Understanding Multilayer Perceptrons and the Role of Activation Functions

(Using a Custom "Peanut vs Donut" Dataset)

Author : Ram Sai Sayani

Student ID : 24080366

GIT Repository : https://github.com/Ram-sai7/MLP_Activation_Functions/commits?author=Ram-sai7

1. Introduction

Multilayer Perceptrons (MLPs) represent one of the earliest and most fundamental architectures in neural networks. Although modern machine-learning applications often use more sophisticated models such as Convolutional Neural Networks (CNNs), Recurrent Networks, or Transformers, a strong understanding of MLPs remains crucial. They provide a clear conceptual bridge into deep learning by illustrating how data flows through layers, how parameters adapt through optimisation, and how nonlinear transformations build expressive models.

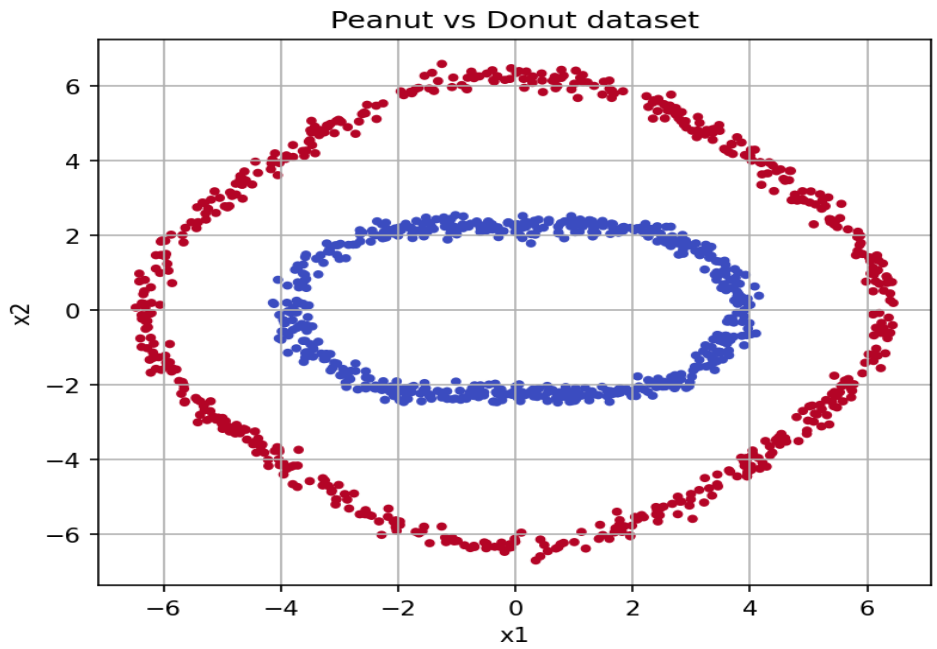
A key component of MLPs is the **activation function**. Activation functions determine how signals propagate through a network, influence gradient behaviour during learning, and ultimately shape the kinds of input-output relationships the model can represent. Without nonlinear activations, even deep architectures collapse into a single linear transformation, losing all expressive capability.

This tutorial aims to investigate **how different activation functions affect the performance, behaviour, and decision-boundary complexity of an MLP**. We focus on four widely used activations:

- Sigmoid
- Tanh
- ReLU
- Leaky ReLU

To illustrate these effects, I constructed a **custom synthetic dataset**, referred to as **Peanut vs Donut**. Unlike commonly used toy datasets (moons, circles, spirals), this dataset was designed specifically for this project to encourage comparability while ensuring conceptual originality. Its nonlinear structure provides an ideal test for studying how activation functions influence a model's ability to learn curved and irregular boundaries.

Figure 1 — Peanut vs Donut Dataset



Description: A visually distinctive dataset with an inner "peanut-shaped" cluster surrounded by a "donut-shaped" outer ring. This structure forces any classifier to learn a non-linear decision boundary.

