Model Type and Activation Function (Handout)

Sireesha Puppala

04-11-2024

# Introduction

In machine learning, the type of task or model architecture often determines which activation function to use. Activation functions introduce non-linearity, enabling neural networks to learn complex patterns. Different tasks (e.g., binary classification, multiclass classification, regression) and model types (e.g., convolutional networks, recurrent networks) benefit from specific activation functions. Here is an overview of common ML tasks, activation functions typically used, and why they’re suited for each:

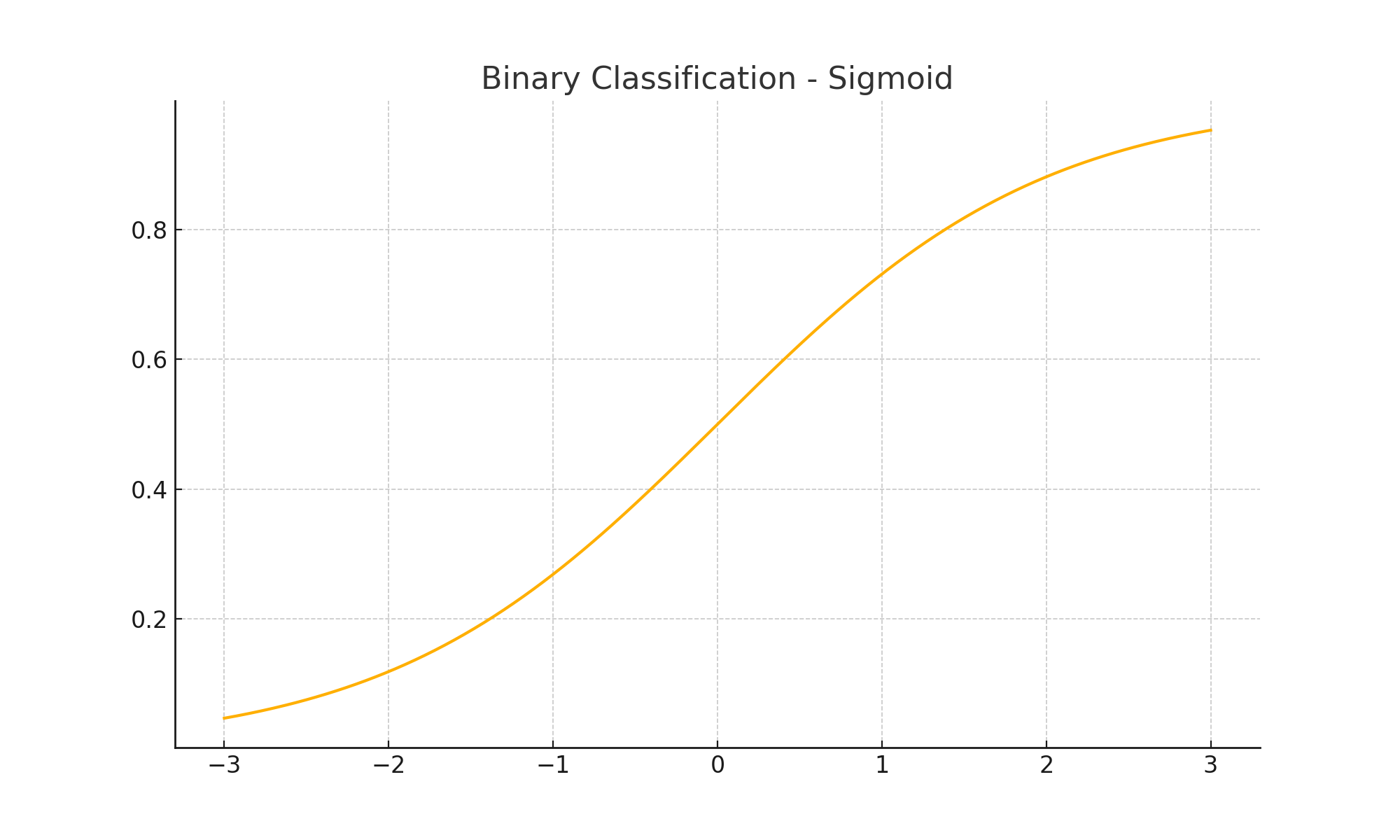
# 1. Binary Classification

**Common Activation** Function: Sigmoid

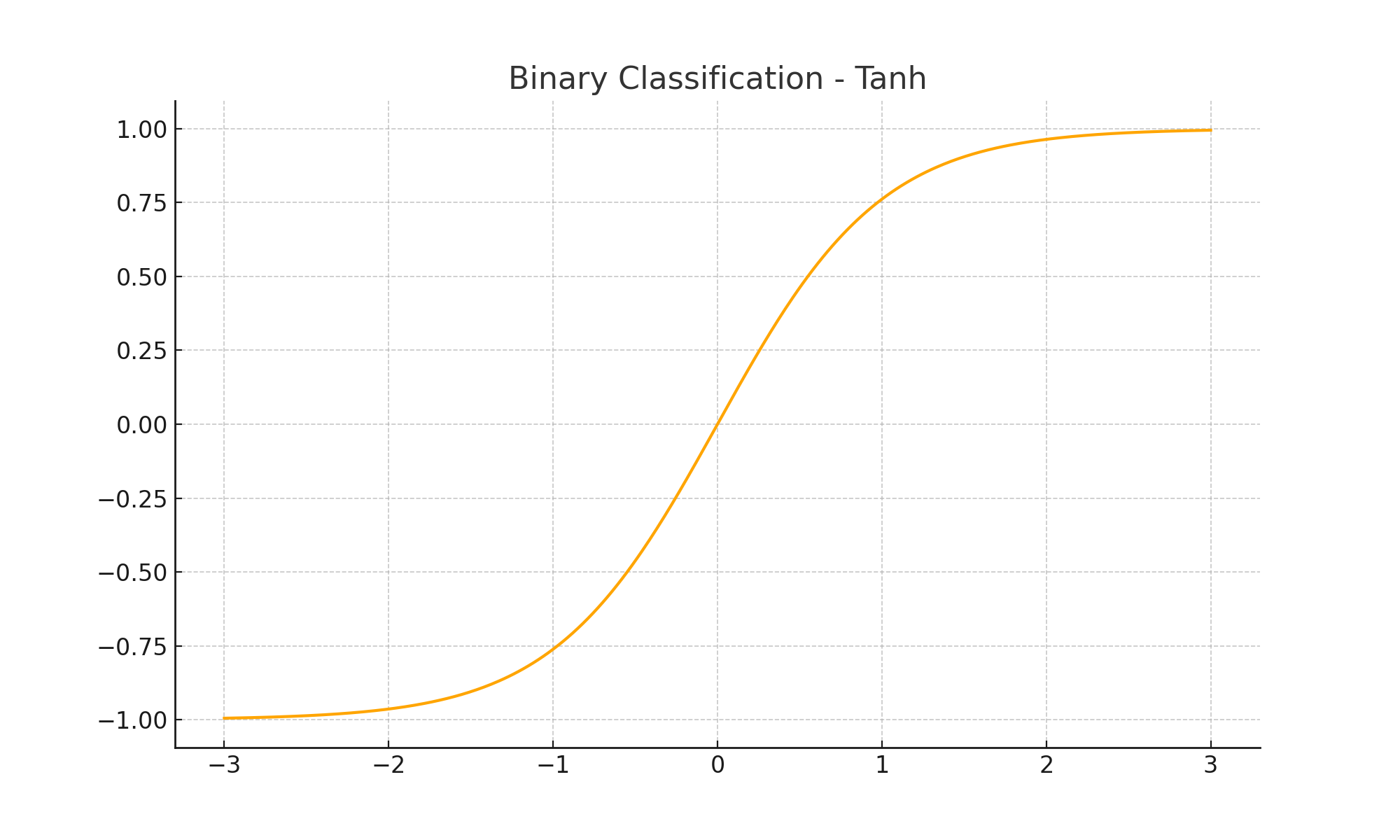
**Why**: The sigmoid function maps outputs to the range (0, 1), which is ideal for binary classification since it can be interpreted as a probability for two classes.

