

Lab 5

Study of activation

Functions and their Role

Aim: To study and analyze different activation functions used in deep learning and neural networks, understand their mathematical formulation, visualize their graphs, and observe their effect on model training.

Description:

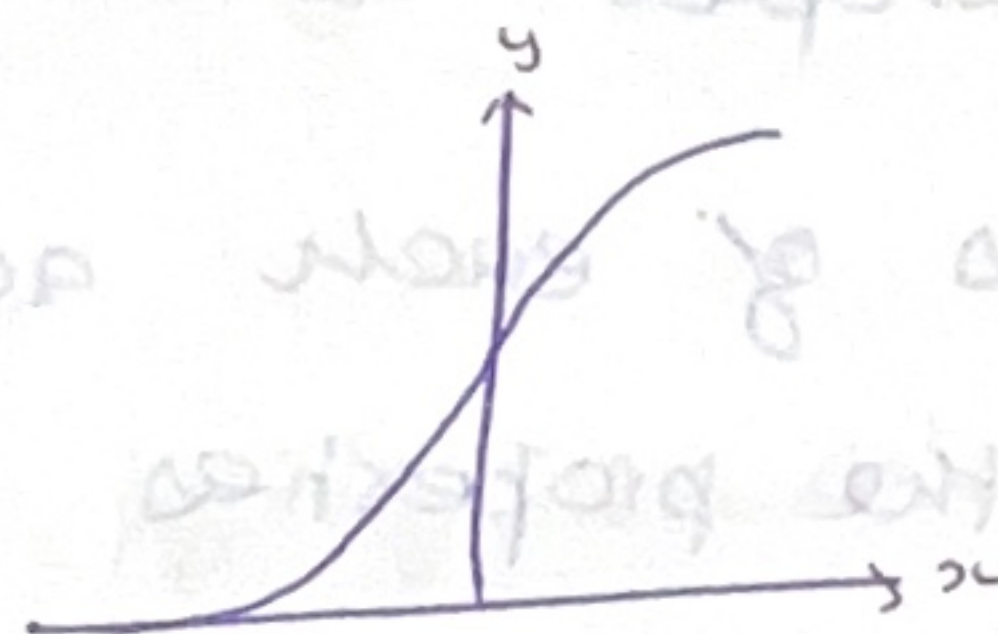
Activation functions introduce non-linearity into neural network, enabling them to learn complex patterns. Without activation layer, the network would act like a linear regression model regardless of the number of layers.

(i) Sigmoid function

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

range: $(0, 1)$

\Rightarrow suffers from vanishing gradient



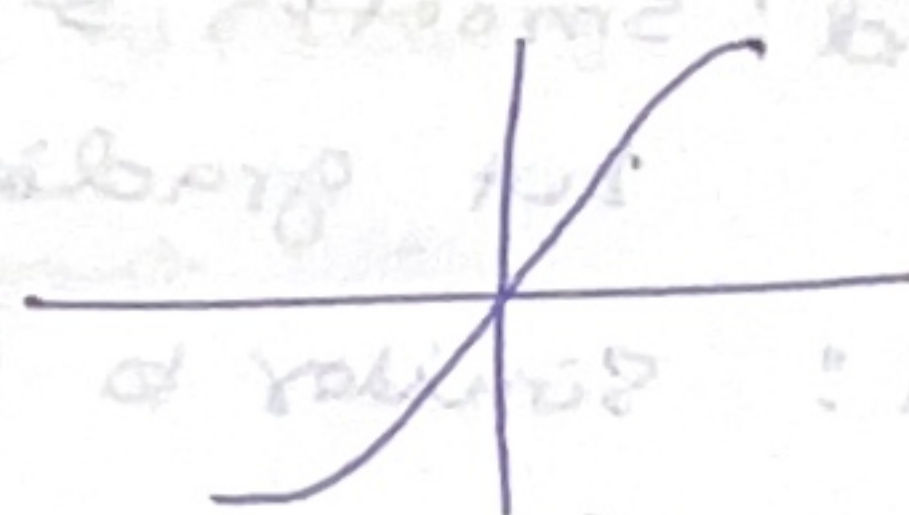
(ii) Hyperbolic tangent (tanh)