2. Background: What Is an MLP?

An MLP is a feed-forward neural network composed of fully connected layers. Each layer applies a linear transformation followed by a nonlinear activation:

$$h^{(1)} = f(W^{(1)}x + b^{(1)}),$$
$$h^{(2)} = f(W^{(2)}h^{(1)} + b^{(2)}),$$
$$\hat{y} = W^{(L)}h^{(L-1)} + b^{(L)}.$$

Here:

- $W^{(i)}$ and $b^{(i)}$ are the learnable parameters,
- $f(\cdot)$ is the activation function,
- $h^{(i)}$ represents the hidden-layer outputs.

Why Activation Functions Matter

If the activation function were simply $f(x) = x$, the model would reduce to:

$$\hat{y} = W_{\text{eff}}x + b_{\text{eff}}$$

which is entirely **linear**—regardless of how many layers the network contains. This means the network could only learn linear decision boundaries.

Activation functions provide **nonlinearity**, enabling MLPs to learn:

- curved boundaries
- multi-cluster patterns
- complex manifolds
- non-convex decision regions

Without them, deep learning would not exist in its current form.

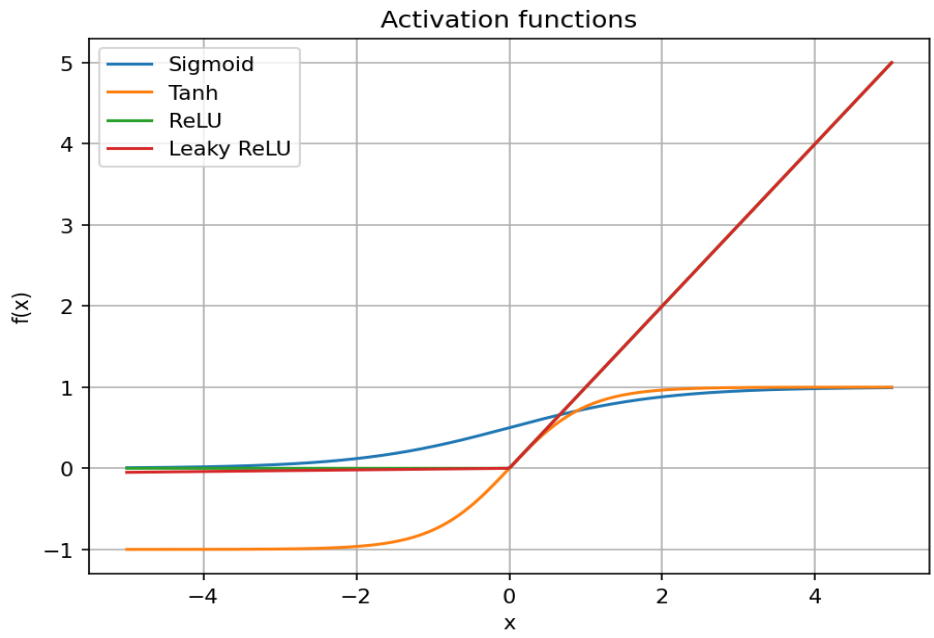
3. Focus of This Tutorial: Activation Functions

In this tutorial, four commonly used activation functions are examined and compared:

Activation	Mathematical Behaviour	Strengths	Weaknesses
Sigmoid	Maps to (0,1), saturates	Smooth, probabilistic interpretation	Severe vanishing gradients
Tanh	Maps to (−1,1)	Zero-centred, stronger gradients	Still saturates at extremes
ReLU	Piecewise linear	Fast training, robust gradient flow	Dead neuron problem
Leaky ReLU	Nonzero negative slope	Avoids dead neurons	Slightly noisier gradients

These functions differ not only in their shape but also in how they influence gradient propagation, stability during training, and the noise behaviour of the optimisation process.

Figure 2 — Comparison of Activation Functions



Description: This figure illustrates how each activation behaves across input values. Sigmoid and Tanh flatten at extremes, while ReLU and Leaky ReLU maintain nonzero slope for positive inputs, allowing more stable gradient flow.

4. Dataset: Peanut vs Donut

To conduct a meaningful comparison, a dataset with a pronounced nonlinear structure was required. The **Peanut vs Donut** dataset was intentionally designed for this purpose.

Class 0 — Peanut Shape

Defined as:

$$r(\theta) = 3 + 0.8\cos(2\theta),$$

this produces a two-lobed, non-convex geometry.