**Alternative**: Tanh (sometimes used but less common due to potential vanishing gradient issues)

## Binary Classification - Sigmoid



## Binary Classification - Tanh



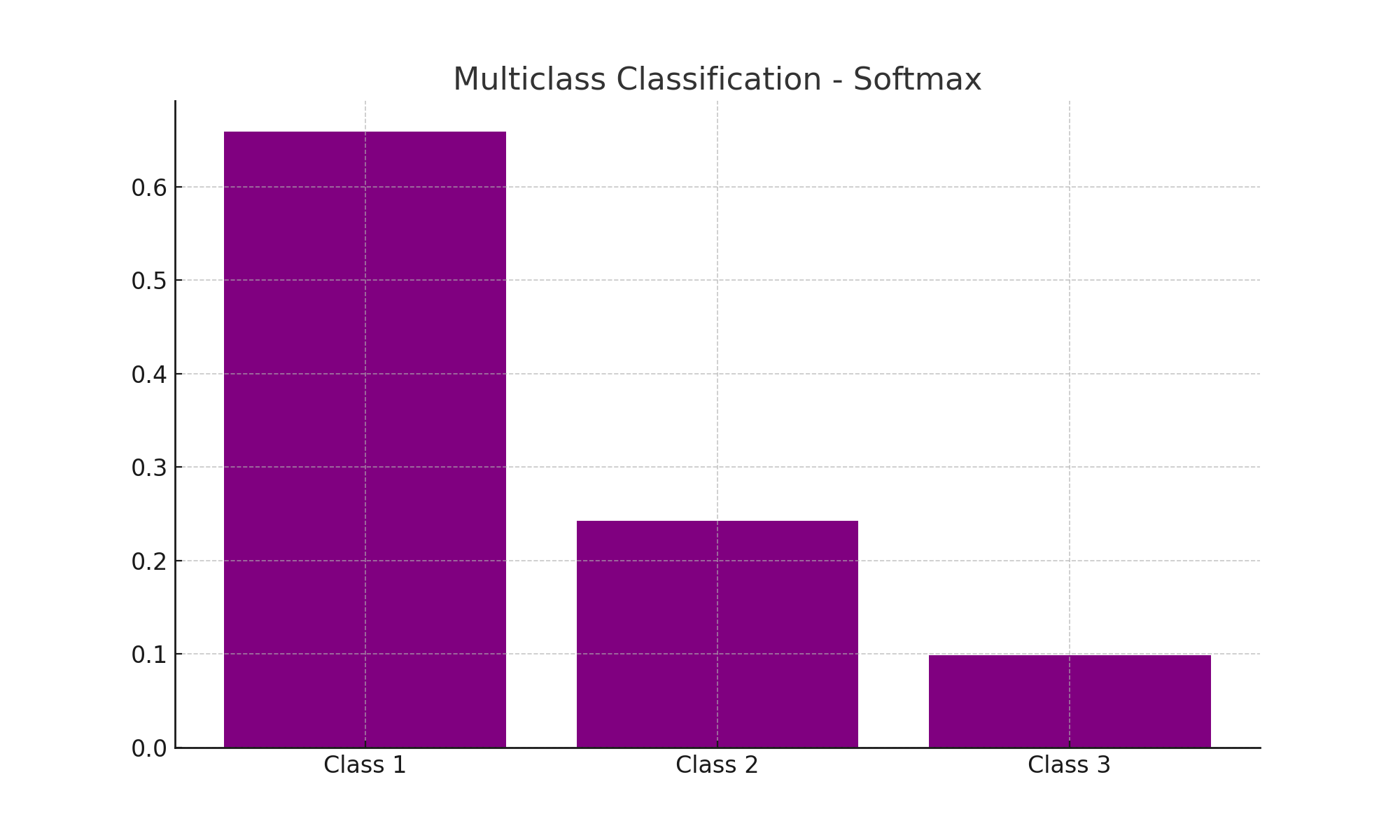
# 2. Multiclass Classification

**Common Activation Function**: Softmax

**Why**: Softmax is typically used in the output layer for multiclass classification. It converts raw model outputs into probabilities for each class by ensuring that all class probabilities sum to 1.

**Alternative:** Sometimes Sigmoid can be used in multi-label classification (when multiple classes can be true simultaneously), applying it independently to each output unit.

