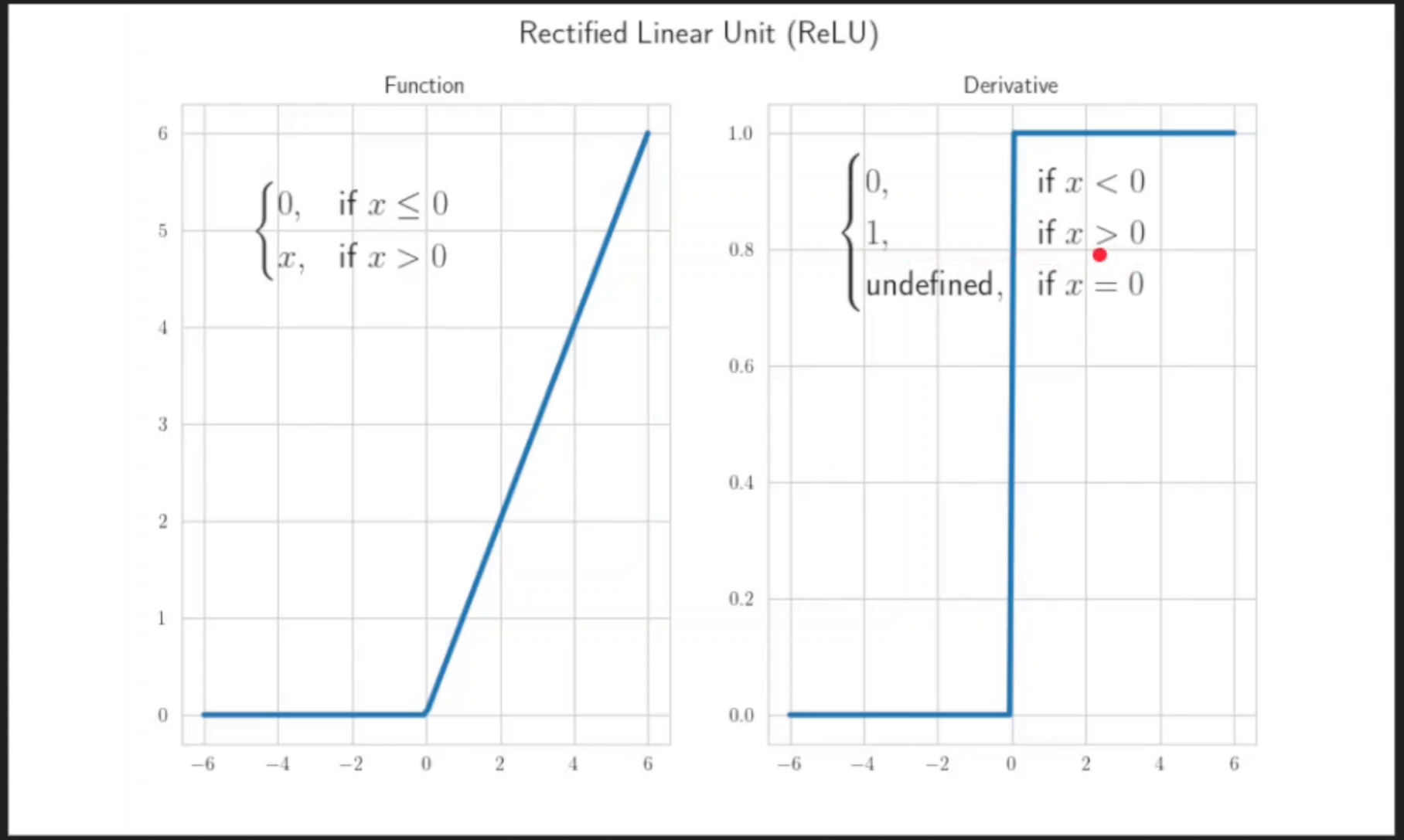

**Arch - (1,8,8,1)**

# Sine wave Approximation

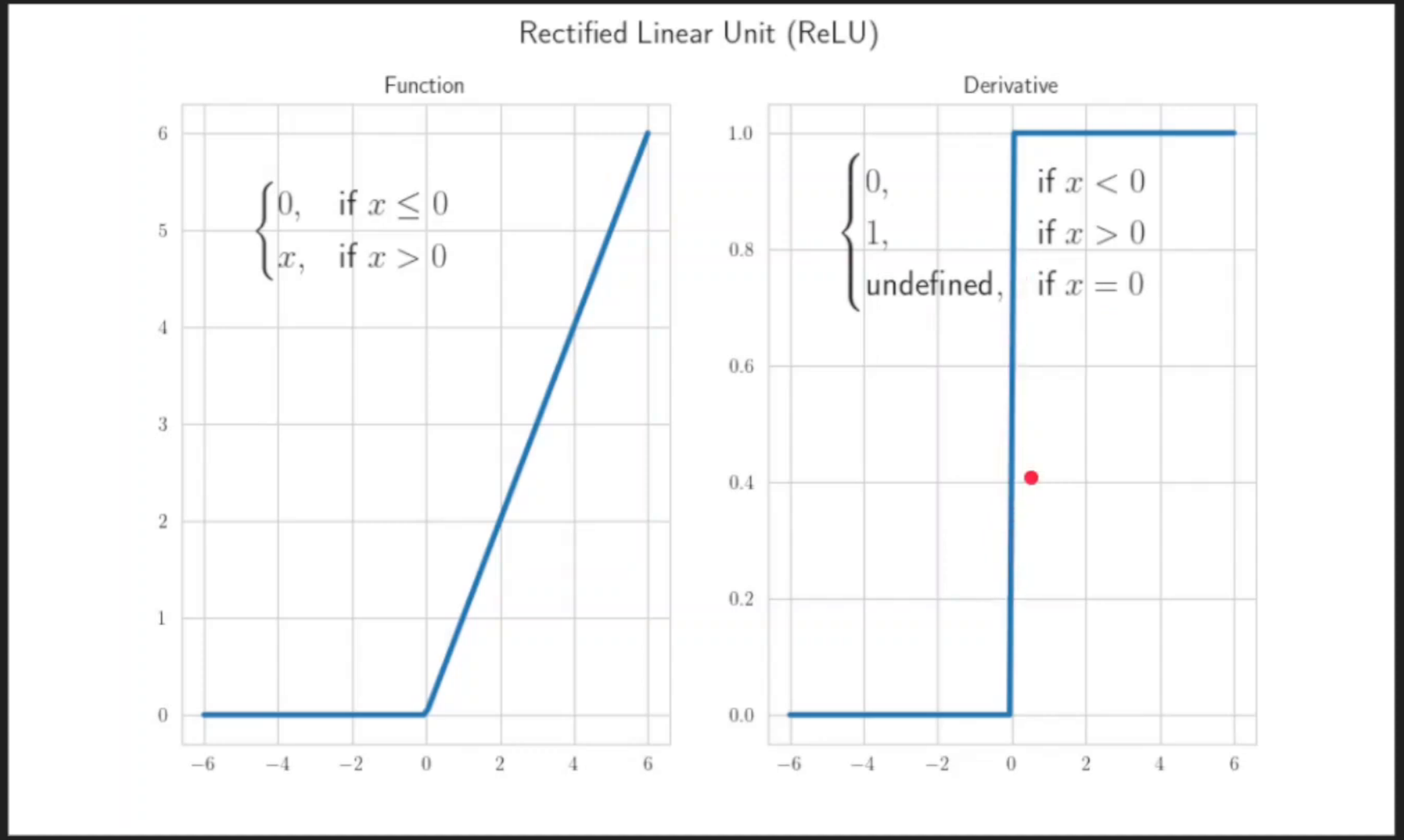


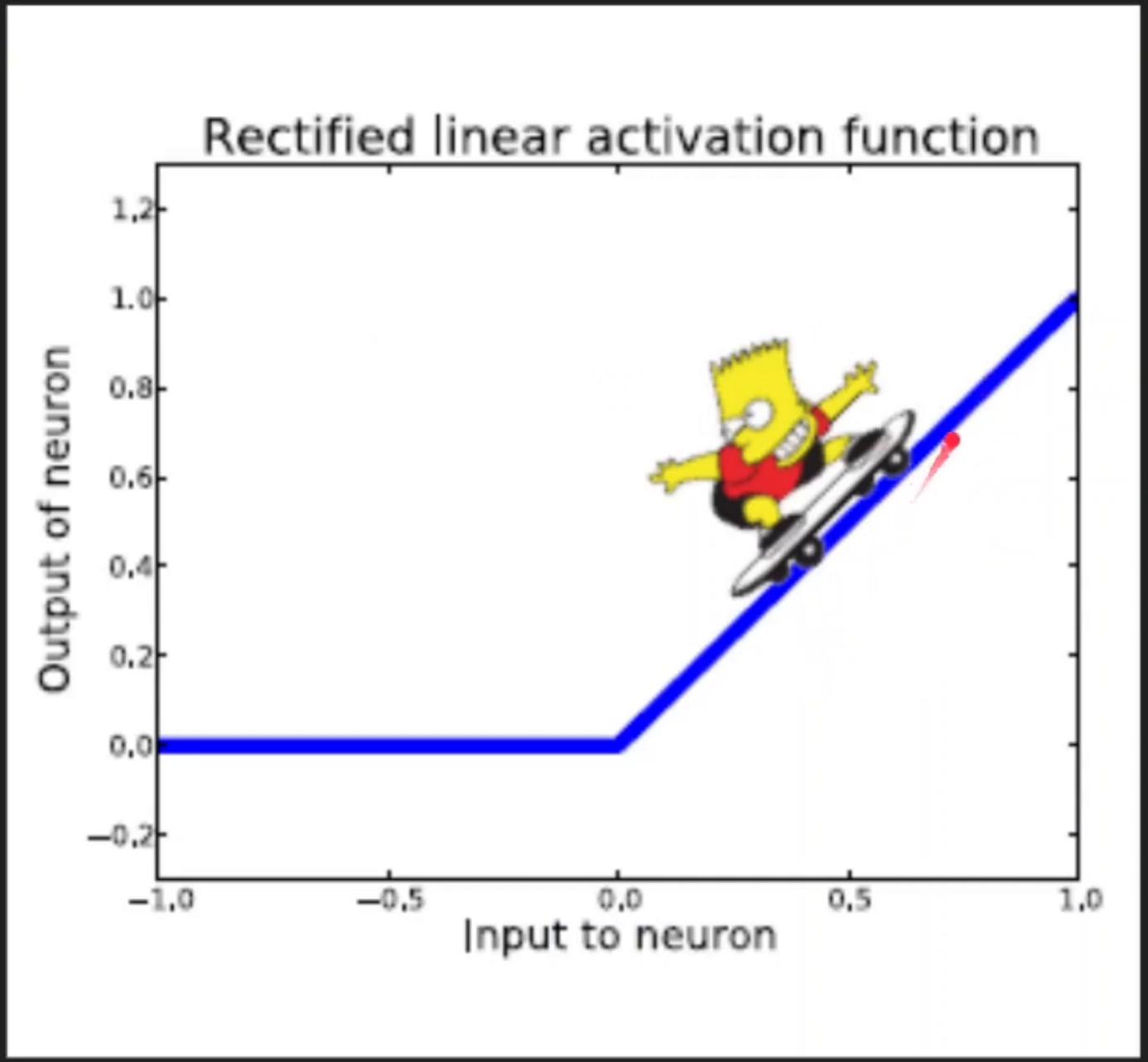

**Arch - (1,64,64,1)**

# Derivative

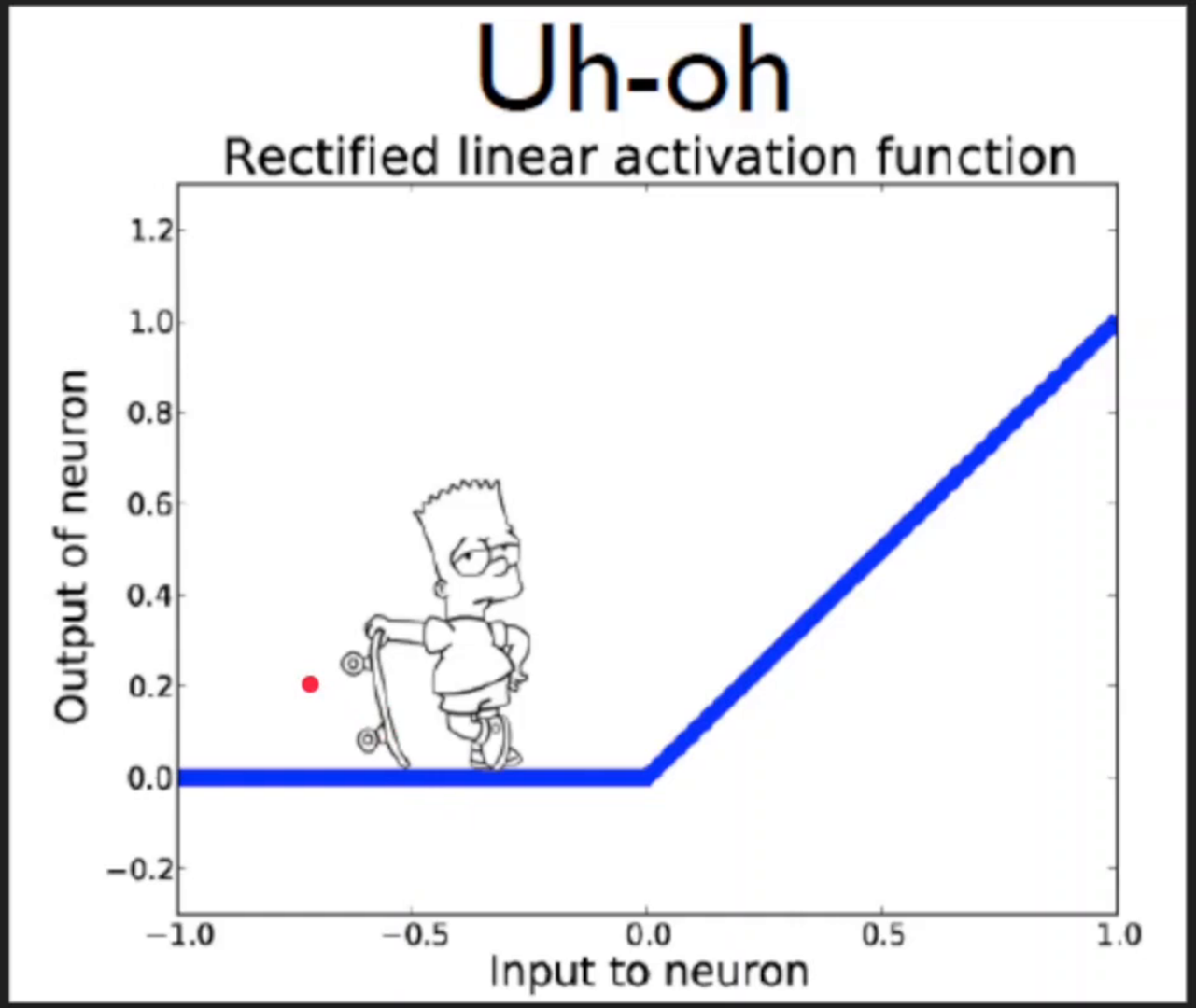

Rectified Linear Unit (ReLU)
Function
$\begin{cases} 0, & \text{if } x \leq 0 \\ x, & \text{if } x > 0 \end{cases}$
6
5
4
3
2
1
0
−6
−4
−2
0
2
4
6
Derivative
$\begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x > 0 \\ \text{undefined}, & \text{if } x = 0 \end{cases}$
1.0
0.8
0.6
0.4
0.2
0.0
−6
−4
−2
0
2
4
6

