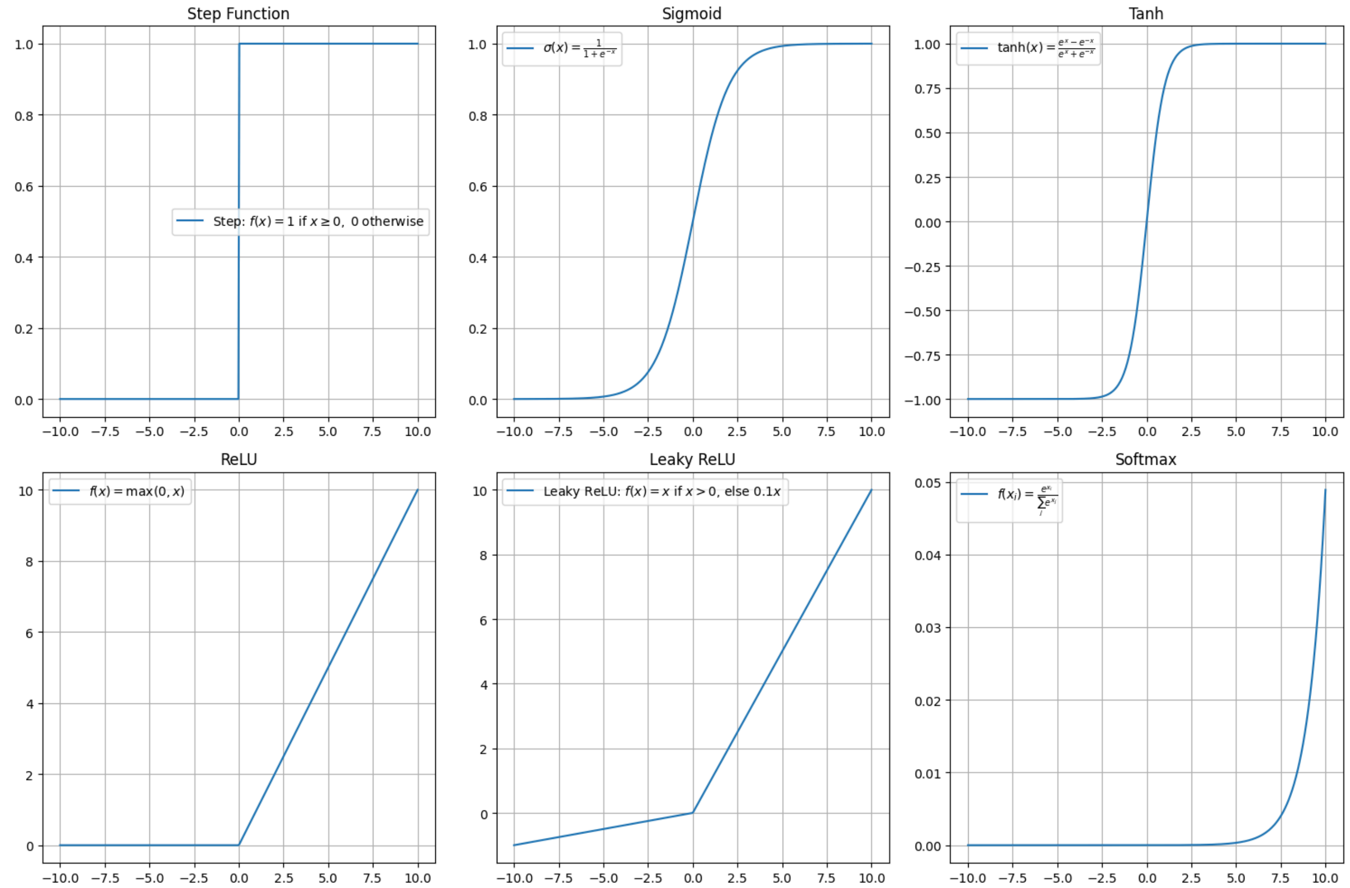
```
# Activation Functions Demo with Formulas & Explanations
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

# Range of x values
x = np.linspace(-10, 10, 400)

# Functions
step = np.where(x >= 0, 1, 0)                              # Step Function
sigmoid = 1 / (1 + np.exp(-x))                             # Sigmoid
tanh = np.tanh(x)                                          # Tanh
relu = np.maximum(0, x)                                    # ReLU
leaky_relu = np.where(x > 0, x, 0.1*x)                     # Leaky ReLU

# Softmax needs 2D input → shape (1, n)
softmax = tf.nn.softmax(x.reshape(1, -1)).numpy().flatten()

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

# Step
plt.subplot(2,3,1)
plt.plot(x, step, label=r'Step: $f(x)=1 \; \text{if } x \geq 0, \; 0 \; \text{otherwise}$')
plt.title("Step Function")
plt.grid(True); plt.legend()

# Sigmoid
plt.subplot(2,3,2)
plt.plot(x, sigmoid, label=r'$\sigma(x)=\frac{1}{1+e^{-x}}$')
plt.title("Sigmoid")
plt.grid(True); plt.legend()

# Tanh
plt.subplot(2,3,3)
plt.plot(x, tanh, label=r'$\tanh(x)=\frac{e^x - e^{-x}}{e^x+e^{-x}}$')
plt.title("Tanh")
plt.grid(True); plt.legend()

# ReLU
plt.subplot(2,3,4)
plt.plot(x, relu, label=r'$f(x)=\max(0,x)$')
plt.title("ReLU")
plt.grid(True); plt.legend()

# Leaky ReLU
plt.subplot(2,3,5)
plt.plot(x, leaky_relu, label=r'Leaky ReLU: $f(x)=x$ if $x>0$, else $0.1x$')
plt.title("Leaky ReLU")
plt.grid(True); plt.legend()

# Softmax
plt.subplot(2,3,6)
plt.plot(x, softmax, label=r'$f(x_i)=\frac{e^{x_i}}{\sum_j e^{x_j}}$')
plt.title("Softmax")
plt.grid(True); plt.legend()

plt.tight_layout()
plt.show()

# Explanations (text output)
print("📘 Activation Functions Explanations:\n")
print("1. Step Function: Outputs 0 or 1. Used in early perceptrons but not differentiable.")
print("2. Sigmoid: Smoothly maps input to (0,1). Good for probabilities but causes vanishing gradients.")
print("3. Tanh: Maps input to (-1,1). Zero-centered but still suffers vanishing gradients.")
print("4. ReLU: Outputs positive values as is, else 0. Very popular, avoids vanishing gradients (mostly).")
print("5. Leaky ReLU: Like ReLU but allows small negative slope. Solves 'dying ReLU' problem.")
print("6. Softmax: Converts vector into probability distribution. Common in output layers for classification.")
```