## Multiclass Classification - Softmax



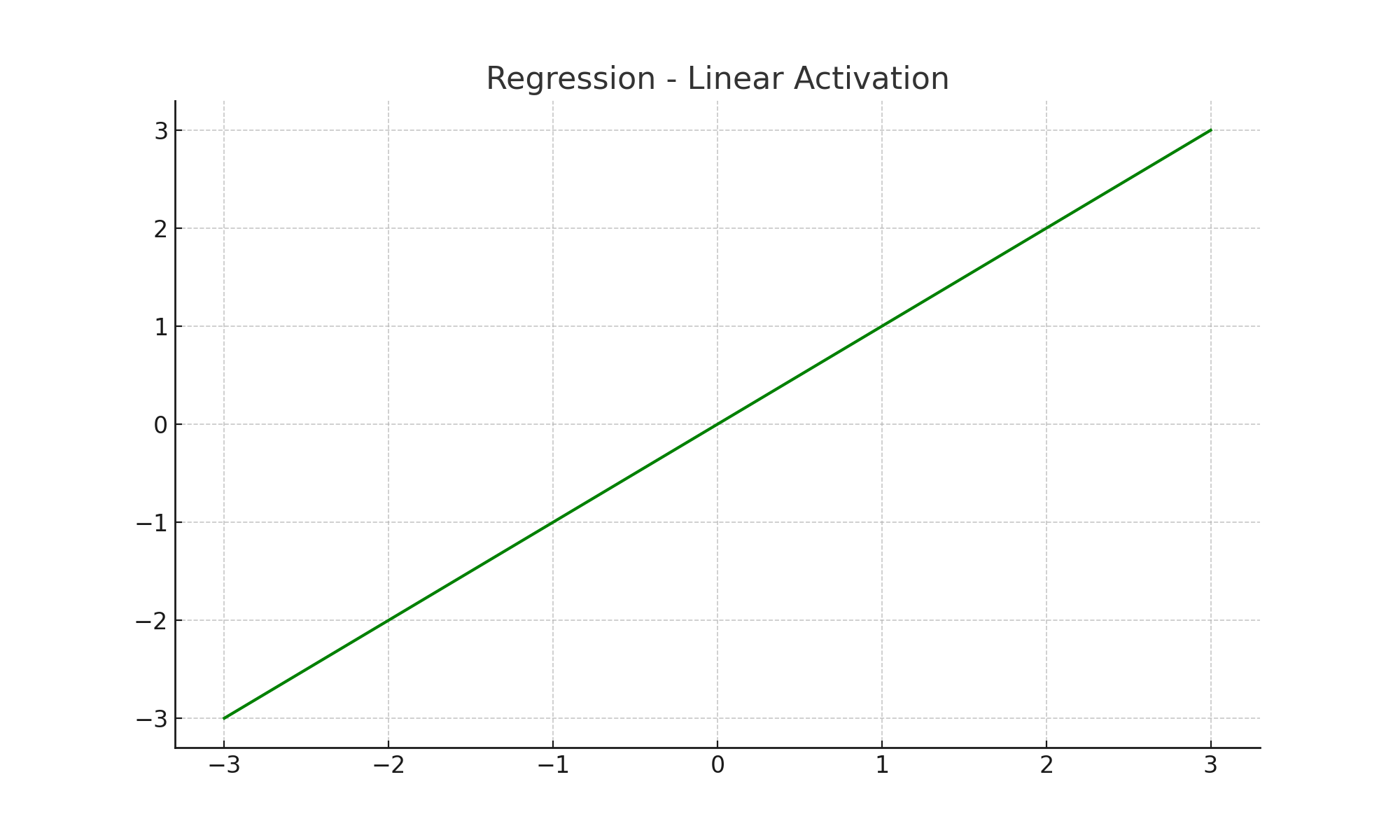
# 3. Regression

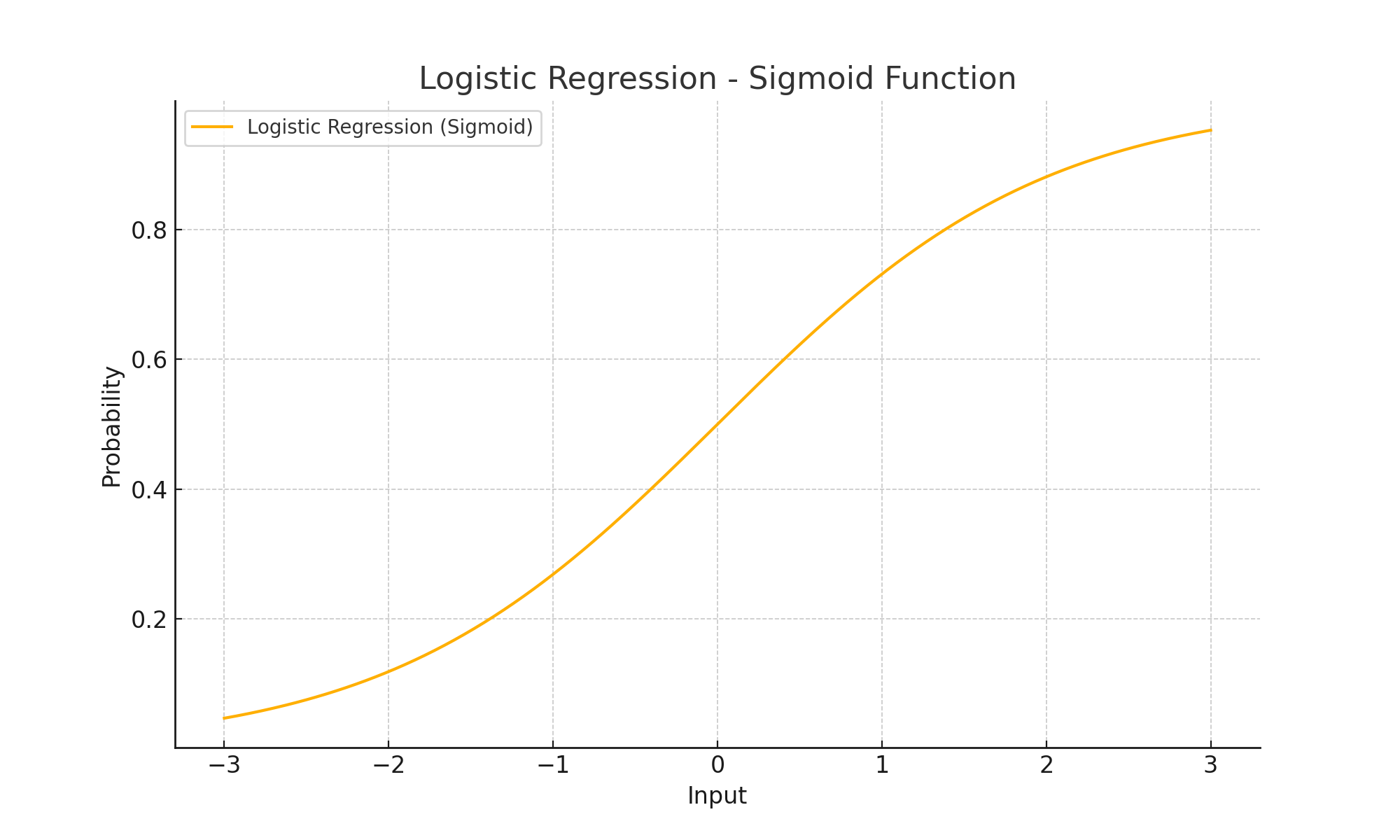
**Common Activation Function**: Linear or Logistic

**Why**: For regression tasks, the output is a continuous value, so the activation function is usually linear (identity function) to produce real-number outputs without any constraints. Logistic regression utilizes the Sigmoid function to map input values to a probability range between 0 and 1, enabling binary classification. It is widely used to predict the likelihood of a class based on input features.

**Alternative**: If the output is expected to fall within a specific range, ReLU (e.g., if outputs are positive-only) or Tanh (e.g., if outputs are between -1 and 1) can sometimes be used.

## Regression - Linear Activation



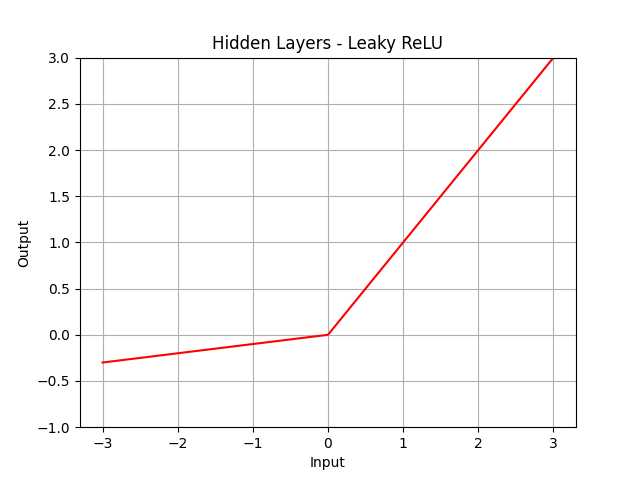


4. Hidden Layers in Feedforward Networks

**Common Activation Functions**: ReLU and Leaky ReLU

**Why**: ReLU is widely used because it’s computationally efficient and helps with faster training by reducing vanishing gradient issues. Leaky ReLU (or variants like Parametric ReLU) is used when small negative gradients are desirable to prevent neurons from "dying" during training.

# 

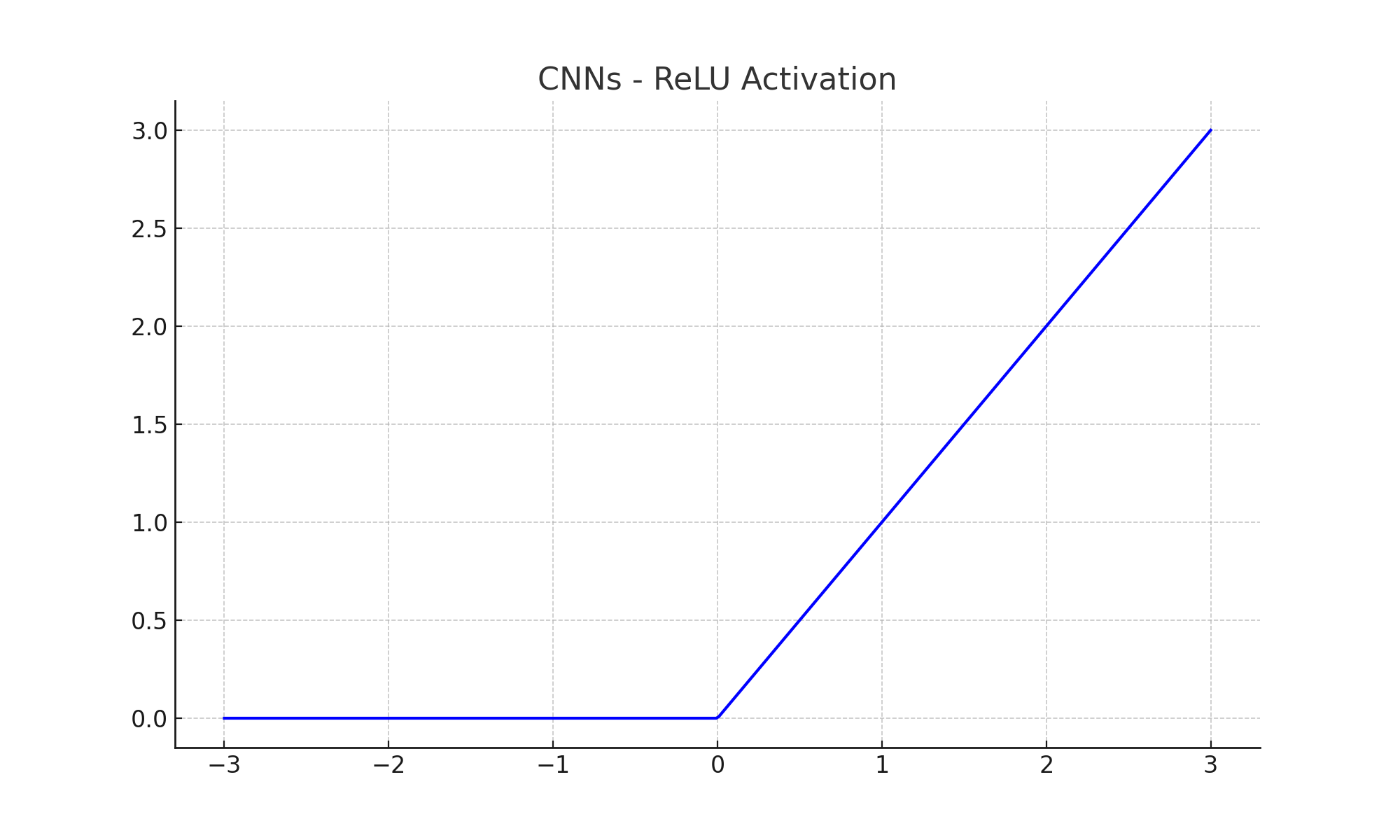


# 5. Convolutional Neural Networks (CNNs)

**Common Activation Function**: ReLU

**Why**: CNNs often use ReLU after convolutional layers due to its simplicity and efficiency, and because it works well with the spatial hierarchical structure of CNNs.

