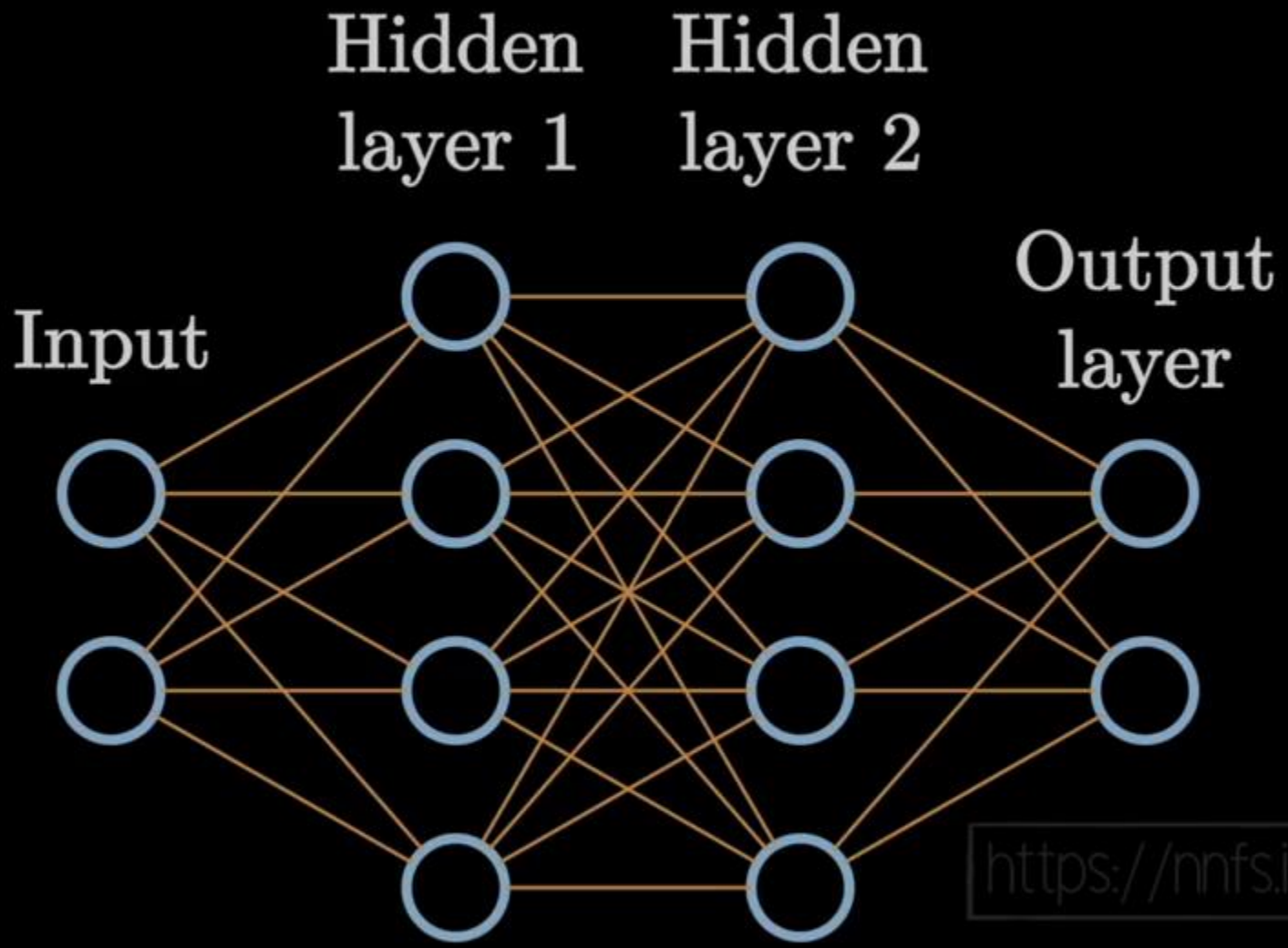
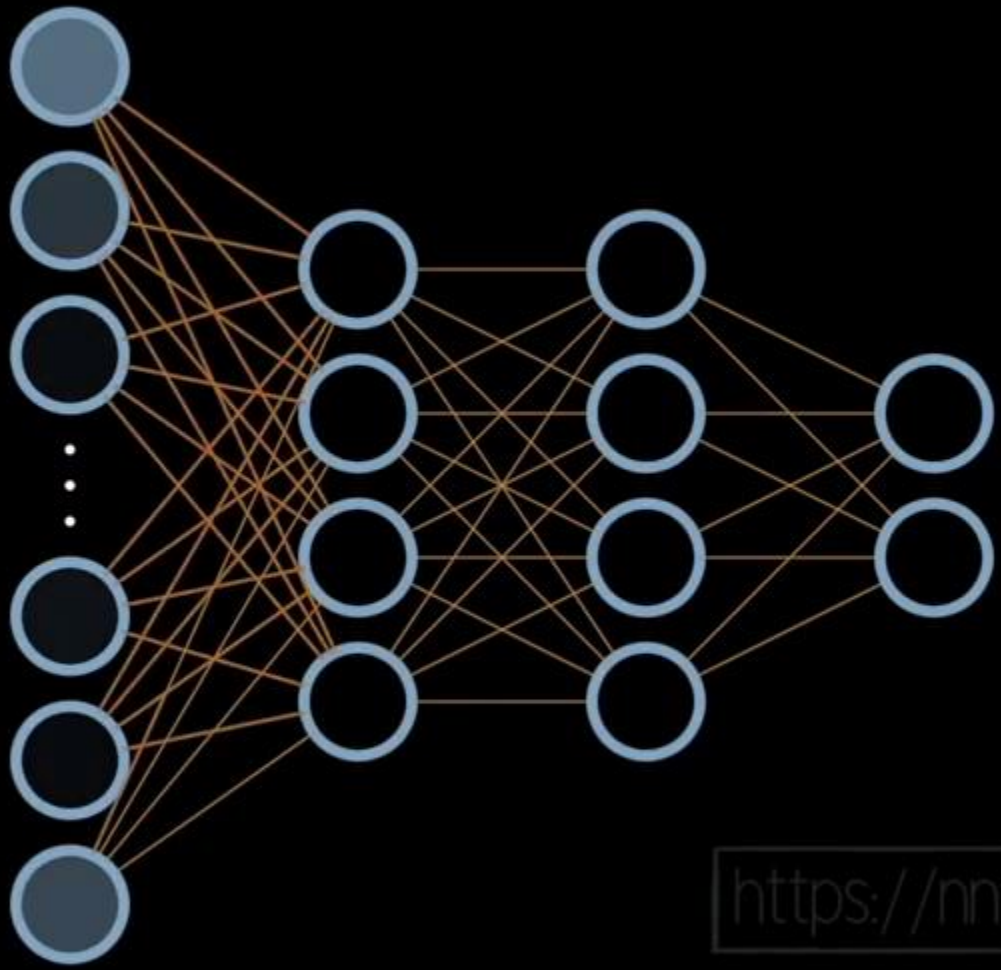
Neural Networks from Scratch in Python

Harrison Kinsley

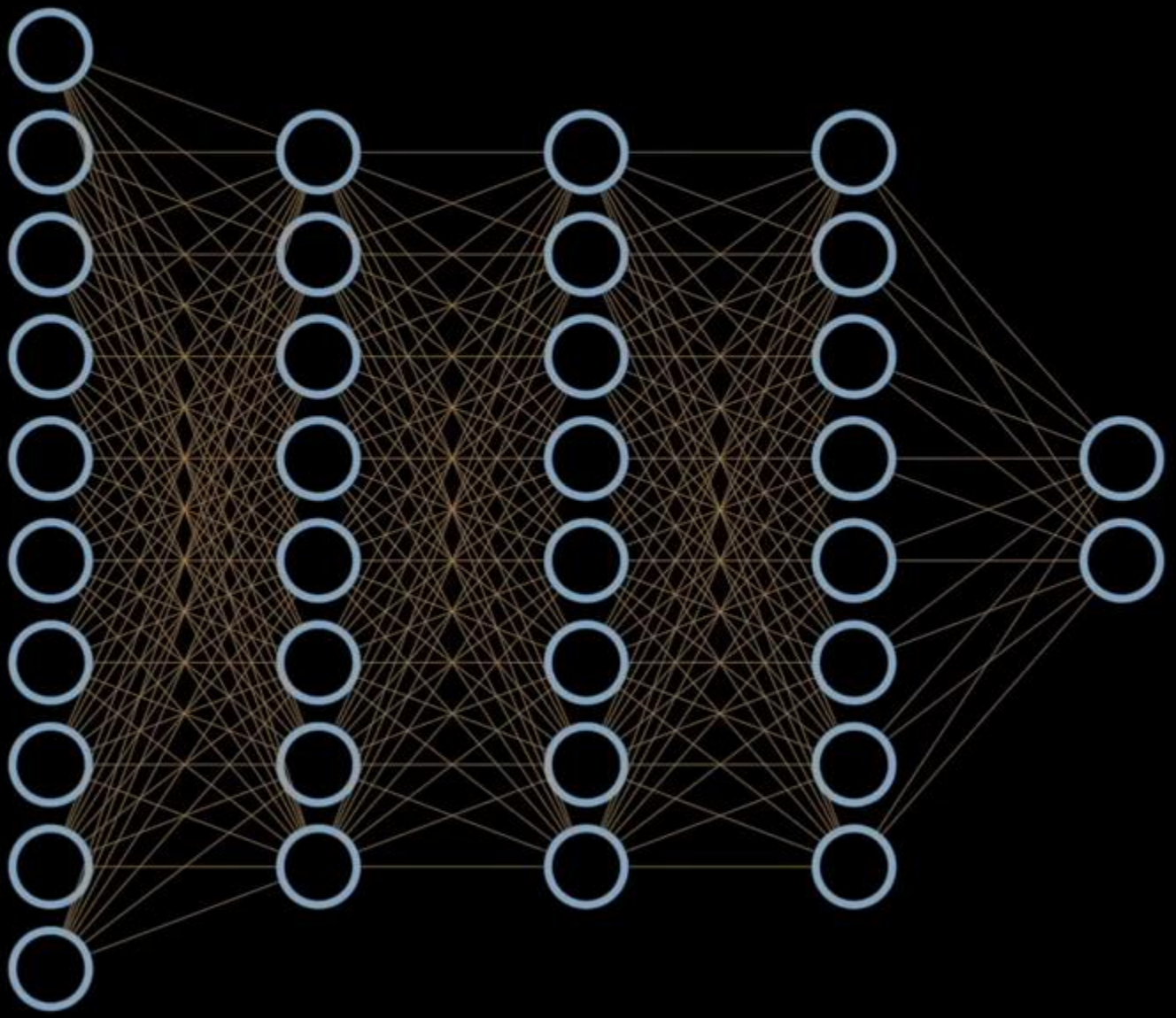
```
loss = -np.log(
    np.sum(
        y * np.exp(
            np.dot(
                np.maximum(
                    0,
                    np.dot(
                        np.maximum(
                            0,
                            np.dot(
                                X,
                                w1.T
                            ) + b1
                        ),
                        w2.T
                    ) + b2
                ),
                w3.T
            ) + b3
        ) /
        np.sum(
            np.exp(
                np.dot(
                    np.maximum(
                        0,
                        np.dot(
                            np.maximum(
                                0,
                                np.dot(
                                    X,
                                    w1.T
                                ) + b1
                            ),
                            w2.T
                        ) + b2
                    ),
                    w3.T
                ) + b3
            ),
            axis=1,
            keepdims=True
        )
    )
)
```


Hidden
layer 1
Hidden
layer 2
Input
Output
layer
https://nnfs.io


https://nnfs.io

Layer sizes: 10, 8, 8, 8, 2

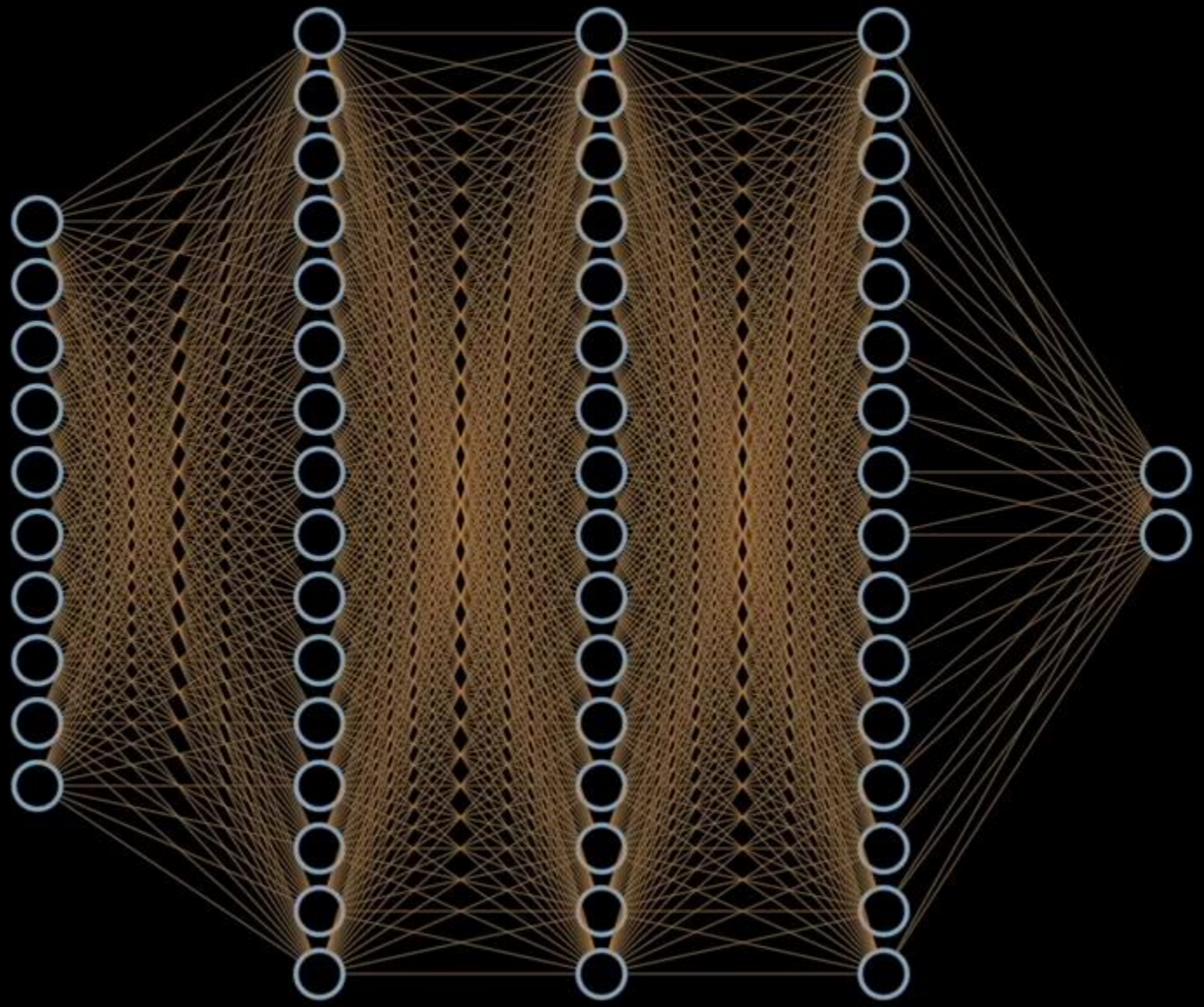
Weights:	224
Biases:	36
Params:	260



https://nnfs.io

Layer sizes: 10, 16, 16, 16, 2

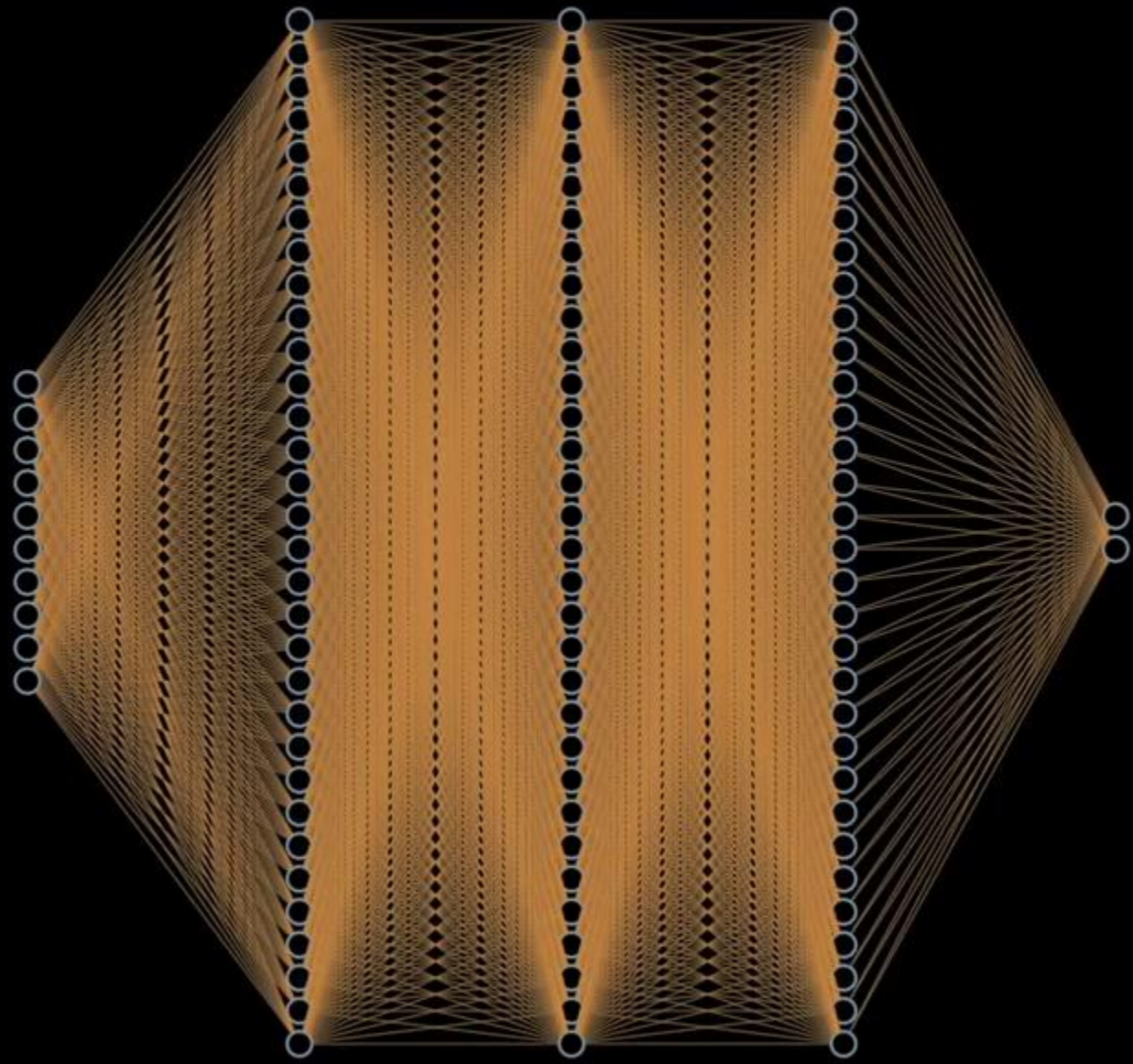
Weights:	704
Biases:	60
Params:	764



https://nnfs.io

Layer sizes: 10, 32, 32, 32, 2

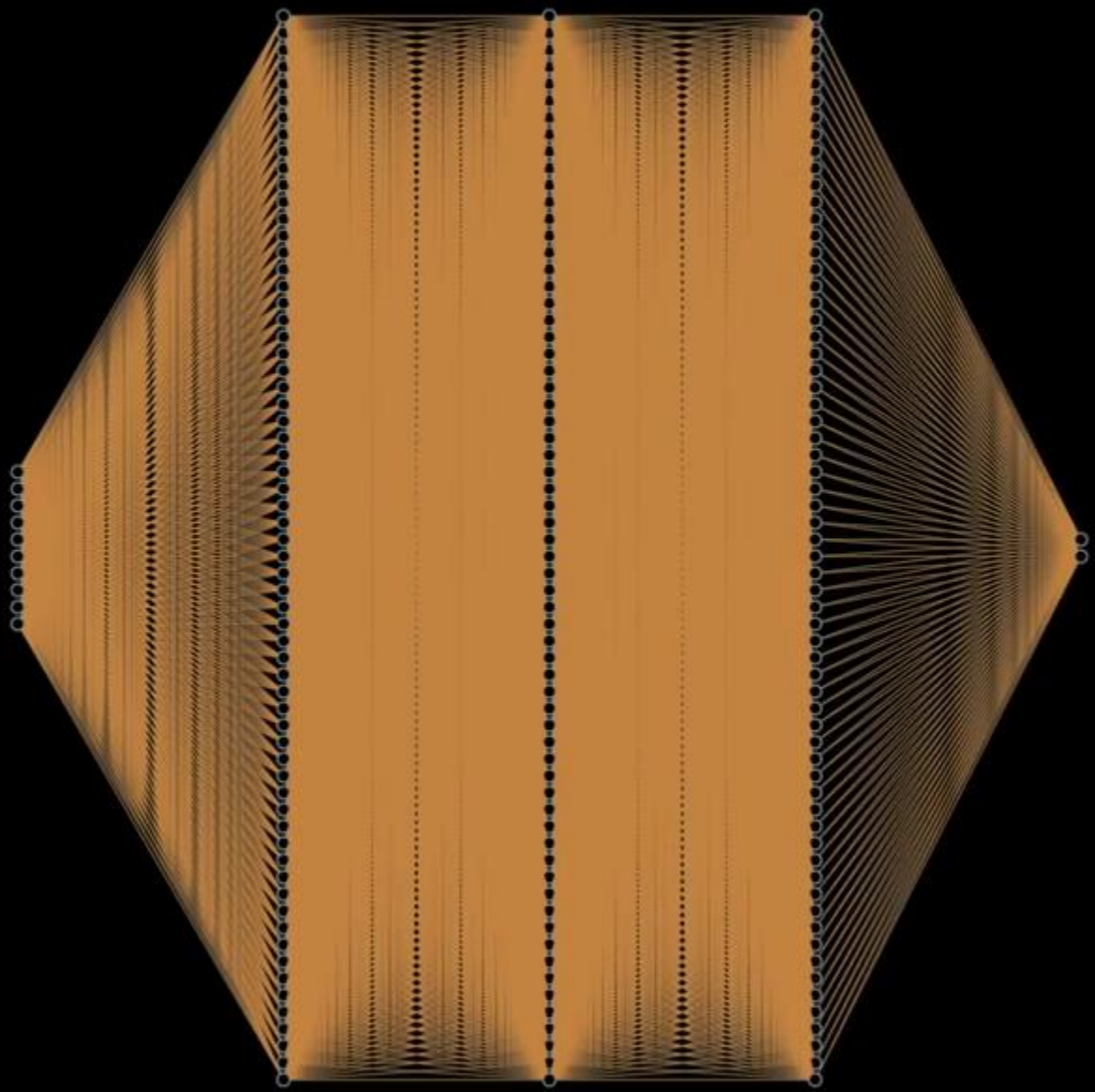
Weights:	2432
Biases:	108
Params:	2540



https://nnfs.io

Layer sizes: 10, 64, 64, 64, 2

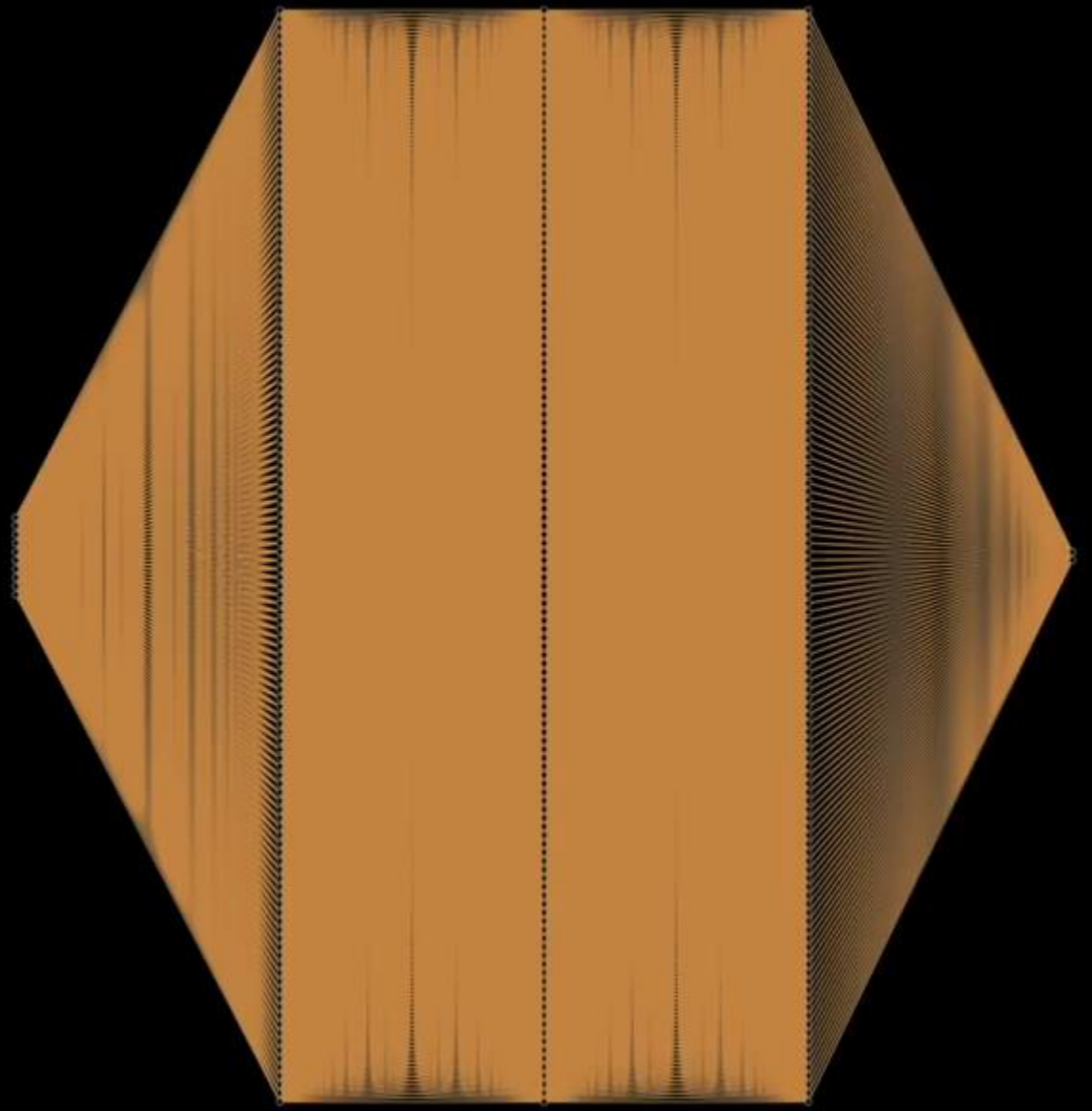
Weights:	8960
Biases:	204
Params:	9164



https://nnfs.io

Layer sizes: 10, 128, 128, 128, 2

Weights:	34304
Biases:	396
Params:	34700



pl.py

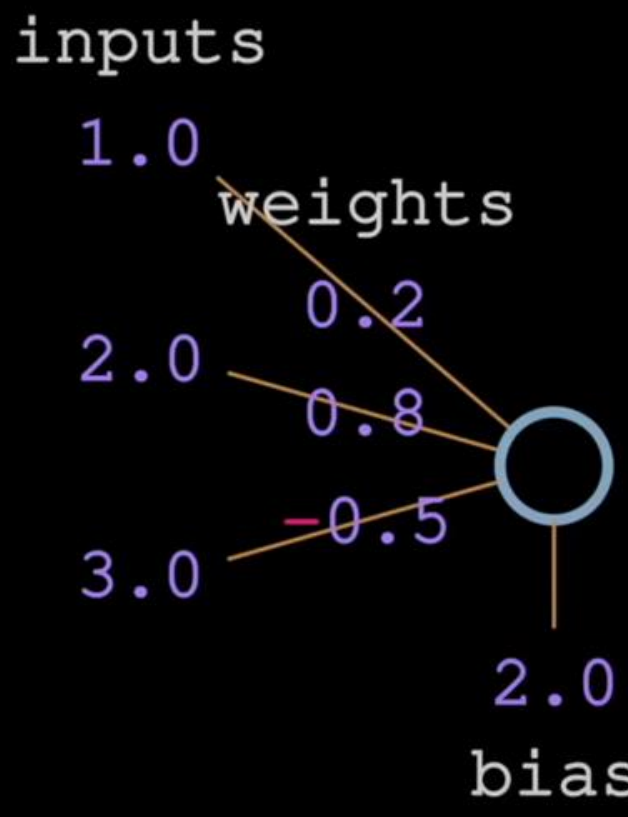
```python
import sys
import numpy as np
import matplotlib


print("Python:", sys.version)
print("Numpy:", np.__version__)
print("Matplotlib: ", matplotlib.__version__)
```

```
Python: 3.7.7 (default, Mar 10 2020, 15:16:38)
[GCC 7.5.0]
Numpy: 1.18.2
Matplotlib:  3.2.1
[Finished in 0.3s]
```

pl.py

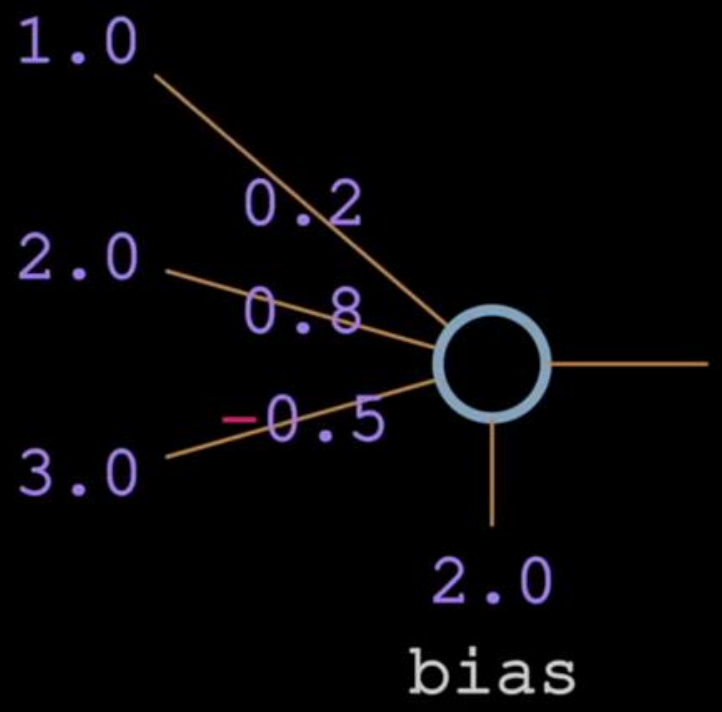
```
inputs = [1.2, 5.1, 2.1]
weights = [3.1, 2.1, 8.7]
bias = 3



output = inputs[0]*weights[0] + inputs[1]*weights[1] + inputs[2]*weights[2] + bias
print(output)
```

```
35.7
[Finished in 0.0s]
```



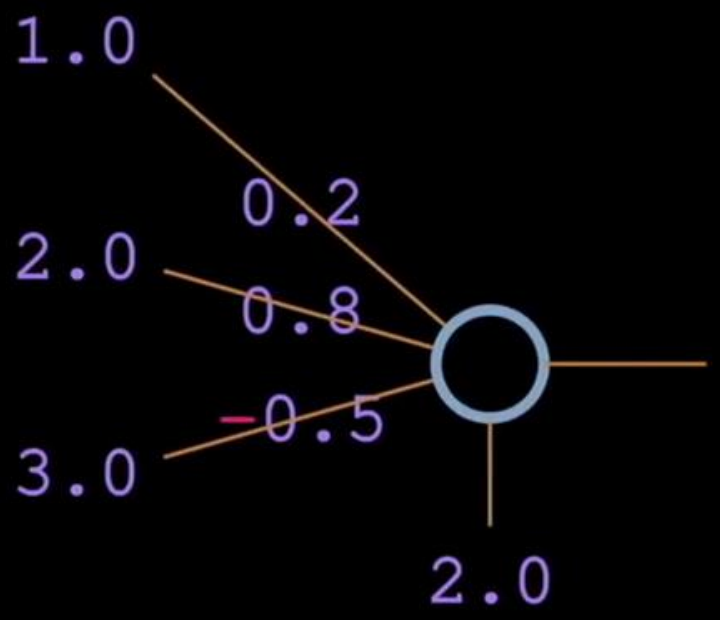
https://nnfs.io

```
inputs = [1.0, 2.0, 3.0]
weights = [0.2, 0.8, -0.5]
```



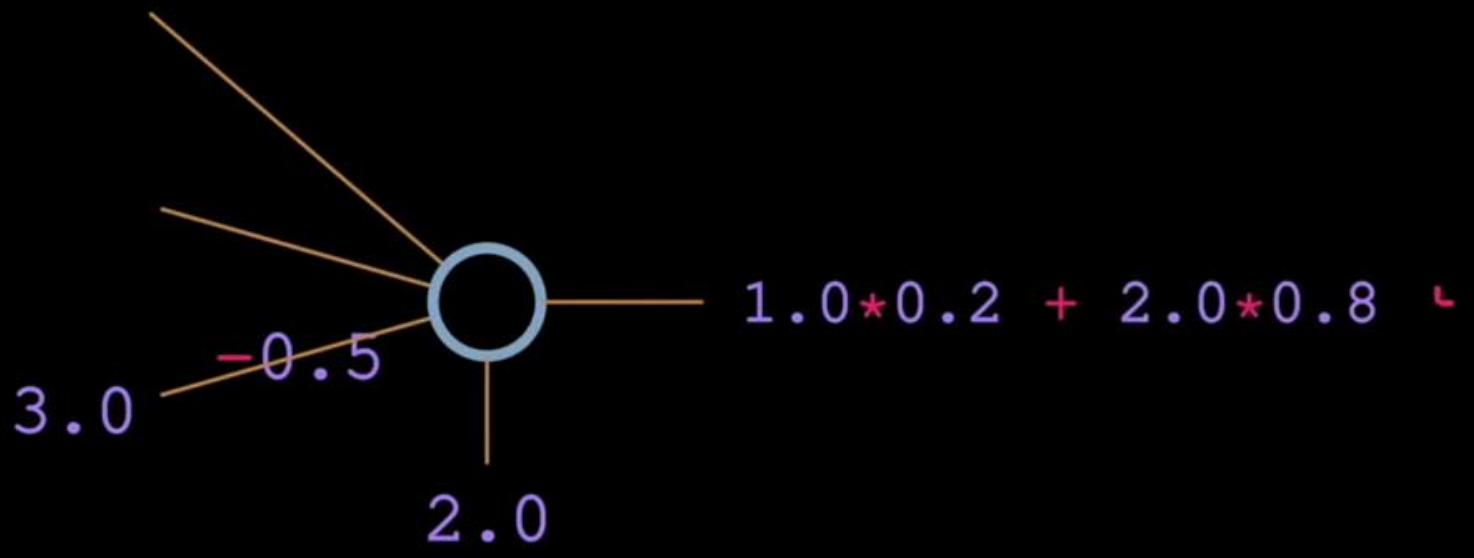
https://nnfs.io

```
inputs = [1.0, 2.0, 3.0]
weights = [0.2, 0.8, -0.5]
bias = 2.0
```



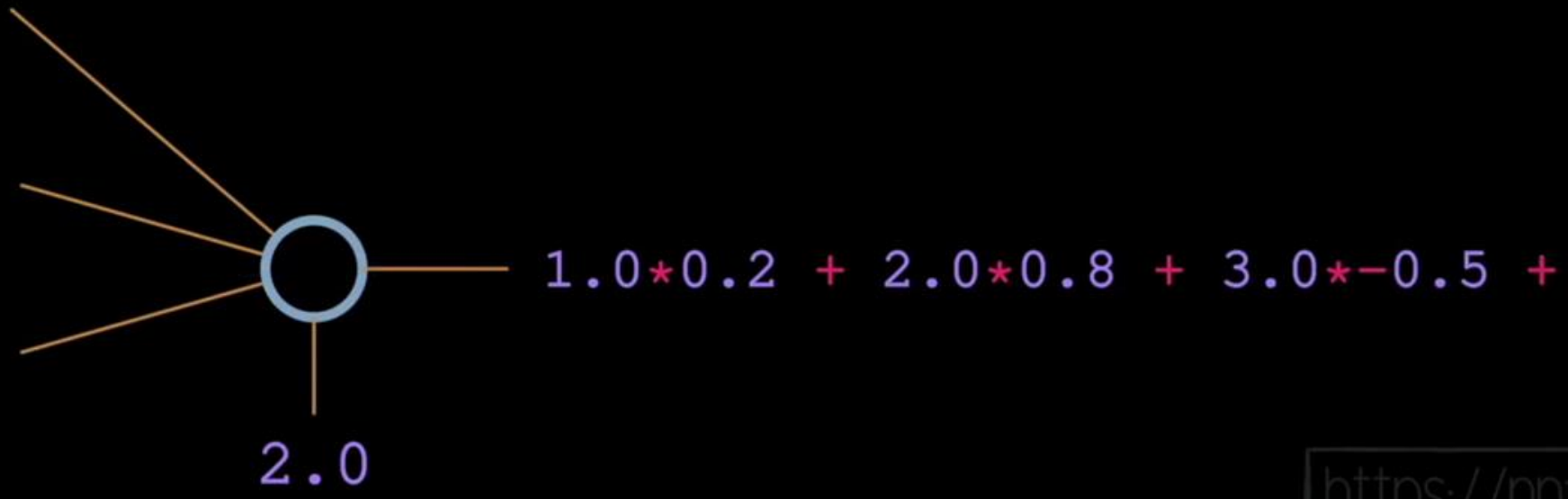
https://nnfs.io

```
inputs = [1.0, 2.0, 3.0]
weights = [0.2, 0.8, -0.5]
bias = 2.0

output = inputs[0]*weights[0] + inputs[1]*weights[1] +
```



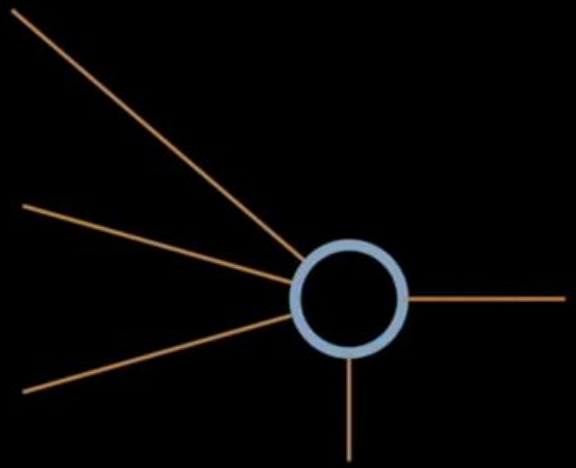
https://nnfs.io

```
inputs = [1.0, 2.0, 3.0]
weights = [0.2, 0.8, -0.5]
bias = 2.0

output = inputs[0]*weights[0] + inputs[1]*weights[1] + inputs[2]*weights[2] +
```



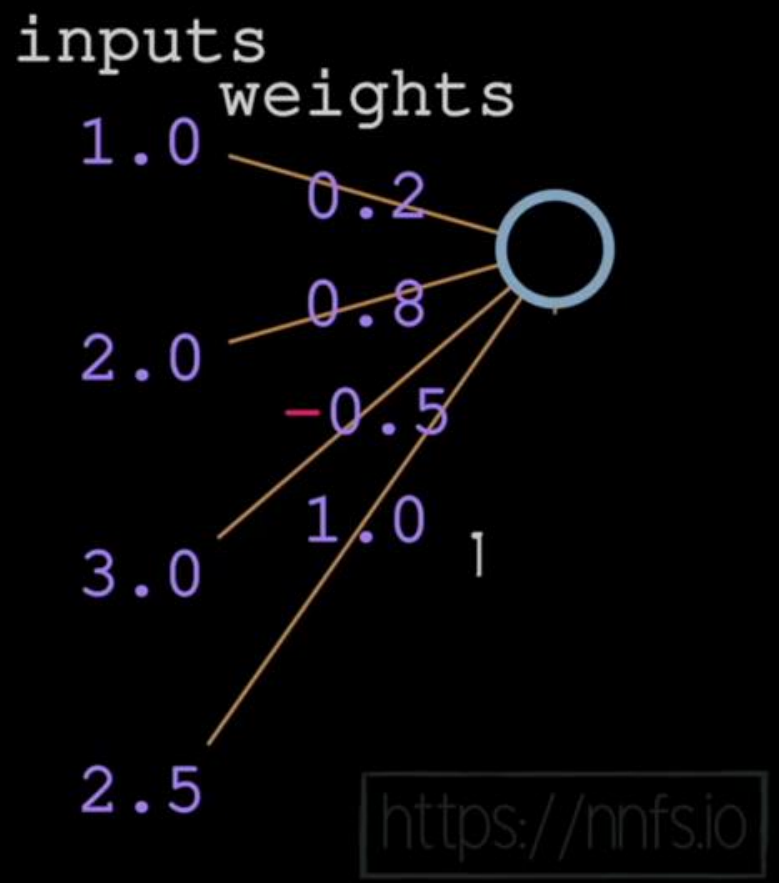
https://nnfs.io

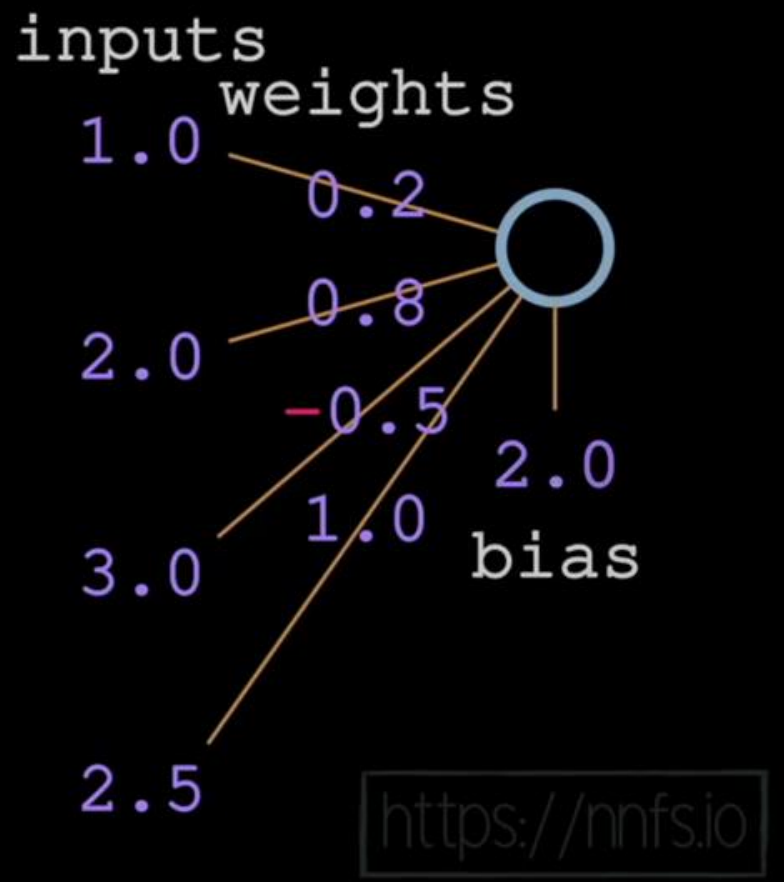
```
inputs = [1.0, 2.0, 3.0]
weights = [0.2, 0.8, -0.5]
bias = 2.0

output = inputs[0]*weights[0] + inputs[1]*weights[1] + inputs[2]*weights[2] + bias
print(output)

>>> 2.3
```



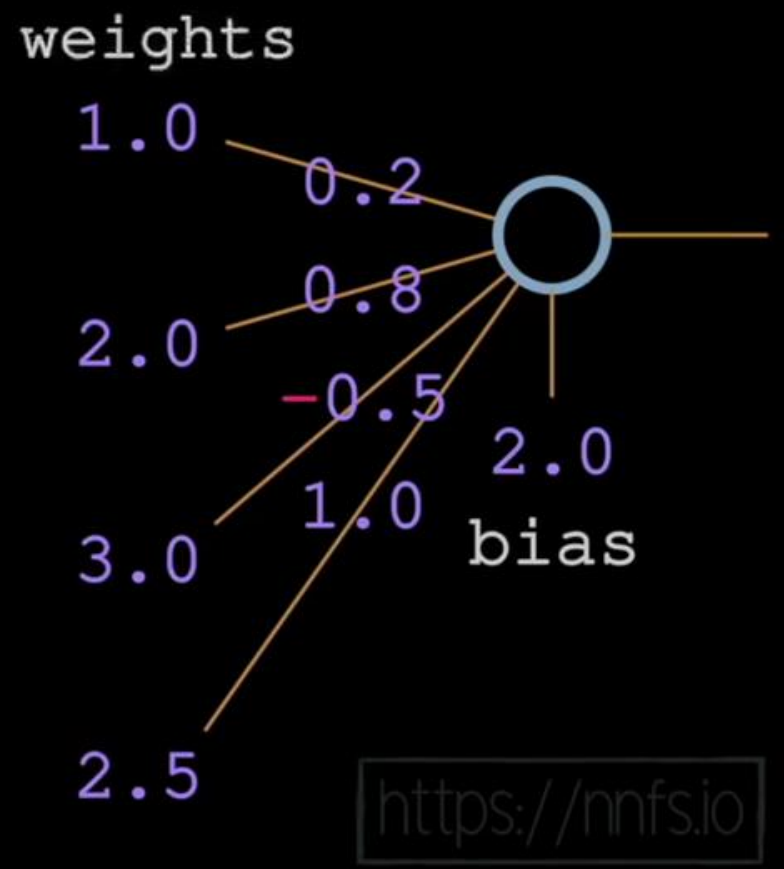
1.0*0.2 + 2.0*0.8 + 3.0*-0.5 + 2.0 = 2.3

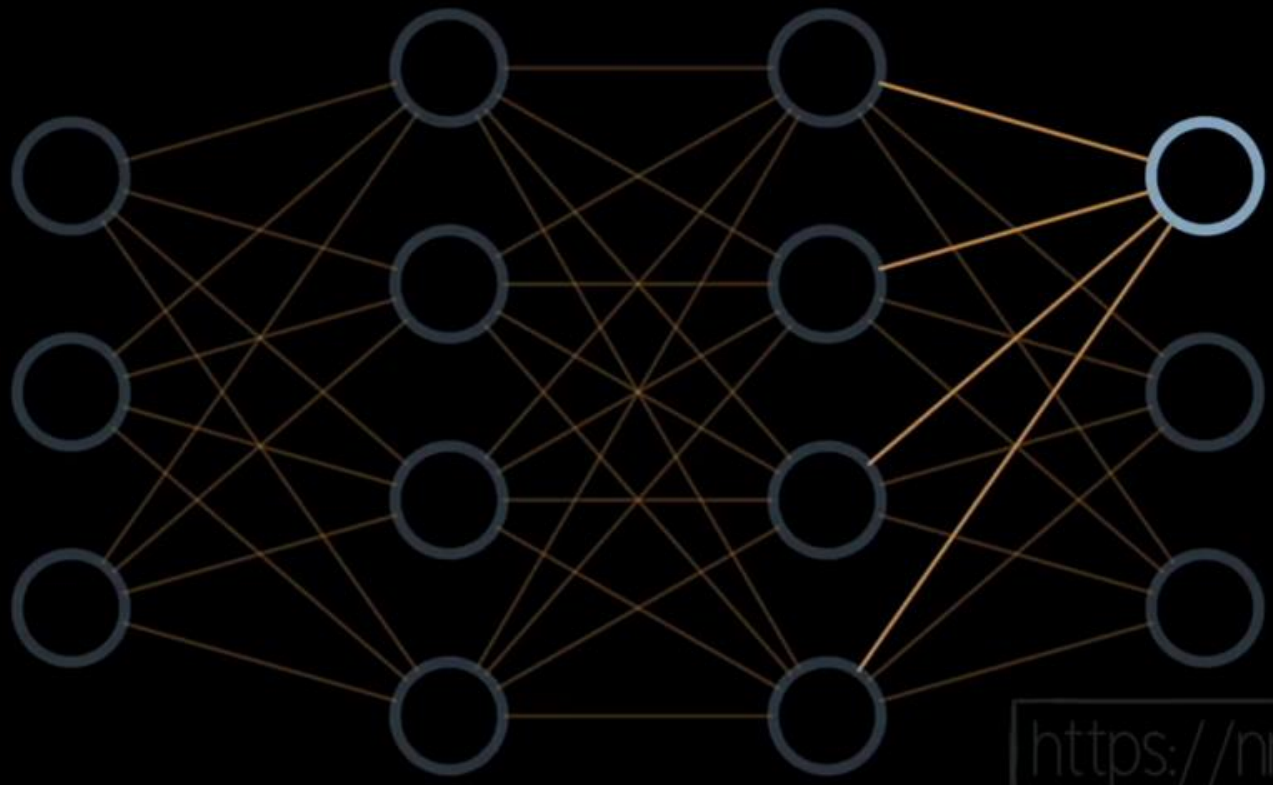
https://nnfs.io


inputs
weights
1.0
0.2
2.0
0.8
-0.5
3.0
1.0
1
2.5
https://nnfs.io

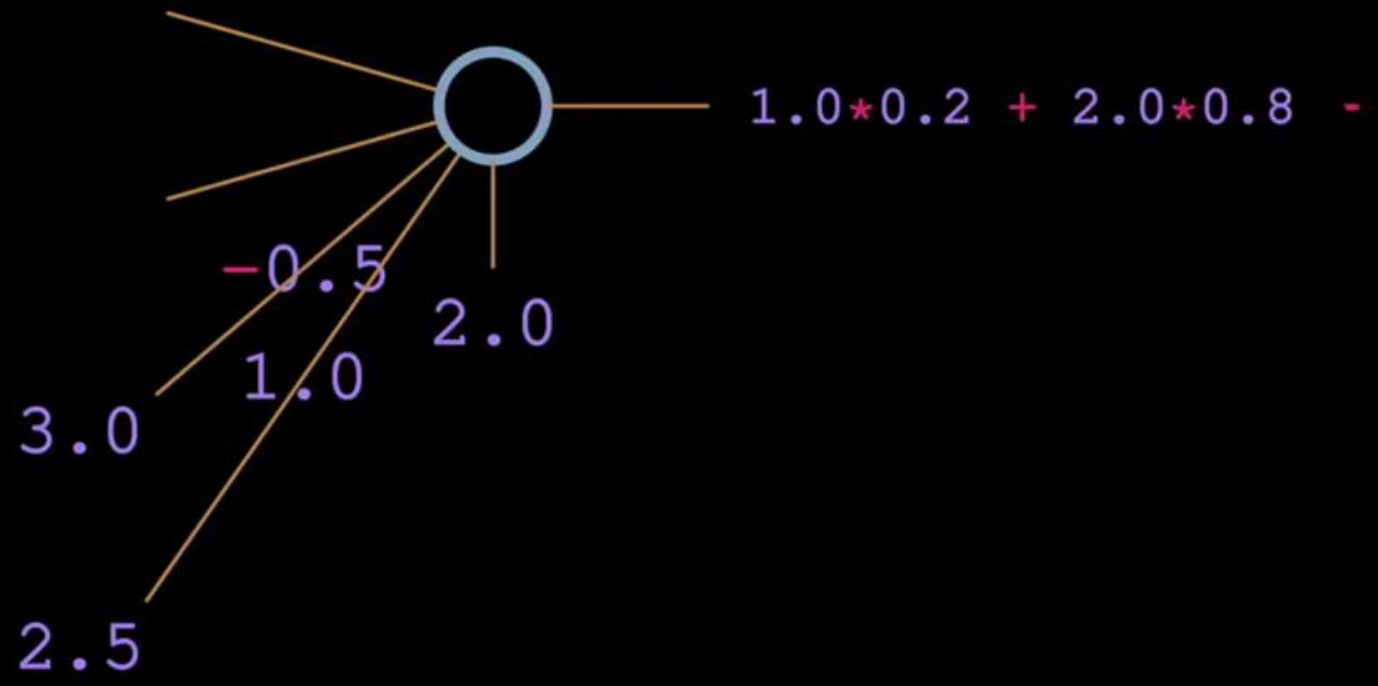

inputs
weights
1.0
0.2
2.0
0.8
−0.5
3.0
1.0
2.5
2.0
bias
https://nnfs.io

```
inputs = [1.0, 2.0, 3.0, 2.5]
```



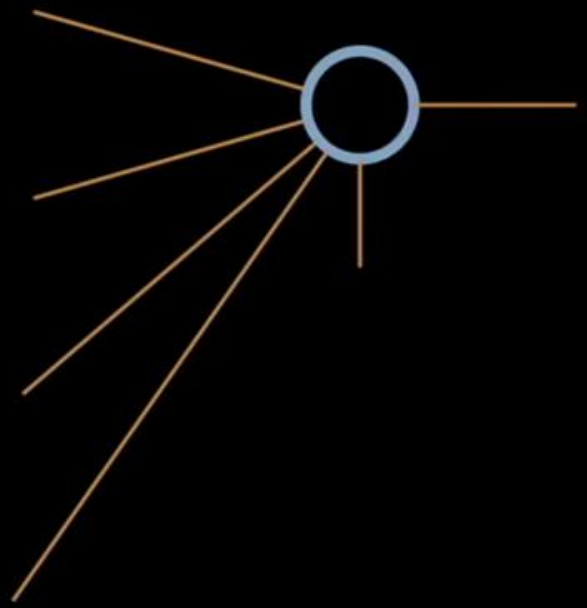

https://nnfs.io

```
inputs = [1.0, 2.0, 3.0, 2.5]
weights = [0.2, 0.8, -0.5, 1.0]
bias = 2.0

output = inputs[0]*weights[0] + inputs[1]*weights[1] -
```



https://nnfs.io

```
inputs = [1.0, 2.0, 3.0, 2.5]
weights = [0.2, 0.8, -0.5, 1.0]
bias = 2.0

output = inputs[0]*weights[0] + inputs[1]*weights[1] + inputs[2]*weights[2] + inputs[3]*weights[3] + bias
print(output)
```



1.0*0.2 + 2.0*0.8 + 3.0*-0.5 + 2.5*1.0 + 2.0 =

https://nnfs.io

https://nnfs.io

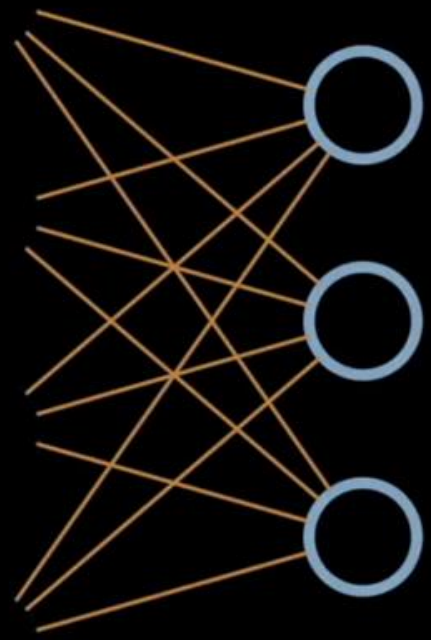
p2.py

```
inputs = [1, 2, 3, 2.5]

weights1 = [0.2, 0.8, -0.5, 1.0]
weights2 = [0.5, -0.91, 0.26, -0.5]
weights3 = [-0.26, -0.27, 0.17, 0.87]

bias1 = 2
bias2 = 3
bias3 = 0.5

output = [inputs[0]*weights1[0] + inputs[1]*weights1[1] + inputs[2]*weights1[2] + inputs[3]*weights1[3] + bias1,
          inputs[0]*weights2[0] + inputs[1]*weights2[1] + inputs[2]*weights2[2] + inputs[3]*weights2[3] + bias2,
          inputs[0]*weights3[0] + inputs[1]*weights3[1] + inputs[2]*weights3[2] + inputs[3]*weights3[3] + bias3]
print(output)
```

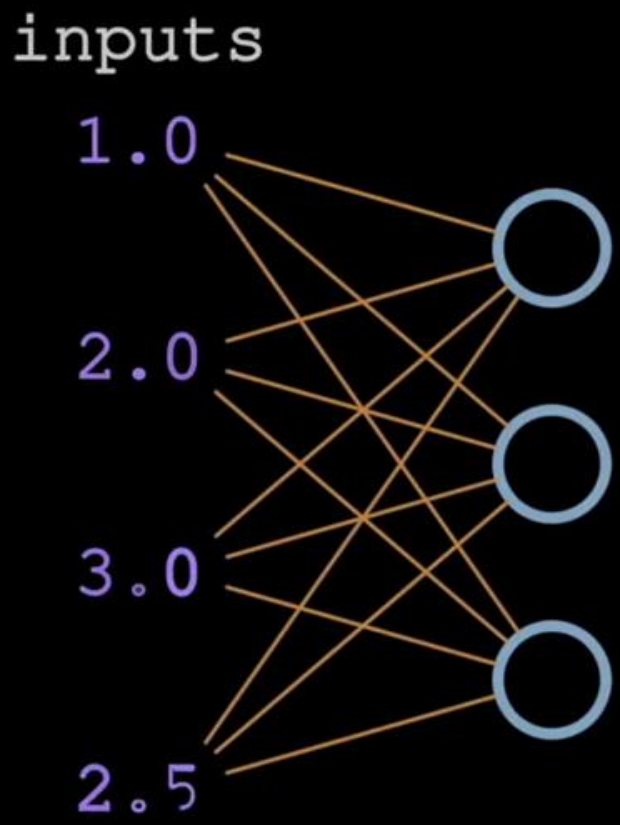
```
[4.8, 1.21, 2.385]
[Finished in 0.0s]
```

https://nnfs.io

ir

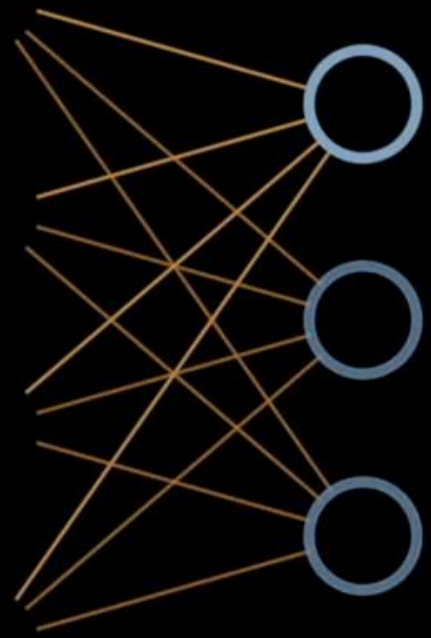


https://nnfs.io



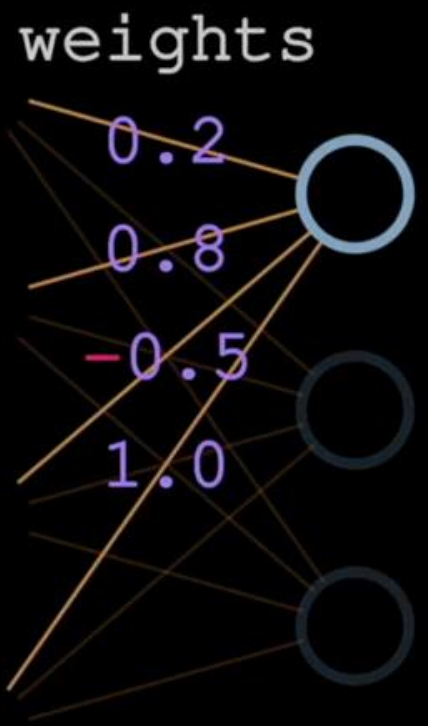
```
inputs = [1.0, 2.0, 3.0, 2.5]
```

https://nnfs.io



```
inputs = [1.0, 2.0, 3.0, 2.5]
```

https://nnfs.io

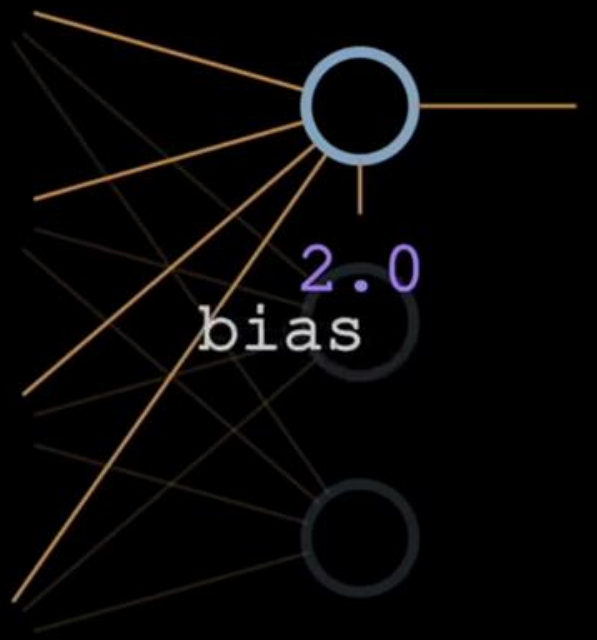


```
inputs = [1.0, 2.0, 3.0, 2.5]
```

https://nnfs.io

```
inputs = [1.0, 2.0, 3.0, 2.5]
weights1 = [0.2, 0.8, -0.5, 1.0]
```

https://nnfs.io

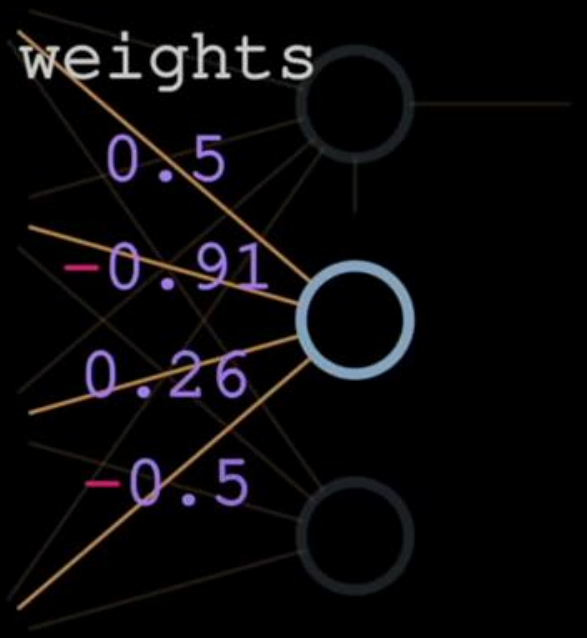


```
inputs = [1.0, 2.0, 3.0, 2.5]
weights1 = [0.2, 0.8, -0.5, 1.0]


bias1 = 2.0
```

https://nnfs.io

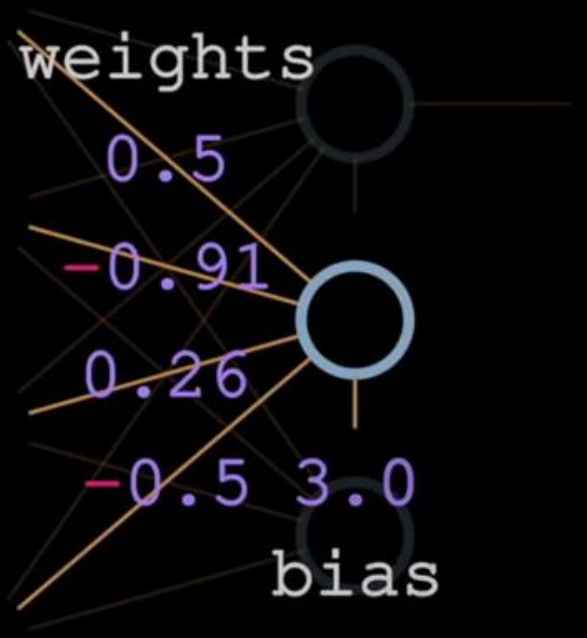
```
inputs = [1.0, 2.0, 3.0, 2.5]
weights1 = [0.2, 0.8, -0.5, 1.0]

bias1 = 2.0
```

https://nnfs.io



```
inputs = [1.0, 2.0, 3.0, 2.5]
weights1 = [0.2, 0.8, -0.5, 1.0]

bias1 = 2.0
```

https://nnfs.io

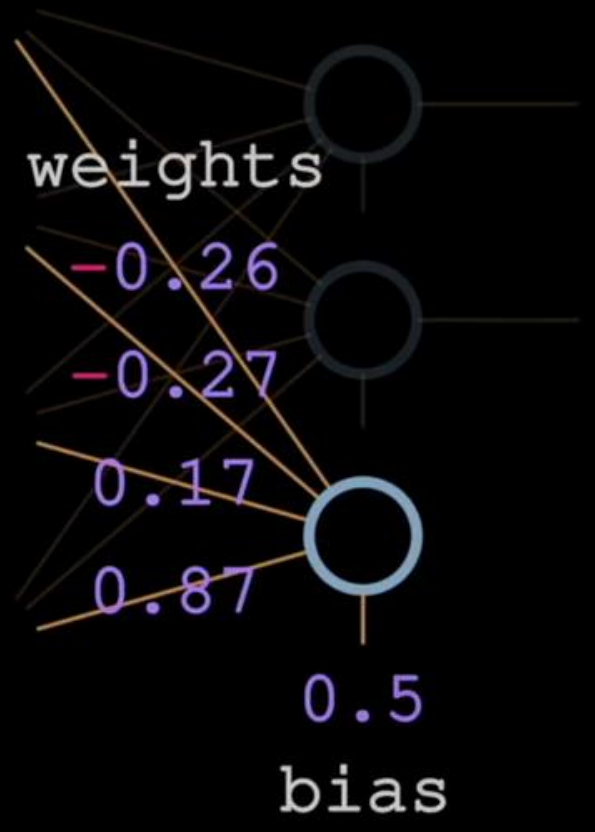


```
inputs = [1.0, 2.0, 3.0, 2.5]
weights1 = [0.2, 0.8, -0.5, 1.0]
weights2 = [0.5, -0.91, 0.26, -0.5]

bias1 = 2.0
bias2 = 3.0
```

https://nnfs.io

```
inputs = [1.0, 2.0, 3.0, 2.5]
weights1 = [0.2, 0.8, -0.5, 1.0]
weights2 = [0.5, -0.91, 0.26, -0.5]

bias1 = 2.0
bias2 = 3.0
```

https://nnfs.io



```
inputs = [1.0, 2.0, 3.0, 2.5]
weights1 = [0.2, 0.8, -0.5, 1.0]
weights2 = [0.5, -0.91, 0.26, -0.5]
weights3 = [-0.26, -0.27, 0.17, 0.87]
bias1 = 2.0
bias2 = 3.0
bias3 = 0.5
```

https://nnfs.io

```
inputs = [1.0, 2.0, 3.0, 2.5]
weights1 = [0.2, 0.8, -0.5, 1.0]
weights2 = [0.5, -0.91, 0.26, -0.5]
weights3 = [-0.26, -0.27, 0.17, 0.87]
bias1 = 2.0
bias2 = 3.0
bias3 = 0.5

outputs = [
    inputs[0]*weights1[0] + inputs[1]*weights1[1] + inputs[2]*weights1[2] + inputs[3]*weights1[3] + bias1
```

https://nnfs.io

```
1.0*0.2 + 2.0*0.8 + 3.0*-0.5 + 2.5*1.0 + 2.0
```

```
inputs = [1.0, 2.0, 3.0, 2.5]
weights1 = [0.2, 0.8, -0.5, 1.0]
weights2 = [0.5, -0.91, 0.26, -0.5]
weights3 = [-0.26, -0.27, 0.17, 0.87]
bias1 = 2.0
bias2 = 3.0
bias3 = 0.5

outputs = [
    inputs[0]*weights1[0] + inputs[1]*weights1[1] + inputs[2]*weights1[2] + inputs[3]*weights1[3] + bias1,
    inputs[0]*weights2[0] + inputs[1]*weights2[1] + inputs[2]*weights2[2] + inputs[3]*weights2[3] + bias2
```

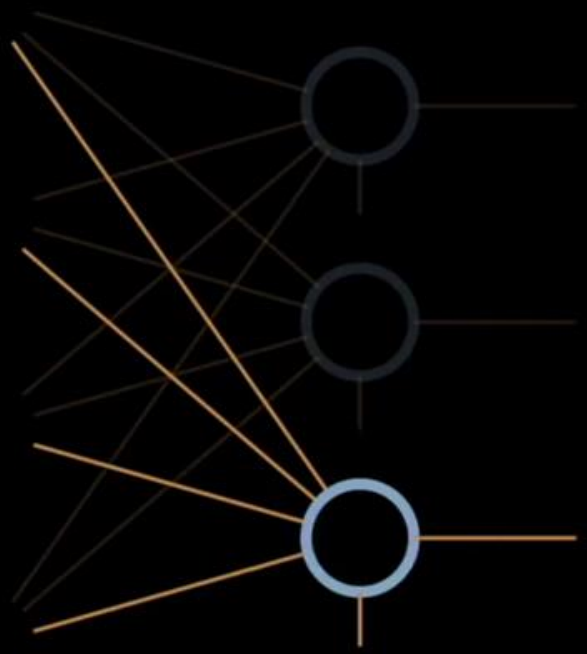
https://nnfs.io

```
1.0*0.2 + 2.0*0.8 + 3.0*-0.5 + 2.5*1.0 + 2.0
```

```
1.0*0.5 + 2.0*-0.91 + 3.0*0.26 + 2.5*-0.5 +3.0
```

```
inputs = [1.0, 2.0, 3.0, 2.5]
weights1 = [0.2, 0.8, -0.5, 1.0]
weights2 = [0.5, -0.91, 0.26, -0.5]
weights3 = [-0.26, -0.27, 0.17, 0.87]
bias1 = 2.0
bias2 = 3.0
bias3 = 0.5

outputs = [
    inputs[0]*weights1[0] + inputs[1]*weights1[1] + inputs[2]*weights1[2] + inputs[3]*weights1[3] + bias1,
    inputs[0]*weights2[0] + inputs[1]*weights2[1] + inputs[2]*weights2[2] + inputs[3]*weights2[3] + bias2,
    inputs[0]*weights3[0] + inputs[1]*weights3[1] + inputs[2]*weights3[2] + inputs[3]*weights3[3] + bias3
```

https://nnfs.io



```
1.0*0.2 + 2.0*0.8 + 3.0*-0.5 + 2.5*1.0 + 2.0
```

```
1.0*0.5 + 2.0*-0.91 + 3.0*0.26 + 2.5*-0.5 + 3.0
```

```
1.0*-0.26 + 2.0*-0.27 + 3.0*0.17 + 2.5*0.87 + 0.5
```

p2.py

```
inputs = [1, 2, 3
weights1 = [0.2,
weights2 = [0.5,
weights3 = [-0.26
bias1 = 2
bias2 = 3
bias3 = 0.5
output = [inputs[
          inputs[
          inputs[
print(output)
```

```
[4.8, 1.21, 2.385]
[Finished in 0.0s]
```

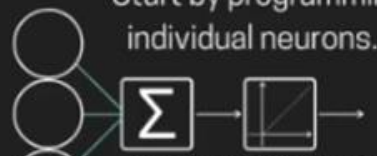
Neural Networks from Scratch

nnfs.io

Neural Networks from Scratch

PREORDER

Start by programming individual neurons...



...and their activation functions

- *Rectified Linear (ReLU)*
- *Sigmoid*
- *Linear*
- *Softmax*

Code and perform gradient computations using backpropagation and parameter updates using optimizers:

- *Stochastic Gradient Descent (SGD)*
- *AdaGrad*
- *RMSprop*
- *Adam*

Neural Networks from Scratch in Python

Building neural networks in raw Python

Learn how loss works and how to calculate loss with *Cross-Entropy*

Build & train neural networks from scratch in raw Python

Classification & Regression

```
bias1,
bias2,
bias3]
```

p3.py

```python
inputs = [1, 2, 3, 2.5]


weights = [[0.2, 0.8, -0.5, 1.0],
           [0.5, -0.91, 0.26, -0.5],
           [-0.26, -0.27, 0.17, 0.87]]

biases = [2, 3, 0.5]

bias1 = 2
bias2 = 3
bias3 = 0.5



layer_outputs = []  # Output of current layer
for neuron_weights, neuron_bias in zip(weights, biases):
    neuron_output = 0  # Output of given neuron
    for n_input, weight in zip(inputs, neuron_weights):
        neuron_output += n_input*weight
    neuron_output += neuron_bias
    layer_outputs.append(neuron_output)

print(layer_outputs)
```

Array :

```
l = [1,5,6,2]
```

Shape :

```
(4,)
```

Type :

1D array, Vector

https://nnfs.io

Array :

```
lol = [[1,5,6,2],
       [3,2,1,3]]
```

Shape :

```
(2, 4)
```

Type :

2D Array, Matrix

Array :

```
lolol = [[[1,5,6,2],
          [3,2,1,3]]
         [[5,2,1,2],
          [6,4,8,4]],
         [[2,8,5,3],
          [1,1,9,4]]]
```

Shape :

(3, 2, 4)

Type :

3D Array

```
a = [1, 2, 3]
b = [2, 3, 4]

dot_product = a[0]*b[0] + a[1]*b[1] + a[2]*b[2]
>>> 20
```

$$\vec{a} \cdot \vec{b} = [1, 2, 3] \cdot [2, 3, 4] = 1 \cdot 2 + 2 \cdot 3 + 3 \cdot 4 = 20$$

p3.py

```python
import numpy as np

inputs = [1, 2, 3, 2.5]
weights = [0.2, 0.8, -0.5, 1.0]
bias = 2


output = np.dot(inputs, weights) + bias
print(output)
```

```
4.8
[Finished in 0.2s]
```

Line 9, Column 13 Spaces: 4 Python

p3.py untitled

```python
import numpy as np

inputs = [1, 2, 3, 2.5]
weights = [0.2, 0.8, -0.5, 1.0]
bias = 2


output = np.dot(weights, inputs) + bias
print(output)
```

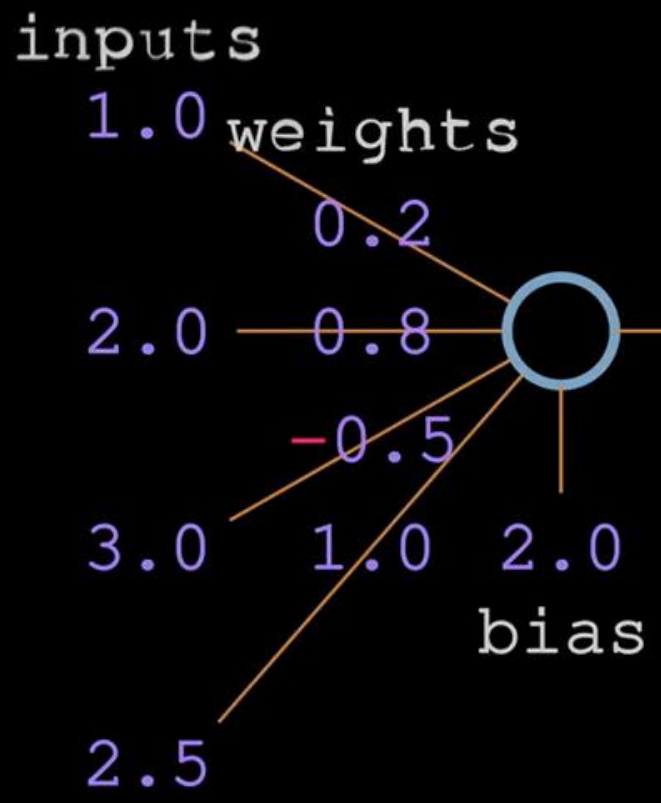
```
4.8
[Finished in 0.2s]
```

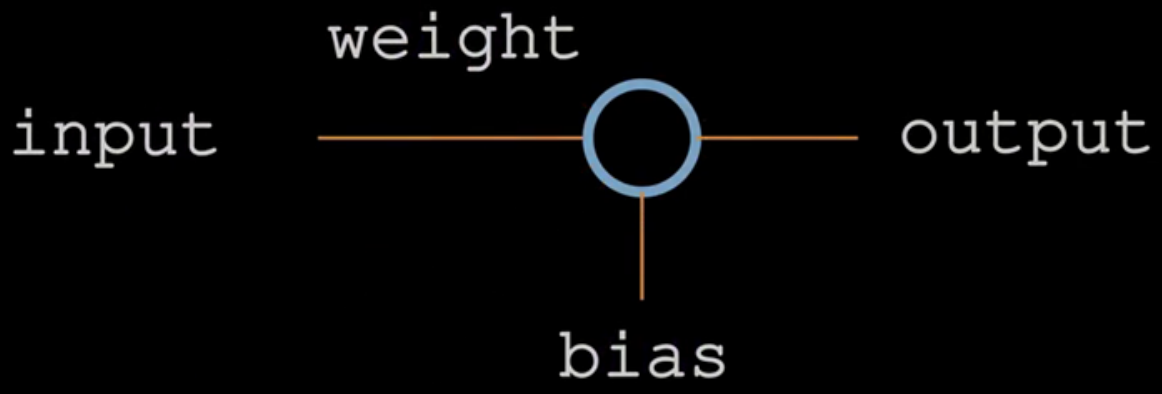
Line 9, Column 14; Build finished Spaces: 4 Python

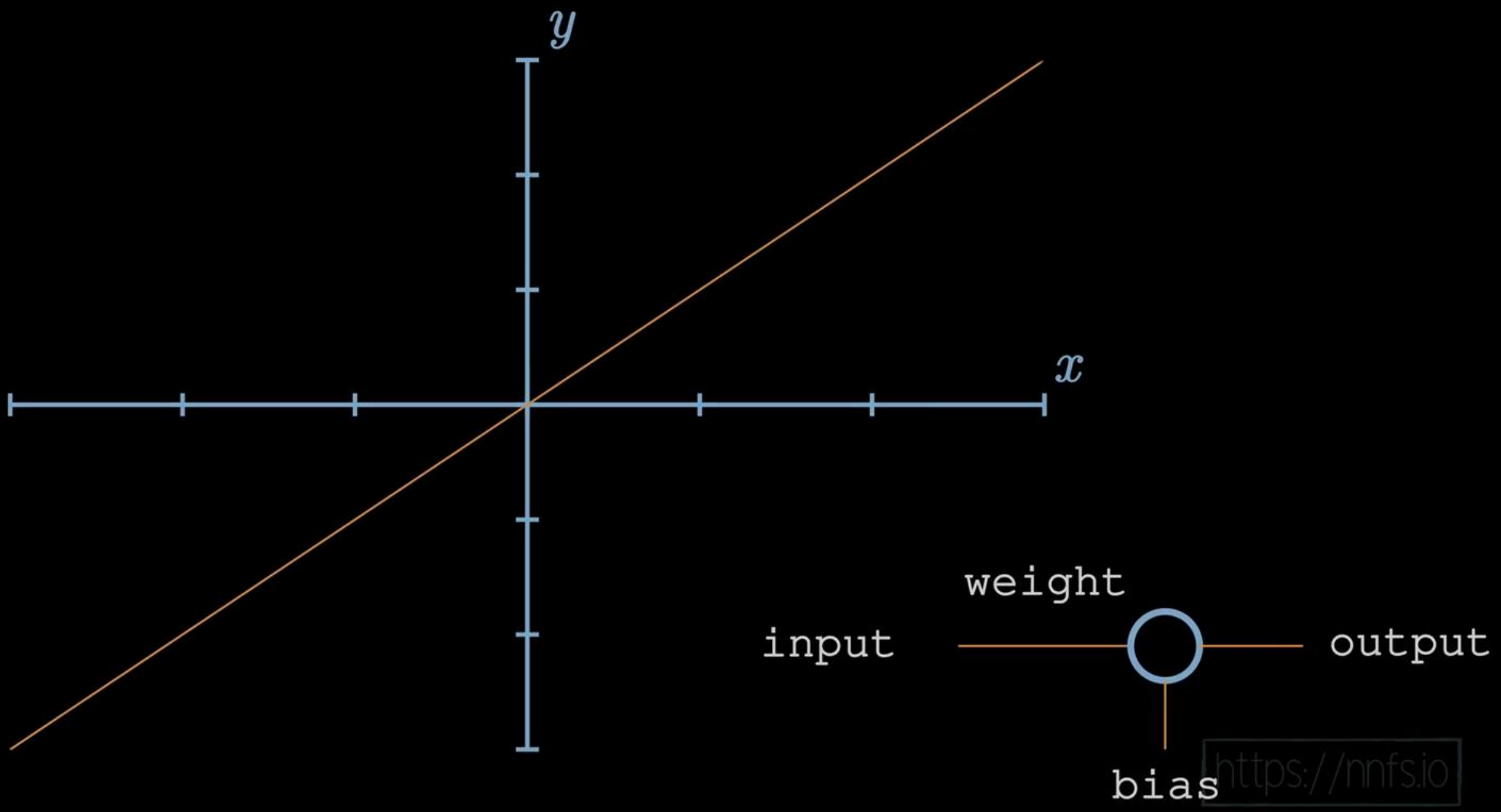
```
inputs = [1.0, 2.0, 3.0, 2.5]
weights = [[0.2, 0.8, -0.5, 1.0]
           [0.5, -0.91, 0.26, -0.5],
           [-0.26, -0.27, 0.17, 0.87]]
biases = [2.0, 3.0, 0.5]

outputs = np.dot(weights, inputs) + biases
```

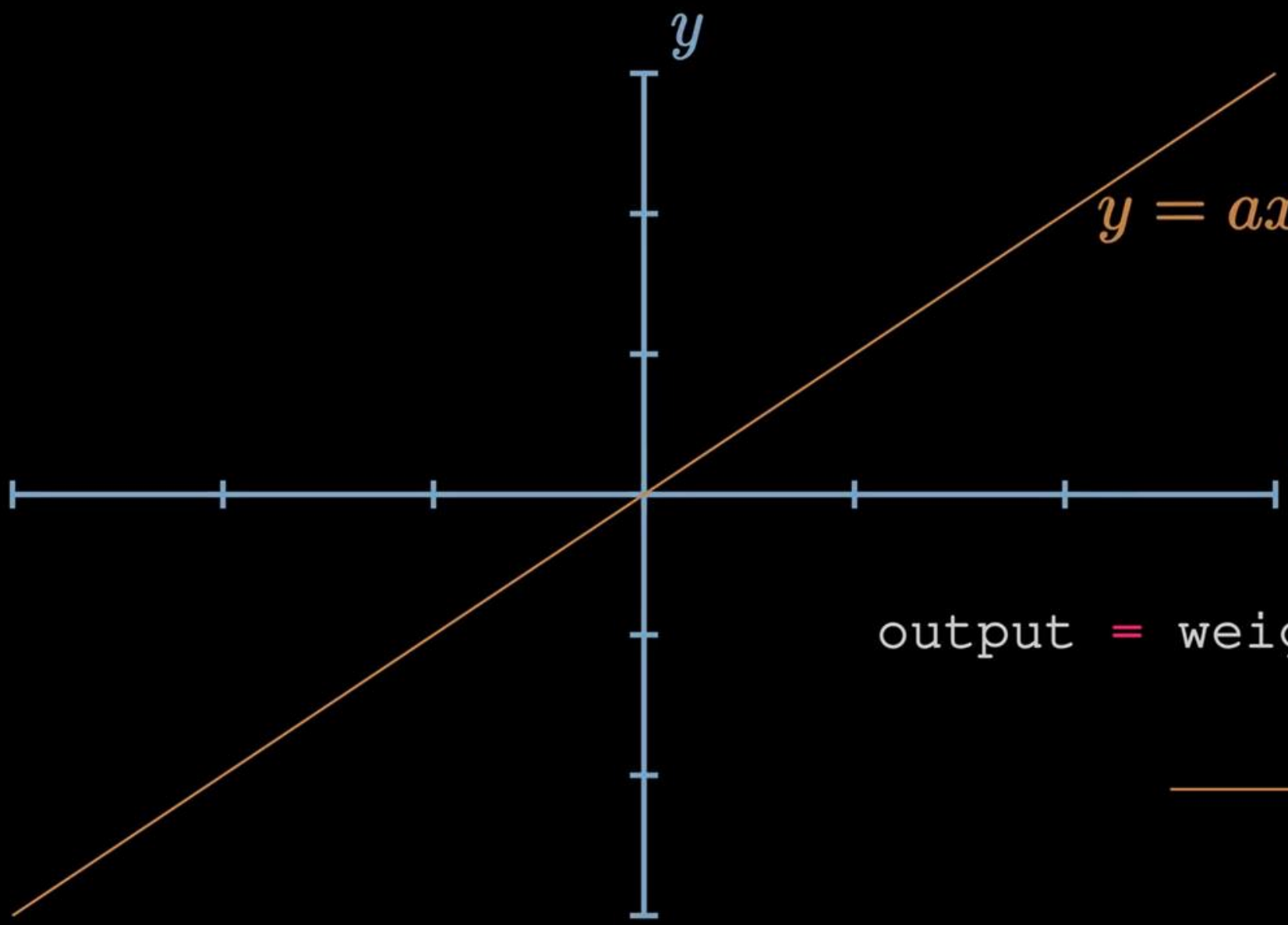
np.dot(weights, inputs)

```
inputs = [1.0, 2.0, 3.0, 2.5]
weights = [[0.2, 0.8, -0.5, 1.0]
           [0.5, -0.91, 0.26, -0.5],
           [-0.26, -0.27, 0.17, 0.87]]
biases = [2.0, 3.0, 0.5]

outputs = np.dot(weights, inputs) + biases
```



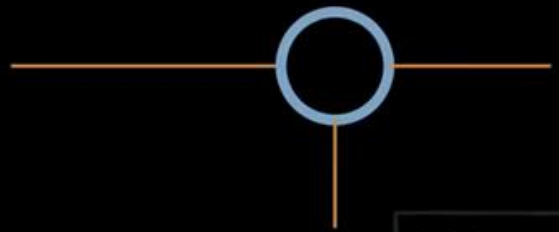



y
x
weight
input
output
bias
https://nnfs.io

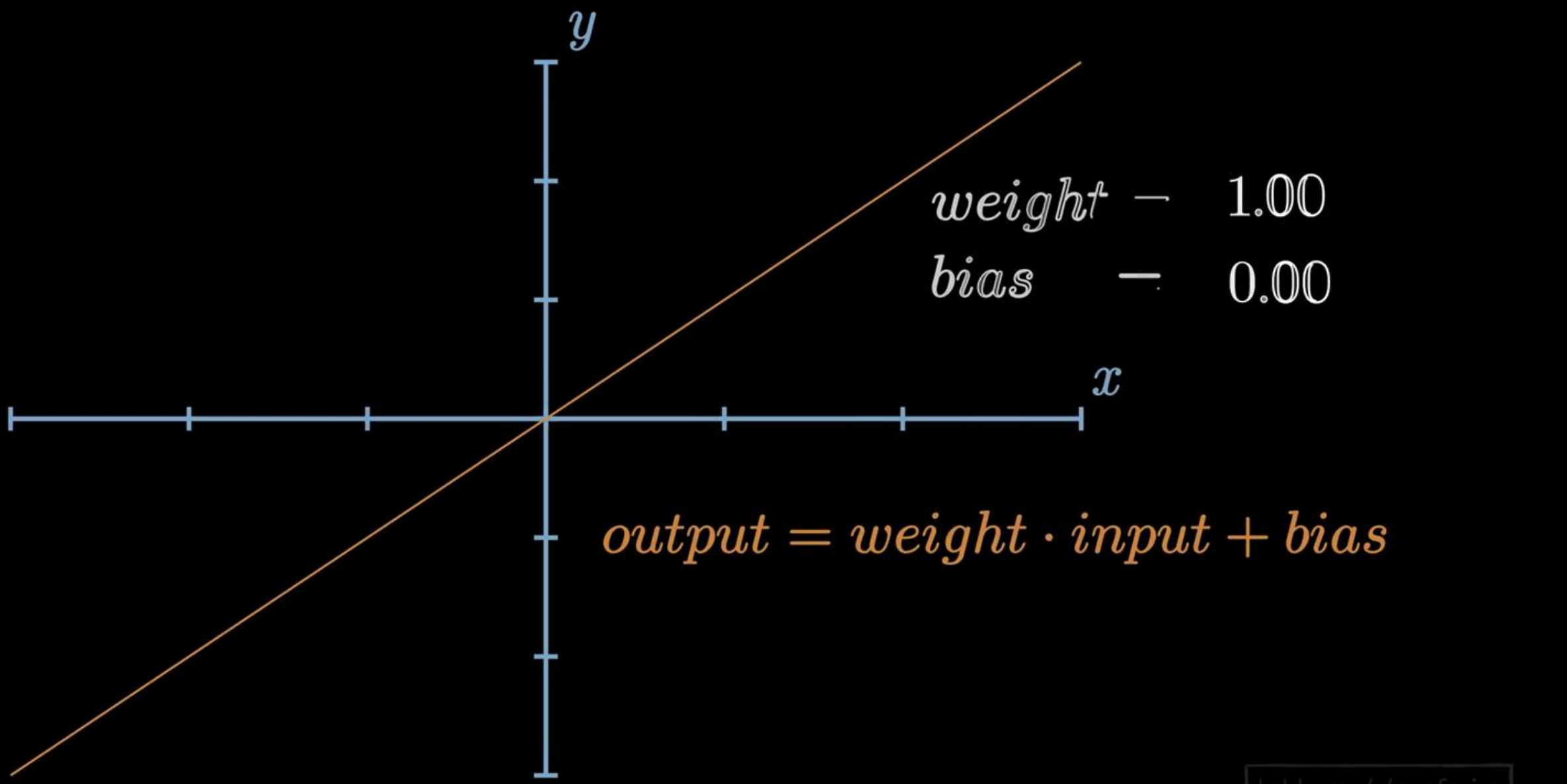


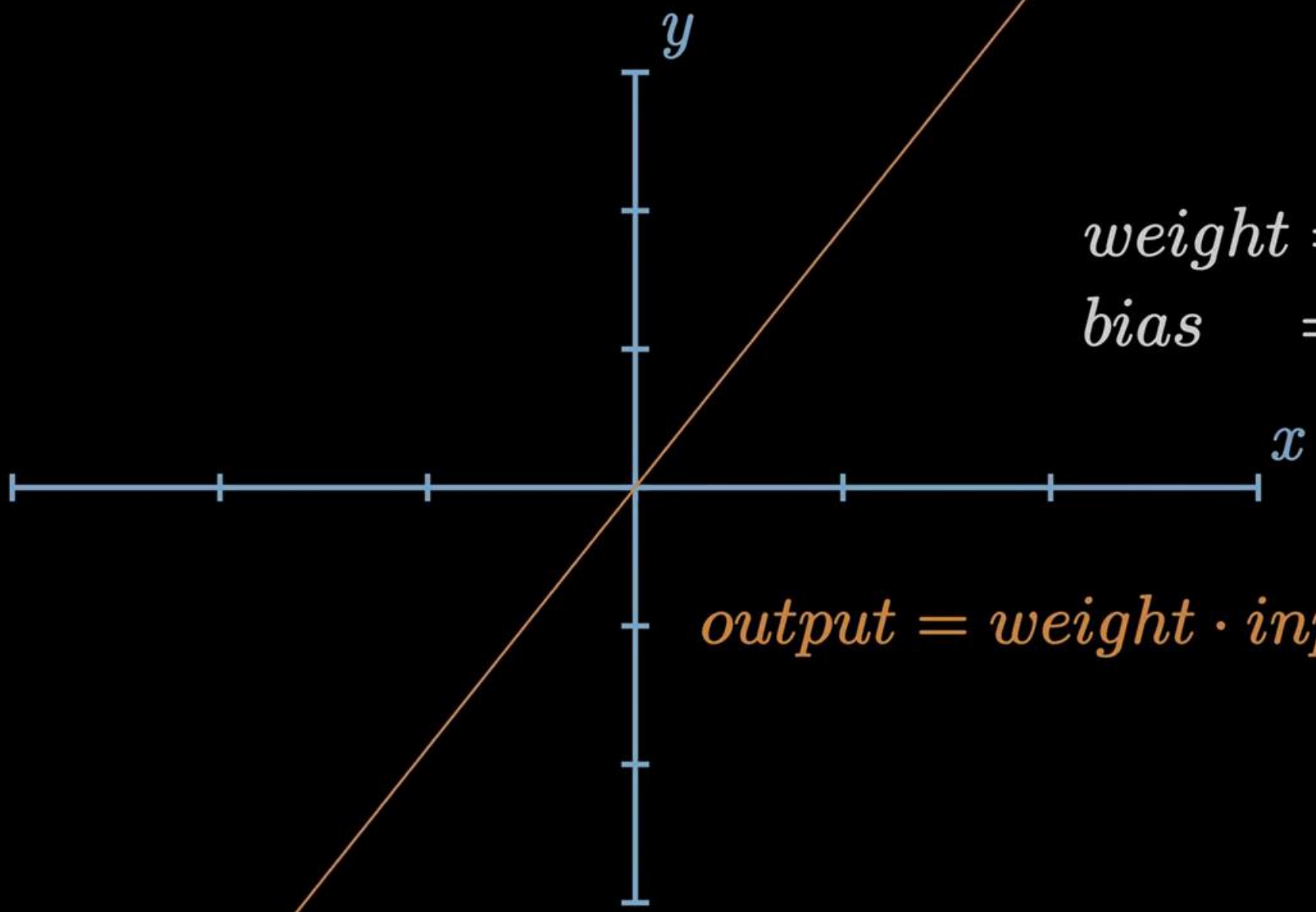
$$y = ax + m$$

```
output = weight*input + bias
```