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

range: $(-1, 1)$

centered at zero

also suffers from vanishing gradient

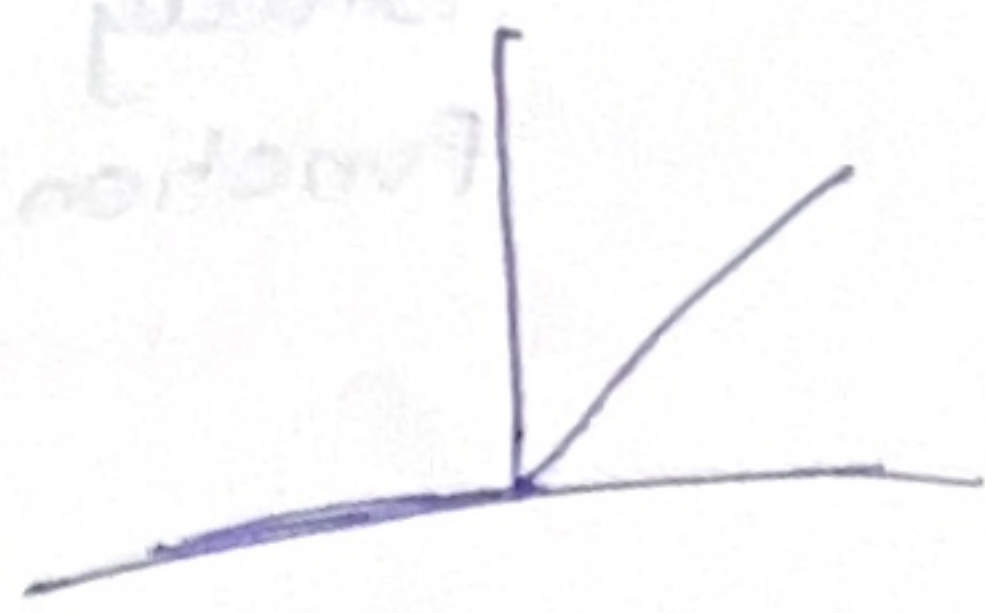


(iii) Rectified Linear Unit (ReLU):

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

range: $[0, \infty]$

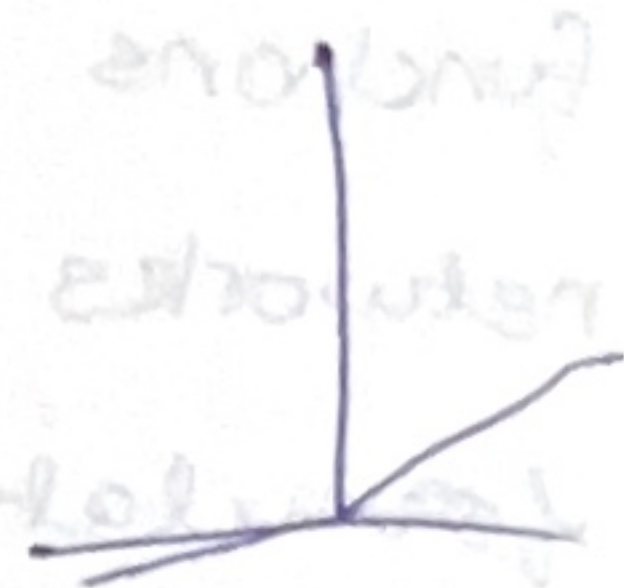
avoids vanishing gradients



(iv) Leaky ReLU:

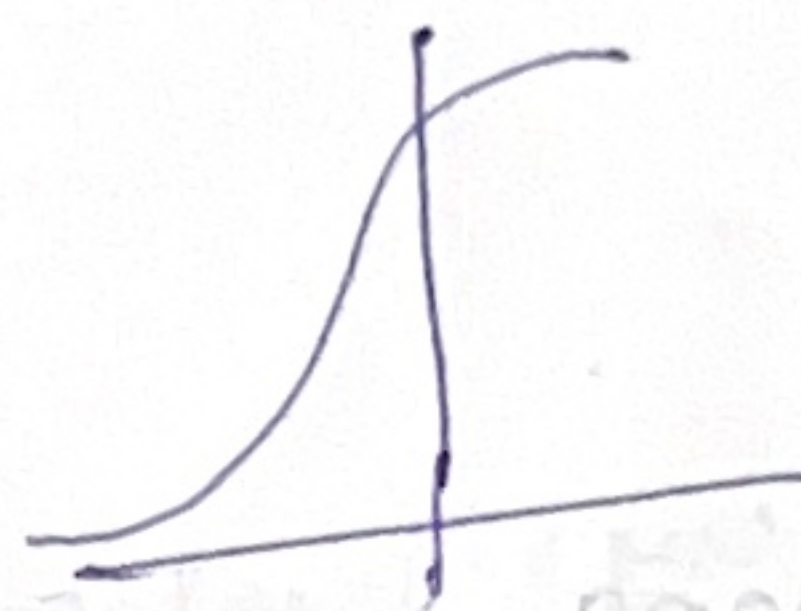
$$f(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases}$$

allows small negative slope



(v) Softmax Function

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$



Procedure:

1. Import necessary libraries (Numpy, matplotlib)
2. Define each activation function mathematically
3. Generate input values and compute outputs
4. Plot graphs of each activation function
5. Observe the properties

Observation

- Sigmoid: smooth S-shaped, compresses values between (0,1) but gradients vanish for large $|x|$.
- Tanh: similar to sigmoid but centered at 0.
- ReLU: Efficient, sparse activation
- Leaky ReLU: Solves ReLU's issue of negative inputs
- Softmax: used for classification, outputs class probabilities

Result:

Implementation of activation functions commonly used in deep learning was successfully done.

```
plt.grid(True)
```

```
plt.subplot(2, 3, 2)
```

```
plt.plot(Z, t, label="Tanh", color='orange')
```

```
plt.title("Tanh")
```

```
plt.grid(True)
```

```
plt.subplot(2, 3, 3)
```

```
plt.plot(Z, r, label="Relu", color='green')
```

```
plt.title("Relu")
```

```
plt.grid(True)
```

```
plt.subplot(2, 3, 4)
```

```
plt.plot(Z, lr, label="Leaky Relu", color='red')
```

```
plt.title("Leaky Relu")
```

```
plt.grid(True)
```

```
plt.subplot(2, 3, 5)
```

```
# Plot softmax of the three example classes
```

```
plt.plot(Z, softmax_vals[0], label='Class 1')
```

```
plt.plot(Z, softmax_vals[1], label='Class 2')
```

```
plt.plot(Z, softmax_vals[2], label='Class 3')
```

```
plt.title("Softmax (3 classes)")
```

```
plt.legend()
```

```
plt.grid(True)
```

```
plt.tight_layout()
```

```
[2]: import numpy as np
import matplotlib.pyplot as plt

# Activation functions
def sigmoid(Z):
    return 1 / (1 + np.exp(-Z))

def tanh(Z):
    return np.tanh(Z)

def relu(Z):
    return np.maximum(0, Z)

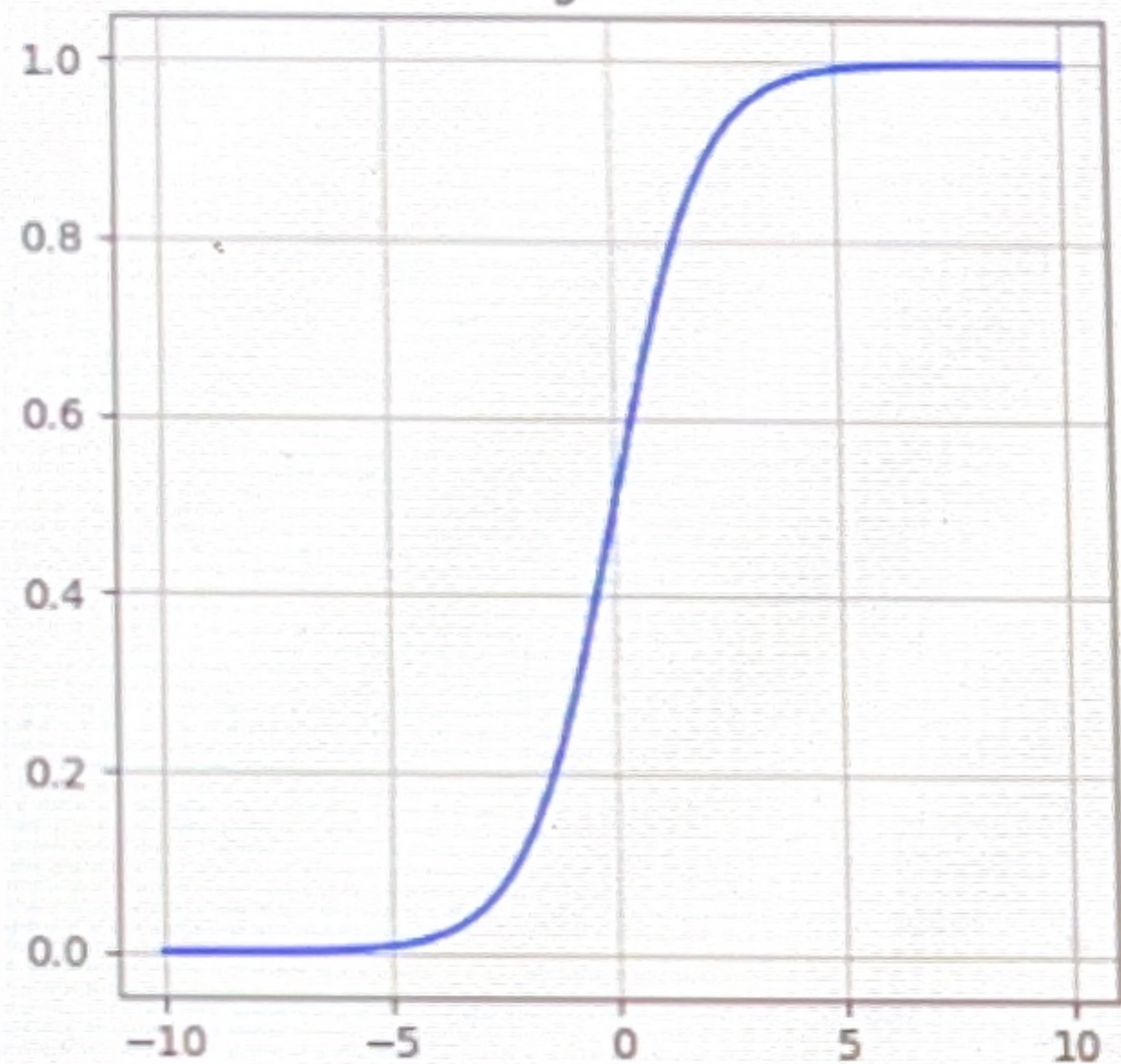
def leaky_relu(Z, alpha=0.01):
    return np.where(Z > 0, Z, alpha * Z)

def softmax(Z):
    expZ = np.exp(Z - np.max(Z))
    return expZ / np.sum(expZ)

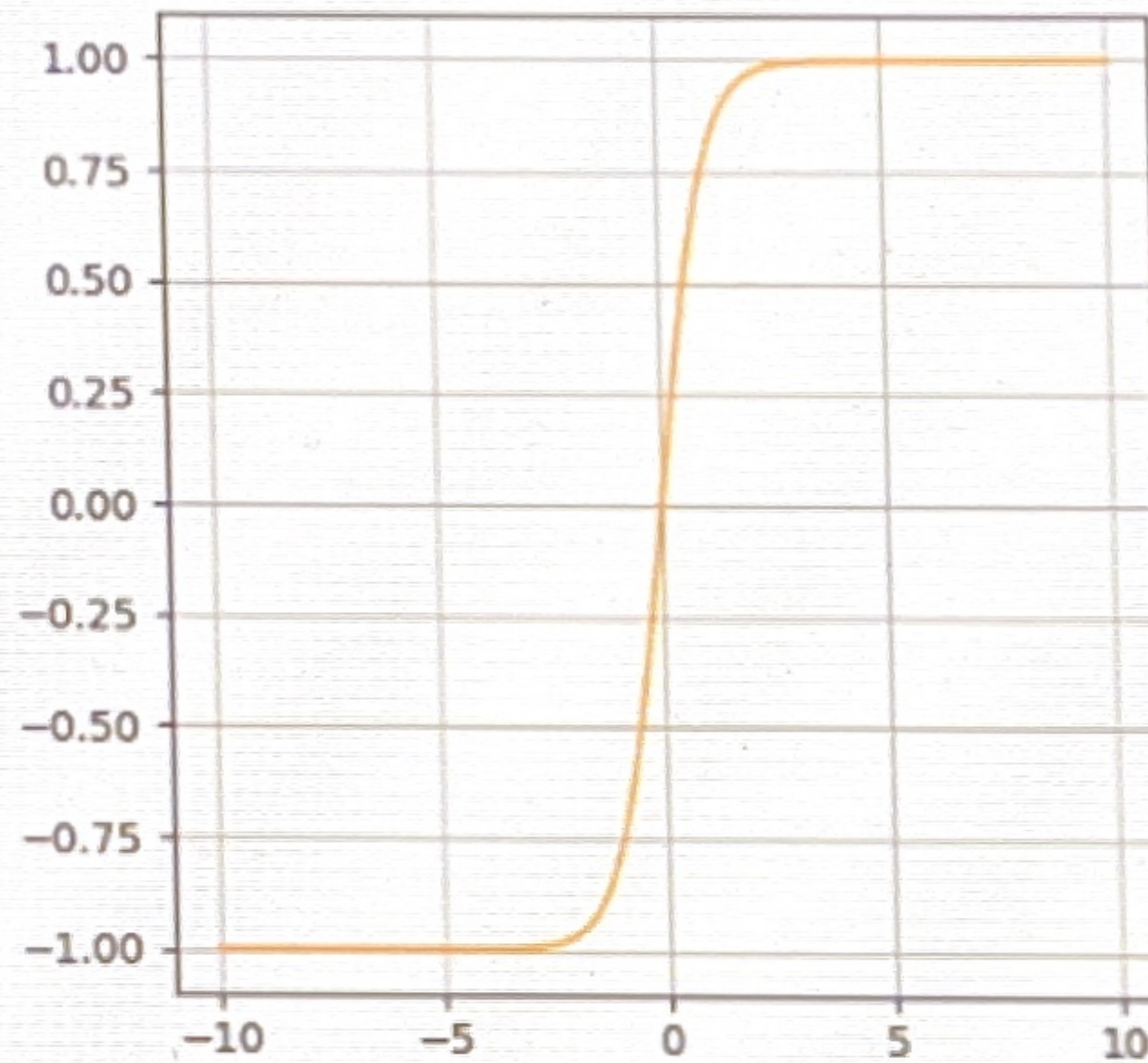
Z = np.linspace(-10, 10, 400)

sig = sigmoid(Z)
t = tanh(Z)
```