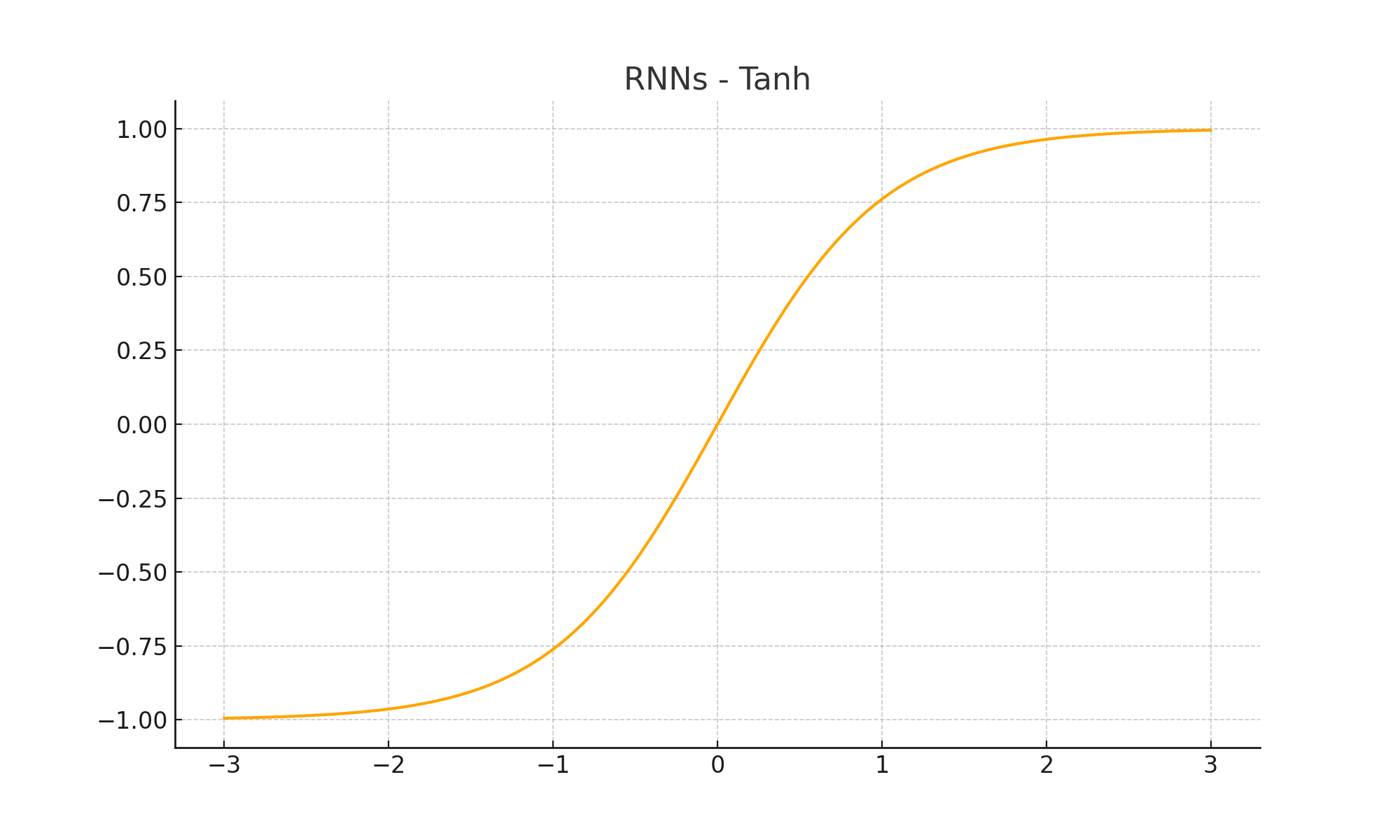
**Alternative**: Leaky ReLU or ELU (Exponential Linear Unit) can sometimes be used to address ReLU’s zero-gradient issue for negative values.

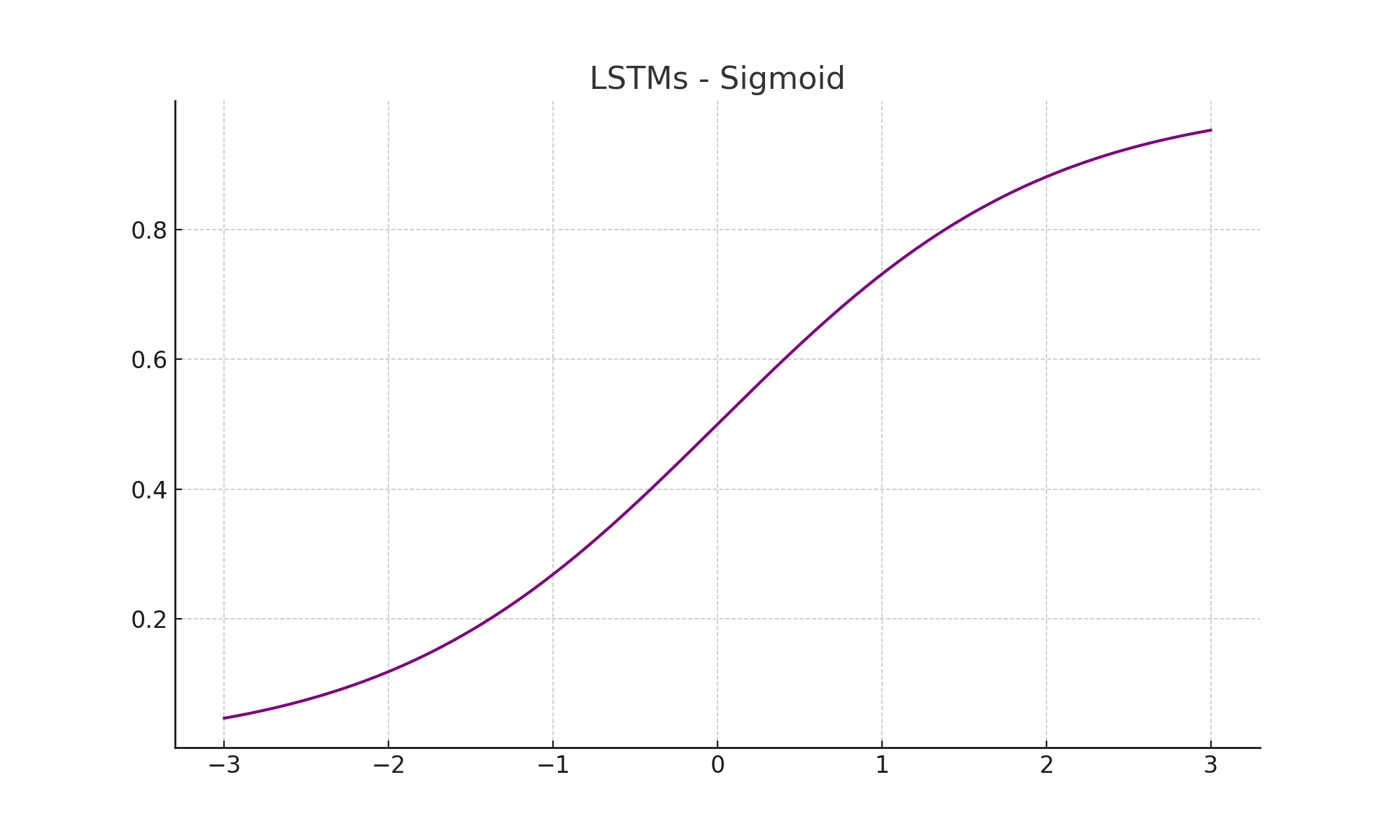


# 6. Recurrent Neural Networks (RNNs) and Long Short-Term Memory Networks (LSTMs)

**Common Activation Function**: Tanh and Sigmoid

**Why**: RNNs often use Tanh or Sigmoid to control the flow of information over time. LSTM units use Sigmoid for gating mechanisms (input, forget, and output gates) and Tanh for cell state updates, helping to manage the temporal dependencies in sequences.