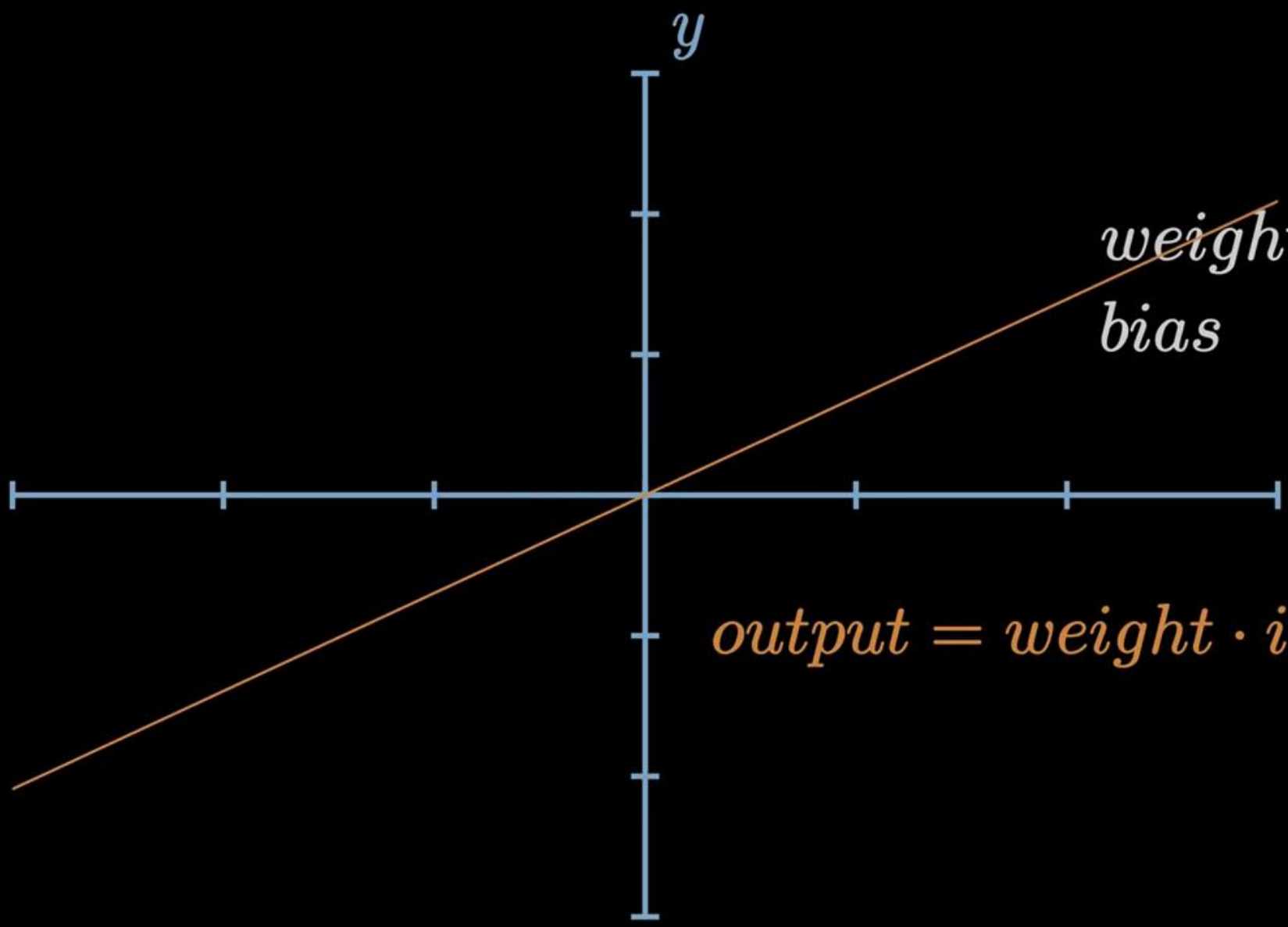
https://nnfs.io





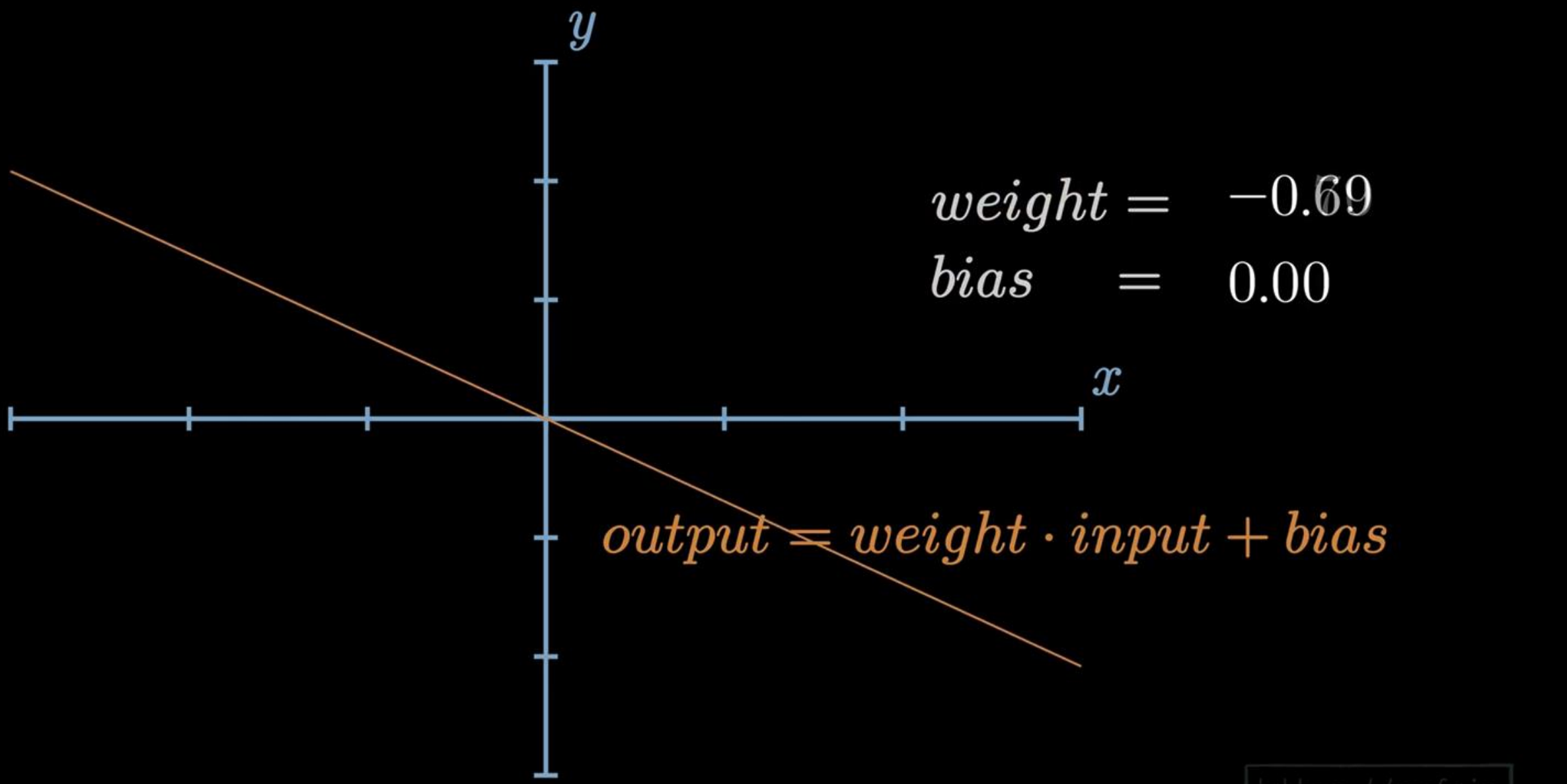
$$weight = 1.88$$
$$bias = 0.00$$

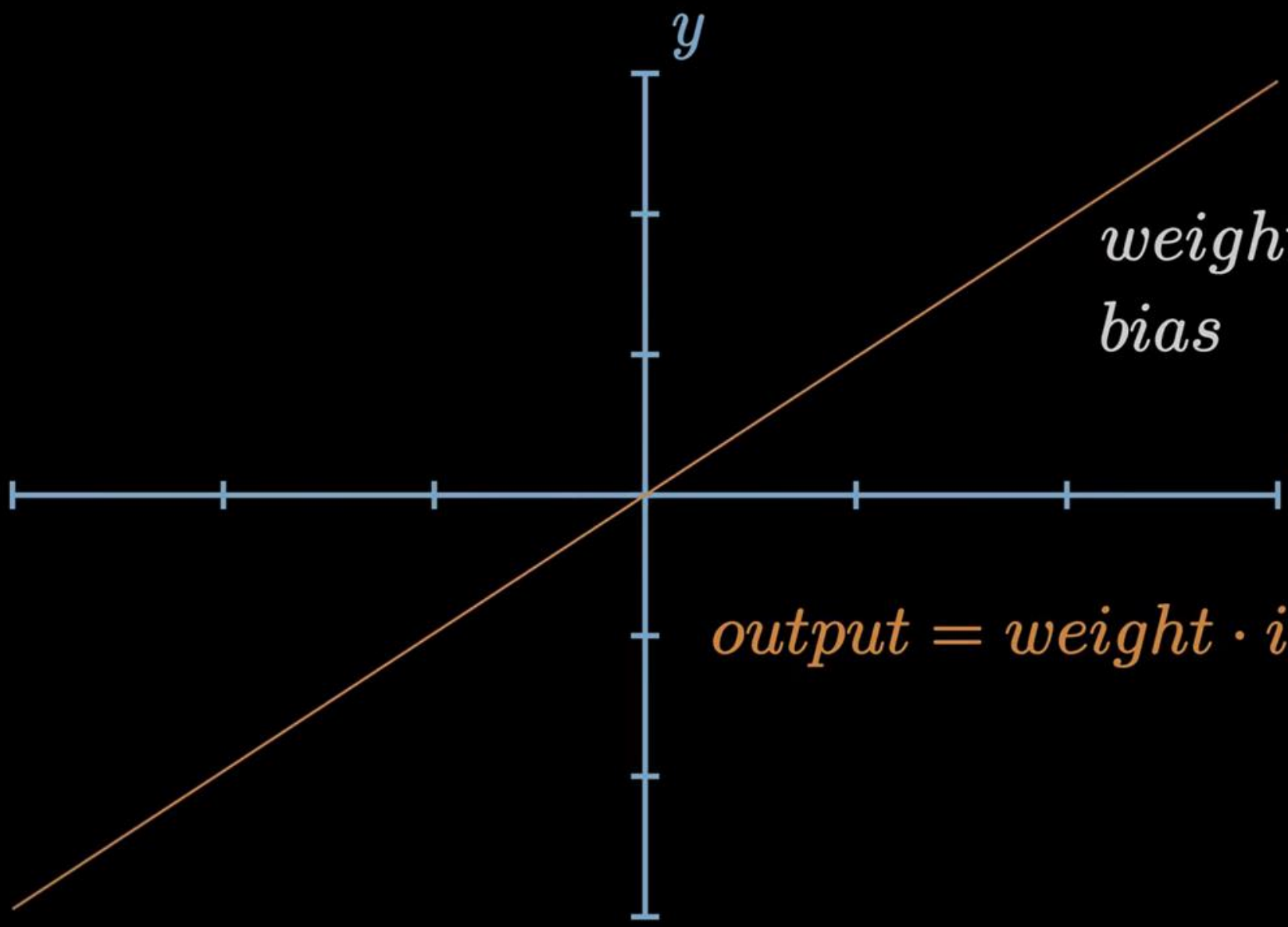
$$output = weight \cdot input + bias$$



$$weight = 0.70$$
$$bias = 0.00$$

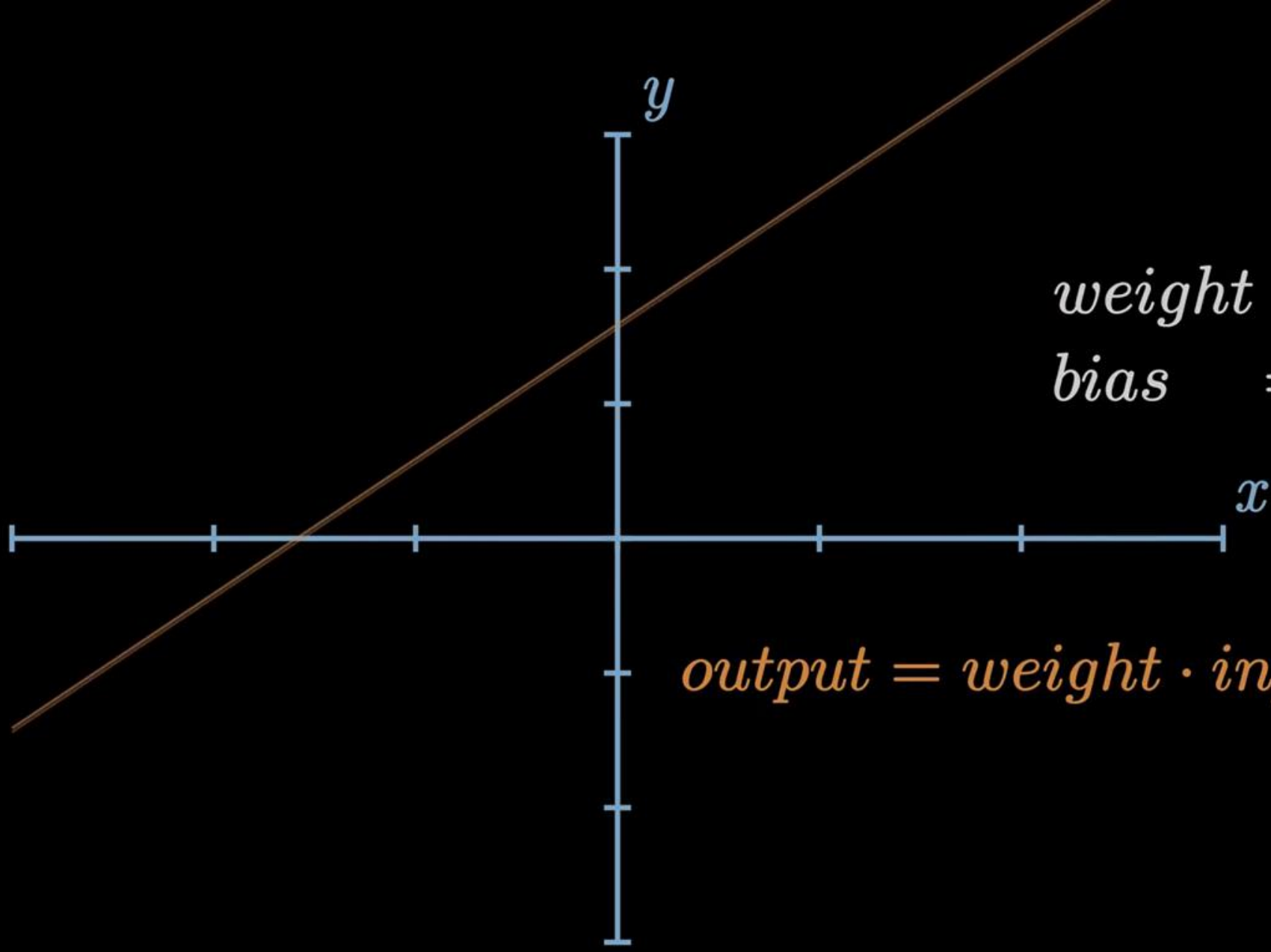
$$output = weight \cdot input + bias$$





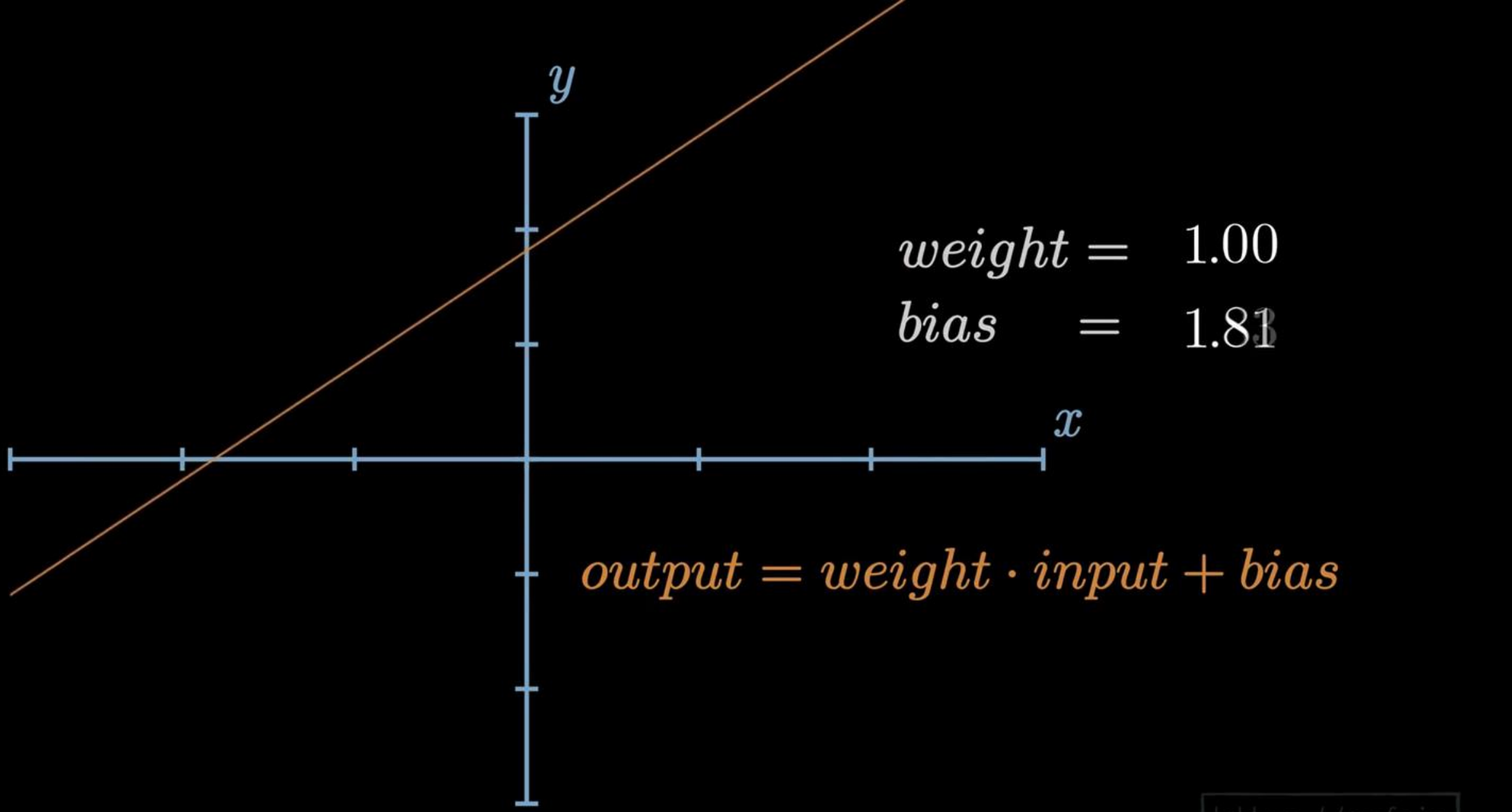
$$weight = 0.98$$
$$bias = 0.00$$

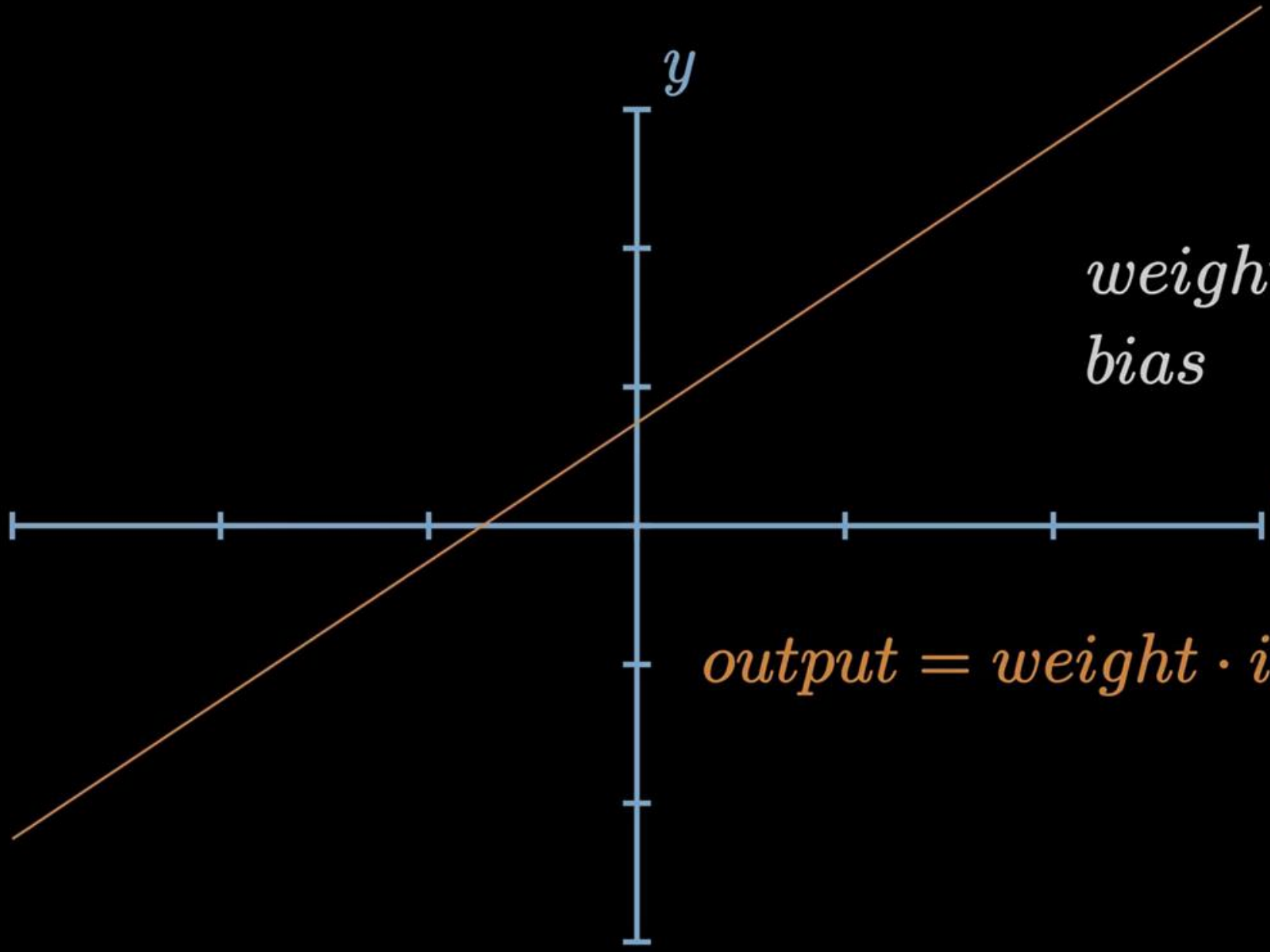
$$output = weight \cdot input + bias$$



$$weight = 1.00$$
$$bias = 1.59$$

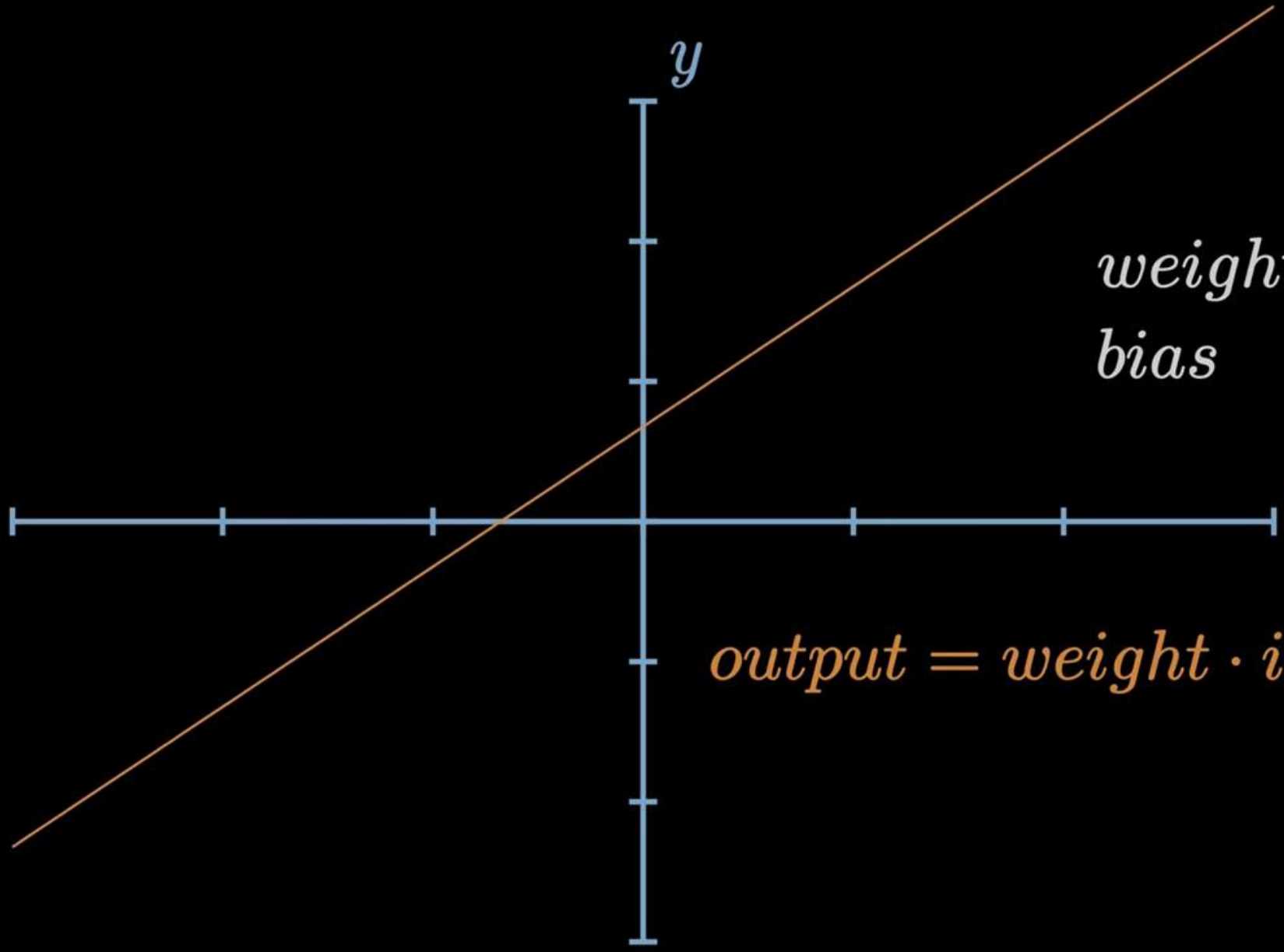
$$output = weight \cdot input + bias$$





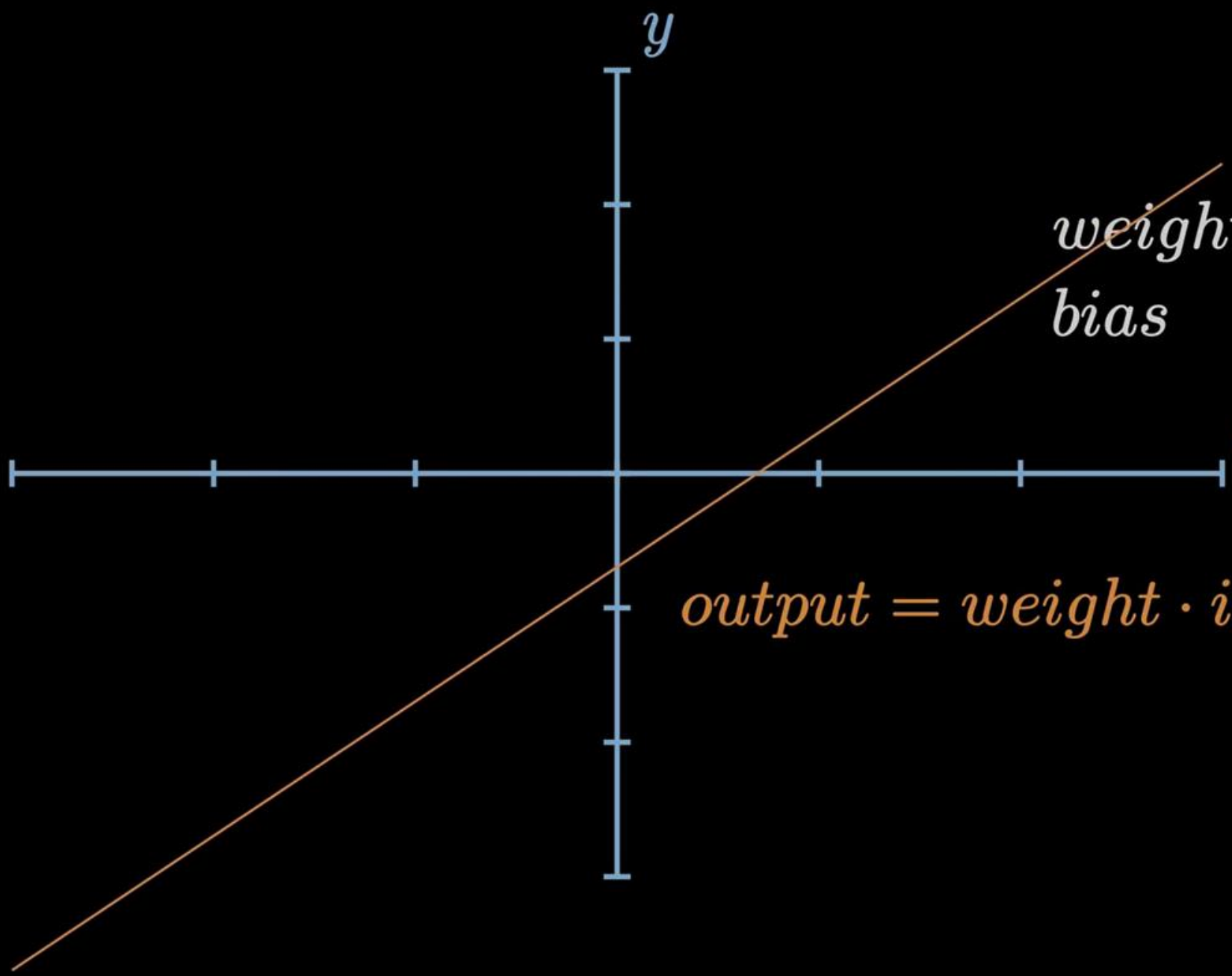
$$weight = 1.00$$
$$bias = 0.74$$

$$output = weight \cdot input + bias$$



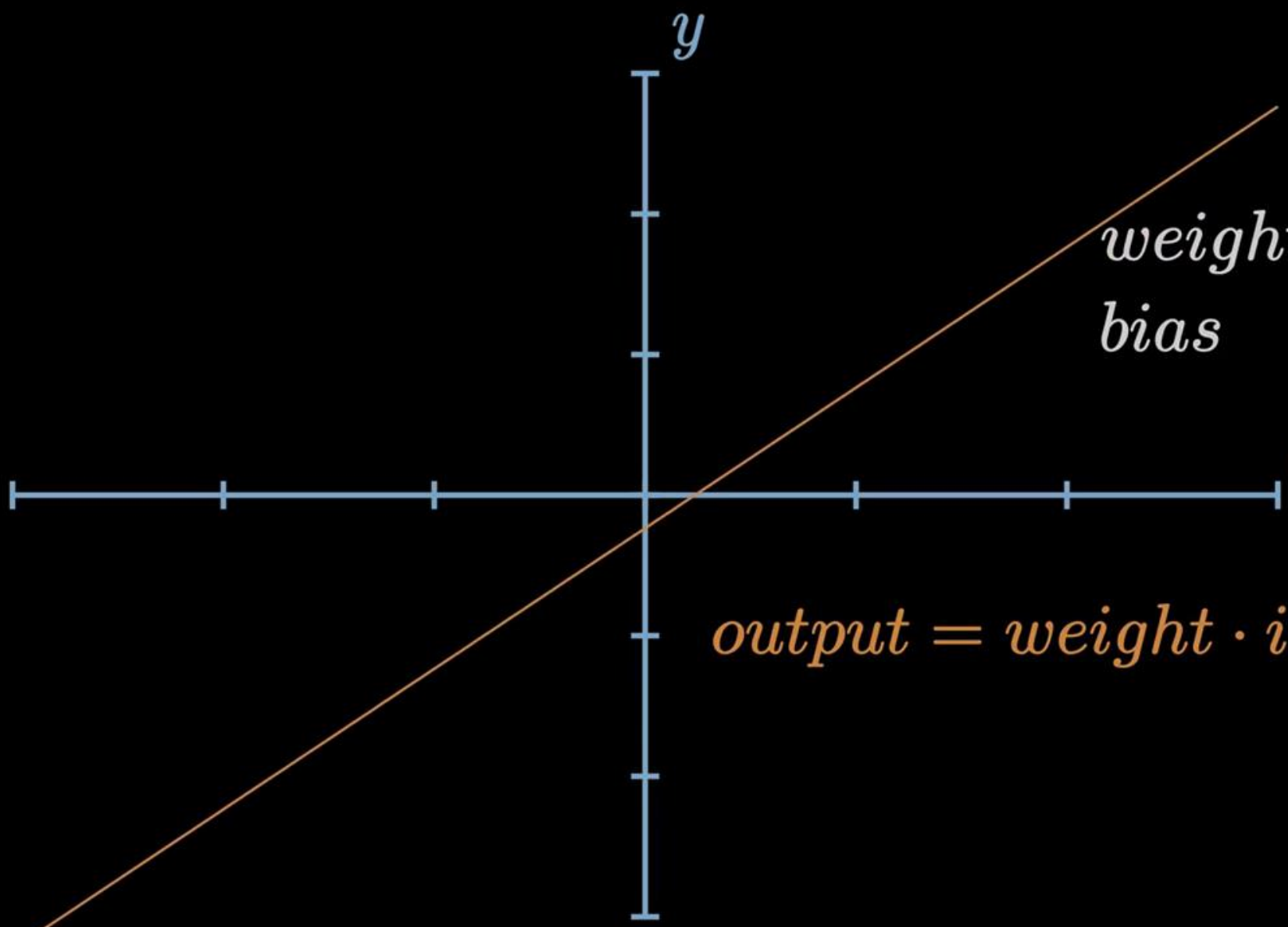
$$weight = 1.00$$
$$bias = 0.68$$

$$output = weight \cdot input + bias$$



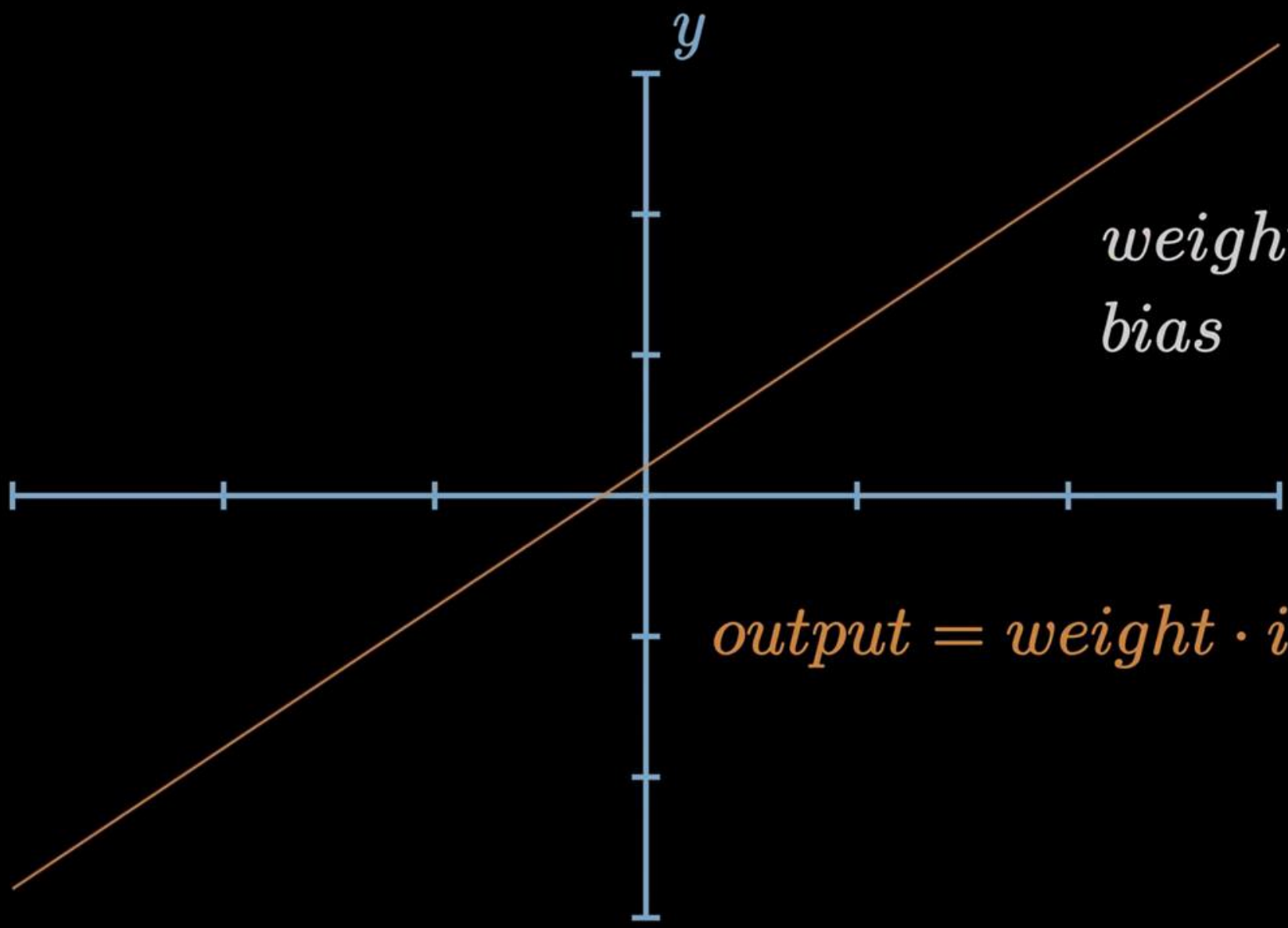
$$weight = 1.00$$
$$bias = -0.70$$

$$output = weight \cdot input + bias$$



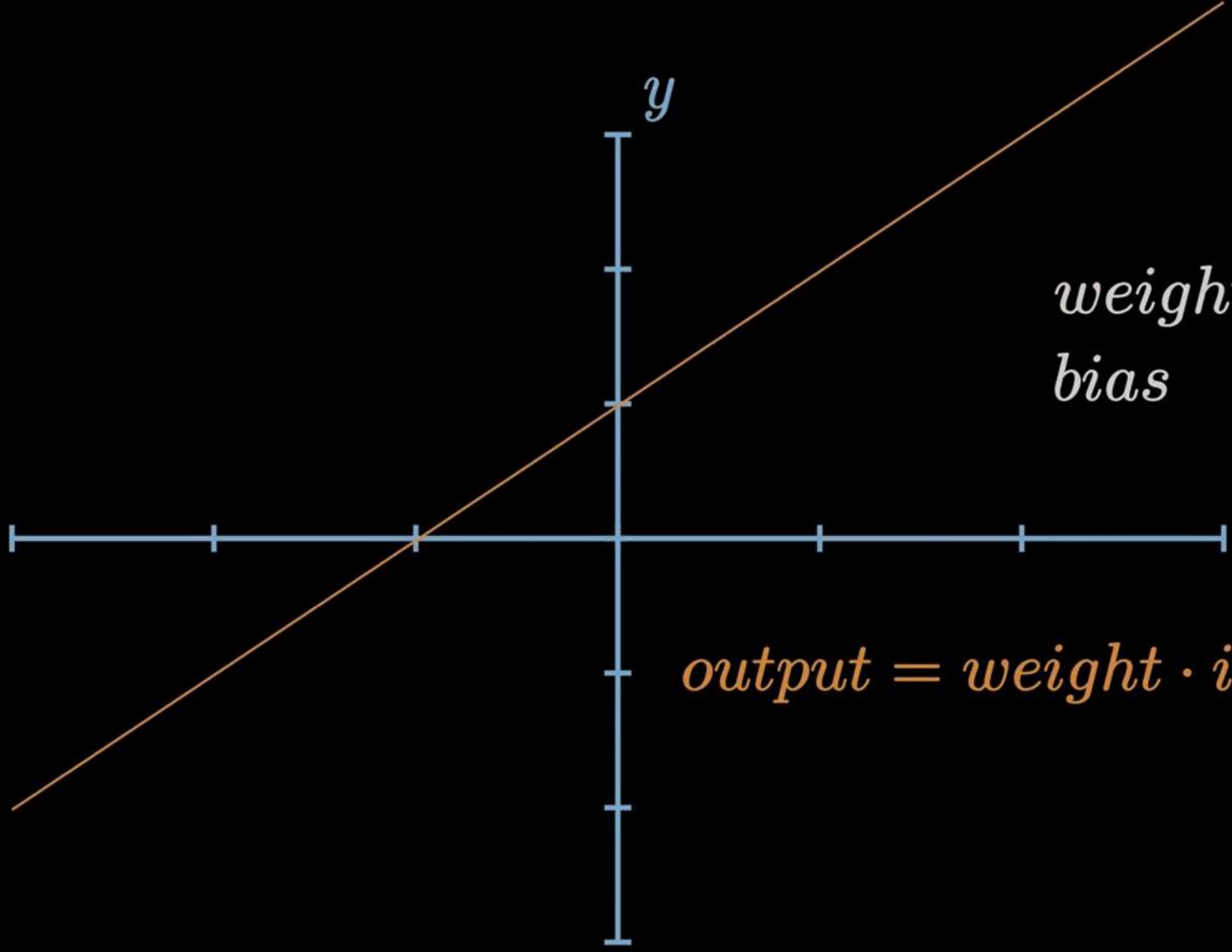
$$weight = 1.00$$
$$bias = -0.24$$

$$output = weight \cdot input + bias$$



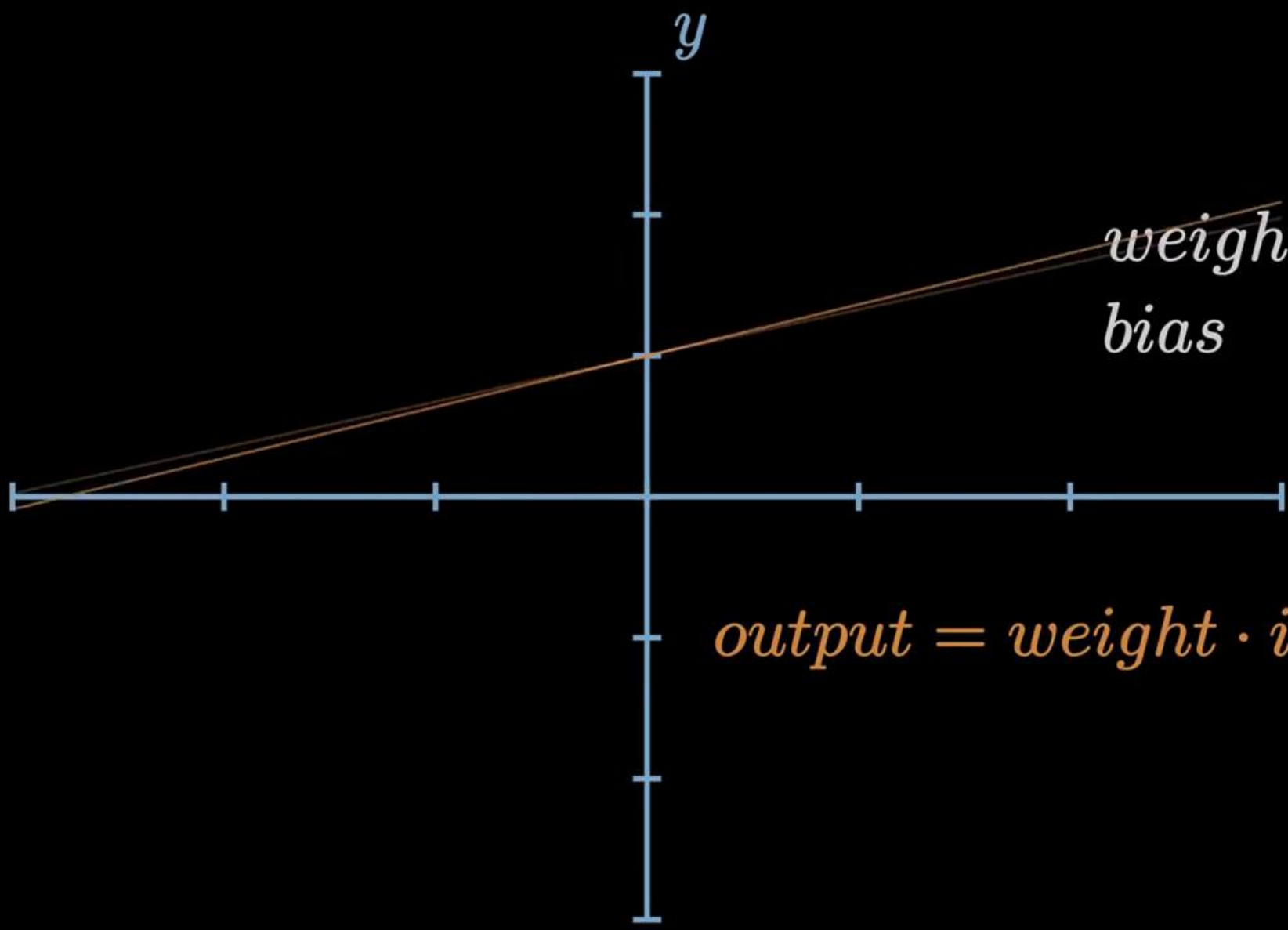
$$weight = 1.00$$
$$bias = 0.20$$

$$output = weight \cdot input + bias$$



$$weight = 1.00$$
$$bias = 0.98$$

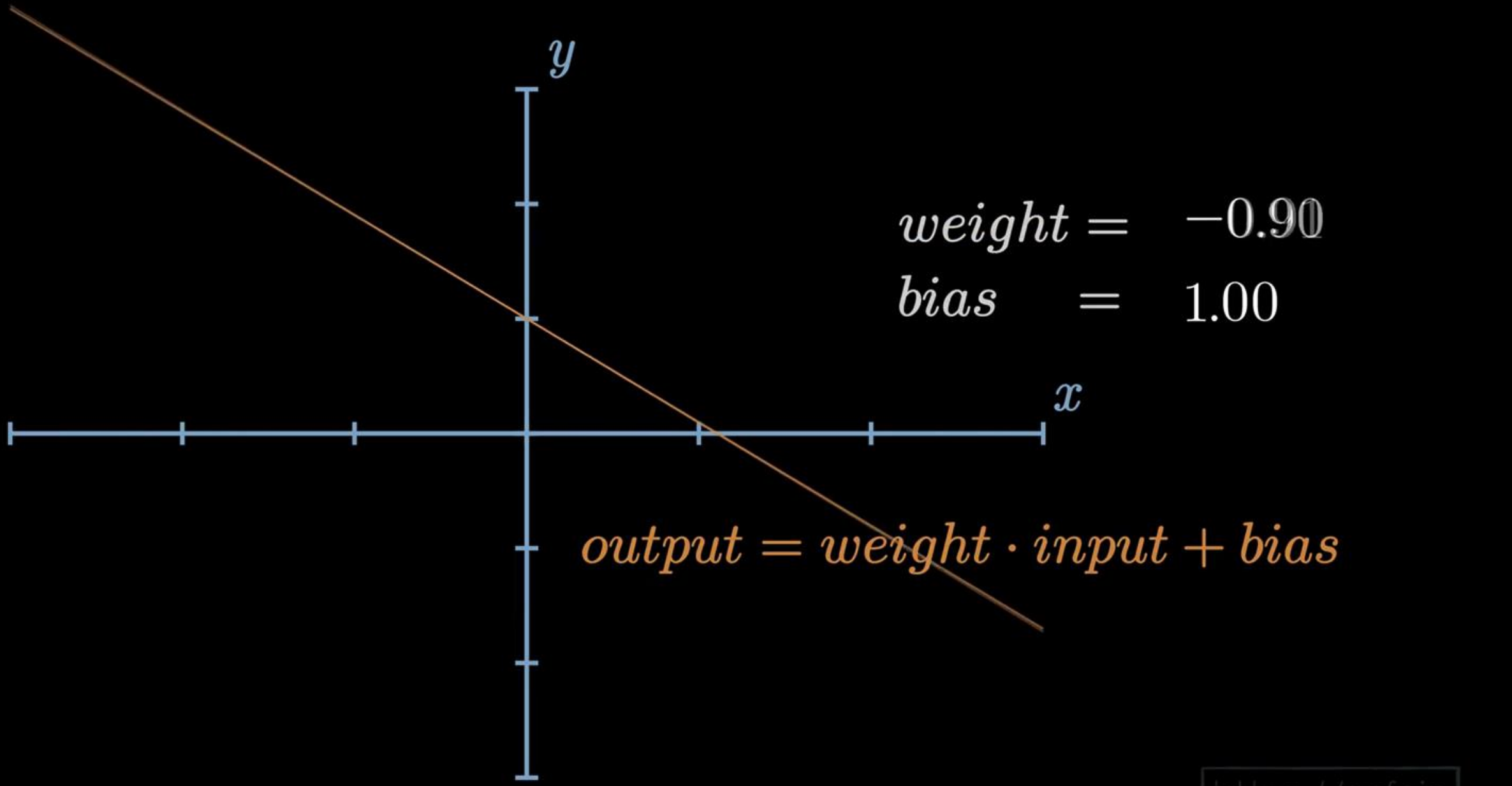
$$output = weight \cdot input + bias$$

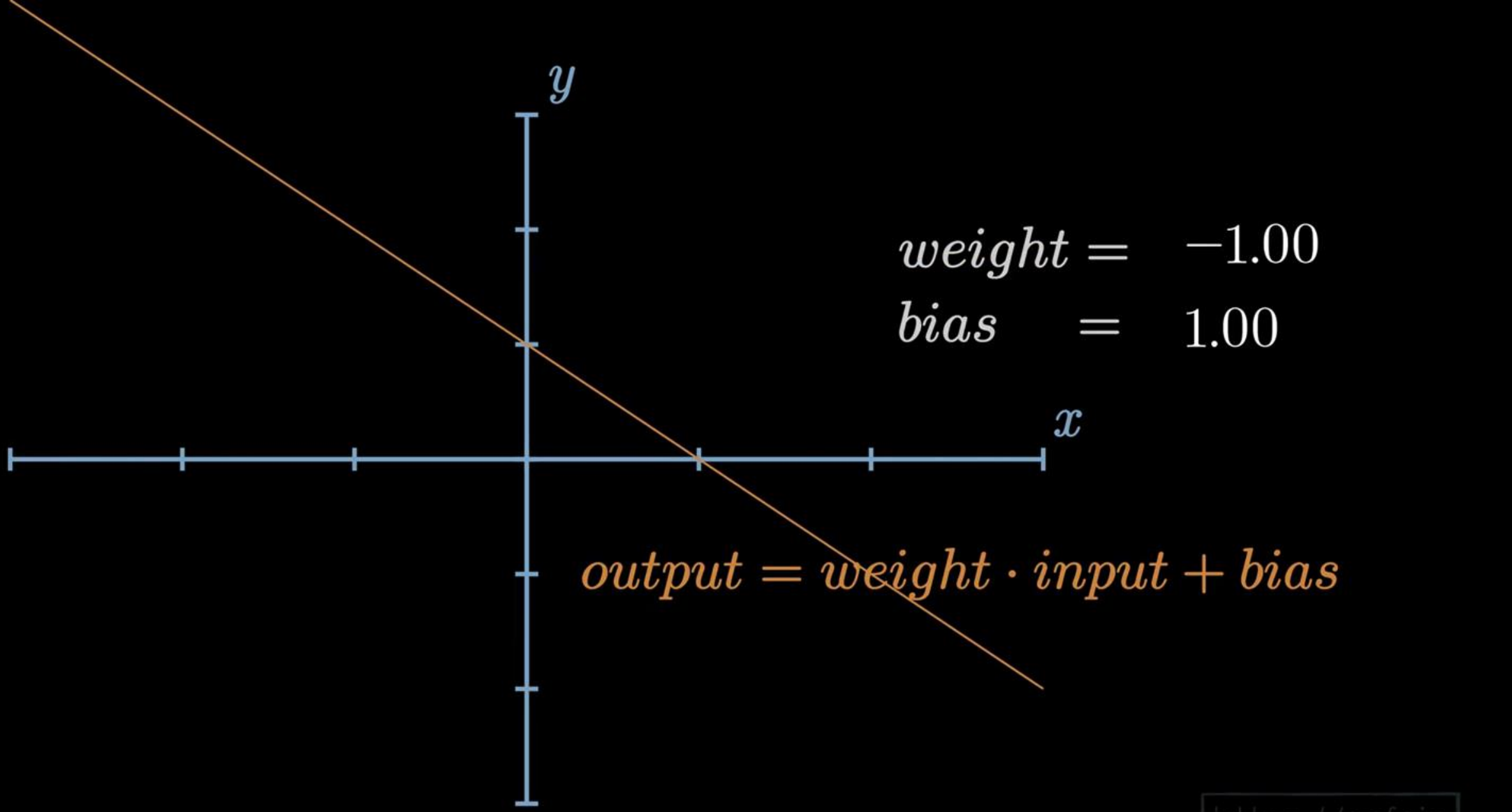


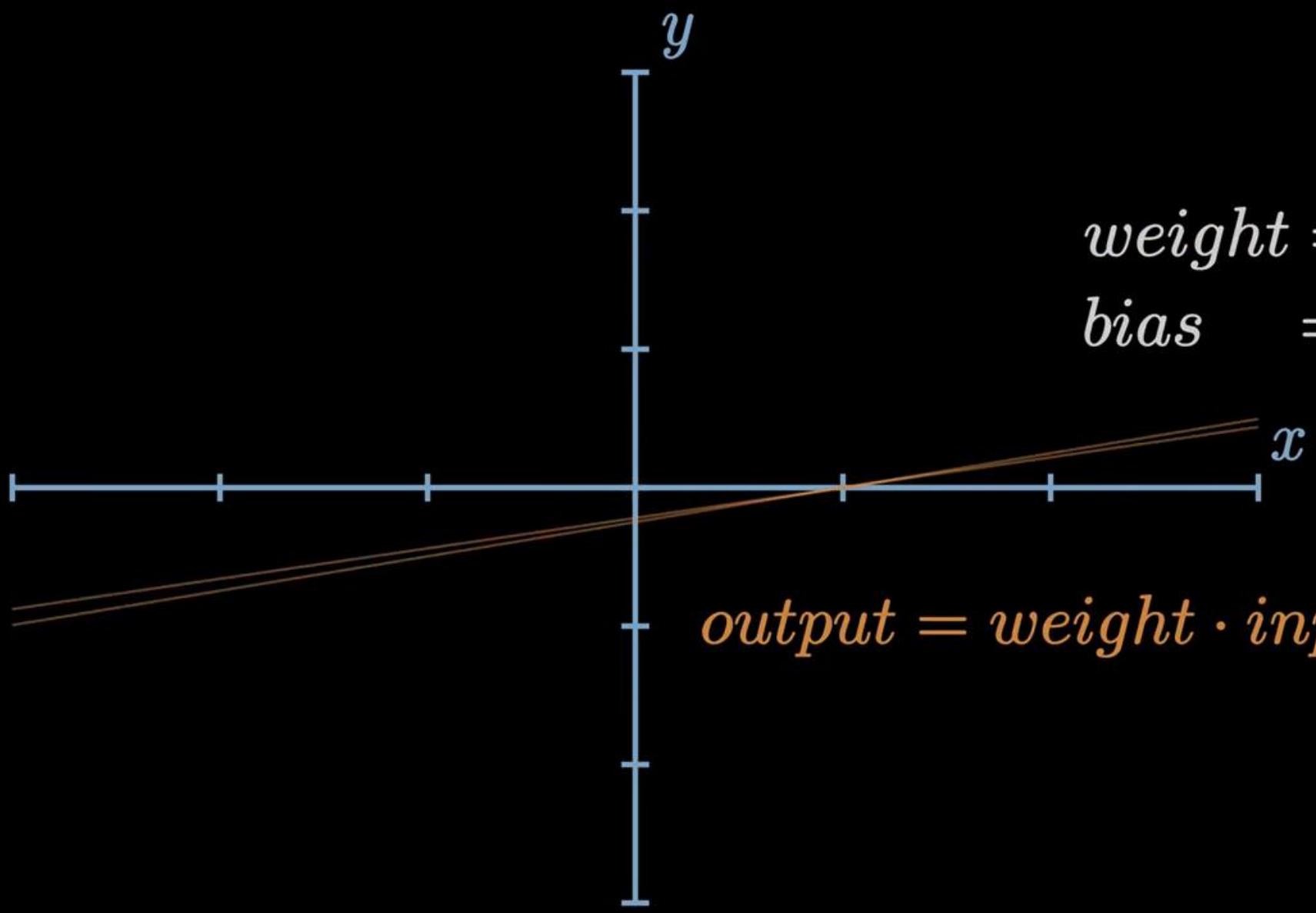
$$weight = 0.36$$

$$bias = 1.00$$

$$output = weight \cdot input + bias$$

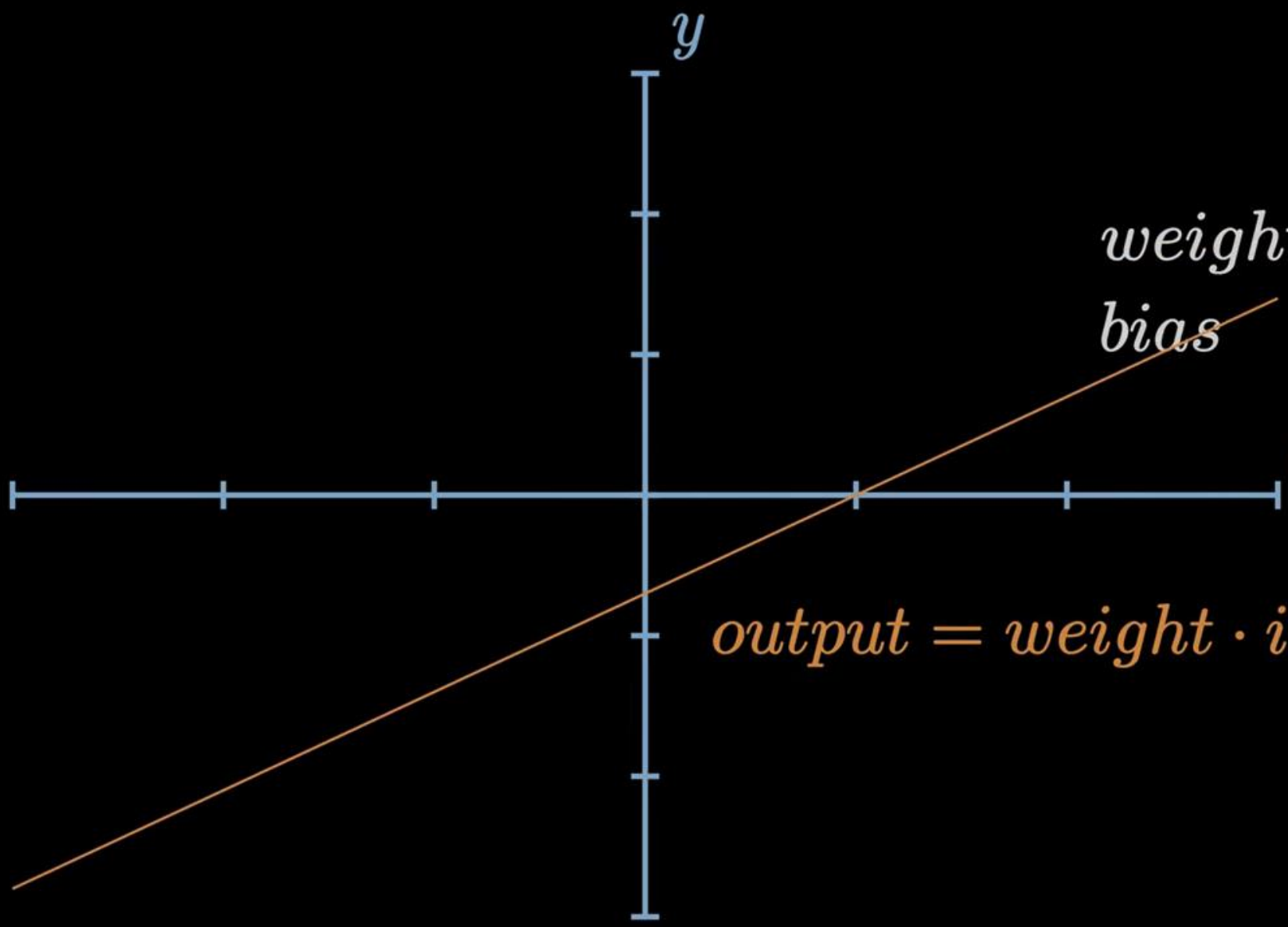






$$weight = 0.30$$
$$bias = -0.25$$

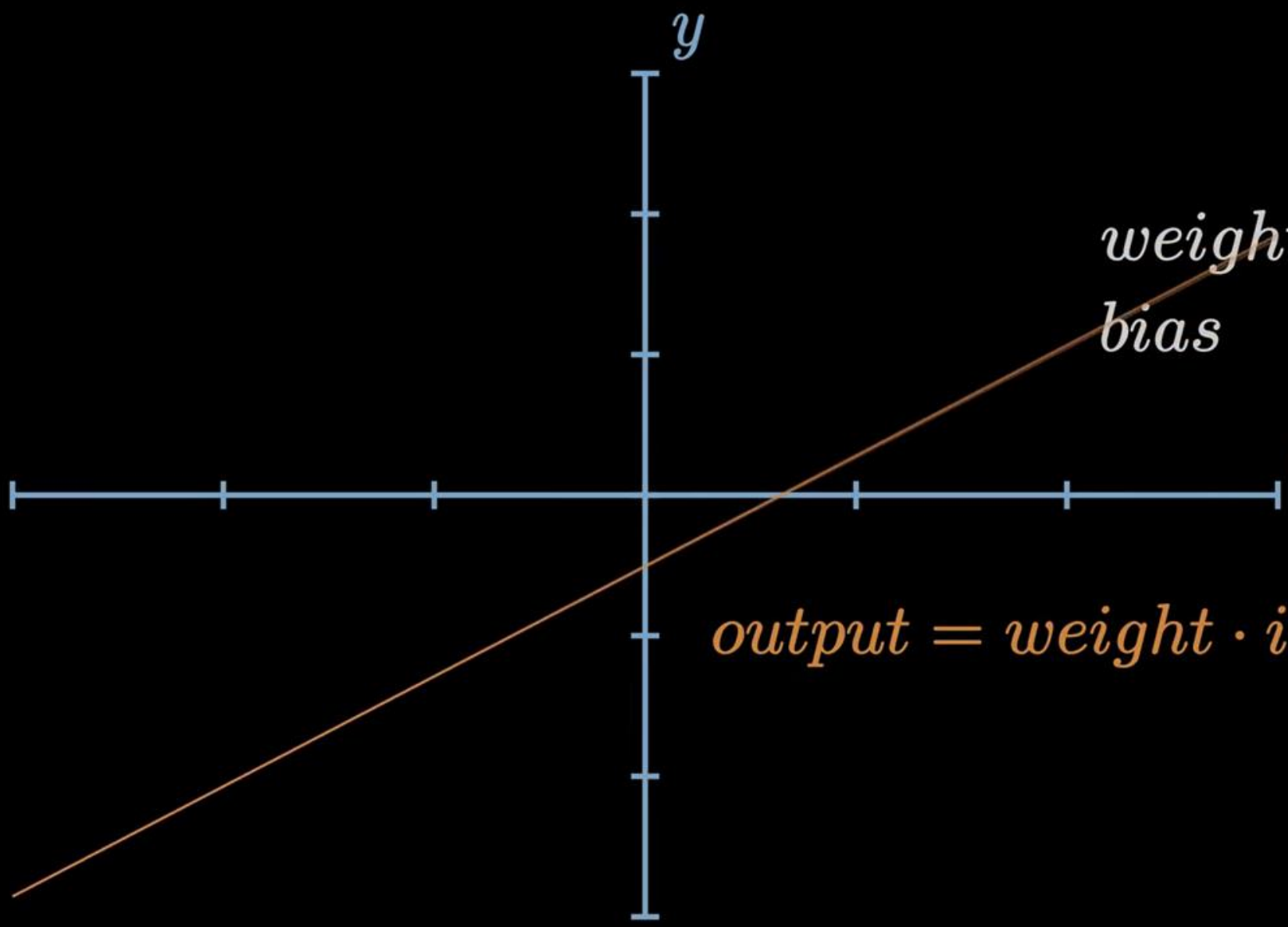
$$output = weight \cdot input + bias$$



$$weight = 0.70$$

$$bias = -0.70$$

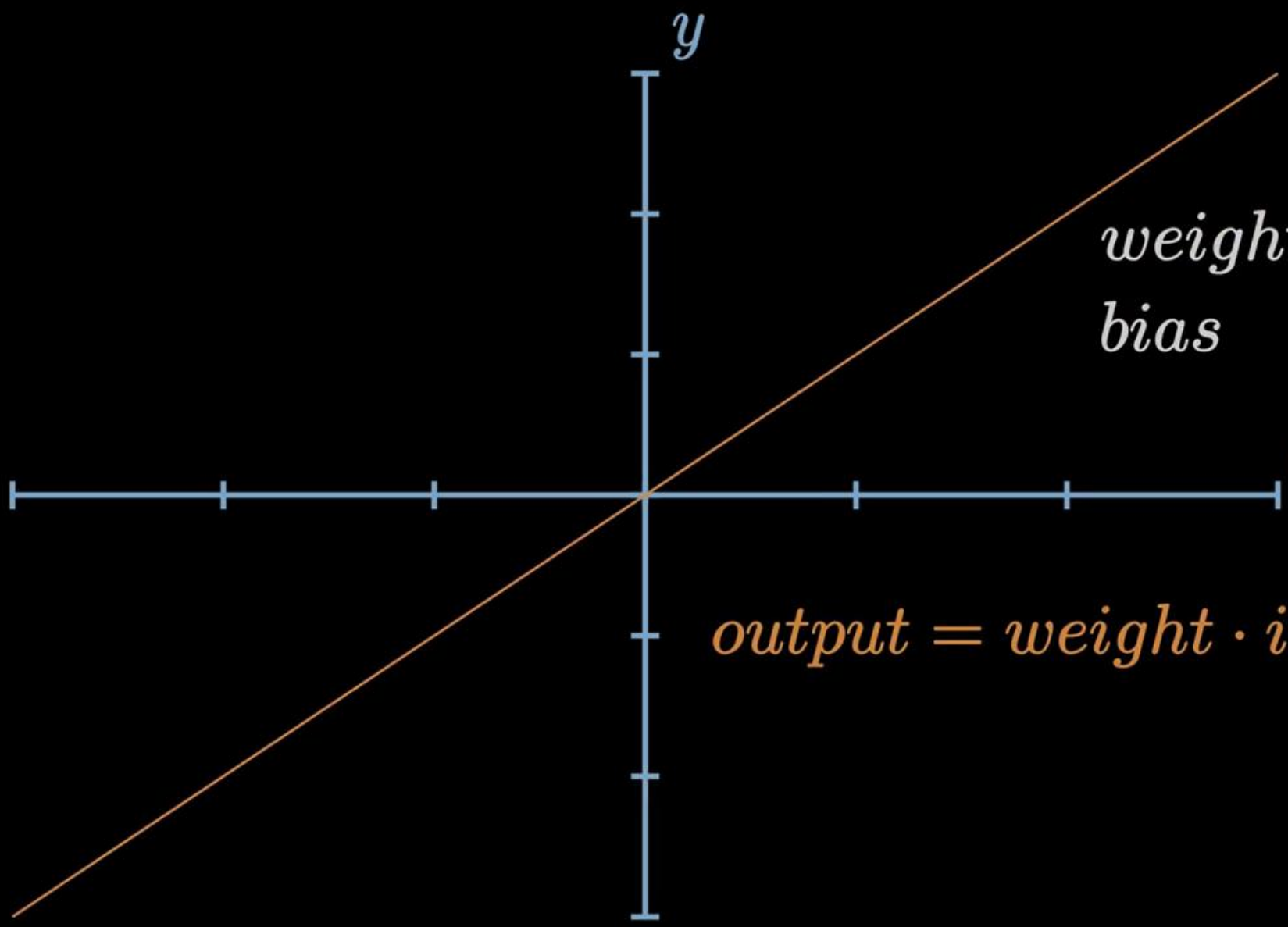
$$output = weight \cdot input + bias$$



$$weight = 0.78$$

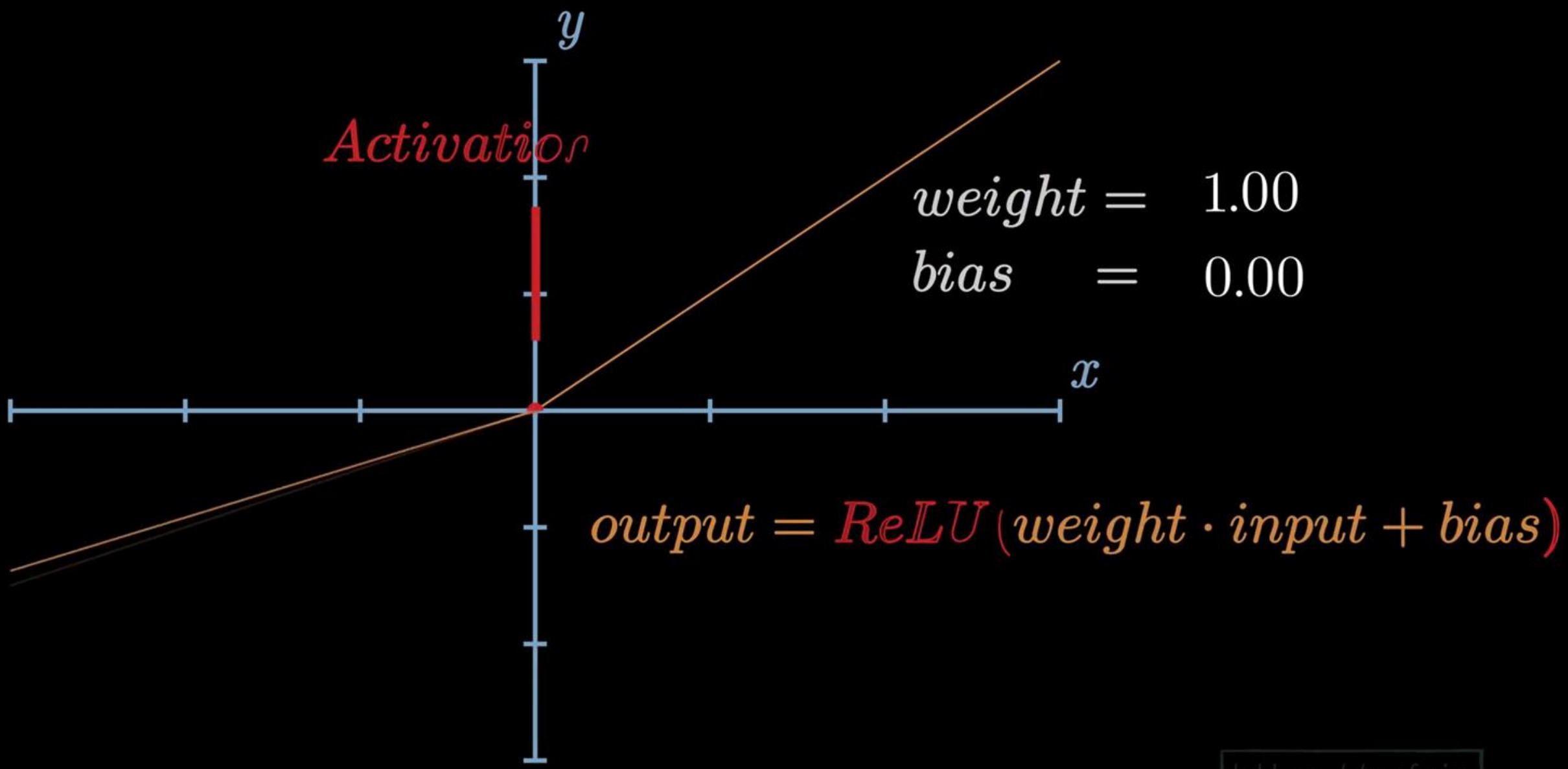
$$bias = -0.50$$

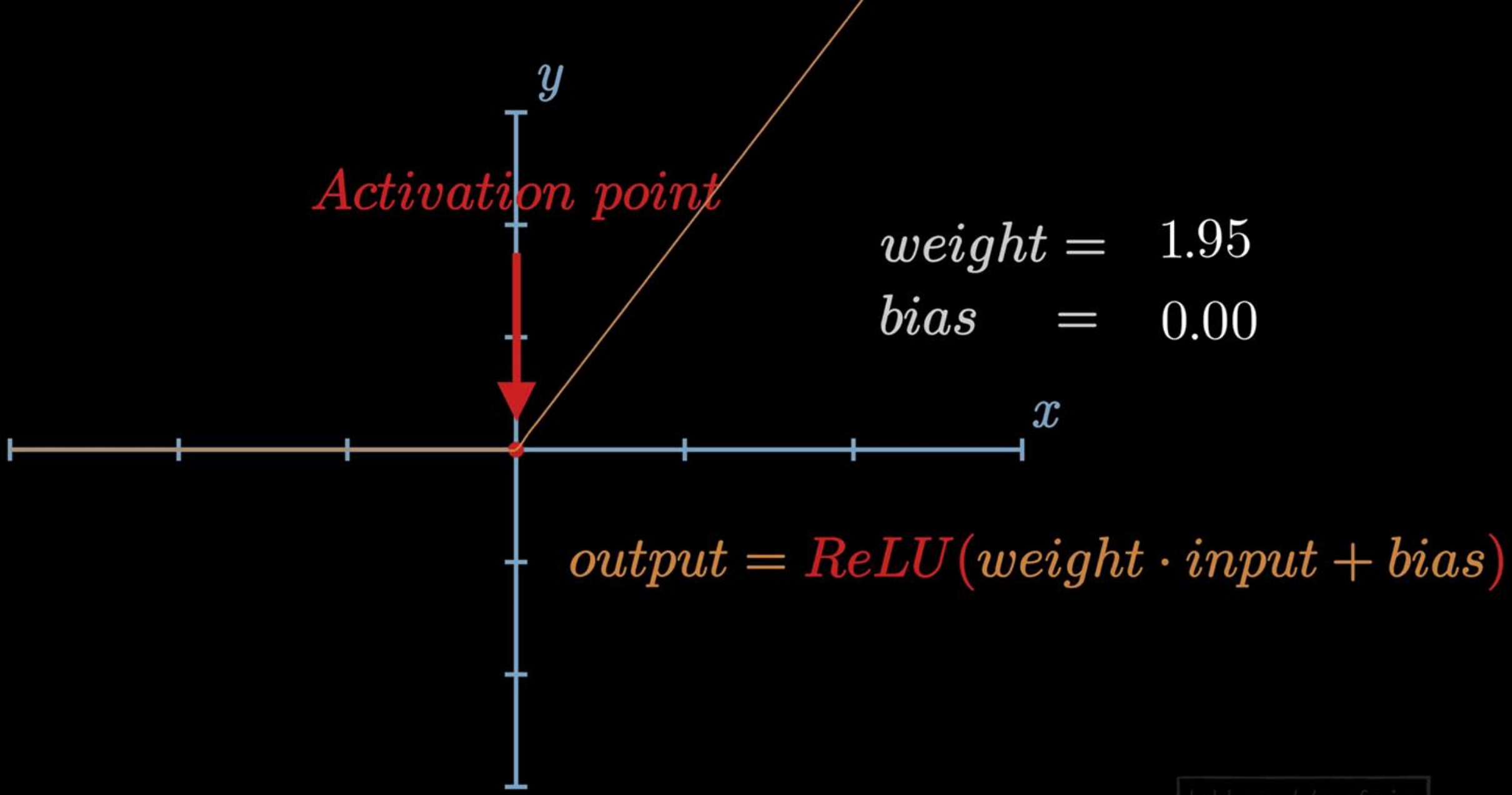
$$output = weight \cdot input + bias$$



$$weight = 1.00$$
$$bias = 0.00$$

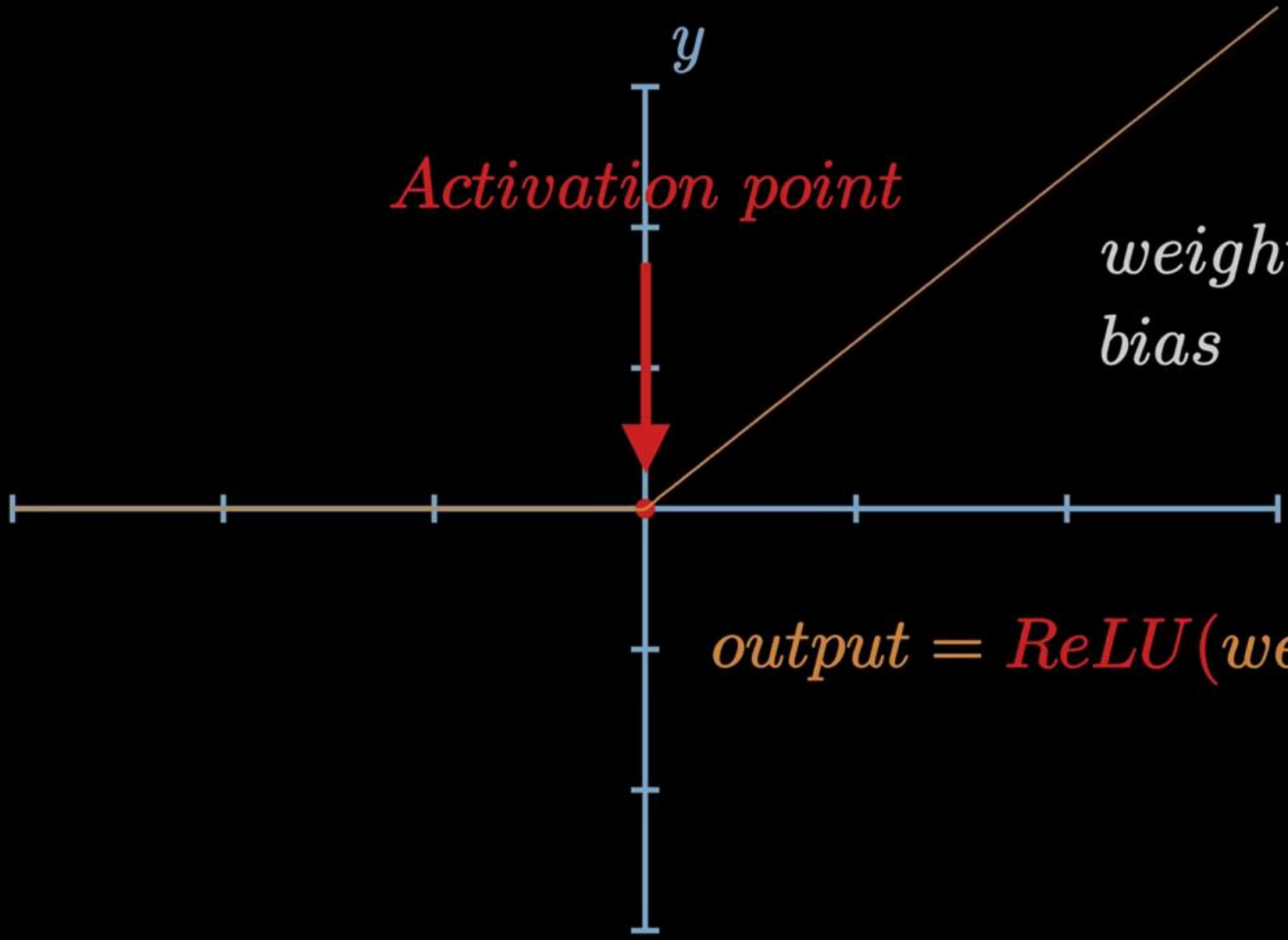
$$output = weight \cdot input + bias$$





$$weight = 1.95$$
$$bias = 0.00$$

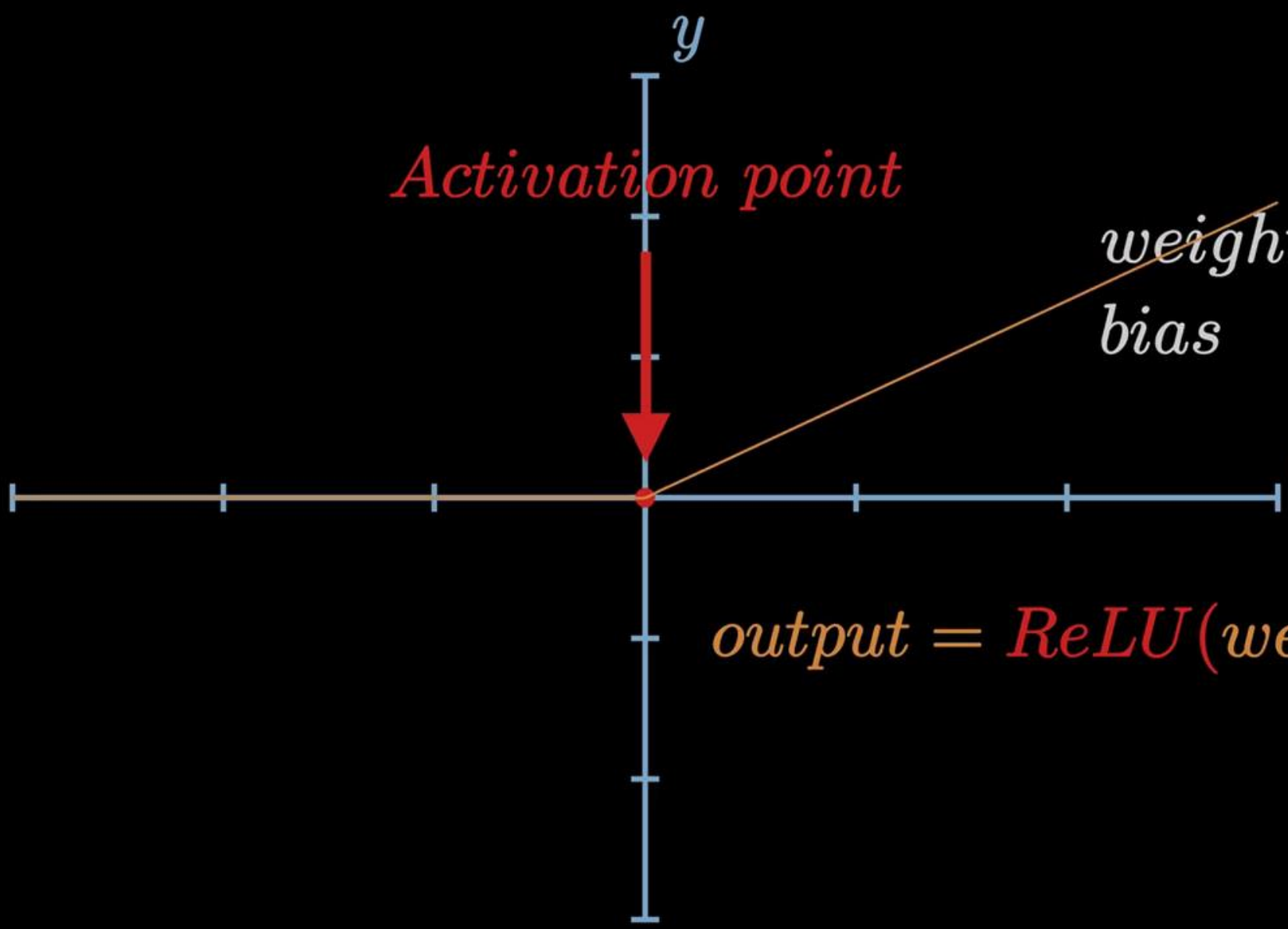
$$output = ReLU(weight \cdot input + bias)$$