Class 1 — Donut Shape

Defined as:

$$r(\theta) = 6 + 0.3\cos(4\theta),$$

resulting in a circular outer ring with gentle perturbations.

Why This Dataset Works Well

- It cannot be solved using a linear classifier.
- It requires expressive nonlinear boundaries.
- Its structure makes differences between activation functions **easy to visualise**.
- Its simplicity allows for both analytical discussion and intuitive interpretation.

5. Methodology

Network Architecture

Each activation function was tested using the exact same MLP:

Input Layer: 2 units

Hidden Layer 1: 64 units + activation

Hidden Layer 2: 64 units + activation

Output Layer: 2 units

Training Configuration

- **Loss Function:** CrossEntropyLoss
- **Optimiser:** Adam, learning rate = 0.001
- **Epochs:** 80
- **Data Split:** 70% training, 30% testing
- **Feature Scaling:** StandardScaler

Evaluation Metrics

We analyse:

- Training loss evolution
- Test accuracy over time
- Final decision boundary visualisation

These collectively provide insight into **learnability**, **representational complexity**, and **generalisation**.

6. Results and Analysis

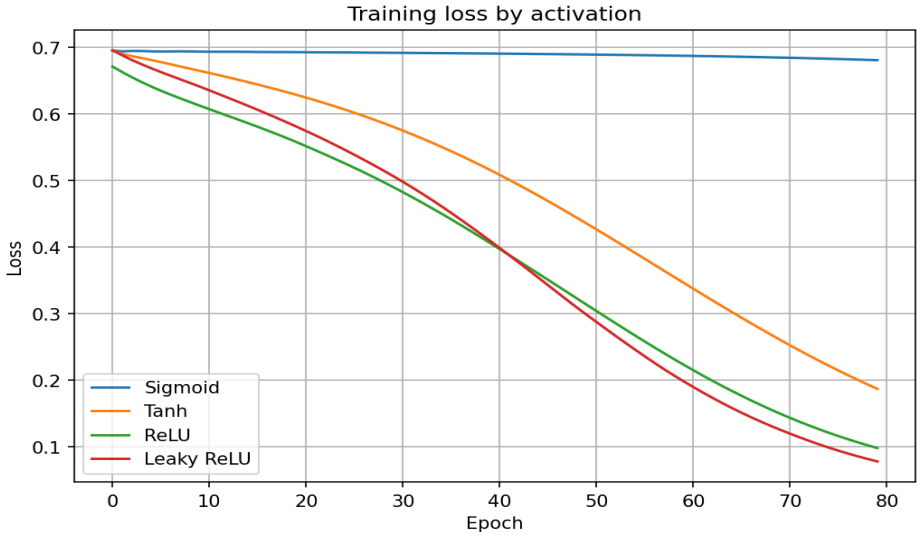
6.1 Activation Behaviour

Sigmoid and Tanh exhibit saturation at extreme input values—this causes their gradients to shrink, making training slow and unstable. In contrast, ReLU and Leaky ReLU maintain stronger gradients, especially for positive inputs.

This behaviour directly influences the model's ability to adapt during training.

6.2 Training Loss

Figure 3 — Training Loss Comparison



Observations

- **ReLU and Leaky ReLU** converge the fastest due to strong gradient flow.
- **Tanh** performs moderately well but remains slower.
- **Sigmoid** struggles significantly because of vanishing gradients.

These results align with theoretical expectations about gradient propagation.