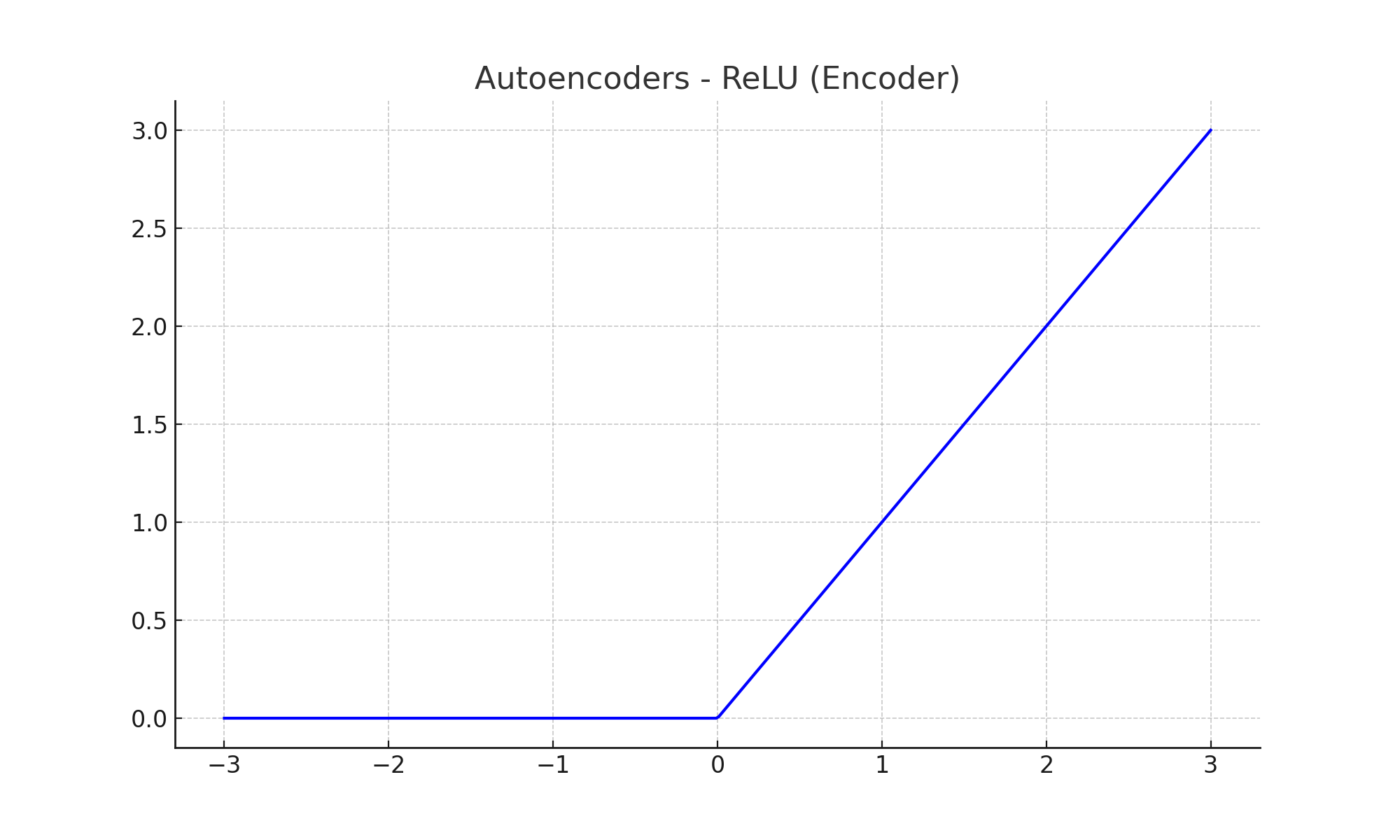
# 7. Autoencoders

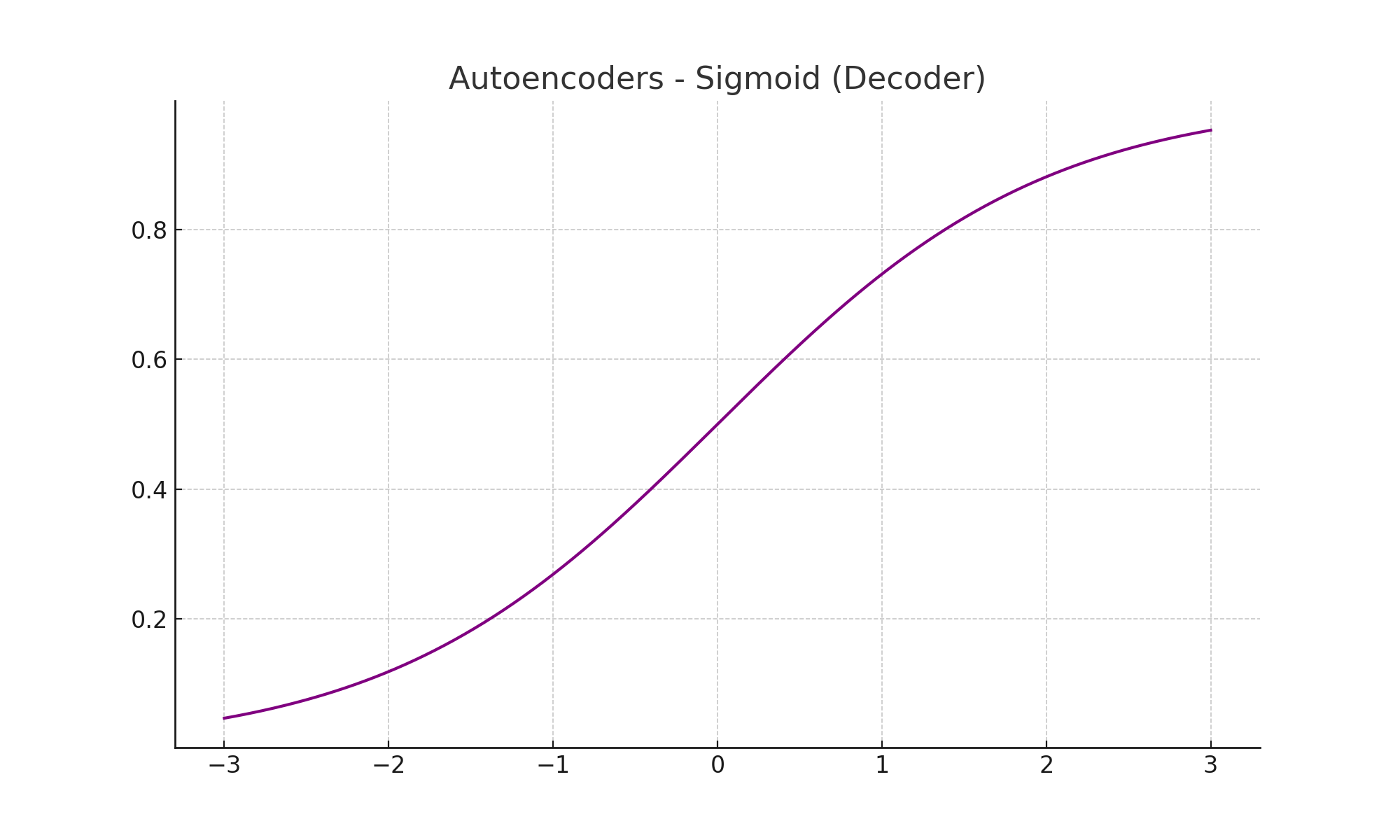
**Common Activation Functions**:

Encoder Layers: ReLU or Tanh

Decoder Layers: Sigmoid or Tanh

**Why**: ReLU is often used in encoding layers to learn a compact representation. The choice of activation in the decoder depends on the desired output range; Sigmoid works well for binary data, while Tanh can be useful for outputs within a specific range.





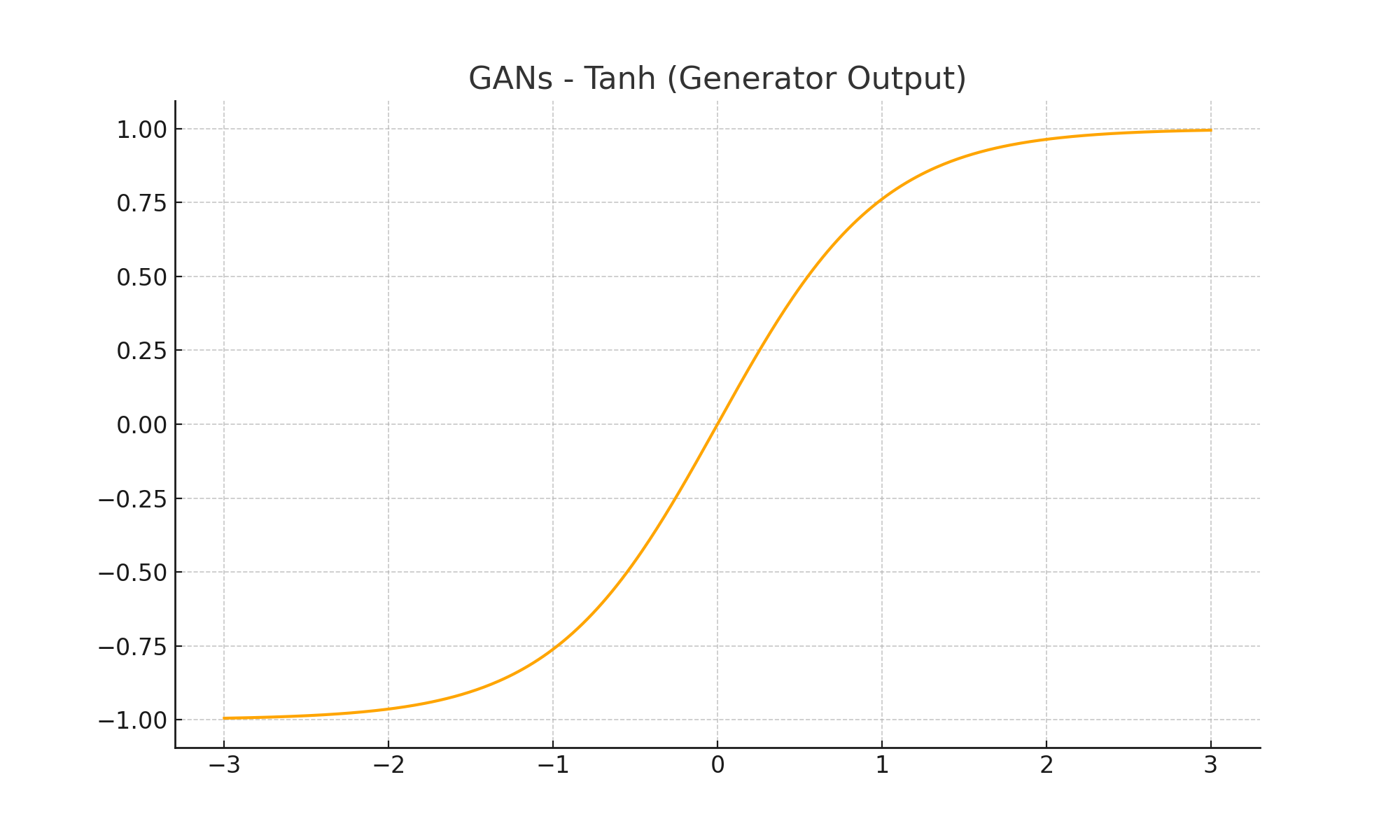
# 8. Generative Adversarial Networks (GANs)