$$weight = 1.19$$

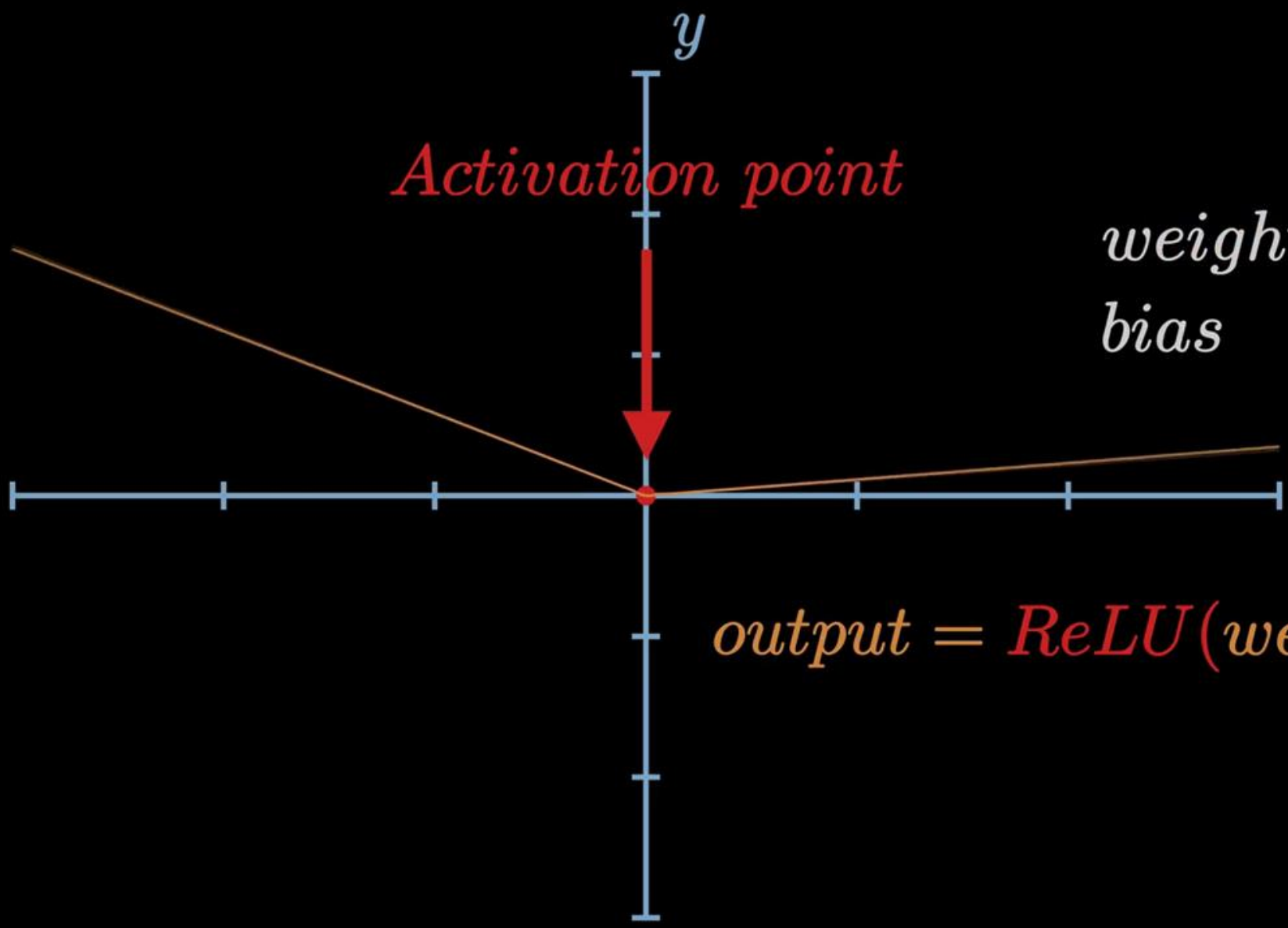
$$bias = 0.00$$

$$output = ReLU(weight \cdot input + bias)$$



$$weight = 0.70$$
$$bias = 0.00$$

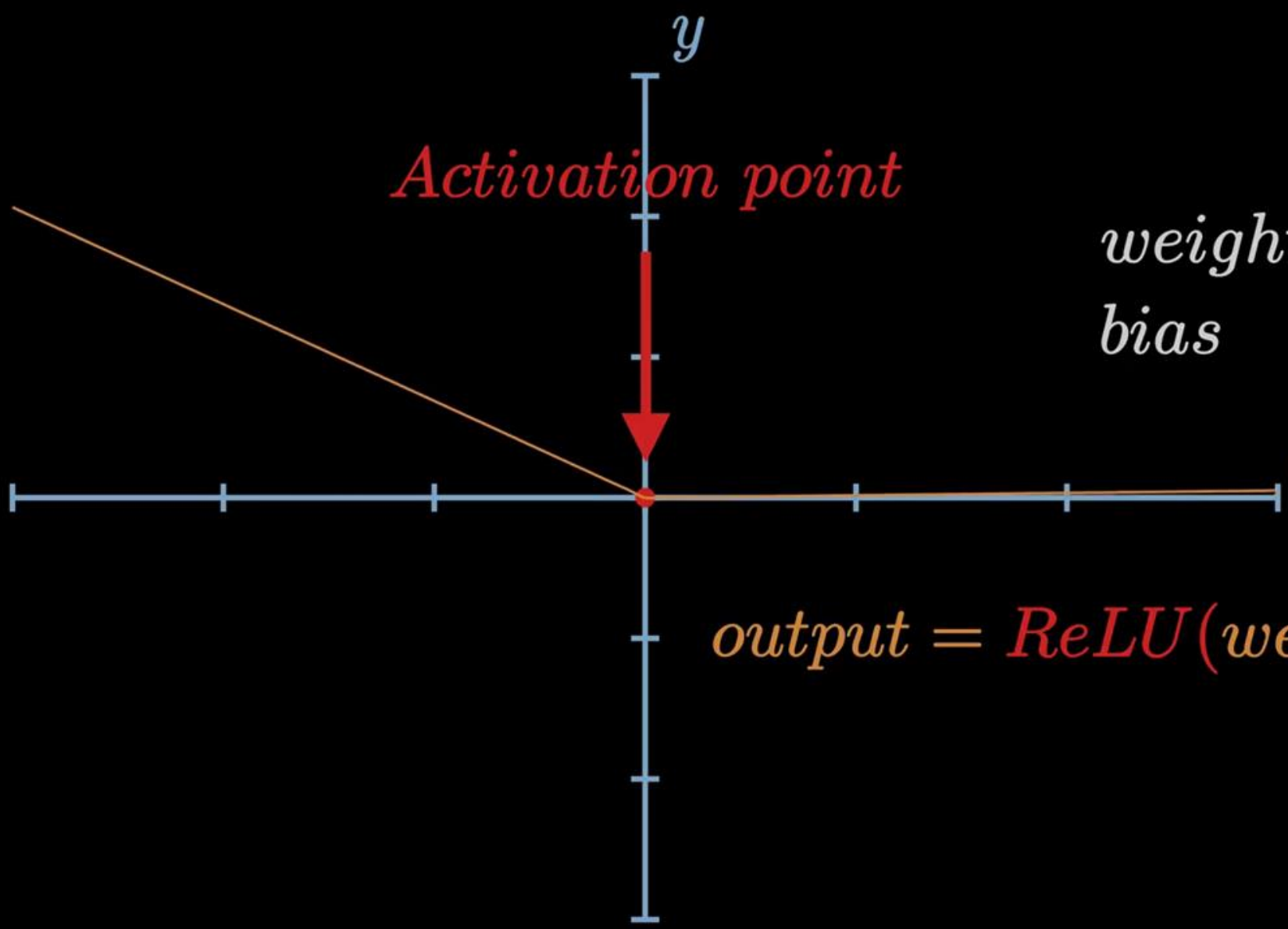
$$output = ReLU(weight \cdot input + bias)$$



$$weight = -0.47$$

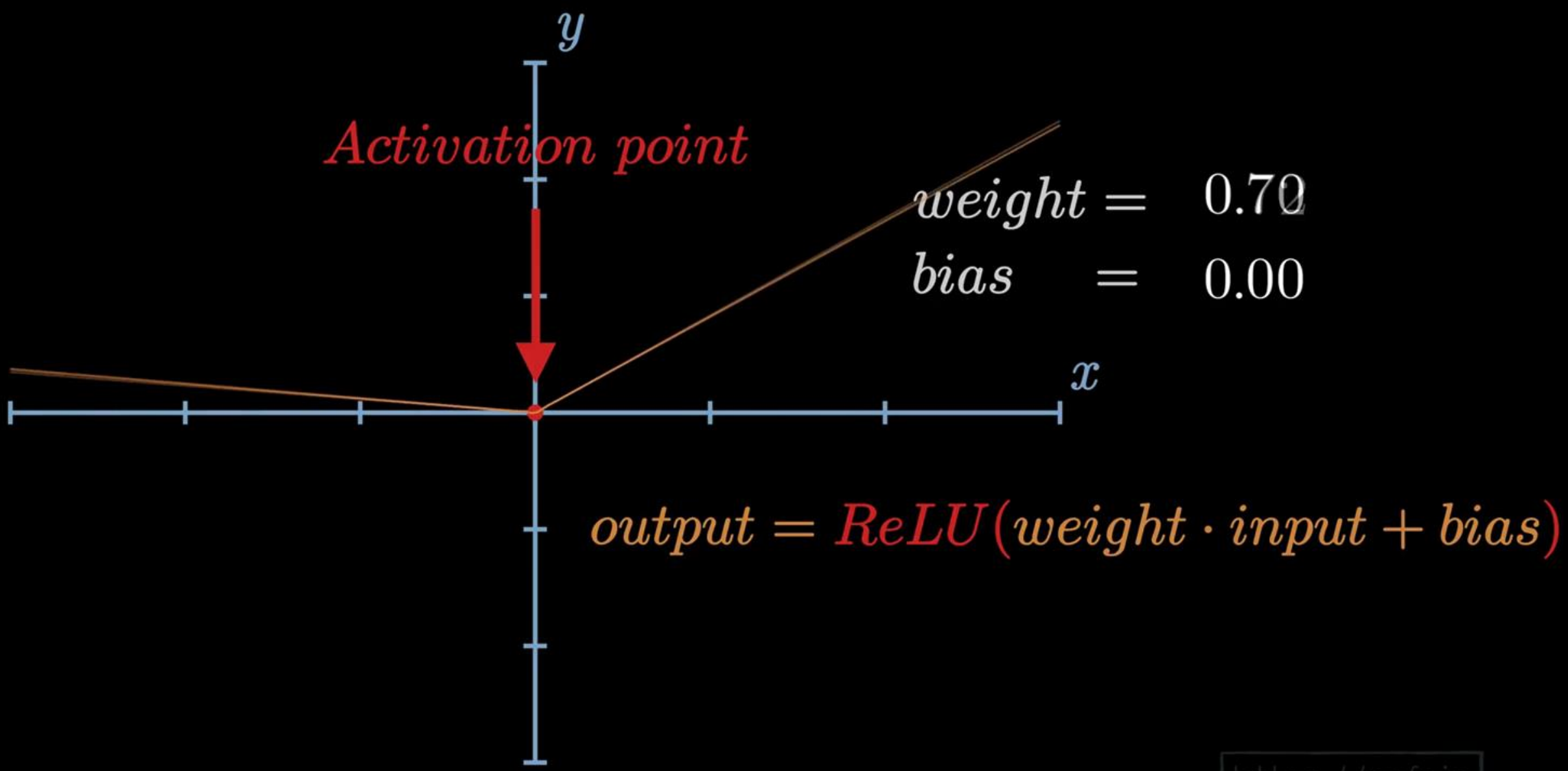
$$bias = 0.00$$

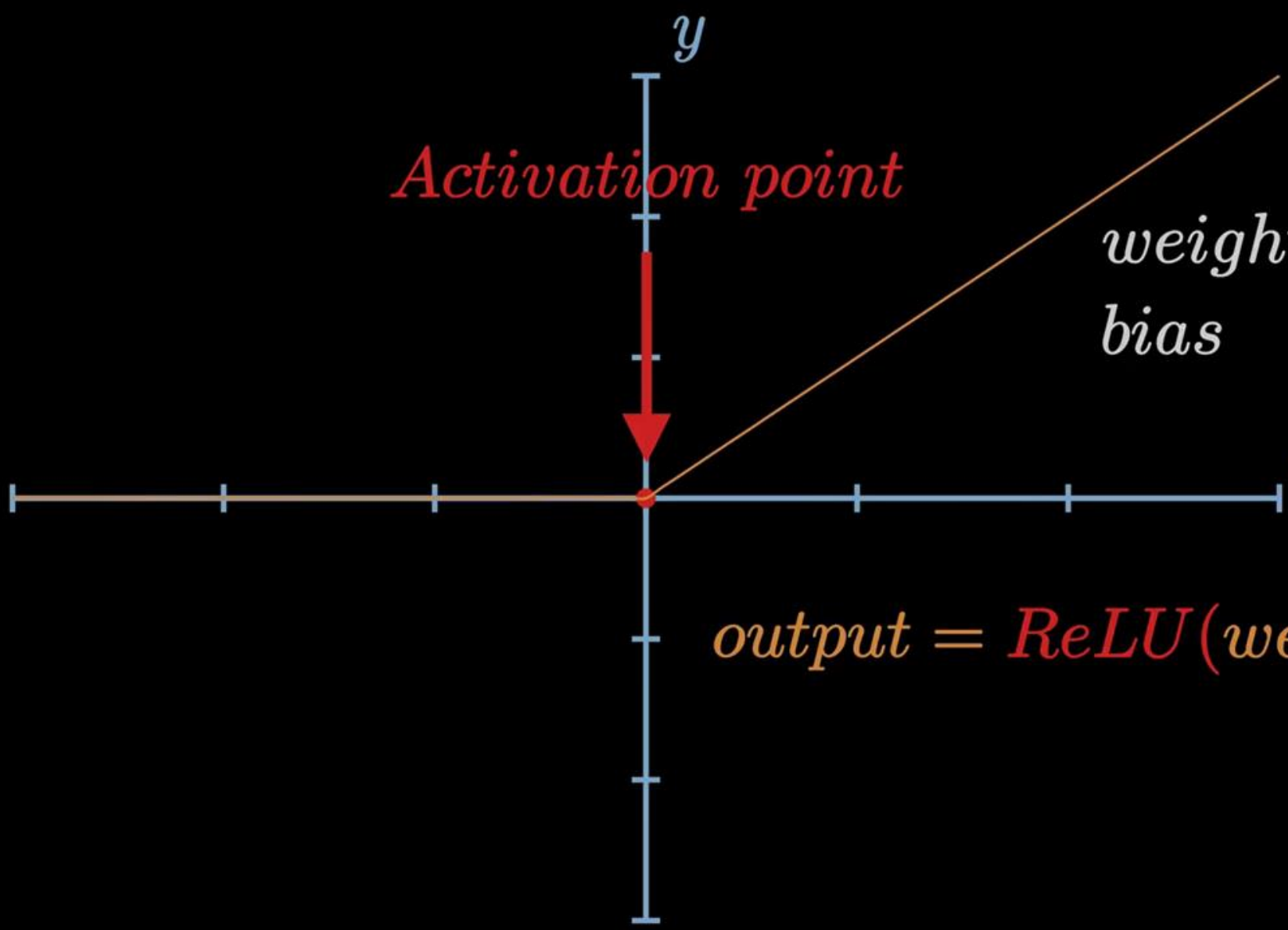
$$output = ReLU(weight \cdot input + bias)$$



$$weight = -0.67$$
$$bias = 0.00$$

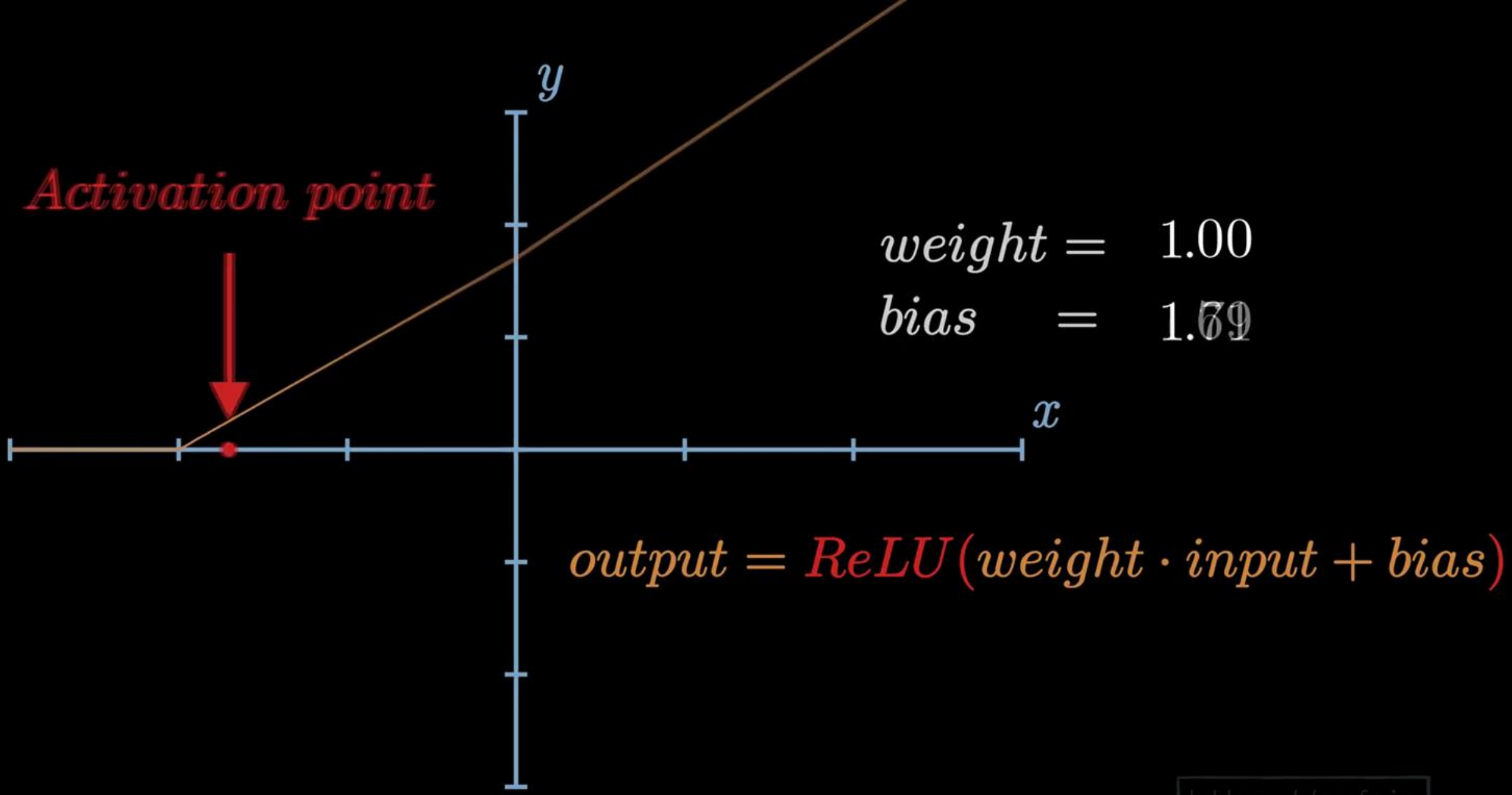
$$output = ReLU(weight \cdot input + bias)$$





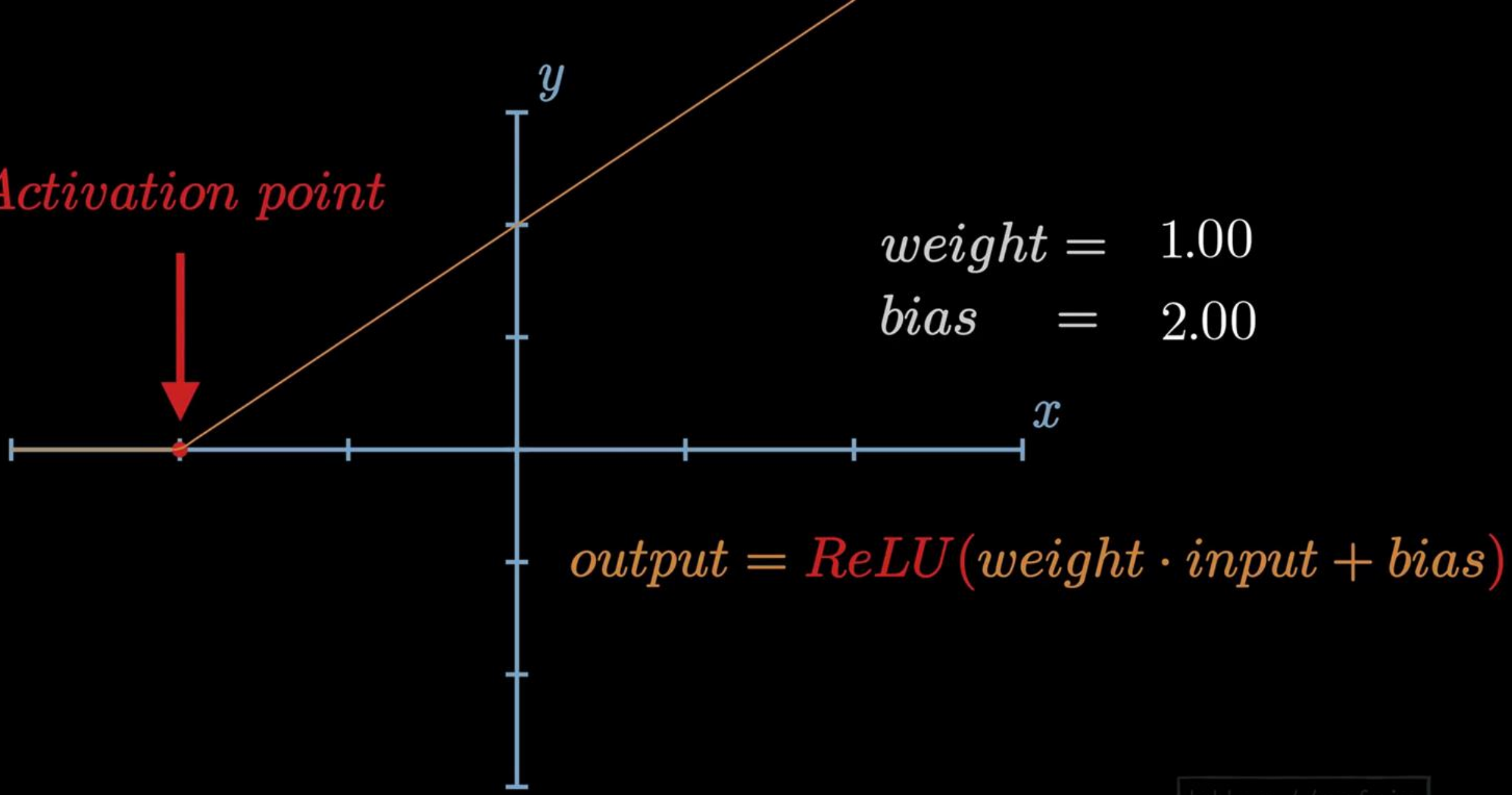
$$weight = 1.00$$
$$bias = 0.00$$

$$output = ReLU(weight \cdot input + bias)$$



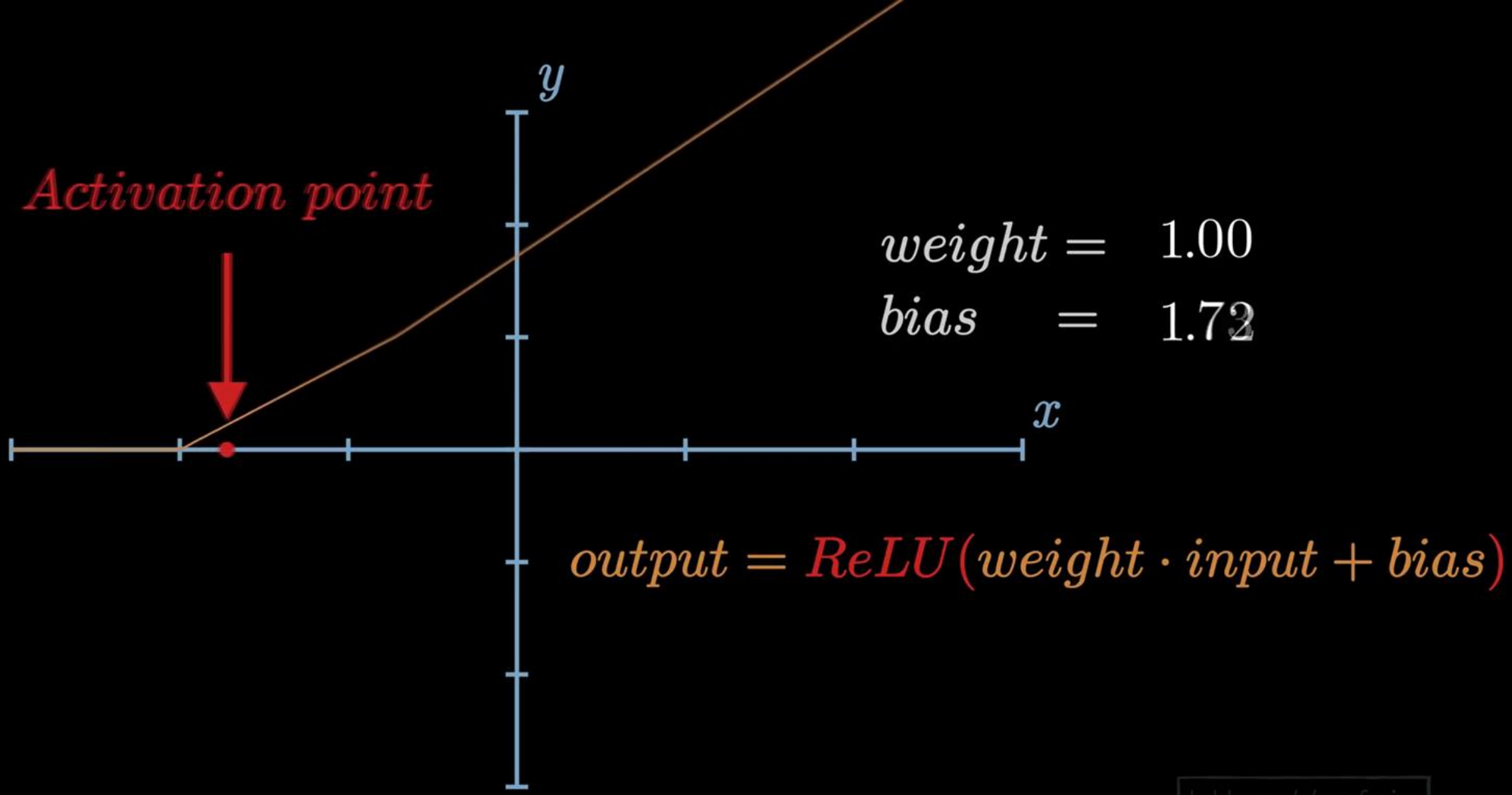
$$weight = 1.00$$
$$bias = 1.69$$

$$output = ReLU(weight \cdot input + bias)$$



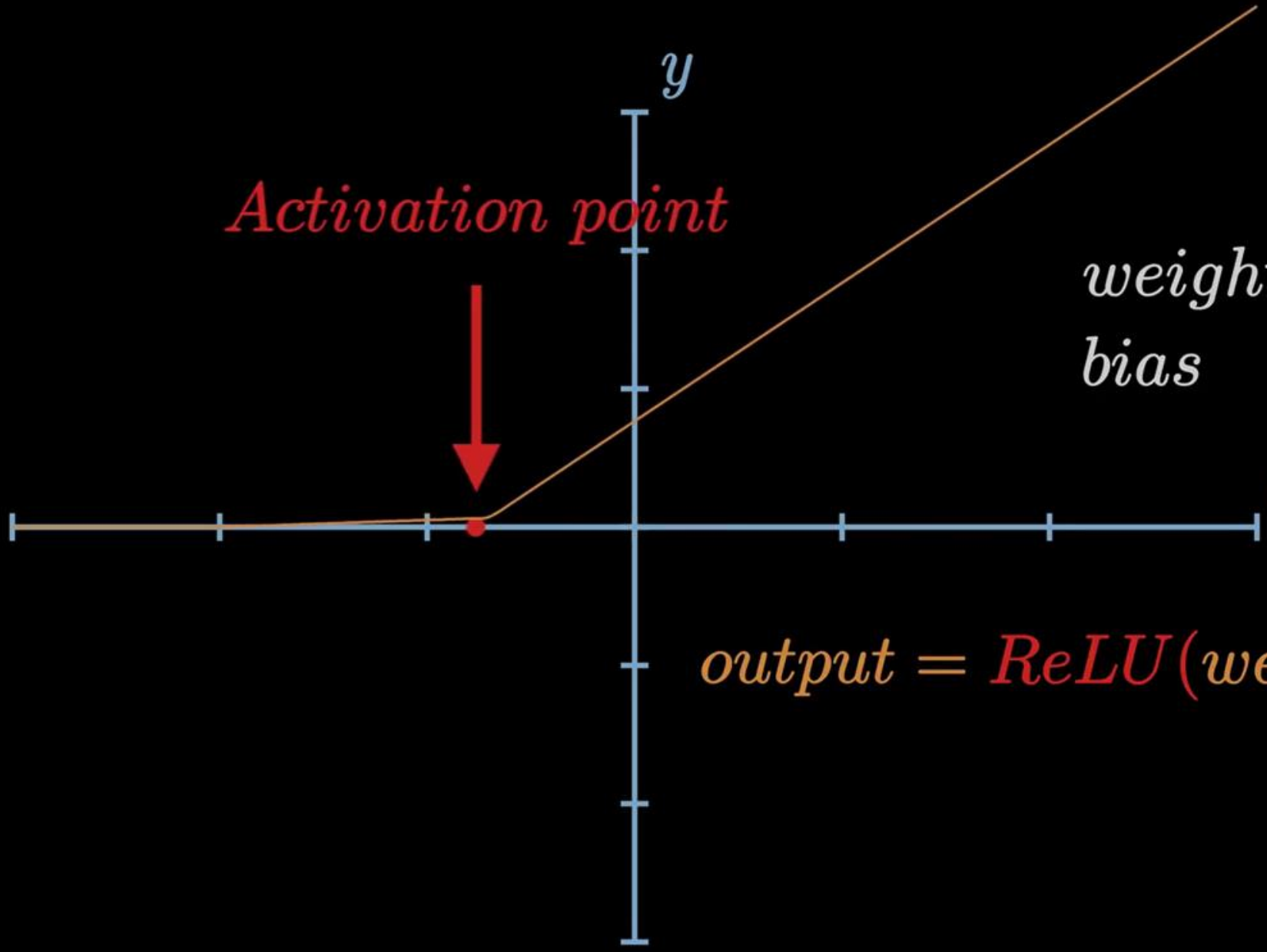
$$weight = 1.00$$
$$bias = 2.00$$

$$output = ReLU(weight \cdot input + bias)$$



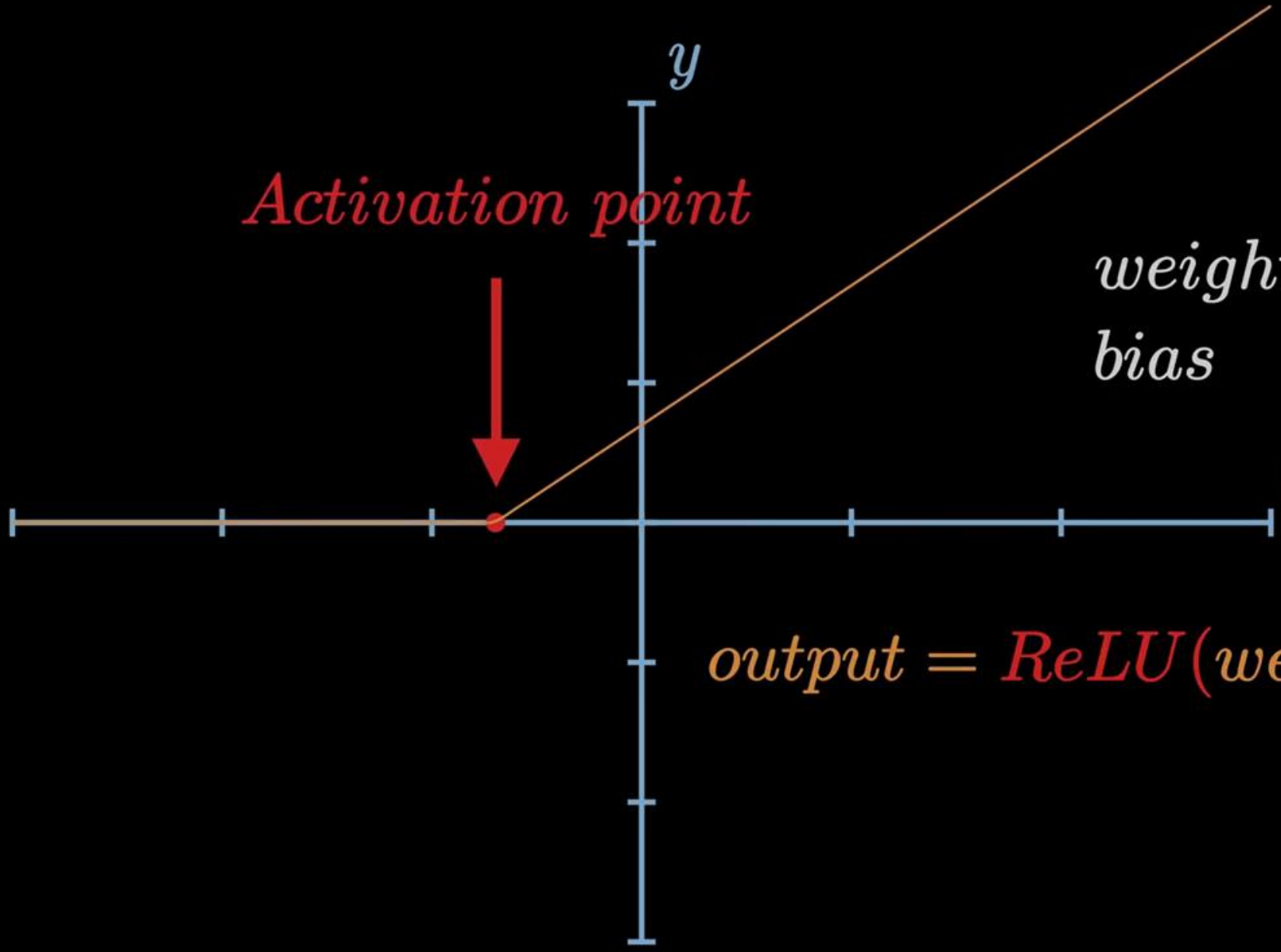
$$weight = 1.00$$
$$bias = 1.72$$

$$output = ReLU(weight \cdot input + bias)$$



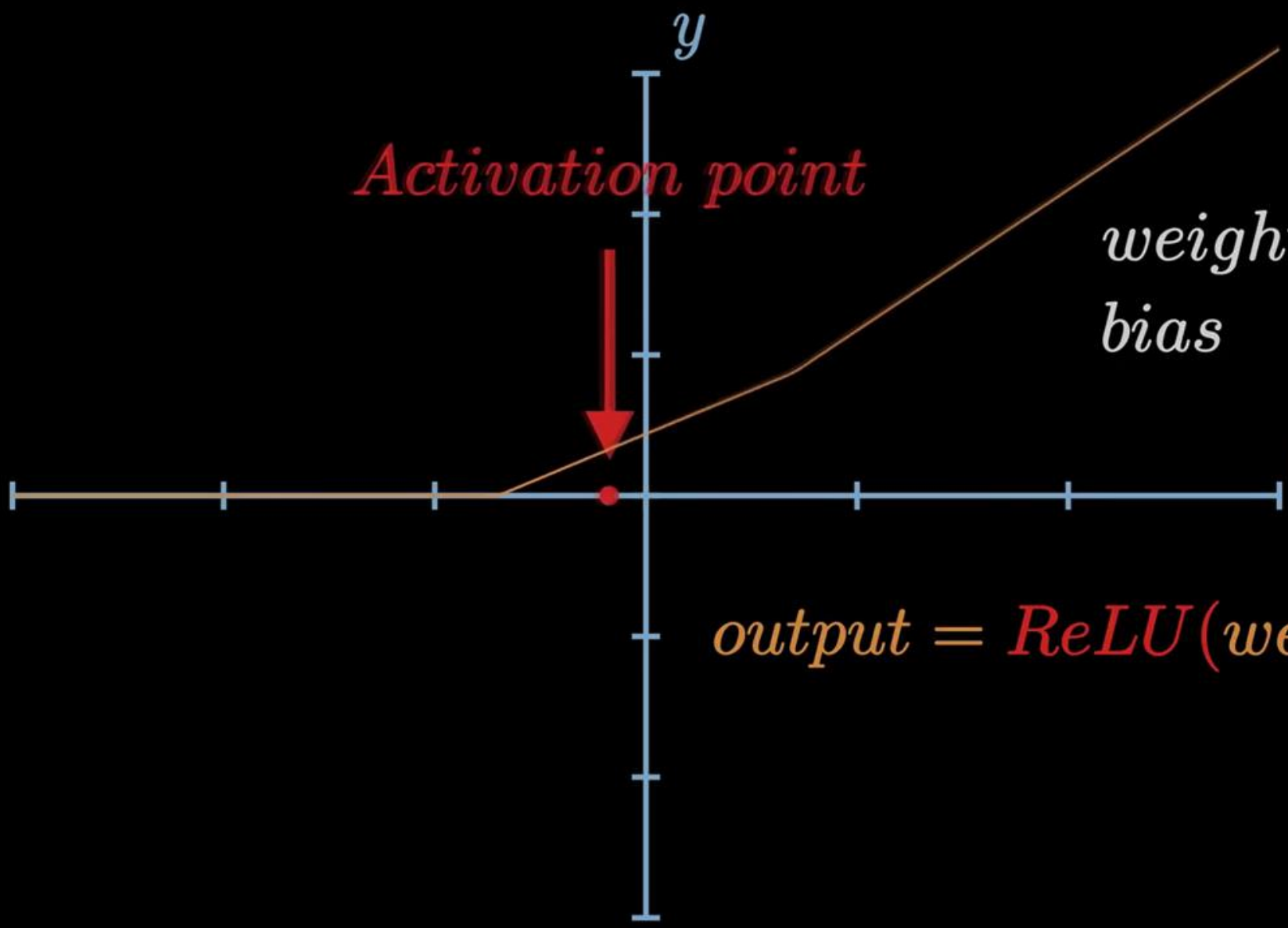
$$weight = 1.00$$
$$bias = 0.76$$

$$output = ReLU(weight \cdot input + bias)$$



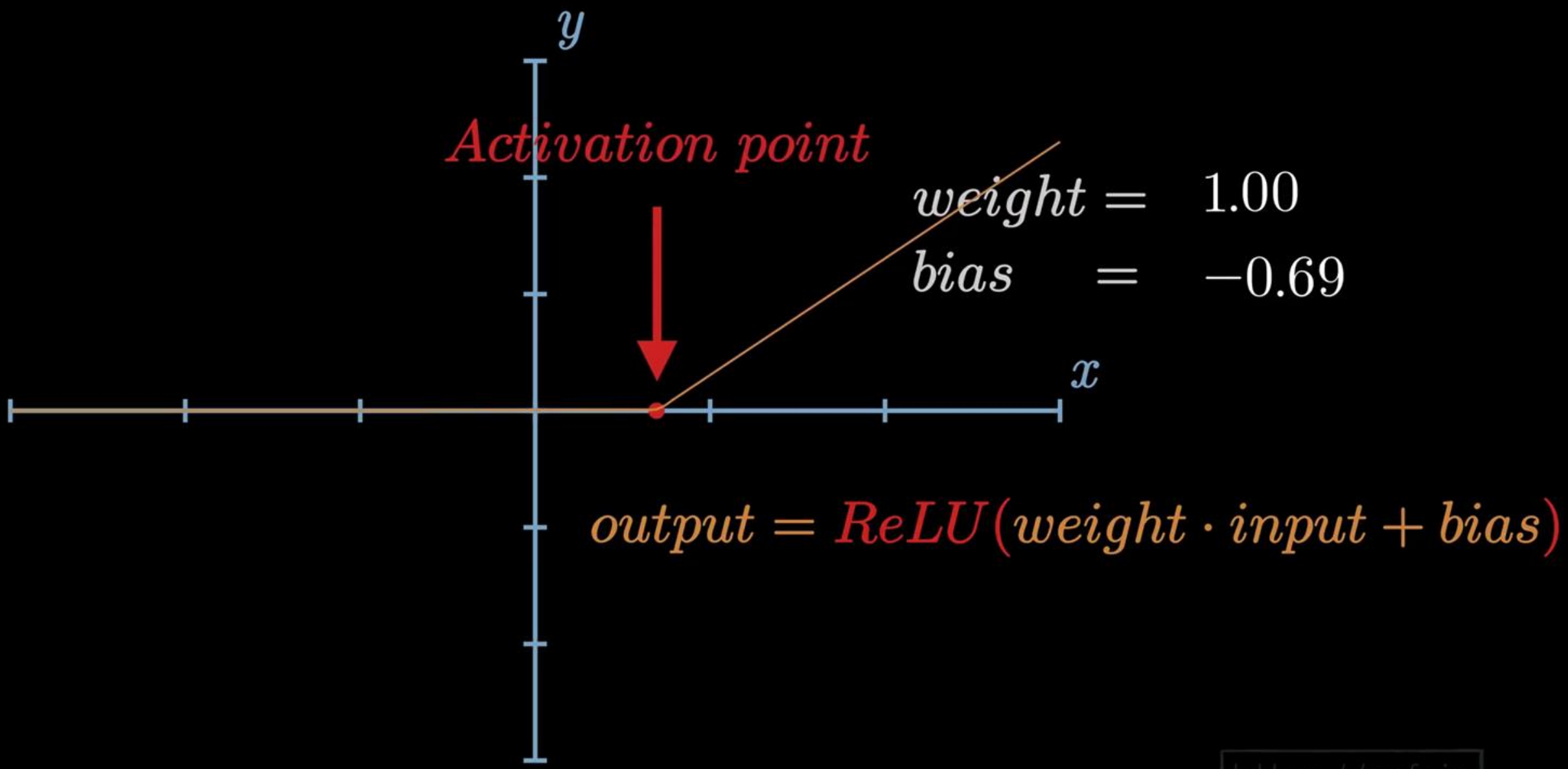
$$weight = 1.00$$
$$bias = 0.70$$

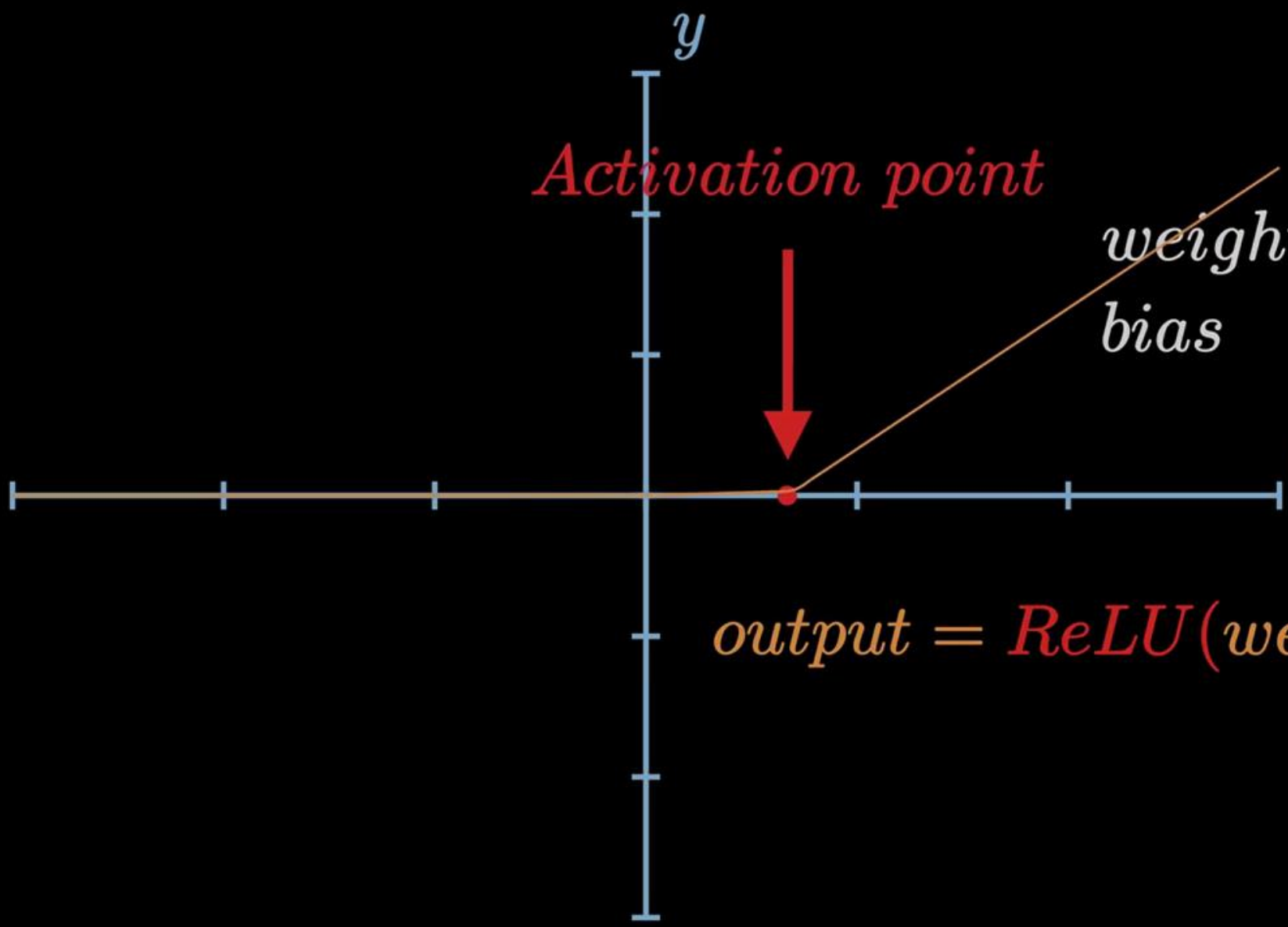
$$output = ReLU(weight \cdot input + bias)$$



$$weight = 1.00$$
$$bias = 0.17$$

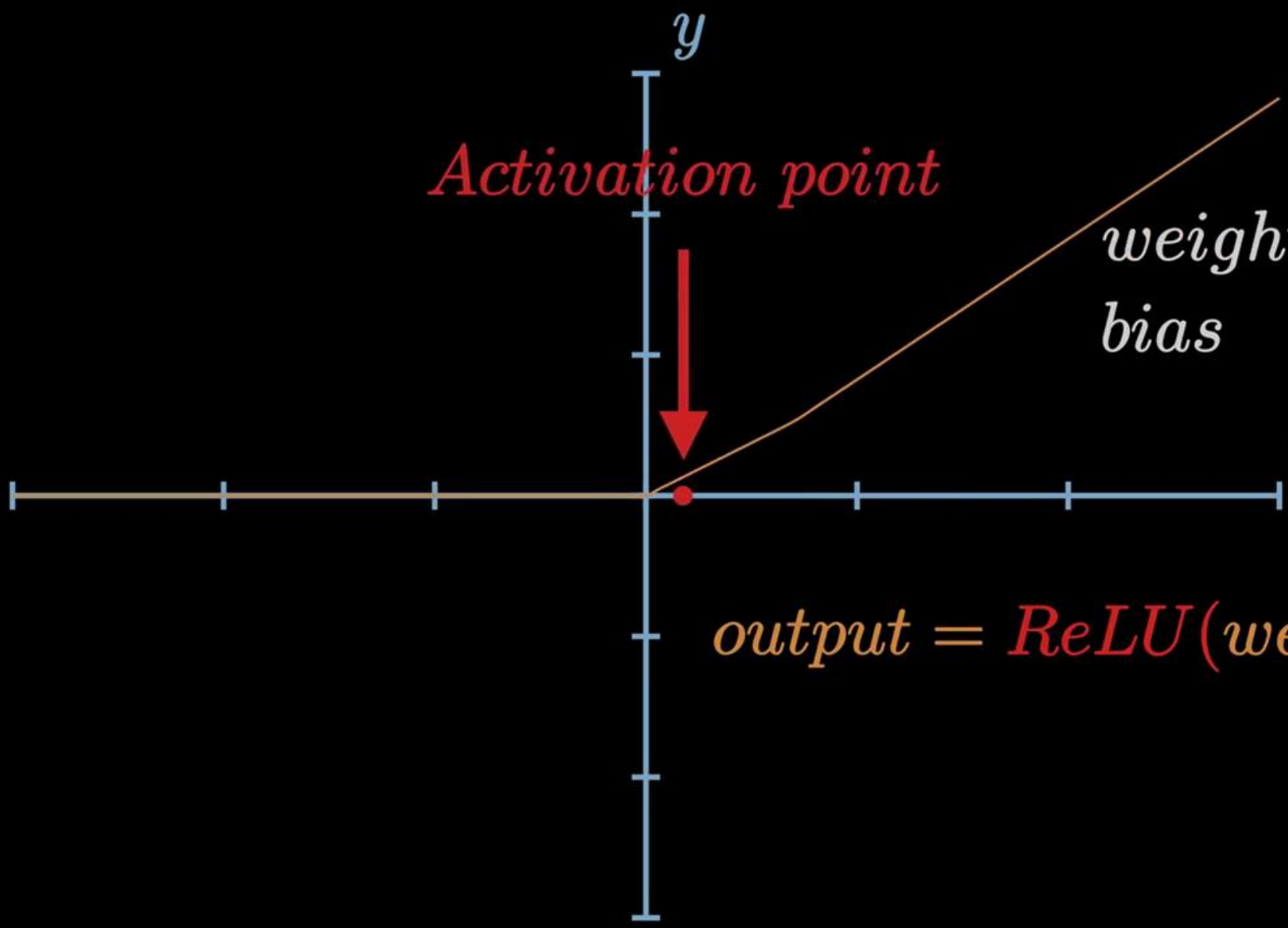
$$output = ReLU(weight \cdot input + bias)$$





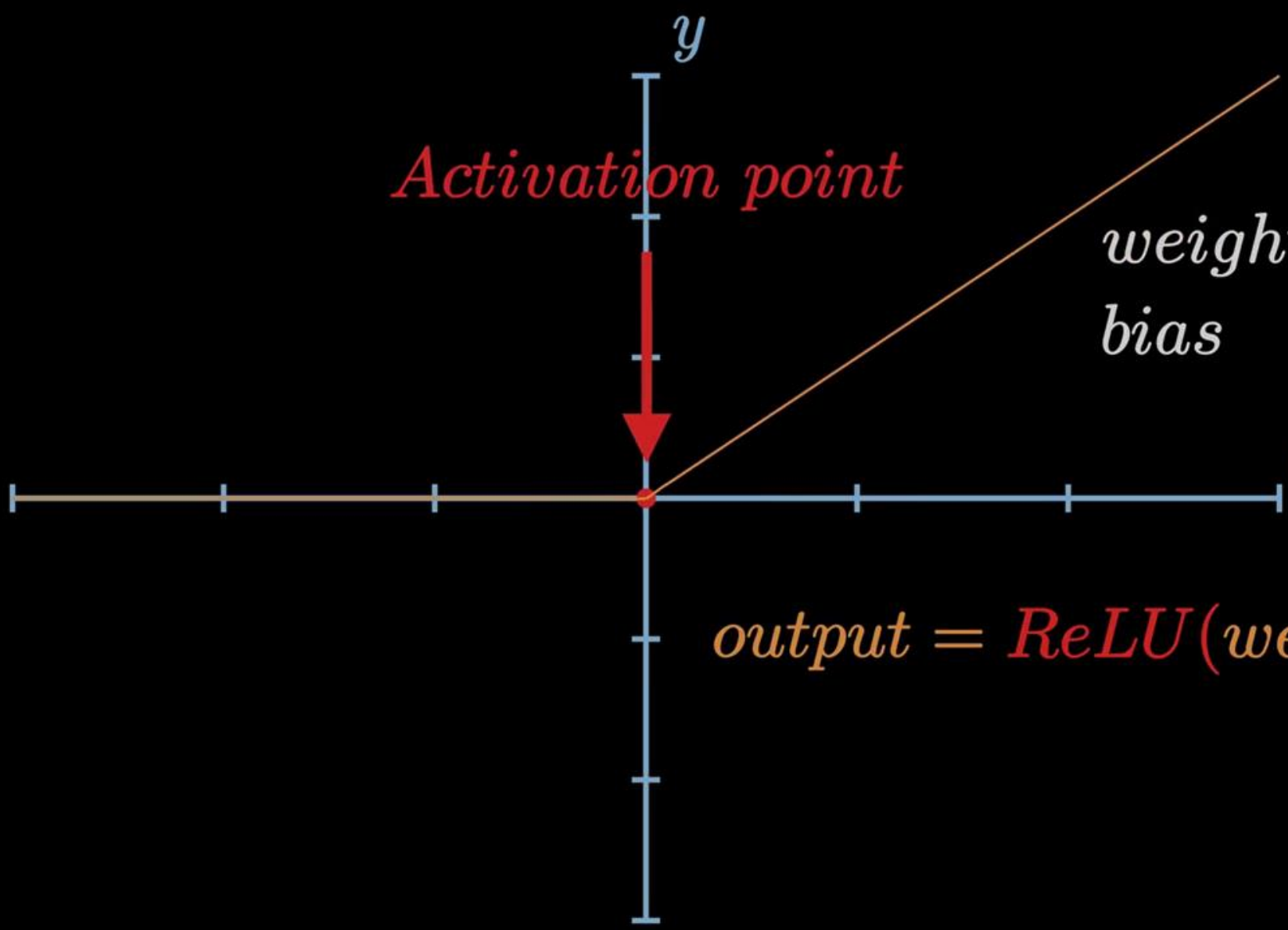
$$weight = 1.00$$
$$bias = -0.67$$

$$output = ReLU(weight \cdot input + bias)$$



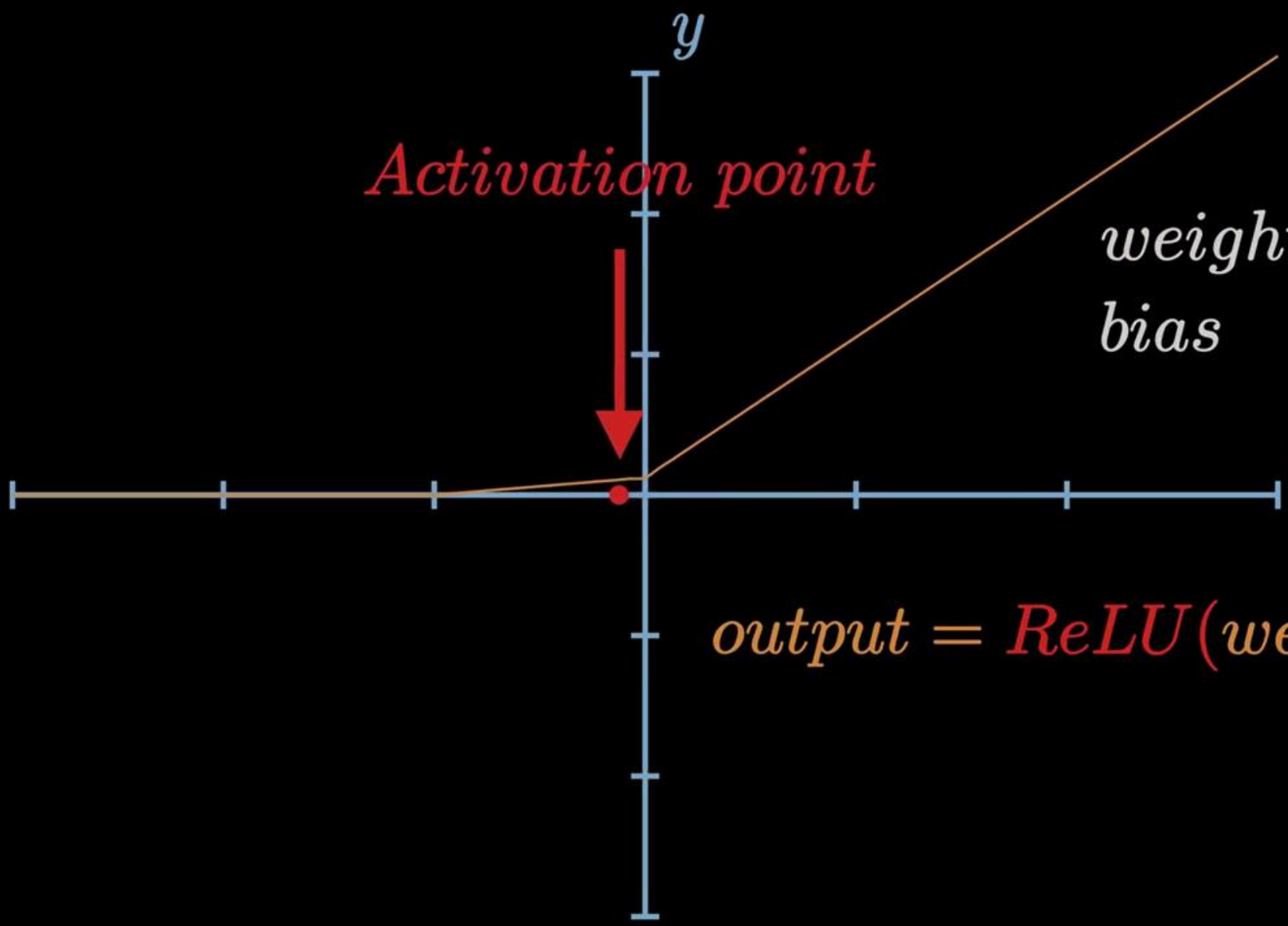
$$weight = 1.00$$
$$bias = -0.17$$

$$output = ReLU(weight \cdot input + bias)$$



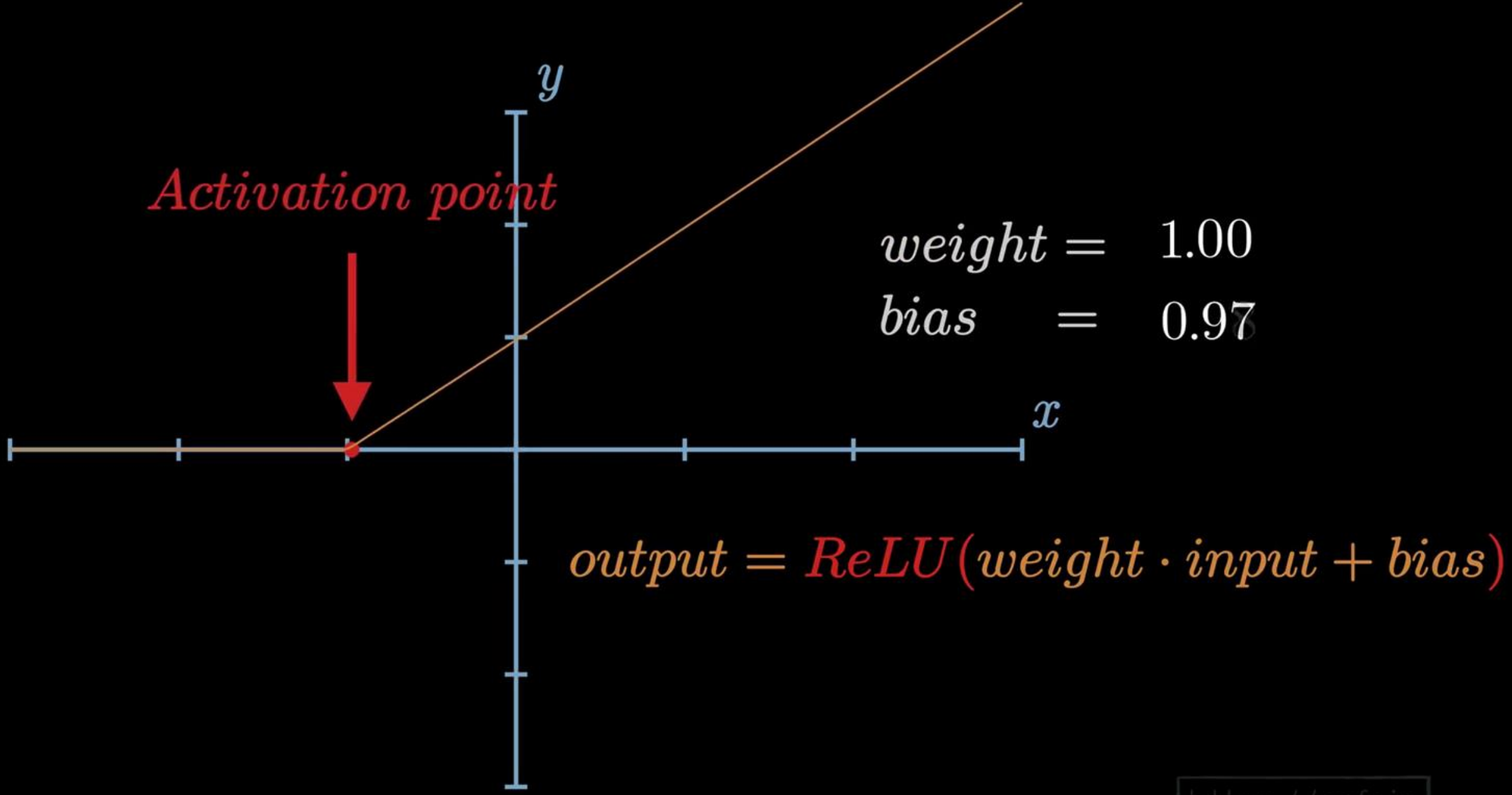
$$weight = 1.00$$
$$bias = 0.00$$

$$output = ReLU(weight \cdot input + bias)$$



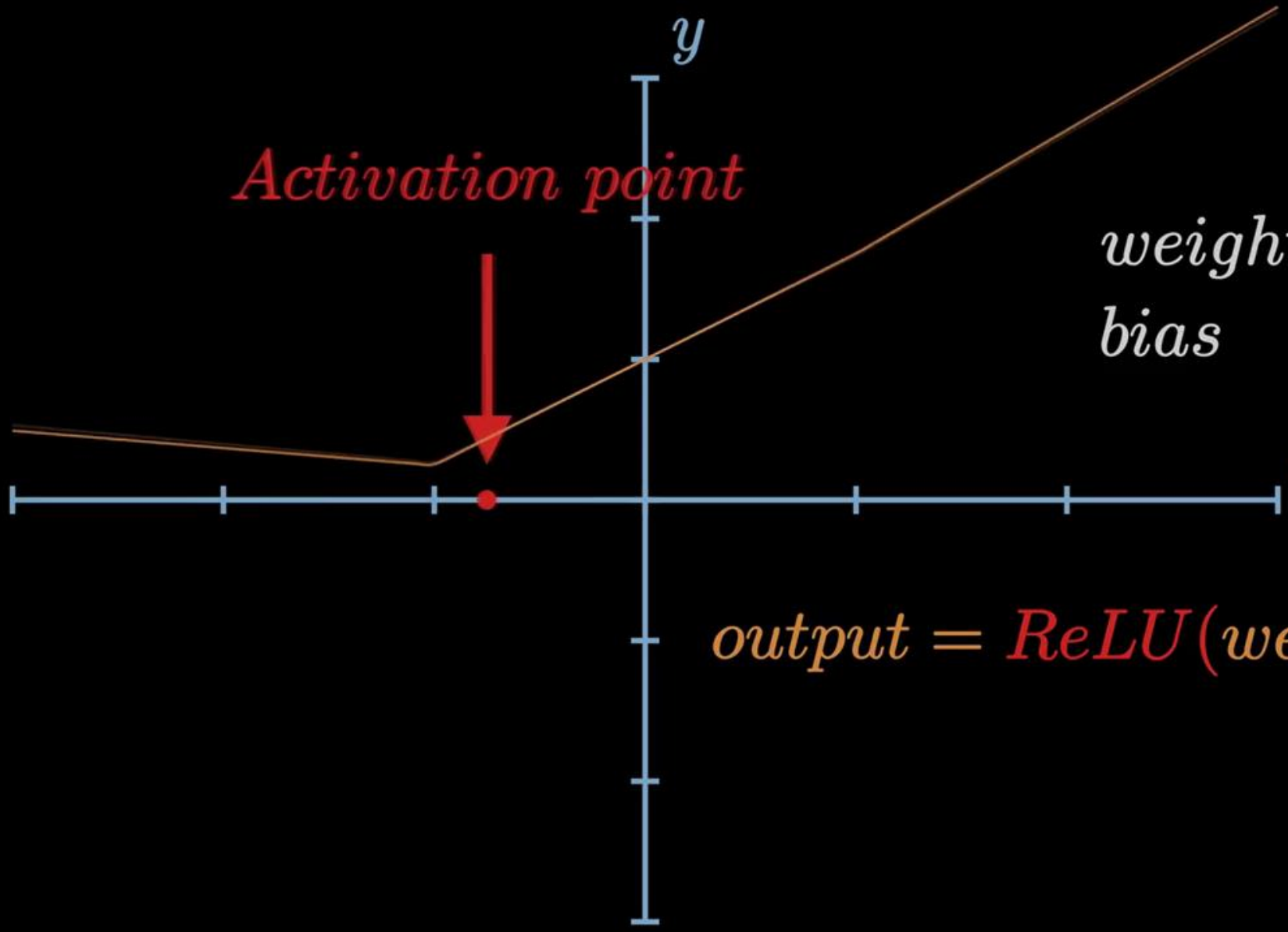
$$weight = 1.00$$
$$bias = 0.12$$

$$output = ReLU(weight \cdot input + bias)$$



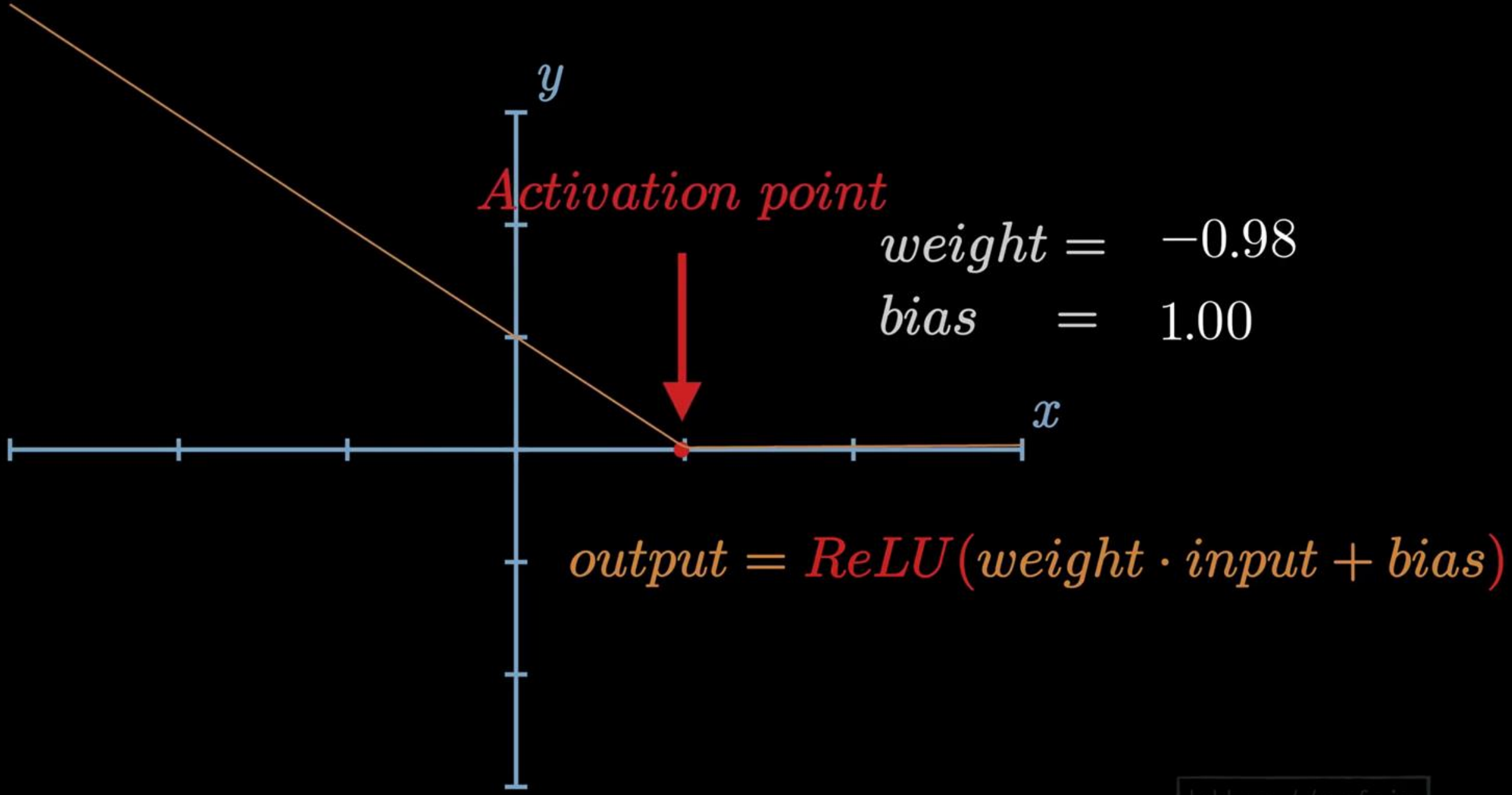
$$weight = 1.00$$
$$bias = 0.97$$

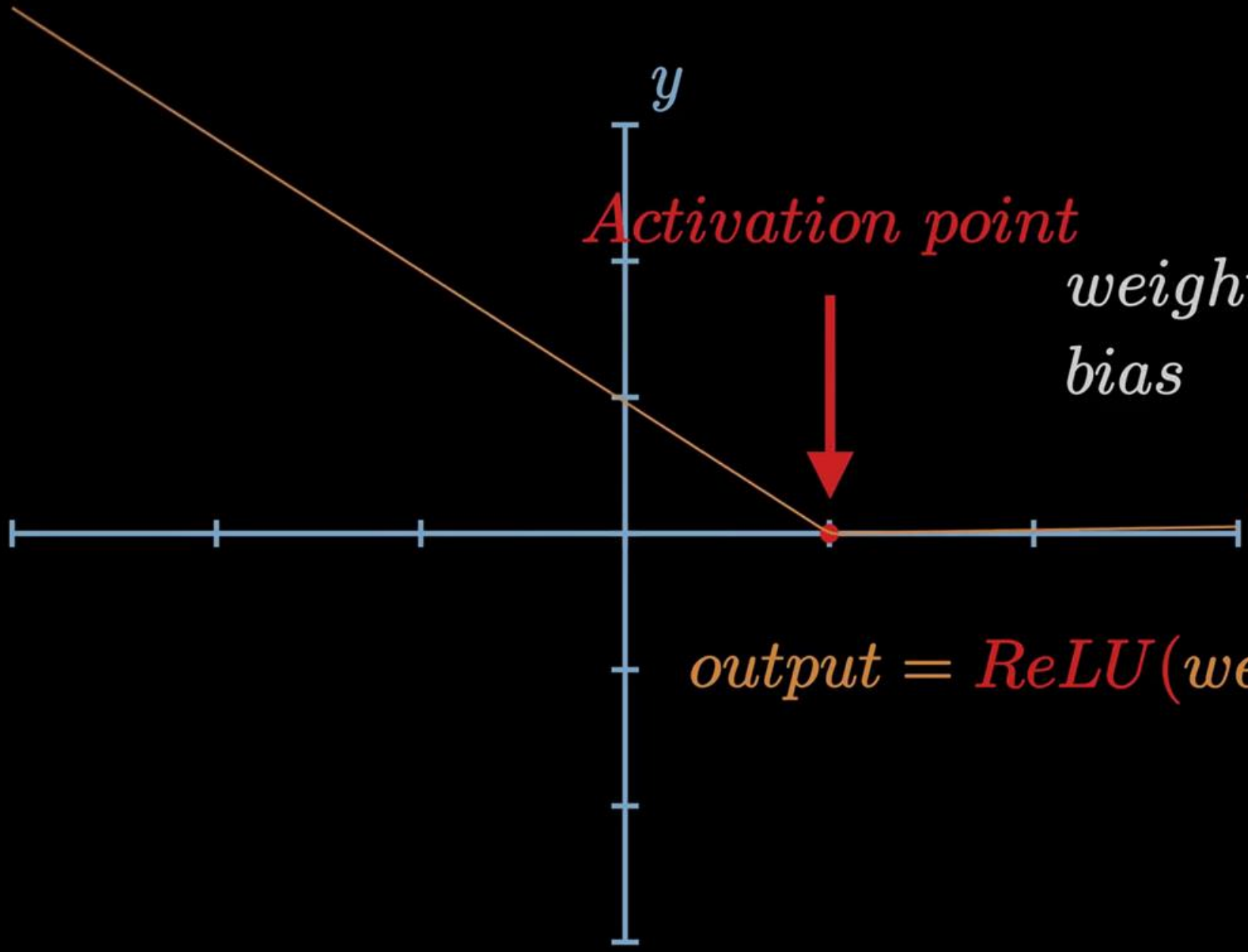
$$output = ReLU(weight \cdot input + bias)$$



$$weight = 0.75$$
$$bias = 1.00$$

$$output = ReLU(weight \cdot input + bias)$$

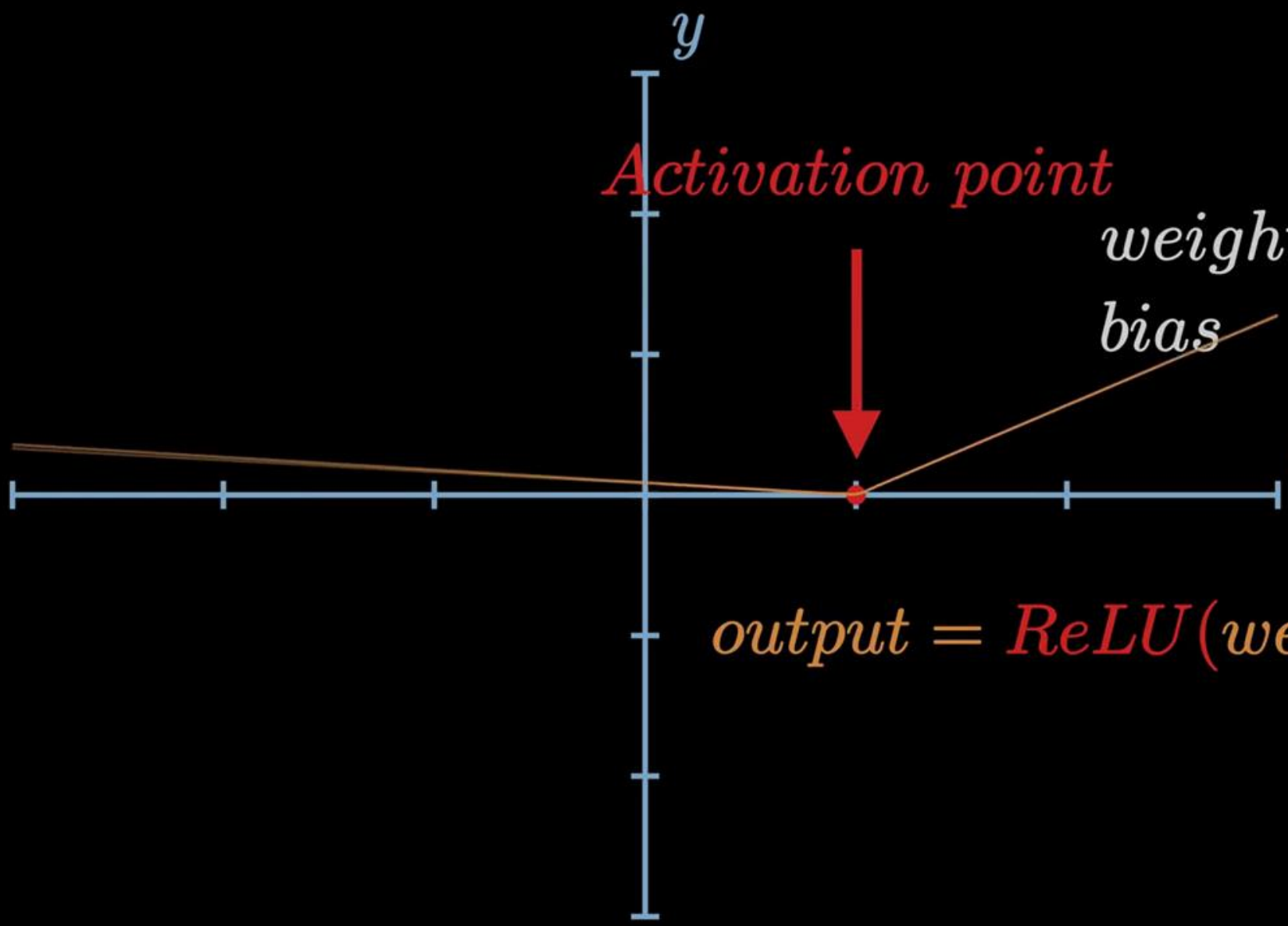




$$weight = -0.65$$

$$bias = 0.94$$

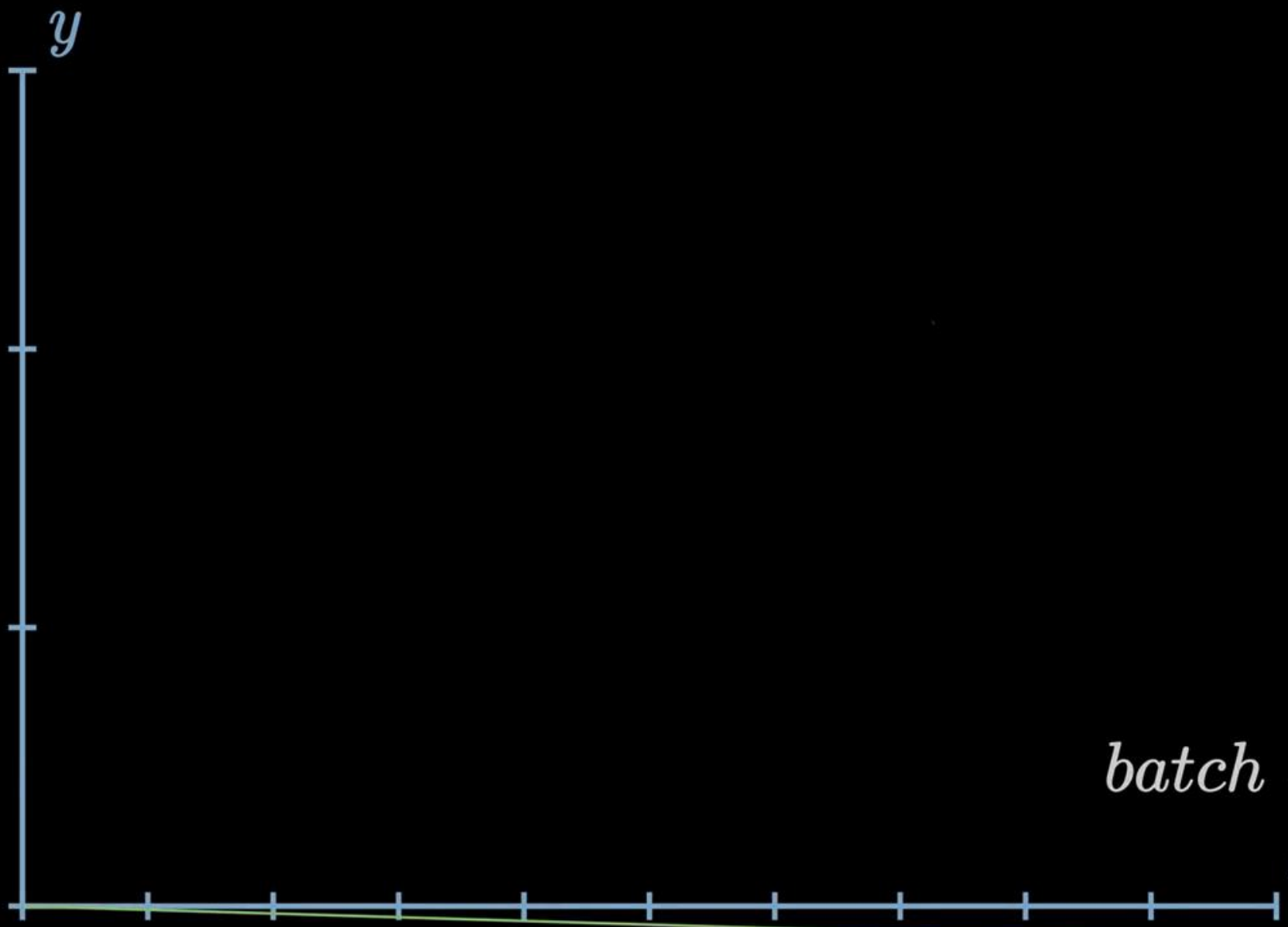
$$output = ReLU(weight \cdot input + bias)$$

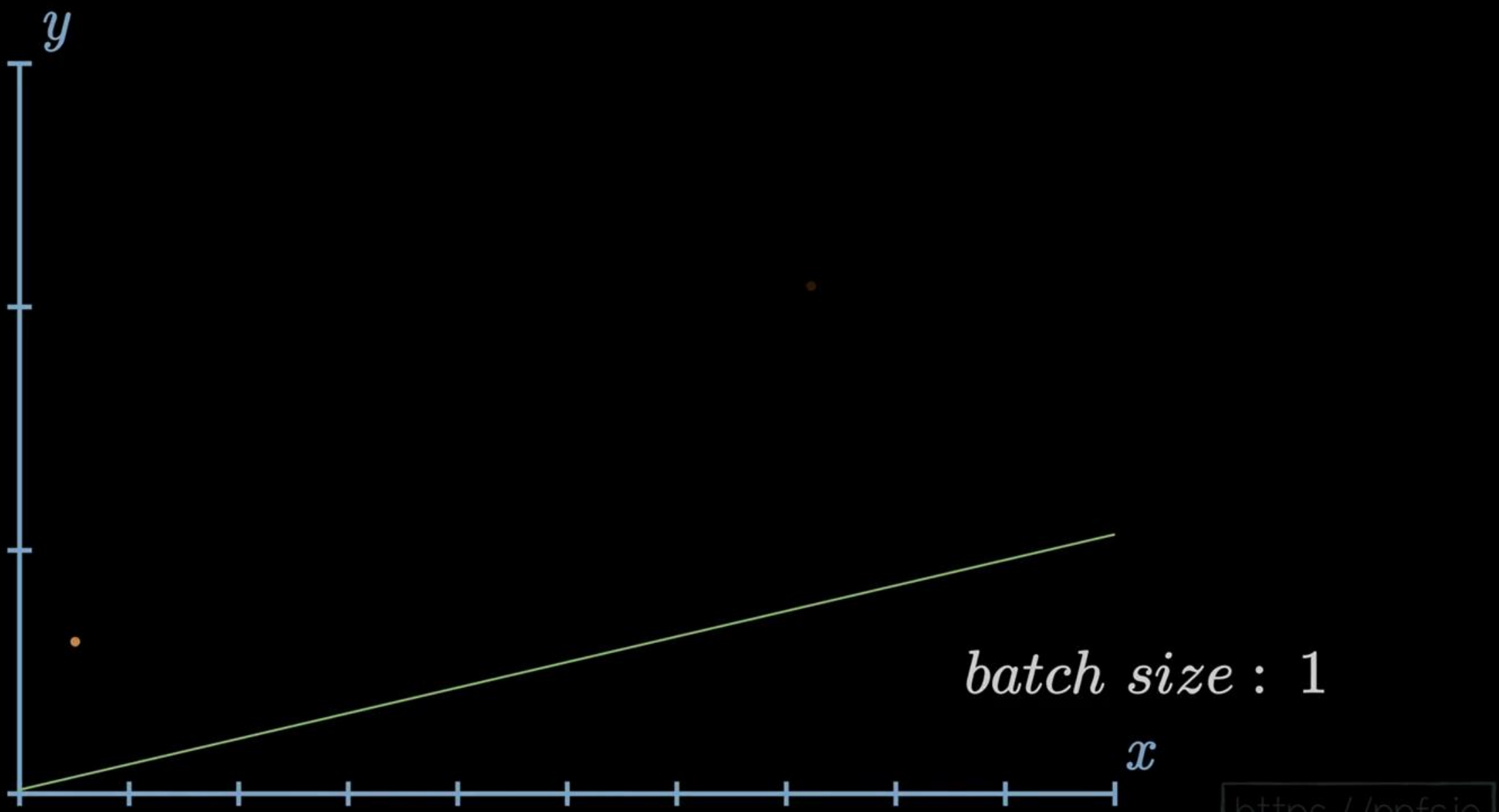


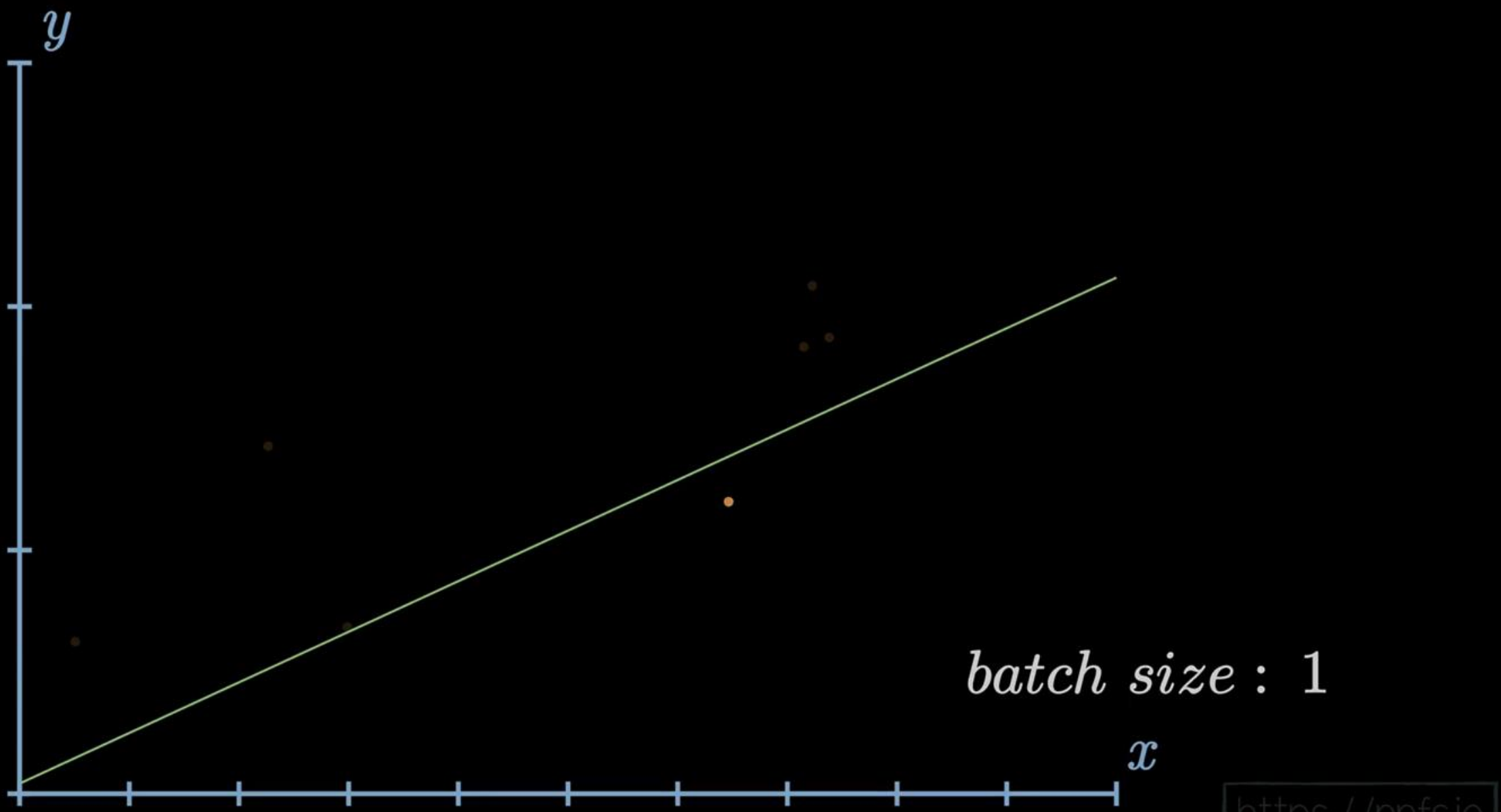
$$weight = 0.58$$

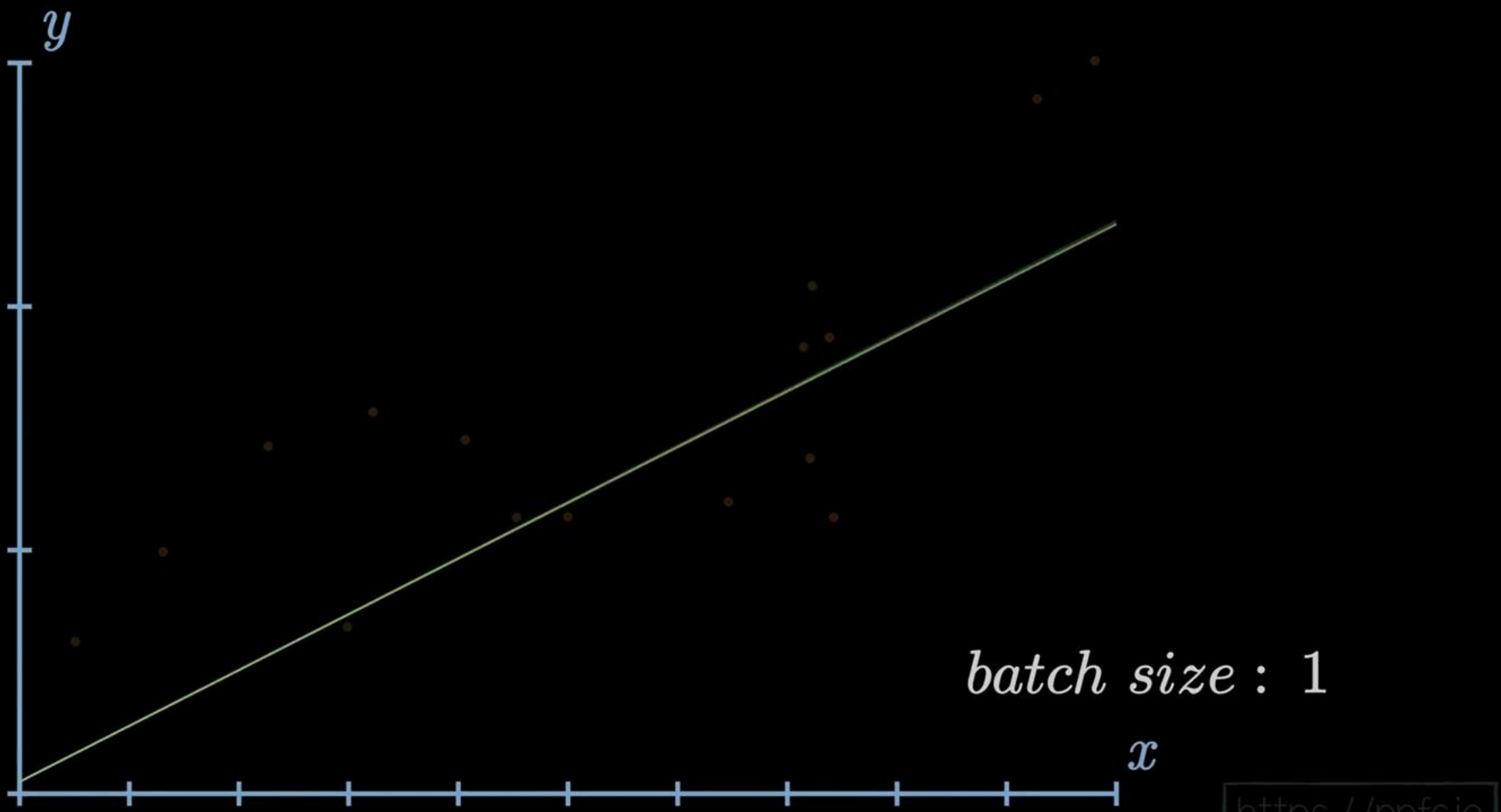
$$bias = -0.55$$

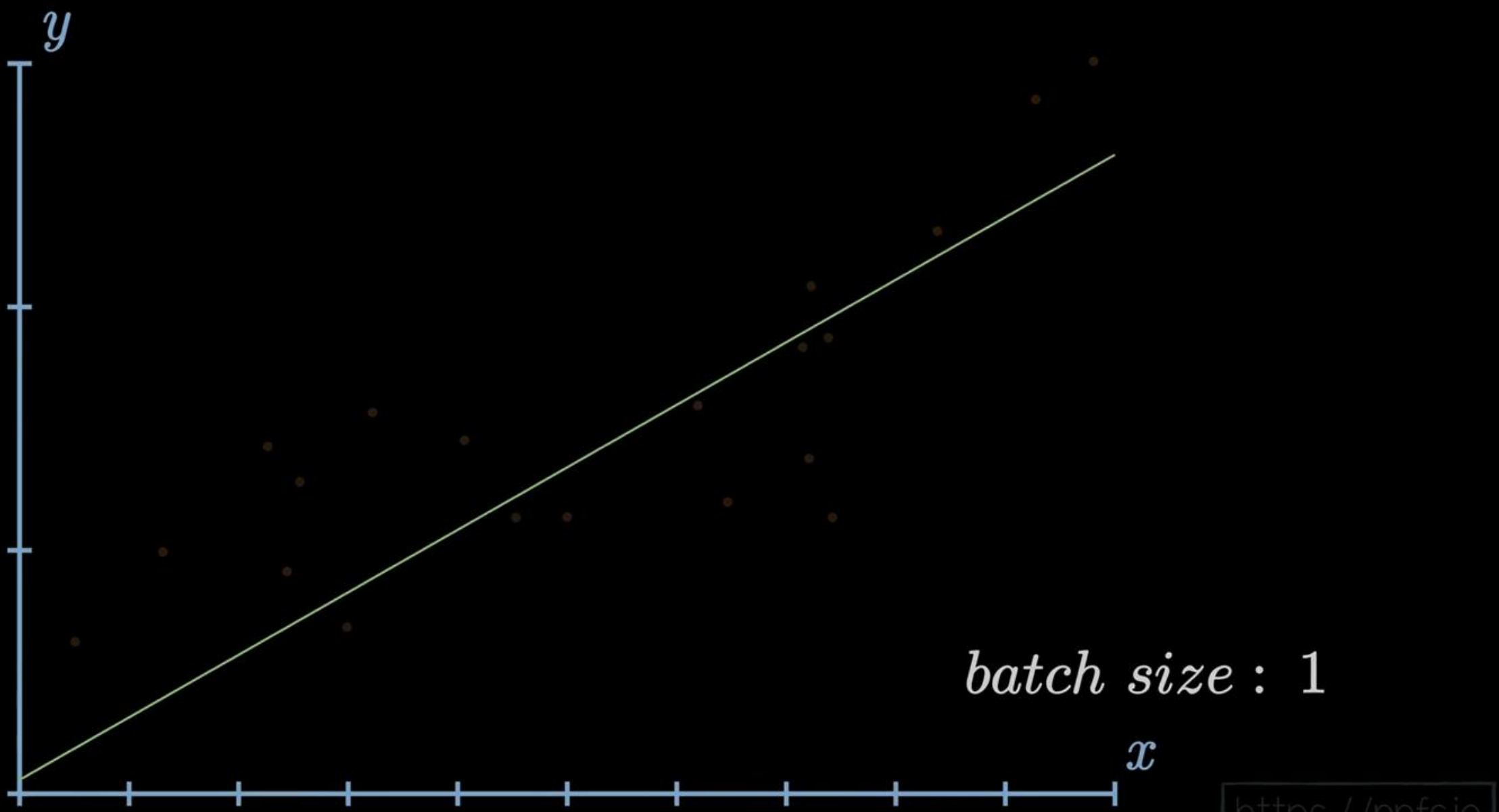
$$output = ReLU(weight \cdot input + bias)$$