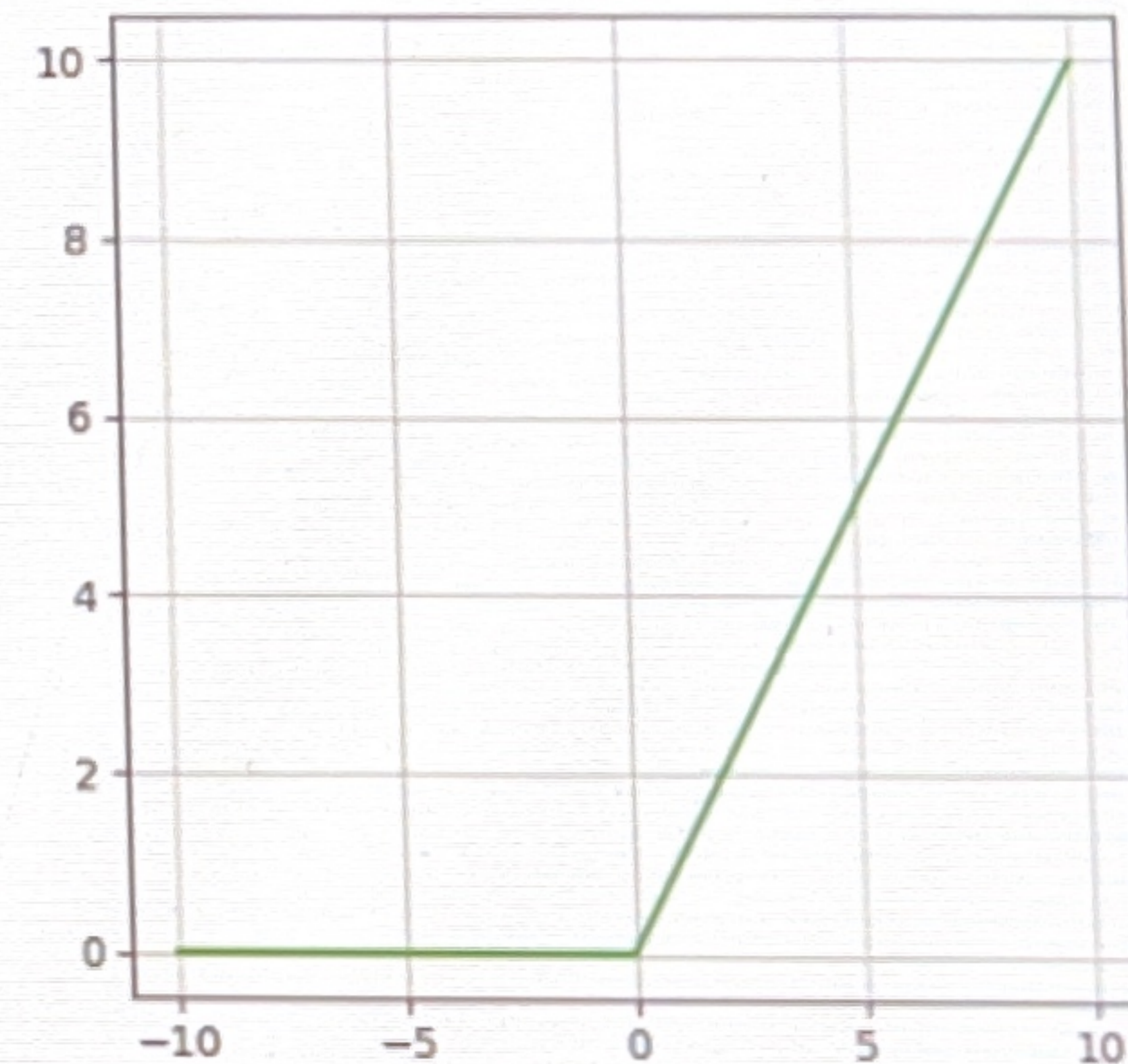
Sigmoid