# Derivative

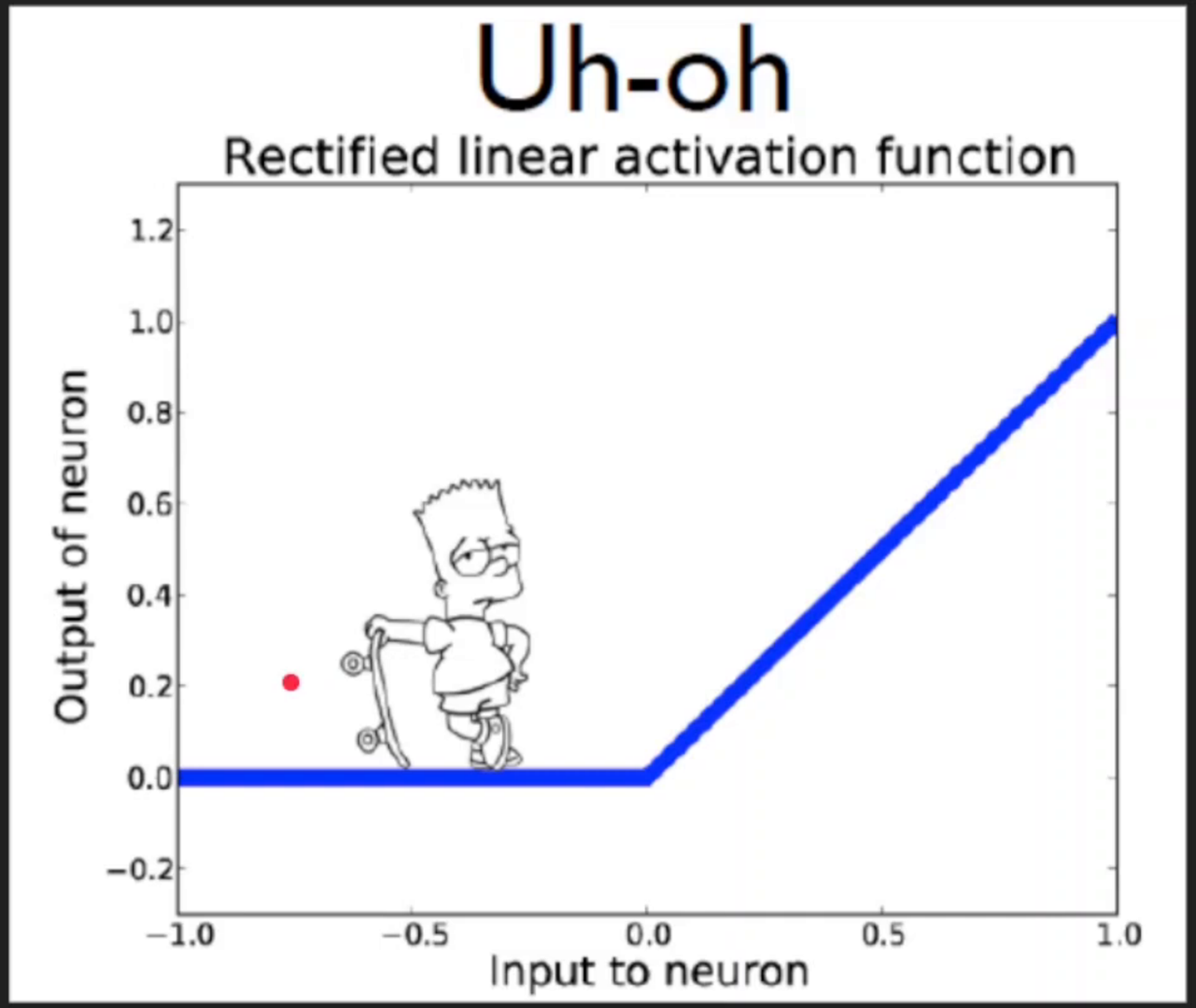

Rectified Linear Unit (ReLU)
Function
$\begin{cases} 0, & \text{if } x \leq 0 \\ x, & \text{if } x > 0 \end{cases}$
Derivative
$\begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x > 0 \\ \text{undefined}, & \text{if } x = 0 \end{cases}$


Rectified linear activation function
Output of neuron
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−1.0
−0.5
0.0
0.5
1.0
Input to neuron