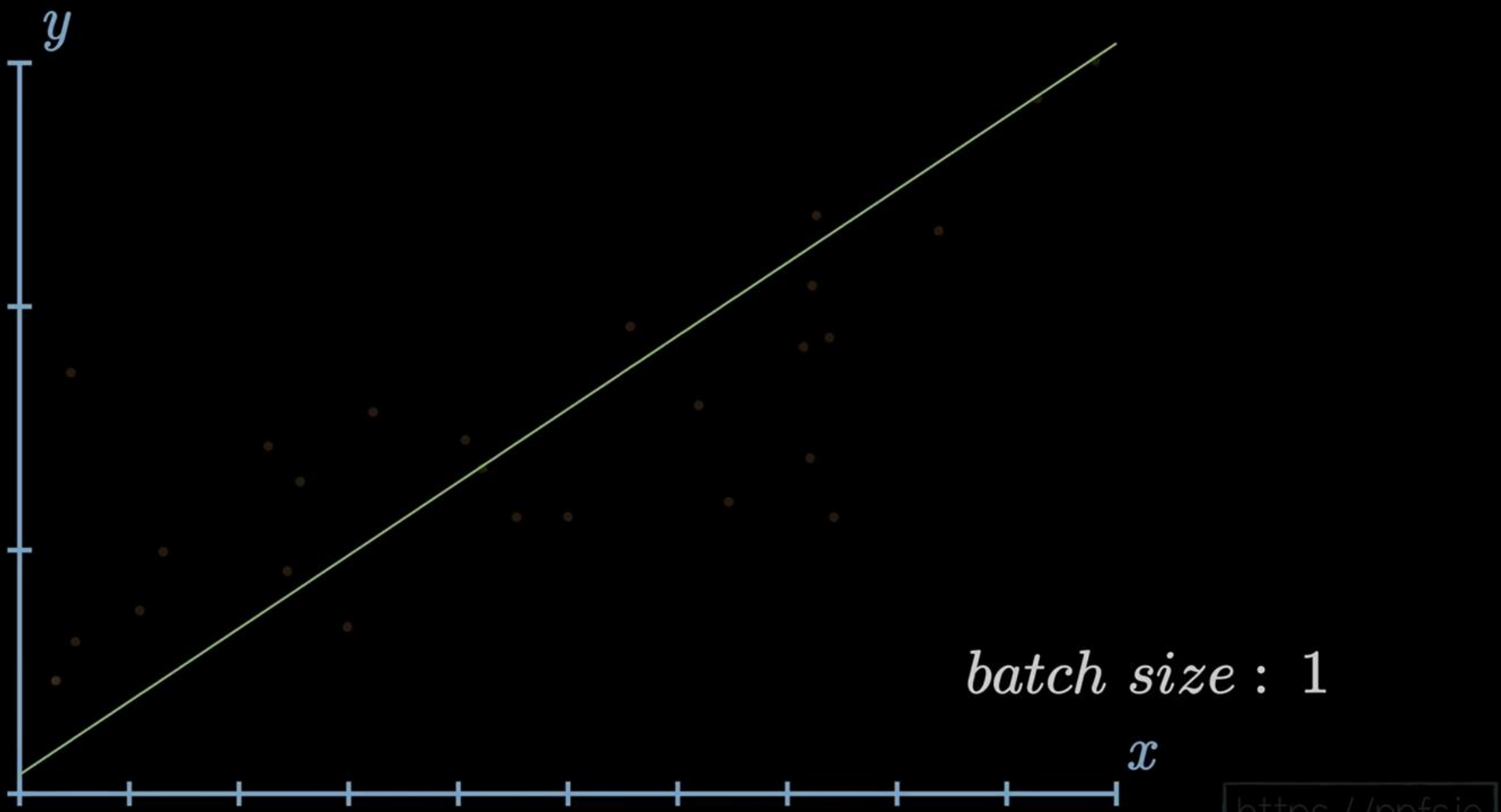


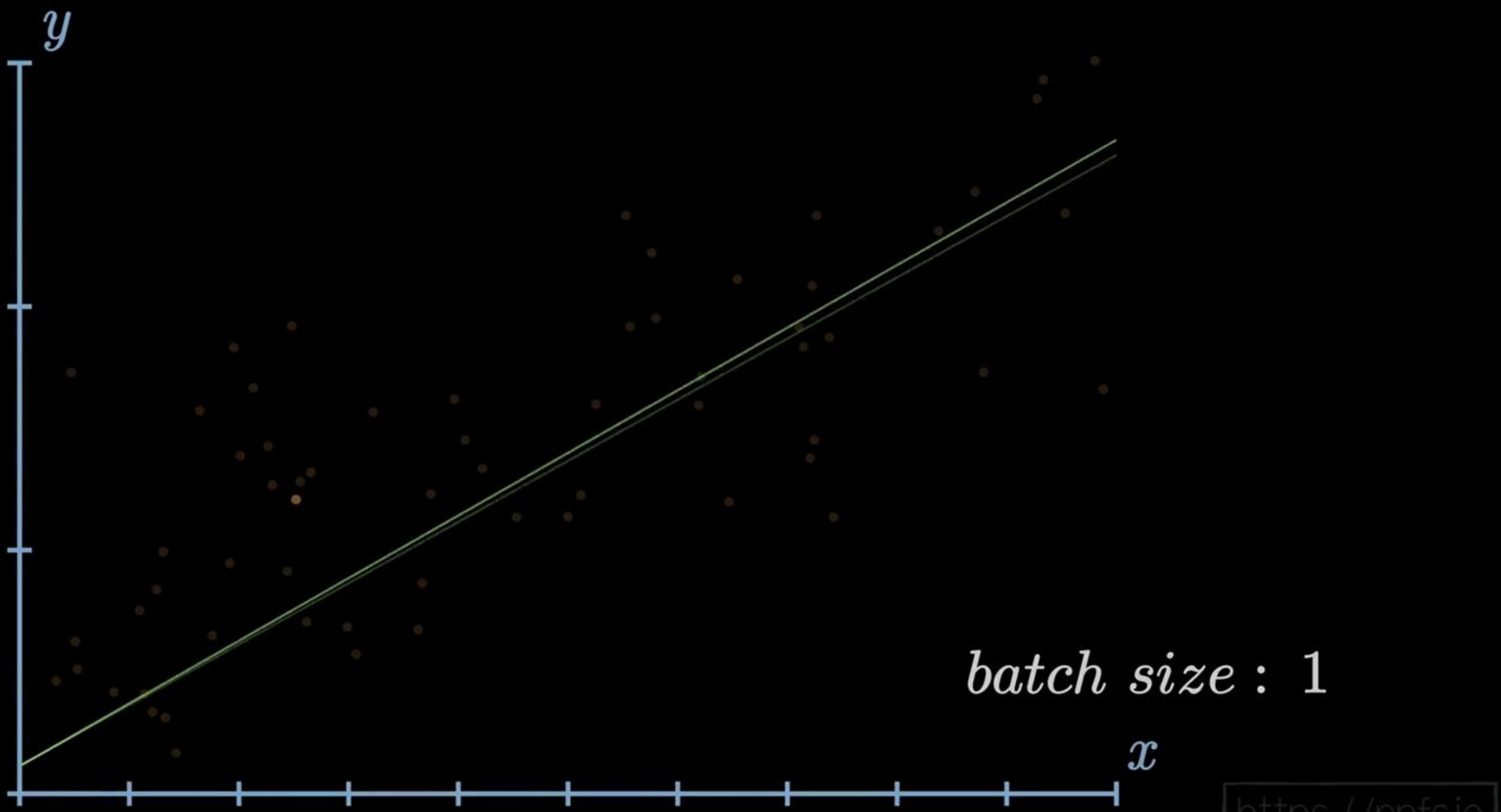

y
batch size : 1
x
https://nnfs.io

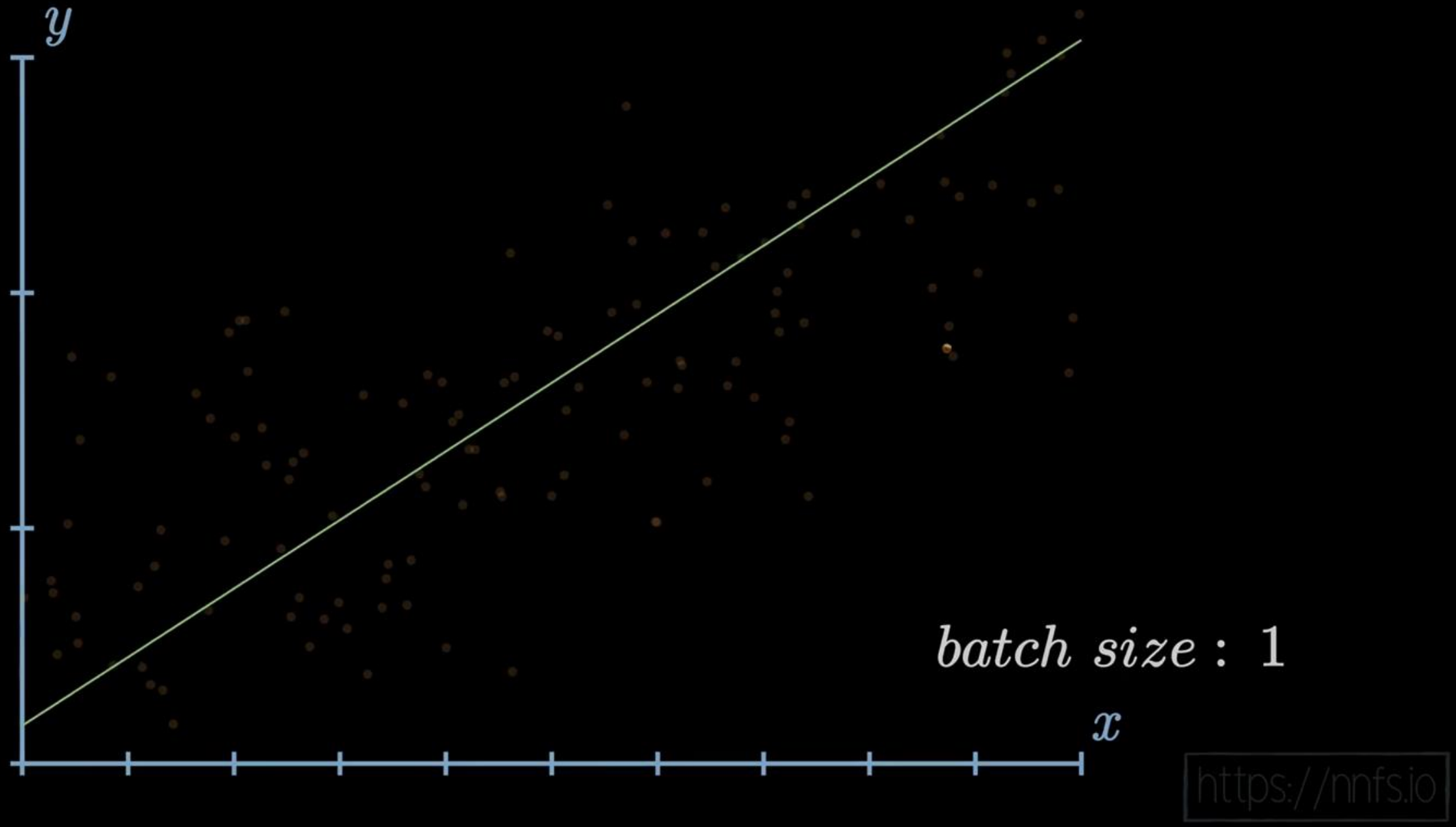

y
batch size : 1
x
https://nnfs.io

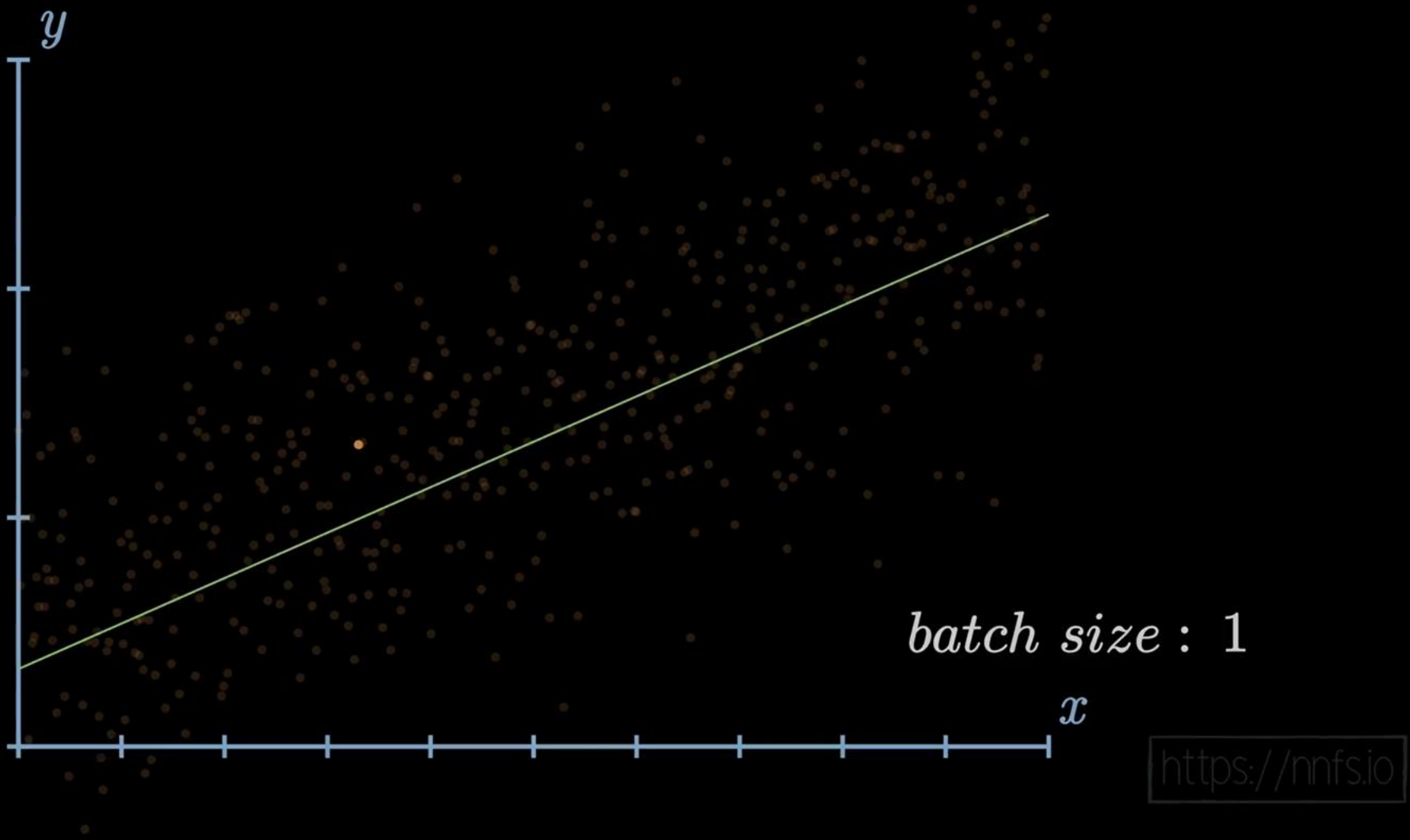

y
x
batch size : 1
https://nnfs.io

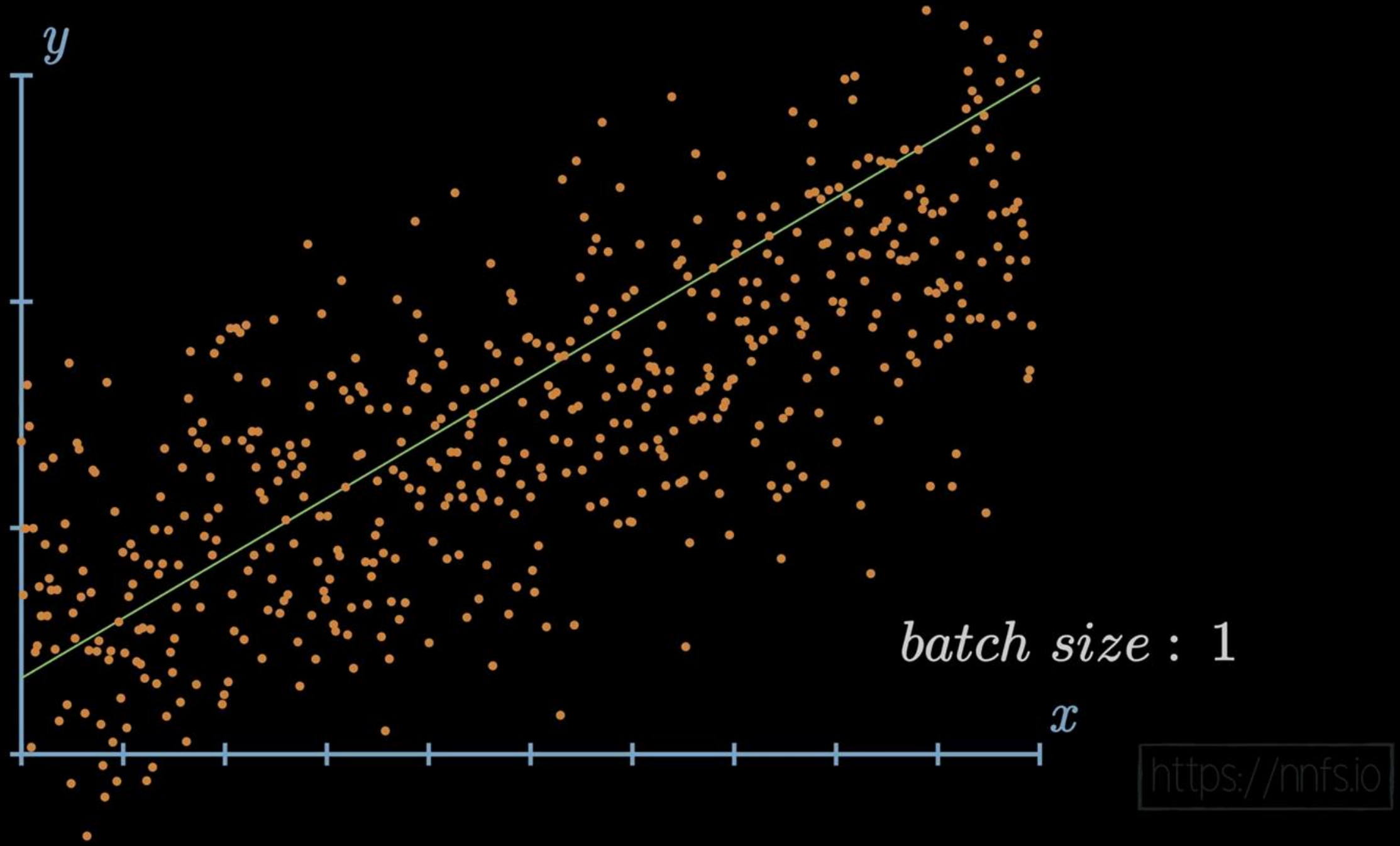

y
batch size : 1
x
https://nnfs.io

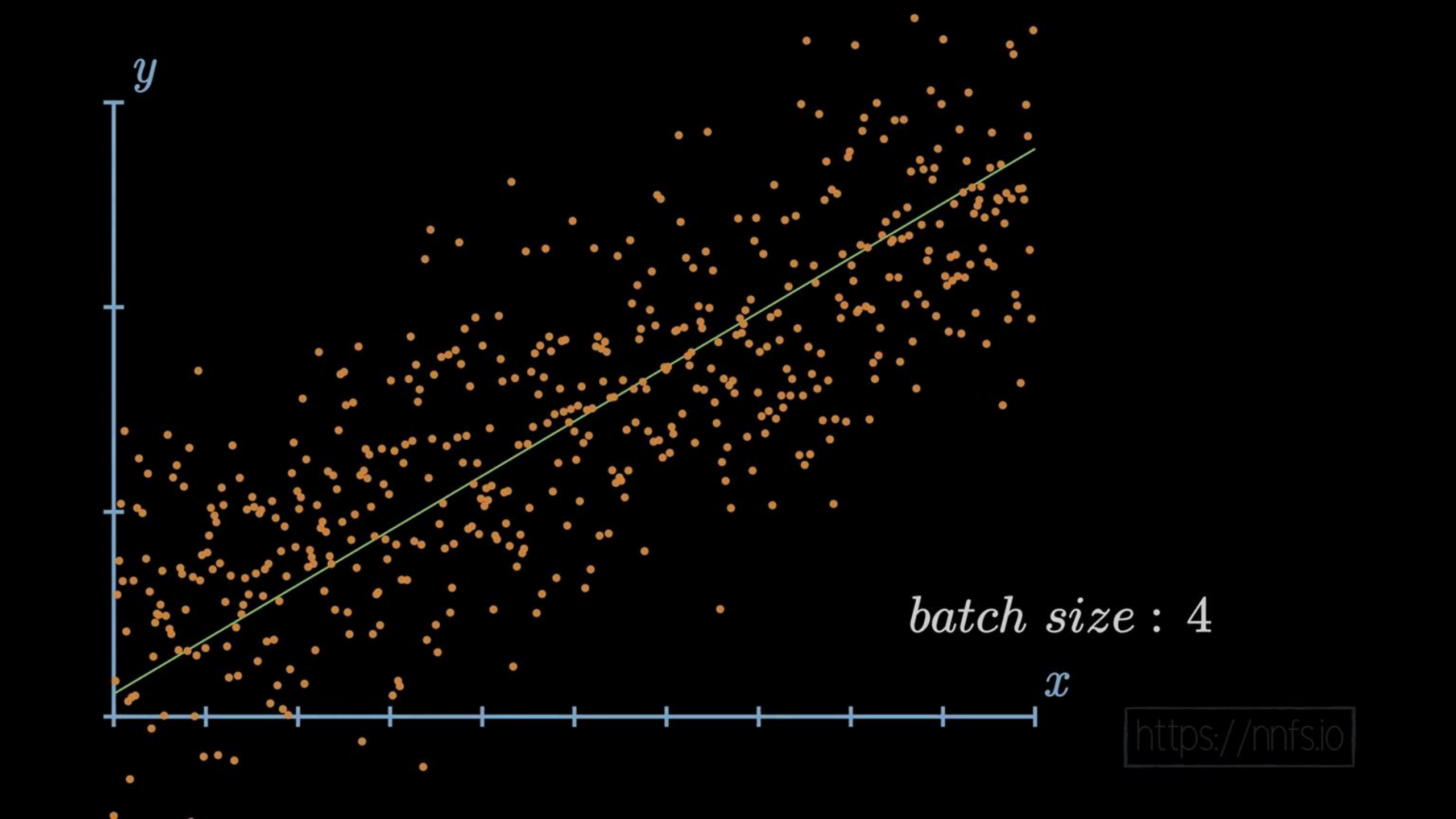

y
batch size : 1
x
https://nnfs.io

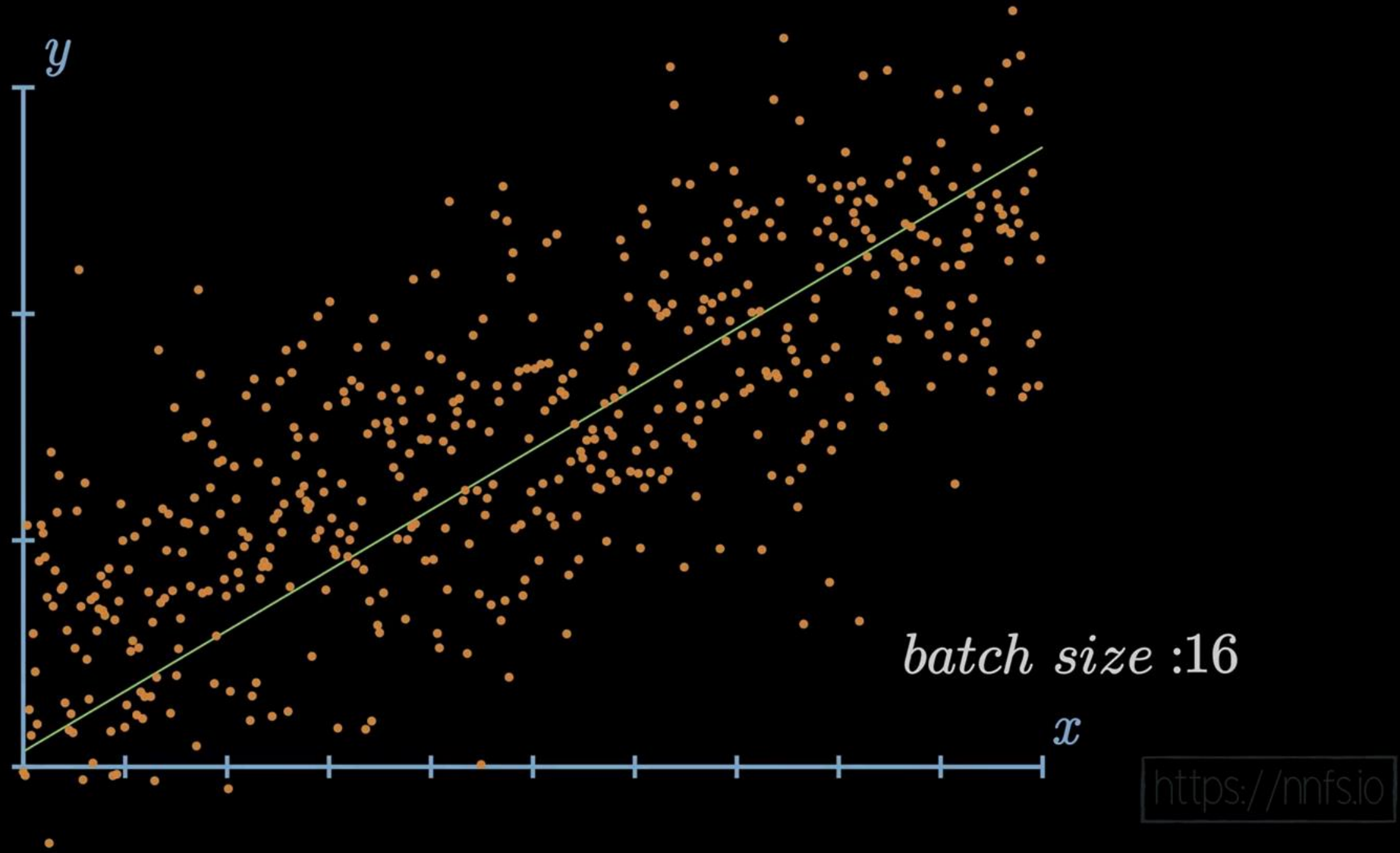

y
x
batch size : 1
https://nnfs.io

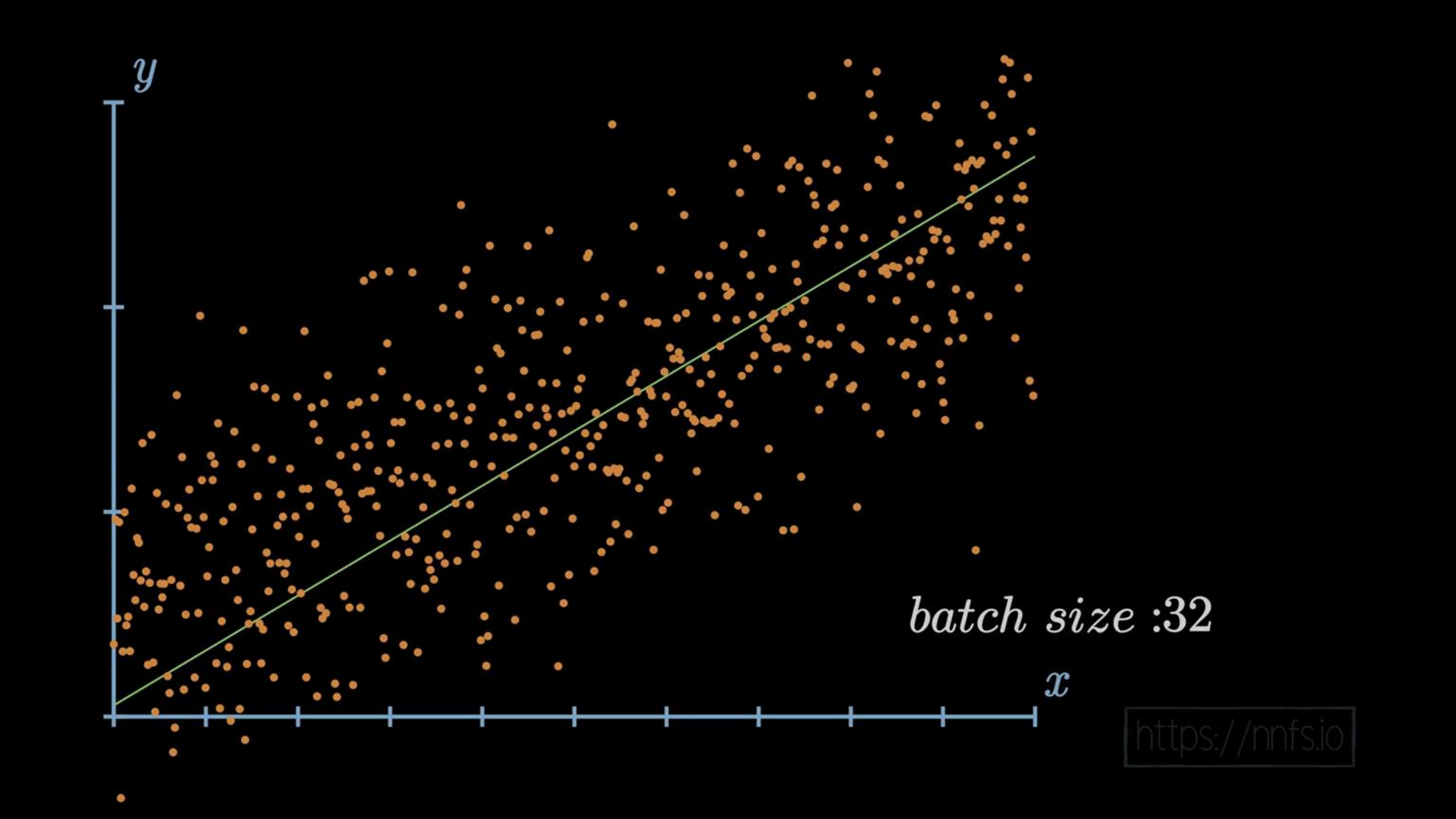

y
x
batch size : 1
https://nnfs.io


y
x
batch size : 1
https://nnfs.io


y
x
batch size : 1
https://nnfs.io


y
x
batch size : 4
https://nnfs.io


y
x
batch size :16
https://nnfs.io


y
x
batch size :32
https://nnfs.io

p4.py

```python
import numpy as np

inputs = [[1, 2, 3, 2.5],
          [2.0, 5.0, -1.0, 2.0],
          [-1.5, 2.7, 3.3, -0.8]]

weights = [[0.2, 0.8, -0.5, 1.0],
           [0.5, -0.91, 0.26, -0.5],
           [-0.26, -0.27, 0.17, 0.87]]

biases = [2, 3, 0.5]

weights2 = [[0.1, -0.14, 0.5],
            [-0.5, 0.12, -0.33],
            [-0.44, 0.73, -0.13]]

biases2 = [-1, 2, -0.5]

layer1_outputs = np.dot(inputs, np.array(weights).T) + biases
layer2_outputs = np.dot(layer1_outputs, np.array(weights2).T) + biases2

print(layer2_outputs)
```

```
[[ 0.5031  -1.04185 -2.03875]
 [ 0.2434  -2.7332  -5.7633 ]
 [-0.99314  1.41254 -0.35655]]
[Finished in 0.2s]
```

Line 12, Column 1 Spaces: 4 Python

p4.py

```python
import numpy as np

X = [[1, 2, 3, 2.5],
     [2.0, 5.0, -1.0, 2.0],
     [-1.5, 2.7, 3.3, -0.8]]

class Layer_Dense:
    def __init__(self):
        pass
    def forward(self):
        pass
```

```
[[ 0.5031  -1.04185 -2.03875]
 [ 0.2434  -2.7332  -5.7633 ]
 [-0.99314  1.41254 -0.35655]]
[Finished in 0.2s]
```

p4.py

```python
import numpy as np

np.random.seed(0)

X = [[1, 2, 3, 2.5],
     [2.0, 5.0, -1.0, 2.0],
     [-1.5, 2.7, 3.3, -0.8]]


class Layer_Dense:
    def __init__(self, n_inputs, n_neurons):
        self.weights = 0.10 * np.random.randn(n_inputs, n_neurons)
        self.biases = np.zeros((1, n_neurons))
    def forward(self, inputs):
        self.output = np.dot(inputs, self.weights) + self.biases

layer1 = Layer_Dense(4,5)
layer2 = Layer_Dense(5,2)
```

```
[[ 0.17640523  0.04001572  0.0978738 ]
 [ 0.22408932  0.1867558  -0.09772779]
 [ 0.09500884 -0.01513572 -0.01032189]
 [ 0.04105985  0.01440436  0.14542735]]
[Finished in 0.2s]
```

Line 18, Column 25 Spaces: 4 Python

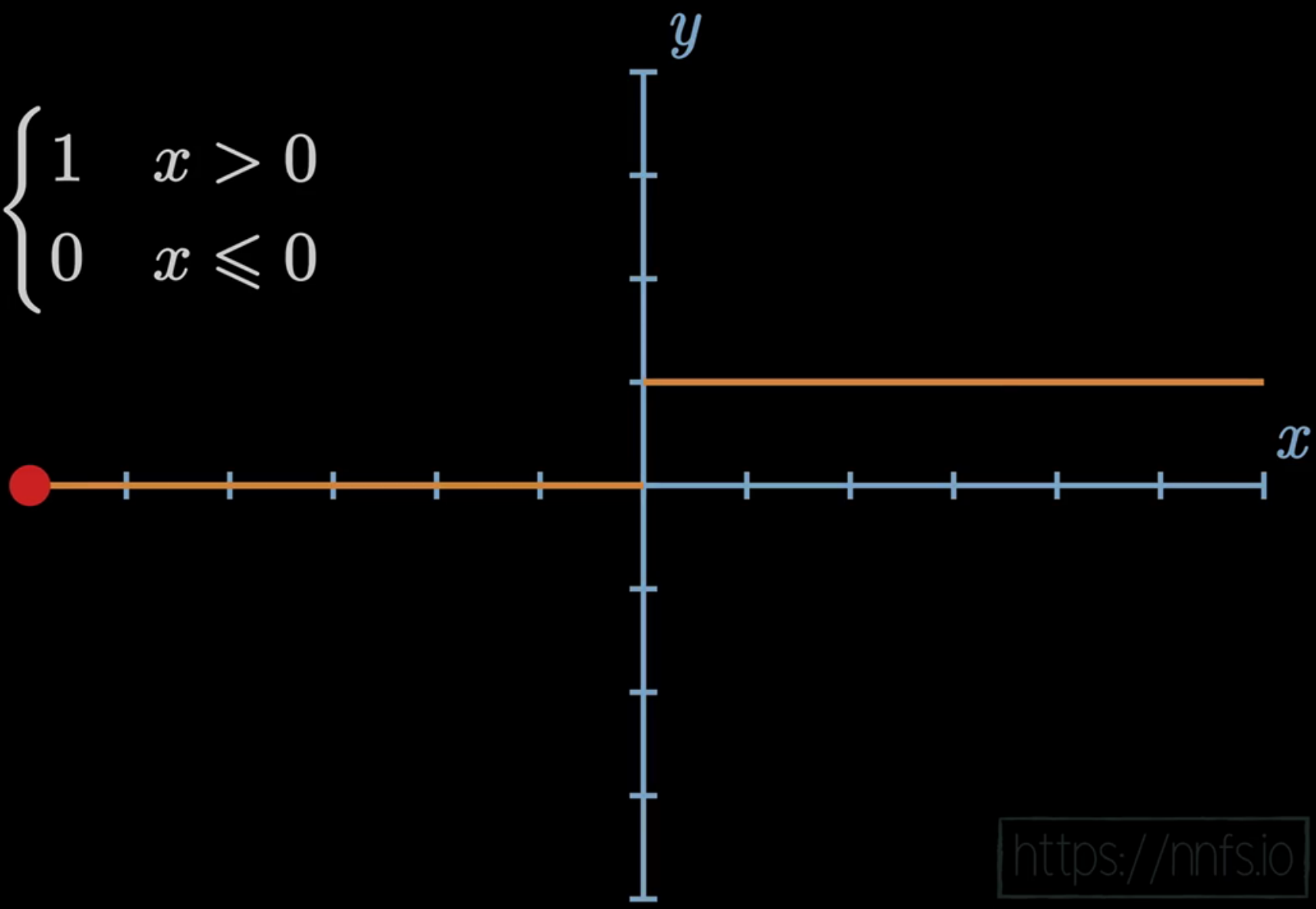
p4.py p3.py

```python
class Layer_Dense:
    def __init__(self, n_inputs, n_neurons):
        self.weights = 0.10 * np.random.randn(n_inputs, n_neurons)
        self.biases = np.zeros((1, n_neurons))
    def forward(self, inputs):
        self.output = np.dot(inputs, self.weights) + self.biases

layer1 = Layer_Dense(4,5)
layer2 = Layer_Dense(5,2)

layer1.forward(X)
#print(layer1.output)
layer2.forward(layer1.output)
print(layer2.output)

```

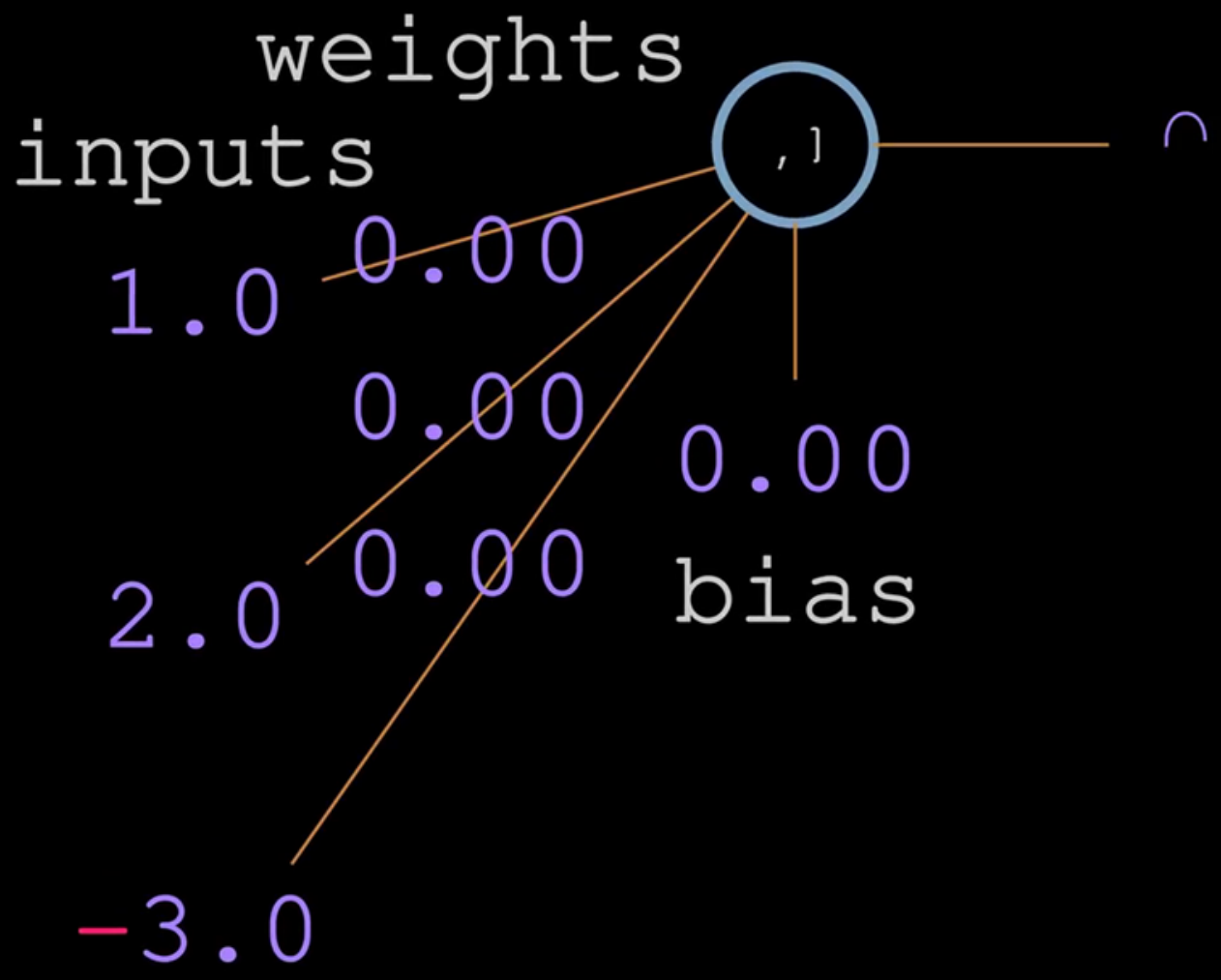
```
[[ 0.148296   -0.08397602]
 [ 0.14100315 -0.01340469]
 [ 0.20124979 -0.07290616]]
[Finished in 0.2s]
```

5 lines, 79 characters selected Spaces: 4 Python

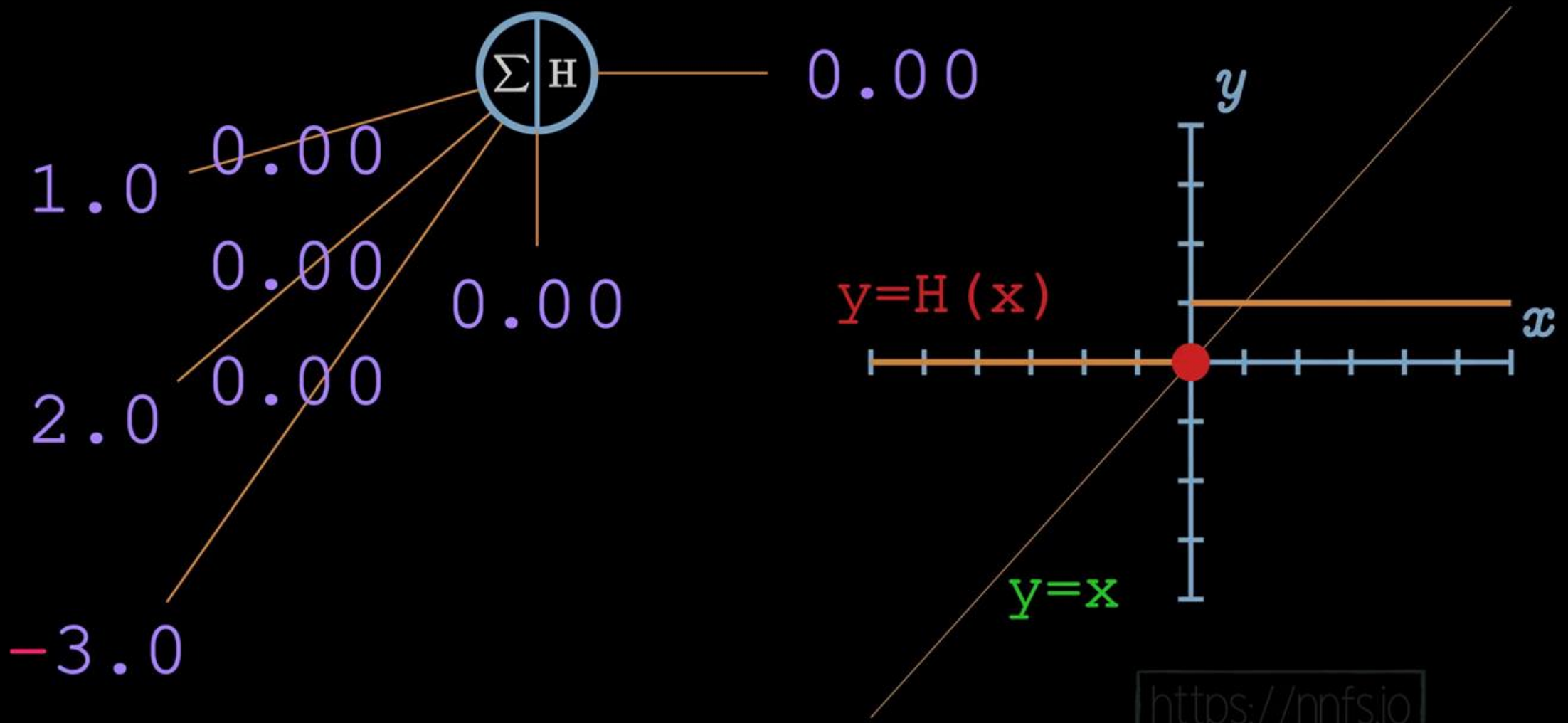
$$y = \begin{cases} 1 & x > 0 \\ 0 & x \leqslant 0 \end{cases}$$

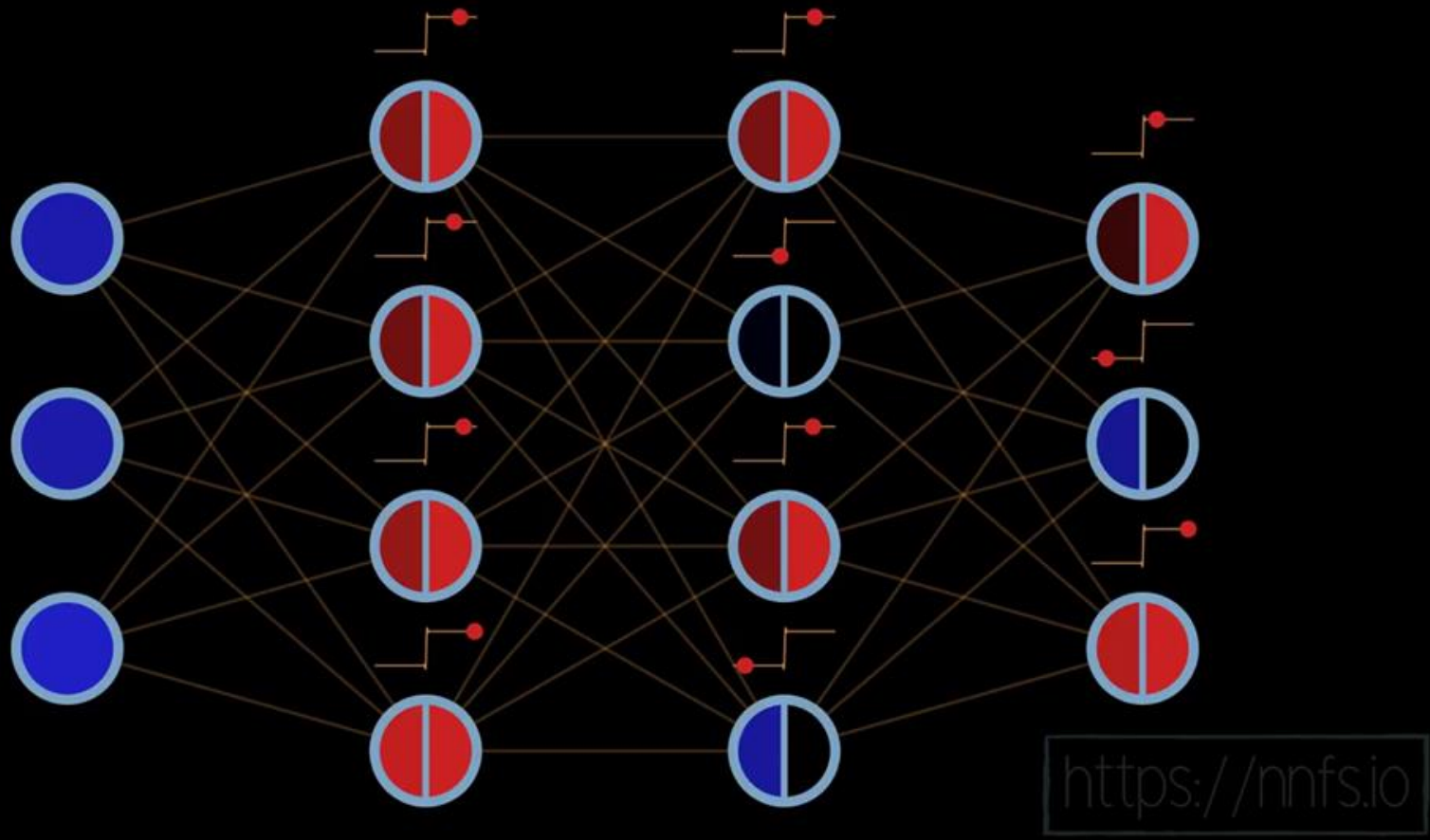


https://nnfs.io

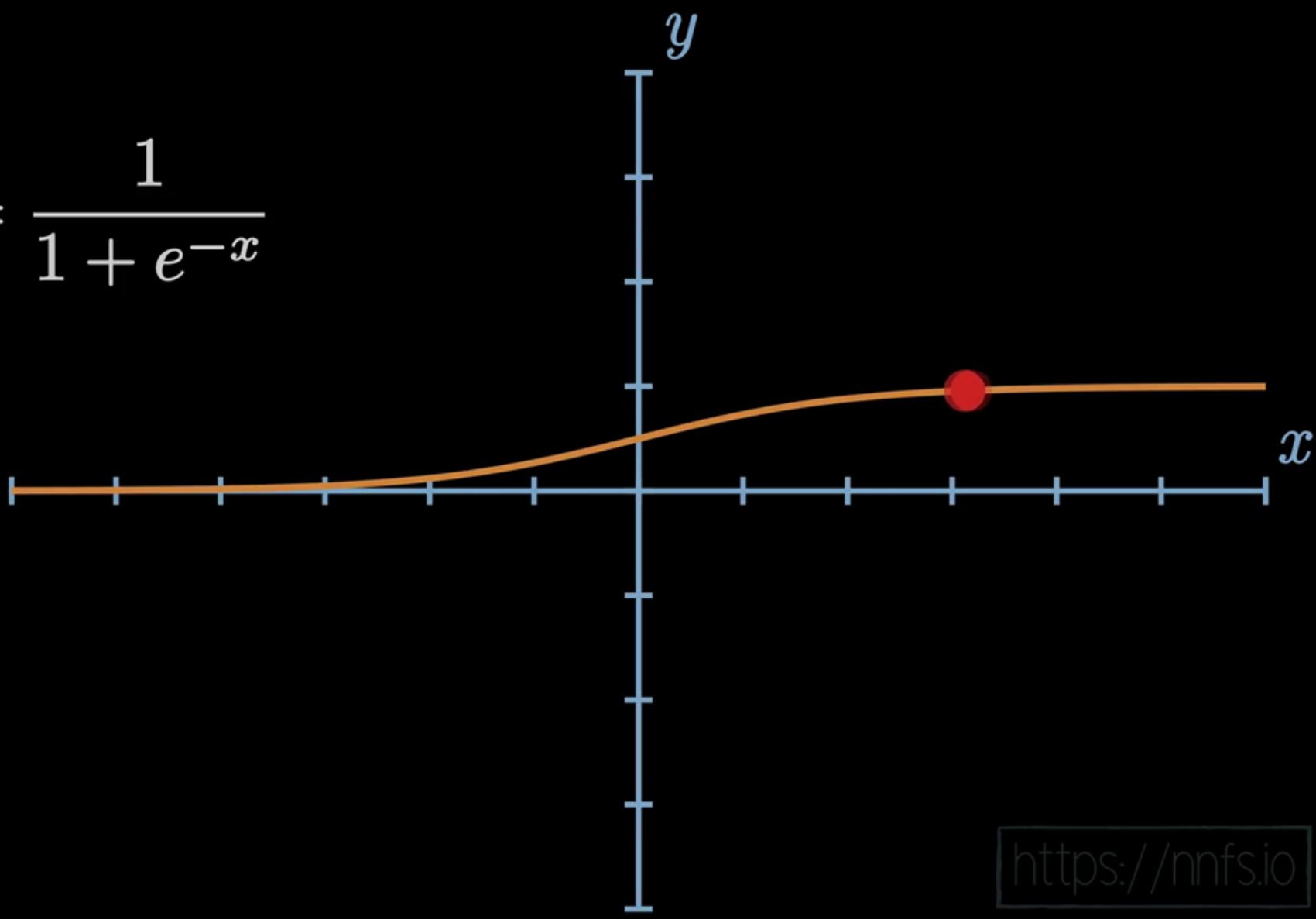


https://nnfs.io

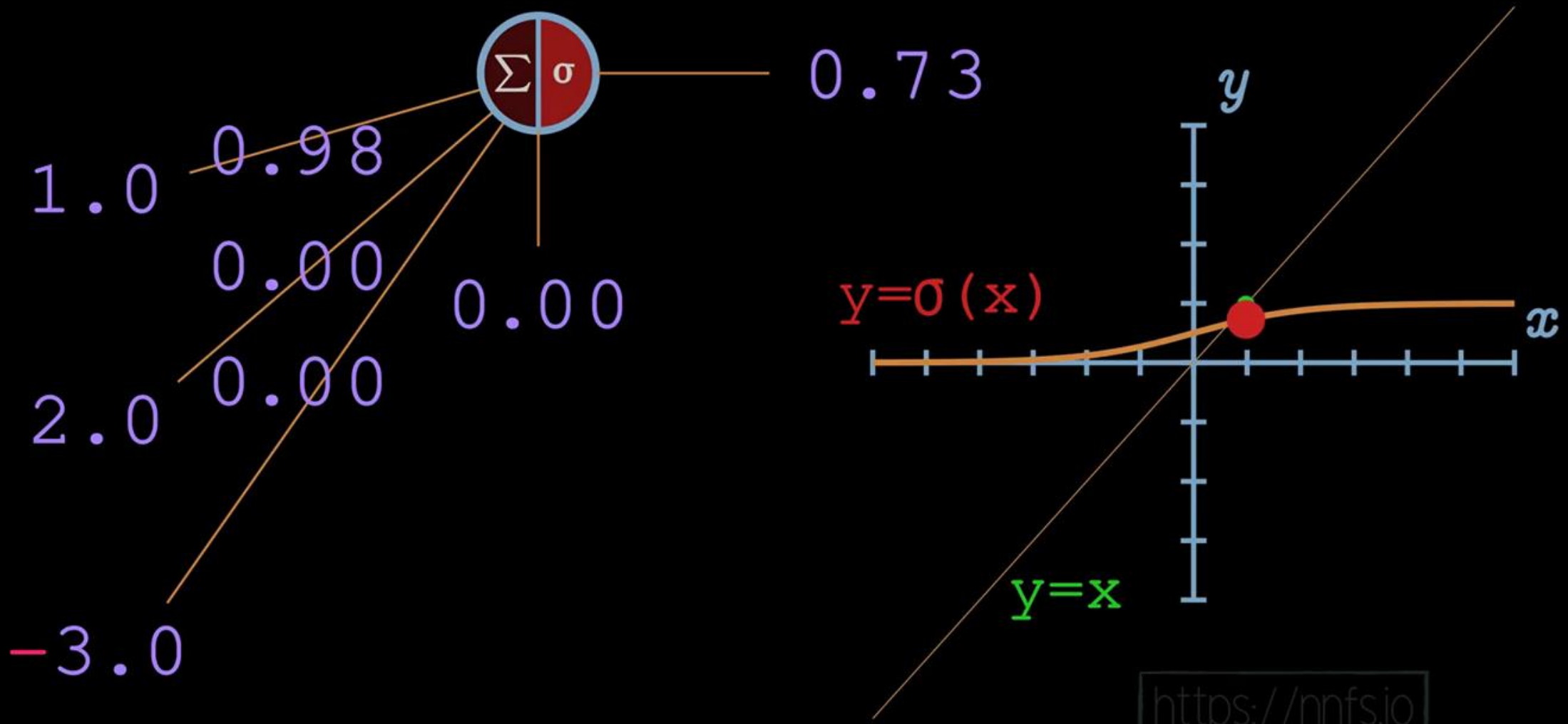

Σ
H
0.00
1.0
0.00
0.00
0.00
2.0
0.00
-3.0
y
x
y=H(x)
y=x
https://nnfs.io

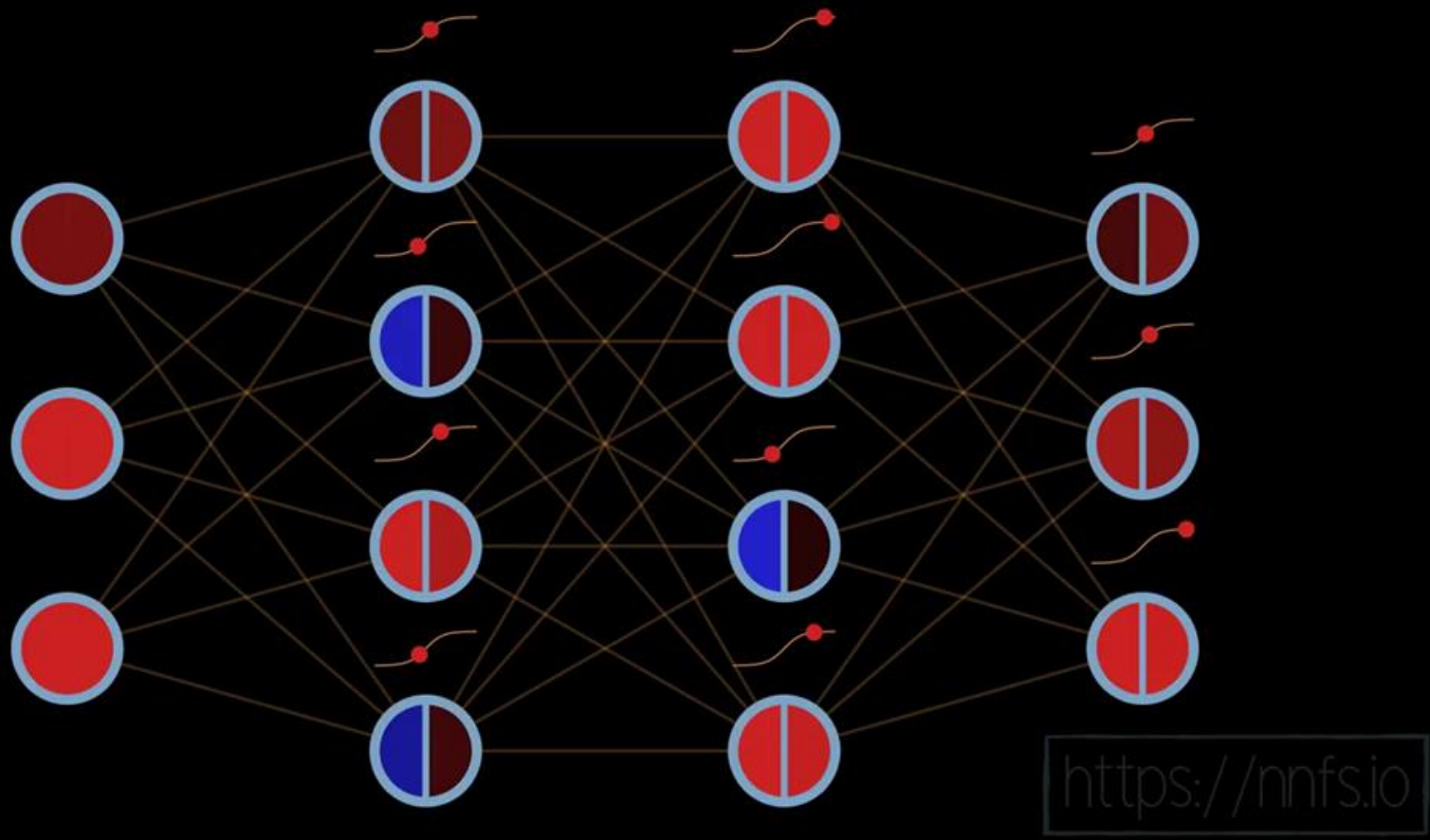

https://nnfs.io

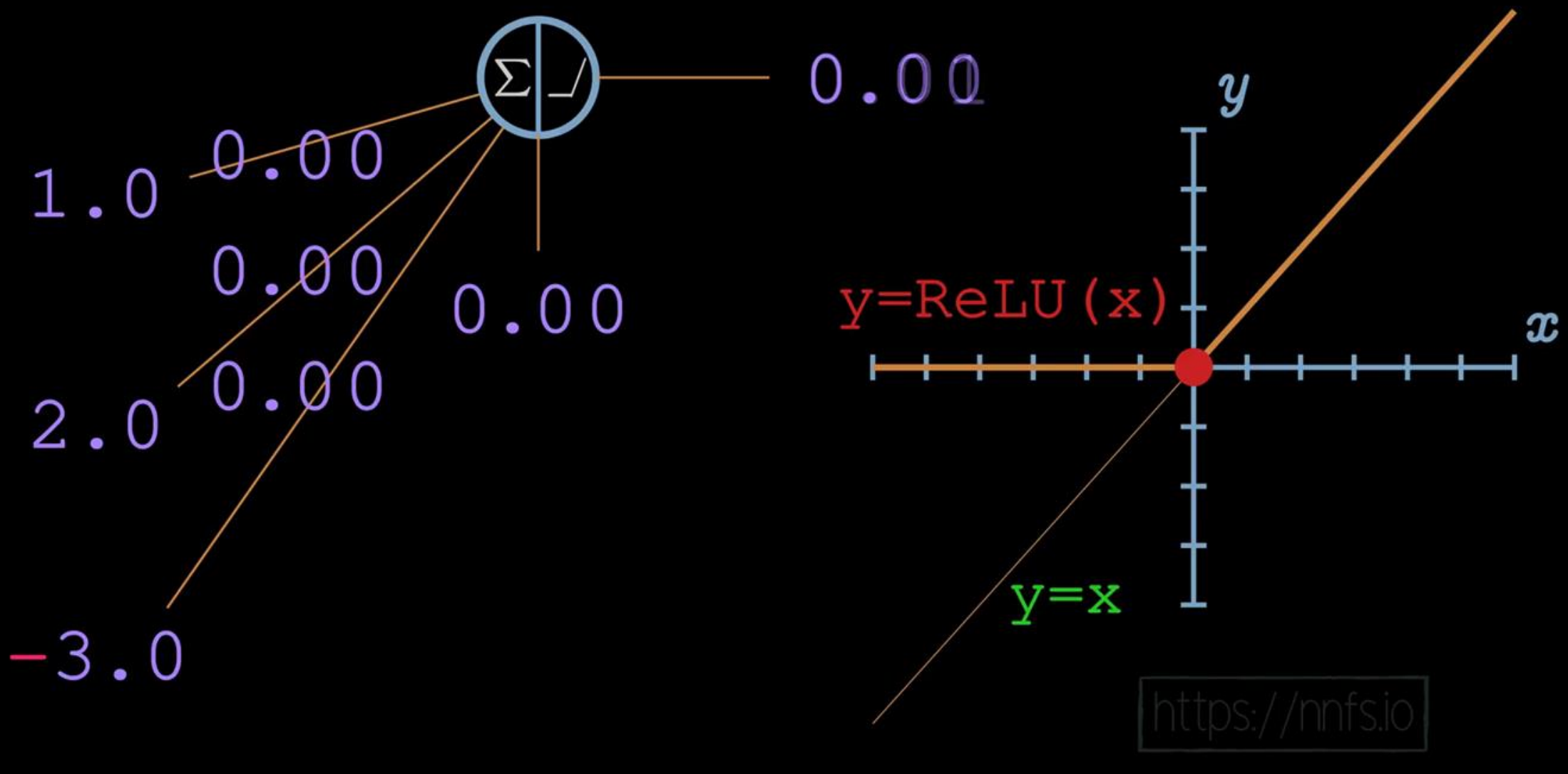
$$y = \frac{1}{1 + e^{-x}}$$

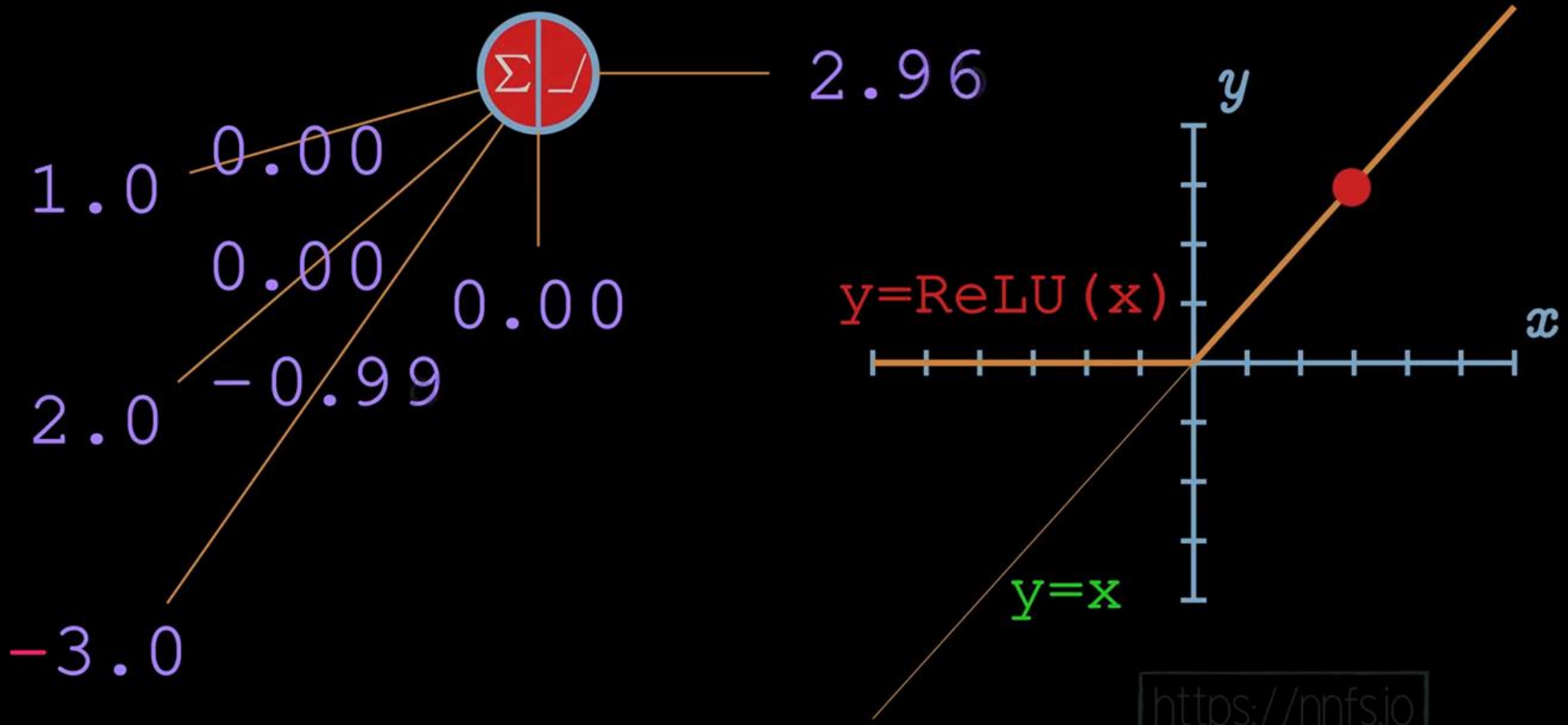


https://nnfs.io

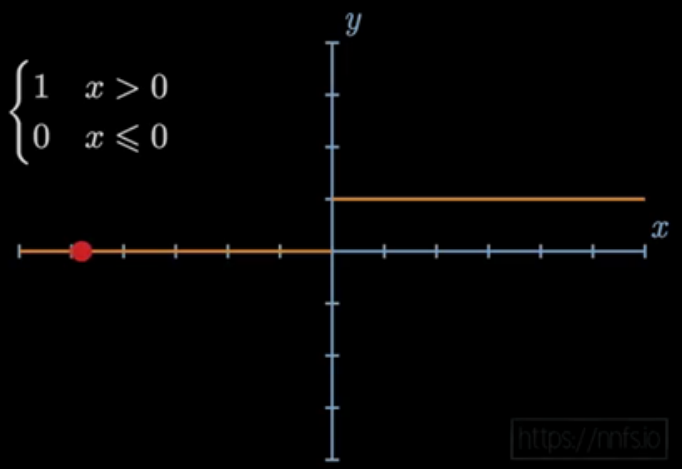

Σ
σ
0.73
1.0
0.98
0.00
0.00
2.0
0.00
−3.0
y
x
y=σ(x)
y=x
https://nnfs.io


https://nnfs.io

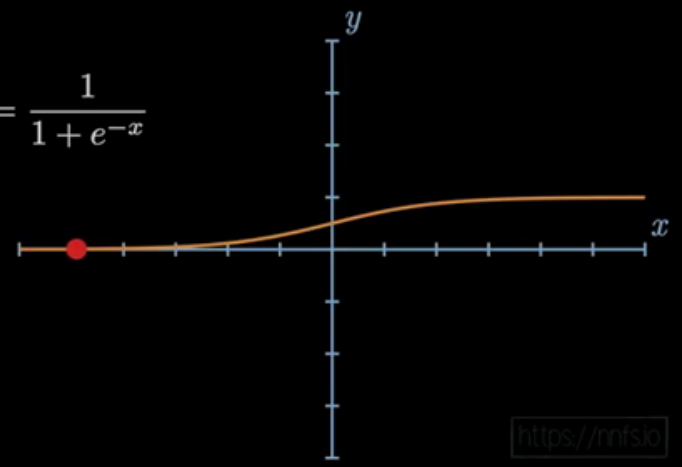

0.00
1.0
0.00
2.0
0.00
−3.0
0.00
0.00
y
x
y=ReLU(x)
y=x
https://nnfs.io


1.0
0.00
2.0
0.00
−0.99
−3.0
0.00
2.96
y
x
y=ReLU(x)
y=x
https://nnfs.io

$$y = \begin{cases} 1 & x > 0 \\ 0 & x \leqslant 0 \end{cases}$$

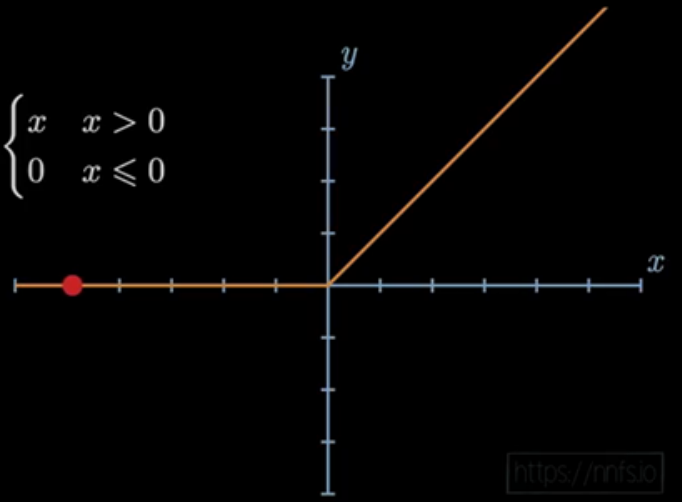


$$y = \frac{1}{1 + e^{-x}}$$

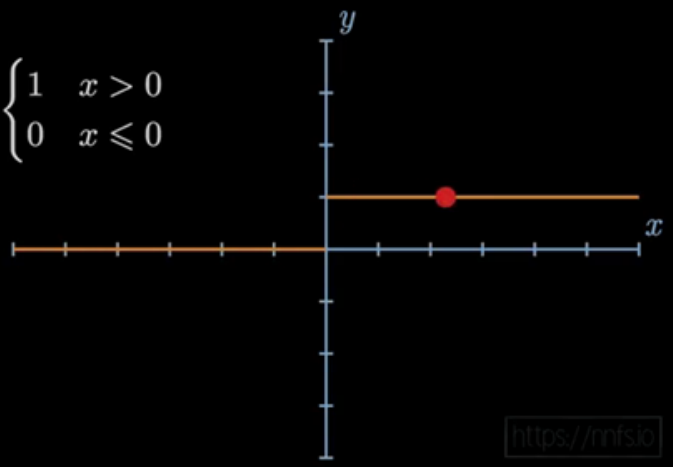


?

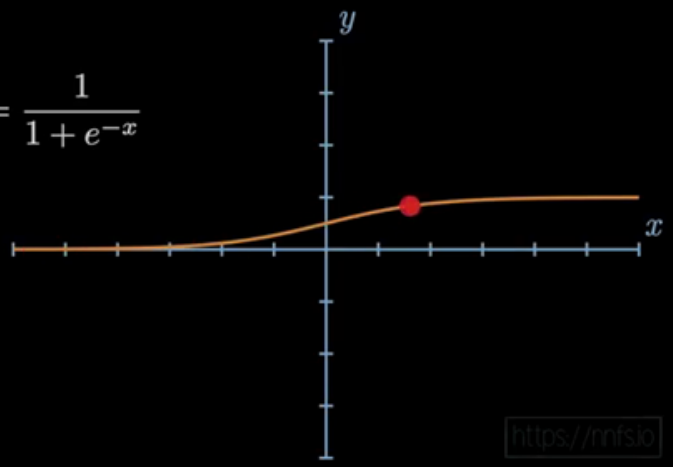
$$y = \begin{cases} x & x > 0 \\ 0 & x \leqslant 0 \end{cases}$$



$$y = \begin{cases} 1 & x > 0 \\ 0 & x \leqslant 0 \end{cases}$$

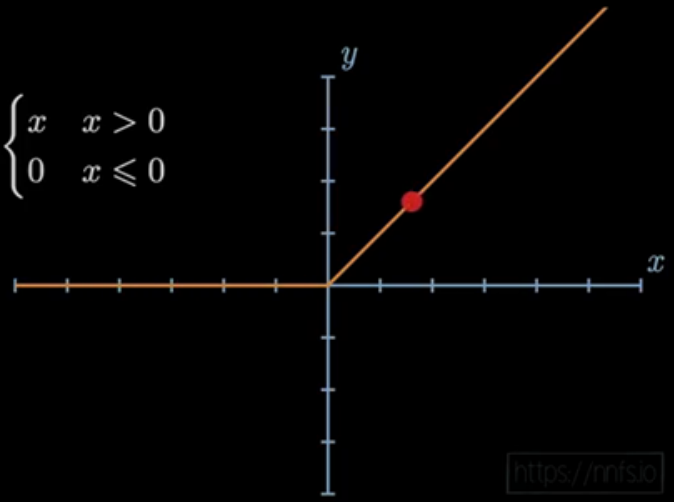


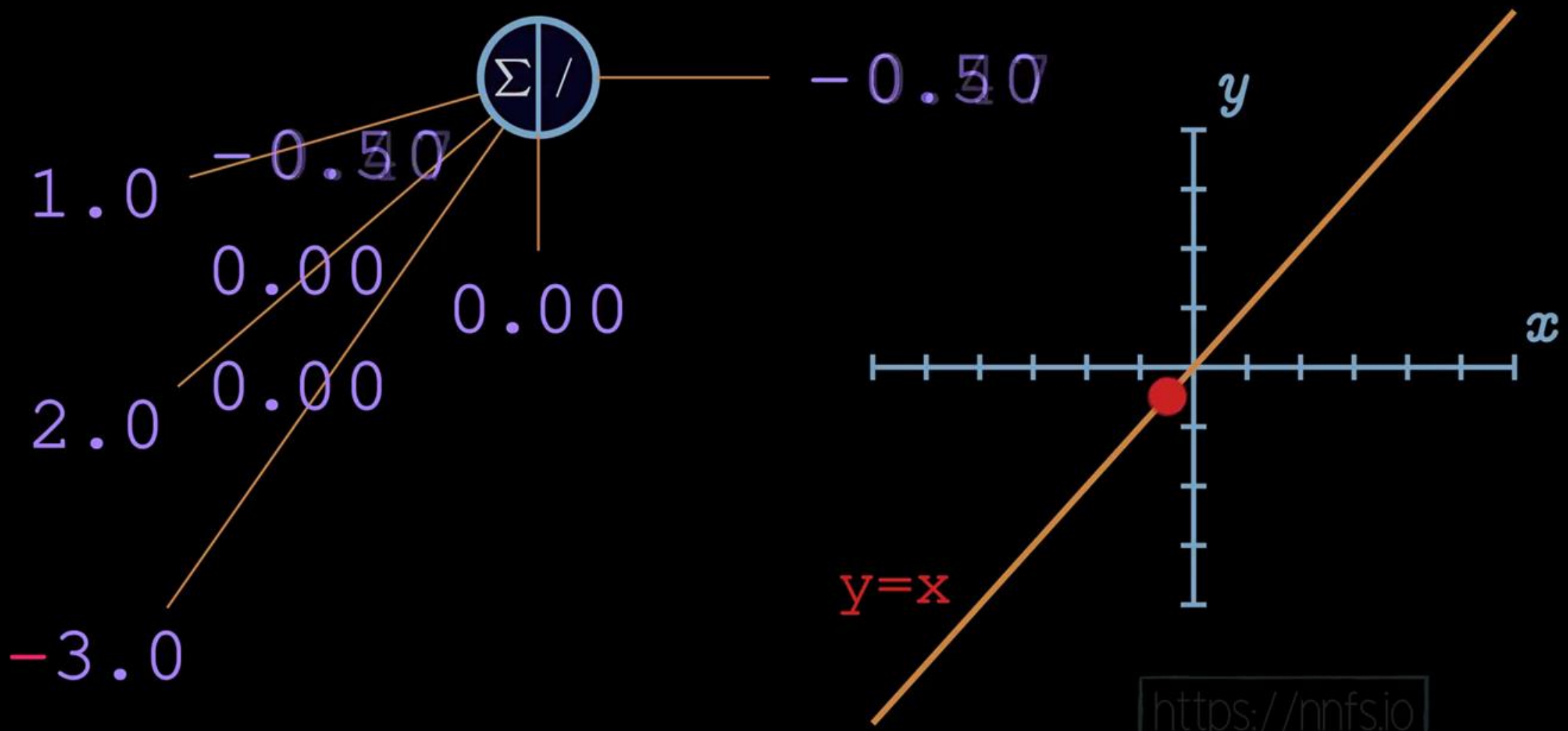
$$y = \frac{1}{1 + e^{-x}}$$

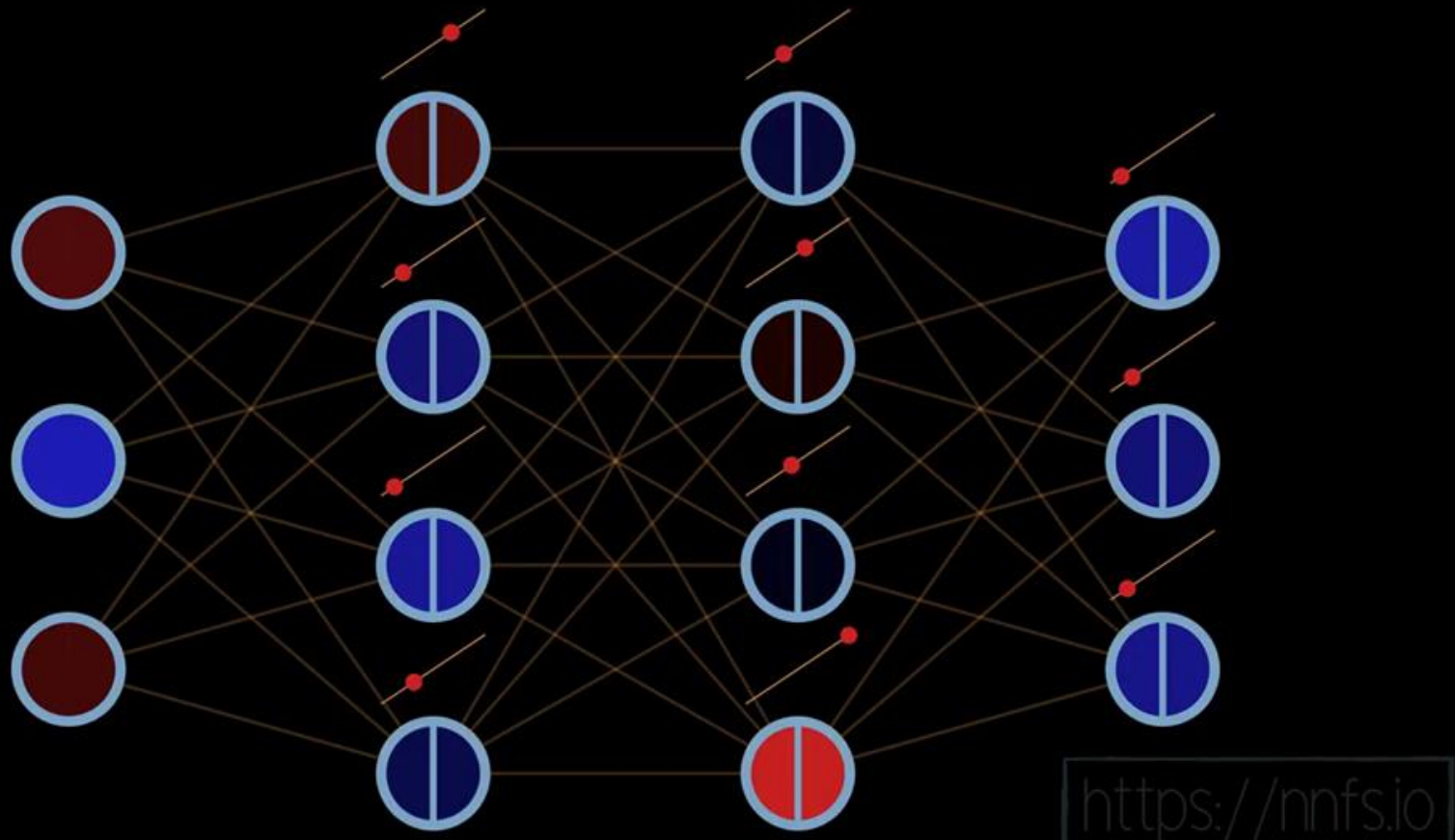


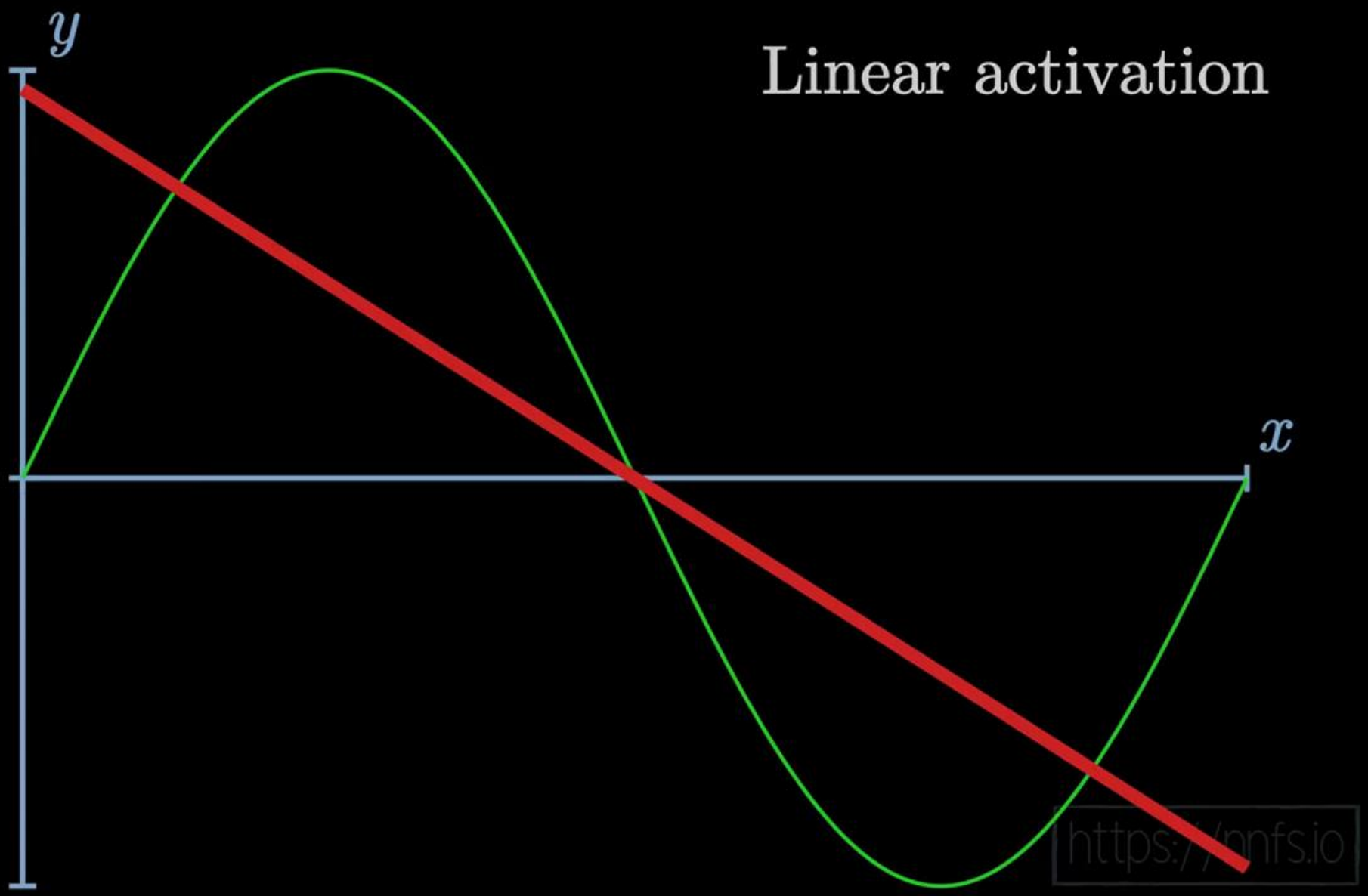
?