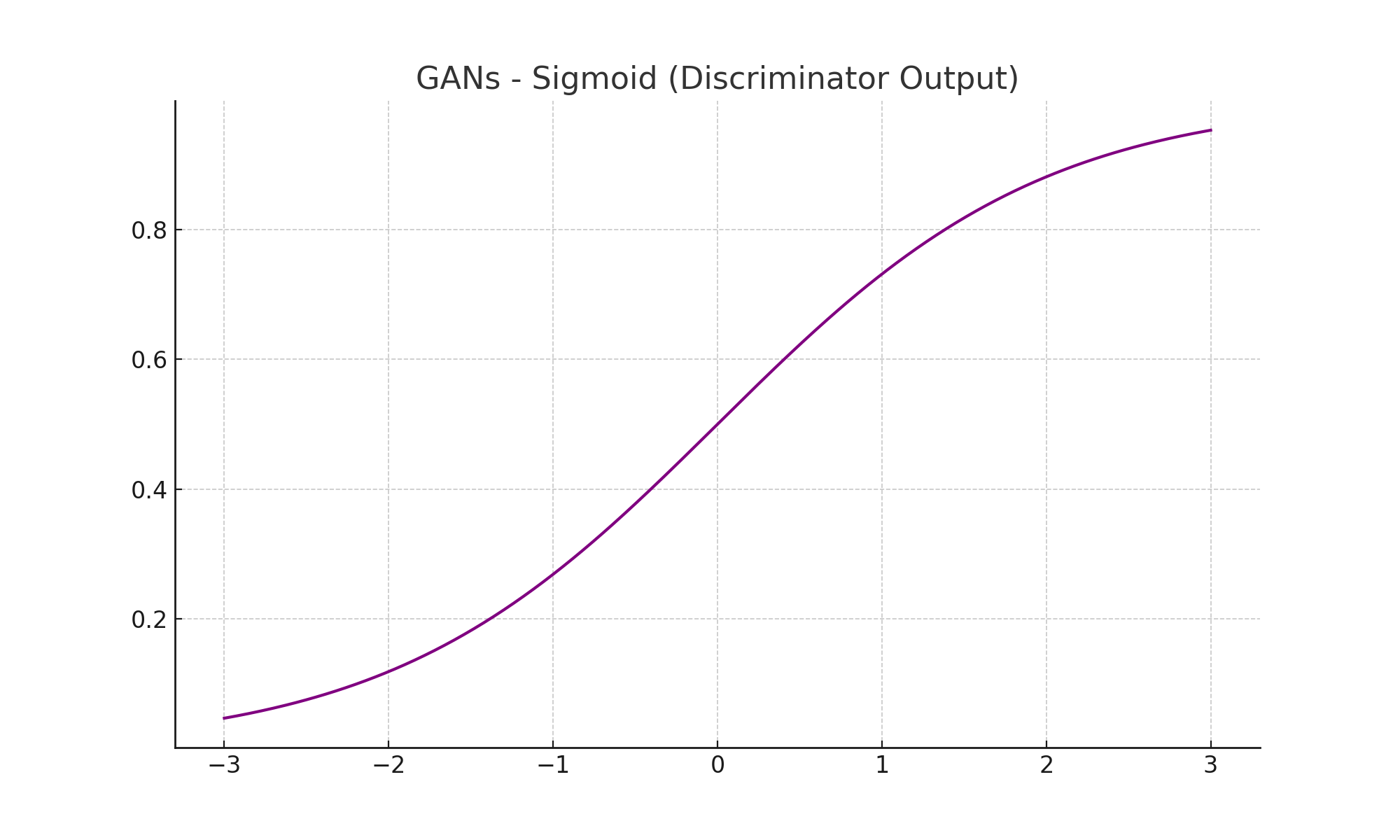
**Common Activation Functions**:

Generator Network: ReLU for hidden layers, Tanh for output

Discriminator Network: Leaky ReLU for hidden layers, Sigmoid for output

**Why**: GAN generators commonly use Tanh in the output layer to generate data within a specified range (e.g., -1 to 1 for normalized image data). The discriminator uses Sigmoid to classify data as real or fake and Leaky ReLU for hidden layers to avoid dead neurons.