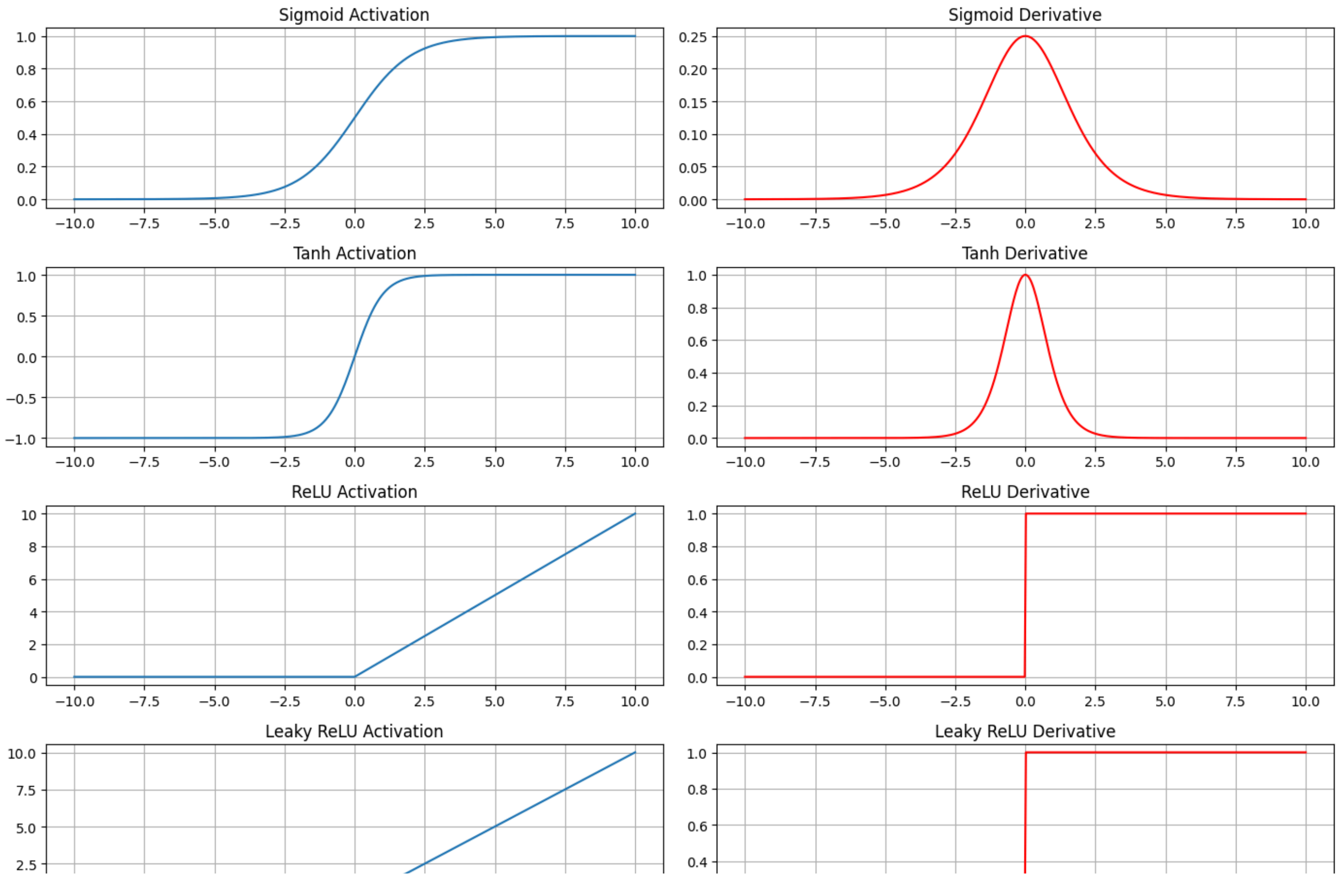


📘 Activation Functions Explanations:

1. Step Function: Outputs 0 or 1. Used in early perceptrons but not differentiable.
2. Sigmoid: Smoothly maps input to (0,1). Good for probabilities but causes vanishing gradients.
3. Tanh: Maps input to (-1,1). Zero-centered but still suffers vanishing gradients.
4. ReLU: Outputs positive values as is, else 0. Very popular, avoids vanishing gradients (mostly).
5. Leaky ReLU: Like ReLU but allows small negative slope. Solves 'dying ReLU' problem.
6. Softmax: Converts vector into probability distribution. Common in output layers for classification.

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

# Input range
x = np.linspace(-10, 10, 500, dtype=np.float32)

# Activation functions
sigmoid = tf.nn.sigmoid(x).numpy()
tanh = tf.nn.tanh(x).numpy()
relu = tf.nn.relu(x).numpy()
leaky_relu = tf.nn.leaky_relu(x, alpha=0.1).numpy()

# Derivatives
# Sigmoid derivative: σ(x)*(1 - σ(x))
sigmoid_deriv = sigmoid * (1 - sigmoid)

# Tanh derivative: 1 - tanh(x)^2
tanh_deriv = 1 - np.power(tanh, 2)

# ReLU derivative: 0 if x<0 else 1
relu_deriv = np.where(x > 0, 1, 0)

# Leaky ReLU derivative: alpha if x<0 else 1
leaky_relu_deriv = np.where(x > 0, 1, 0.1)