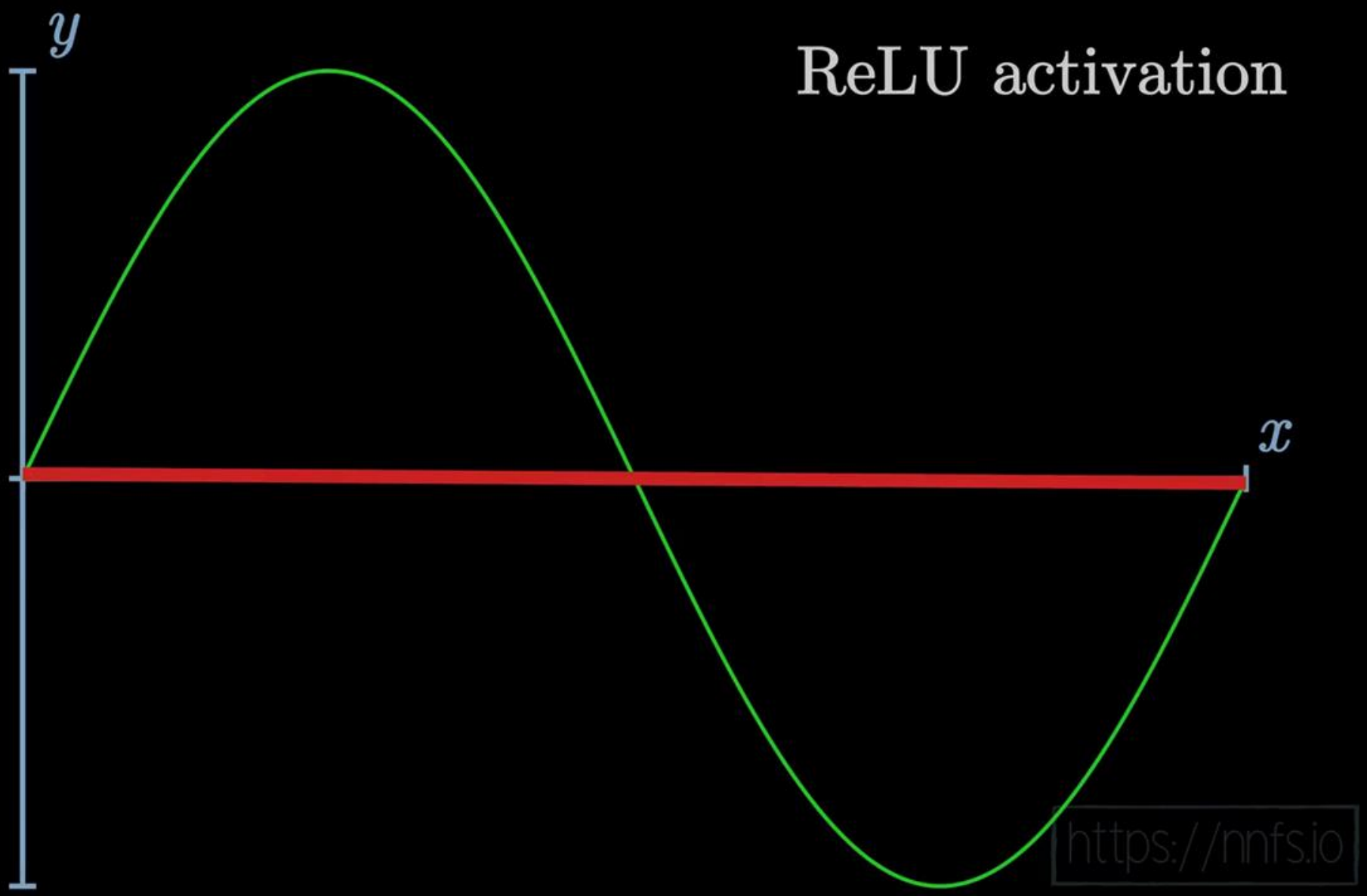
$$y = \begin{cases} x & x > 0 \\ 0 & x \leqslant 0 \end{cases}$$

