Experiment 4: Plot Various Types of Activation Functions Used in Artificial Neural Networks

Aim:

To plot and visualize various activation functions commonly used in Artificial Neural Networks (ANNs).

Apparatus Required:

- Python installed (Anaconda/Miniconda or standard Python installation)
- Jupyter Notebook or any Python IDE
- Libraries: matplotlib, numpy

Theory:

Activation functions introduce non-linearity into the output of a neuron, enabling neural networks to learn and represent complex relationships in data. Without activation functions, the network would simply behave like a linear regression model. The choice of activation function impacts the training process and the network's performance.

Common types of activation functions include:

1. **Step Function**: Outputs 0 if the input is less than 0, otherwise outputs 1. Rarely used in deep learning.

   Activation function: f(x) = 1 if x >= 0 else 0
   Range: [0, 1]
   Description: Used for binary decisions. Example: Clap to turn on light.

2. **Sigmoid Function**: Maps input values to the range (0, 1). Suitable for binary classification.

   Activation function: f(x) = 1 / (1 + e^(-x))
   Range: (0, 1)
   Description: Smoothly maps inputs to range between 0 and 1. Good for probabilities.

3. **Hyperbolic Tangent (Tanh) Function**: Maps inputs to the range (-1, 1), offering zero-cantered output.

   Activation function: f(x) = (e^x - e^(-x)) / (e^x + e^(-x))
   Range: (-1, 1)
   Description: Zero-centered, good for balanced data like left/right steering.

4. **ReLU (Rectified Linear Unit)**: Outputs 0 for negative inputs and linear for positive inputs. Common in hidden layers of deep networks.

   Activation function: f(x) = max(0, x)
   Range: [0, ∞)
   Description: Allows only positive values. Used for fast training and performance.

5. **Leaky ReLU**: A variation of ReLU that allows a small, non-zero gradient when the unit is not active.

   Activation function: f(x) = x if x > 0 else 0.01 * x
   Range: (-∞, ∞)
   Description: Allows small negative slope to avoid dead neurons.

6. **Swish Function**: A newer function defined as x * sigmoid(x), introduced by Google, performs better in some deep learning tasks.

   Activation function: f(x) = x * sigmoid(x)
   Range: (-0.28, ∞) approximately
   Description: Smooth and adaptive function developed by Google.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
def step(x):
    return np.where(x >= 0, 1, 0)
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
def tanh(x):
    return np.tanh(x)
def relu(x):
    return np.maximum(0, x)
def leaky_relu(x):
    return np.where(x > 0, x, 0.01 * x)
def swish(x):
    return x * sigmoid(x)
x = np.linspace(-10, 10, 1000)
plt.figure(figsize=(12, 8))
```

```
plt.subplot(2, 3, 1)
plt.plot(x, step(x))
plt.title("Step Function")
plt.grid(True)
plt.subplot(2, 3, 2)
plt.plot(x, sigmoid(x))
plt.title("Sigmoid Function")
plt.grid(True)
plt.subplot(2, 3, 3)
plt.plot(x, tanh(x))
plt.title("Tanh Function")
plt.grid(True)
plt.subplot(2, 3, 4)
plt.plot(x, relu(x))
plt.title("ReLU Function")
plt.grid(True)
plt.subplot(2, 3, 5)
plt.plot(x, leaky_relu(x))
plt.title("Leaky ReLU Function")
plt.grid(True)
plt.subplot(2, 3, 6)
plt.plot(x, swish(x))
plt.title("Swish Function")
plt.grid(True)
plt.tight_layout()
plt.show()
```

Result:

The activation functions Step, Sigmoid, Tanh, ReLU, Leaky ReLU, and Swish were successfully plotted. Each plot visually demonstrates the functional behavior and output range of the corresponding activation function, enhancing understanding of their application in neural networks.

Activation	Real Example	Meaning
Step	Clap to turn on light	Simple binary control
Sigmoid	Voice-controlled speaker	Responds softly to soft voice, strongly to loud
Tanh	Drone steering left or right	Balanced, symmetric response
ReLU	Gas pedal	Only accelerates when pressed (no backward)
Leaky	Faulty switch – responds even to small push	Keeps info from dying for negatives
Swish	Smart light adapting to environment	Smooth & self-adjusting

CODE:

```
import numpy as np
import matplotlib.pyplot as plt

# Activation Functions
def step(x):
    return np.where(x >= 0, 1, 0)

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

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

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

def leaky_relu(x):
    return np.where(x > 0, x, 0.01 * x)

def swish(x):
    return x * sigmoid(x)

# Input Range
```

```
x = np.linspace(-10, 10, 1000)

# Plotting
plt.figure(figsize=(15, 10))

# 1. Step Function: Clap → Light ON
plt.subplot(2, 3, 1)
plt.plot(x, step(x), label="Clap > Threshold")
plt.axvline(x=0, color='r', linestyle='--', label="Clap threshold (0)")
plt.title("Step Function (Clap to Turn ON Light)")
plt.xlabel("Clap Intensity")
plt.ylabel("Light Status")
plt.grid(True)
plt.legend()

# 2. Sigmoid: Whisper → Soft, Shout → Strong
plt.subplot(2, 3, 2)
plt.plot(x, sigmoid(x), label="Voice Amplifier")
plt.axhline(y=0.5, color='r', linestyle='--', label="Half Output")
plt.title("Sigmoid (Voice Sensitivity)")
plt.xlabel("Voice Level")
plt.ylabel("Device Response")
plt.grid(True)
plt.legend()

# 3. Tanh: Steering Left ↔ Right
plt.subplot(2, 3, 3)
plt.plot(x, tanh(x), label="Steering Direction")
plt.axhline(y=0, color='k', linestyle='--')
plt.title("Tanh (Balance Control: Left ↔ Right)")
plt.xlabel("Control Input")
plt.ylabel("Steering Output")
plt.grid(True)
plt.legend()
```

```
# 4. ReLU: Pedal → Acceleration
plt.subplot(2, 3, 4)
plt.plot(x, relu(x), label="Accelerator")
plt.axvline(x=0, color='r', linestyle='--', label="No Press")
plt.title("ReLU (Gas Pedal Control)")
plt.xlabel("Pedal Pressure")
plt.ylabel("Speed")
plt.grid(True)
plt.legend()

# 5. Leaky ReLU: Old Switch → Still Flickers on Tap
plt.subplot(2, 3, 5)
plt.plot(x, leaky_relu(x), label="Leaky Switch")
plt.title("Leaky ReLU (Minor Response for Negative Input)")
plt.xlabel("Input")
plt.ylabel("Response")
plt.grid(True)
plt.legend()

# 6. Swish: Smart Light → Adjusts Smoothly
plt.subplot(2, 3, 6)
plt.plot(x, swish(x), label="Smart Dimmer")
plt.title("Swish (Smooth & Adaptive Response)")
plt.xlabel("Ambient Trigger")
plt.ylabel("Light Level")
plt.grid(True)
plt.legend()

plt.tight_layout()
plt.show()
```