# Dead Neurons

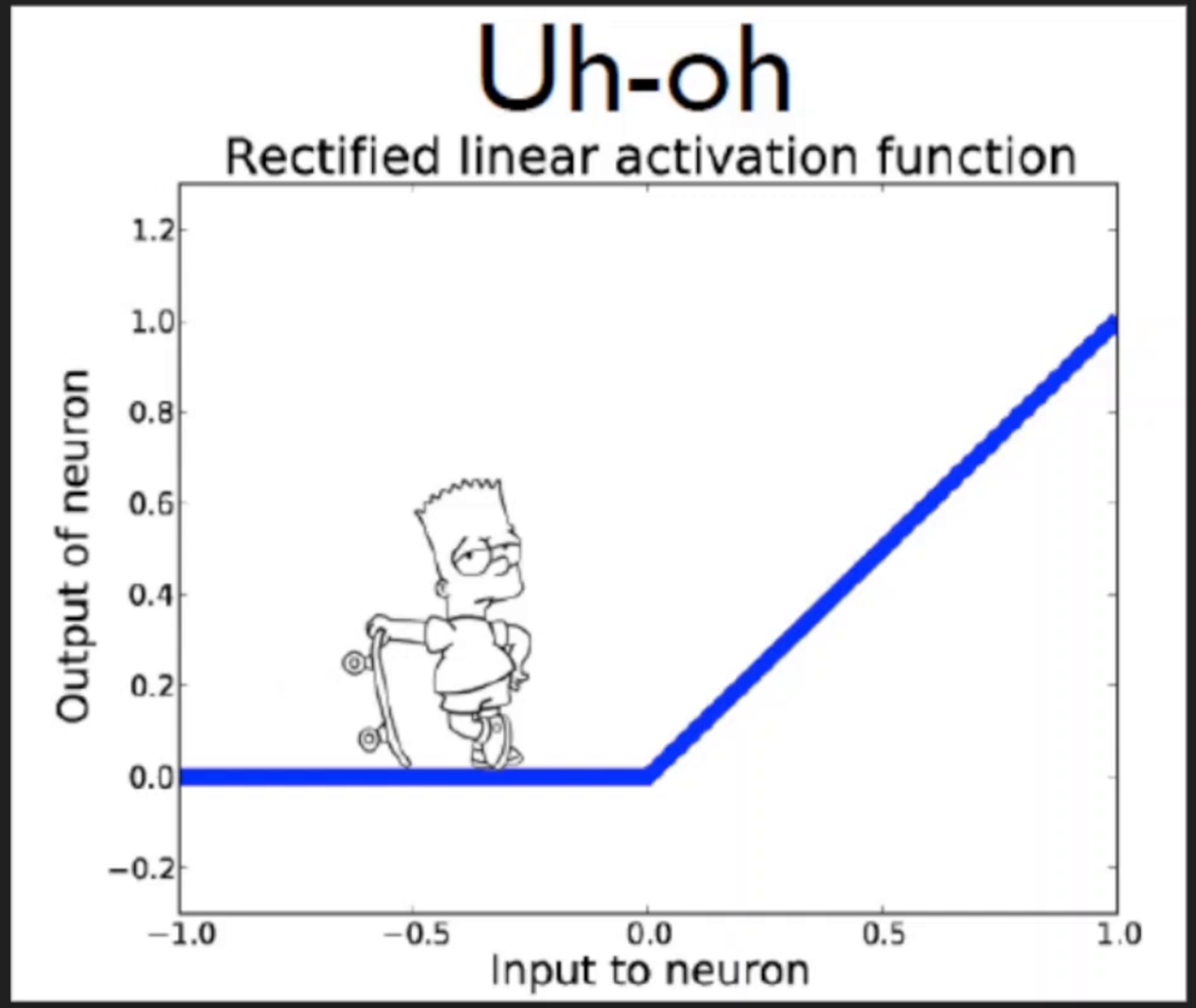


# Dead Neurons



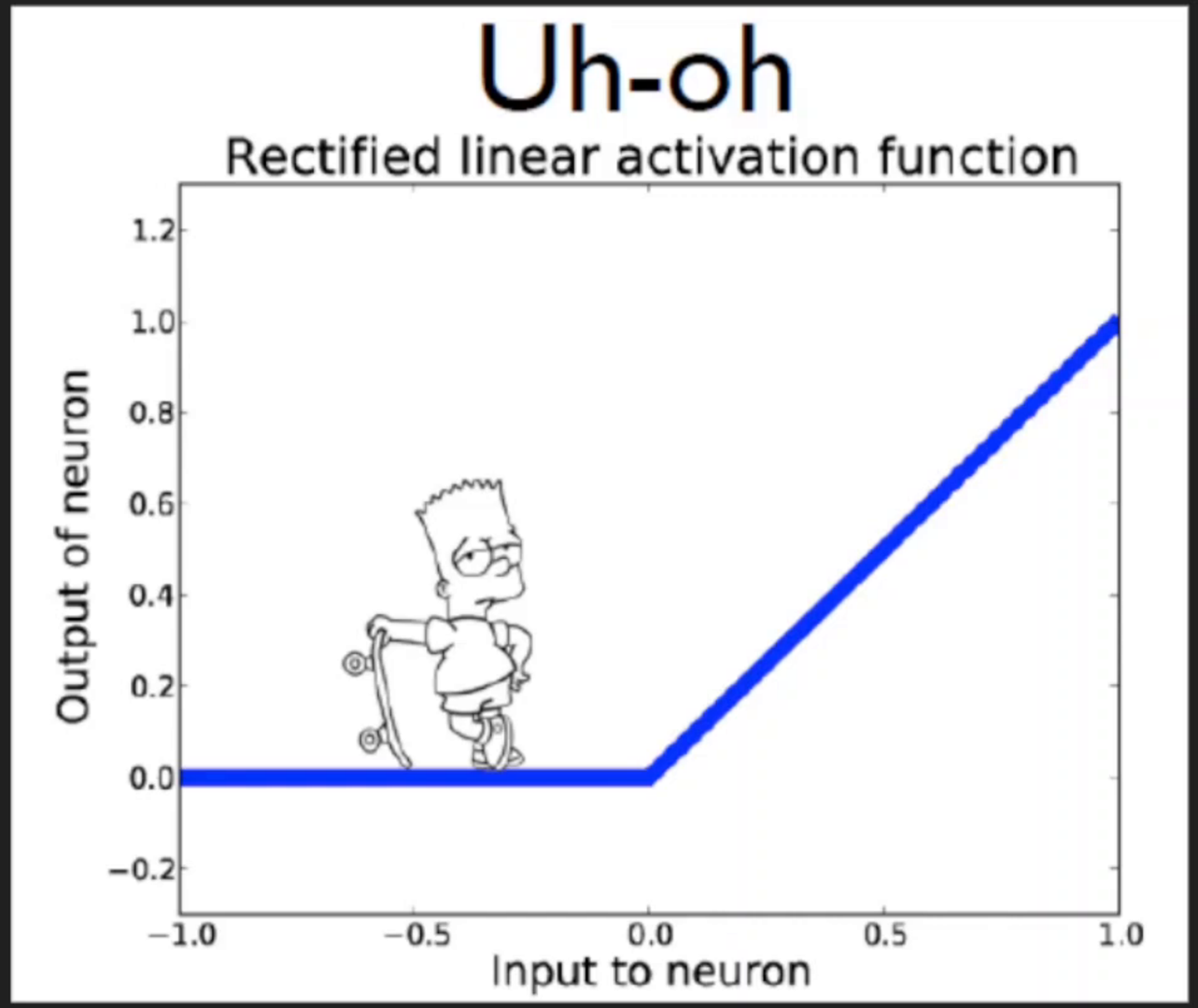
# Dead Neurons

$$Y = \mathrm{ReLU}(W*X + b)$$



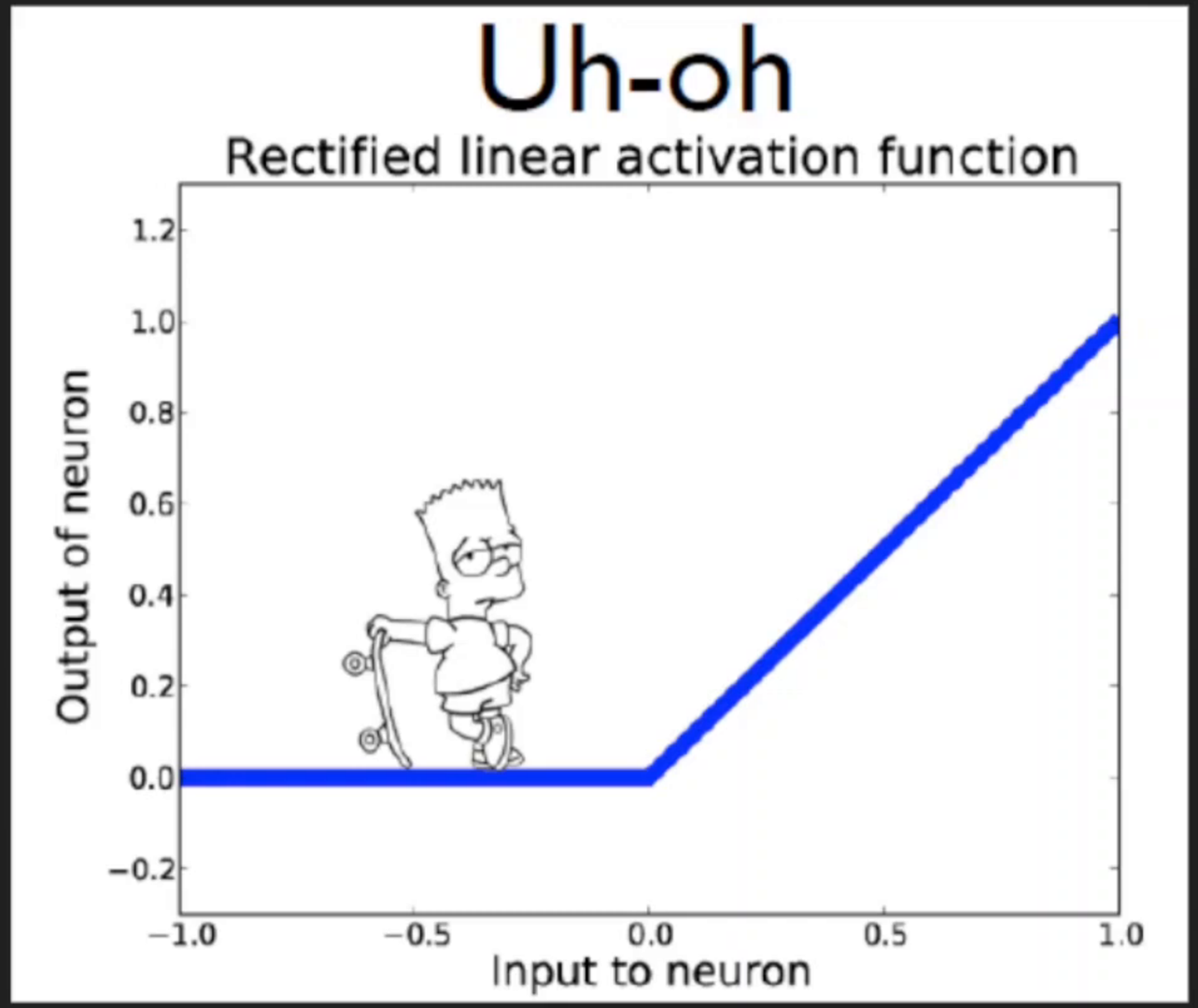
# Dead Neurons

$$Y = \mathrm{ReLU}(W*X + b)$$



# Dead Neurons

$$Y = \mathrm{ReLU}(W*X + b)$$



# Advantages

- Computationally Simple
- Representational Sparsity
- Linear Behaviour