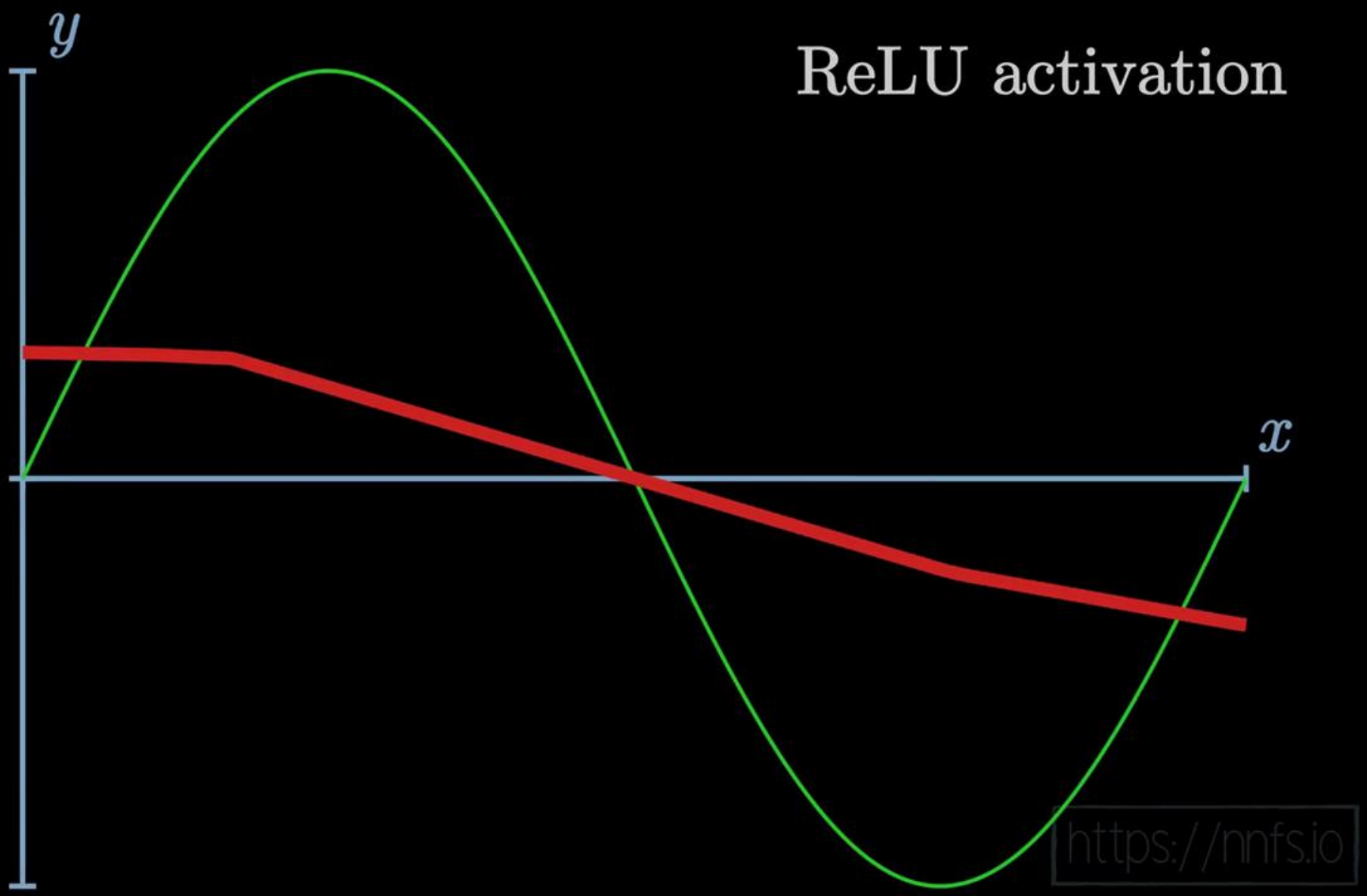


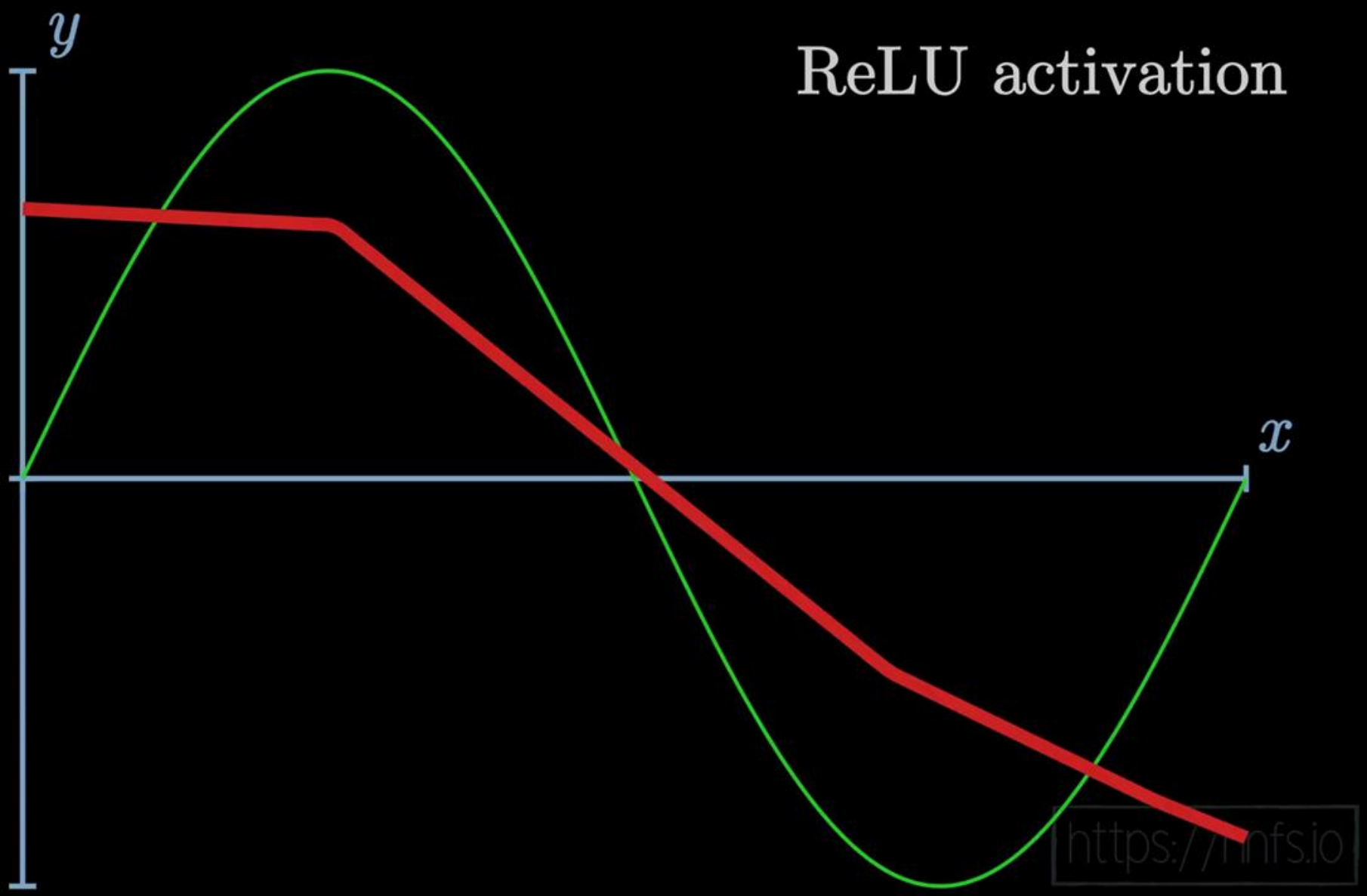

Σ
/
1.0
2.0
−3.0
0.00
0.00
0.00
−0.50
−0.50
y
x
y=x
https://nnfs.io

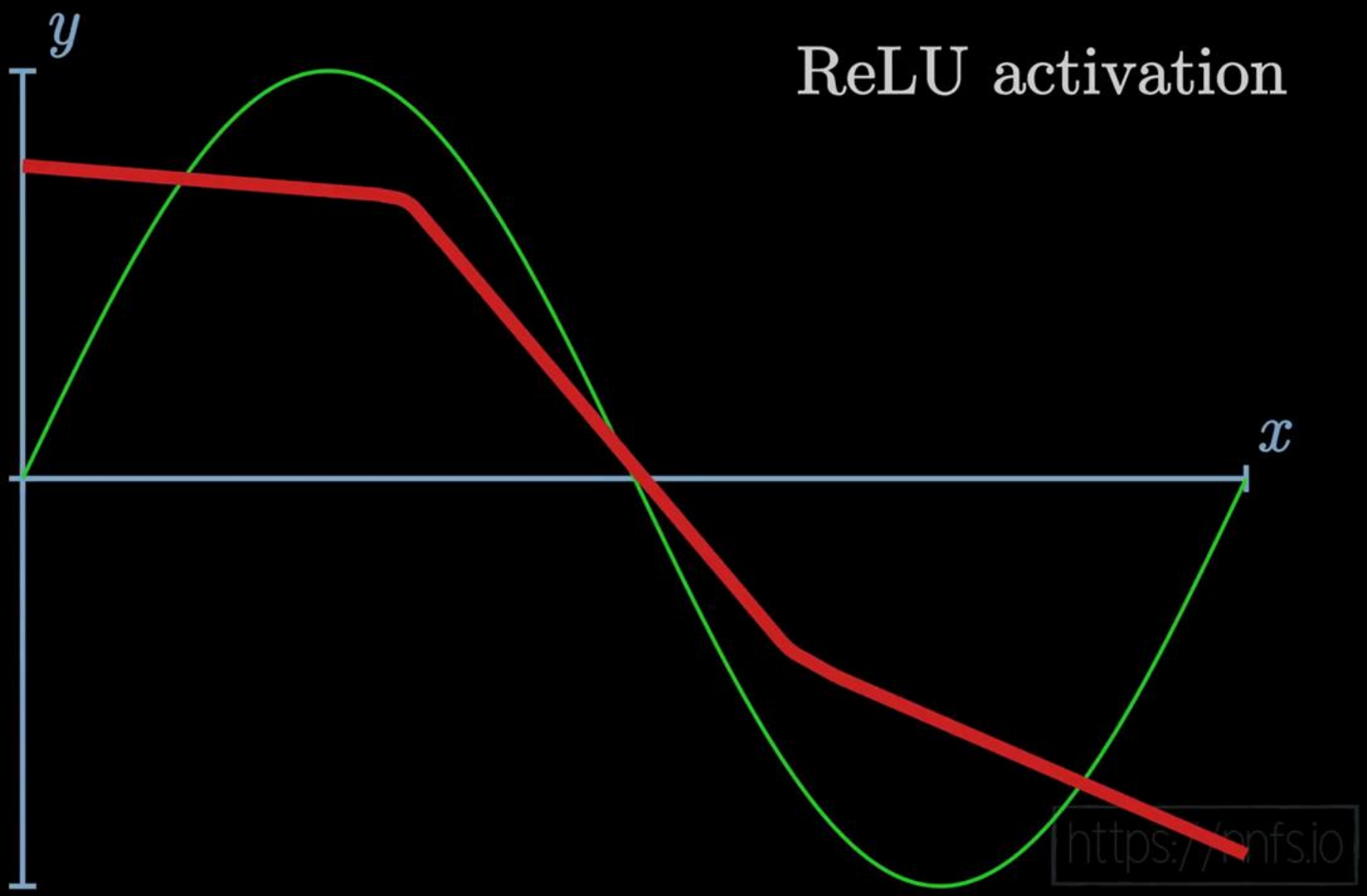

https://nnfs.io

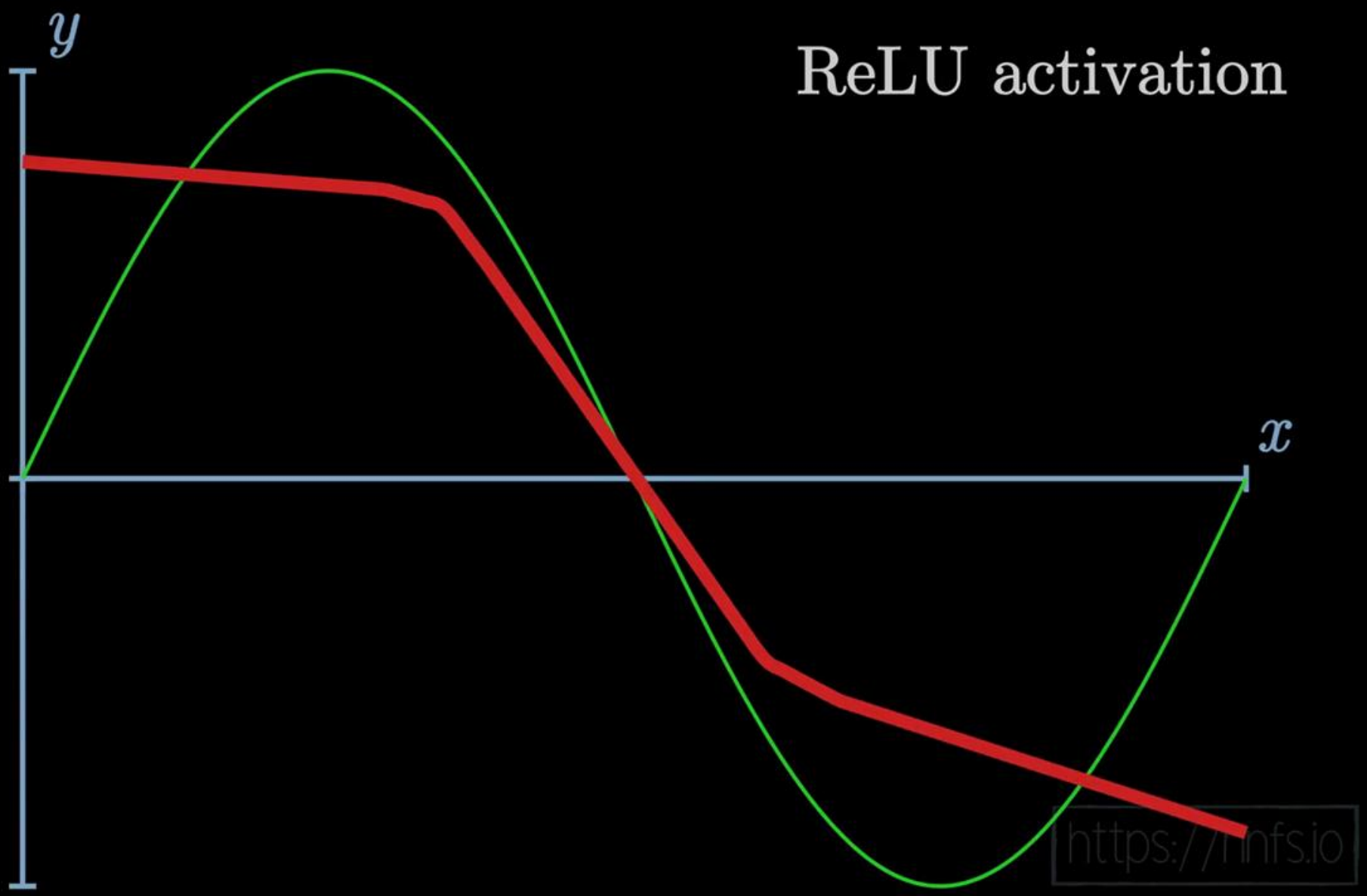

y
Linear activation
x
https://nnfs.io

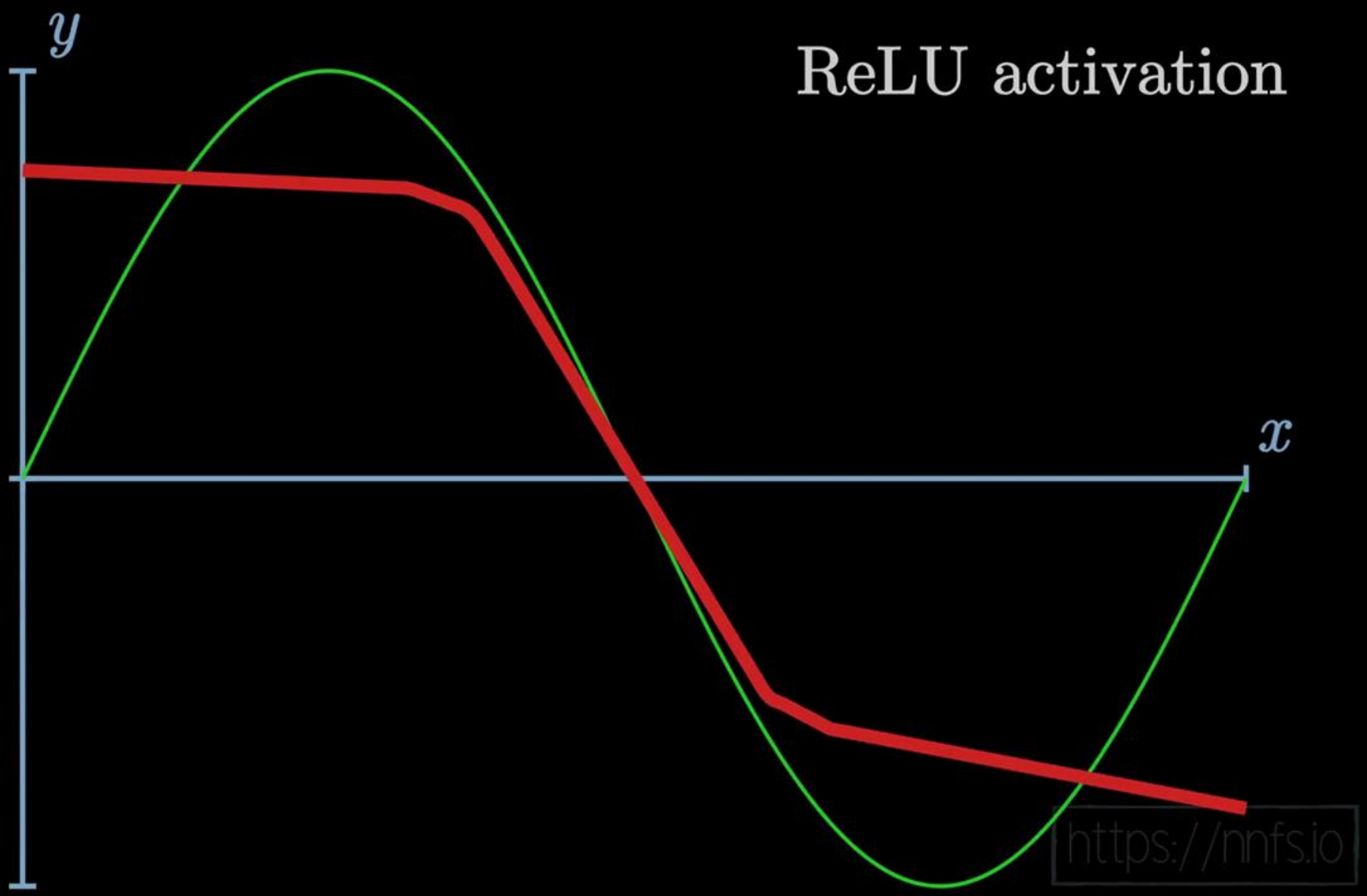

y
ReLU activation
x
https://nnfs.io

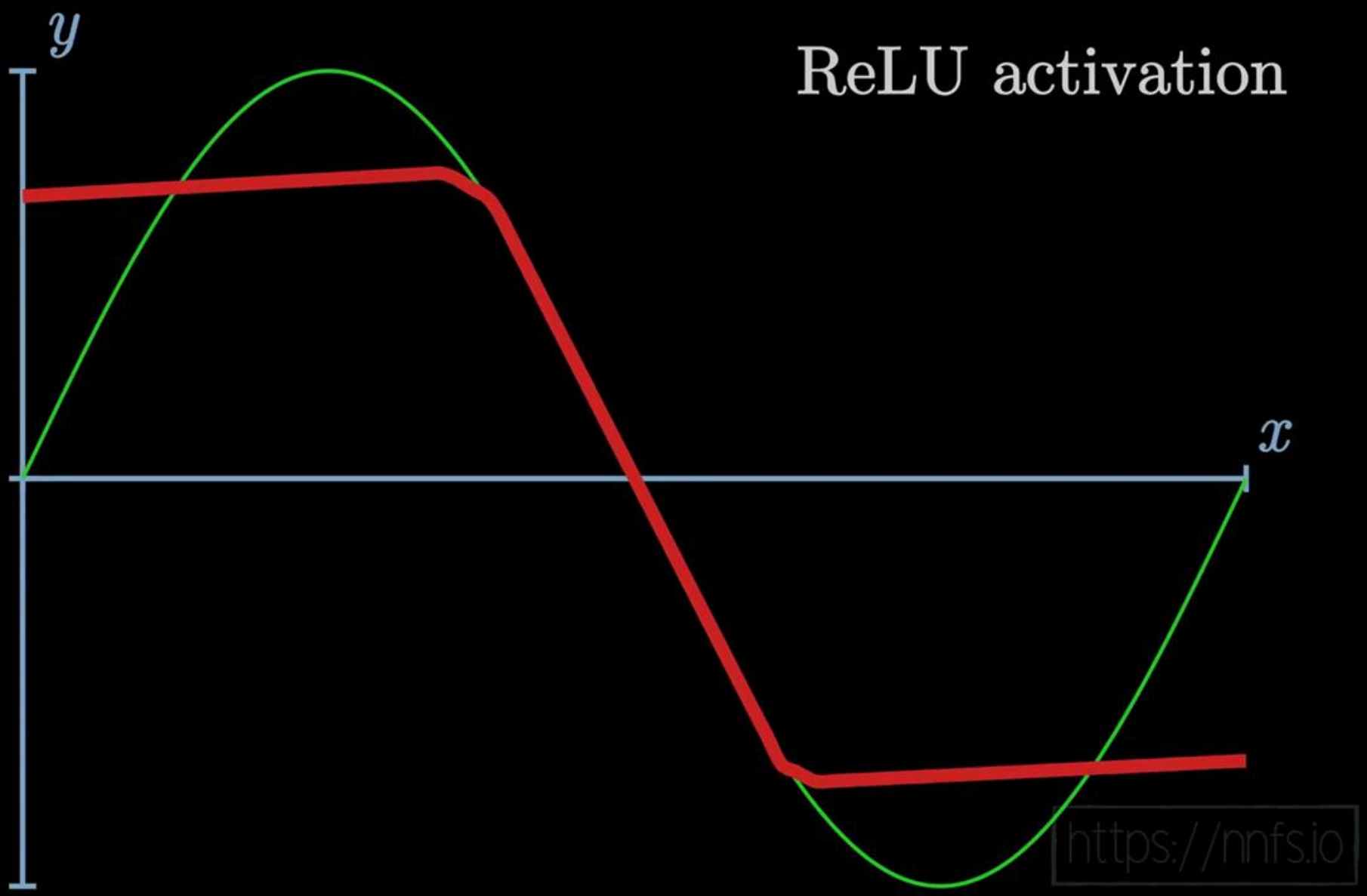

y
ReLU activation
x
https://nnfs.io

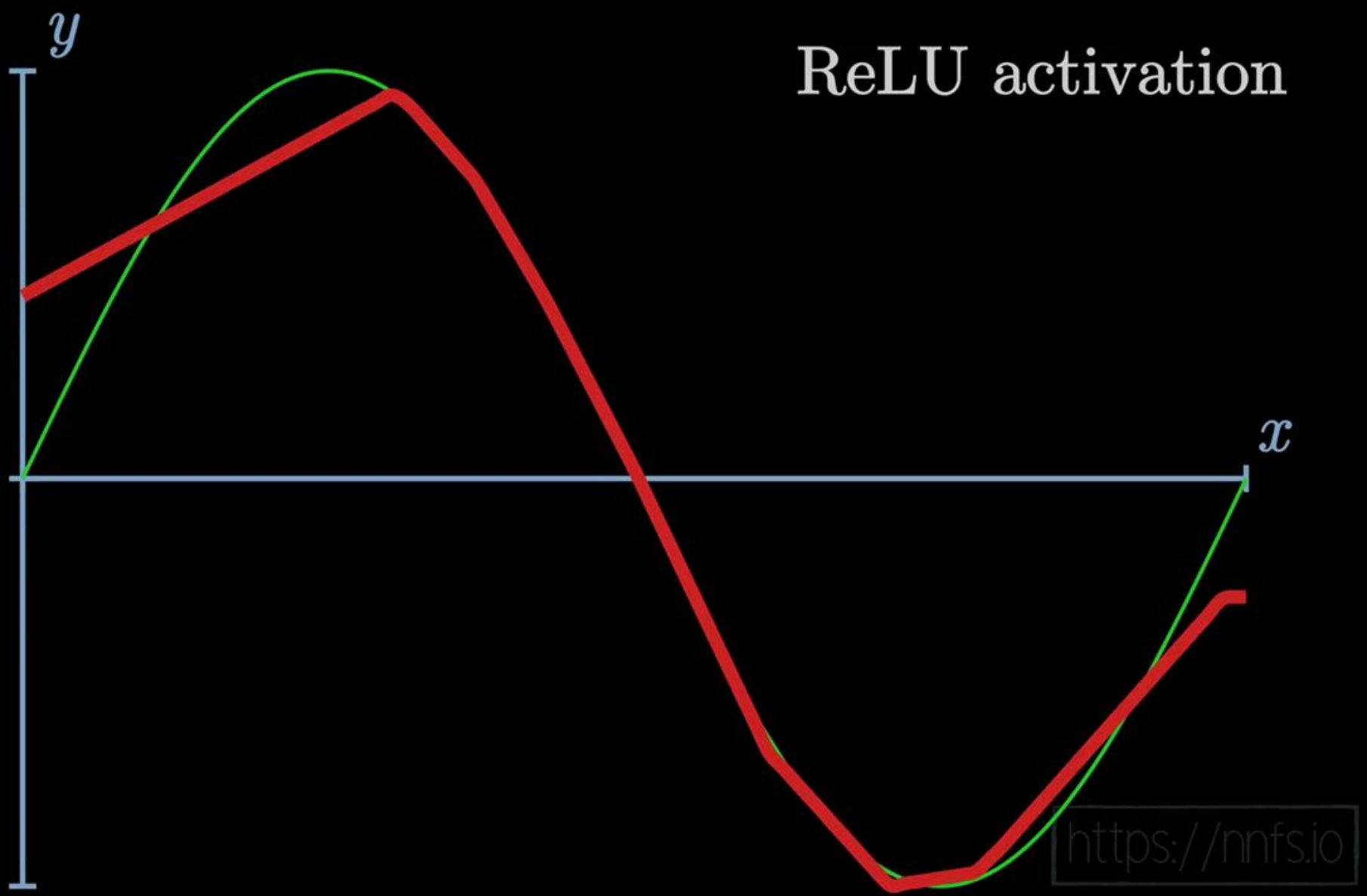

y
ReLU activation
x
https://nnfs.io

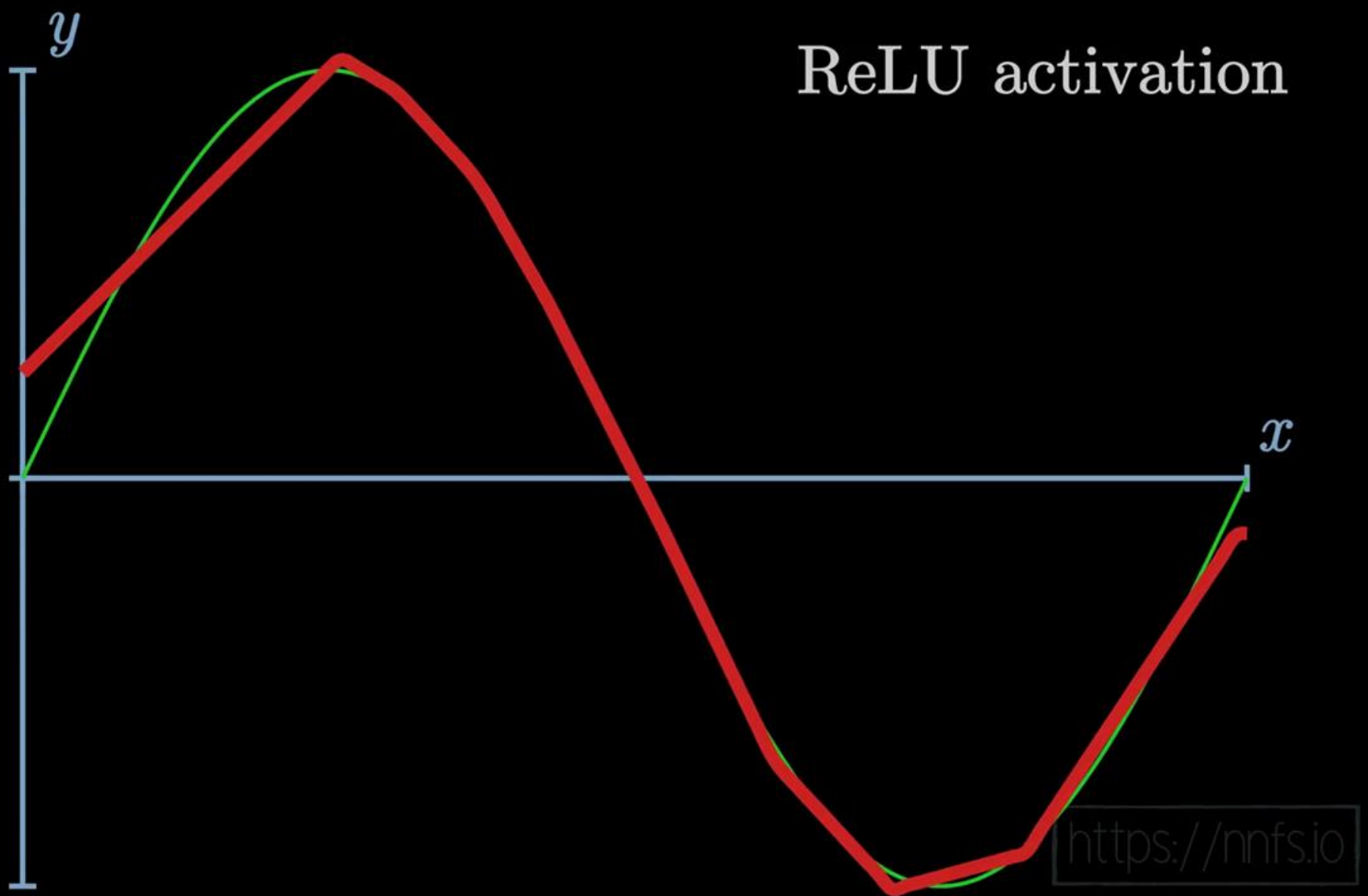

y
ReLU activation
x
https://pnfs.io

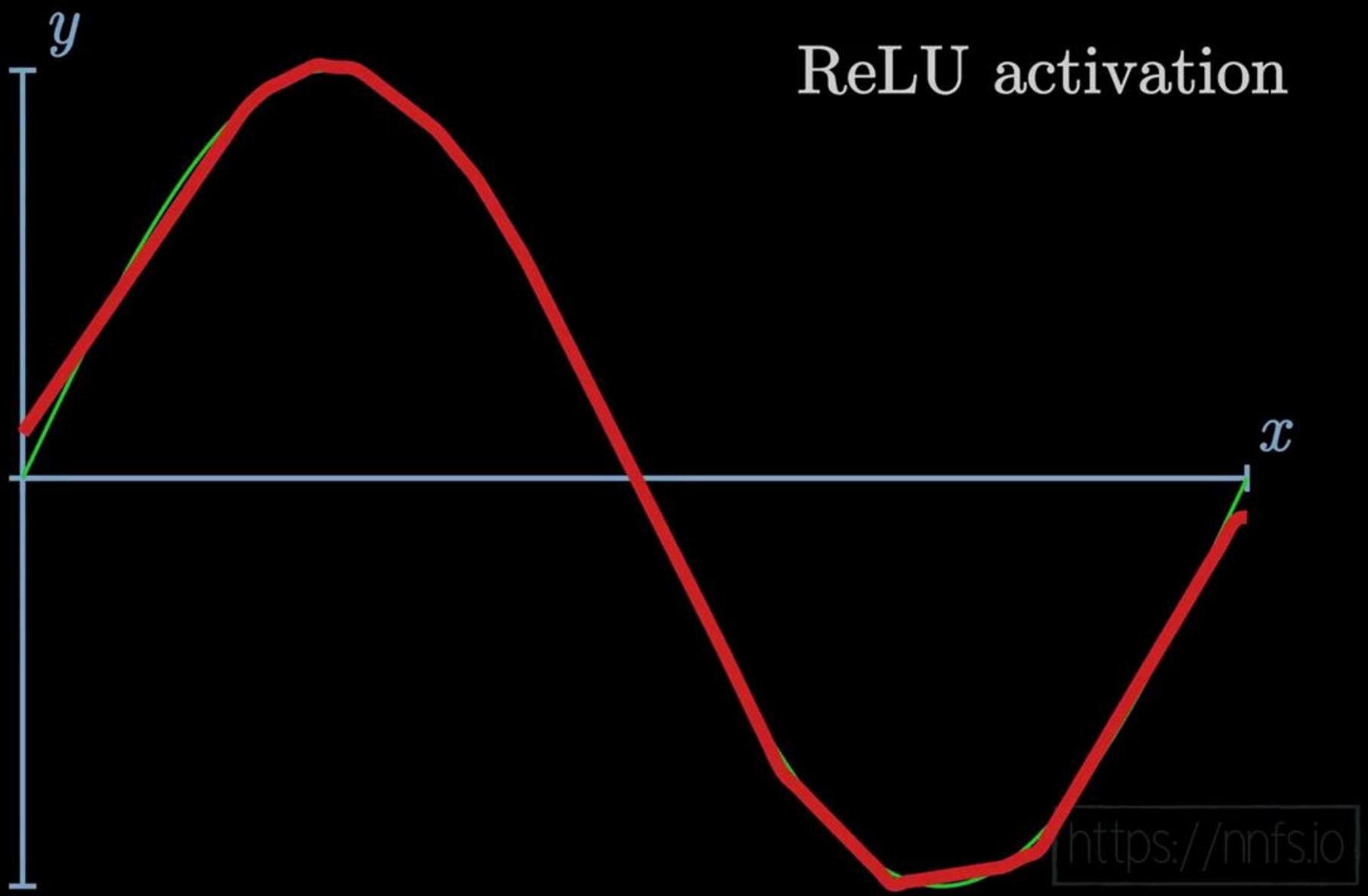

y
ReLU activation
x
https://nnfs.io

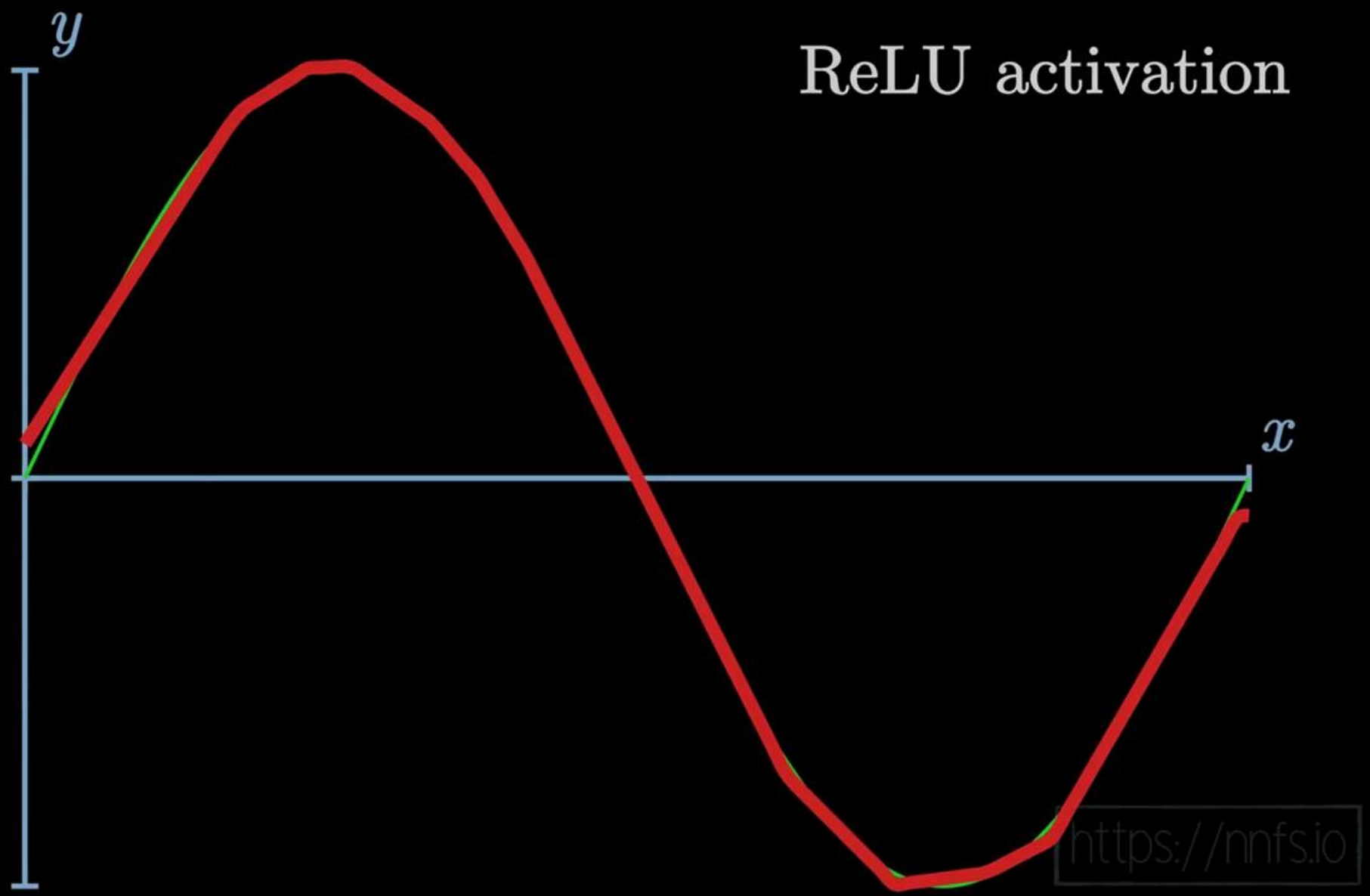

y
ReLU activation
x
https://nnfs.io

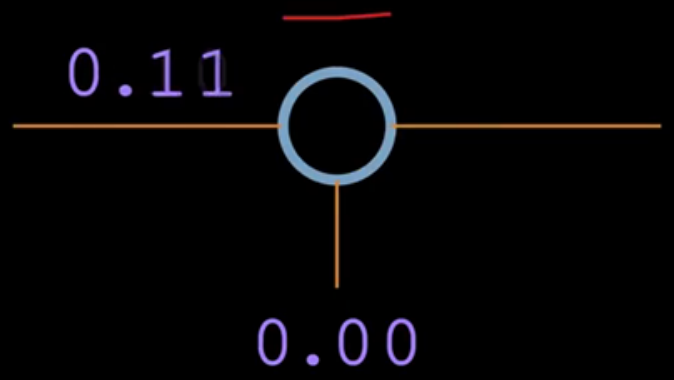

y
ReLU activation
x
https://nnfs.io

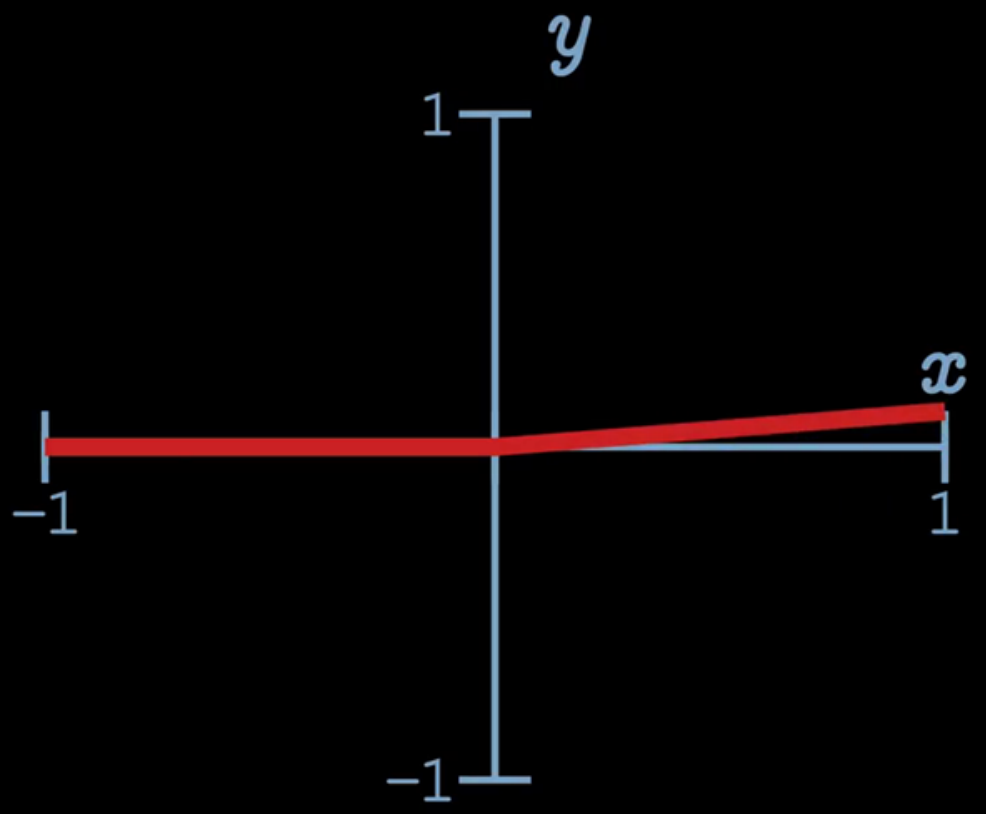

y
ReLU activation
x
https://nnfs.io

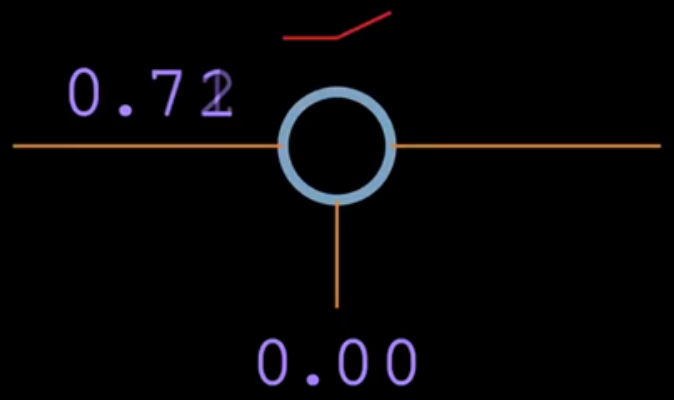

y
ReLU activation
x
https://nnfs.io

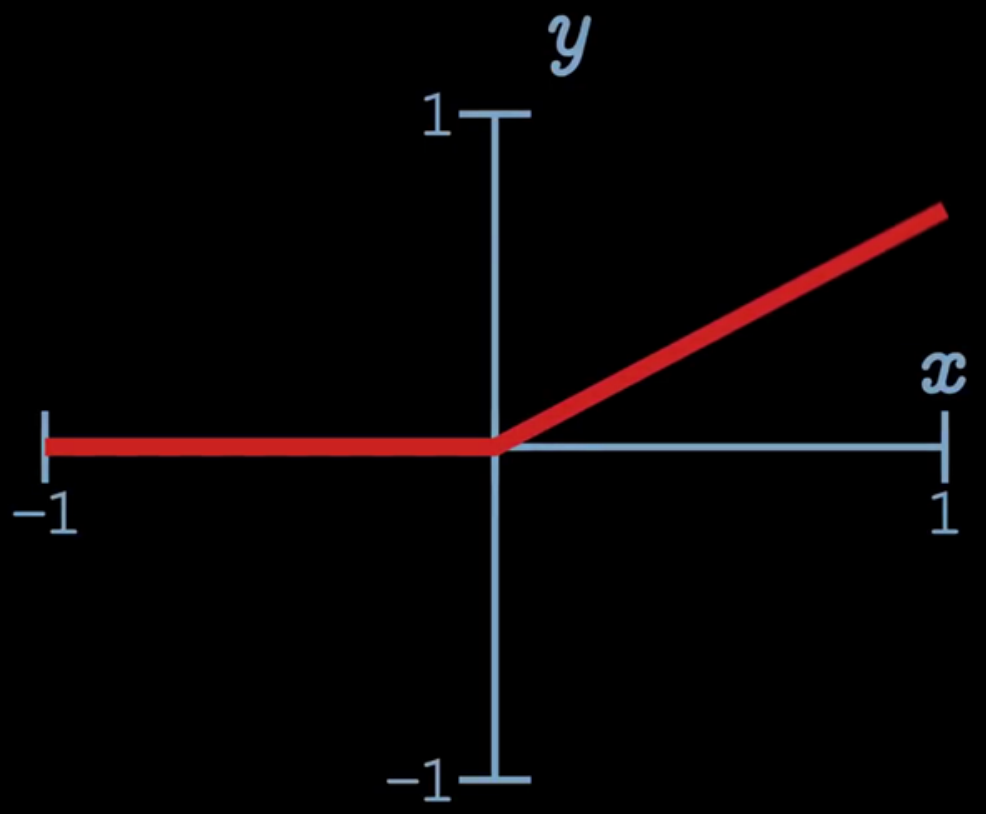

y
x
ReLU activation
https://nnfs.io

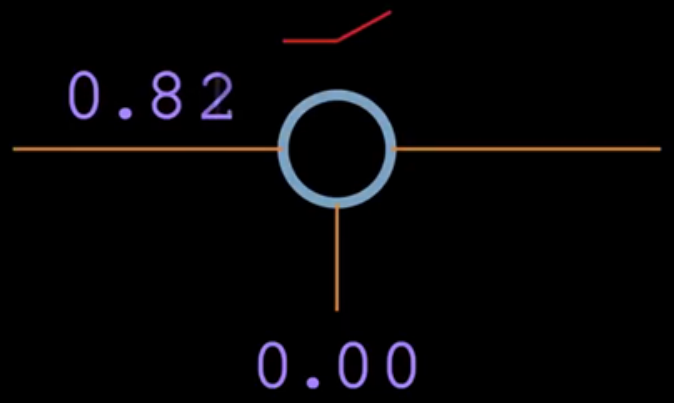

y
x
ReLU activation
https://nnfs.io

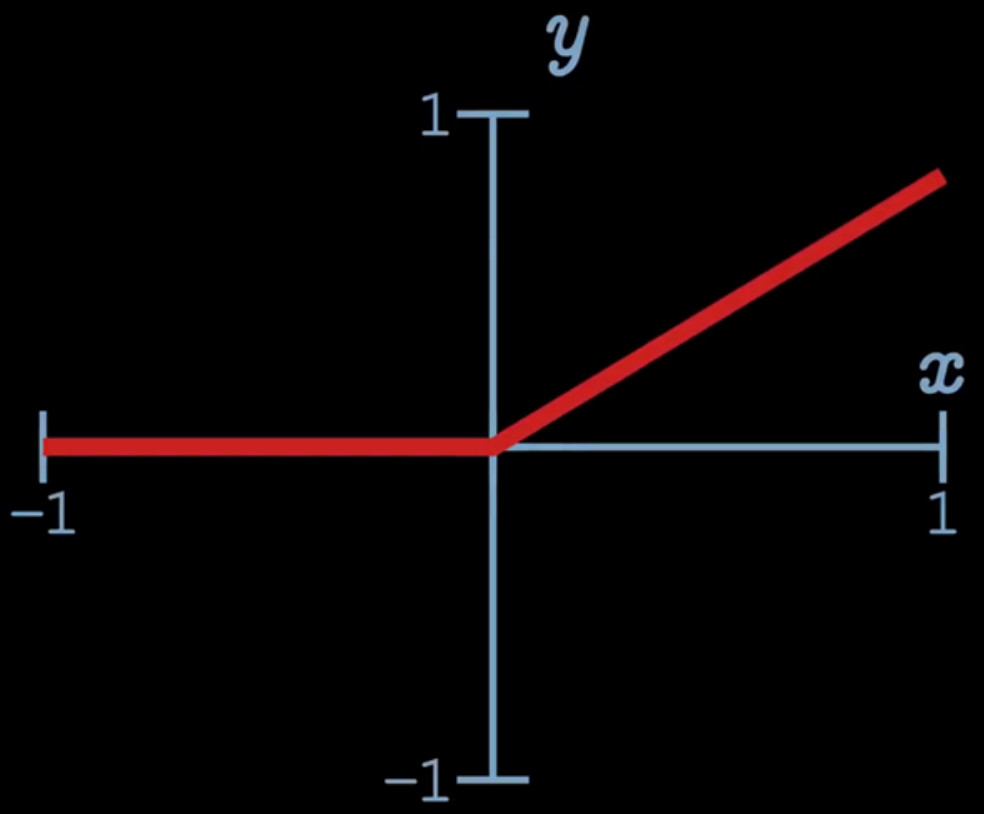


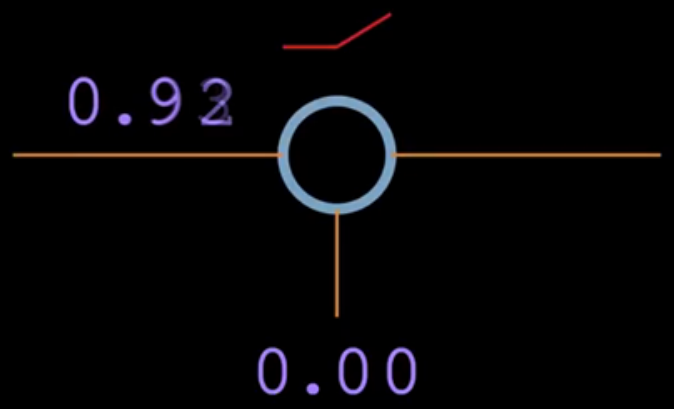


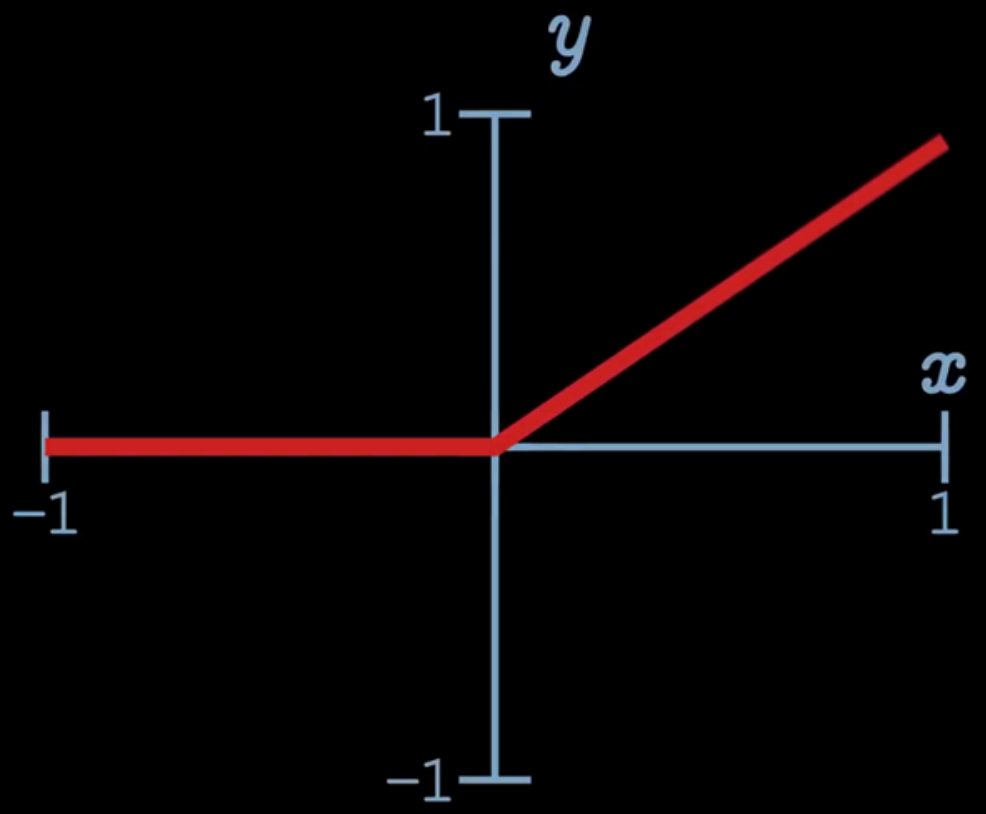


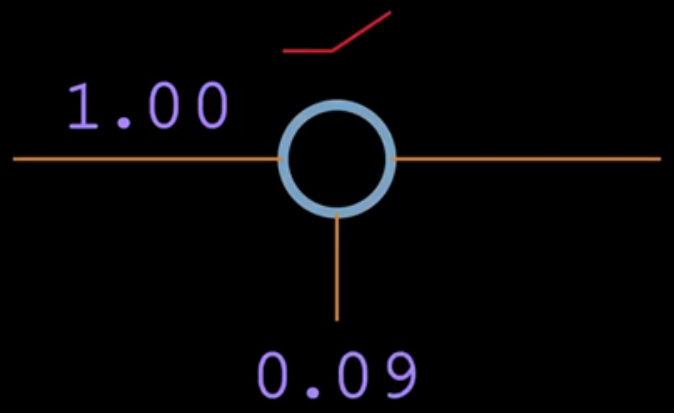


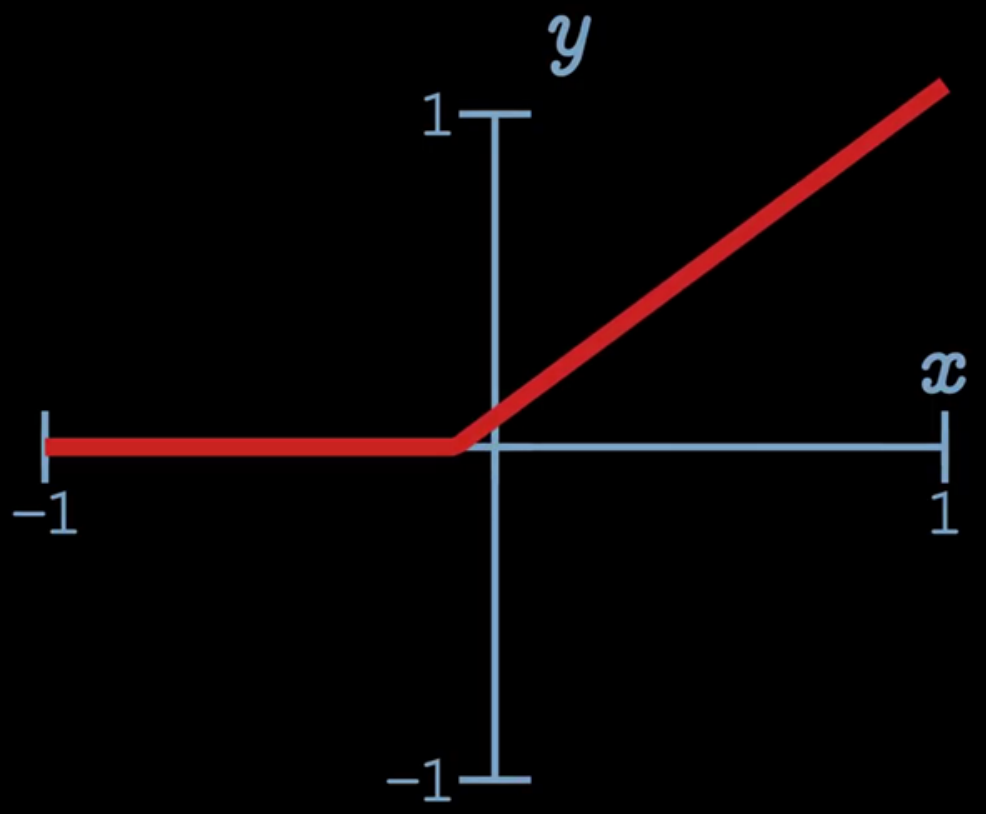


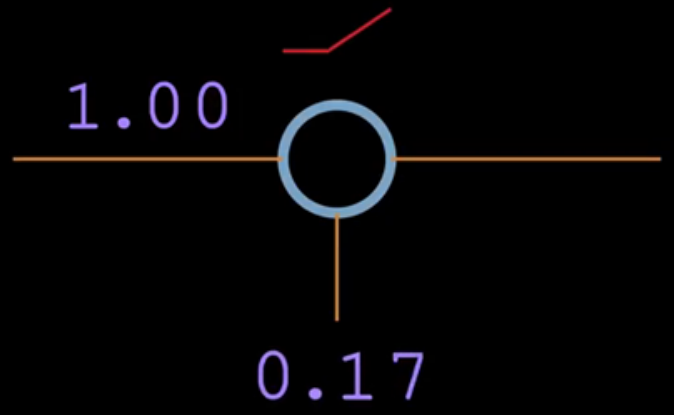


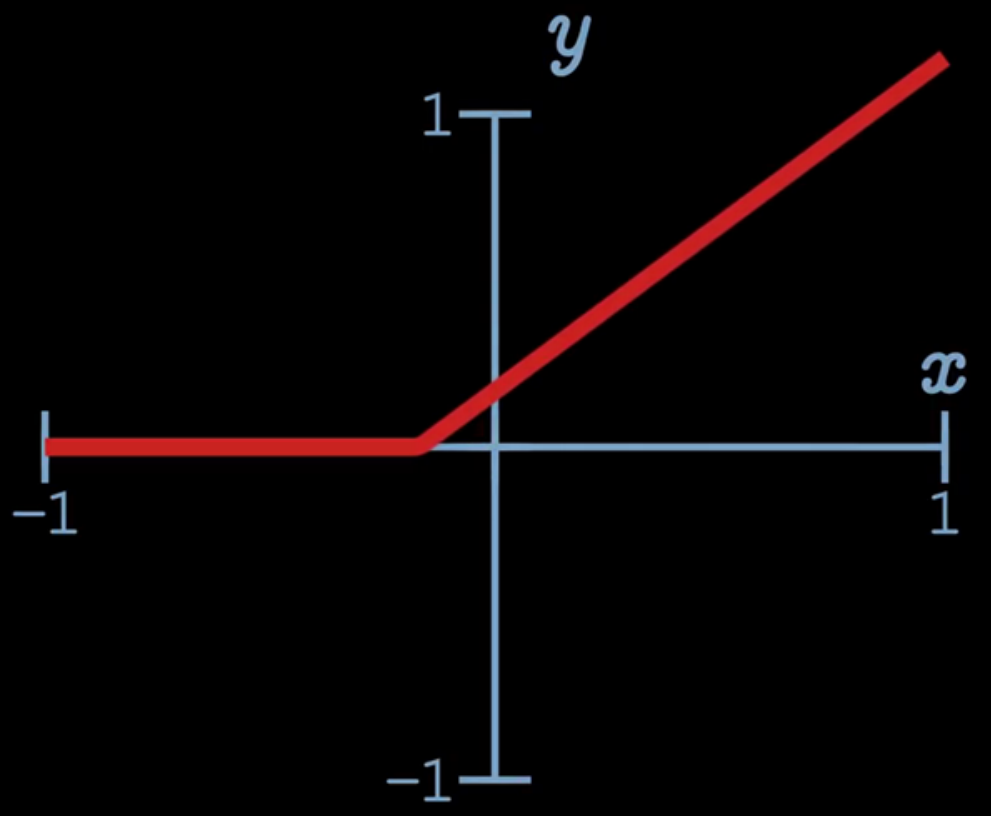


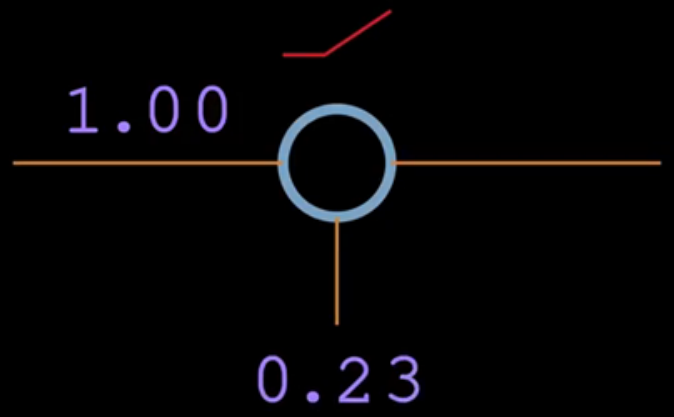


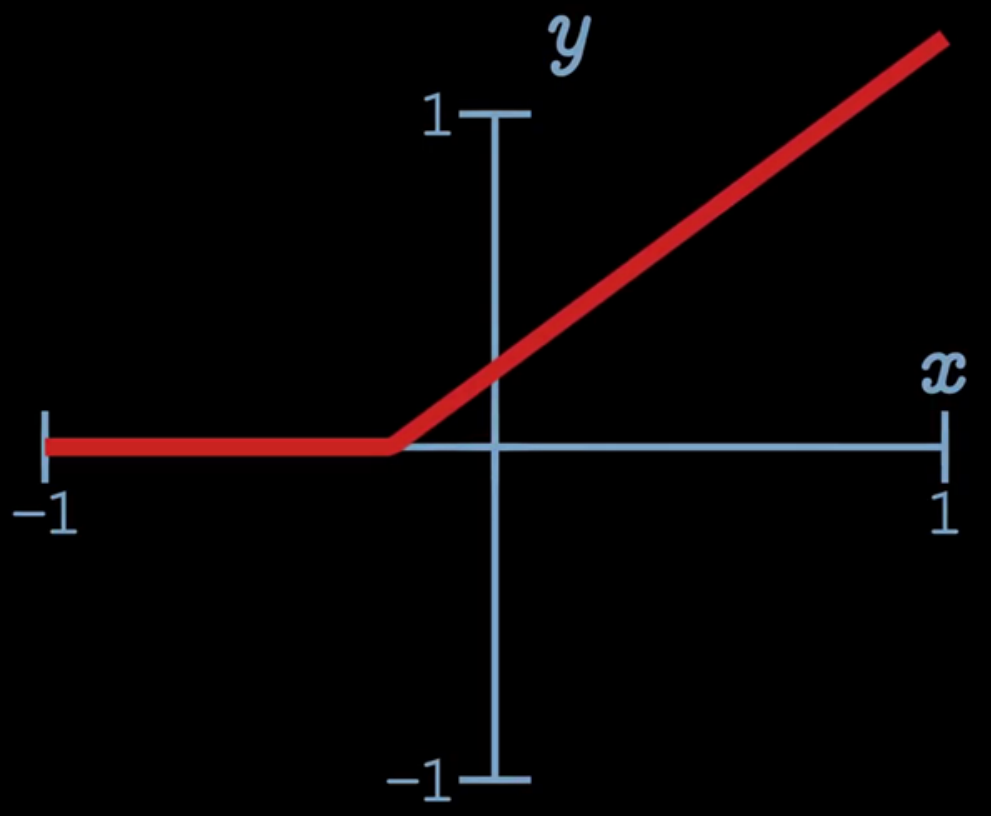


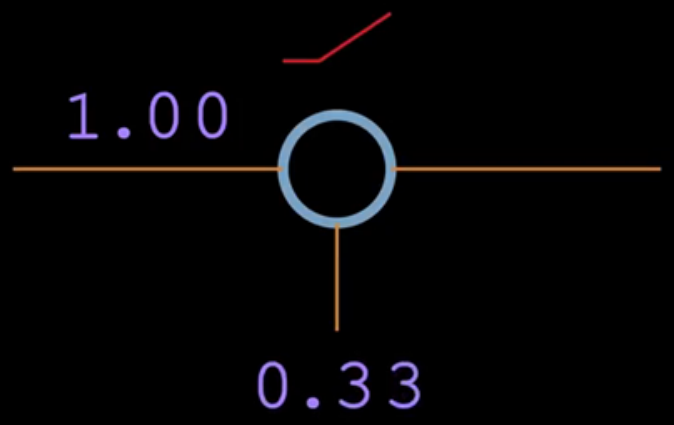


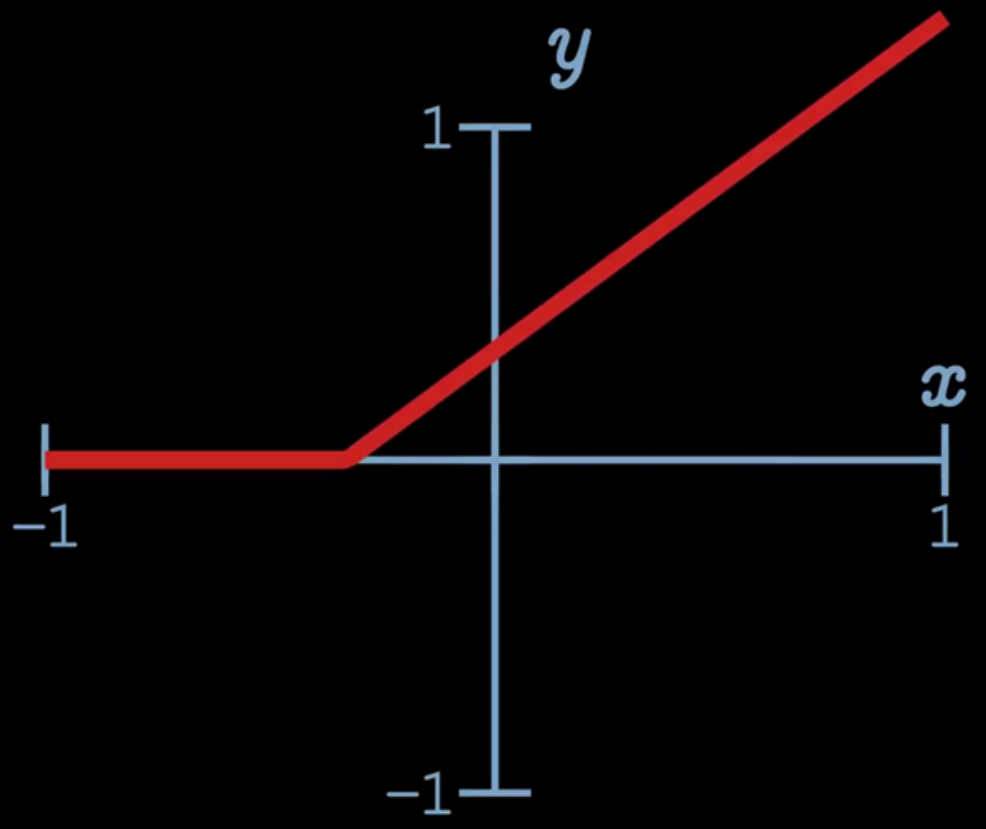


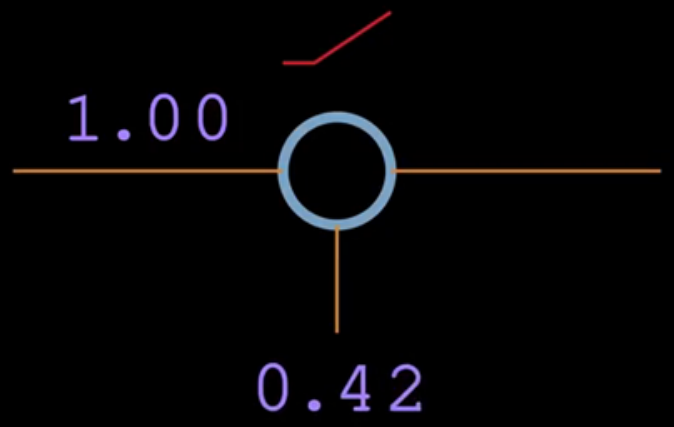


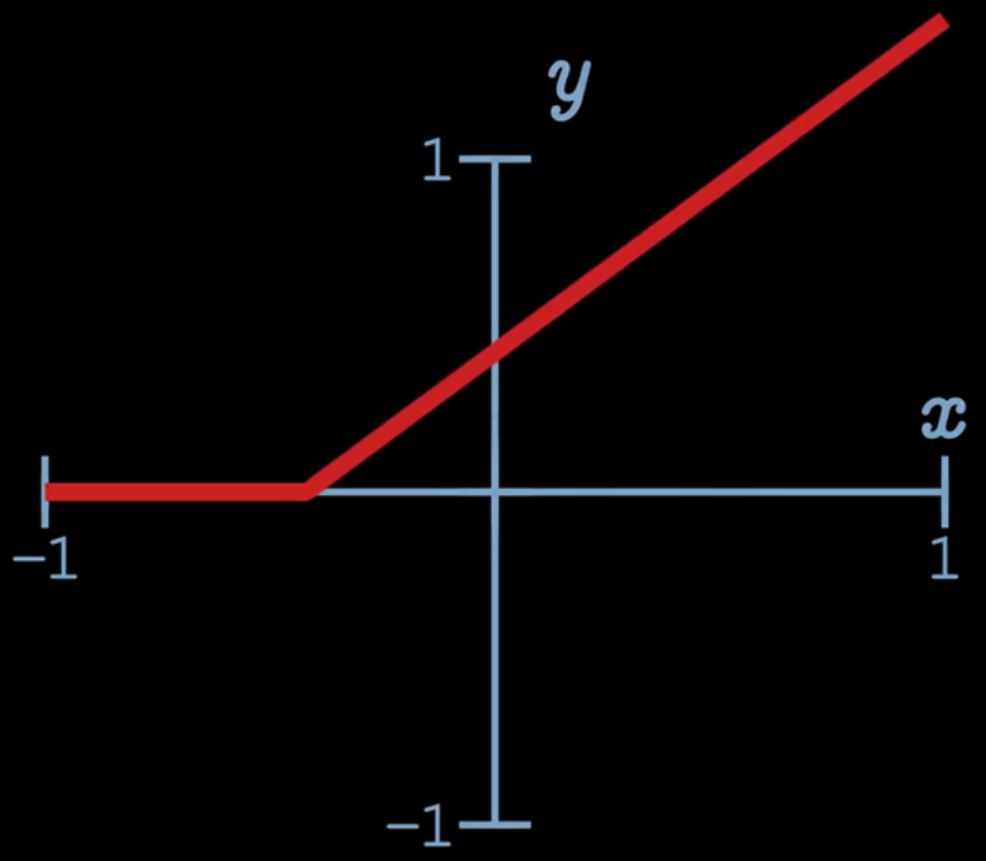


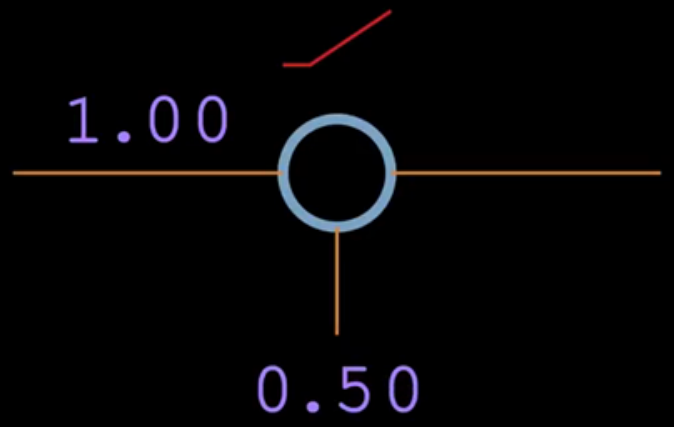


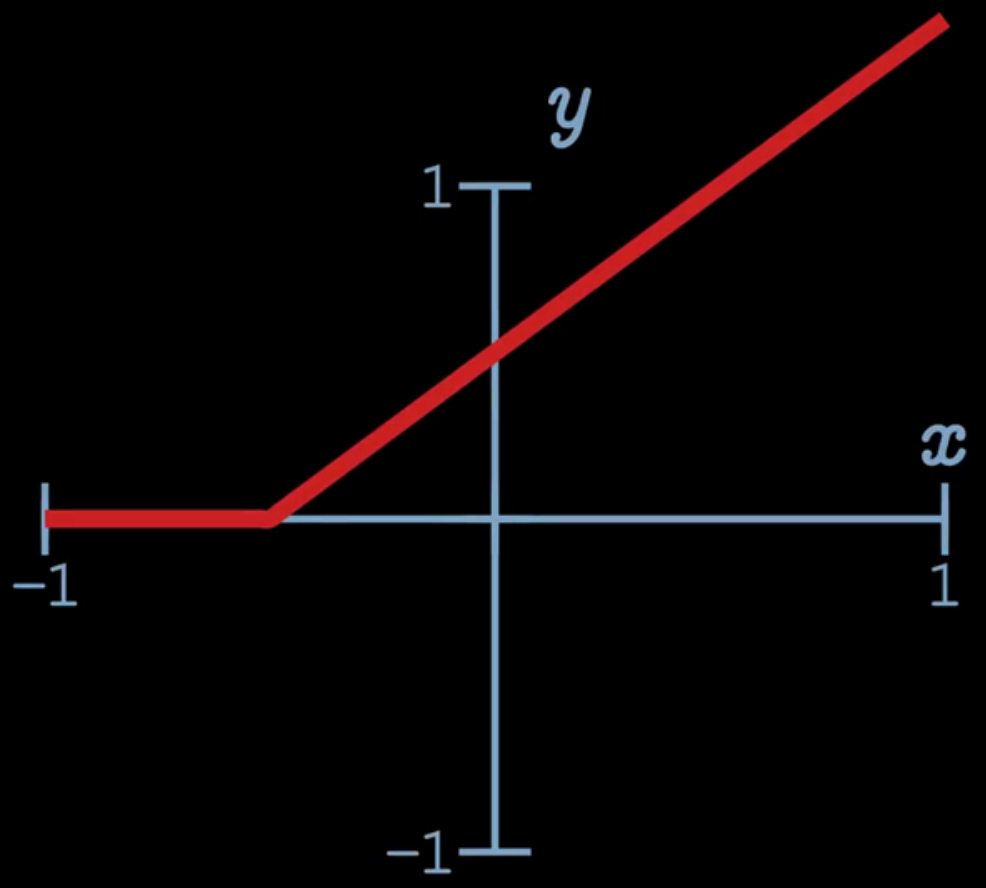


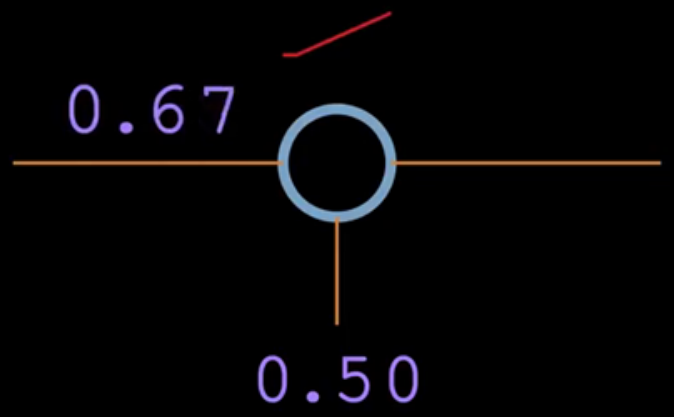


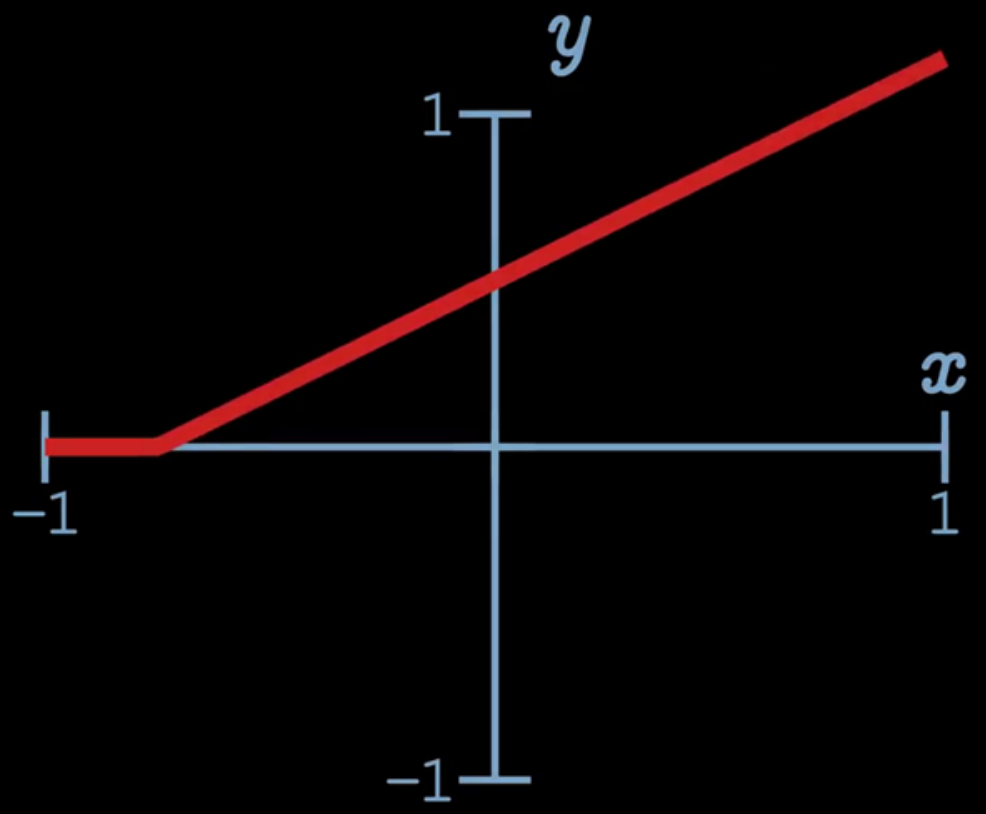


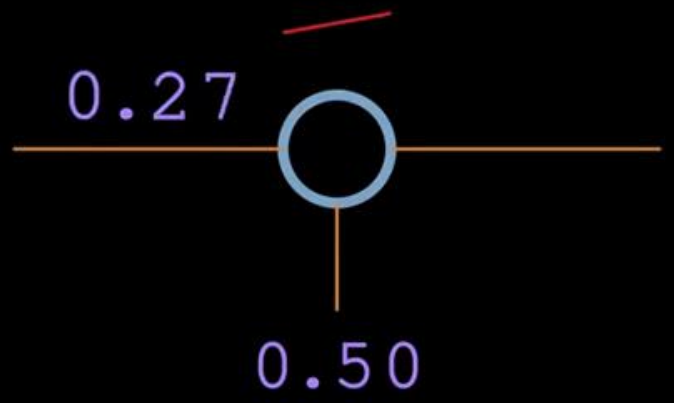


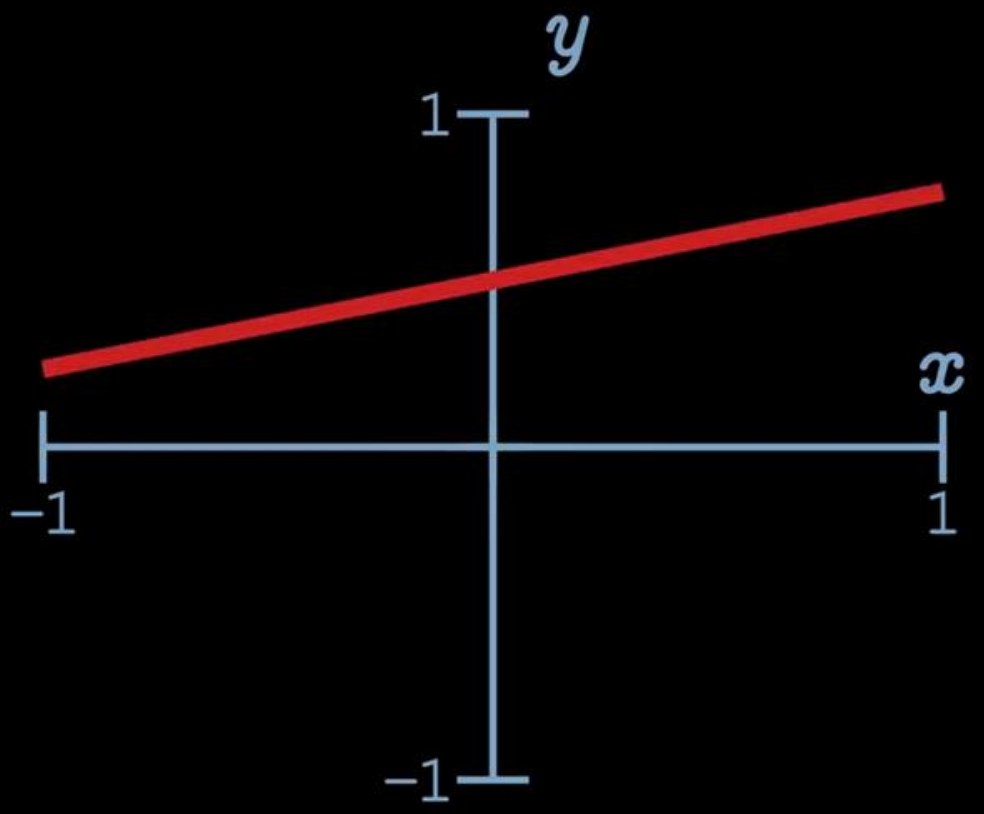


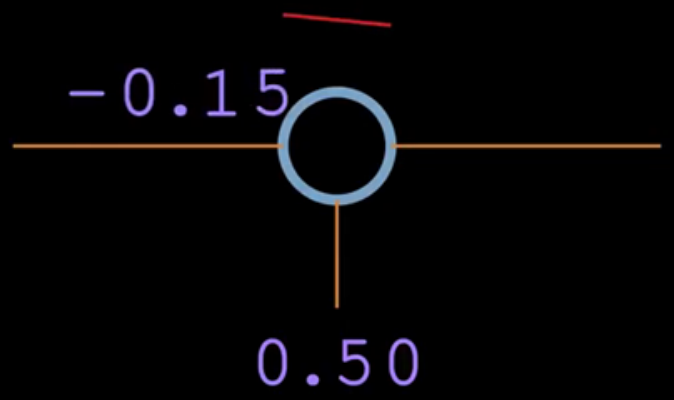


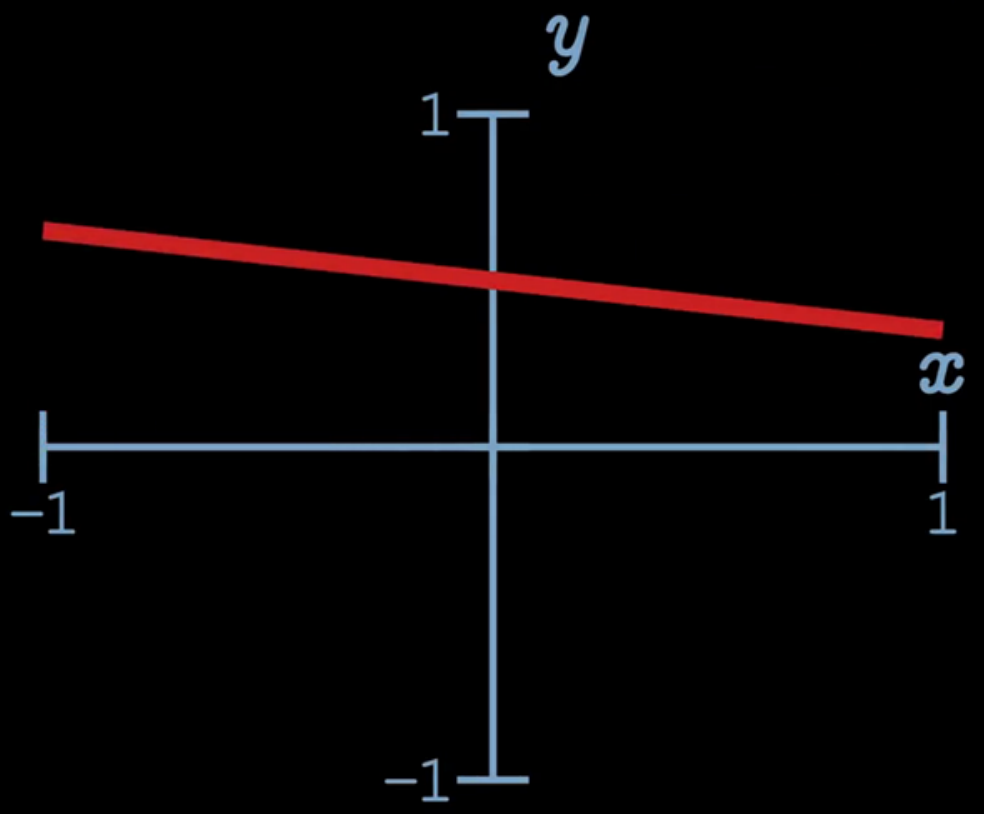


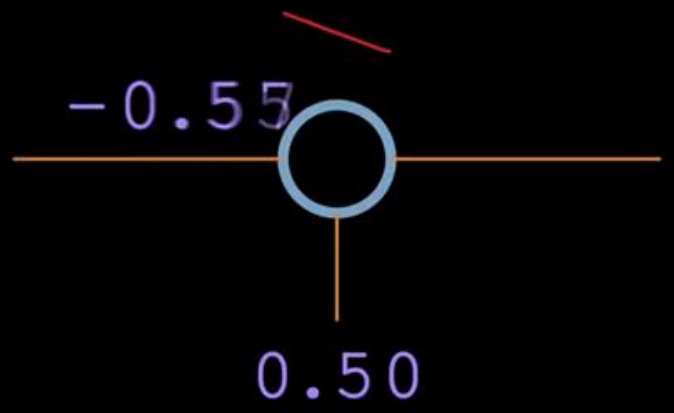


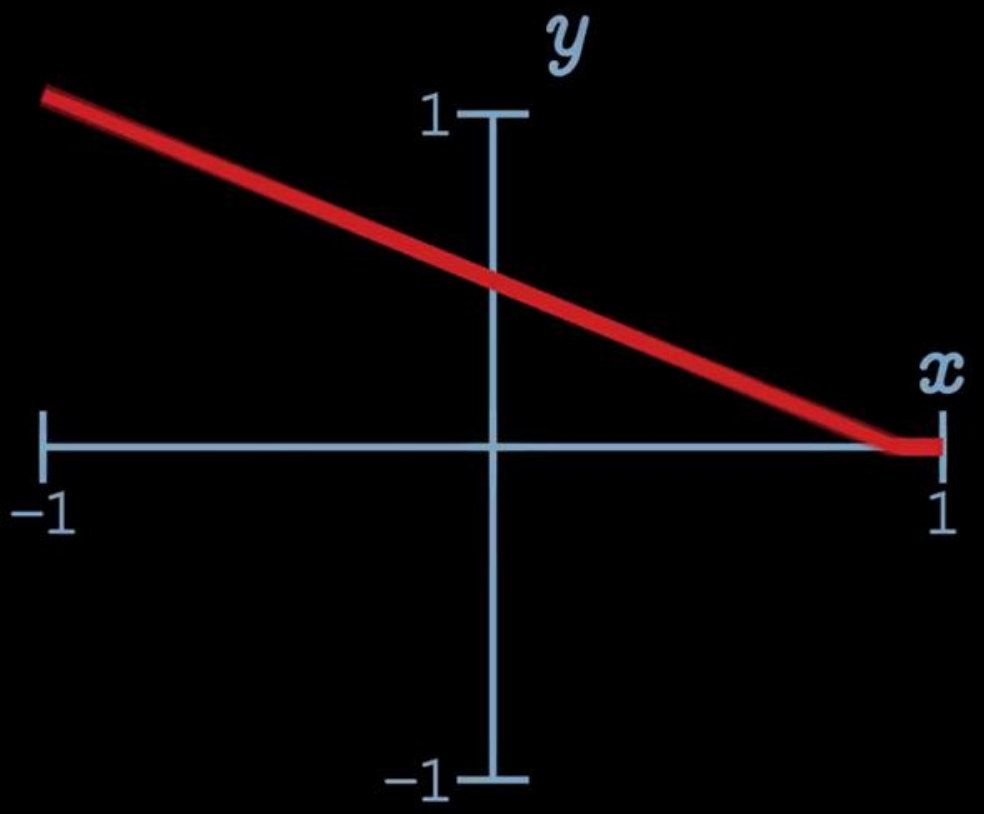


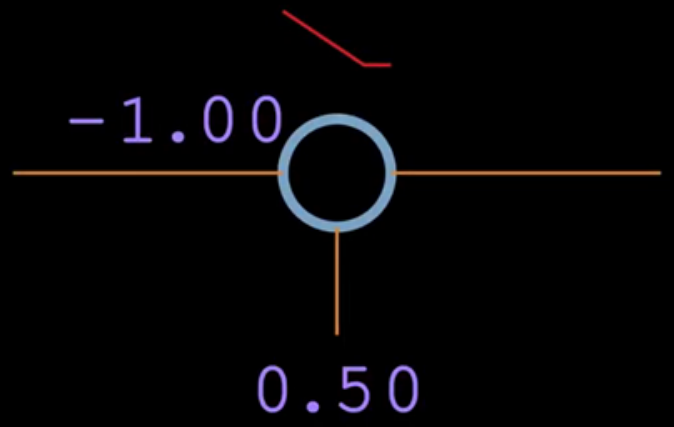


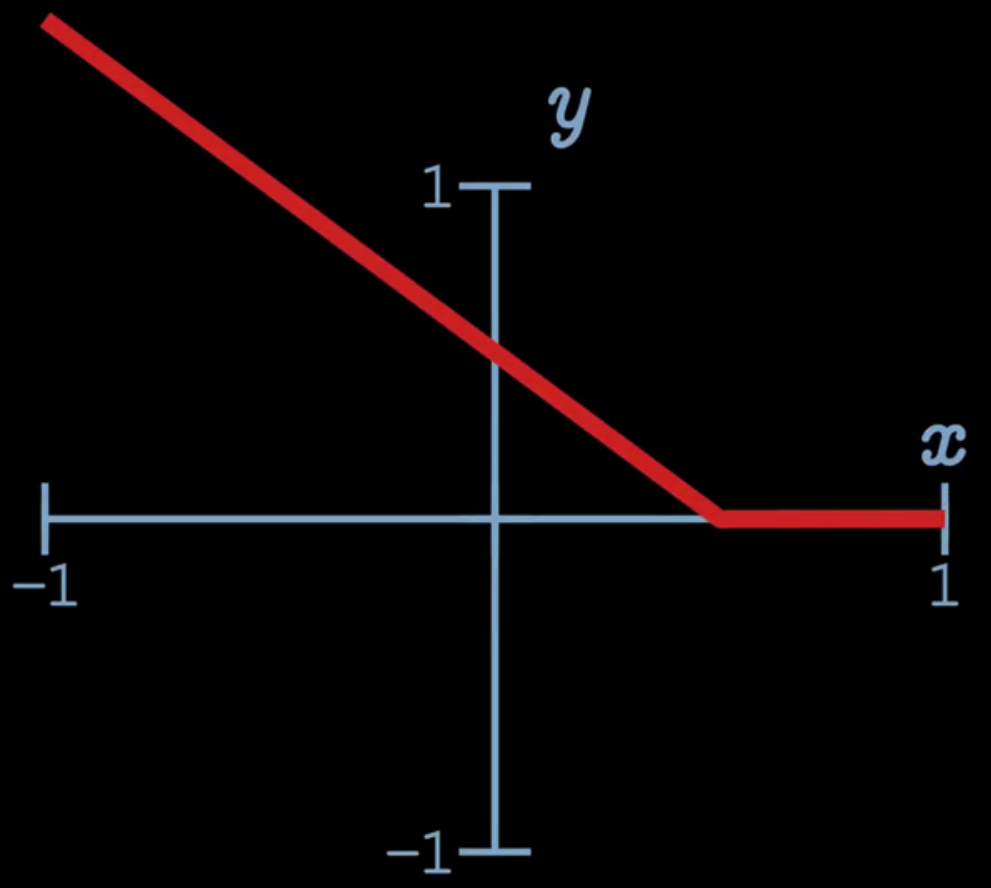


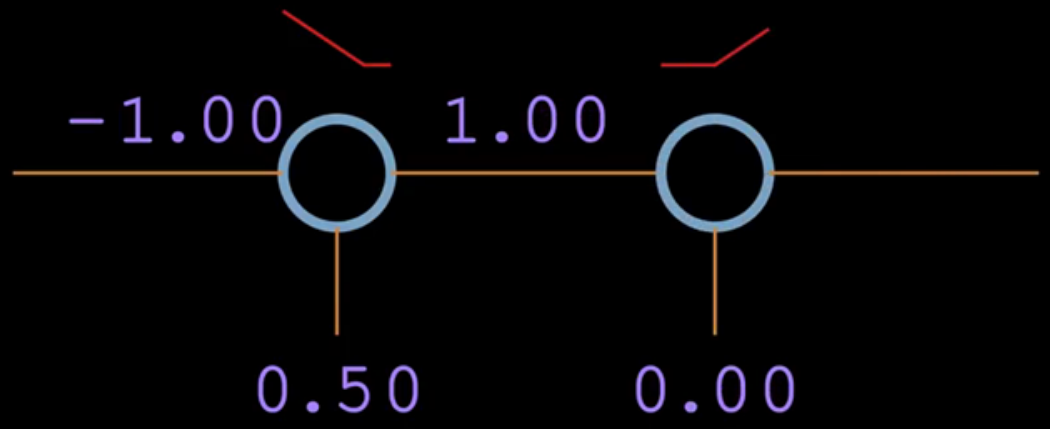


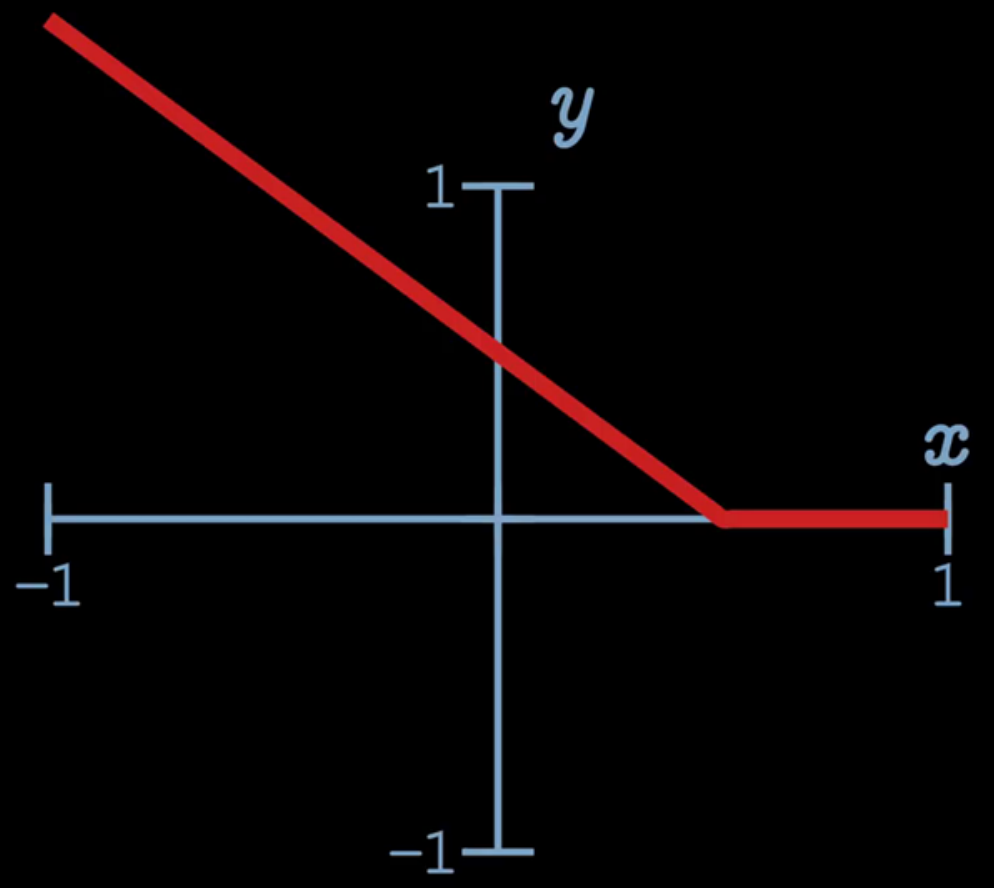


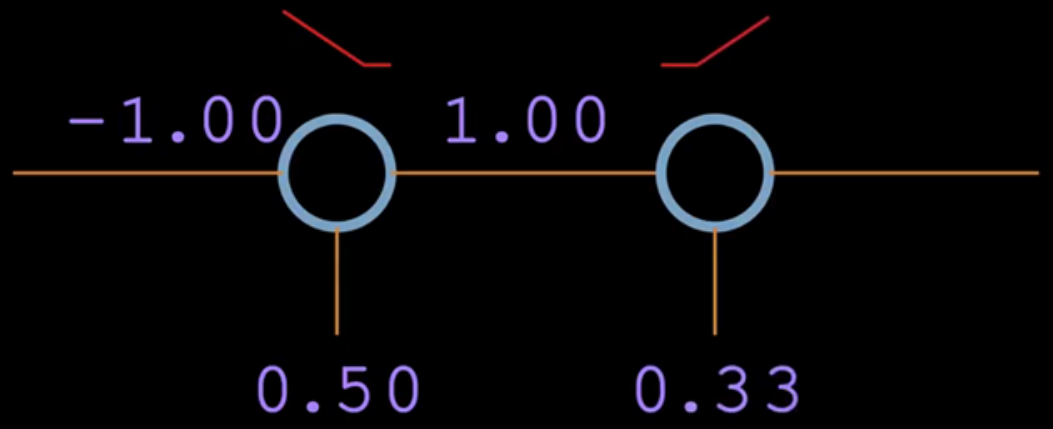


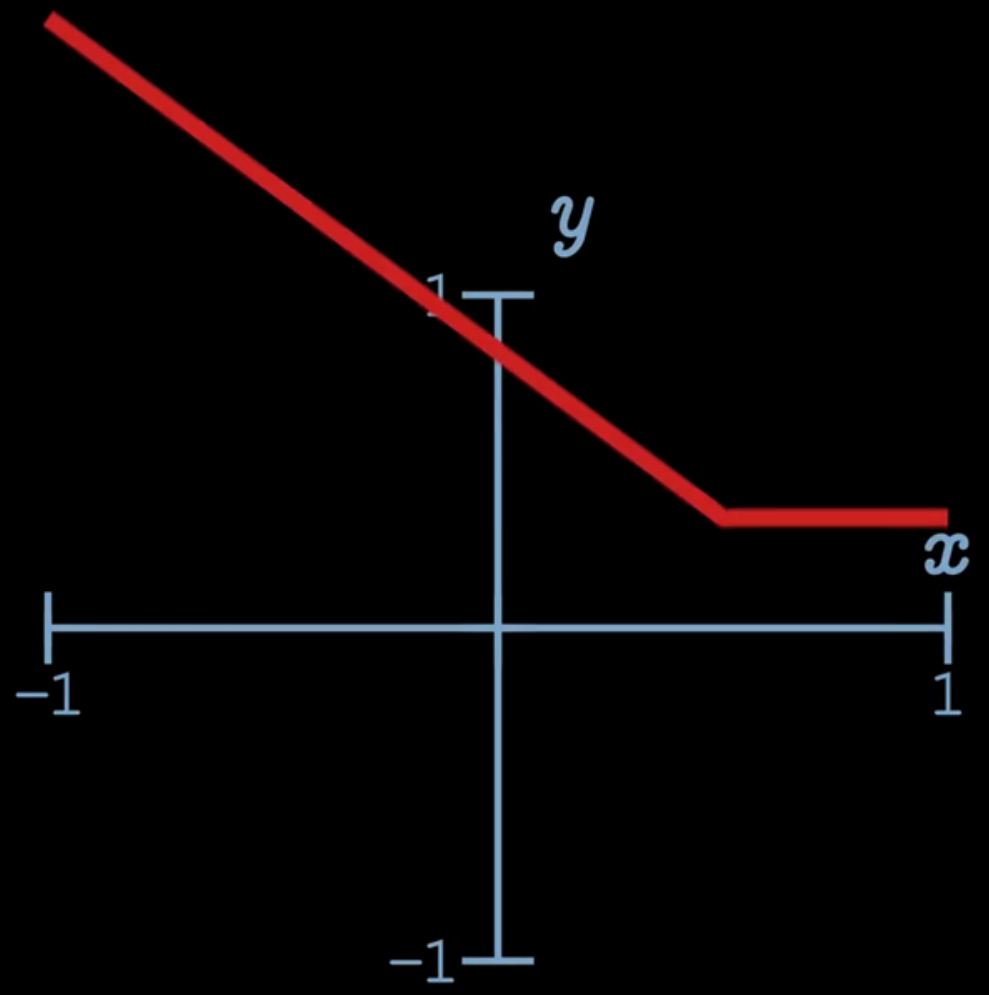


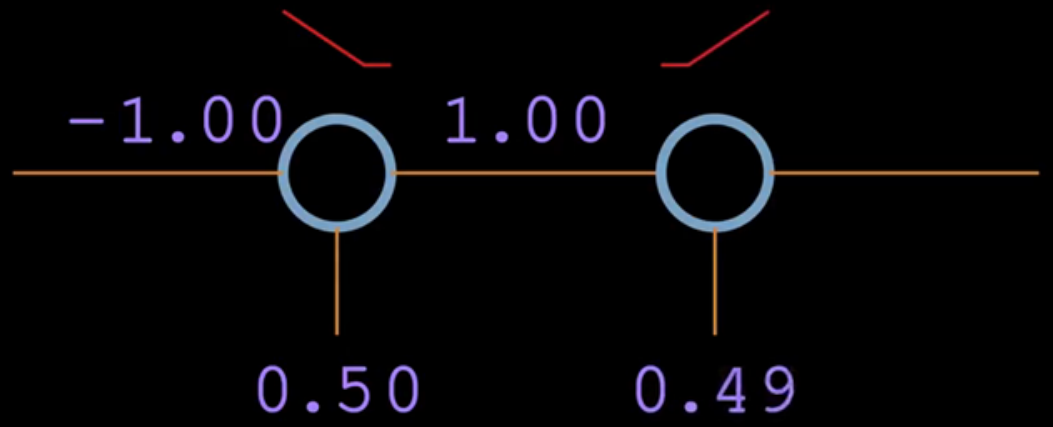


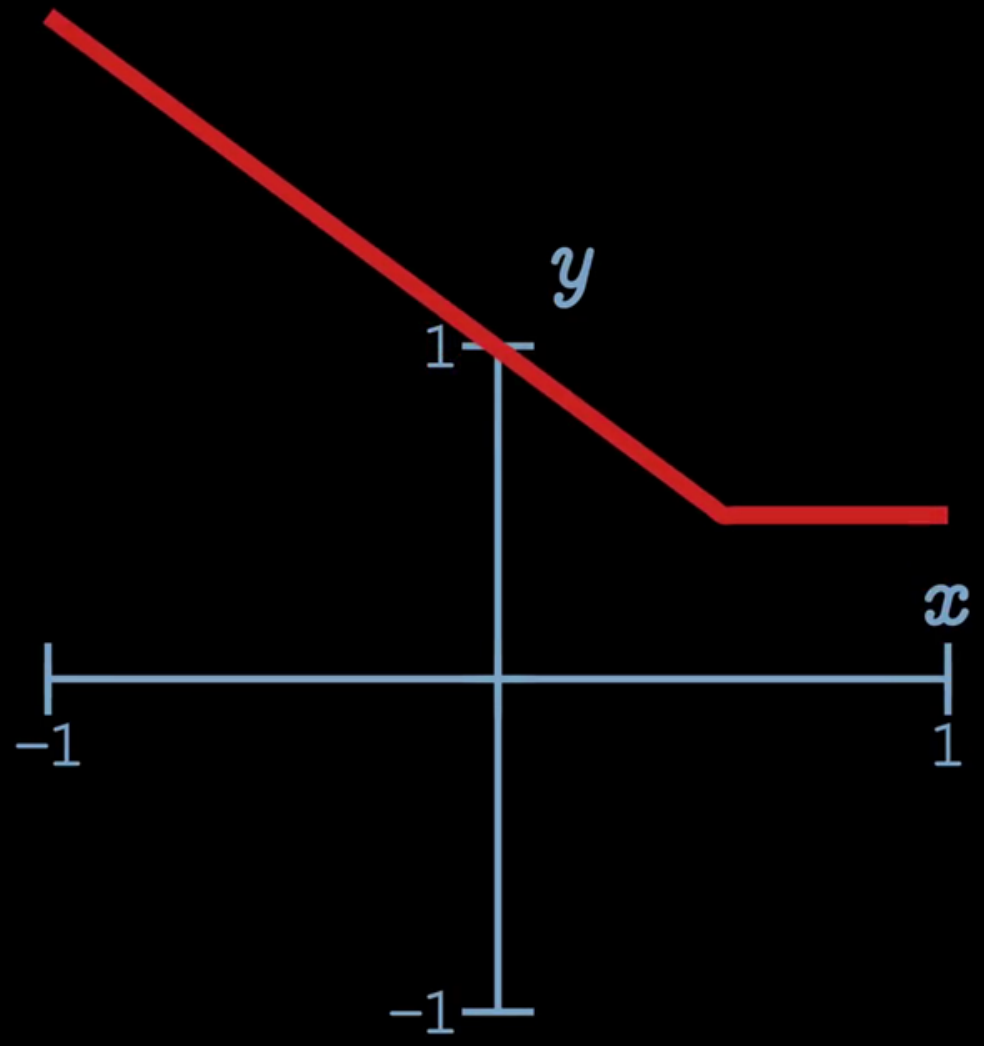


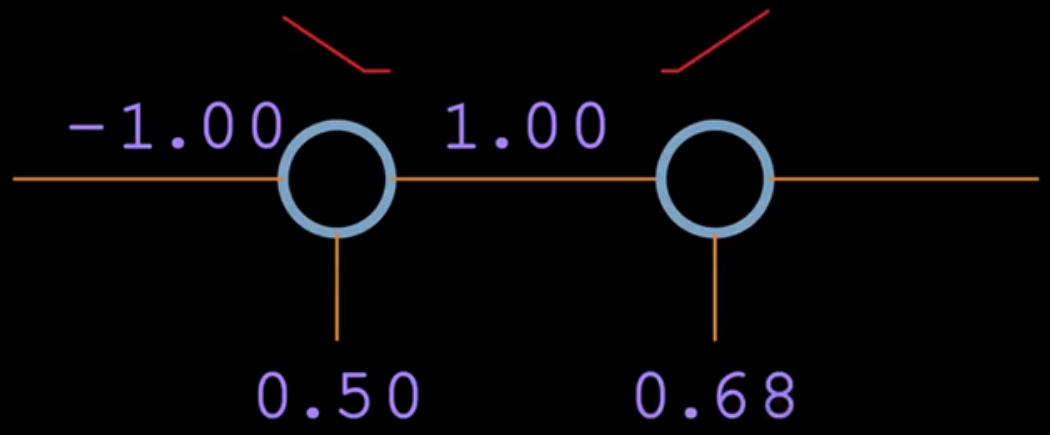


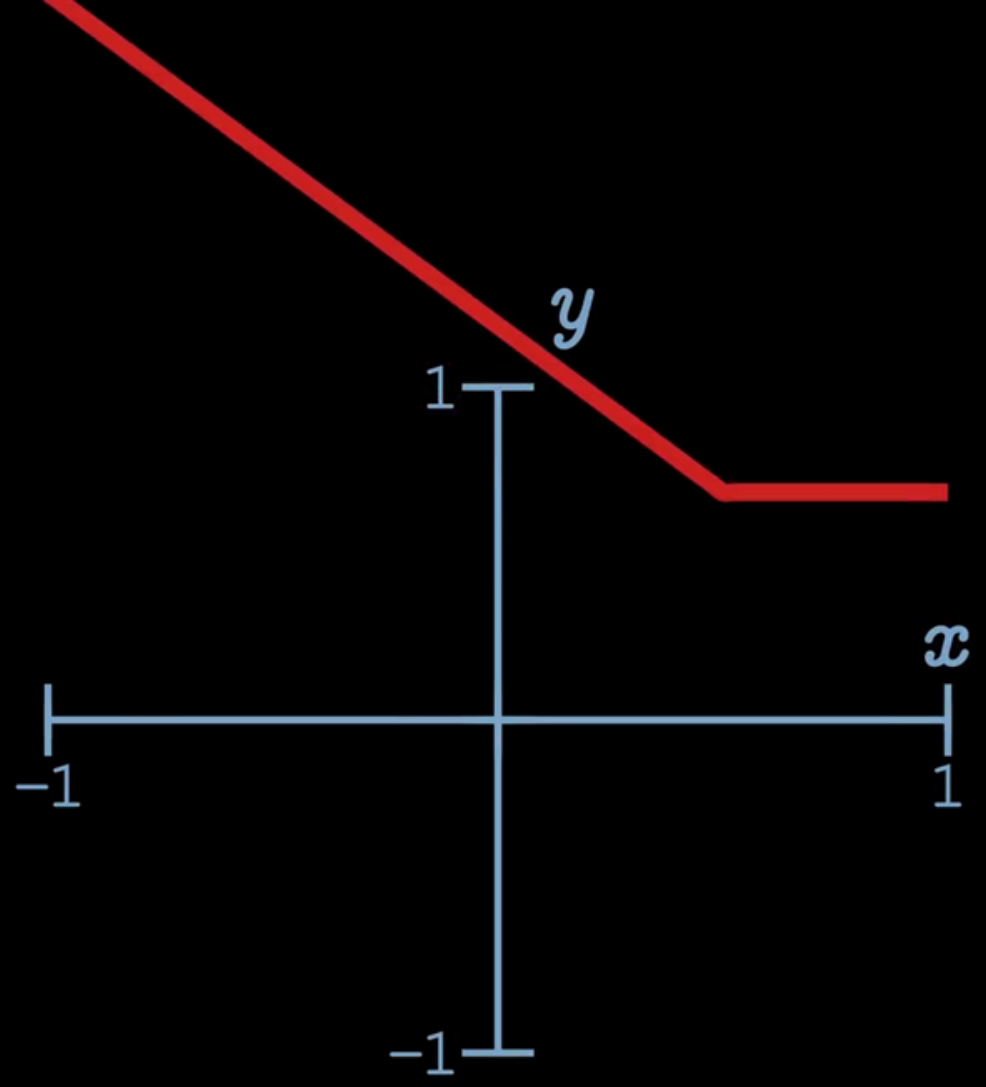


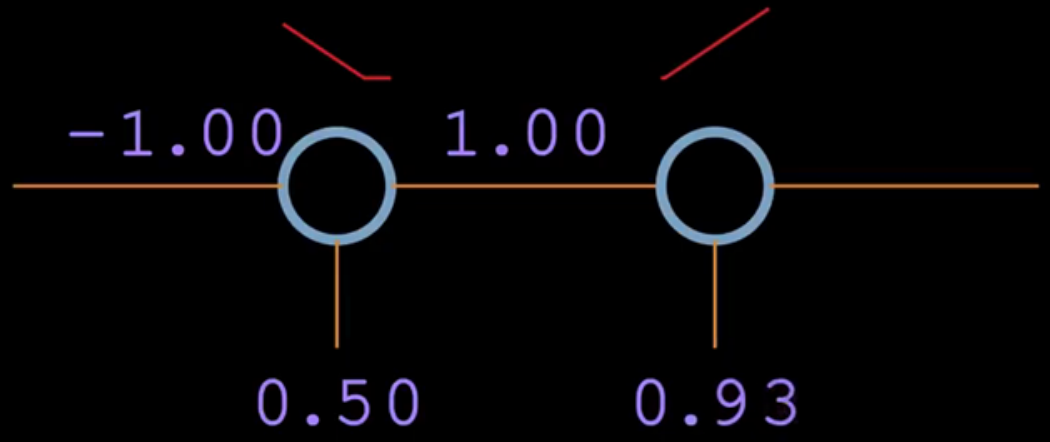


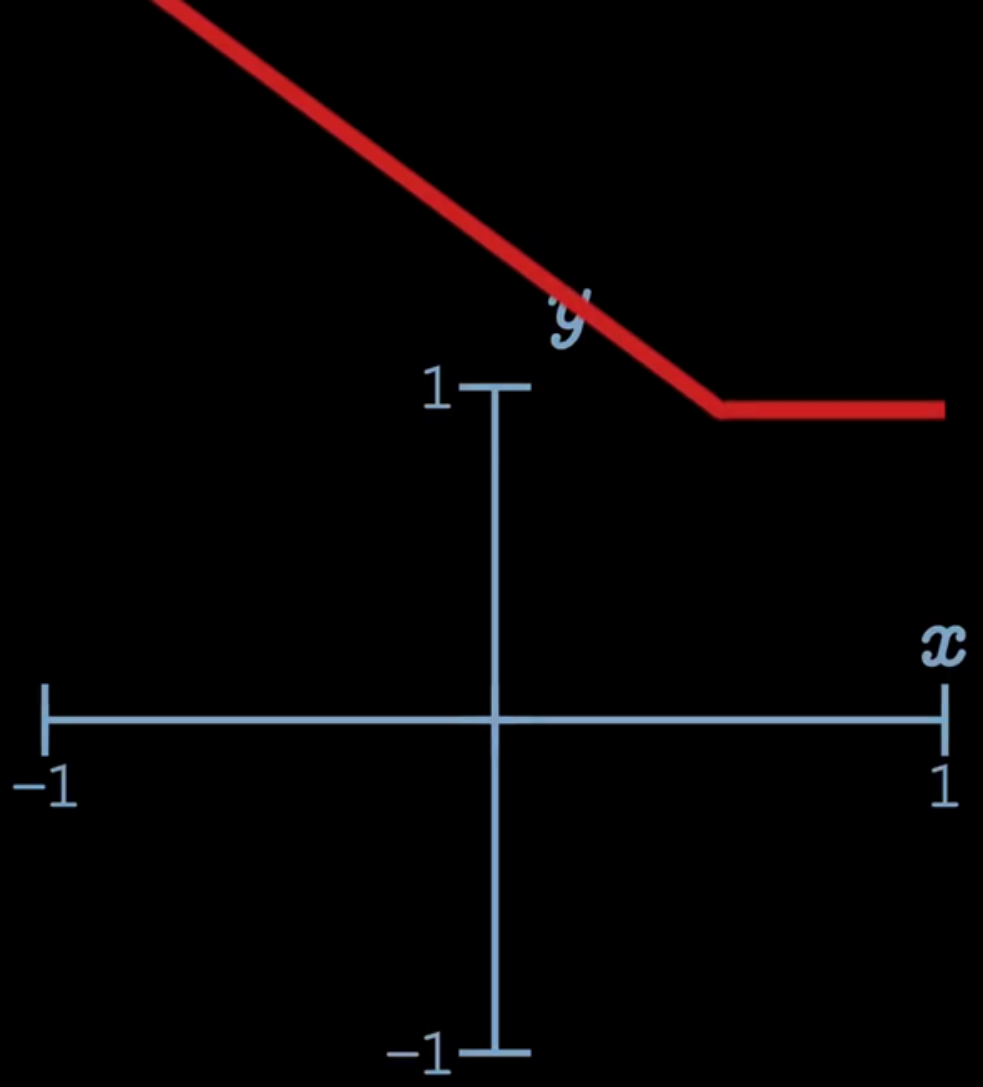


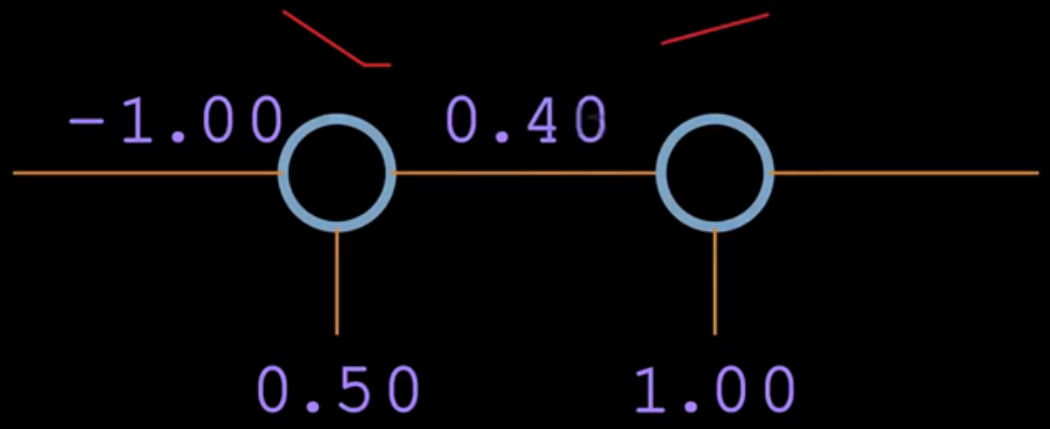


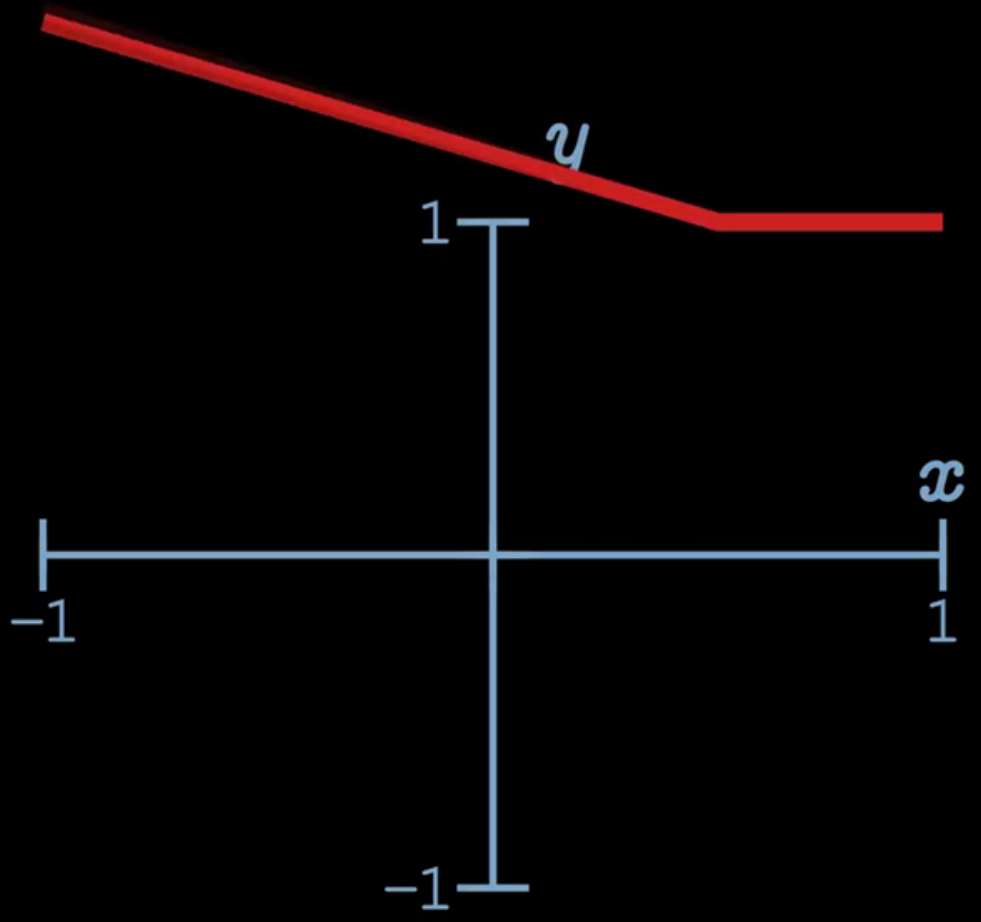


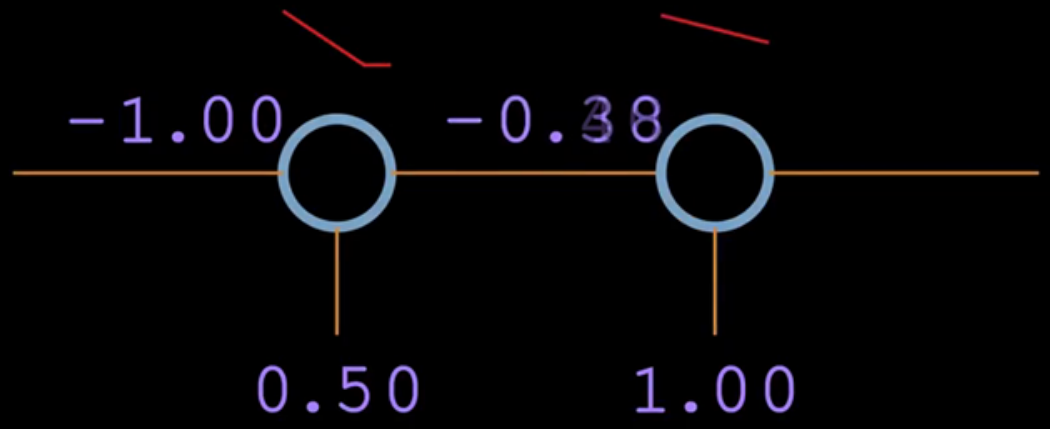


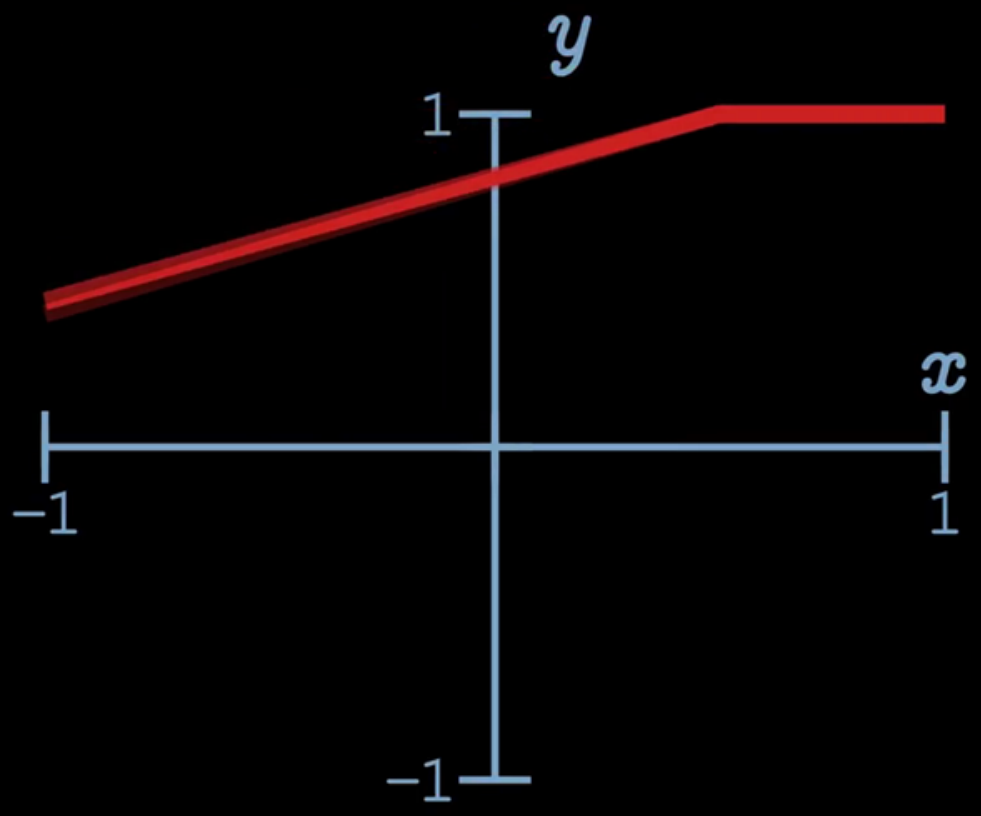


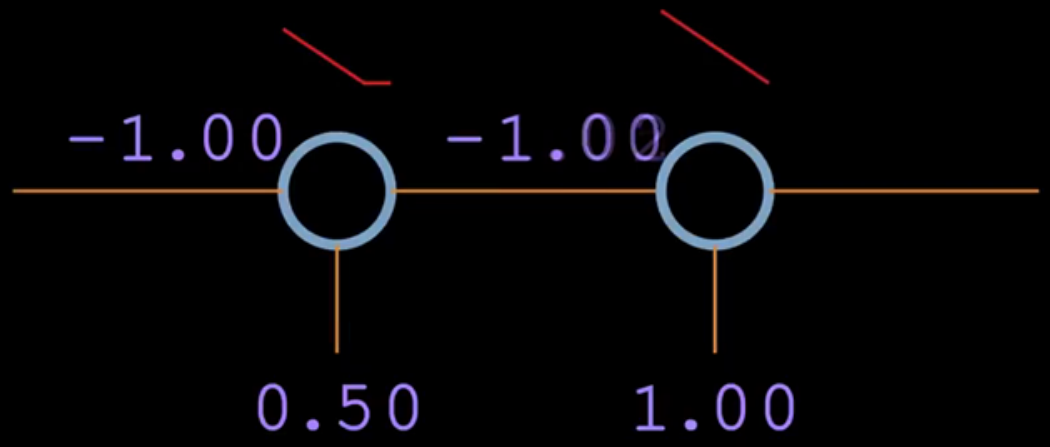


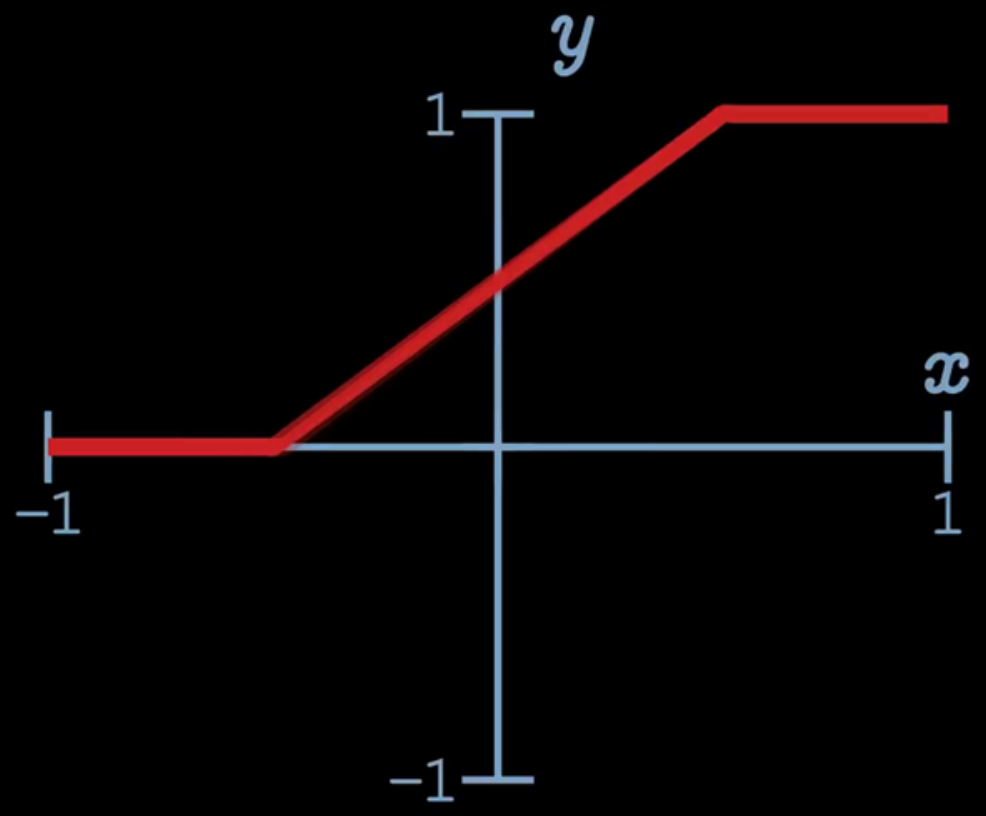


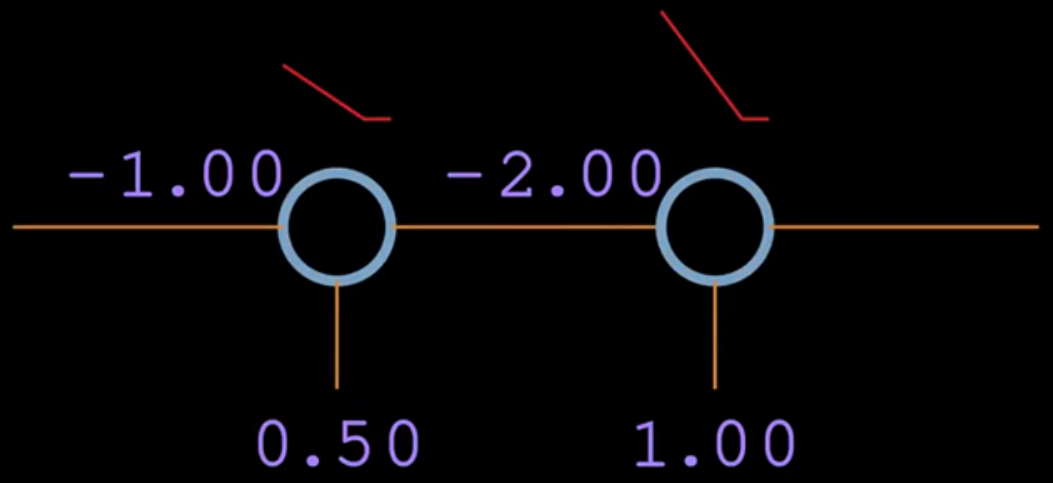


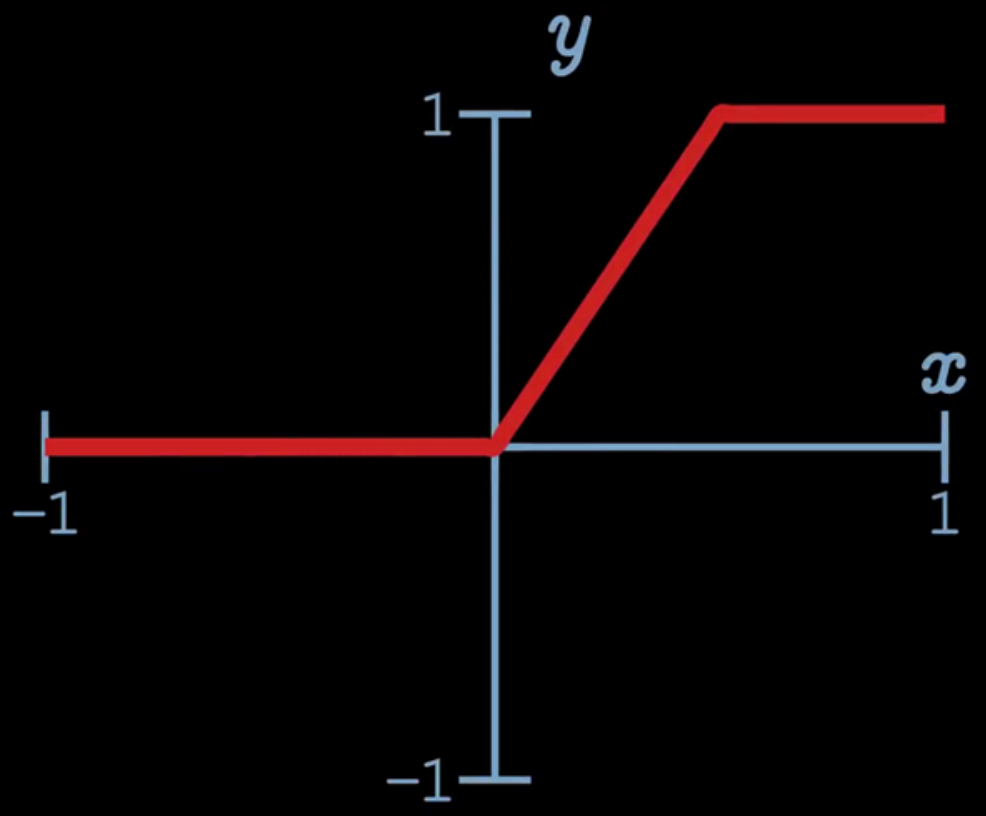


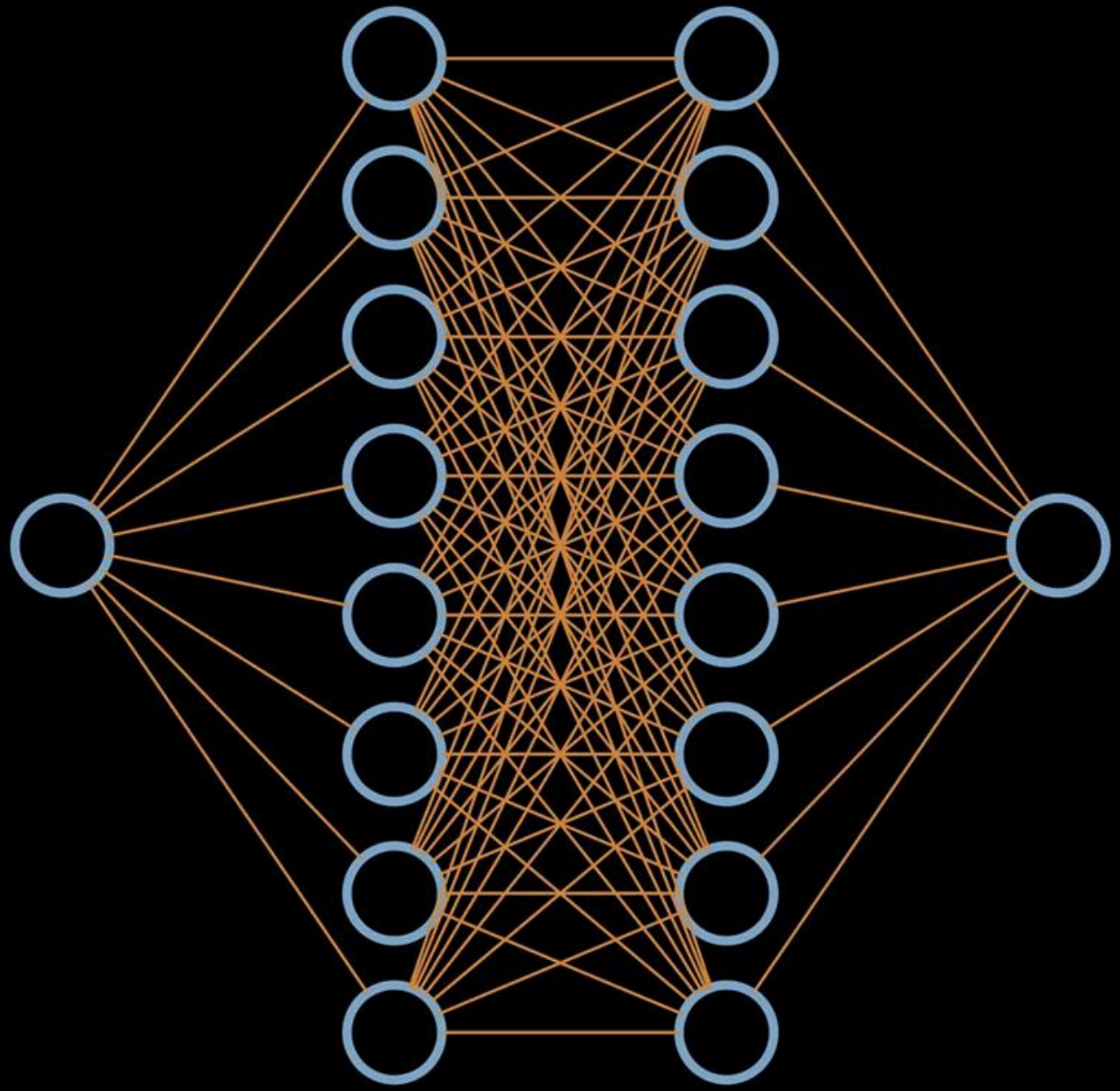




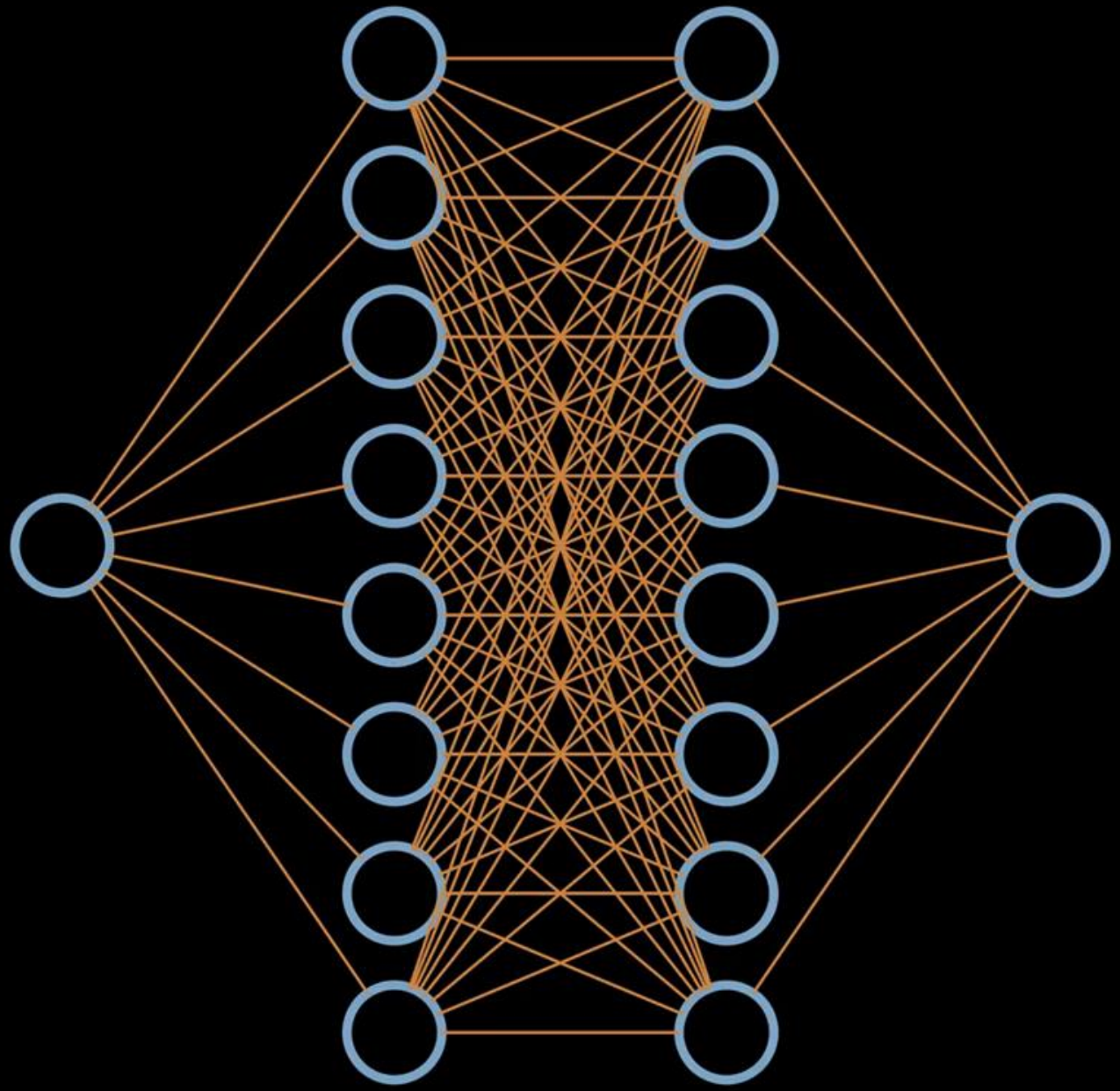


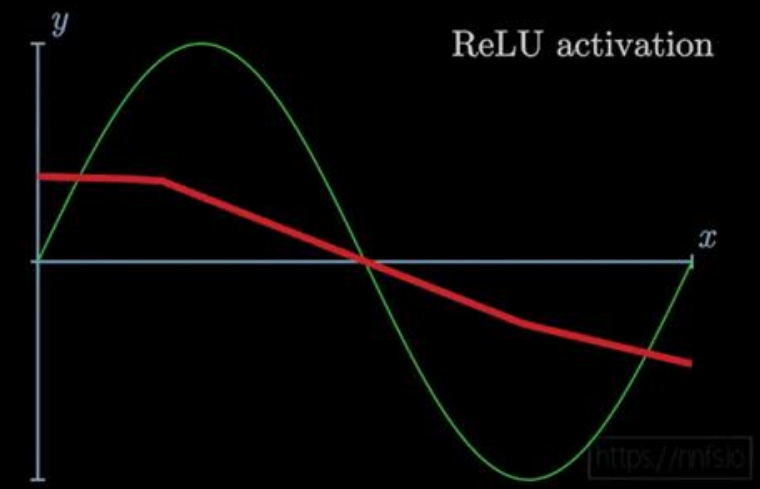


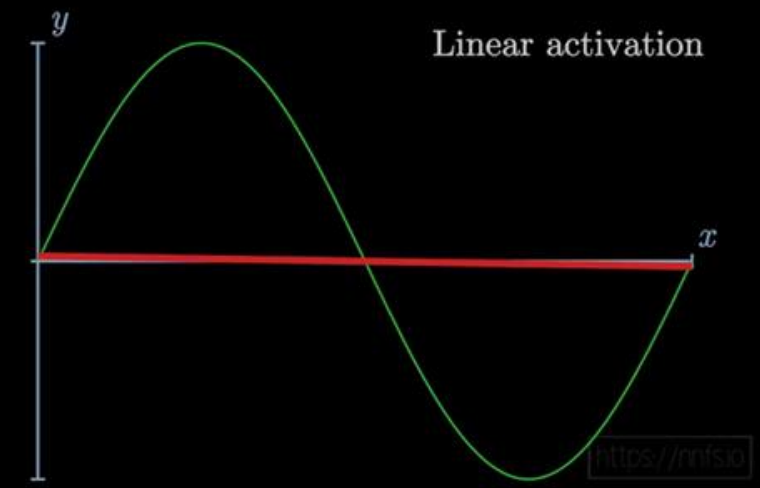


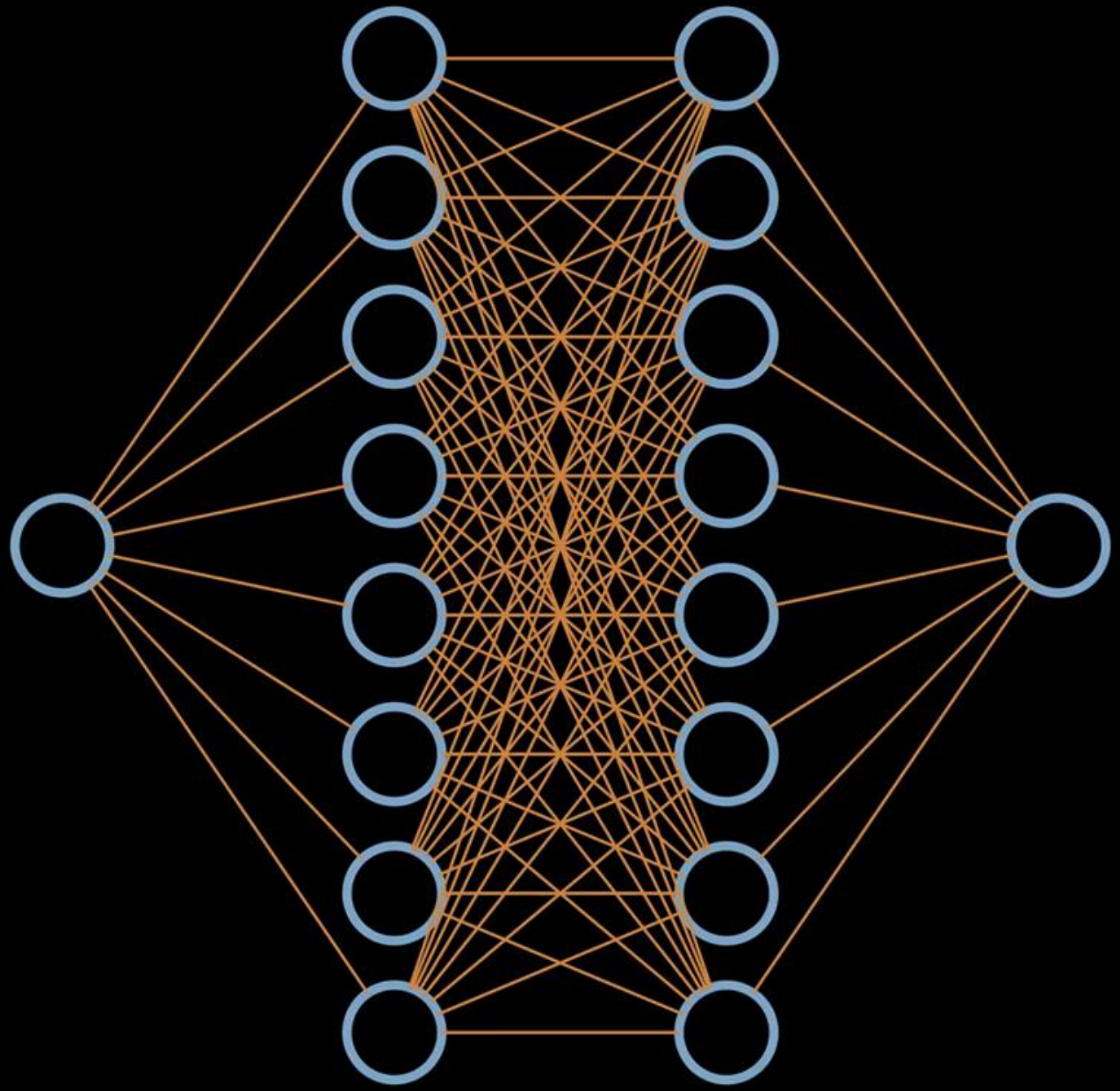


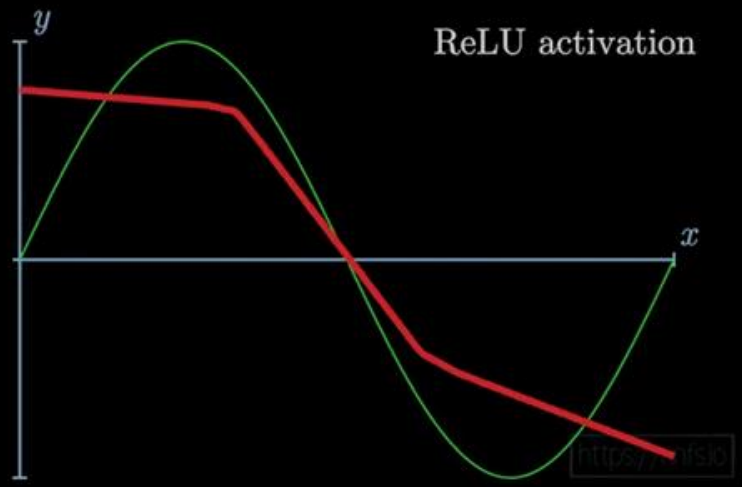
https://nnfs.io

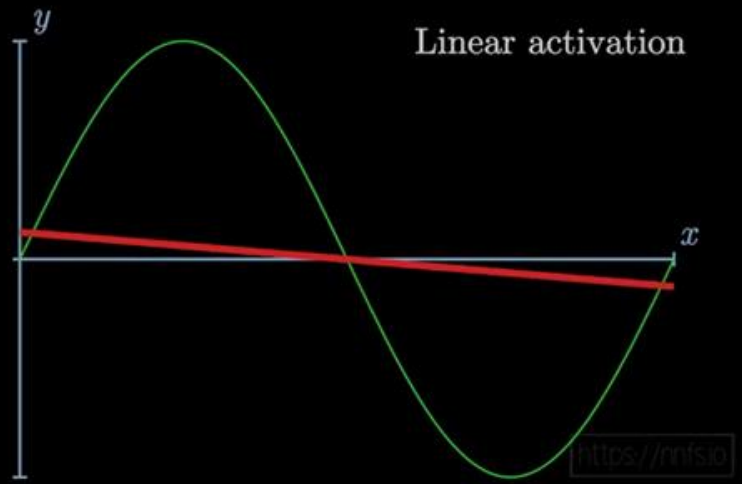




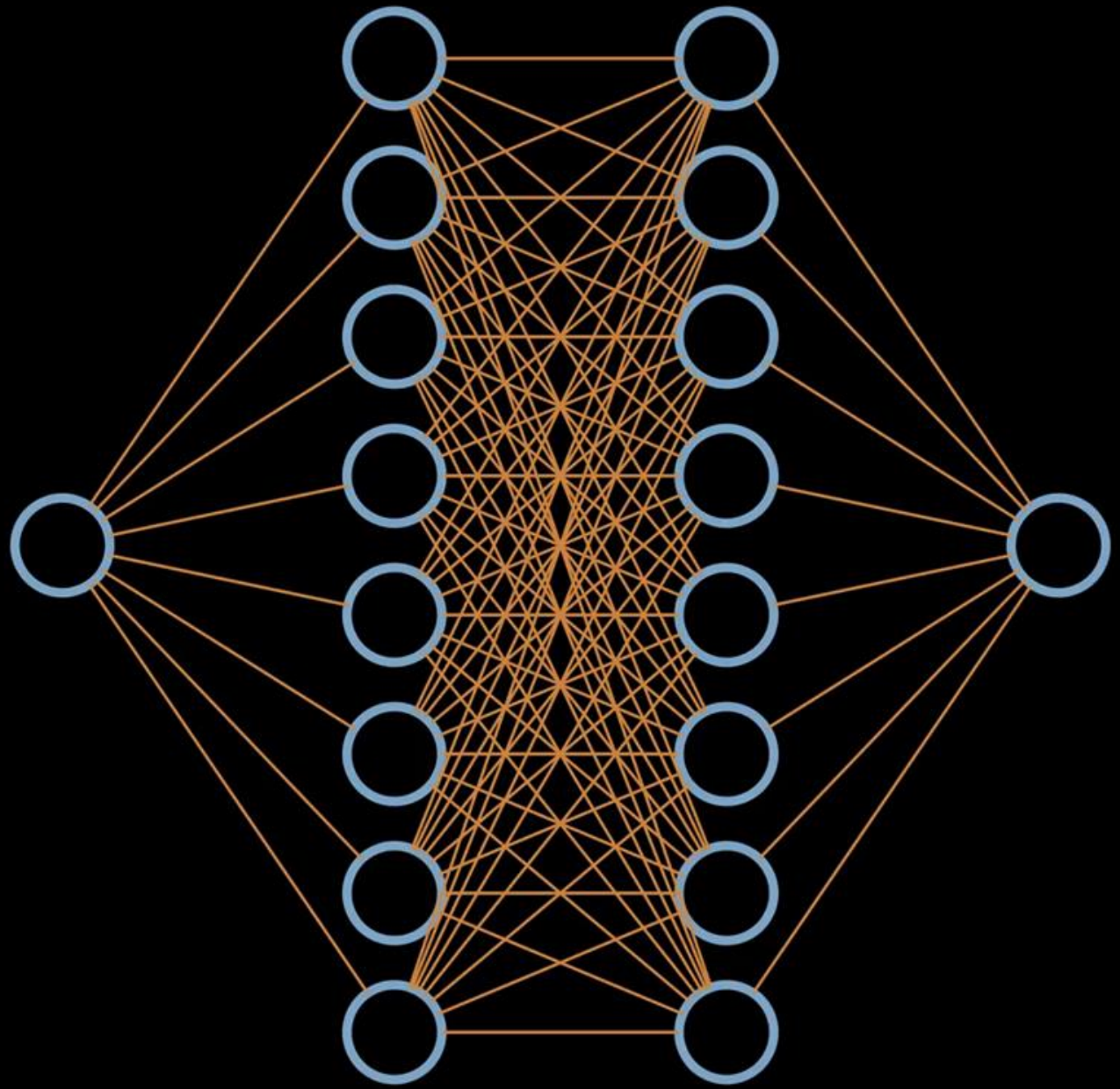


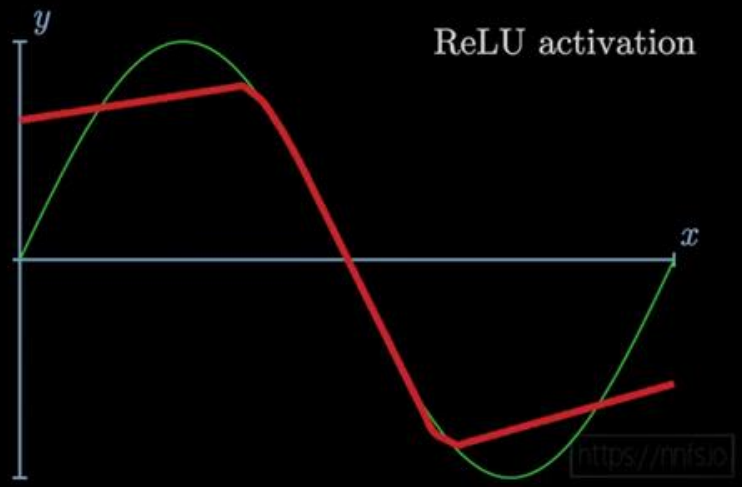


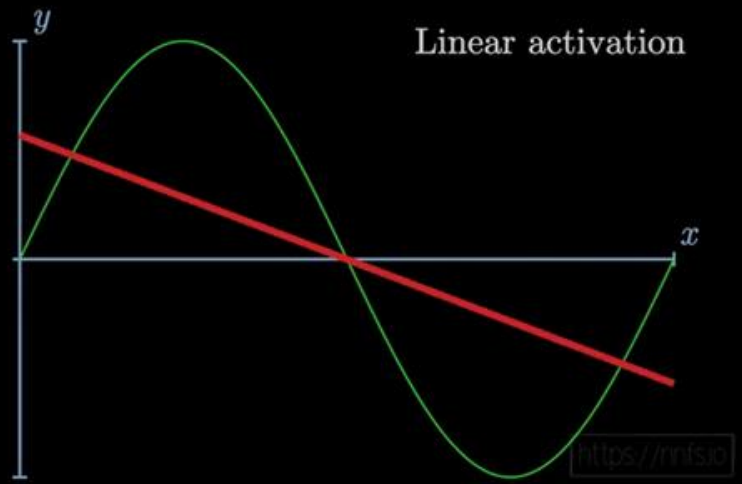


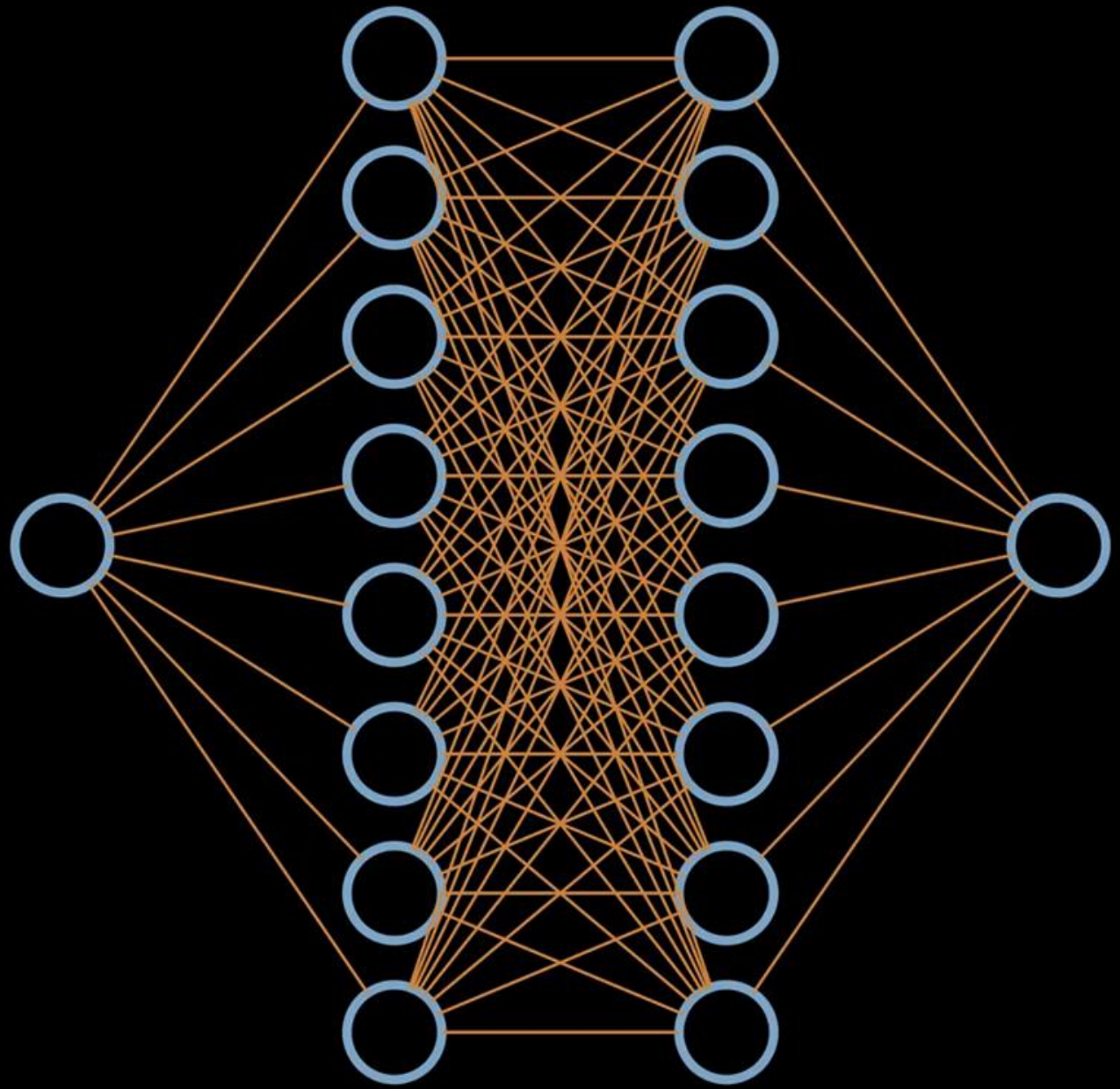


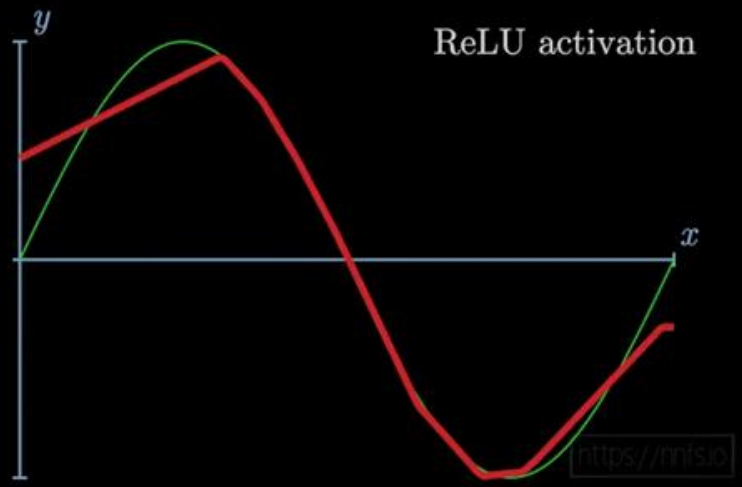
https://nnfs.io

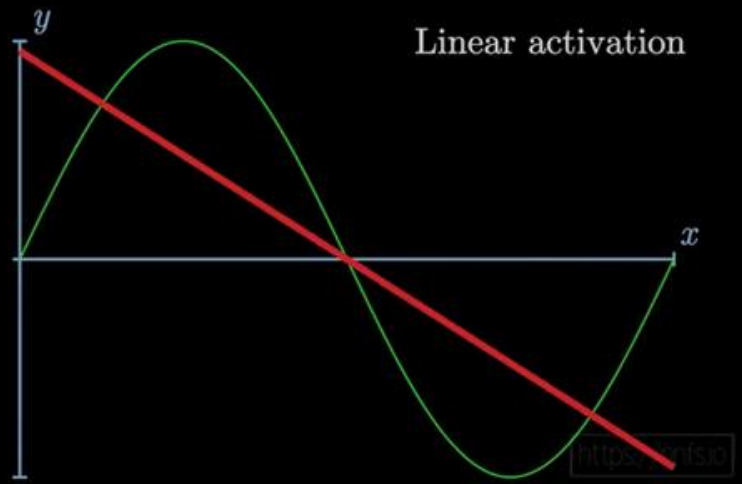


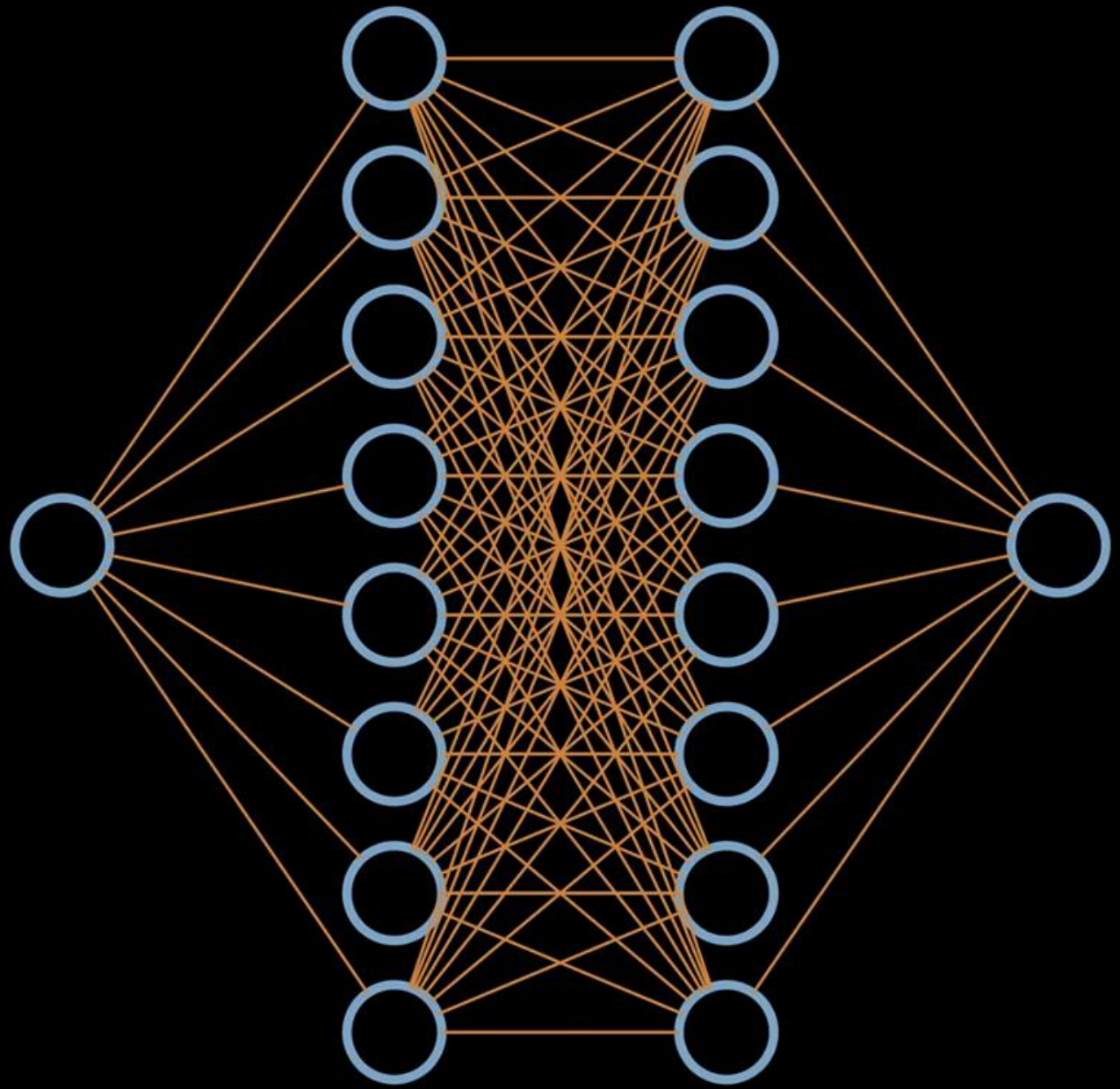


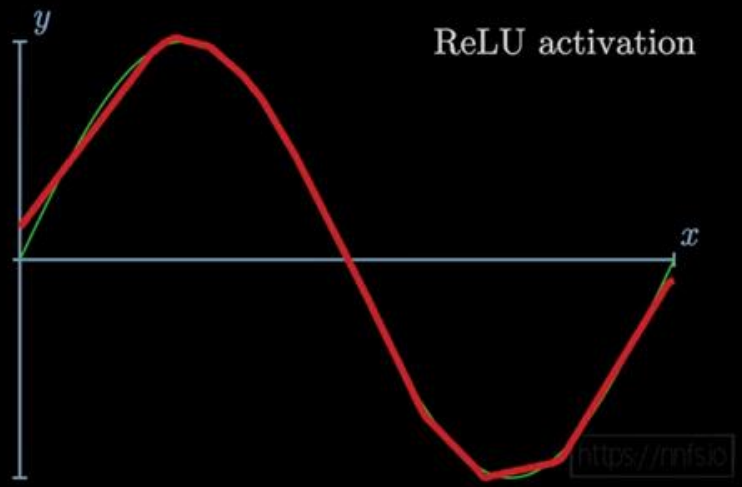


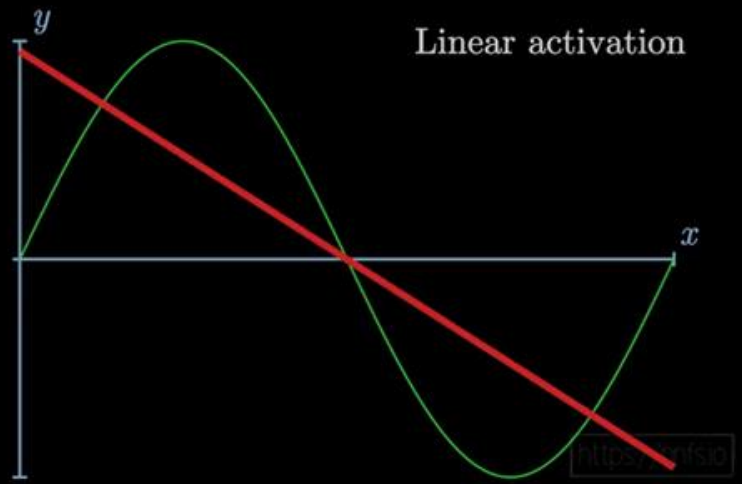




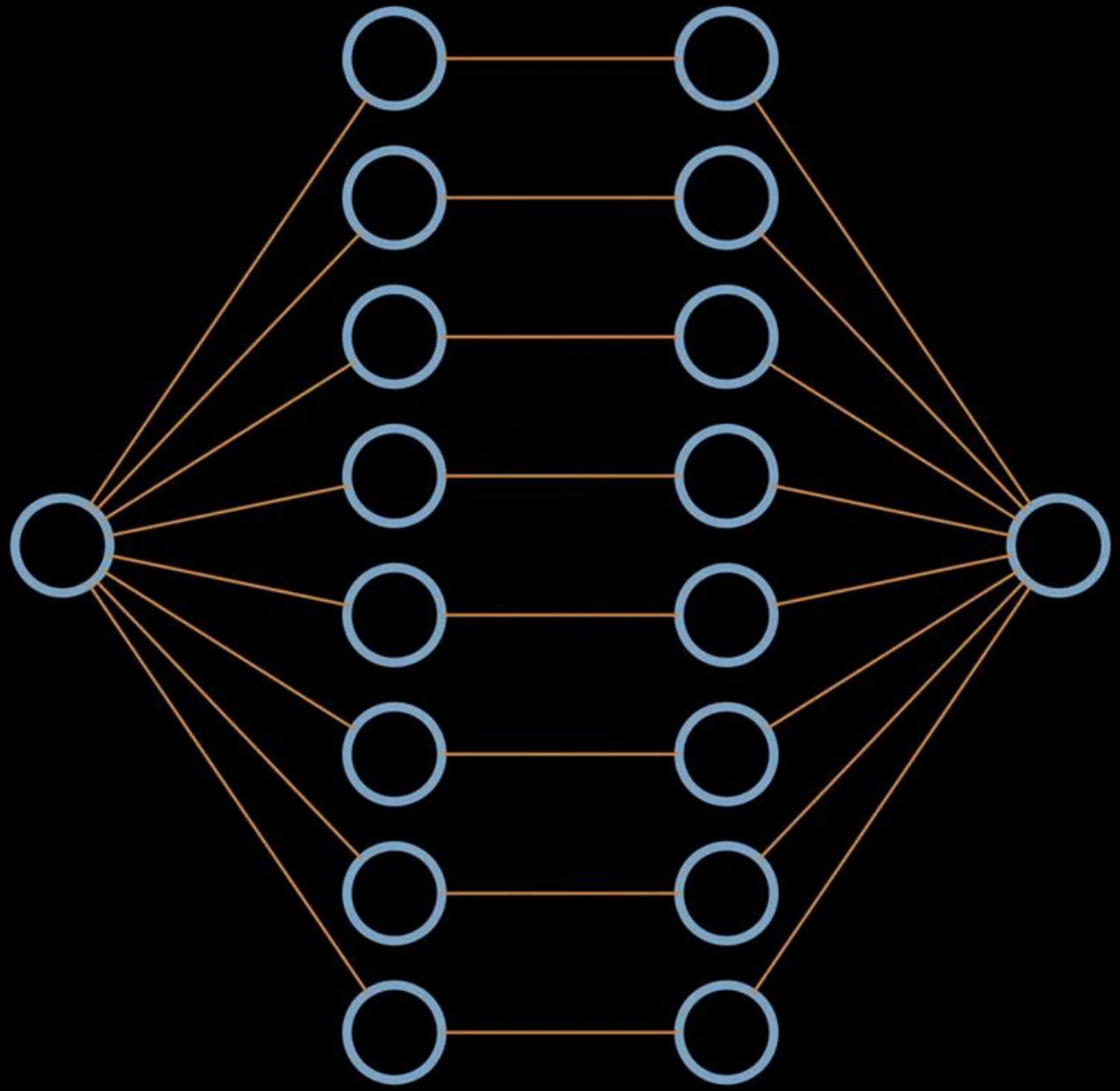




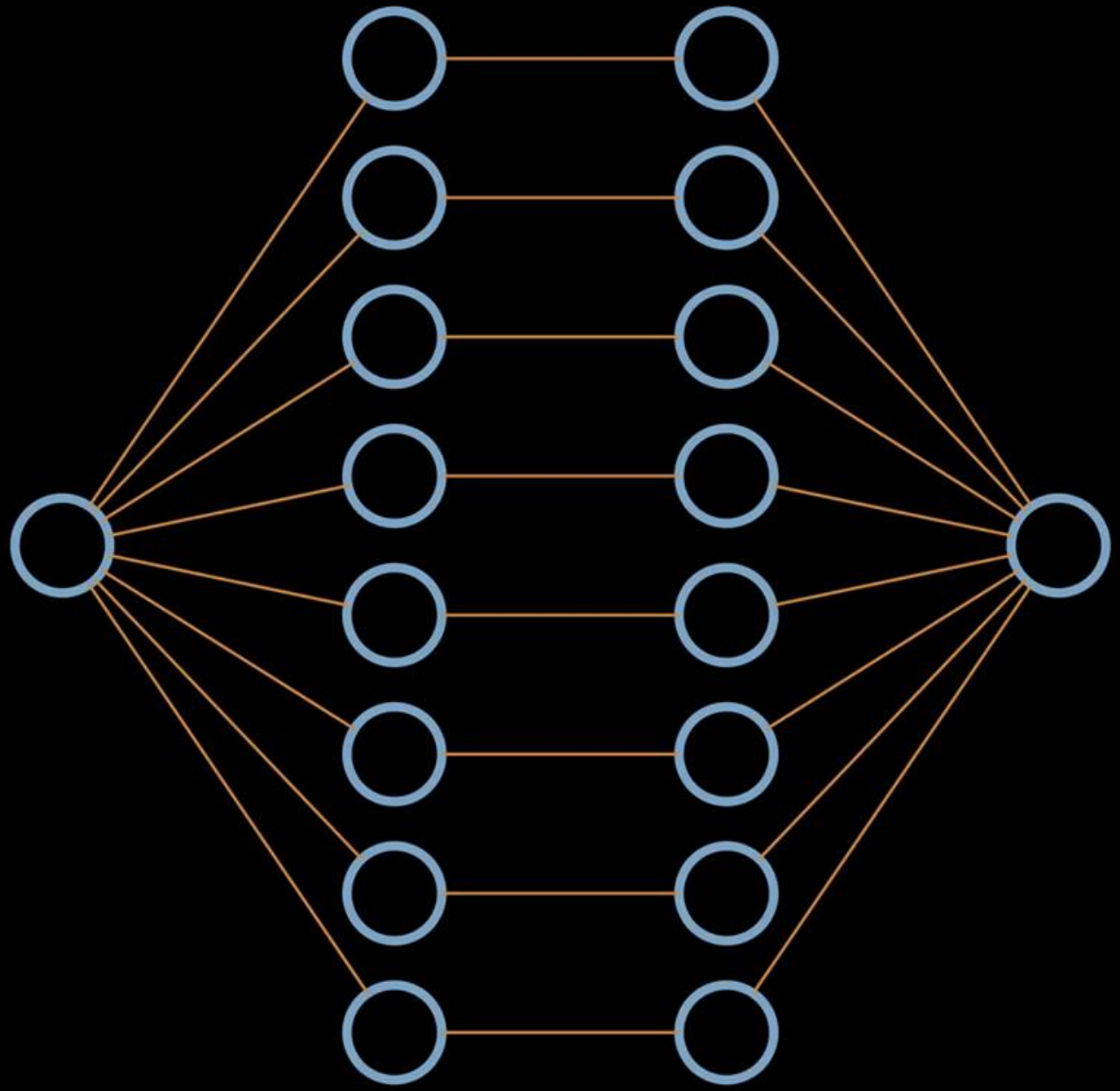


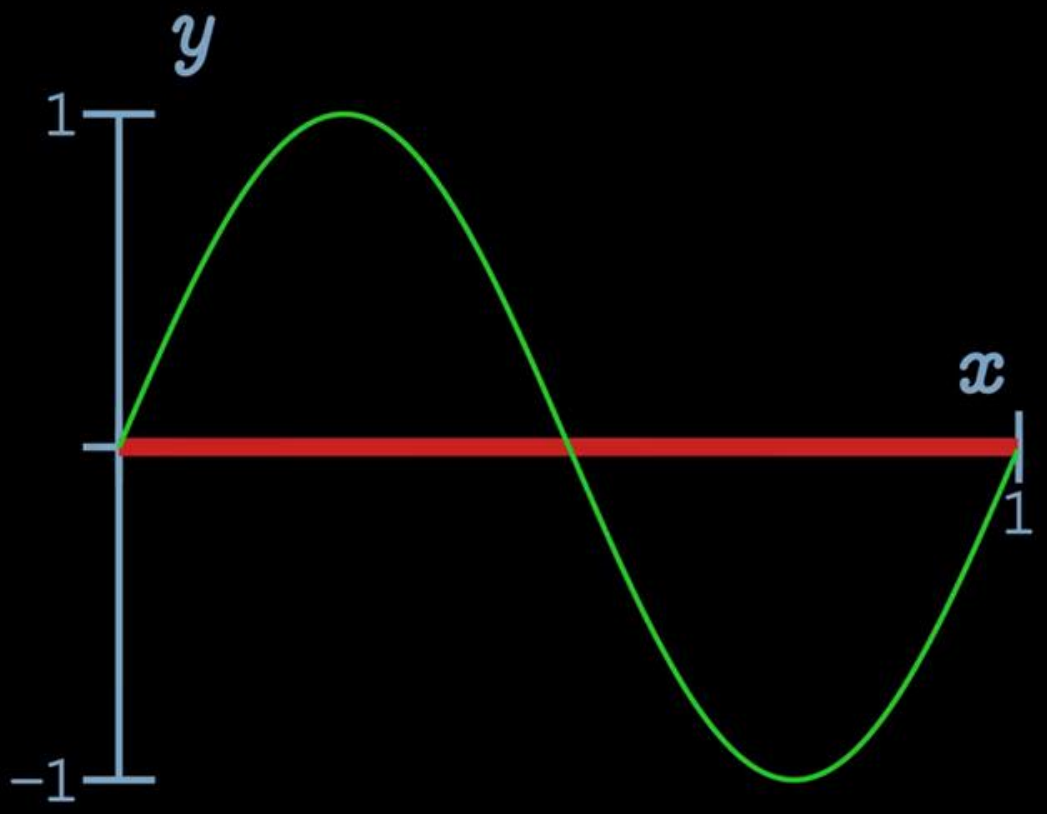


https://nnfs.io

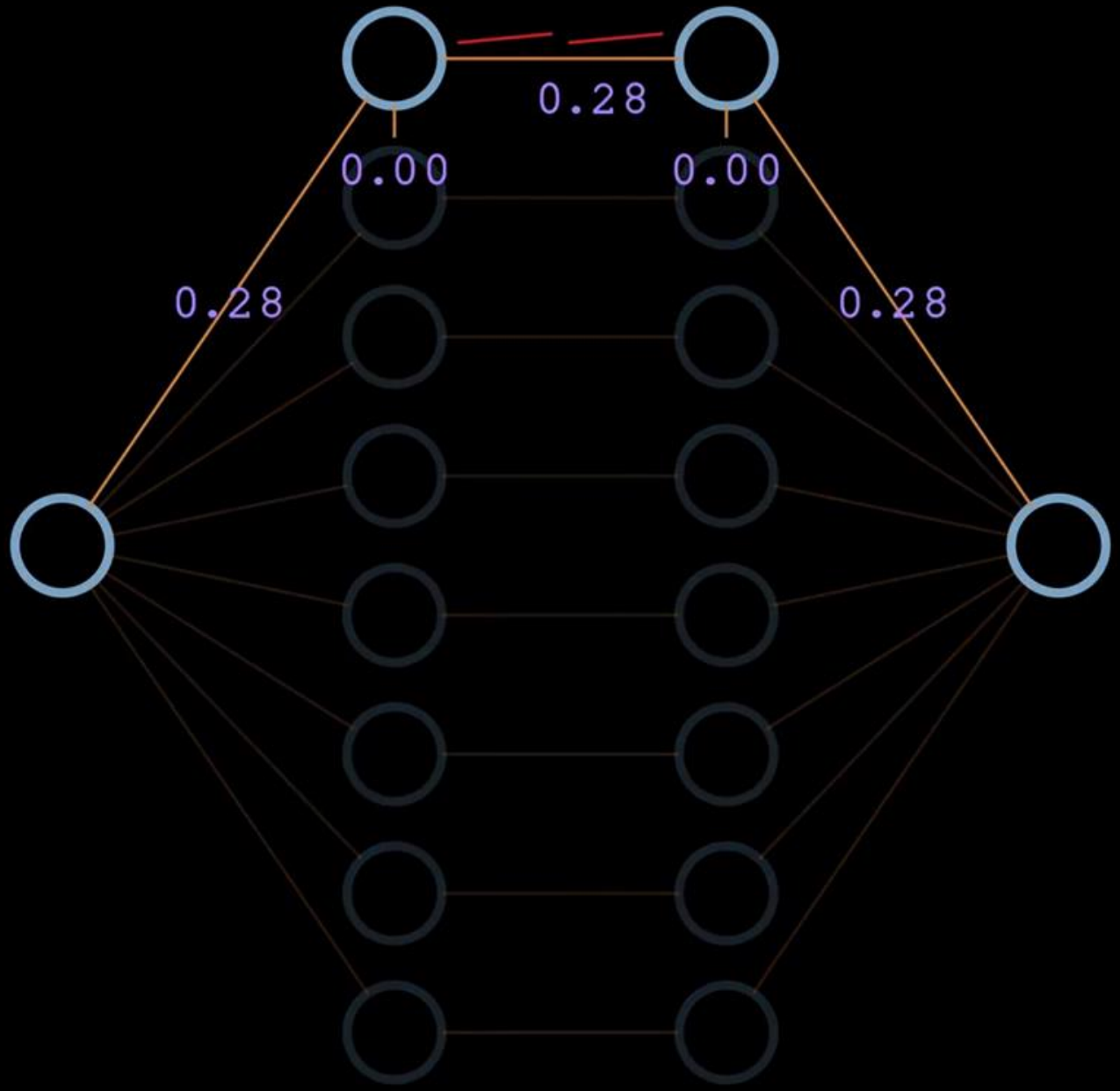


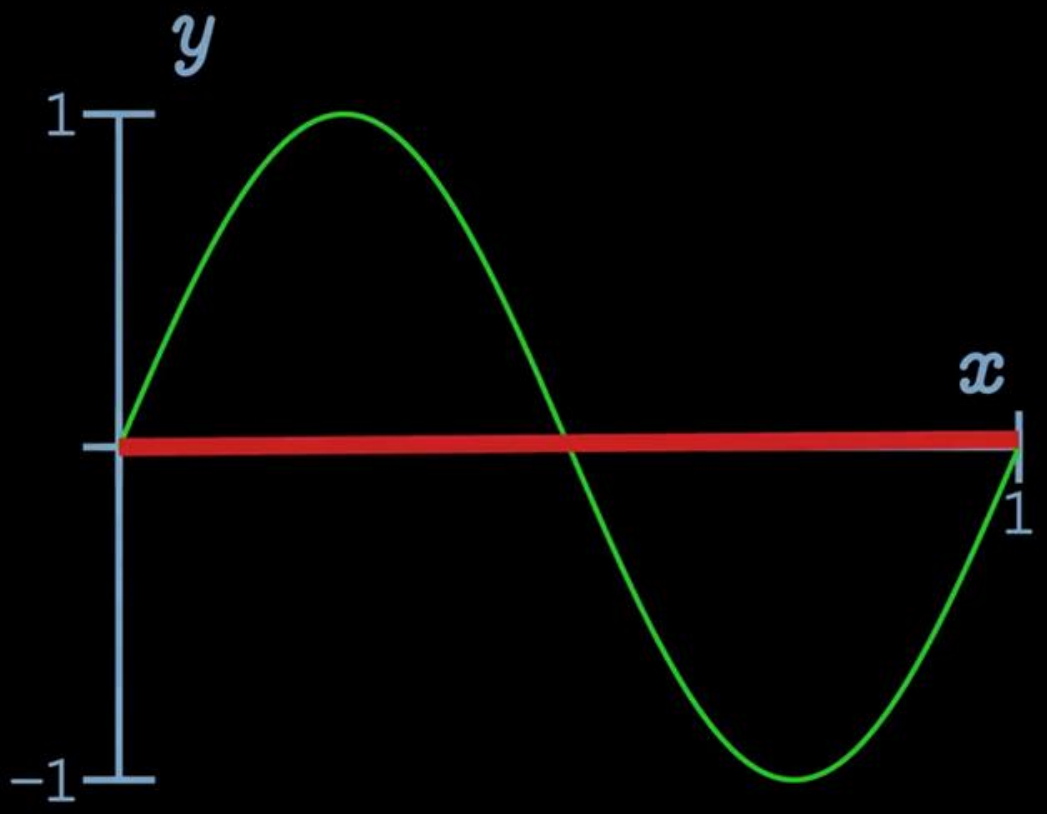
https://nnfs.io



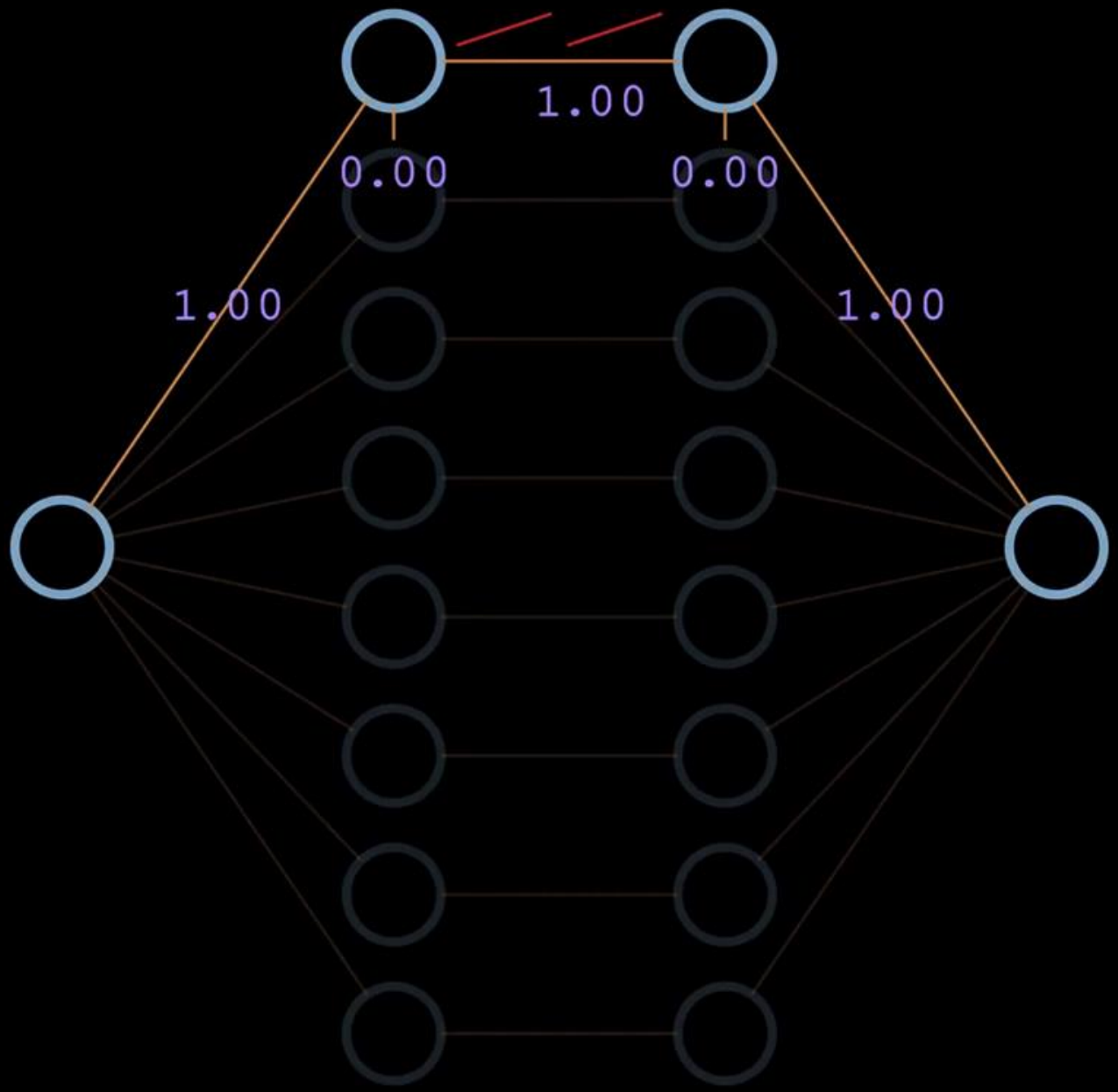


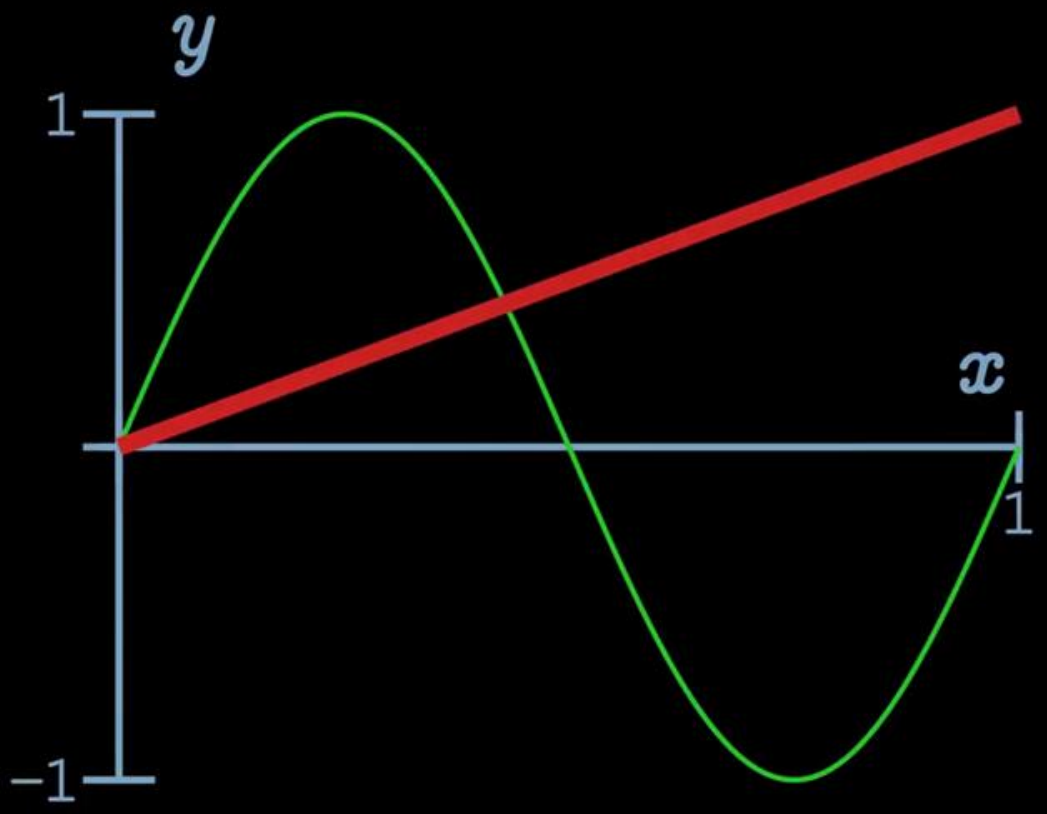
https://nnfs.io



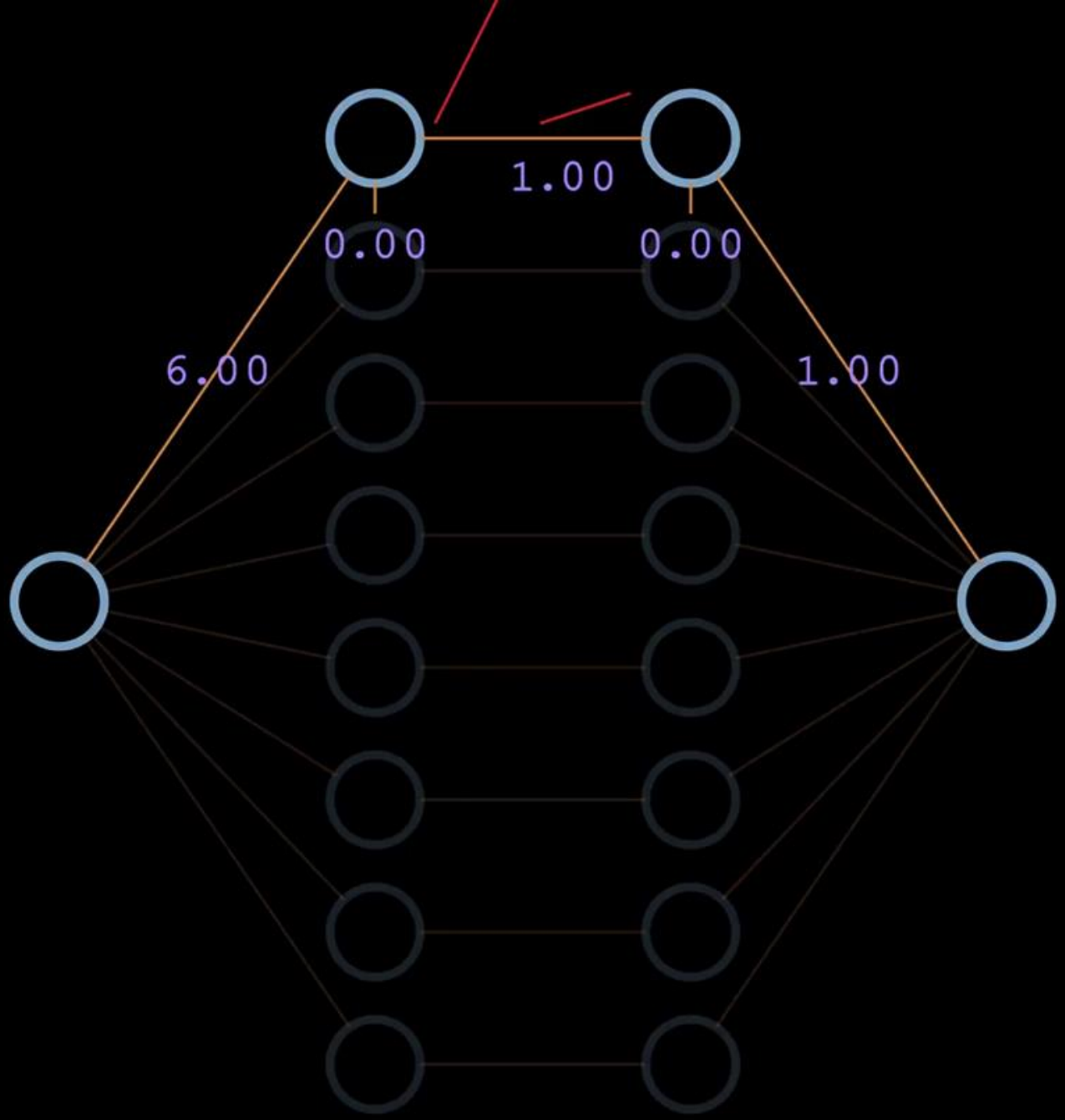


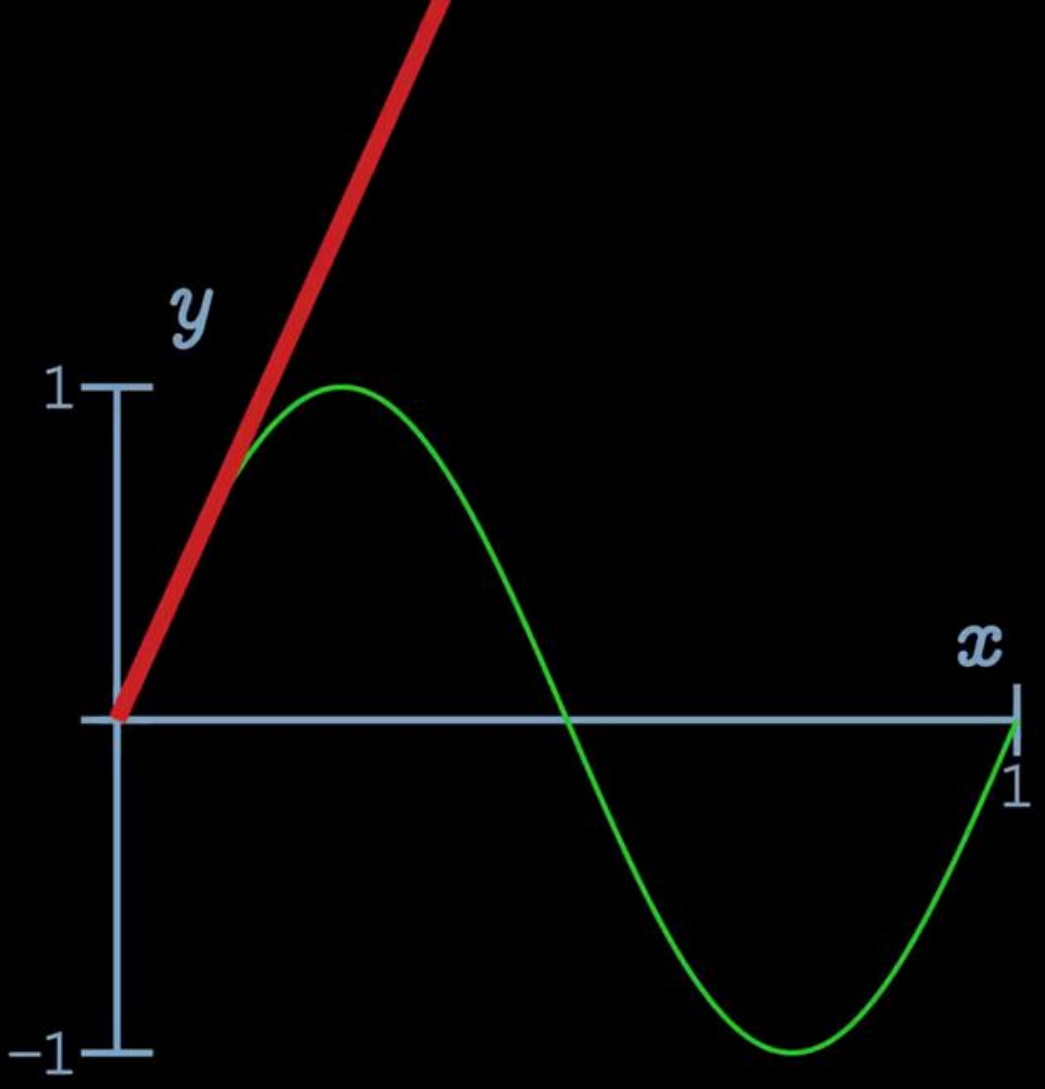
https://nnfs.io



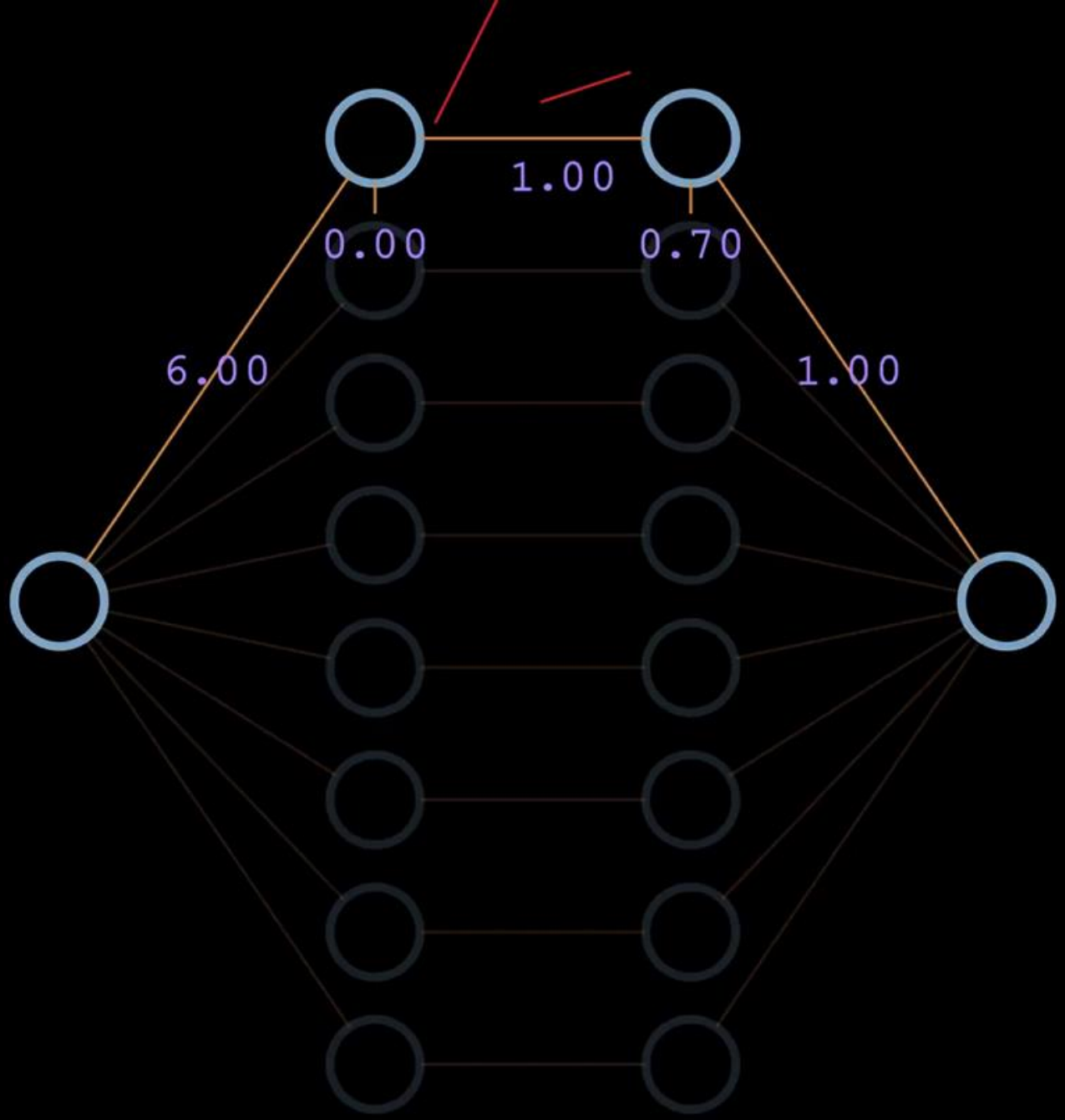