# Plot
plt.figure(figsize=(14,10))
```

```
# Sigmoid
plt.subplot(4,2,1)
plt.plot(x, sigmoid, label="Sigmoid")
plt.title("Sigmoid Activation")
plt.grid(True)

plt.subplot(4,2,2)
plt.plot(x, sigmoid_deriv, label="Sigmoid'", color="red")
plt.title("Sigmoid Derivative")
plt.grid(True)

# Tanh
plt.subplot(4,2,3)
plt.plot(x, tanh, label="Tanh")
plt.title("Tanh Activation")
plt.grid(True)

plt.subplot(4,2,4)
plt.plot(x, tanh_deriv, label="Tanh'", color="red")
plt.title("Tanh Derivative")
plt.grid(True)

# ReLU
plt.subplot(4,2,5)
plt.plot(x, relu, label="ReLU")
plt.title("ReLU Activation")
plt.grid(True)

plt.subplot(4,2,6)
plt.plot(x, relu_deriv, label="ReLU'", color="red")
plt.title("ReLU Derivative")
plt.grid(True)

# Leaky ReLU
plt.subplot(4,2,7)
plt.plot(x, leaky_relu, label="Leaky ReLU")
plt.title("Leaky ReLU Activation")
plt.grid(True)

plt.subplot(4,2,8)
plt.plot(x, leaky_relu_deriv, label="Leaky ReLU'", color="red")
plt.title("Leaky ReLU Derivative")
plt.grid(True)

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


Sigmoid Activation
Sigmoid Derivative
Tanh Activation
Tanh Derivative
ReLU Activation
ReLU Derivative
Leaky ReLU Activation
Leaky ReLU Derivative