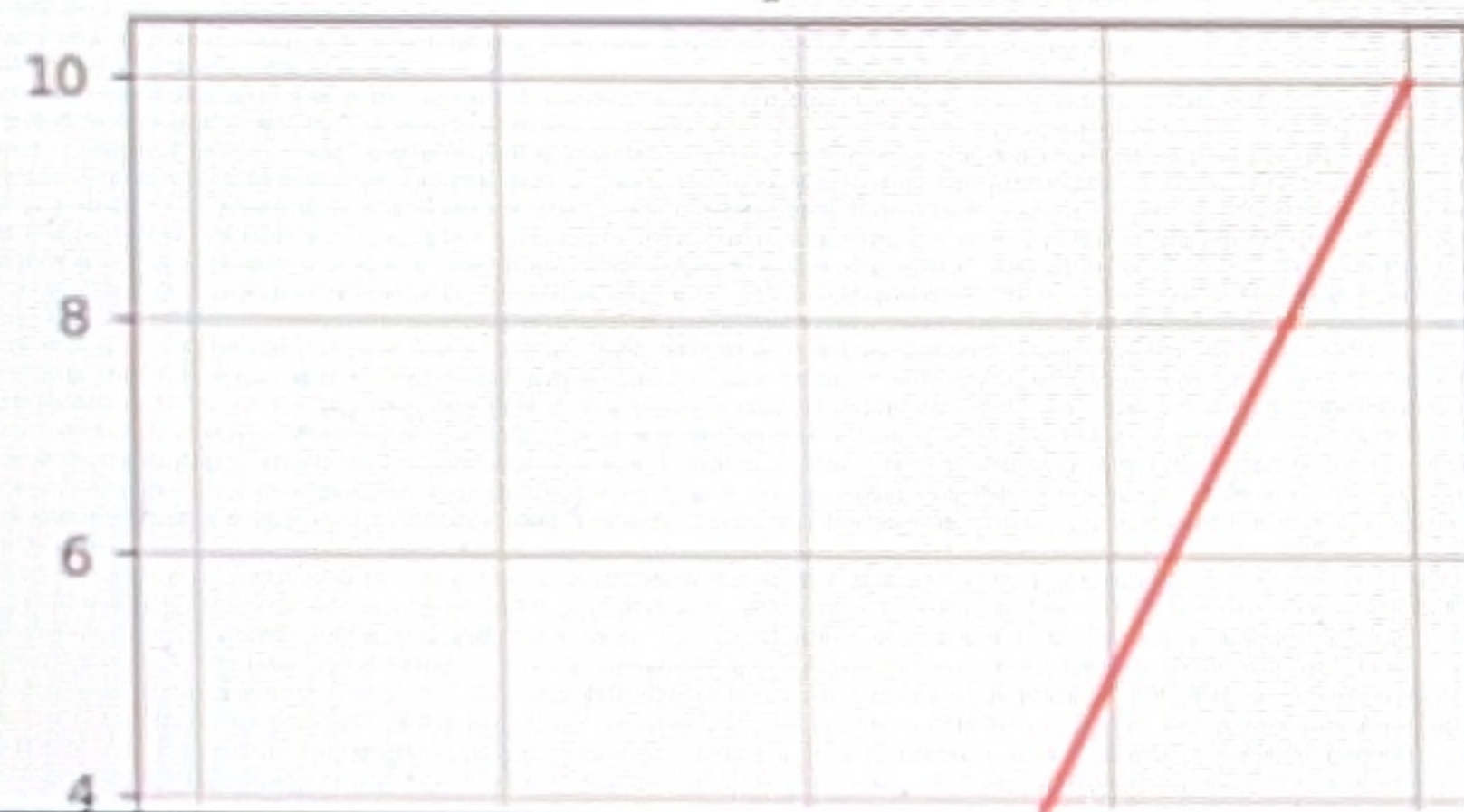
Tanh



ReLU



Leaky ReLU



Softmax (3 classes)