- Train deeper networks

$$ReLU$$

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

# Advantages

- Computationally Simple
- Representational Sparsity
- Linear Behaviour
- Train deeper networks

$$ReLU$$

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

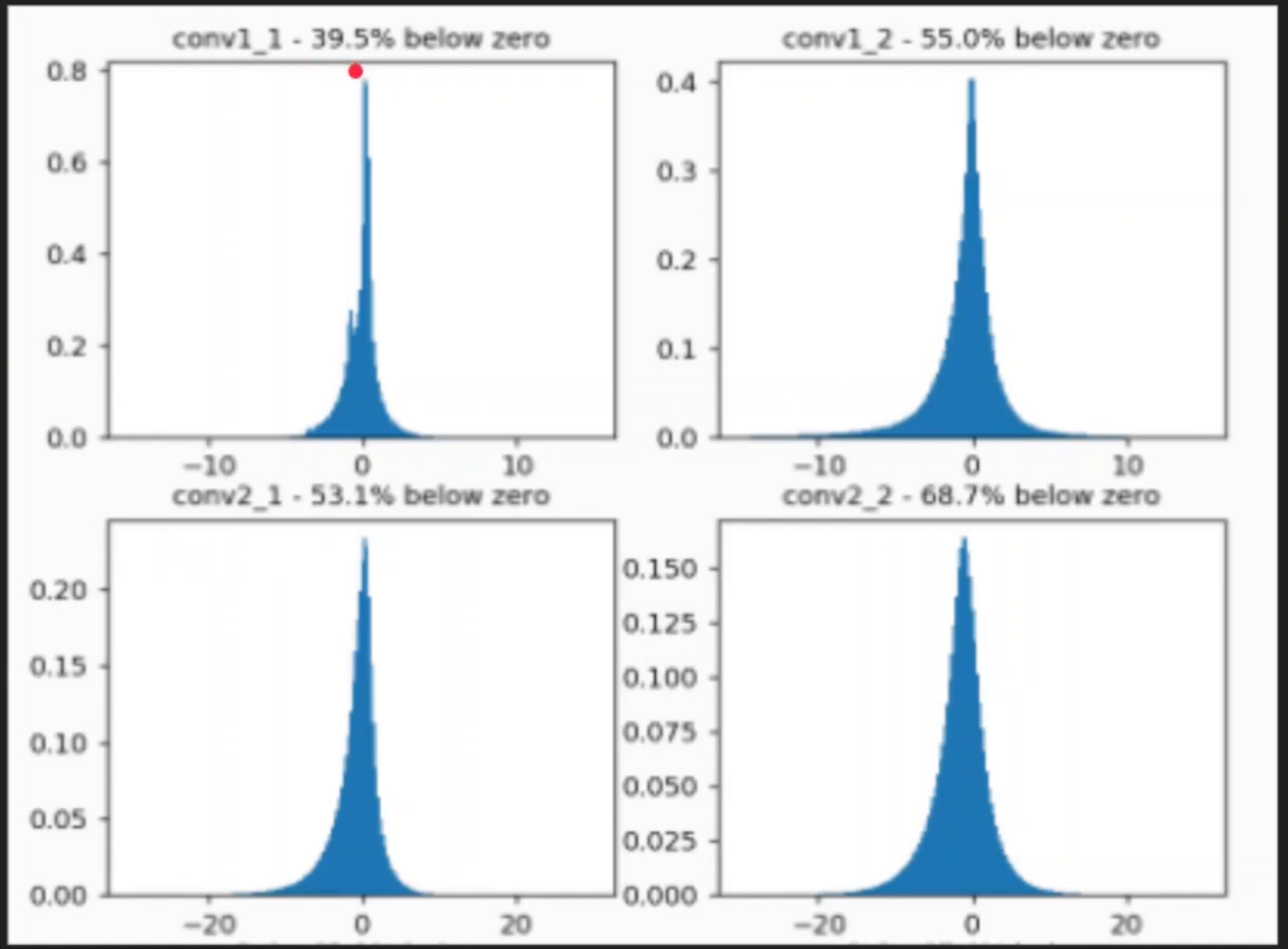
# Advantages

- Computationally Simple
- Representational Sparsity
- Linear Behaviour
- Train deeper networks