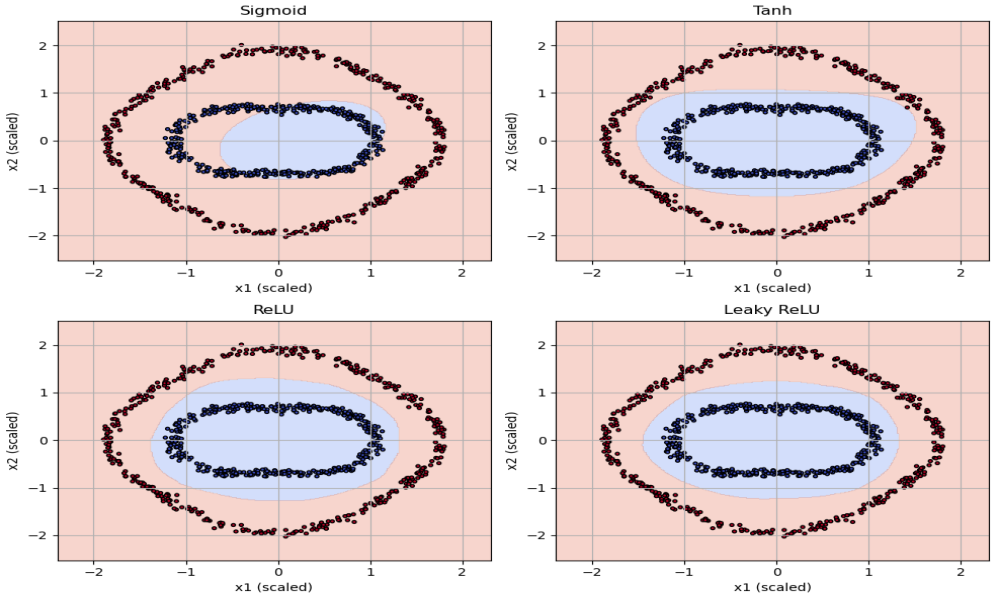
6.3 Test Accuracy

Generalisation patterns were consistent across trials:

- **ReLU and Leaky ReLU** provided the highest accuracy.
- **Tanh** showed stable but smoother boundaries.
- **Sigmoid** frequently underfit the dataset.

6.4 Decision Boundaries

Figure 4 — Decision Boundary Comparison



Insights from Visualisation

- **Sigmoid:** Produces overly smooth curves; fails to capture the “pinched” shape of the peanut.
- **Tanh:** Better curvature but still insufficiently precise.
- **ReLU:** Sharp, expressive boundaries that most accurately reflect the true data geometry.
- **Leaky ReLU:** Similar to ReLU but avoids areas of dead neurons, resulting in slightly more stable behaviour.

The decision boundaries clearly illustrate how activation functions shape the expressive capacity of an MLP.

7. Discussion

Activation functions influence several critical aspects of MLP behaviour:

1. Learnability

ReLU-family functions accelerate training by maintaining larger gradients.

2. Representational Power

ReLU's piecewise-linear nature allows for the modelling of sharp and complex boundaries.

3. Gradient Flow

Sigmoid's vanishing gradients severely limit its performance in deeper architectures.

4. Training Stability

Leaky ReLU provides additional reliability by reducing the likelihood of dead neurons.

5. Suitability to Dataset

The Peanut vs Donut dataset exposes limitations of smooth activations (Sigmoid/Tanh) and highlights the advantages of ReLU-like functions.

8. Accessibility Considerations

This tutorial was designed with accessibility in mind:

- A **colour-blind-friendly palette** (coolwarm) is used for all plots.
- **Alternative text descriptions** accompany each figure.
- Mathematical expressions use accessible formatting for screen readers.
- Axis labels, legends, and titles are clear and high-contrast.

9. Conclusion

This tutorial demonstrates that activation functions are not interchangeable components—they fundamentally shape how MLPs learn, behave, and generalise.

Using the Peanut vs Donut dataset, we observed that:

- **ReLU and Leaky ReLU** offer fast convergence, stable gradients, and expressive decision boundaries.
- **Tanh** performs moderately, offering smoother but less precise boundaries.
- **Sigmoid** is unsuitable for complex tasks due to gradient saturation.

The accompanying Jupyter notebook provides full reproducibility, enabling readers to experiment further by adjusting network depth, width, activation variants, or optimisation methods.

10. References

- Bishop, C. M. *Pattern Recognition and Machine Learning.* Springer, 2006.
- Goodfellow, I., Bengio, Y., Courville, A. *Deep Learning.* MIT Press, 2016.
- Nair, V., Hinton, G. “Rectified Linear Units Improve Restricted Boltzmann Machines.” ICML, 2010.
- Maas, A. et al. “Rectifier Nonlinearities Improve Neural Network Acoustic Models.” ICML, 2013.
- Glorot, X., Bordes, A., Bengio, Y. “Deep Sparse Rectifier Neural Networks.” AISTATS, 2011.