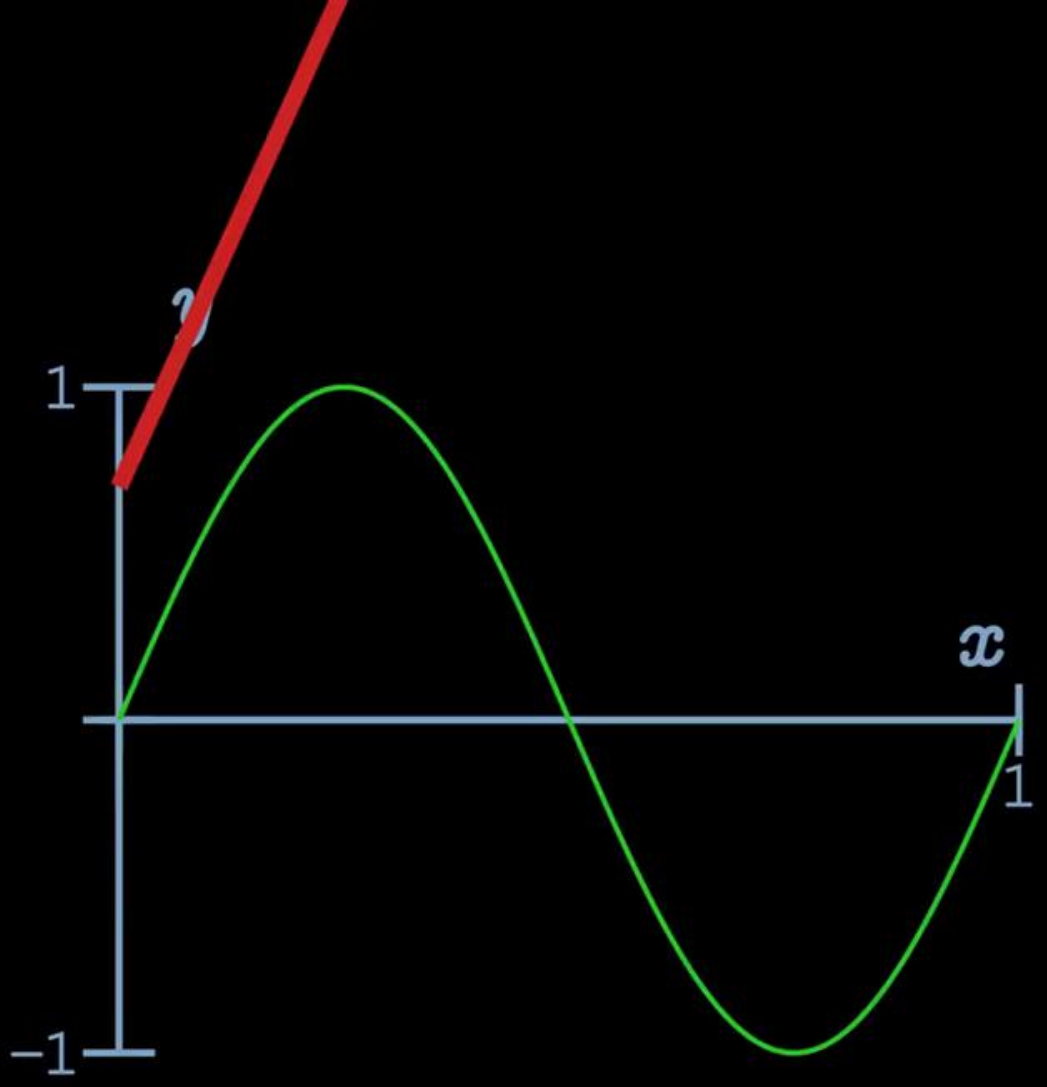
https://nnfs.io



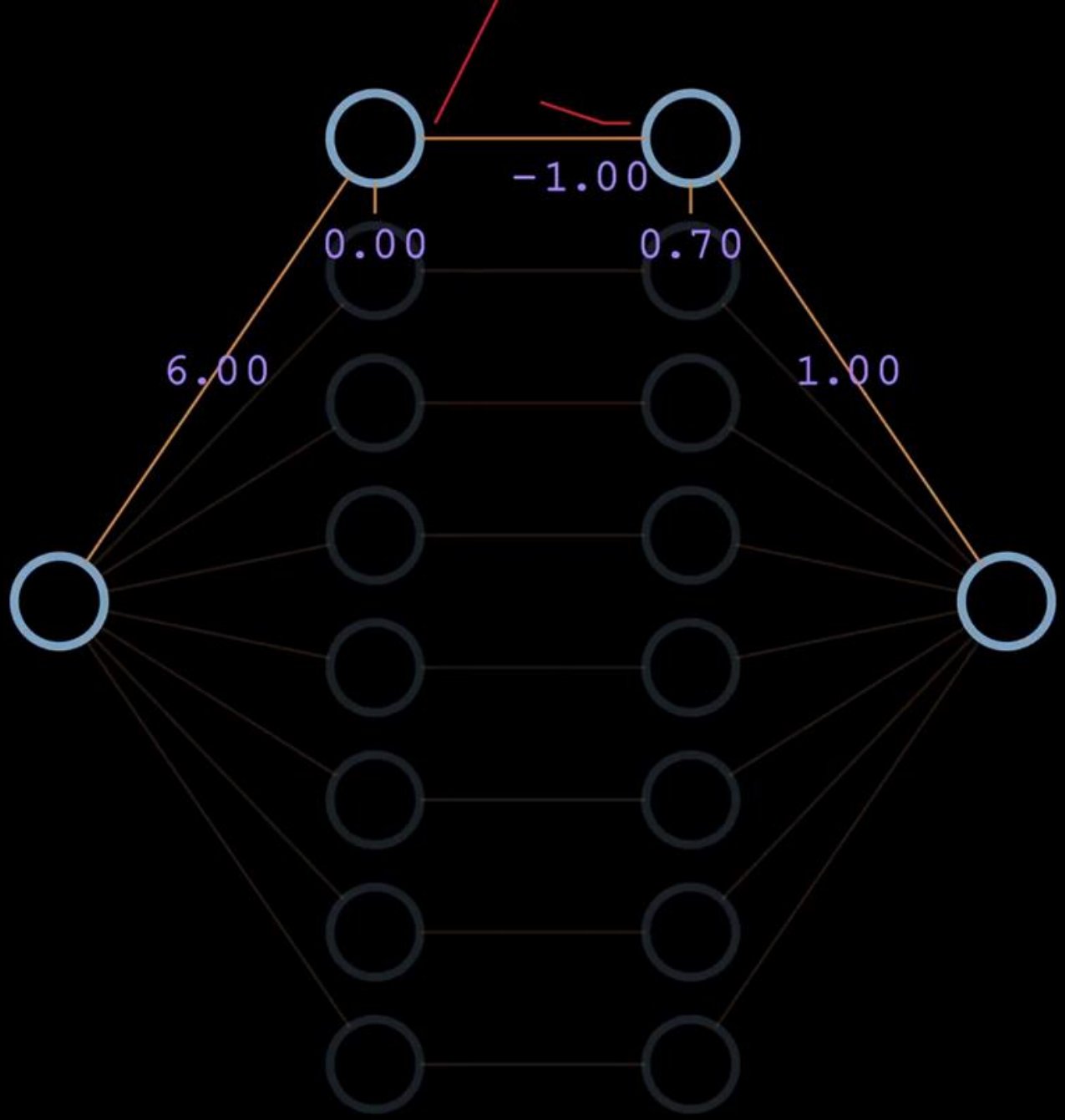


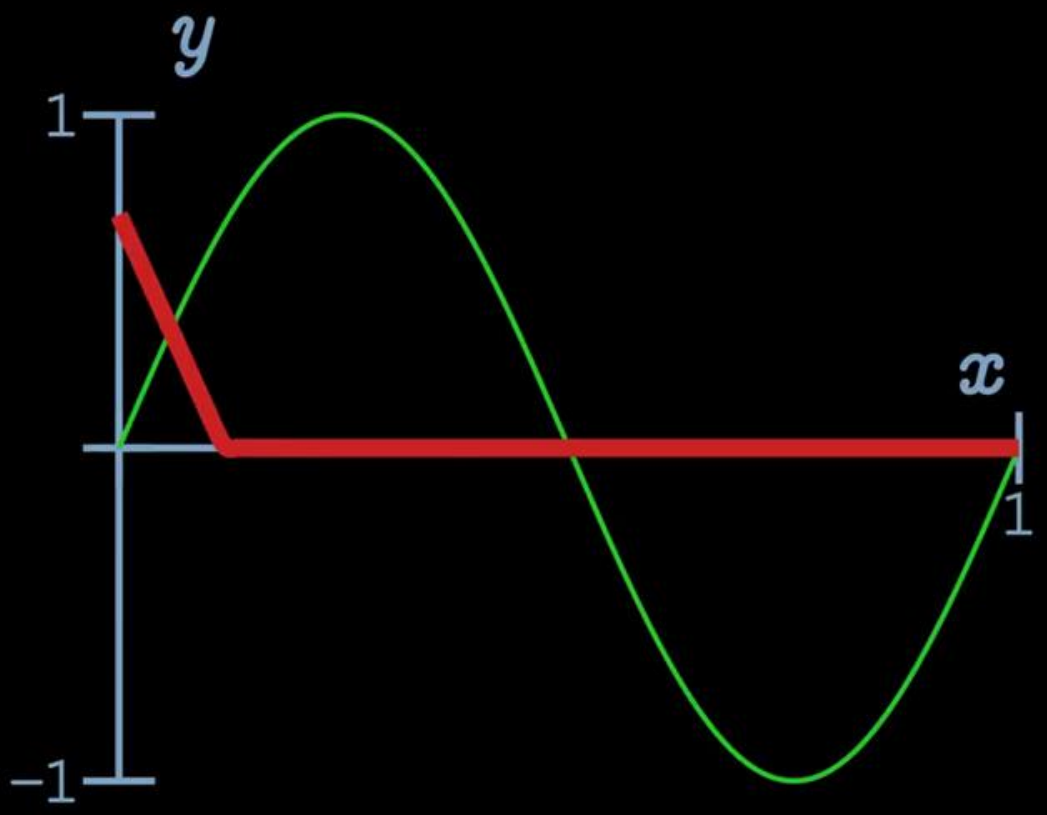
https://nnfs.io



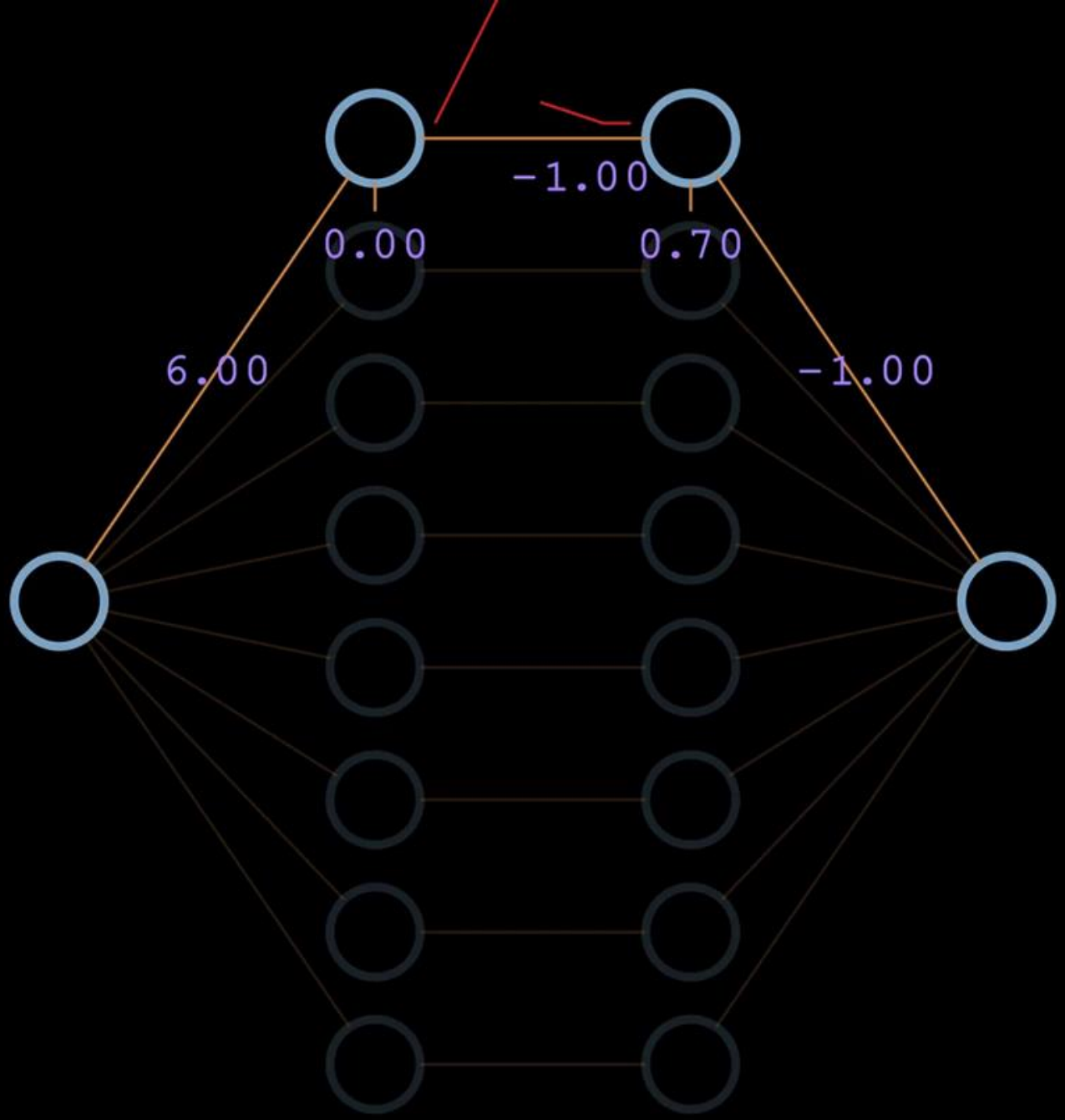


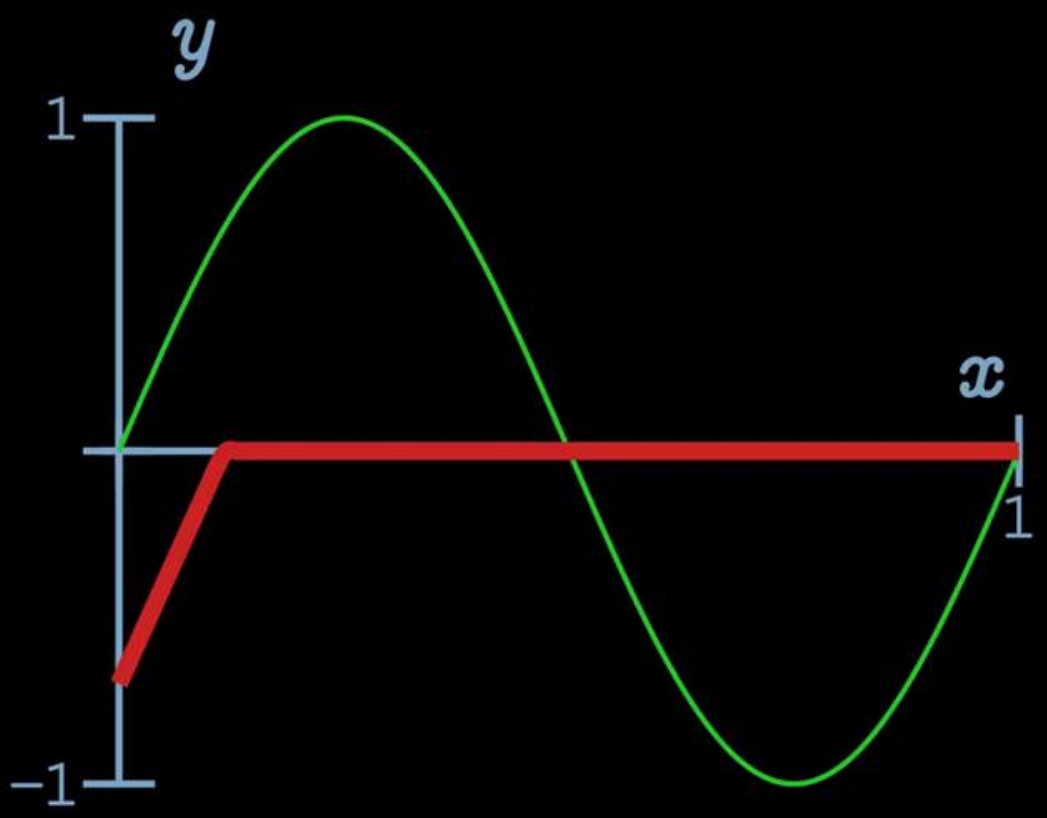
https://nnfs.io



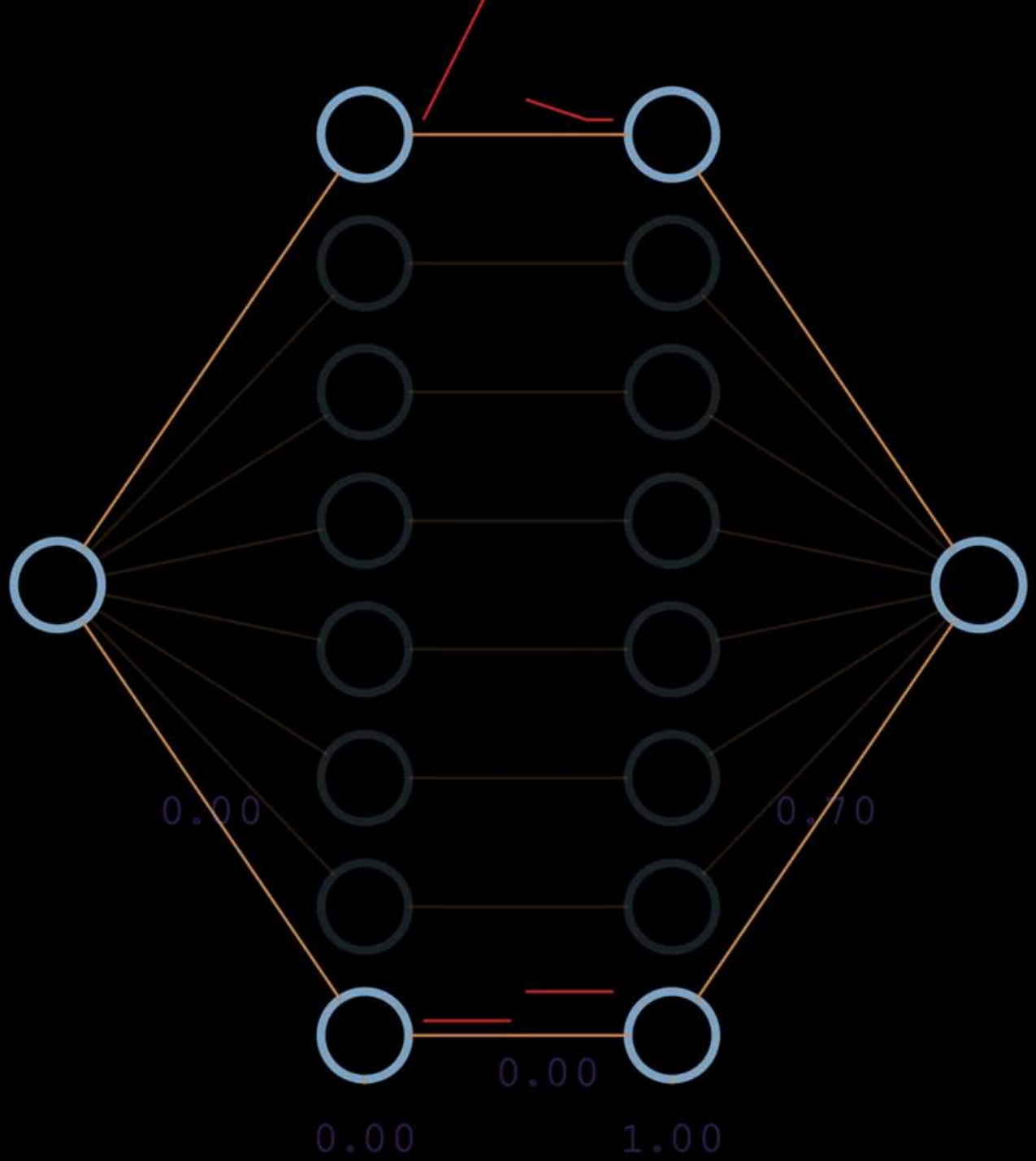


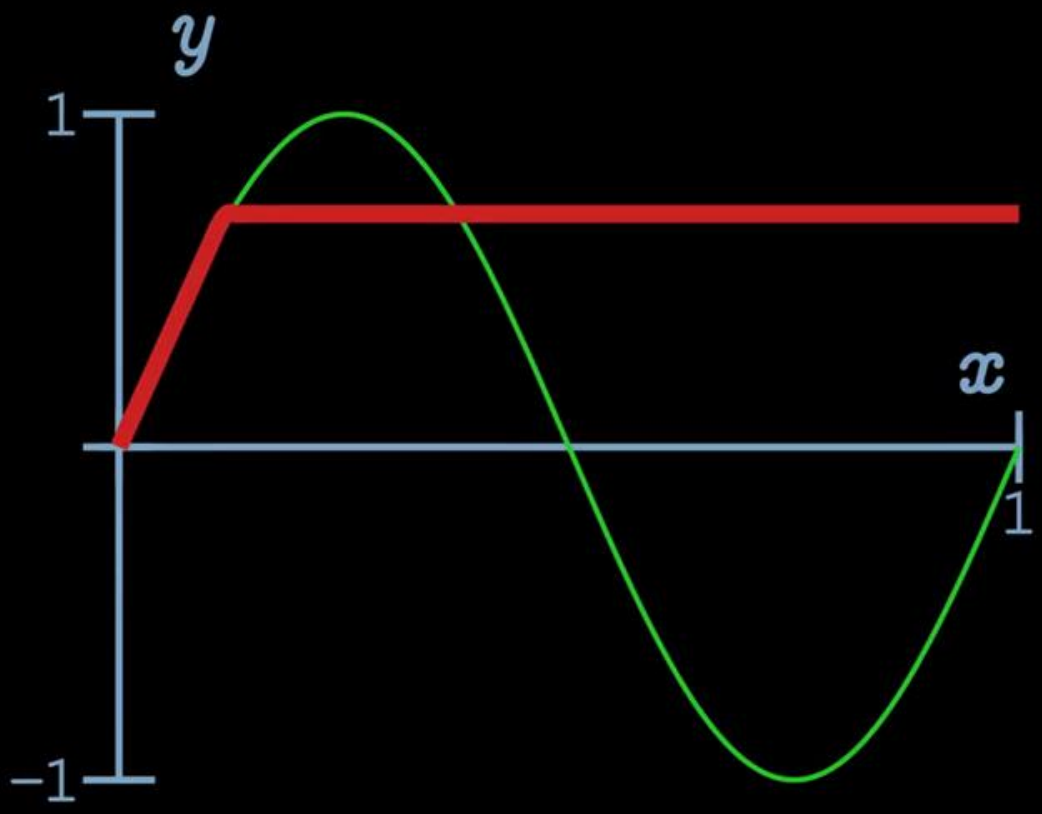
https://nnfs.io



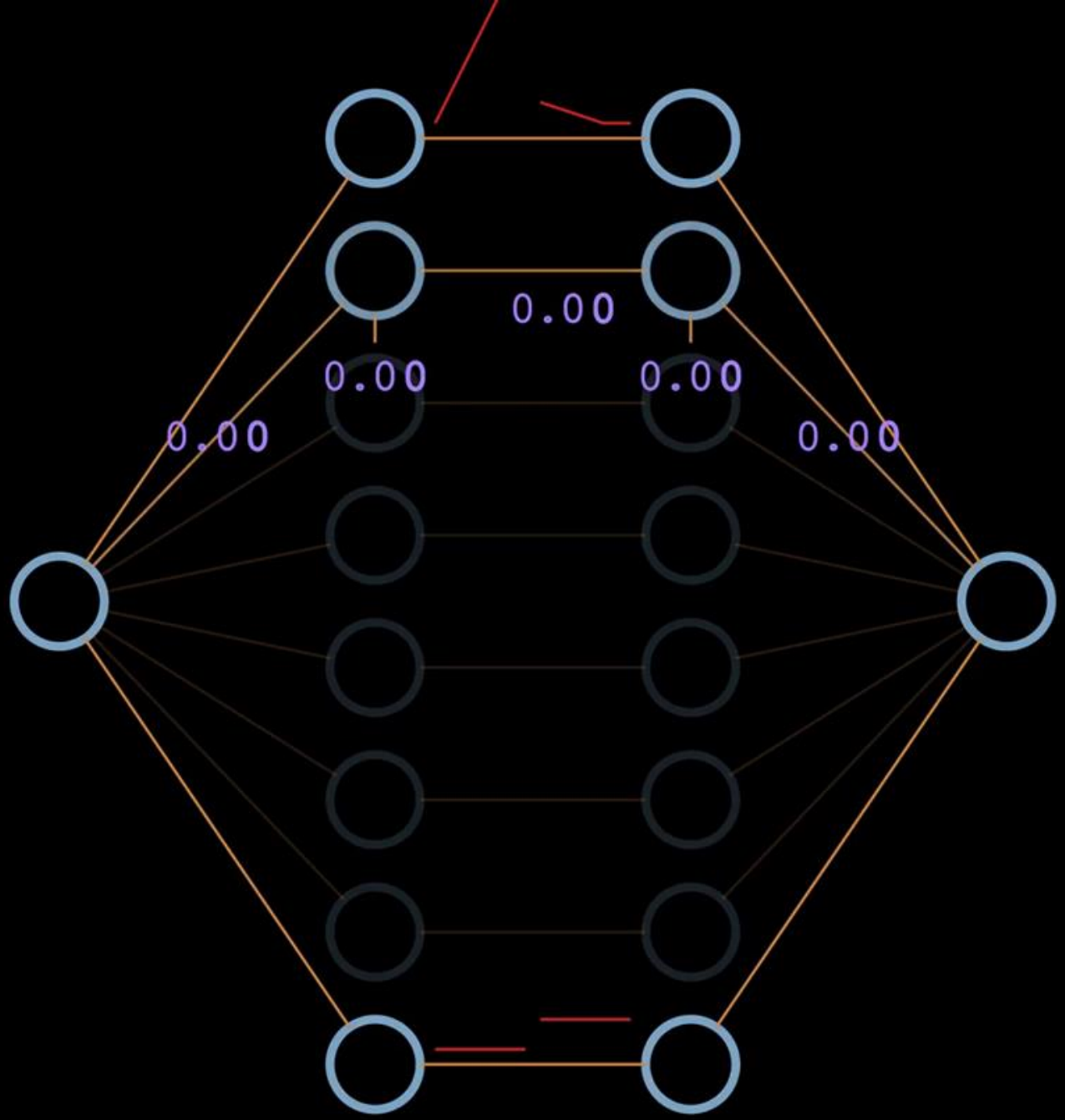


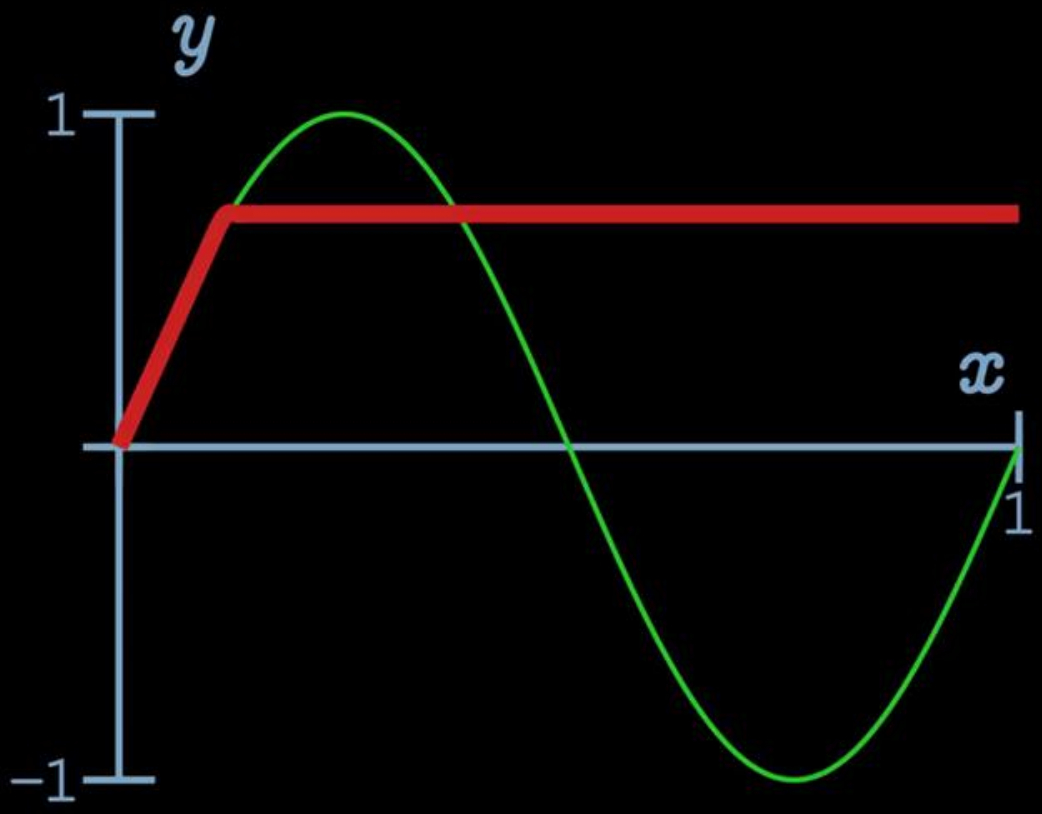
https://nnfs.io



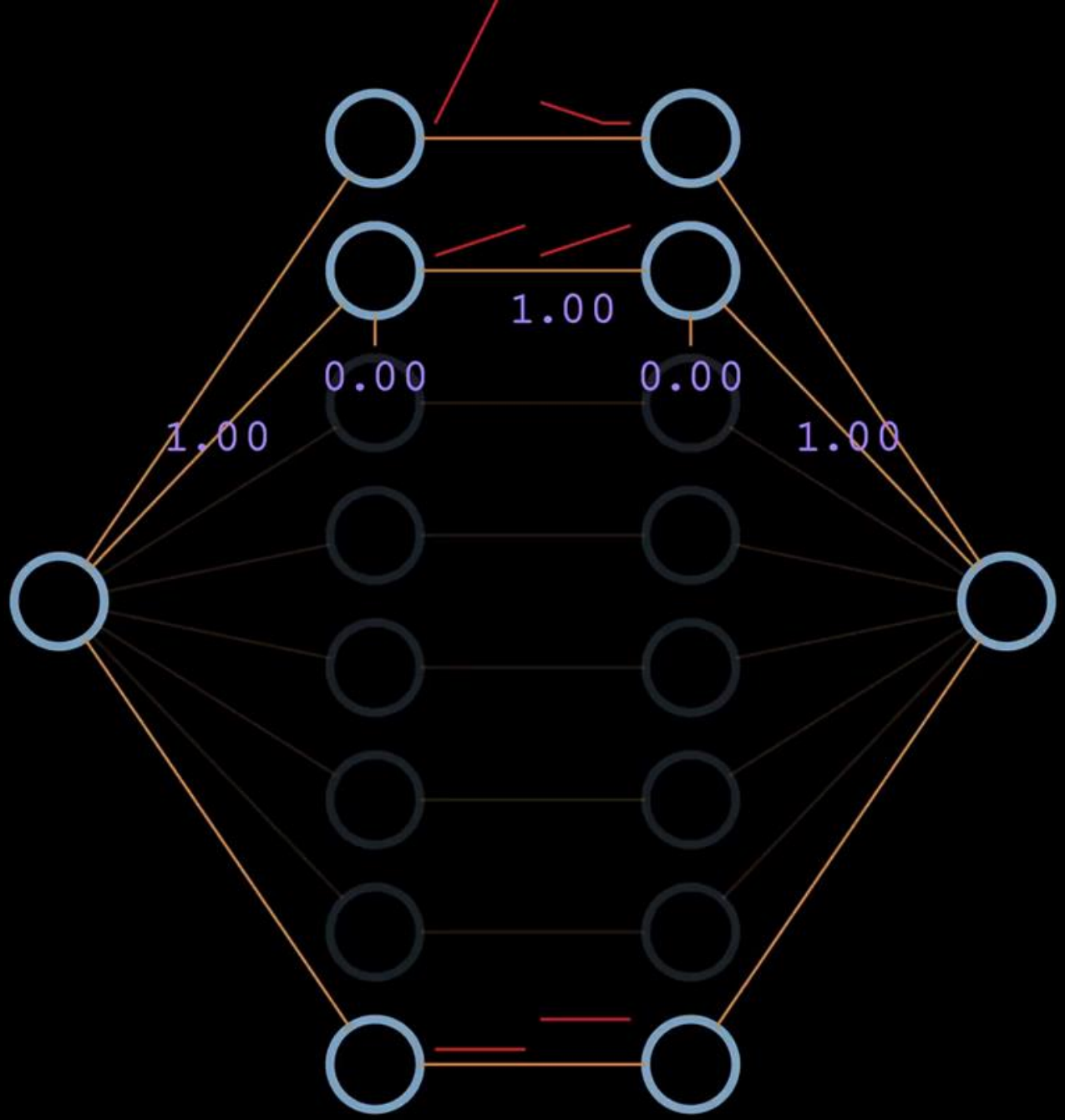


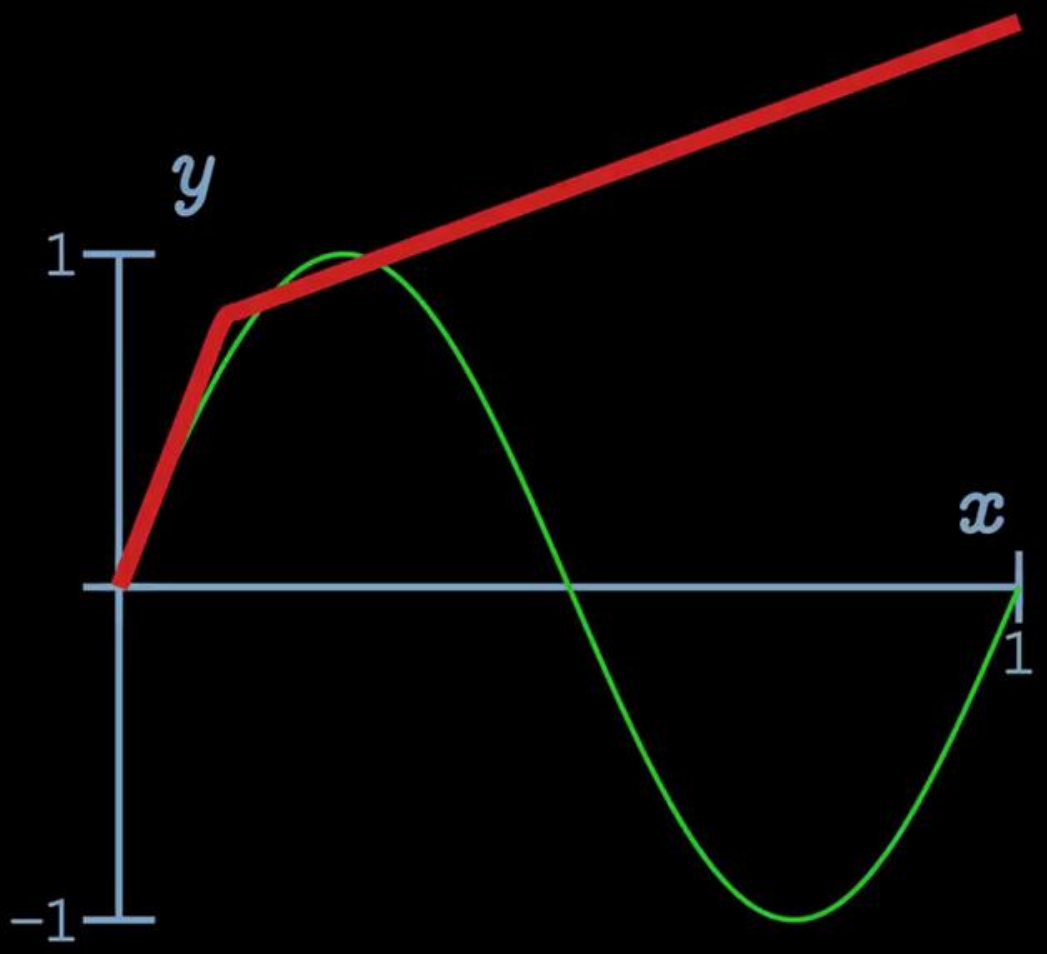
https://nnfs.io



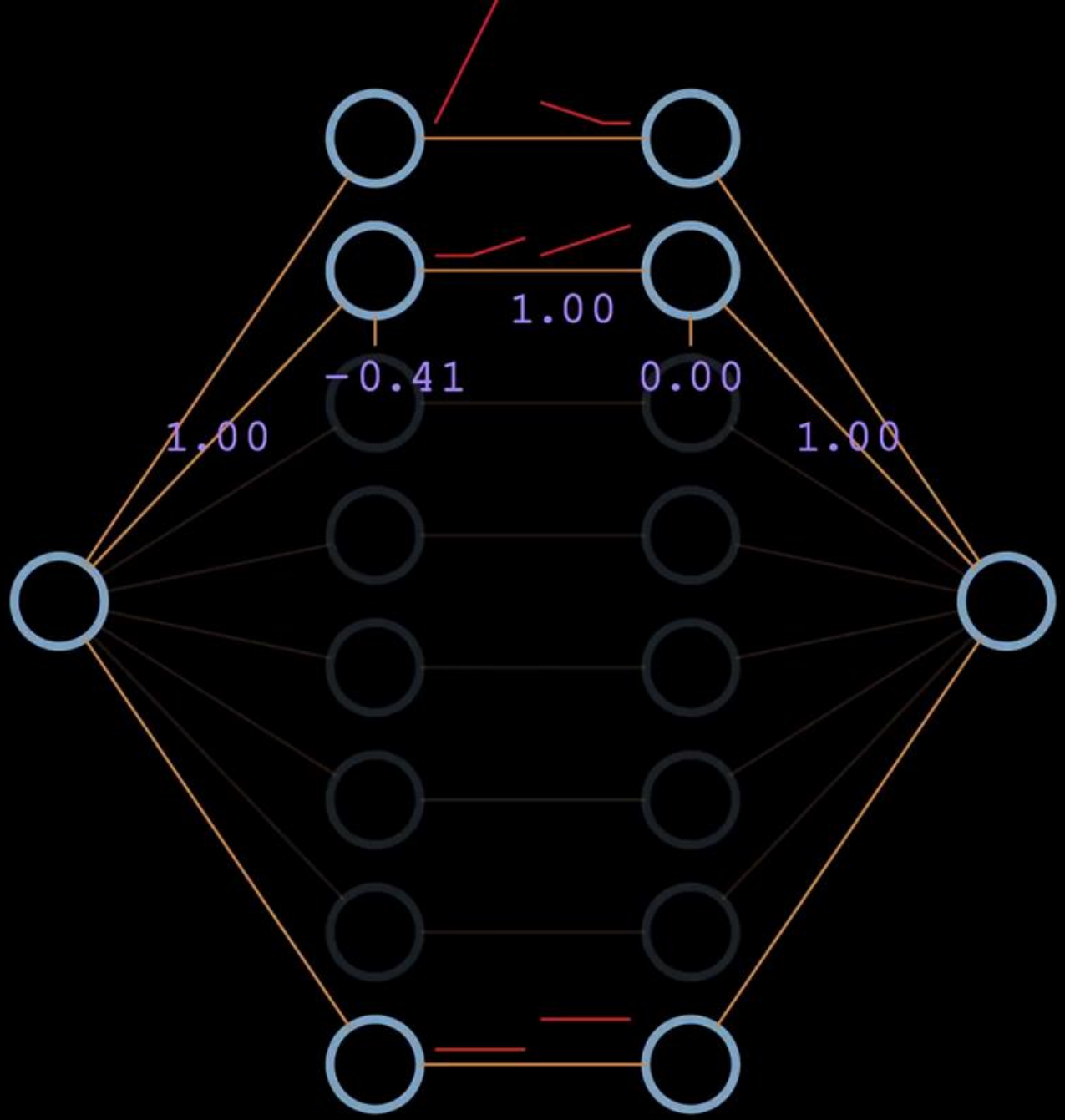


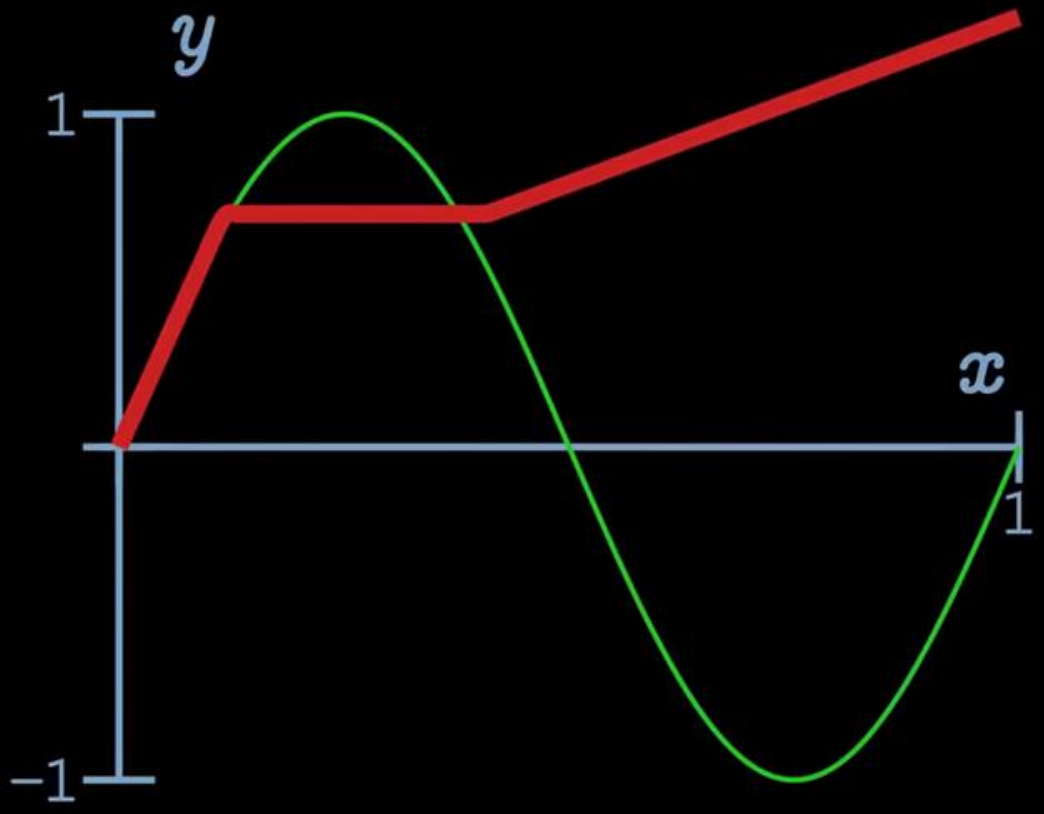
https://nnfs.io



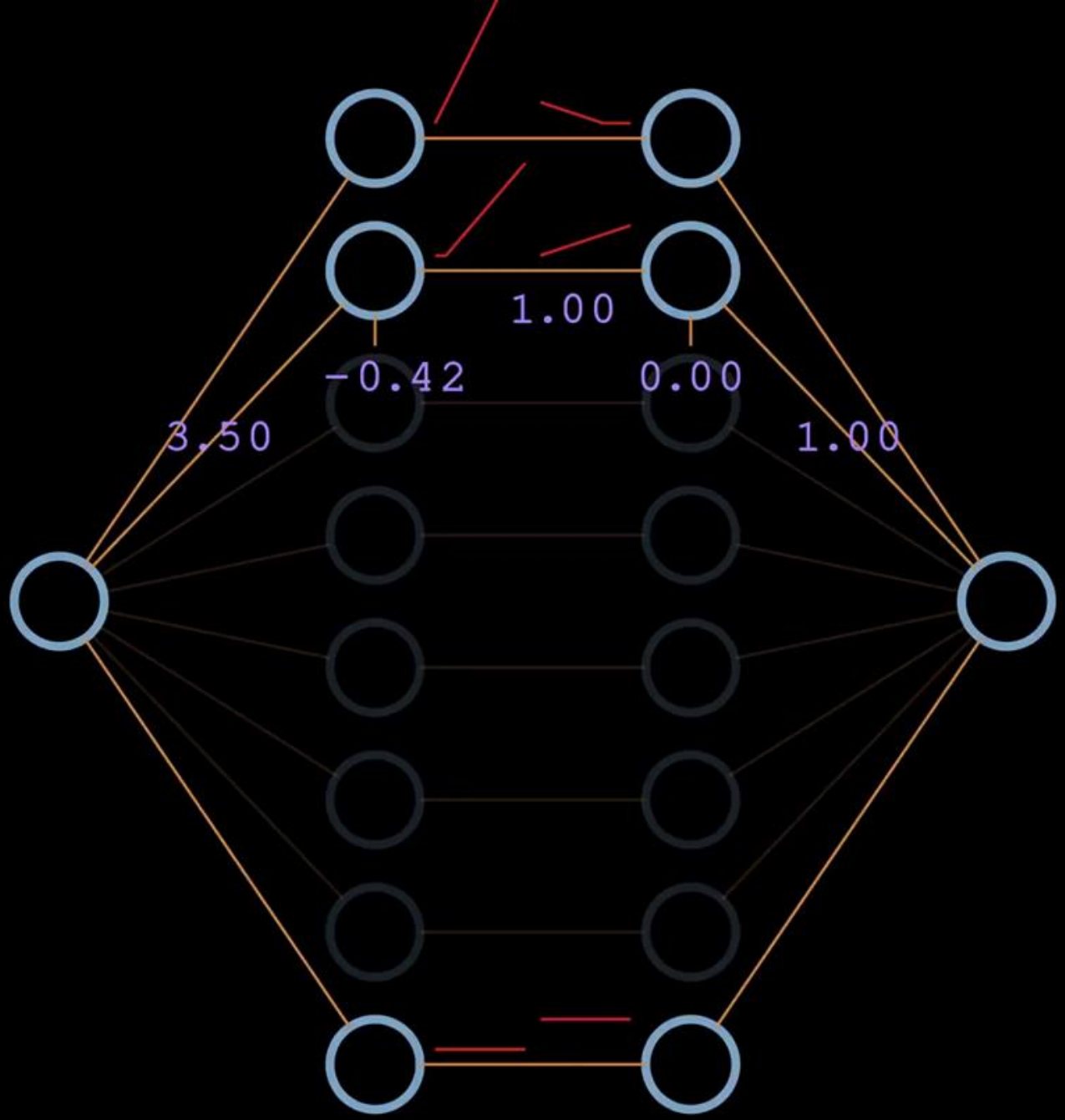


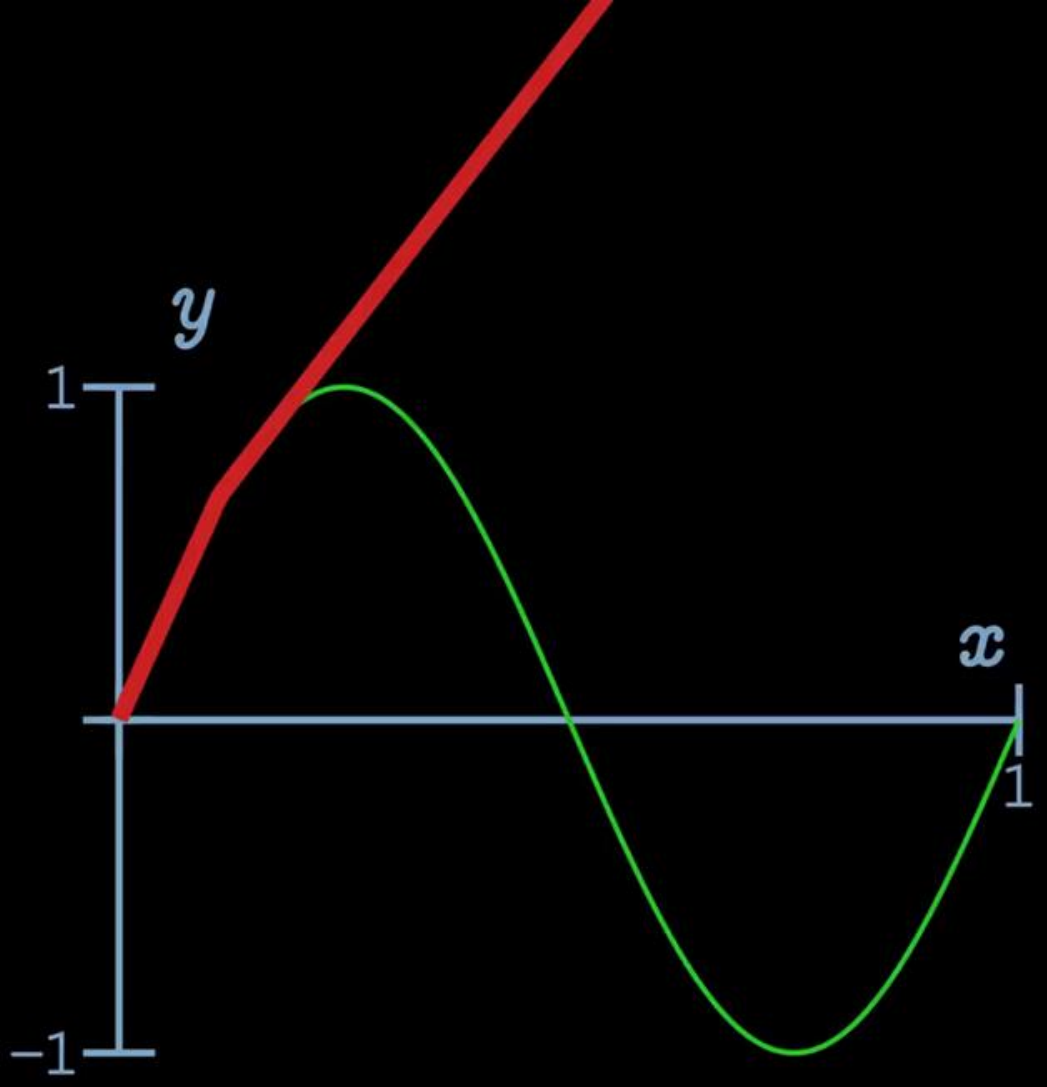
https://nnfs.io



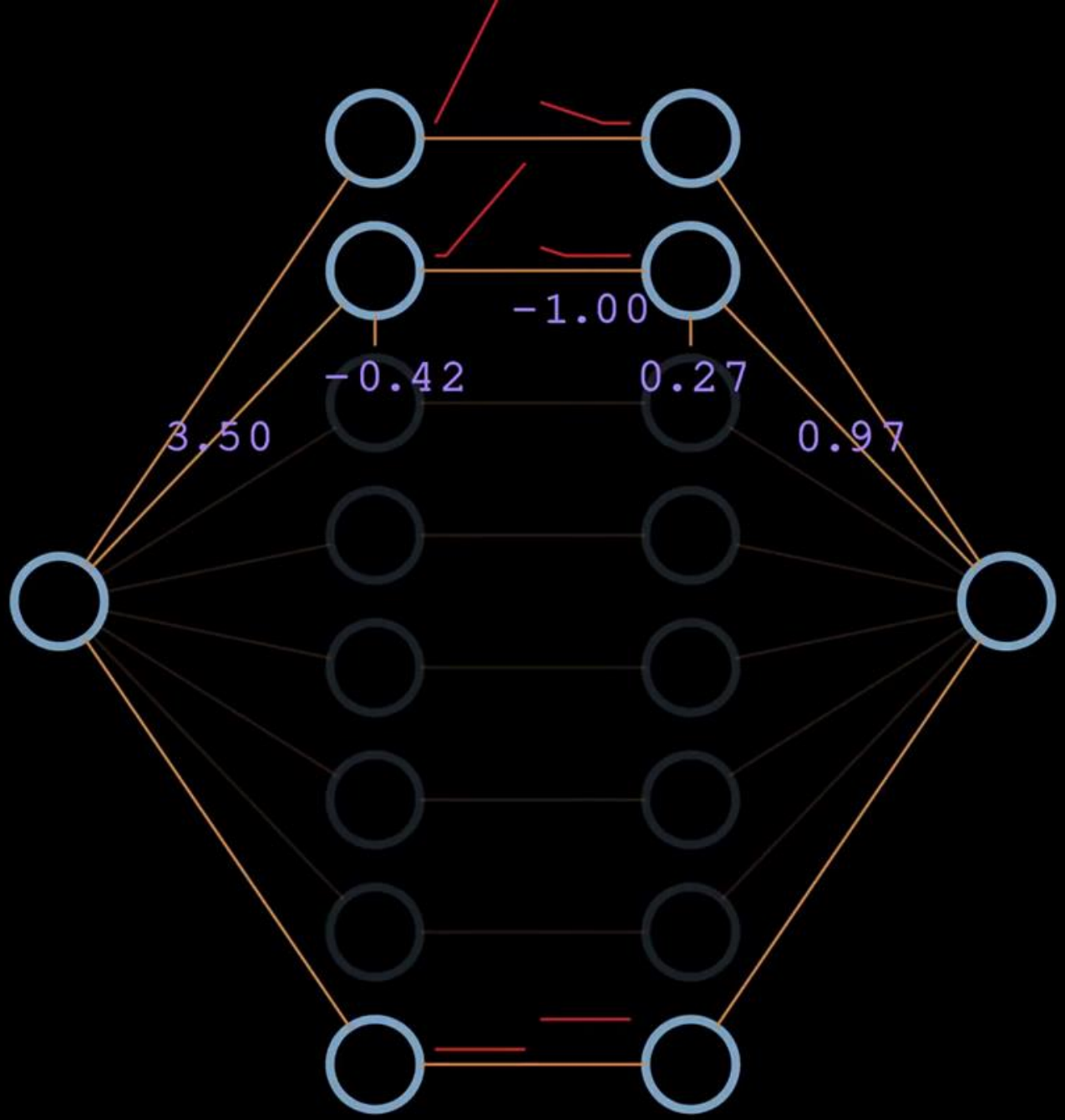


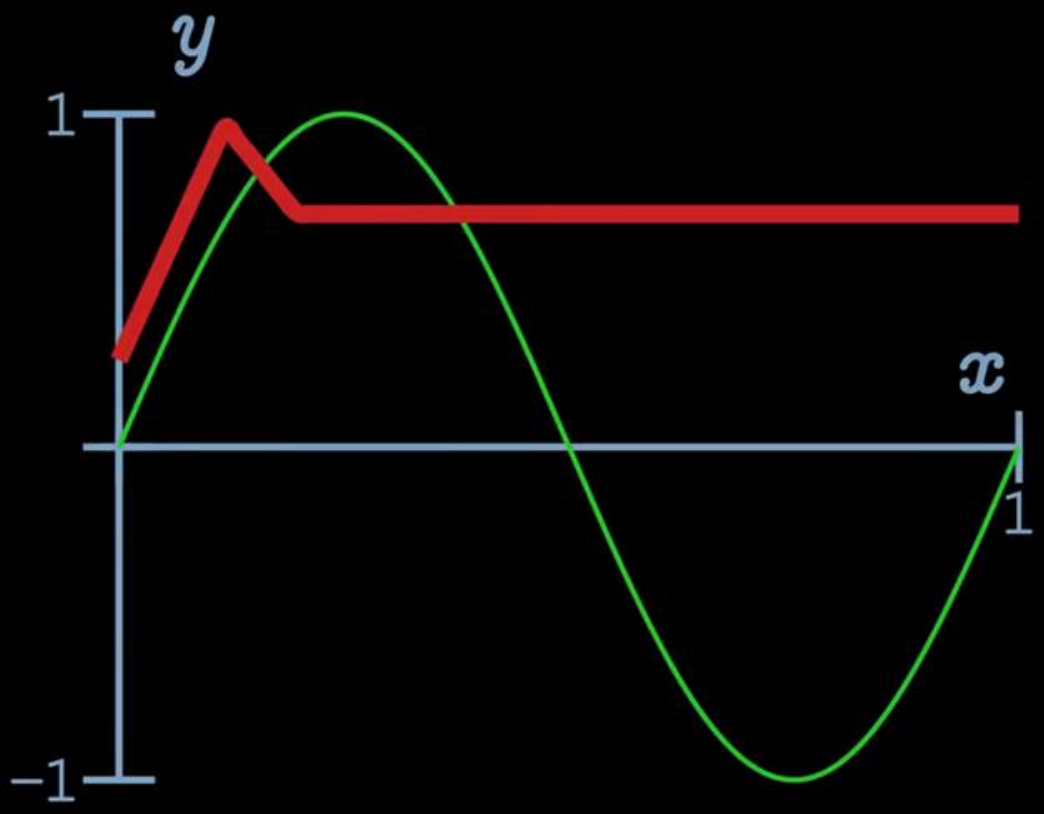
https://nnfs.io



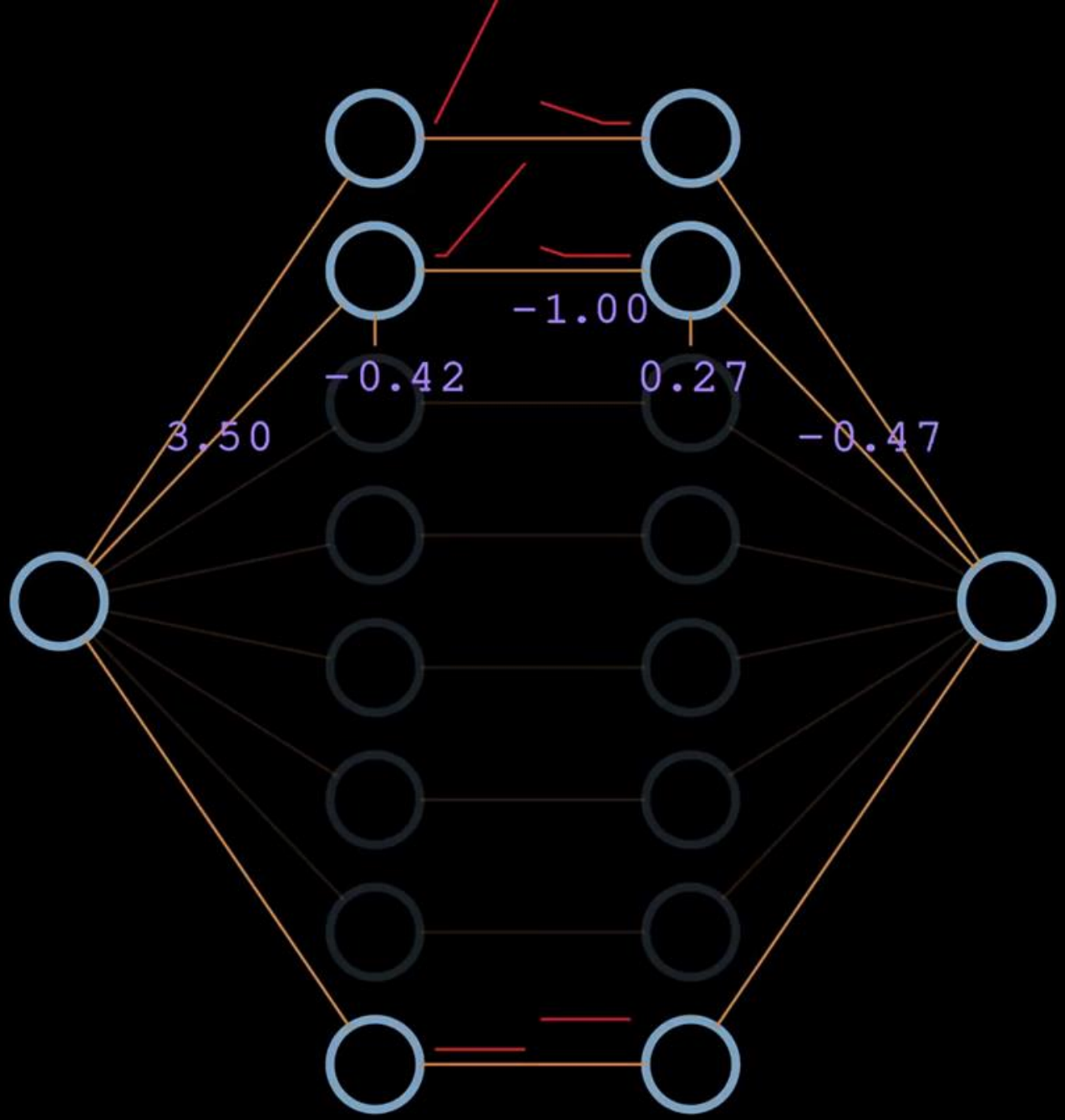


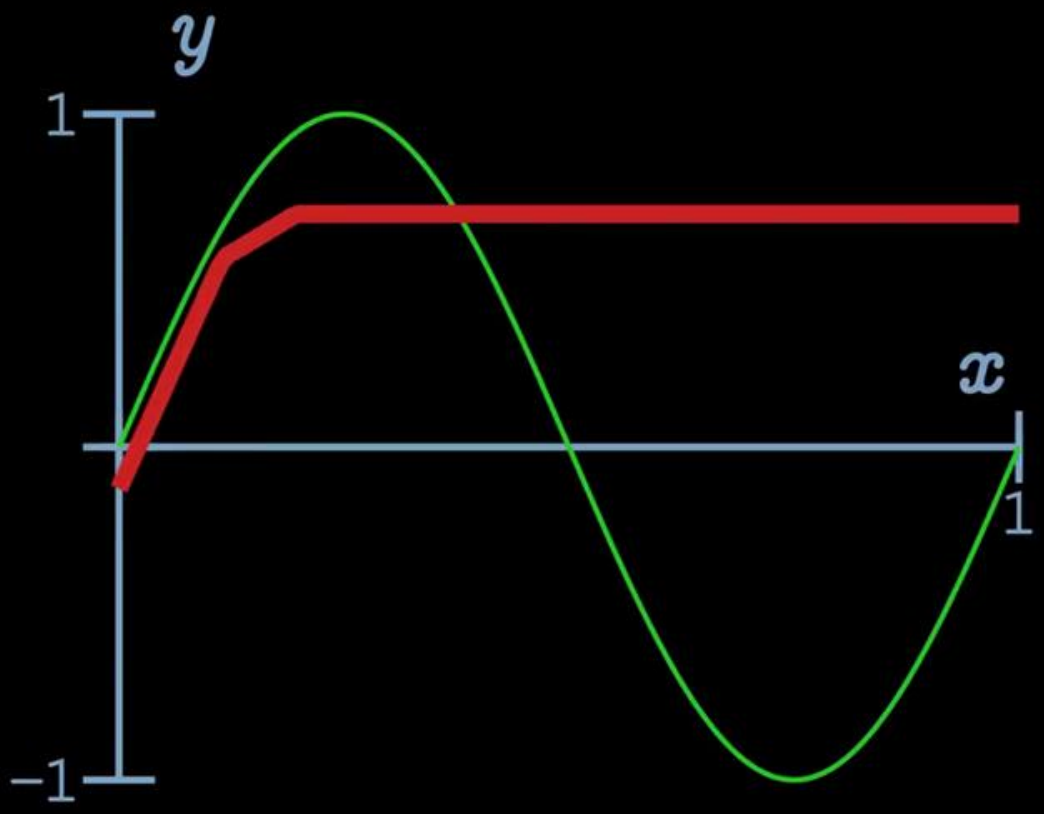
https://nnfs.io



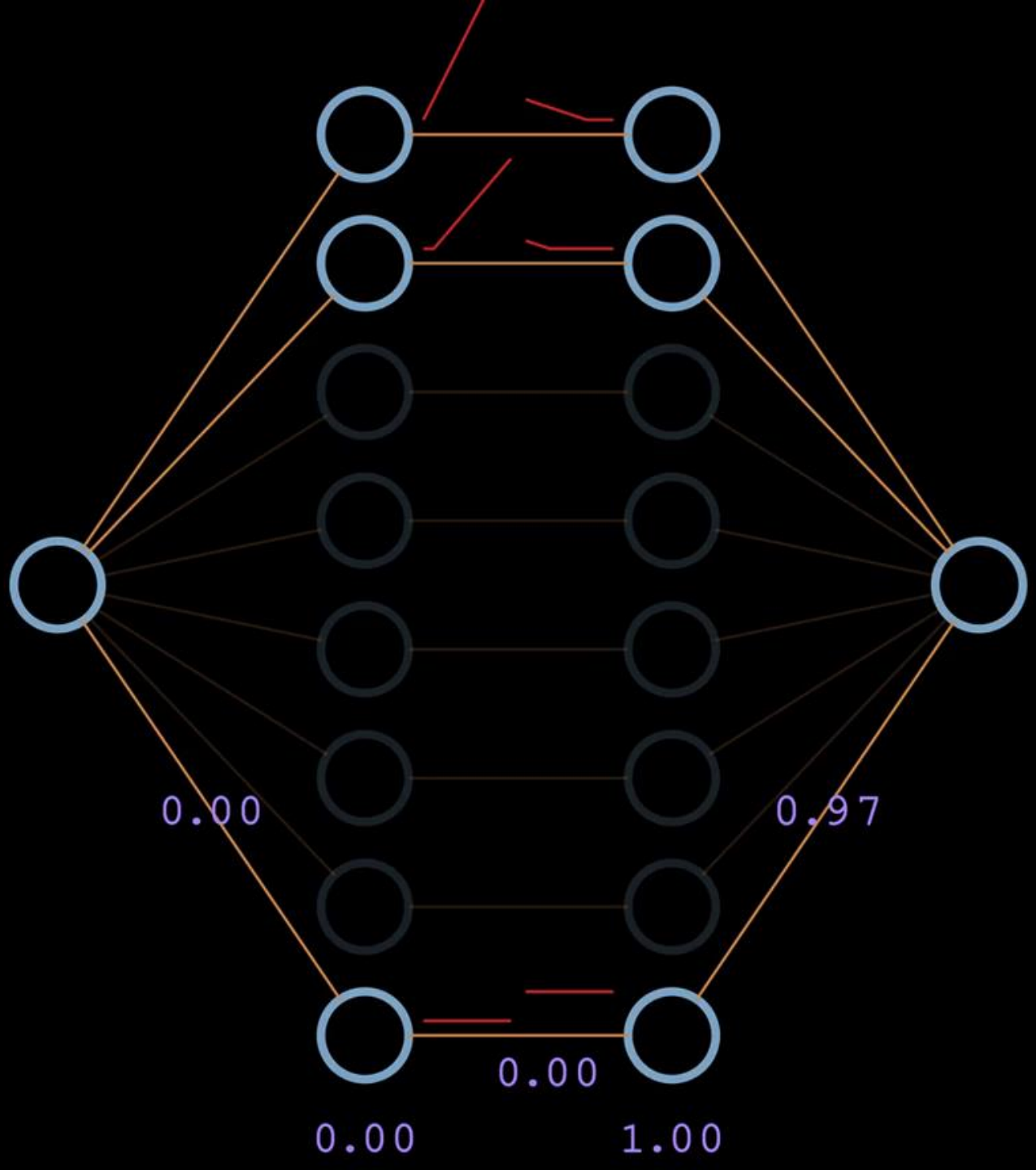


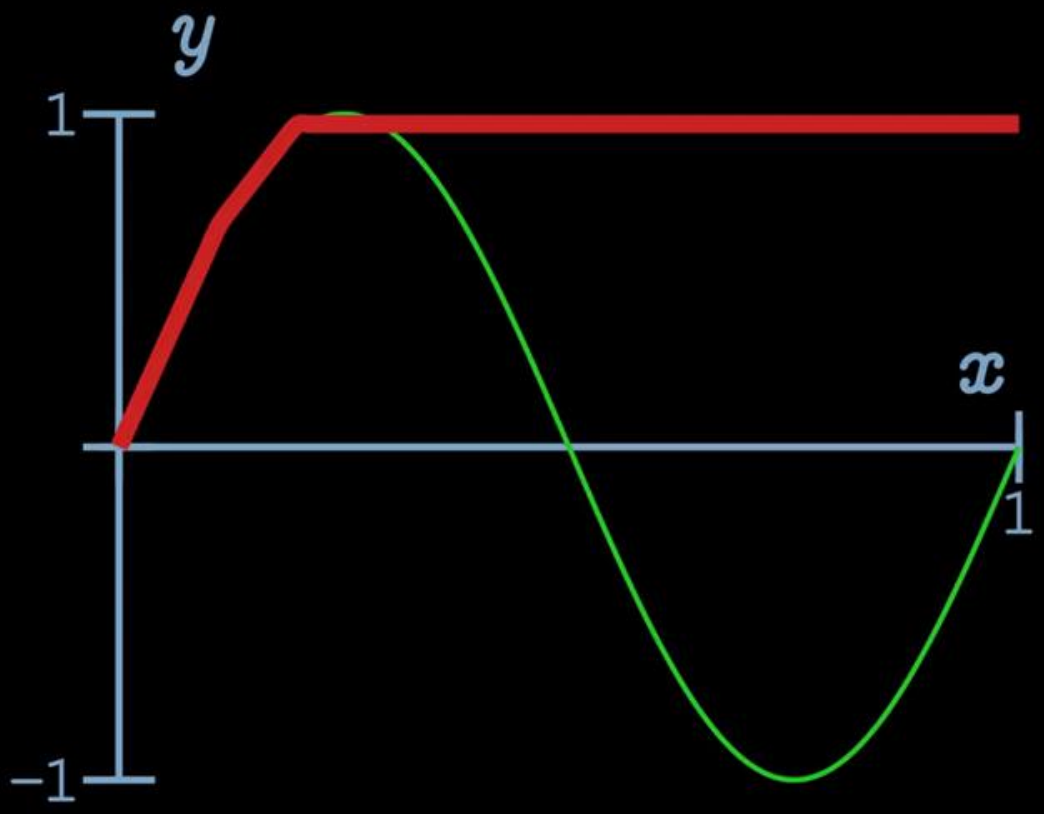
https://nnfs.io



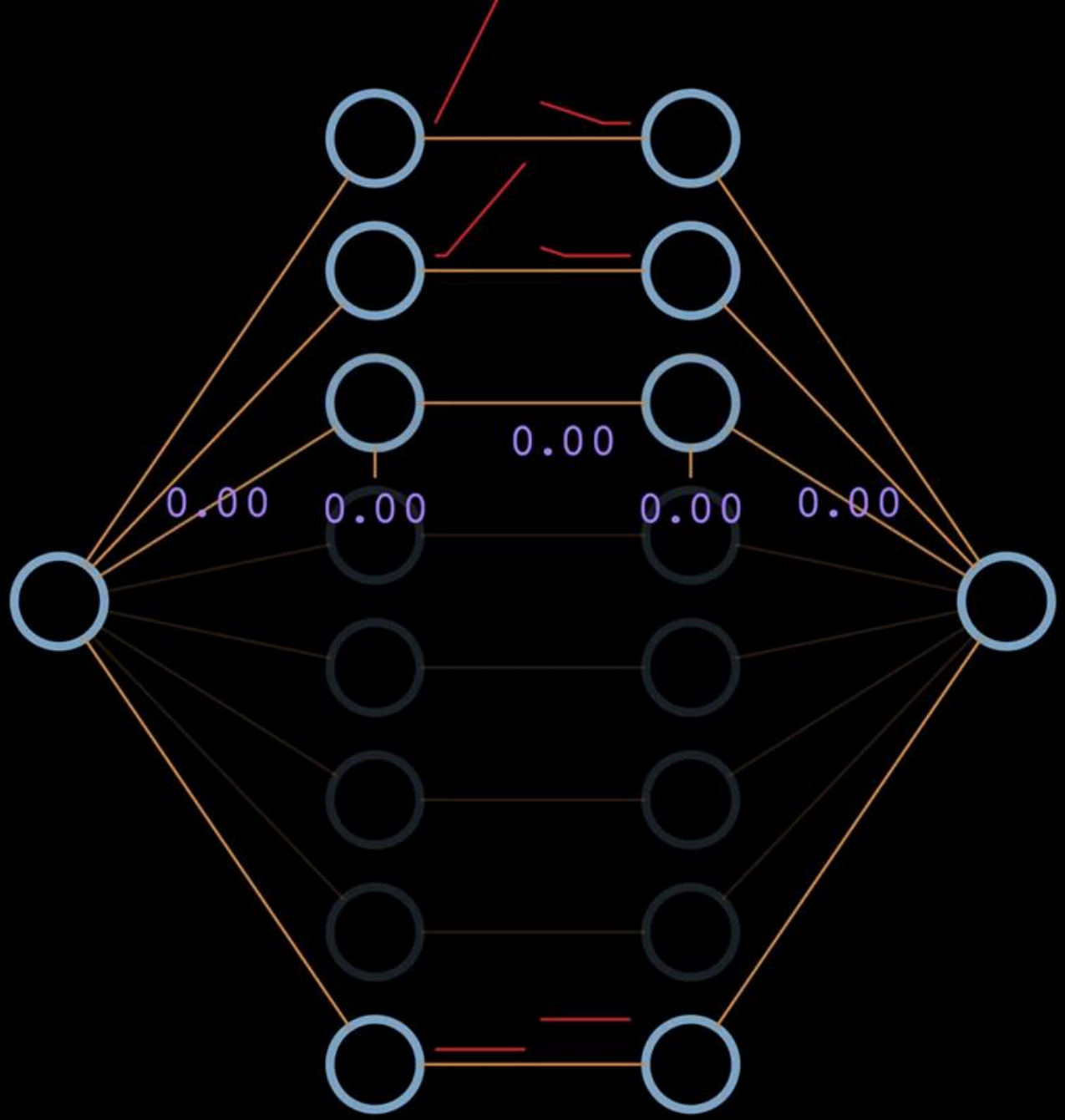


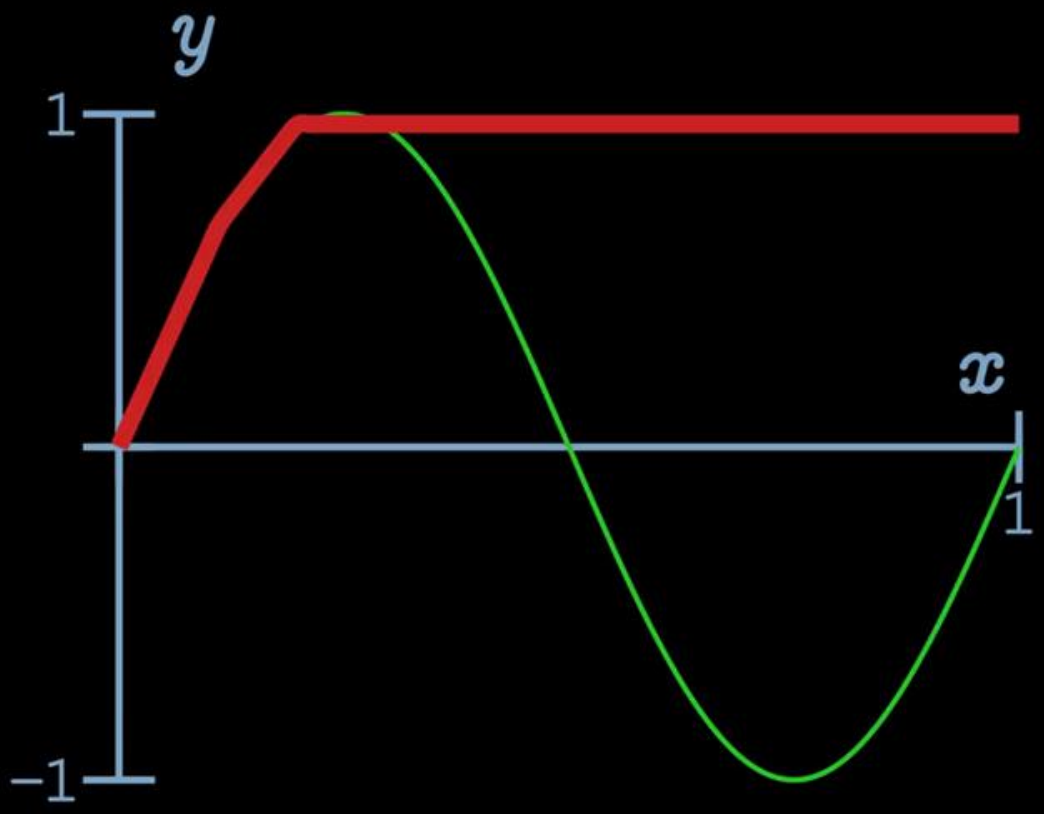
https://nnfs.io



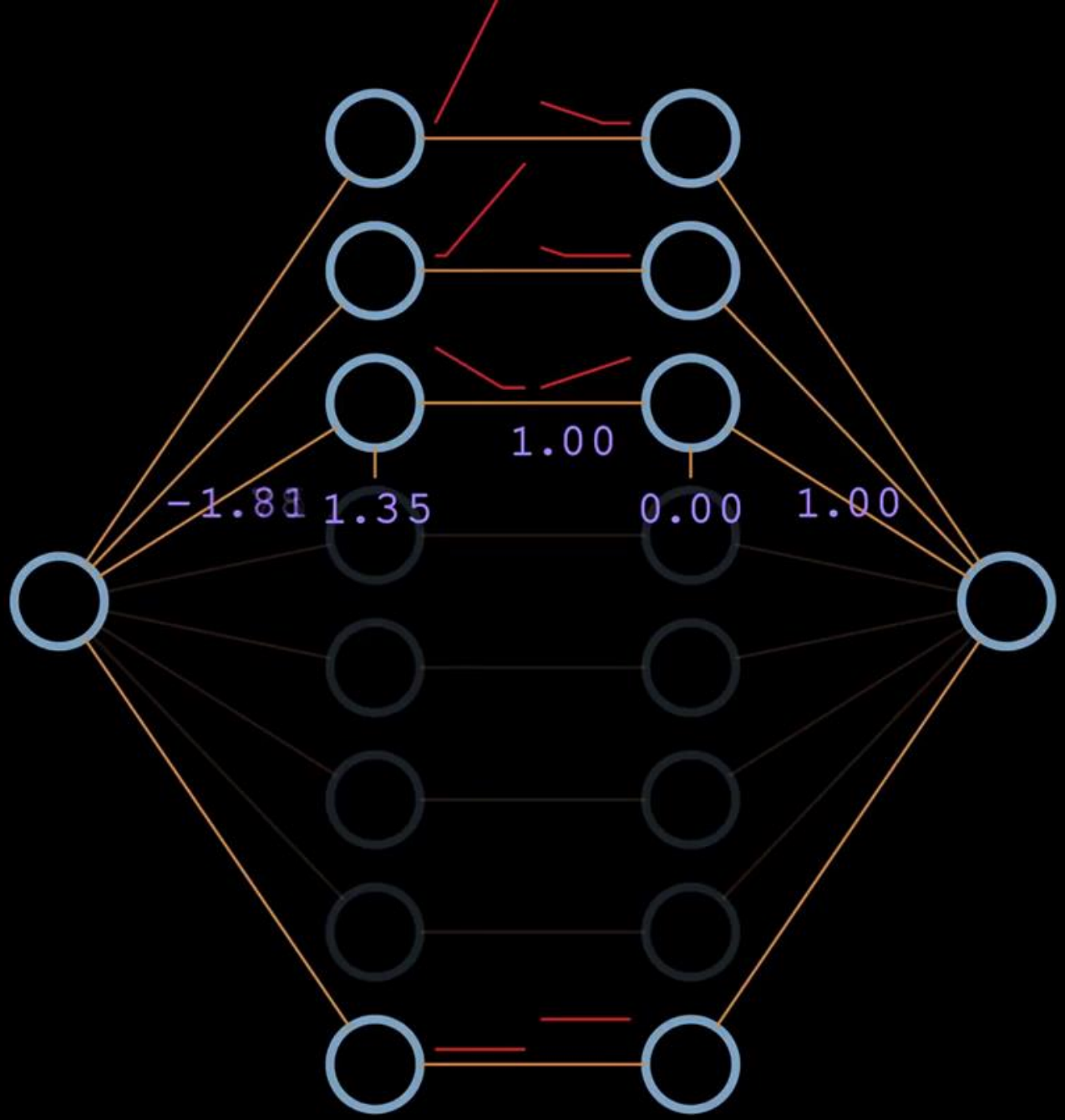


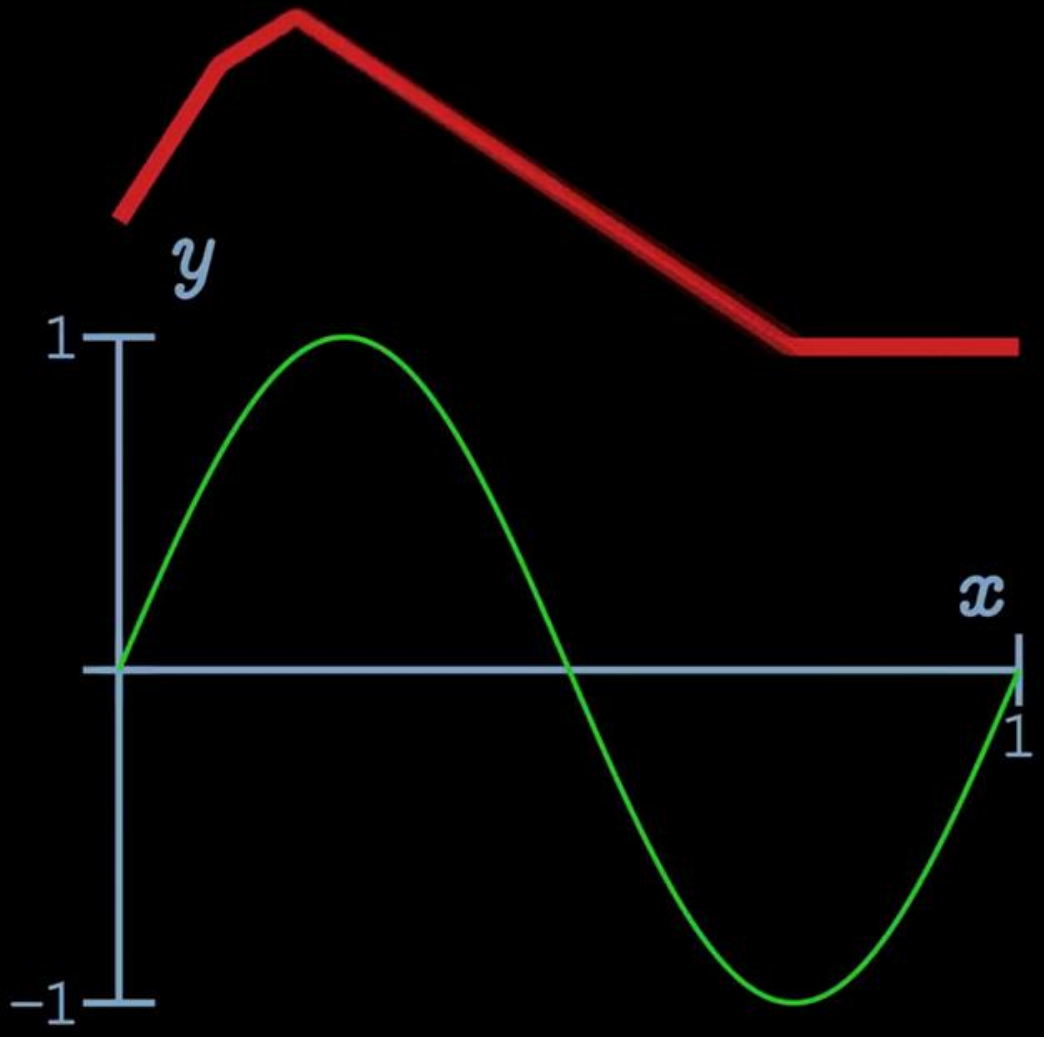
https://nnfs.io



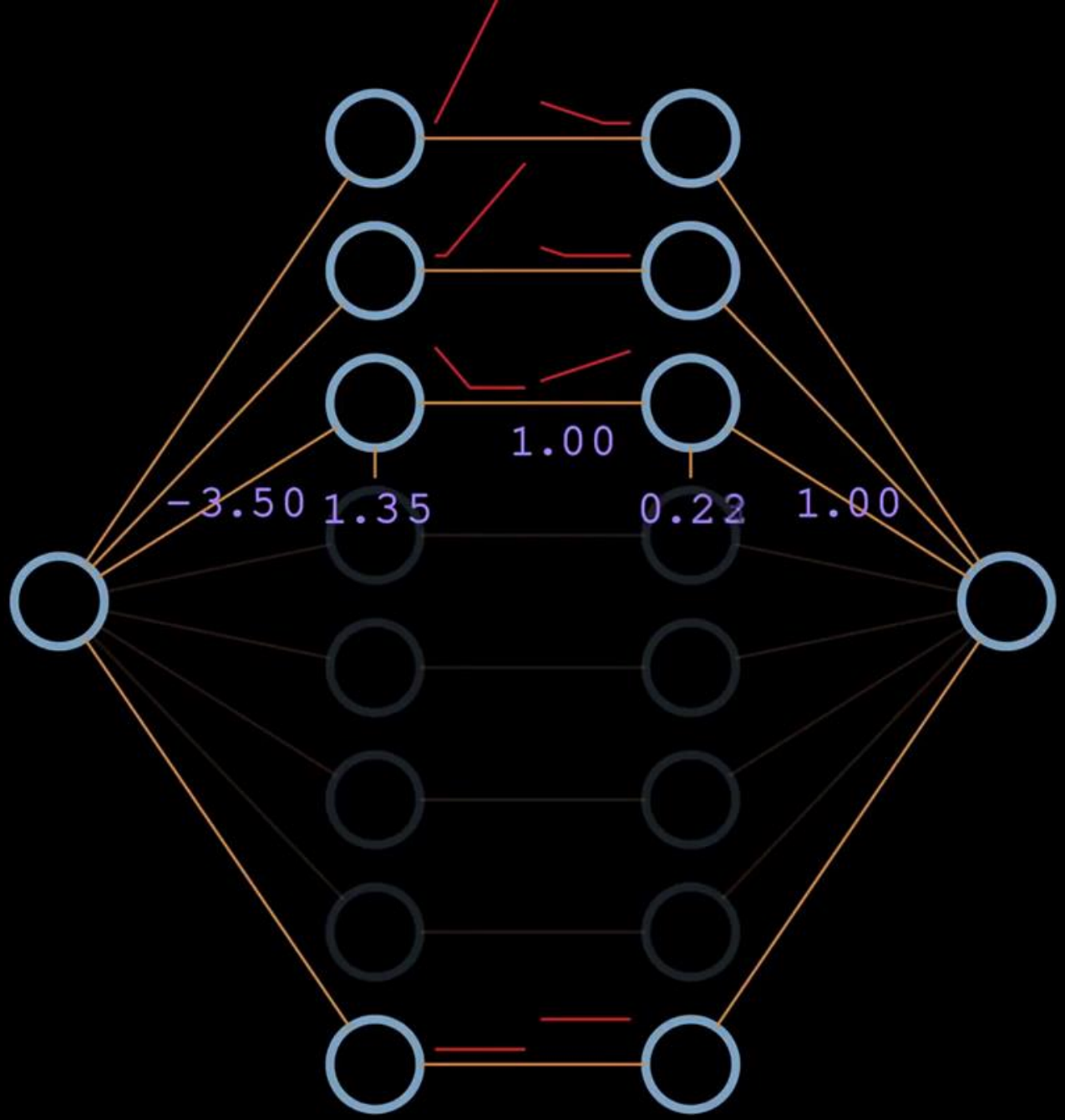


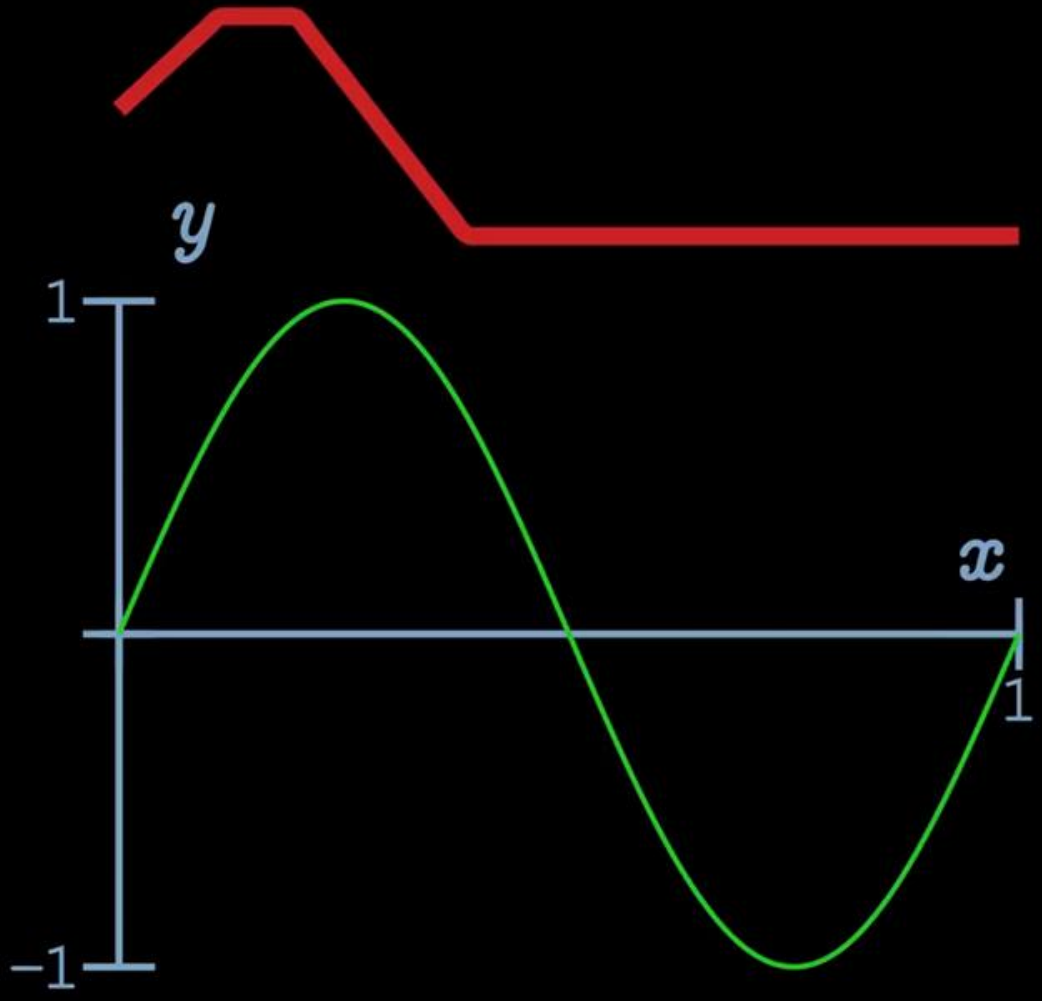
https://nnfs.io

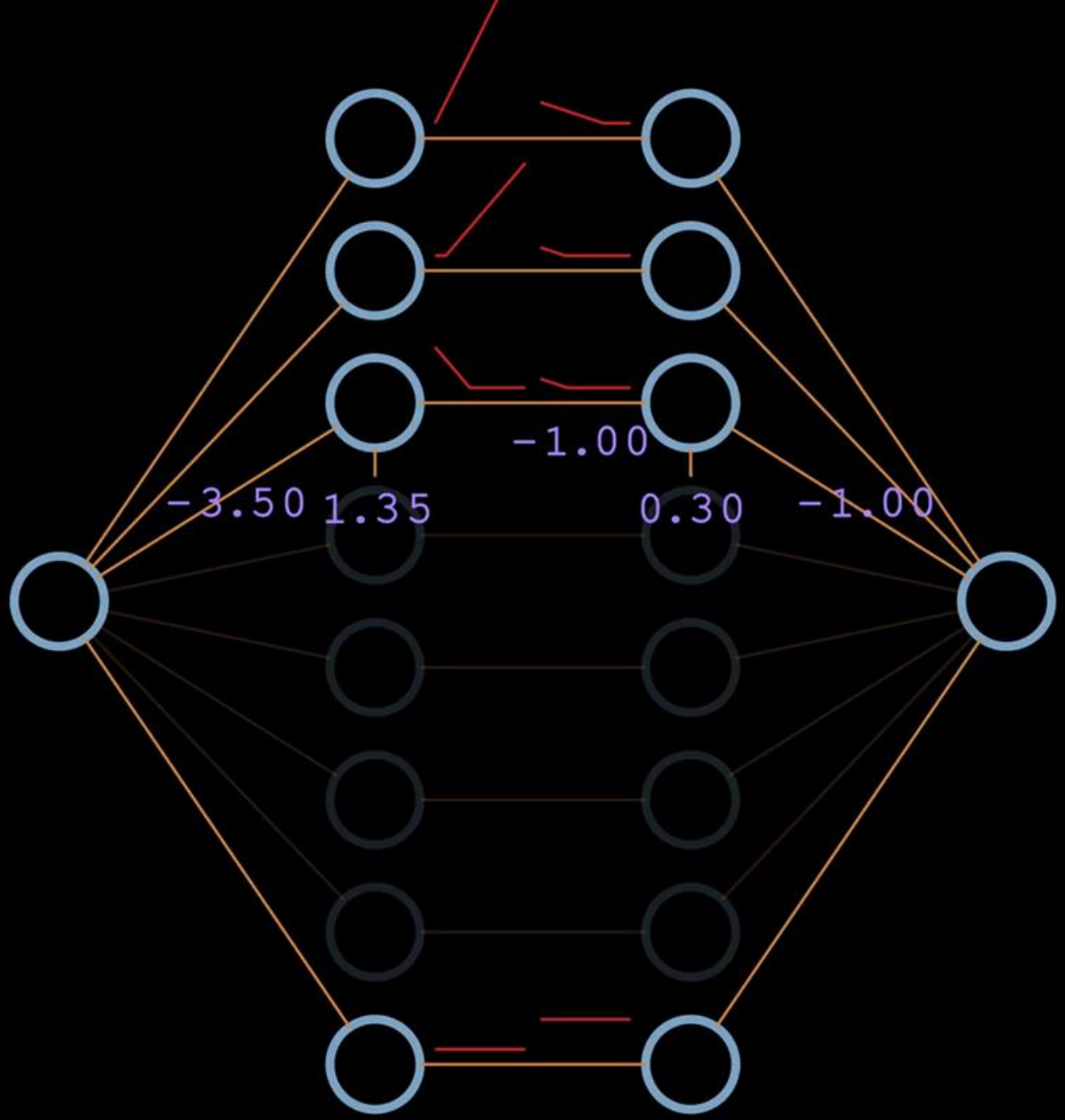


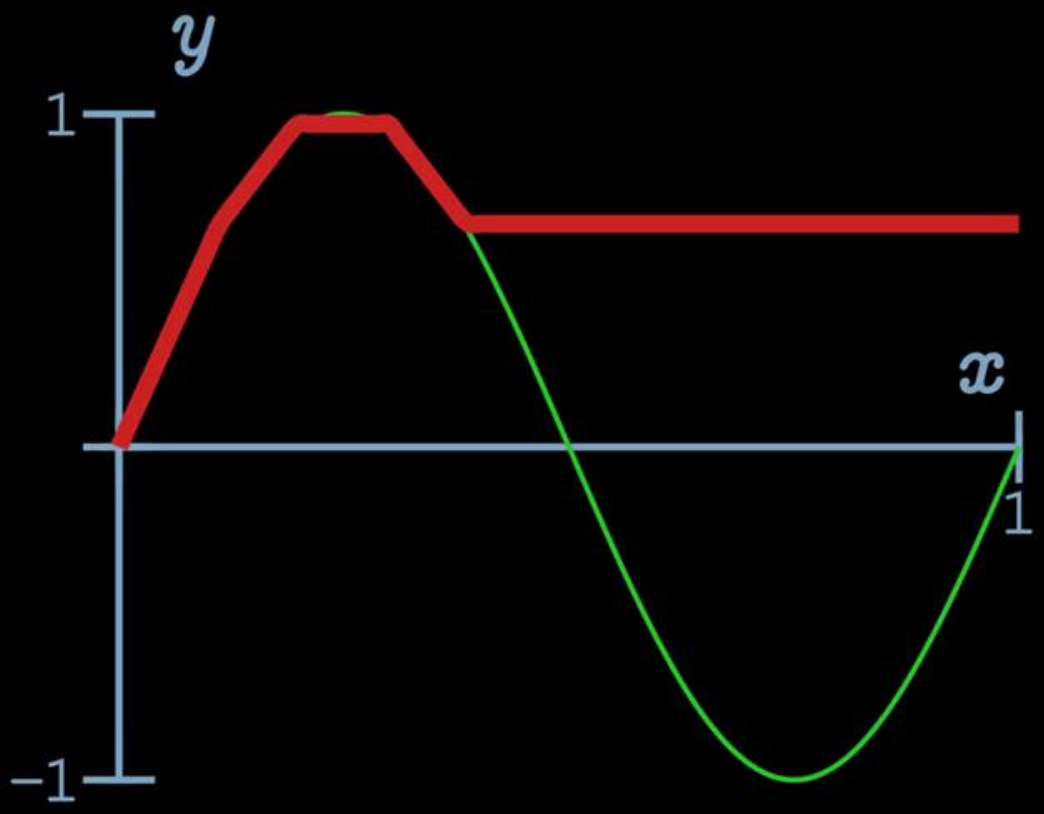


https://nnfs.io

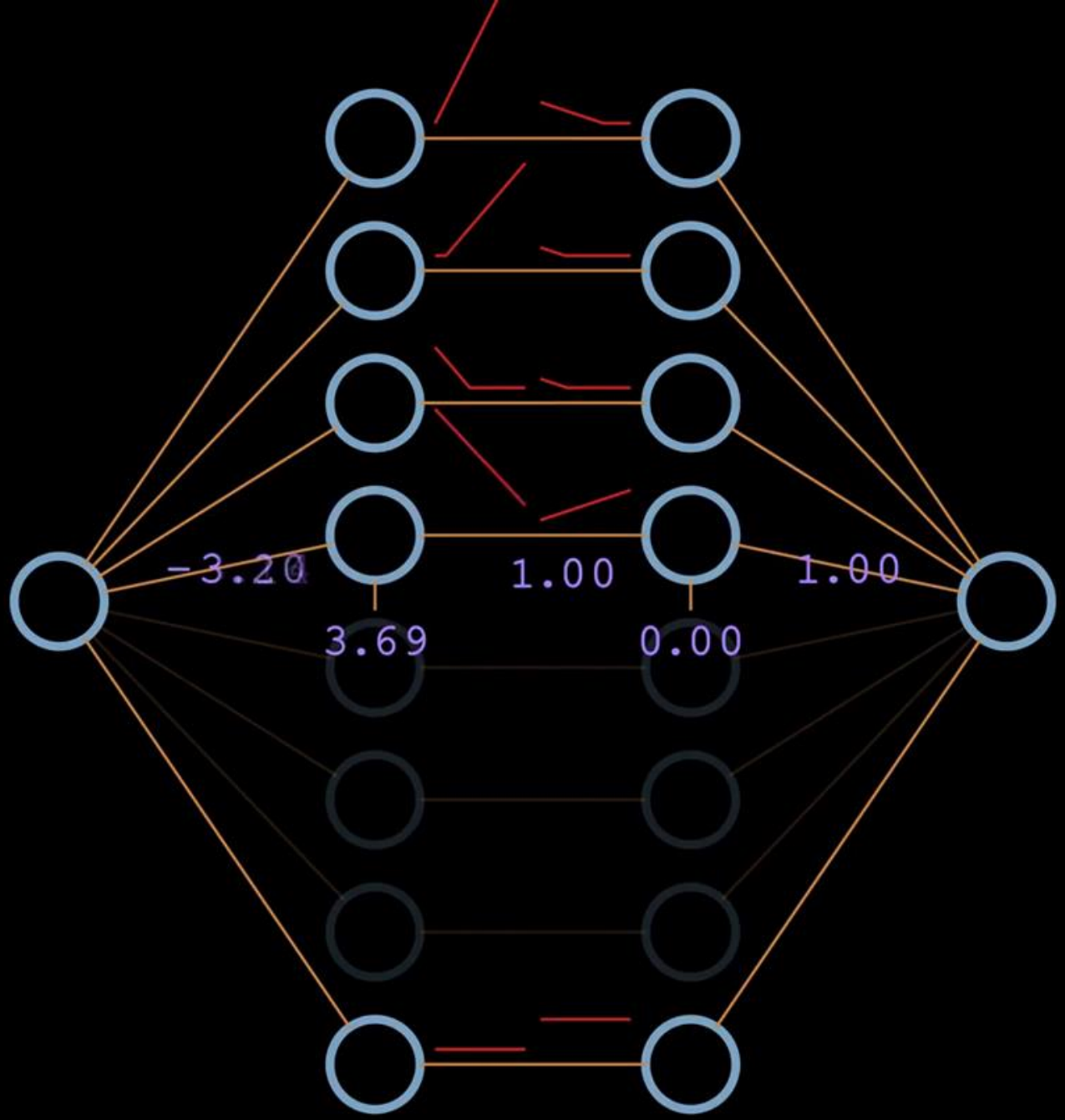


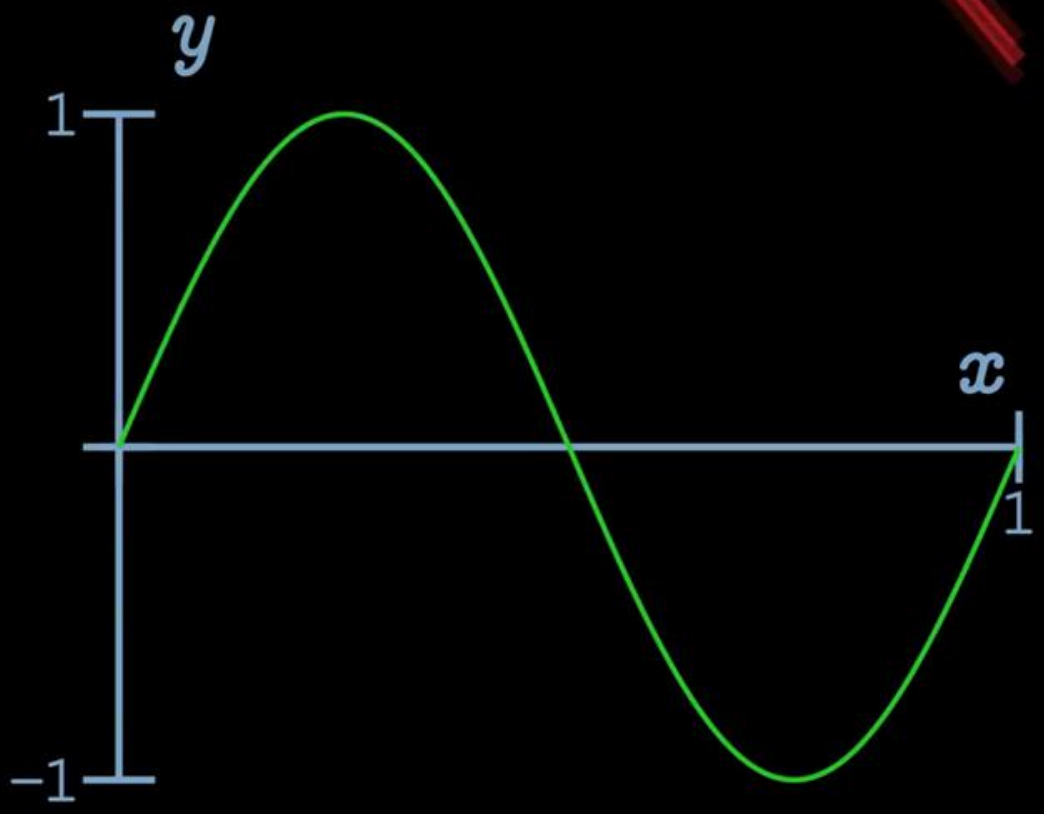




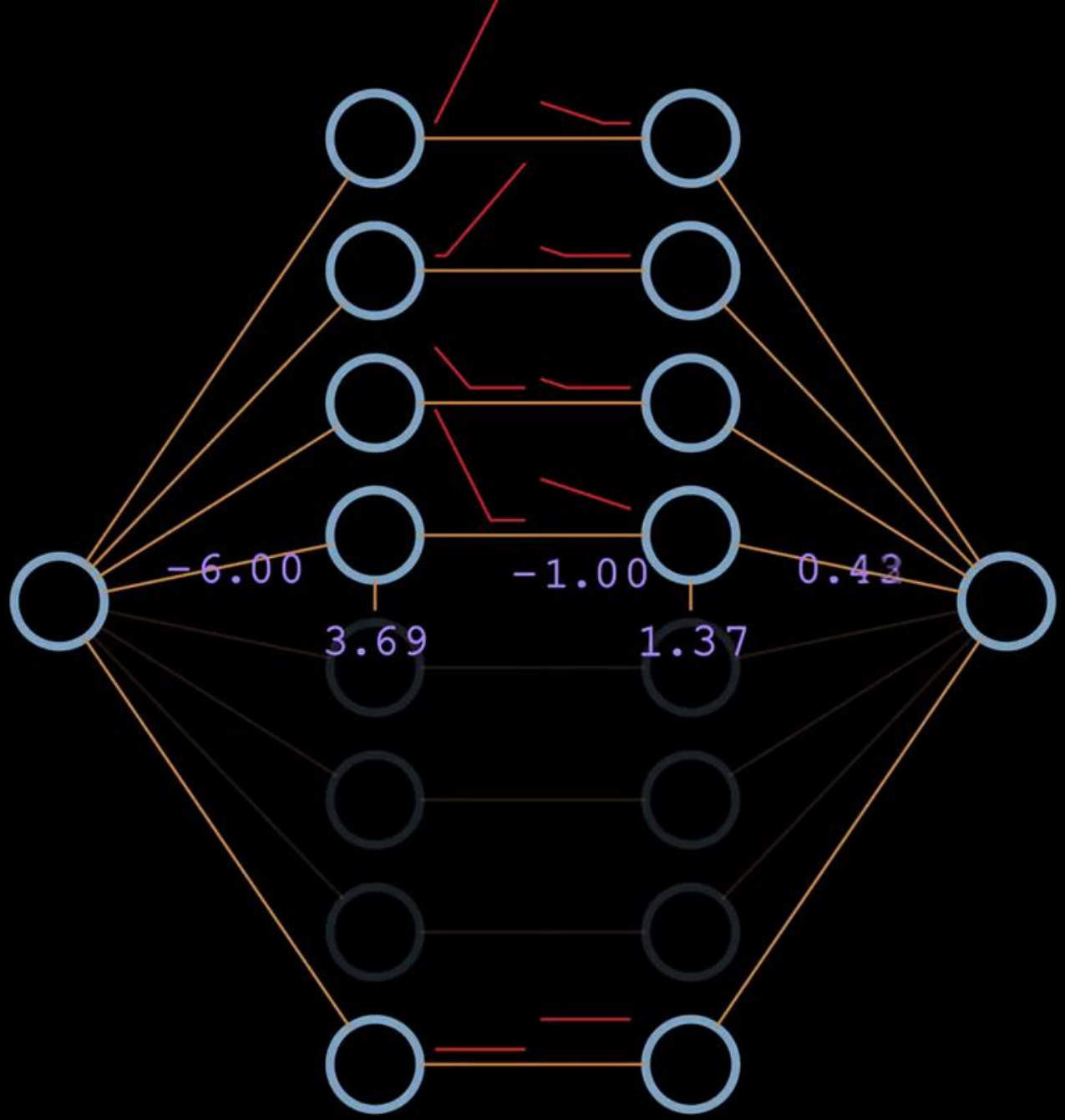


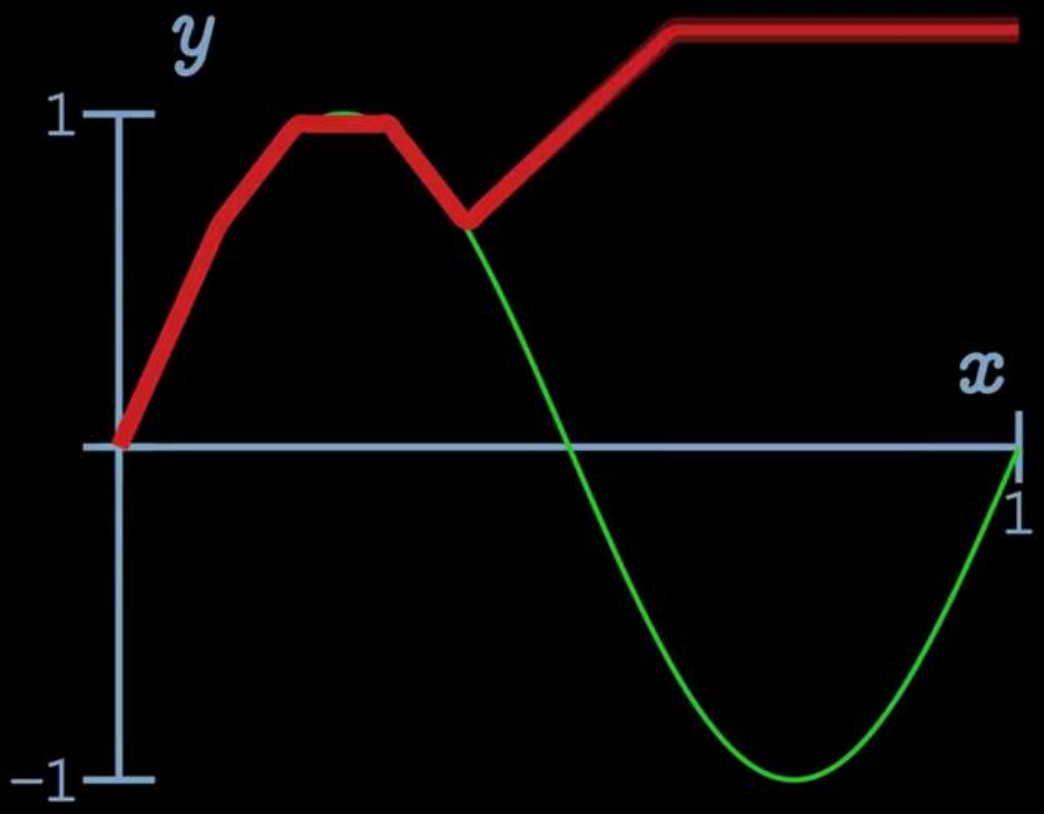
https://nnfs.io



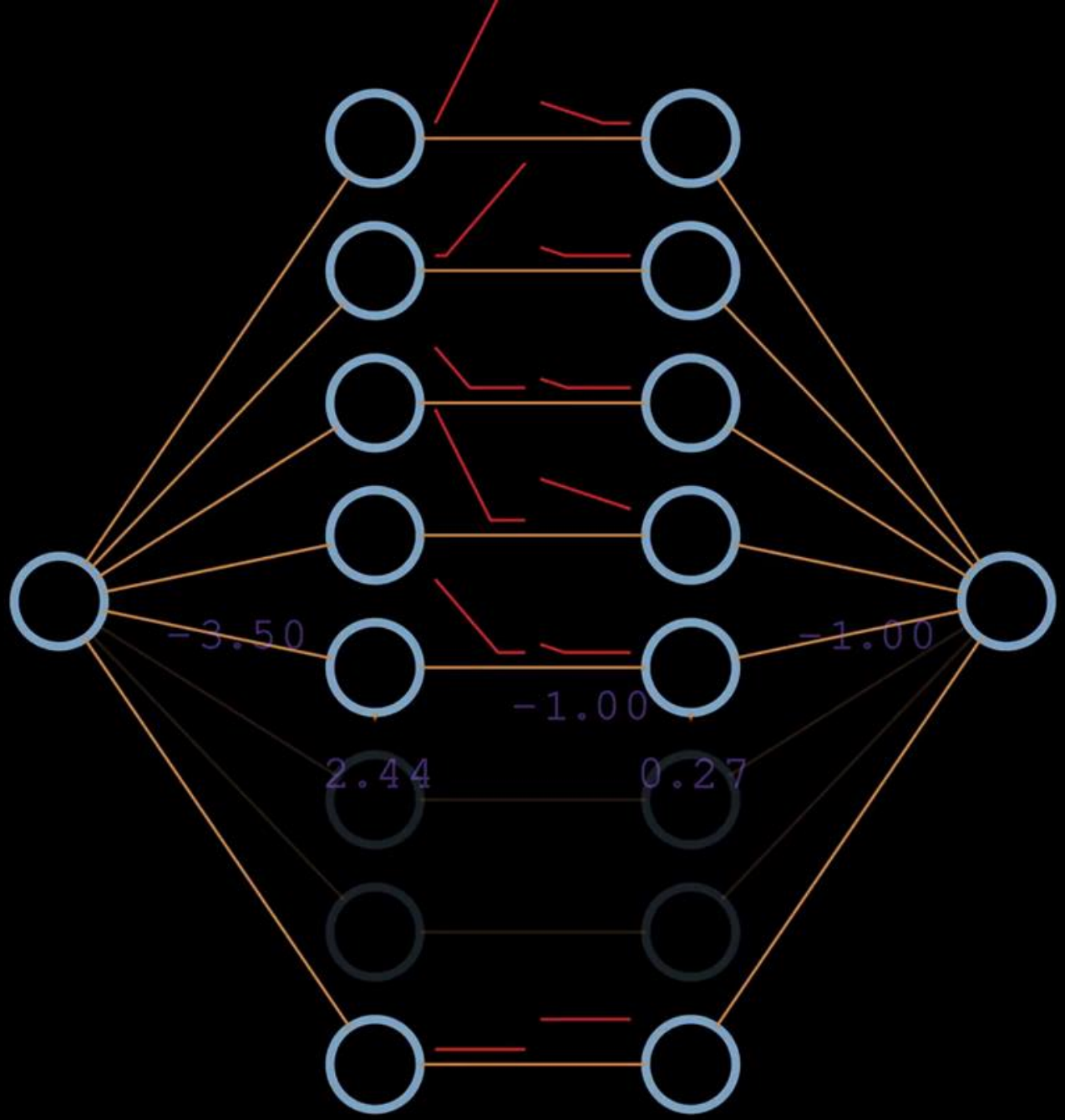


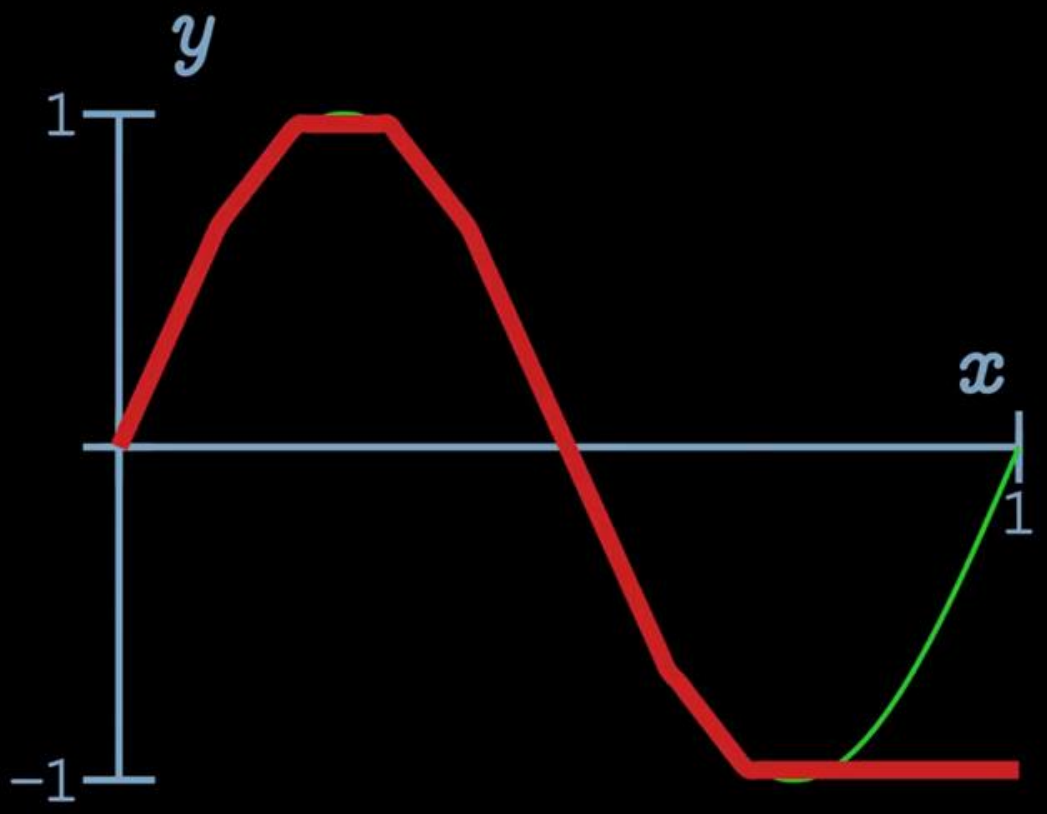
https://nnfs.io



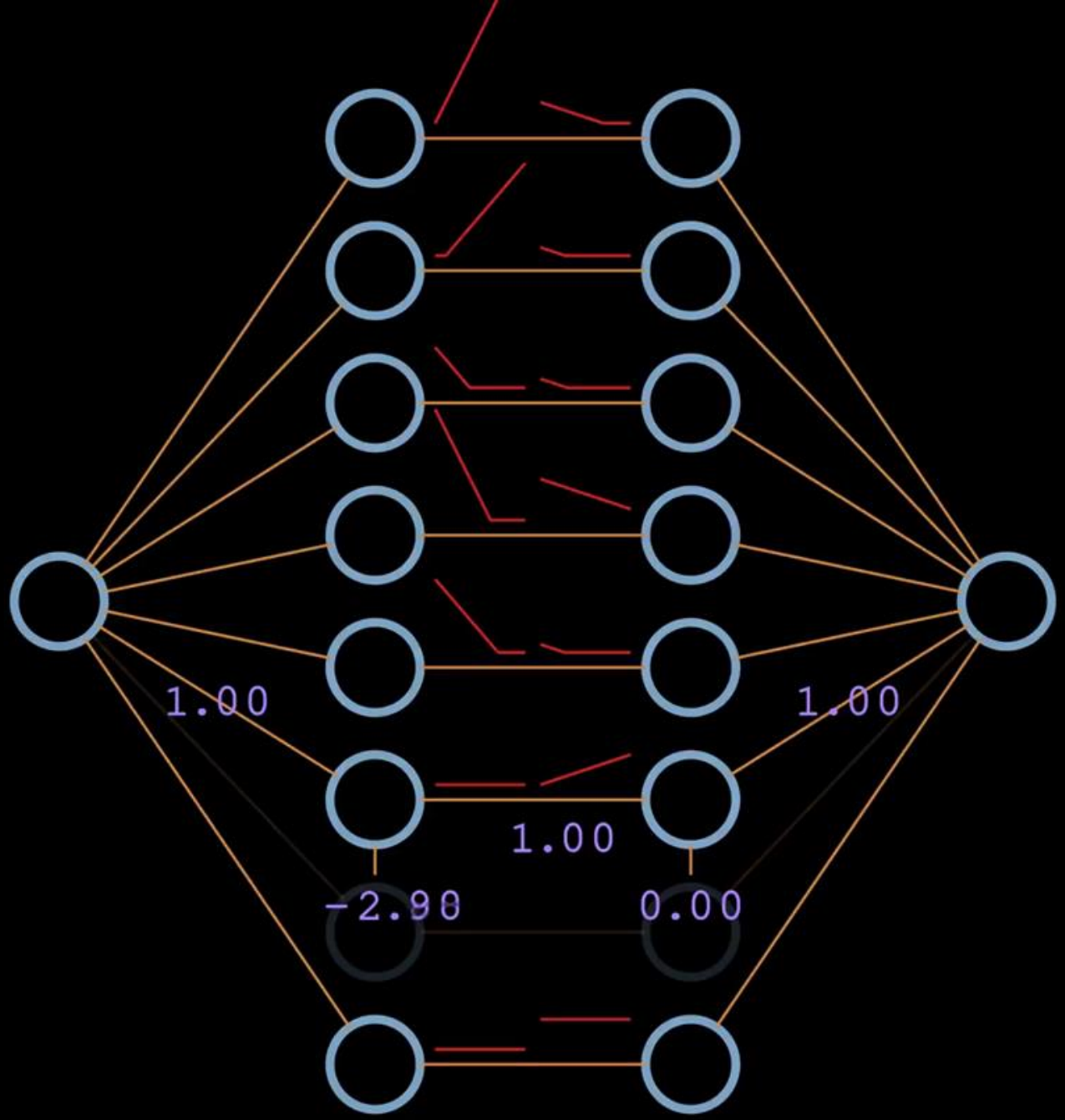


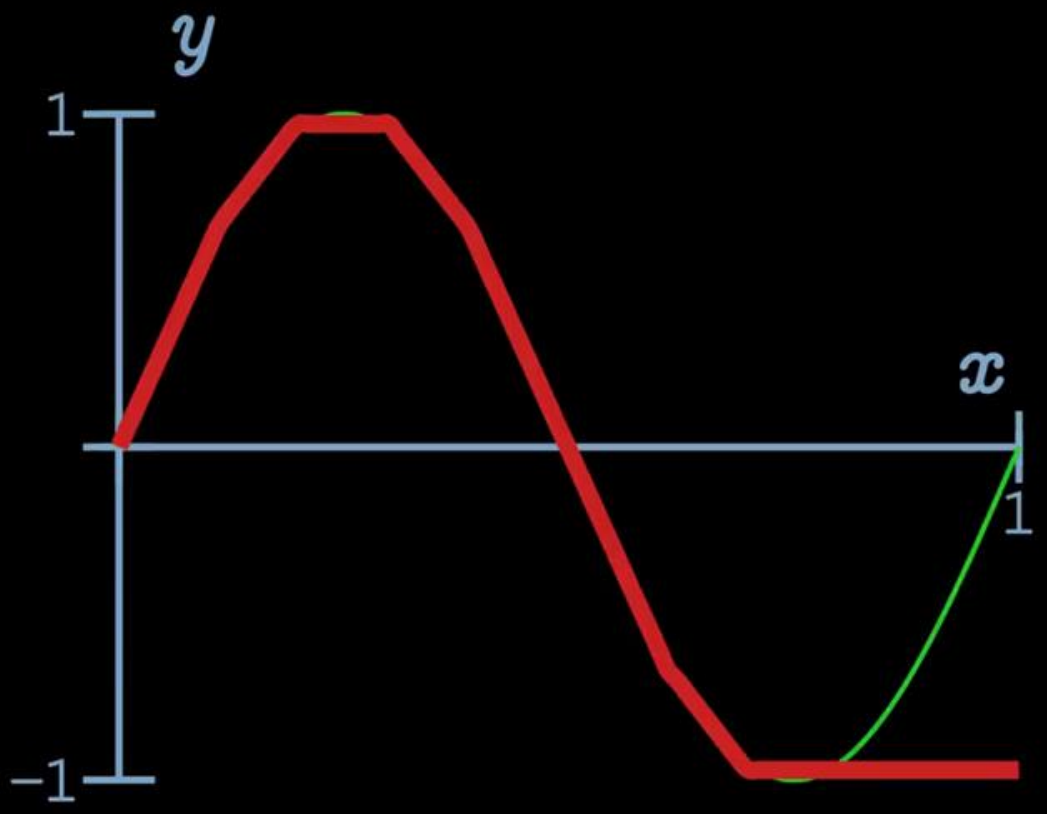
https://nnfs.io



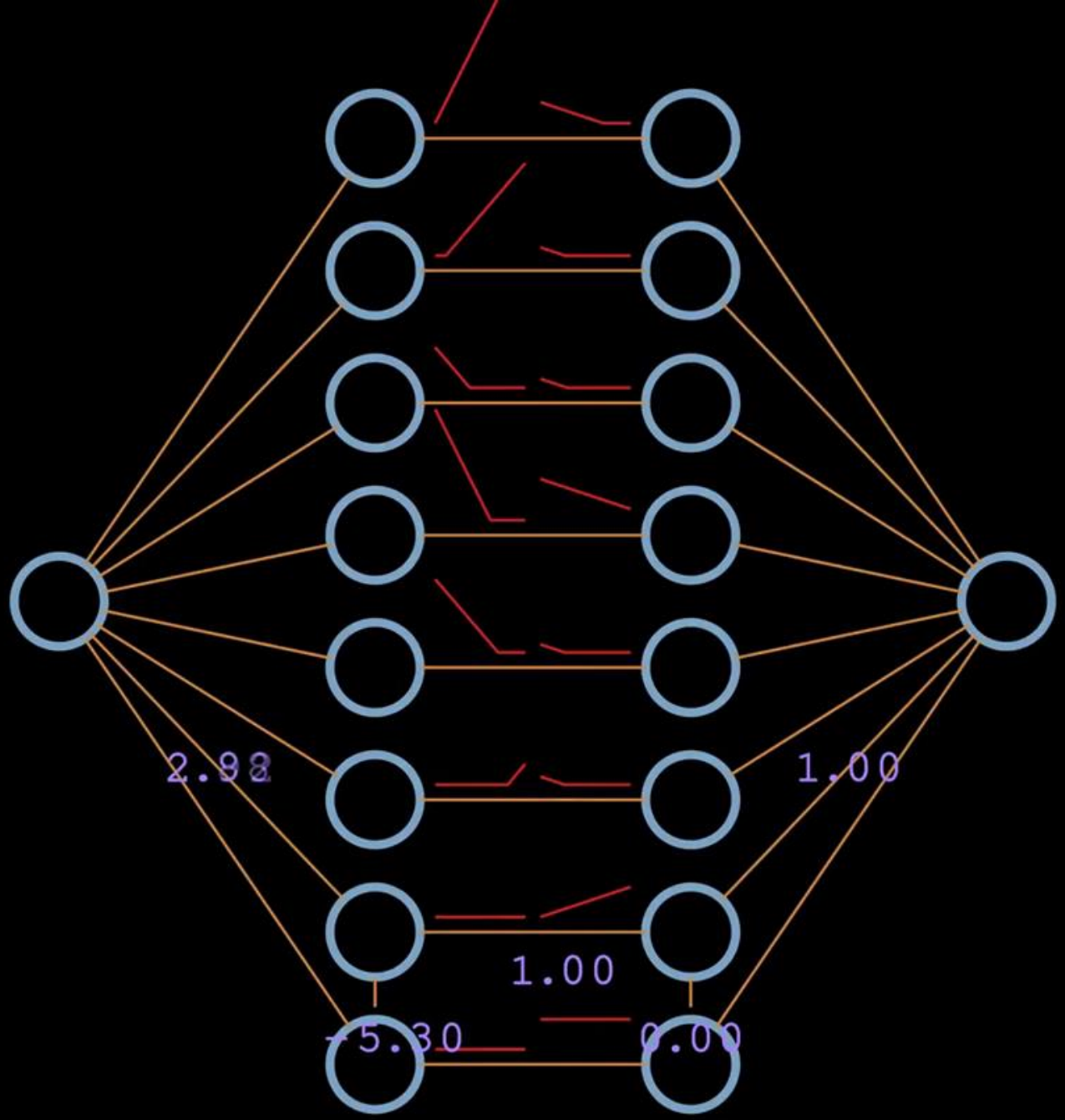


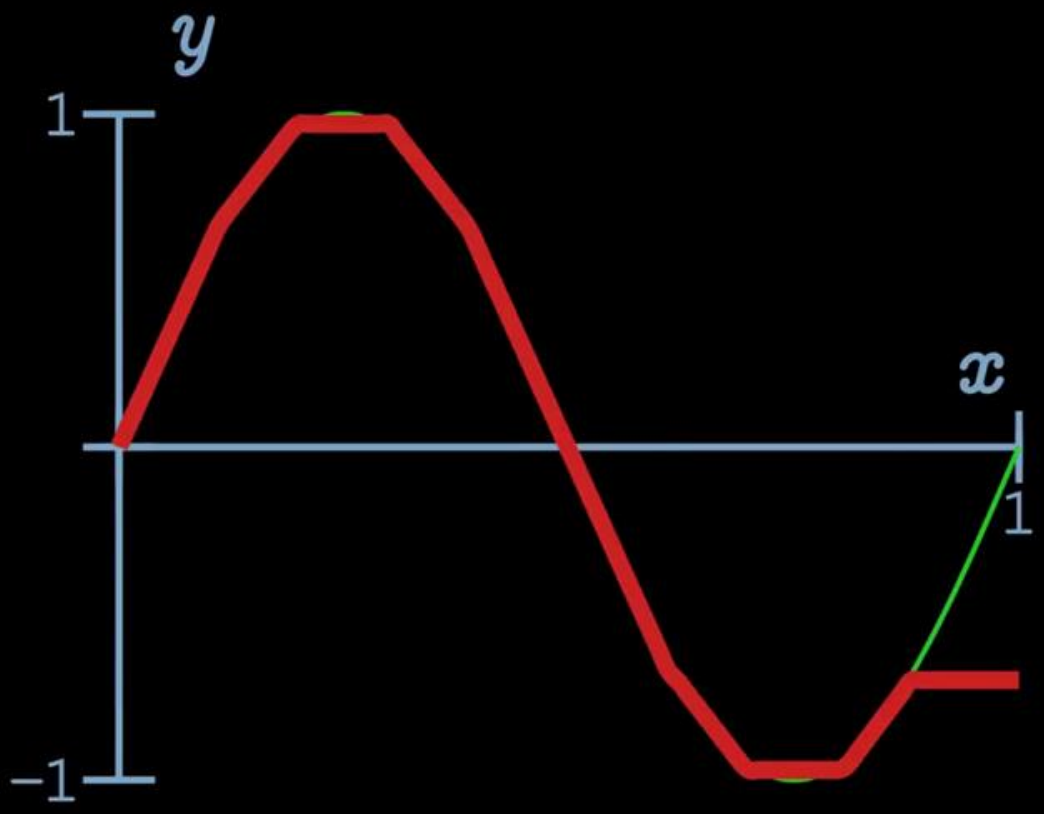
https://nnfs.io



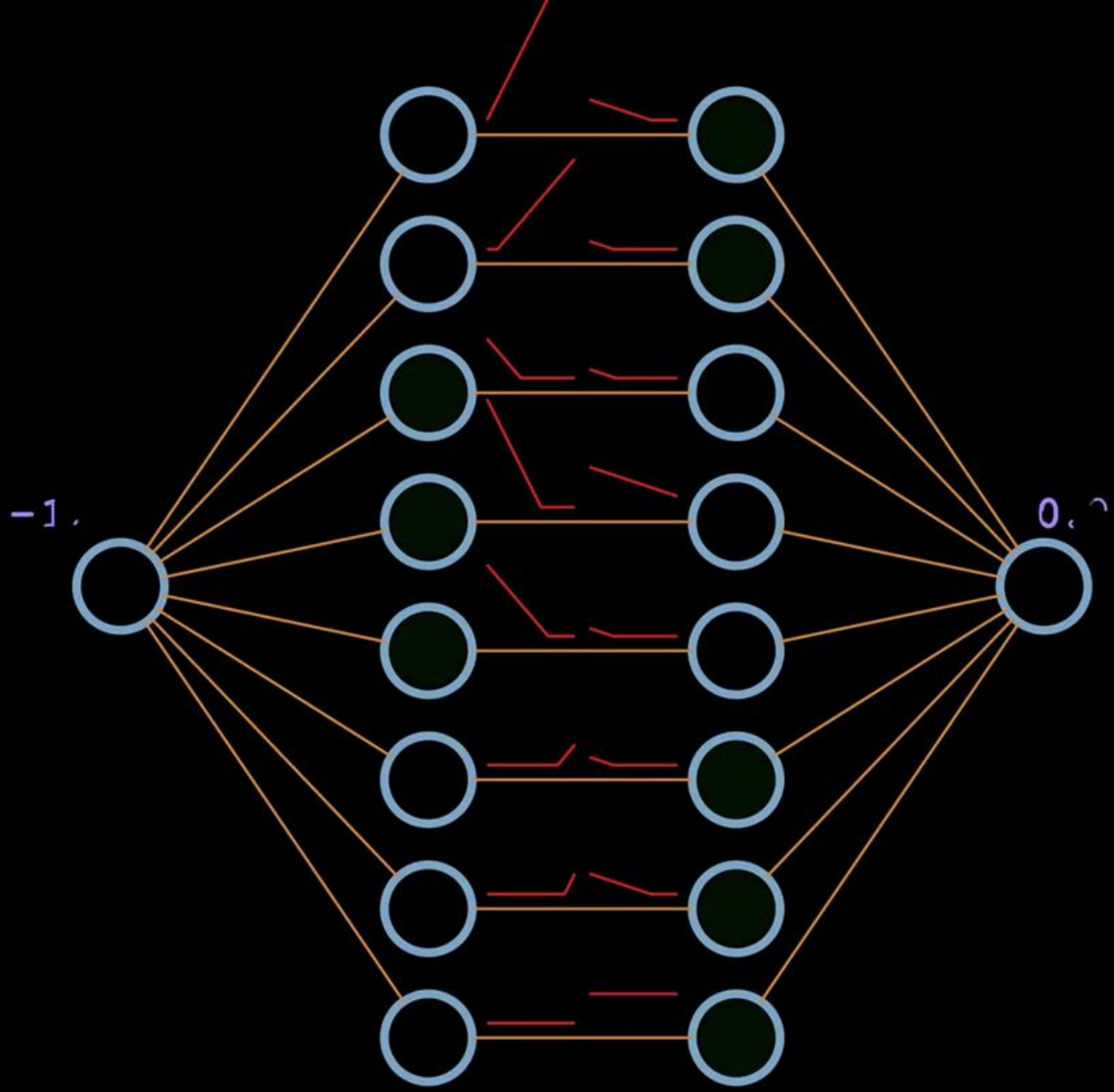


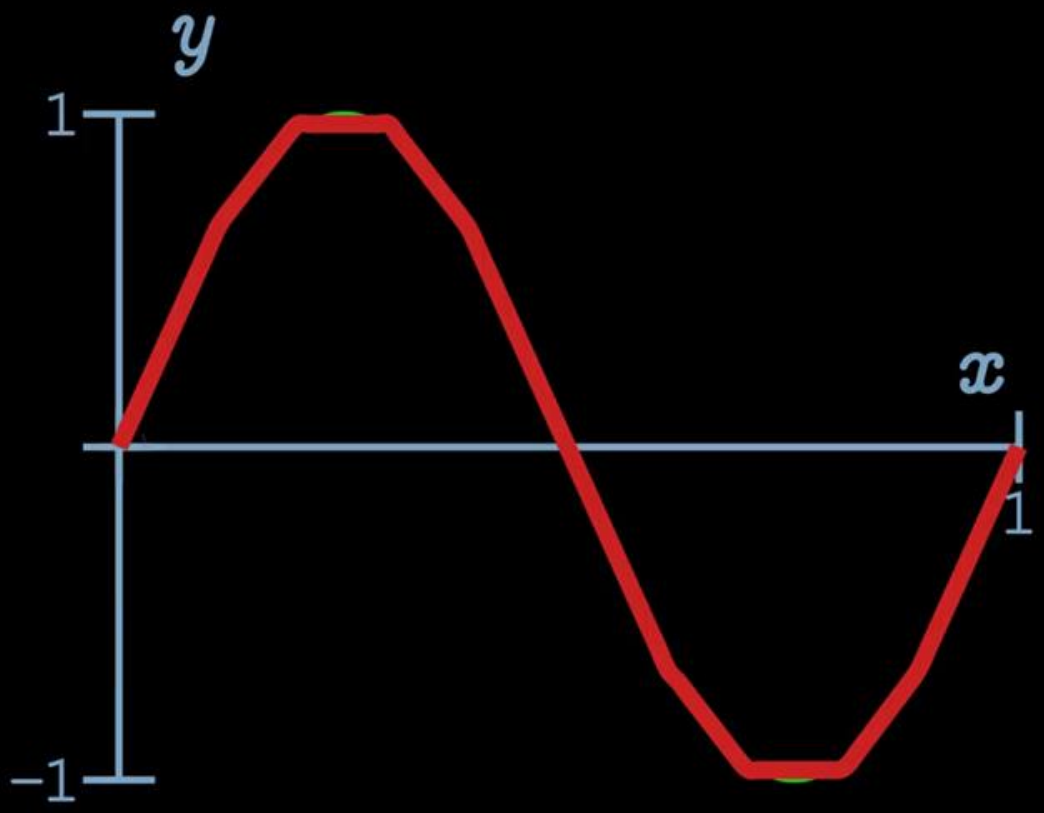
https://nnfs.io

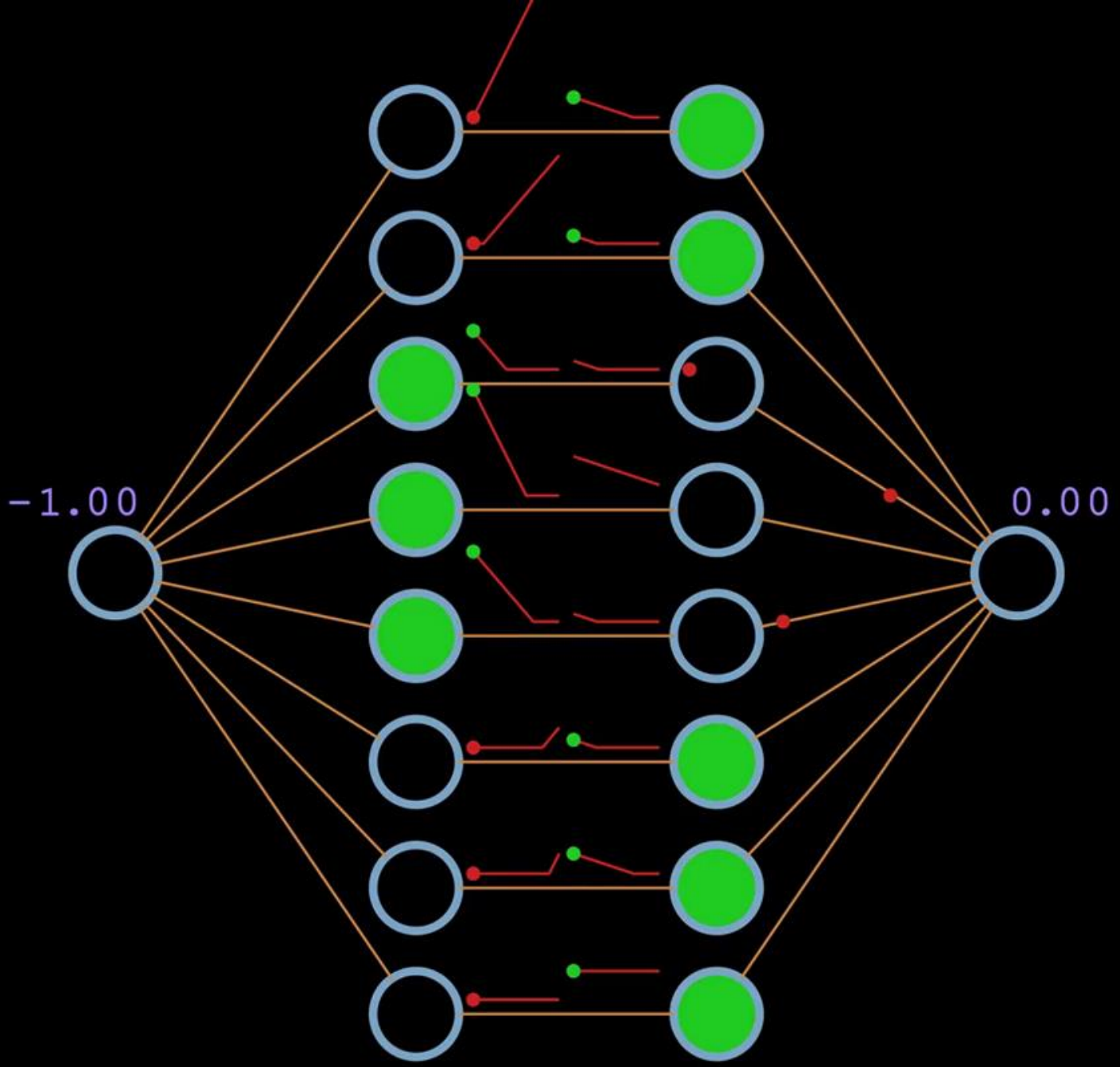


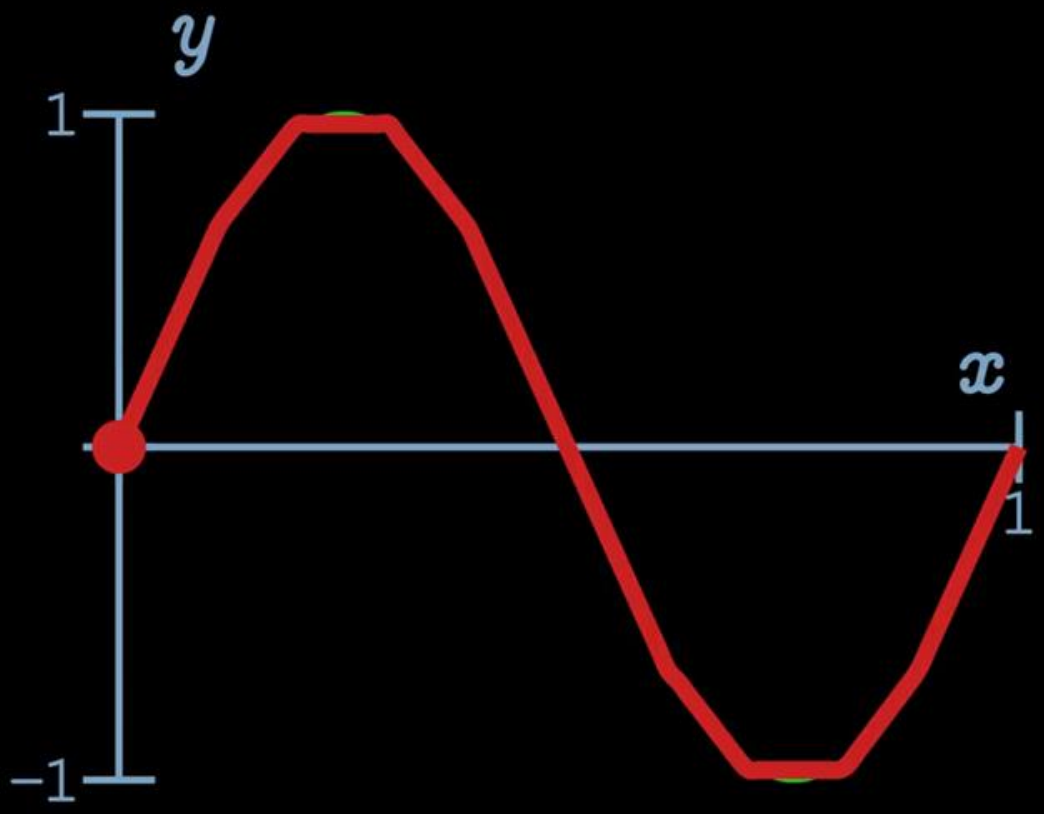


https://nnfs.io

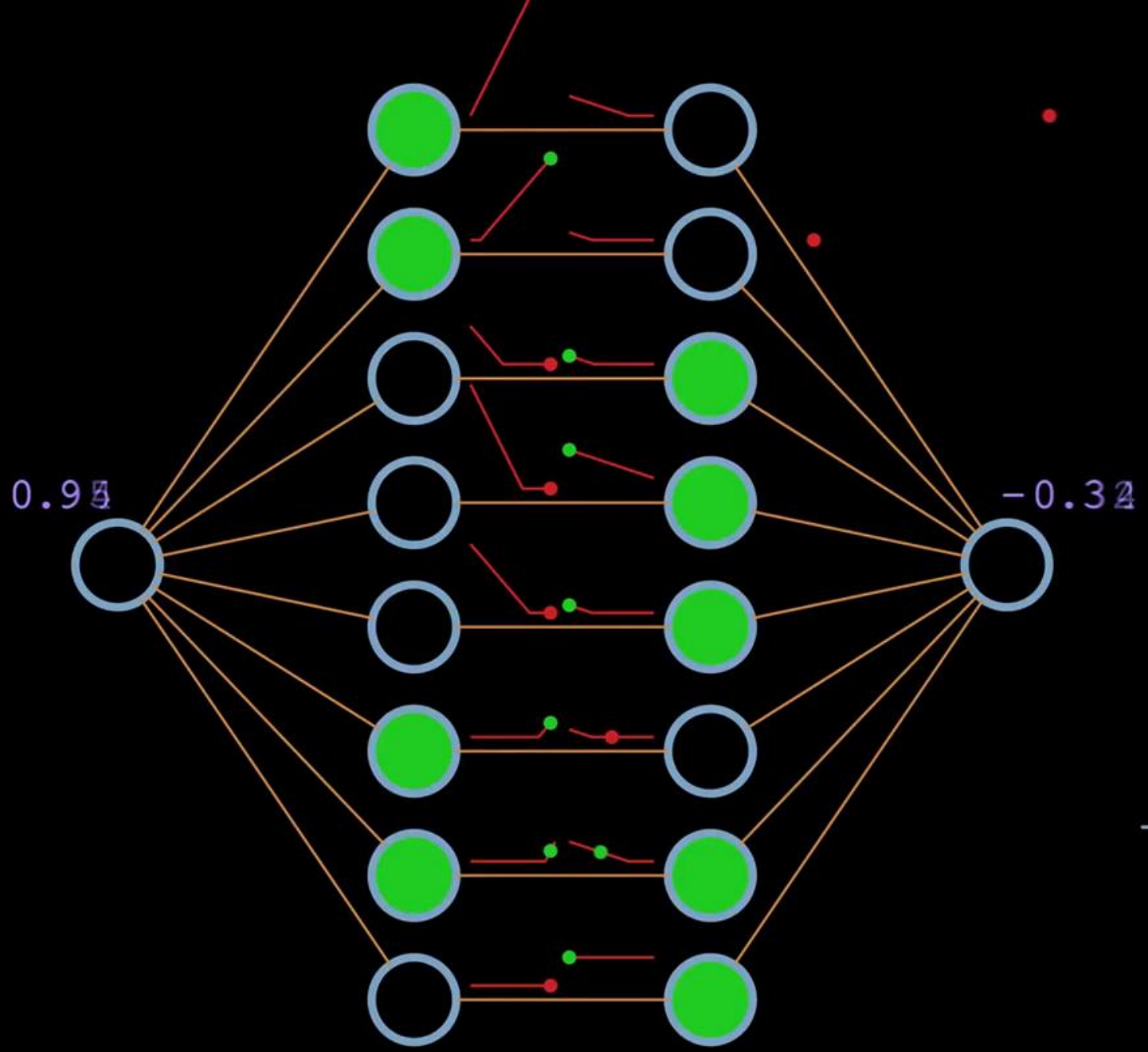


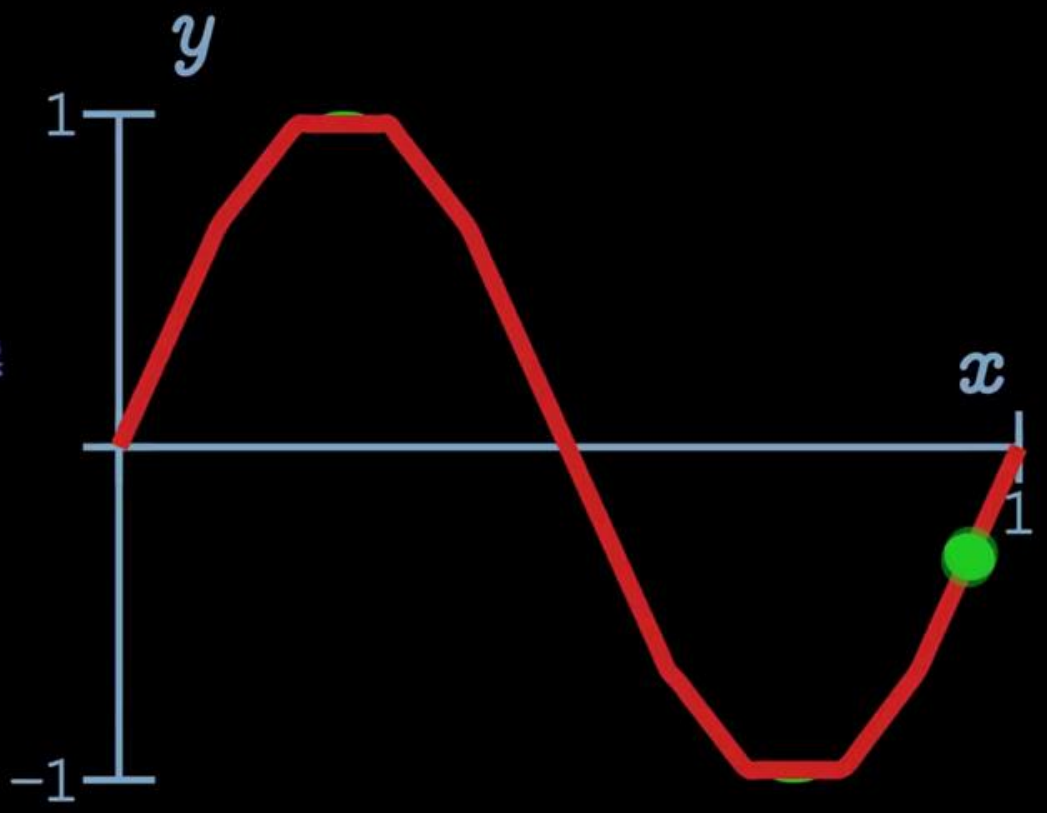






https://nnfs.io





https://nnfs.io

p5.py | cd.py

```python
import numpy as np

np.random.seed(0)

X = [[1, 2, 3, 2.5],
     [2.0, 5.0, -1.0, 2.0],
     [-1.5, 2.7, 3.3, -0.8]]


inputs = [0, 2, -1, 3.3, -2.7, 1.1, 2.2, -100]
output = []

for i in inputs:
    if i > 0:
        output.append(i)
    elif i <= 0:
        output.append(0)

print(output)

'''
class Layer_Dense:
    def __init__(self, n_inputs, n_neurons):
        self.weights = 0.10 * np.random.randn(n_inputs, n_neurons)
```

```
[0, 2, 0, 3.3, 0, 1.1, 2.2, 0]
[Finished in 0.2s]
```

p5.py

```python
import numpy as np

np.random.seed(0)

X = [[1, 2, 3, 2.5],
     [2.0, 5.0, -1.0, 2.0],
     [-1.5, 2.7, 3.3, -0.8]]


class Layer_Dense:
    def __init__(self, n_inputs, n_neurons):
        self.weights = 0.10 * np.random.randn(n_inputs, n_neurons)
        self.biases = np.zeros((1, n_neurons))
    def forward(self, inputs):
        self.output = np.dot(inputs, self.weights) + self.biases

class Activation_ReLU:
    def forward(self, inputs):
        self.output = np.maximum(0, inputs)


layer1 = Layer_Dense(4,5)
layer2 = Layer_Dense(5,2)
```

```
[0, 2, 0, 3.3, 0, 1.1, 2.2, 0]
[Finished in 0.2s]
```

h@Boxx: ~

File Edit View Search Terminal Help

```
h@Boxx:~$ pip install nnfs
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: nnfs in ./.local/lib/python3.7/site-packages (0.4
.0)
Requirement already satisfied: numpy in ./.local/lib/python3.7/site-packages (fr
om nnfs) (1.18.2)
WARNING: You are using pip version 20.0.2; however, version 20.1 is available.
You should consider upgrading via the '/usr/bin/python3.7 -m pip install --upgra
de pip' command.
h@Boxx:~$
```

```
layer1 = Layer_Dense(4,5)
layer2 = Layer_Dense(5,2)
```

```
[0, 2, 0, 3.3, 0, 1.1, 2.2, 0]
[Finished in 0.2s]
```

h@Boxx: ~

File Edit View Search Terminal Help

```
Requirement already satisfied: numpy in ./.local/lib/python3.7/site-packages (fr
om nnfs) (1.18.2)