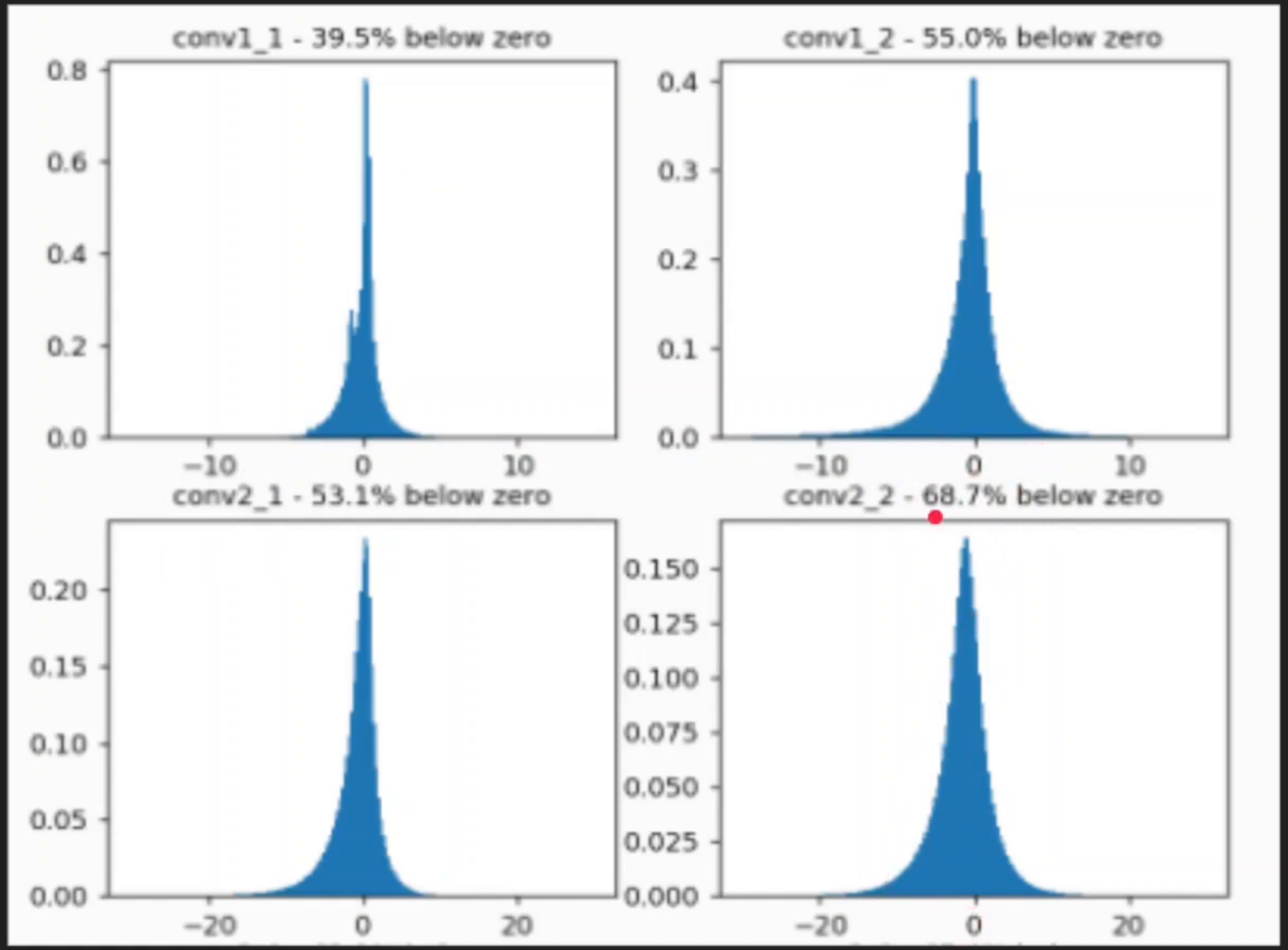
# Advantages

- Computationally Simple
- Representational Sparsity
- Linear Behaviour
- Train deeper networks