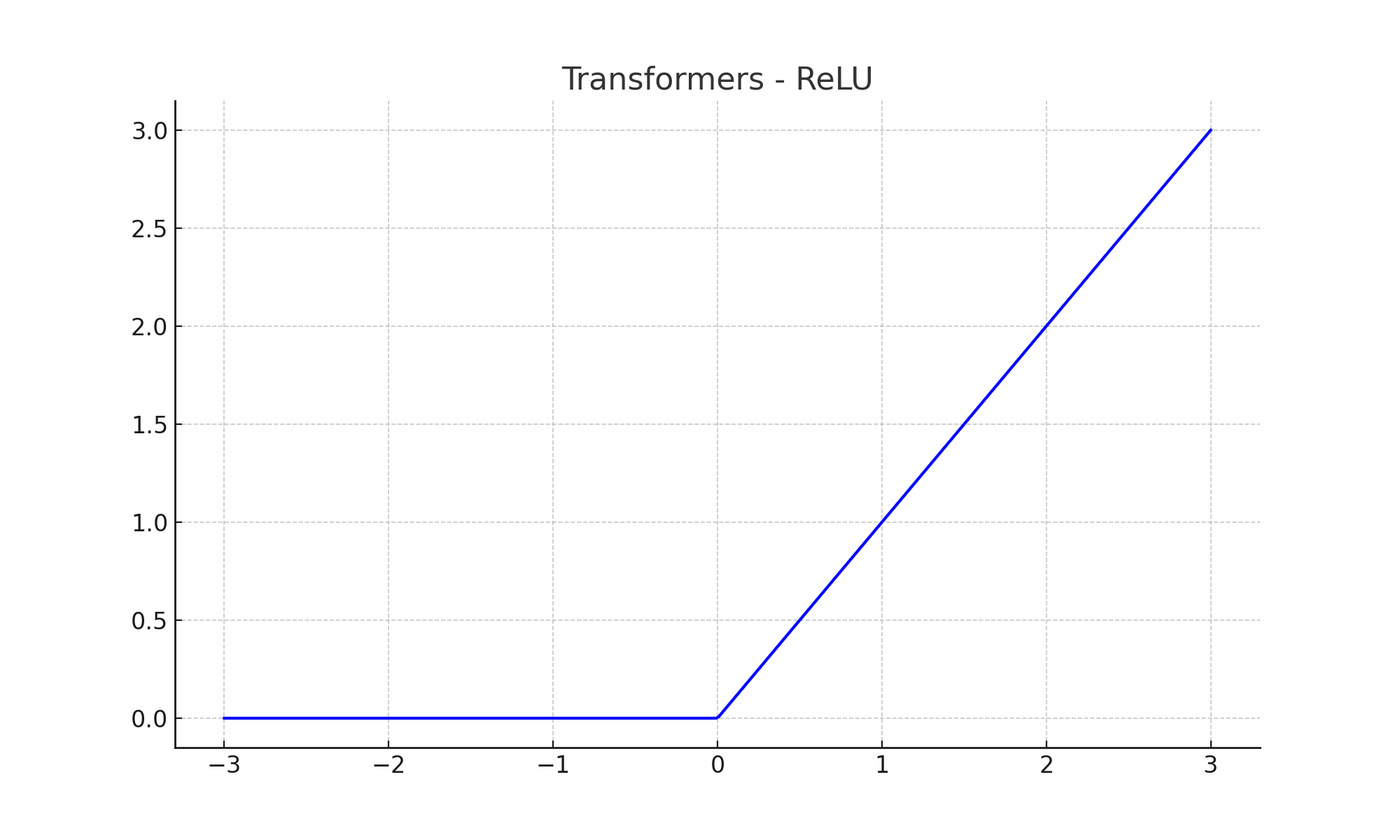


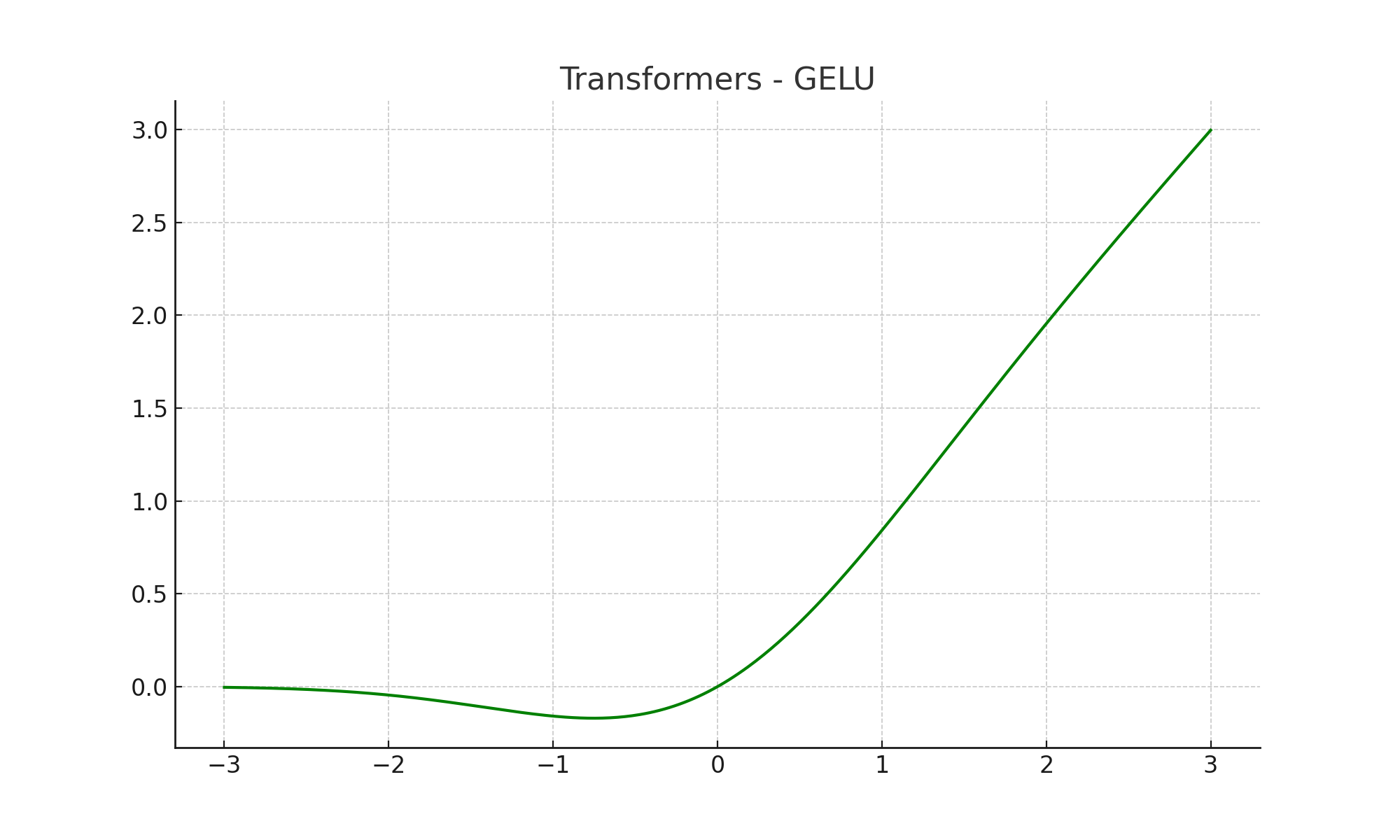


# 9. Transformers and Attention Mechanisms

**Common Activation Function**: ReLU or GELU (Gaussian Error Linear Unit)

**Why**: ReLU is often used, but GELU is preferred in Transformer models (like BERT) because it improves gradient flow, especially in deep architectures with attention mechanisms.

