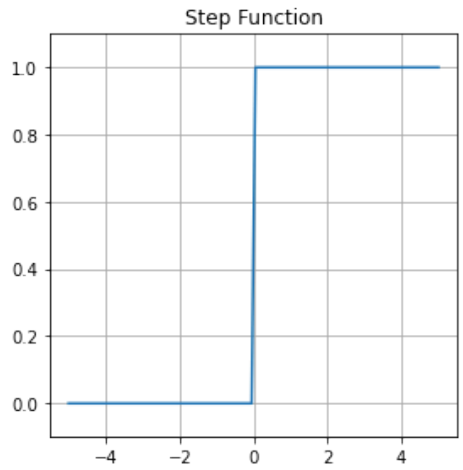
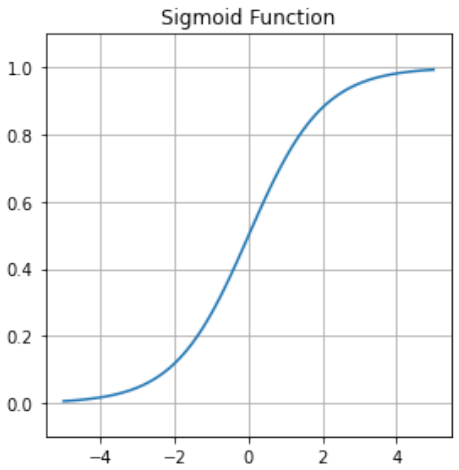
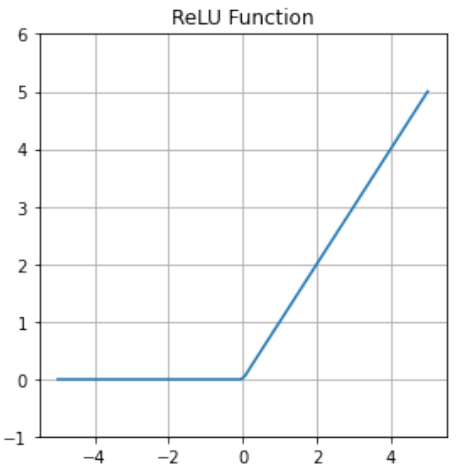
In [10]:

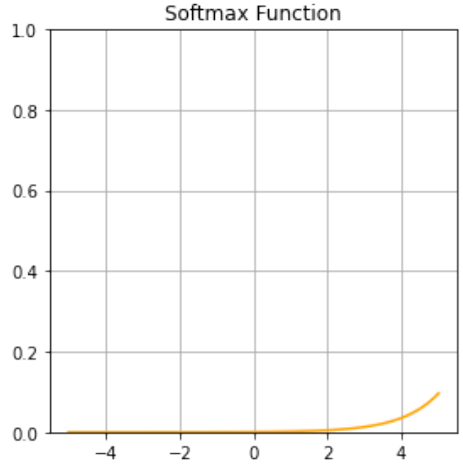
```python
import numpy as np
import matplotlib.pyplot as plt
def step_function(x):
    return np.where(x < 0, 0, 1)
def sigmoid_function(x):
    return 1 / (1 + np.exp(-x))
def relu_function(x):
    return np.maximum(0, x)
def softmax_function(x):
    exp_x = np.exp(x - np.max(x))
    return exp_x / np.sum(exp_x)
def tanh_function(x):
    return np.tanh(x)
x = np.linspace(-5, 5, 100)
step_values = step_function(x)
sigmoid_values = sigmoid_function(x)
relu_values = relu_function(x)
softmax_values = softmax_function(x)
tanh_values = tanh_function(x)
plt.figure(figsize=(12, 8))
plt.subplot(2, 3, 1)
plt.title("Step Function")
plt.plot(x, step_values)
plt.ylim(-0.1, 1.1)
plt.grid()
plt.subplot(2, 3, 2)
plt.title("Sigmoid Function")
plt.plot(x, sigmoid_values)
plt.ylim(-0.1, 1.1)
plt.grid()
plt.subplot(2, 3, 3)
```

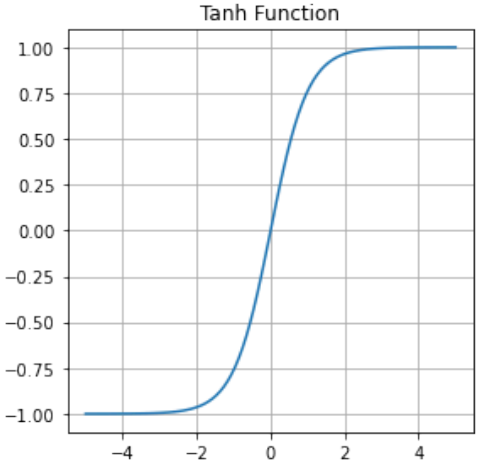
```
plt.title("ReLU Function")
plt.plot(x, relu_values)
plt.ylim(-1, 6)
plt.grid()

plt.subplot(2, 3, 4)
plt.title("Softmax Function")
plt.plot(x, softmax_function(x), label='Softmax', color='orange')
plt.ylim(0, 1)
plt.grid()
plt.subplot(2, 3, 5)
plt.title("Tanh Function")
plt.plot(x, tanh_values)
plt.ylim(-1.1, 1.1)
plt.grid()
plt.tight_layout()
plt.show()

```


Step Function
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4


Sigmoid Function
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4


ReLU Function
6
5
4
3
2
1
0
−1
−4
−2
0
2
4


Softmax Function
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4


Tanh Function
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−4
−2
0
2
4

In [11]:

```python
import numpy as np
class Neuron:
    def __init__(self, n_inputs):
        self.weights = np.random.rand(n_inputs)
        self.bias = np.random.rand(1)
    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))
    def feedforward(self, inputs):
        weighted_sum = np.dot(self.weights, inputs) + self.bias
        return self.sigmoid(weighted_sum)
if __name__ == "__main__":
    neuron = Neuron(n_inputs=3)
    inputs = np.array([0.5, 0.3, 0.2])
    output = neuron.feedforward(inputs)
    print("Weights:", neuron.weights)
    print("Bias:", neuron.bias)
    print("Inputs:", inputs)
    print("Output:", output)

```

```
Weights: [0.42648847 0.49859015 0.58836058]
Bias: [0.55513913]
Inputs: [0.5 0.3 0.2]
Output: [0.73800642]
```

In [ ]:

```python

```