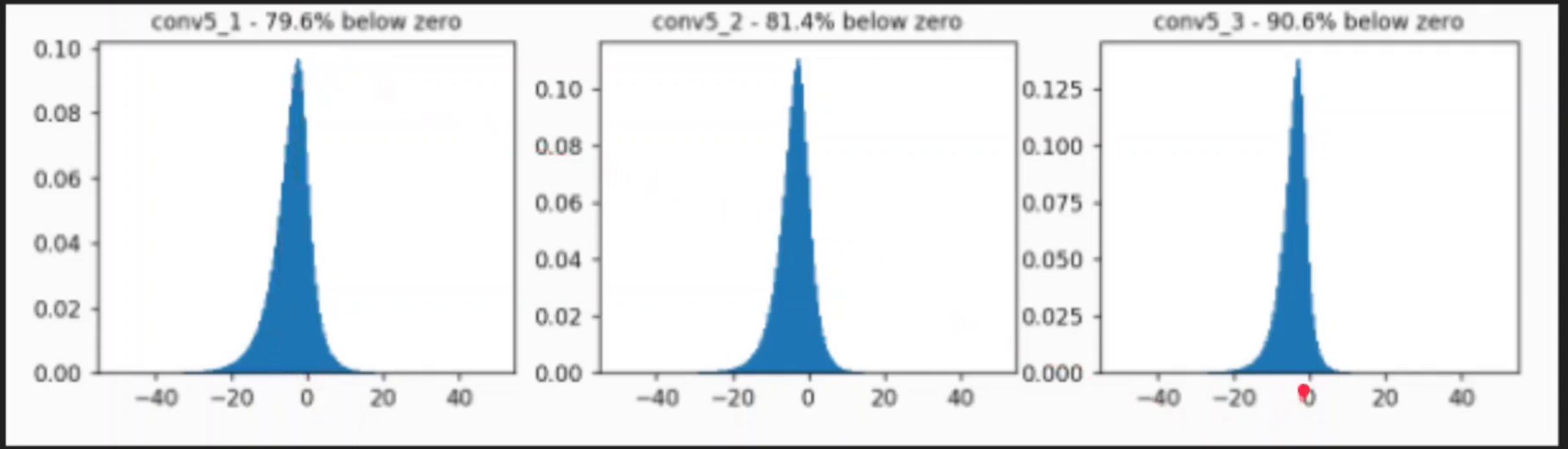
# Negative inputs to Activation



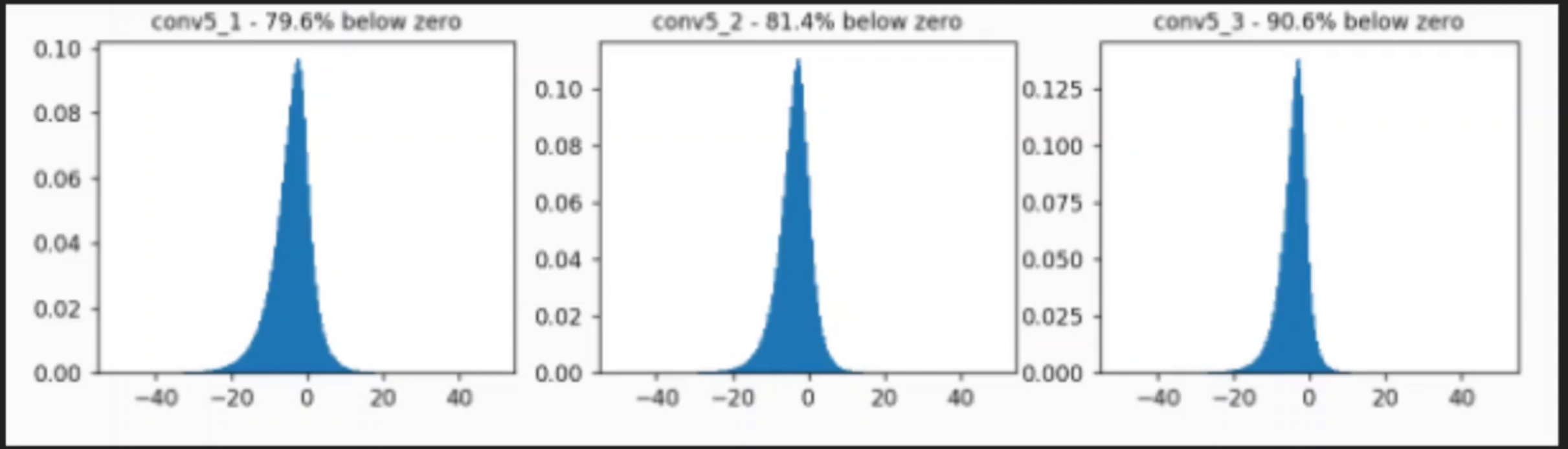
# Negative inputs to Activation



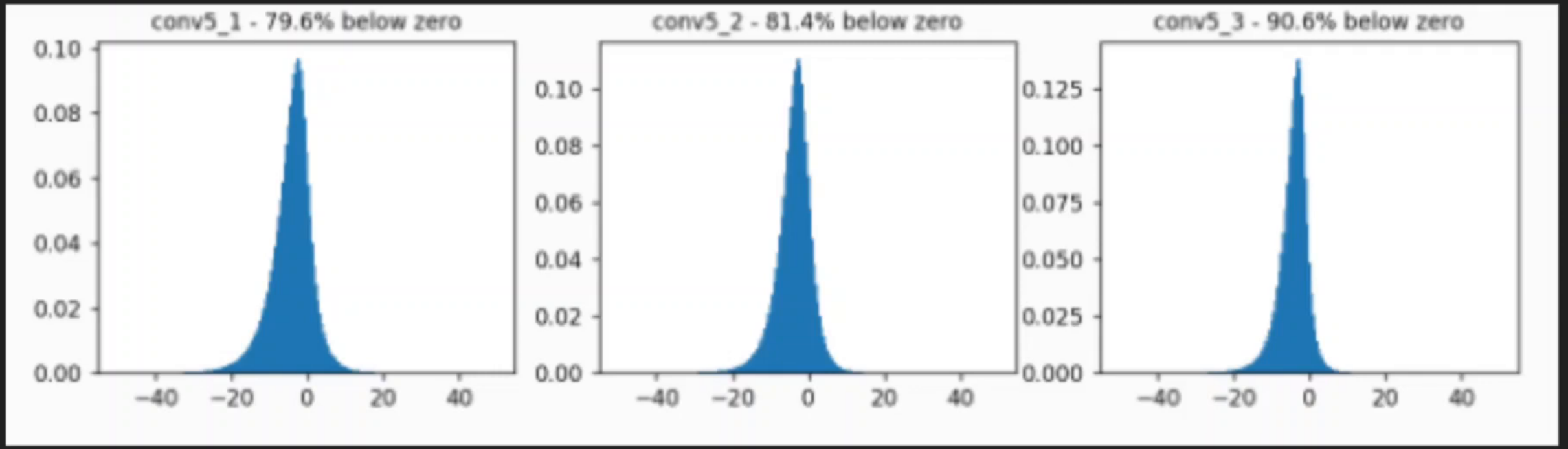
# Negative inputs to Activation



# Negative inputs to Activation



# Negative inputs to Activation



# Advantages

- Computationally Simple
- Representational Sparsity
- Linear Behaviour
- Train deeper networks

# Advantages

- Computationally Simple
- Representational Sparsity
- Linear Behaviour
- Train deeper networks