WARNING: You are using pip version 20.0.2; however, version 20.1 is available.
You should consider upgrading via the '/usr/bin/python3.7 -m pip install --upgra
de pip' command.
h@Boxx:~$ nnfs

Neural Networks from Scratch in Python Tool.

Basic usage:
nnfs command [parameter1 [parameter2]]

Detailed usage:
nnfs info | code video_part [destination]

Commands:
  info    Prints information about the book
  code    Creates a file containing the code of given video part
          in given location. Location is optional, example:
          nnfs code 2 nnfs/p02.py
          will create a file p02.py in a nnfs folder containing
          the code of part 2 of video tutorial

h@Boxx:~$
```

```
layer1 = Layer_Dense(4,5)
layer2 = Layer_Dense(5,2)
```

```
[0, 2, 0, 3.3, 0, 1.1, 2.2, 0]
[Finished in 0.2s]
```

p5.py

```python
import numpy as np

np.random.seed(0)

X = [[1, 2, 3, 2.5],
     [2.0, 5.0, -1.0, 2.0],
     [-1.5, 2.7, 3.3, -0.8]]


class Layer_Dense:
    def __init__(self, n_inputs, n_neurons):
        self.weights = 0.10 * np.random.randn(n_inputs, n_neurons)
        self.biases = np.zeros((1, n_neurons))
    def forward(self, inputs):
        self.output = np.dot(inputs, self.weights) + self.biases


class Activation_ReLU:
    def forward(self, inputs):
        self.output = np.maximum(0, inputs)


layer1 = Layer_Dense(4,5)
```

```
[0, 2, 0, 3.3, 0, 1.1, 2.2, 0]
[Finished in 0.2s]
```

p5.py

cd.py

```python
import numpy as np
import nnfs

nnfs.init()

X = [[1, 2, 3, 2.5],
     [2.0, 5.0, -1.0, 2.0],
     [-1.5, 2.7, 3.3, -0.8]]


class Layer_Dense:
    def __init__(self, n_inputs, n_neurons):
        self.weights = 0.10 * np.random.randn(n_inputs, n_neurons)
        self.biases = np.zeros((1, n_neurons))
    def forward(self, inputs):
        self.output = np.dot(inputs, self.weights) + self.biases


class Activation_ReLU:
    def forward(self, inputs):
        self.output = np.maximum(0, inputs)


layer1 = Layer_Dense(4,5)
```

```
[0, 2, 0, 3.3, 0, 1.1, 2.2, 0]
[Finished in 0.2s]
```

p5.py

```python
import numpy as np
import nnfs
from nnfs.datasets import spiral_data

nnfs.init()

X = [[1, 2, 3, 2.5],
     [2.0, 5.0, -1.0, 2.0],
     [-1.5, 2.7, 3.3, -0.8]]



class Layer_Dense:
    def __init__(self, n_inputs, n_neurons):
        self.weights = 0.10 * np.random.randn(n_inputs, n_neurons)
        self.biases = np.zeros((1, n_neurons))
    def forward(self, inputs):
        self.output = np.dot(inputs, self.weights) + self.biases


class Activation_ReLU:
    def forward(self, inputs):
```

```
here
[Finished in 58.6s]
```

p5.py

```
X = [[1, 2, 3, 2.5],
     [2.0, 5.0, -1.0, 2.0],
     [-1.5, 2.7, 3.3, -0.8]]


X, y = spiral_data(100, 3)


class Layer_Dense:
    def __init__(self, n_inputs, n_neurons):
        self.weights = 0.10 * np.random.randn(n_inputs, n_neurons)
        self.biases = np.zeros((1, n_neurons))
    def forward(self, inputs):
        self.output = np.dot(inputs, self.weights) + self.biases


class Activation_ReLU:
    def forward(self, inputs):
        self.output = np.maximum(0, inputs)


layer1 = Layer_Dense(4,5)
layer2 = Layer_Dense(5,2)
```

```
here
[Finished in 58.6s]
```

p5.py cd.py

```
16      def __init__(self, n_inputs, n_neurons):
17          self.weights = 0.10 * np.random.randn(n_inputs, n_neurons)
18          self.biases = np.zeros((1, n_neurons))
19      def forward(self, inputs):
20          self.output = np.dot(inputs, self.weights) + self.biases
21
22
23  class Activation_ReLU:
24      def forward(self, inputs):
25          self.output = np.maximum(0, inputs)
26
27
28  layer1 = Layer_Dense(2,5)
29  activation1 = Activation_ReLU()
30
31  layer1.forward(X)
32
33  print(layer1.output)
34  #activation1.foward(layer1.output)
35
```

```
[[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00]
 [-8.35815910e-04 -7.90404272e-04 -1.33452227e-03  4.65504505e-04
   4.56846210e-05]
 [-2.39994470e-03  5.93469958e-05 -2.24808278e-03  2.03573116e-04
   6.10024377e-04]
 ...
 [ 1.13291524e-01 -1.89262271e-01 -2.06855070e-02  8.11079666e-02
  -6.71350807e-02]
 [ 1.34588361e-01 -1.43197834e-01  3.09493970e-02  5.66337556e-02
  -6.29687458e-02]
 [ 1.07817926e-01 -2.00809643e-01 -3.37579325e-02  8.72561932e-02
  -6.81458861e-02]]
[Finished in 0.2s]
```