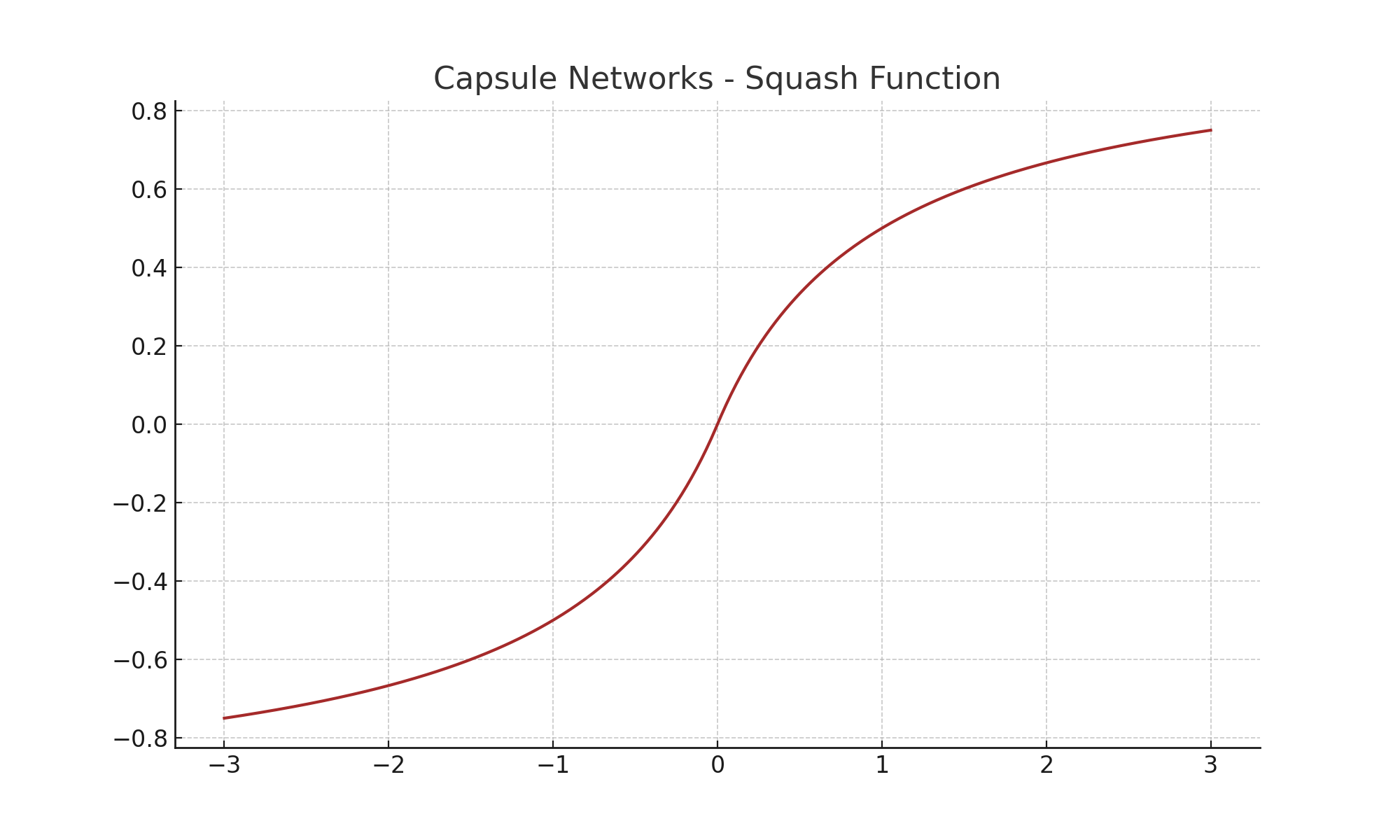




# 10. Capsule Networks

**Common Activation Function**: Squash Function

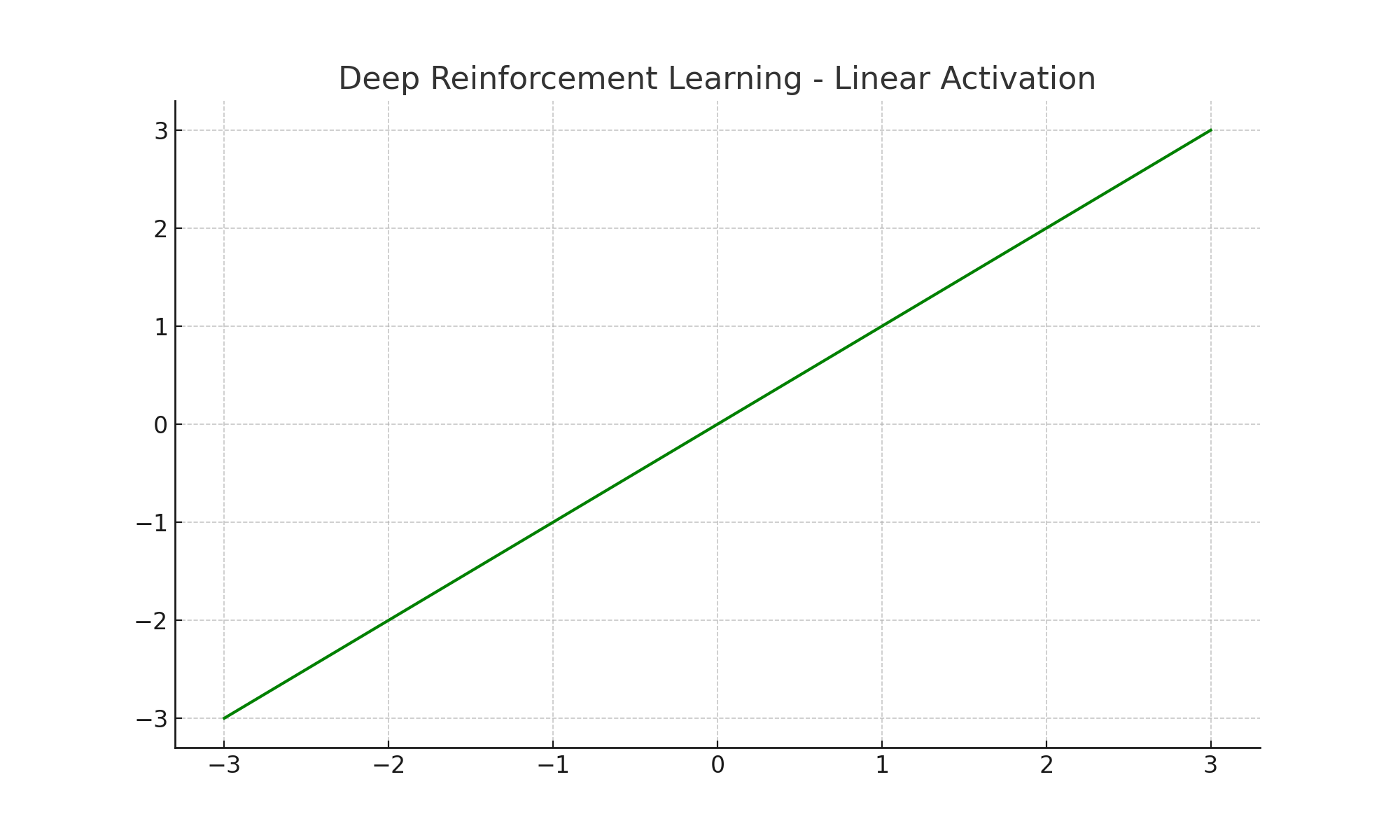
**Why**: The Squash function normalizes the output to lie between 0 and 1 while preserving vector orientation, which is essential for capsule networks that represent features as vectors rather than scalars.

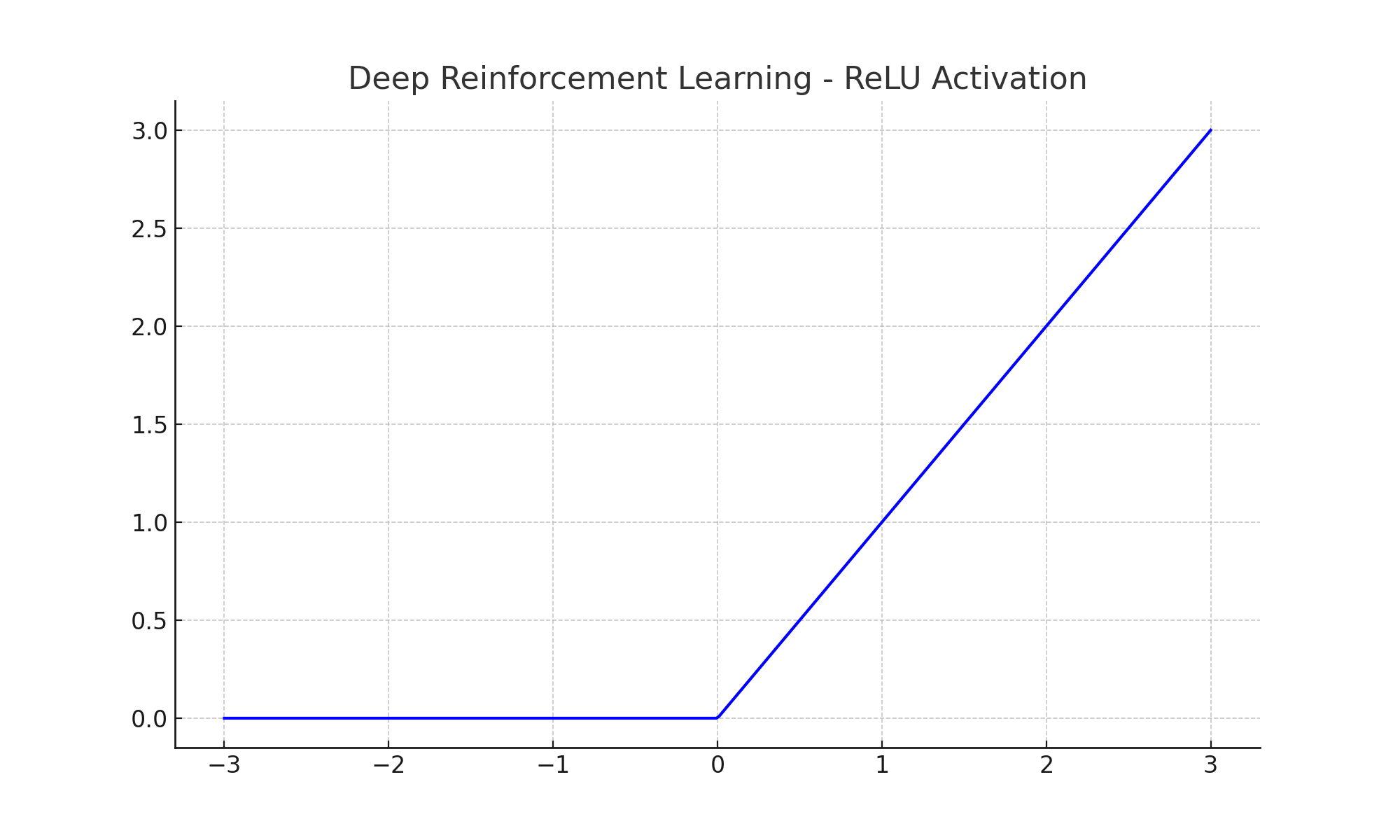


# 11. Deep Reinforcement Learning (e.g., Deep Q-Networks)

**Common Activation Functions**: ReLU in hidden layers, Linear for output (Q-value prediction)

**Why**: ReLU is commonly used for hidden layers in reinforcement learning due to its computational efficiency, while the output layer might use a linear activation if the output is a continuous value like Q-values.





# Summary Table of Tasks and Activation Functions

|  |  |  |
| --- | --- | --- |
| Task | Typical Activation Function (Output Layer) | Hidden Layer Activations |
| Binary Classification | Sigmoid | ReLU |
| Multiclass Classification | Softmax | Sigmoid |
| Regression | Linear or Logistic | ReLU, Tanh (Linear)  Sigmoid (Logistic) |
| CNNs | ReLU (all layers) | ReLU, Leaky ReLU |
| RNNs / LSTMs | Tanh, Sigmoid | Tanh, ReLU (hidden layers) |
| Autoencoders | Sigmoid (decoder) | ReLU (encoder), Tanh (decoder) |
| GANs | Tanh (generator), Sigmoid (discriminator) | ReLU, Leaky ReLU |
| Transformers | ReLU, GELU | ReLU, GELU |
| Capsule Networks | Squash | Custom activation |
| Deep Reinforcement Learning | Linear (output) | ReLU, Tanh |

This variety ensures flexibility in training networks for a wide range of applications. Each function has advantages based on the task requirements, data characteristics, and network architecture.