# Drawbacks

- Exploding Gradient
- Dead Neurons

# Drawbacks

- Exploding Gradient
- Dead Neurons

# Drawbacks

- Exploding Gradient
- Dead Neurons

$$W_{new} = W_{old} - \eta \Delta W$$

Learning rate

# Drawbacks

- Exploding Gradient
- Dead Neurons

$$W_{new} = W_{old} - \eta \Delta W$$

Learning rate

# Drawbacks

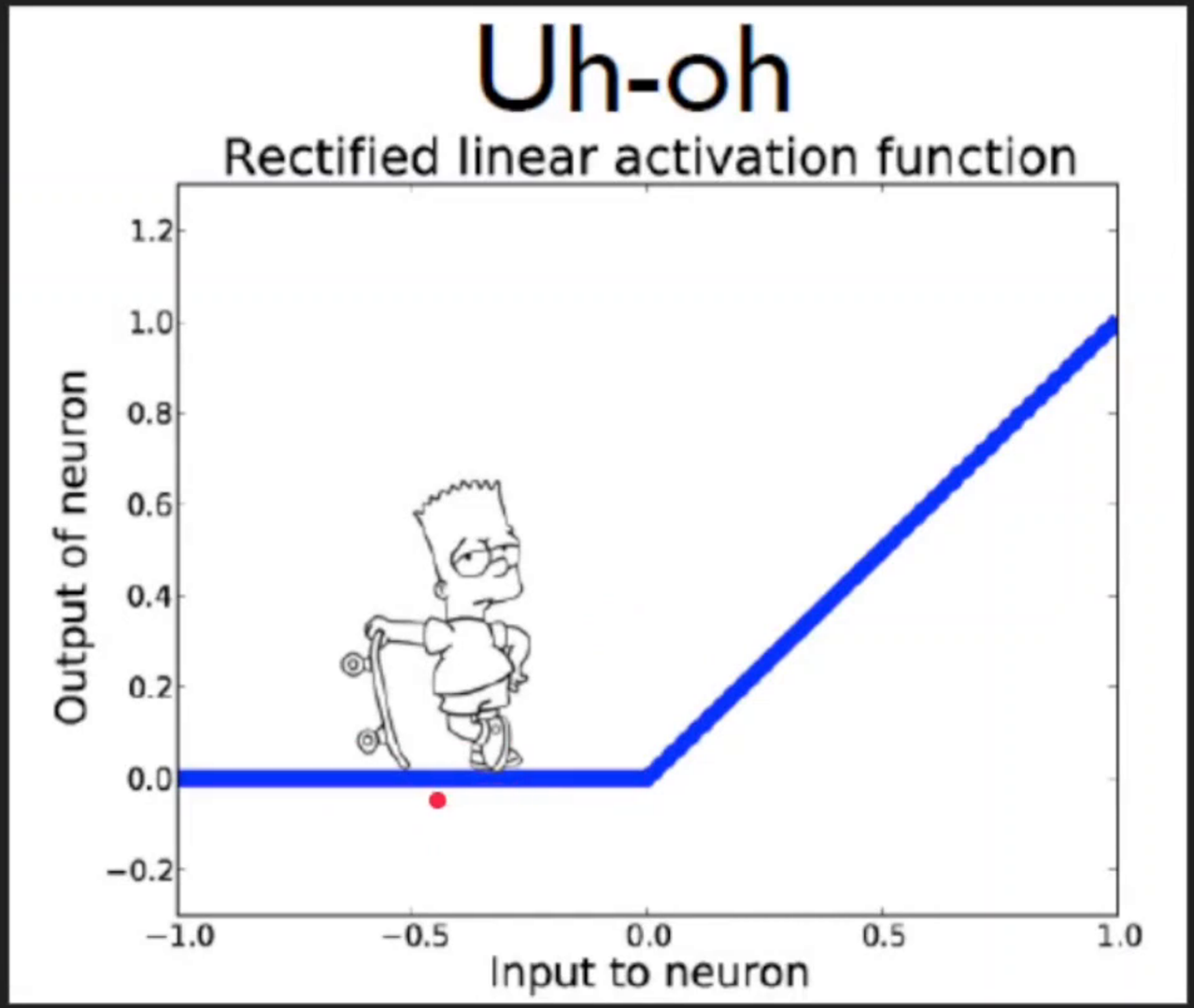
- Exploding Gradient
- Dead Neurons

$$W_{new} = W_{old} - \eta \Delta W$$

Learning rate

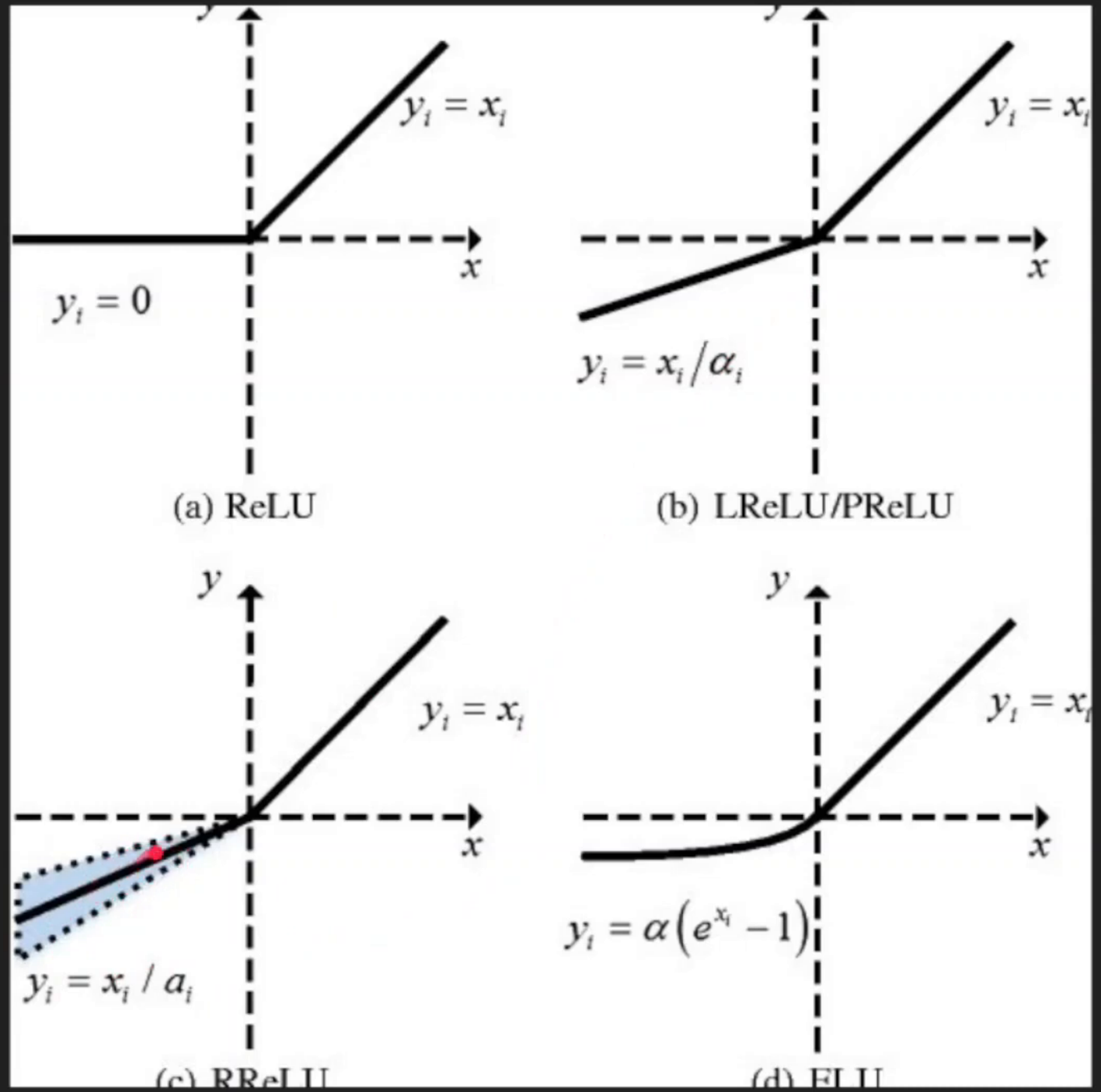
# Drawbacks

- Exploding Gradient
- Dead Neurons



# Drawbacks

- Exploding Gradient
- Dead Neurons



# Tips

- Default Activation
- Smaller bias values
- He Weight Initialization
- Scale input data
- Weight penalty

# Tips

- Default Activation
- Smaller bias values
- He Weight Initialization
- Scale input data
- Weight penalty

# Tips

- Default Activation
- Smaller bias values
- He Weight Initialization
- Scale input data
- Weight penalty

# Tips

- Default Activation
- Smaller bias values
- He Weight Initialization
- Scale input data
- Weight penalty

# Tips

- Default Activation
- Smaller bias values
- He Weight Initialization
- Scale input data
- Weight penalty

# Tips

- Default Activation
- Smaller bias values
- He Weight Initialization
- Scale input data
- Weight penalty

# Tips

- Default Activation
- Smaller bias values
- He Weight Initialization
- Scale input data
- Weight penalty

# Tips

- Default Activation
- Smaller bias values
- He Weight Initialization
- Scale input data
- Weight penalty

# Tips

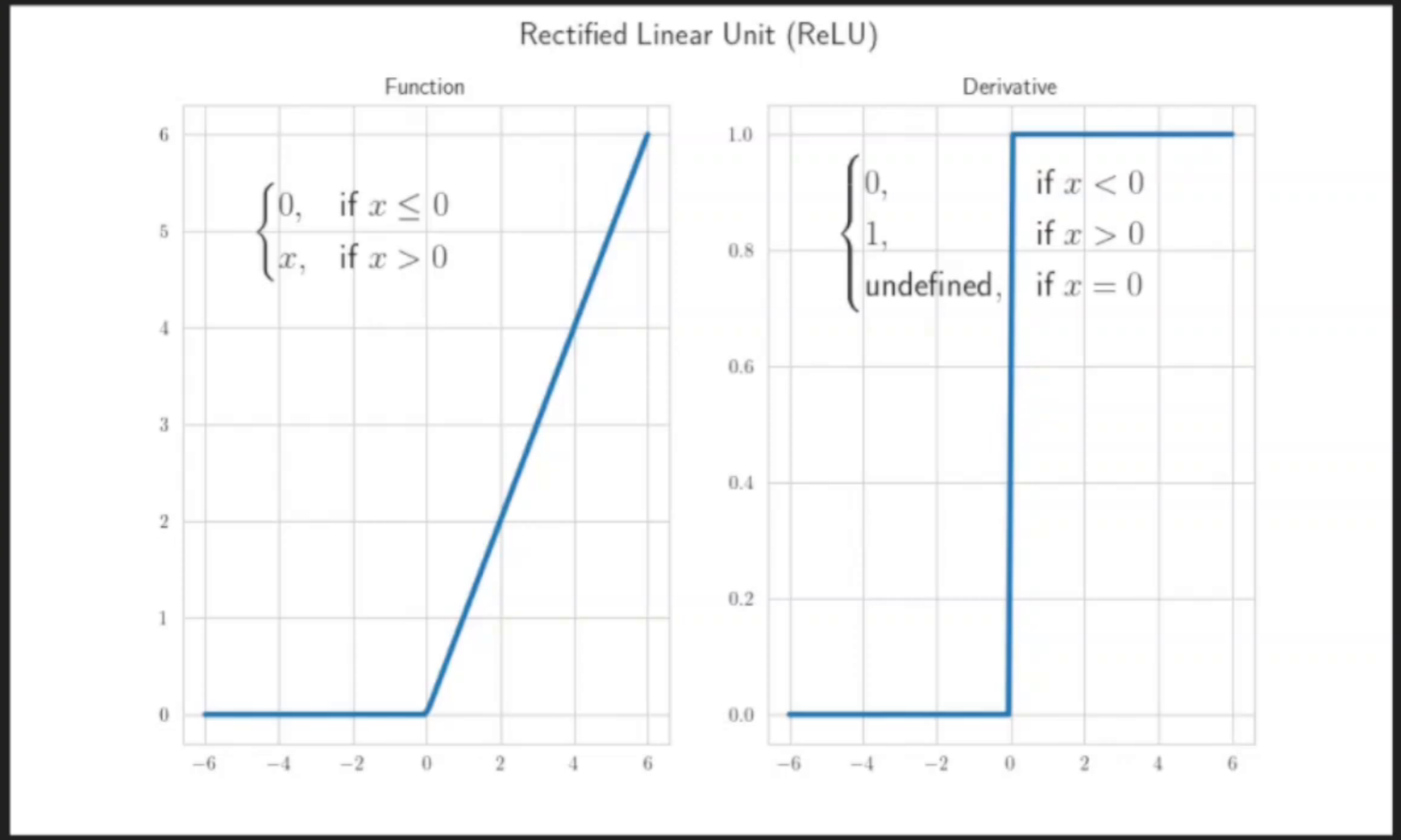
- Default Activation
- Smaller bias values
- He Weight Initialization
- Scale input data
- Weight penalty

# Python Implementation

```
def relu(x):
    return max(0,x)
```

**Derivative:**

```
    if x>0:
        return 1
    else:
        return 0
```



# Python Implementation

```
def relu(x):
    return max(0,x)
```

**Derivative:**

```
if x>0:
    return 1
else:
    return 0
```