Report: The Impact of Activation Functions on Neural Network Performance

February 6, 2026

1 Introduction

Activation functions introduce the non-linearity necessary for neural networks to learn complex patterns. This study utilizes the **Fashion-MNIST** dataset to evaluate the impact of various activation functions on a Multi-Layer Perceptron (MLP). By isolating the activation function as the sole variable, we compare traditional methods (Sigmoid, Tanh) against modern alternatives (ReLU, Leaky ReLU, SELU, SiLU).

2 Methodology

2.1 Dataset

The **Fashion-MNIST** dataset was used for this experiment. It consists of 70,000 grayscale images of clothing items, each 28x28 pixels in size. The dataset is divided into 60,000 training images and 10,000 testing images, categorized into 10 distinct classes (e.g., T-shirt/top, Trouser, Pullover). It serves as a more challenging alternative to the classic MNIST dataset of handwritten digits.

Before being fed into the network, the images were transformed into tensors and normalized to a range of [-1, 1].

2.2 Model Architecture

A simple Multi-Layer Perceptron (MLP) was implemented using PyTorch. The architecture was designed to be straightforward to ensure that the performance differences could be primarily attributed to the activation functions.

The MLP consists of:

1. An input layer that flattens the 28x28 images into a 784-dimensional vector.
2. A hidden layer with 128 neurons, followed by the activation function being tested.
3. A second hidden layer with 10 neurons, also followed by the same activation function.
4. An output layer with 10 neurons, corresponding to the 10 classes in the dataset.

The same architecture was used for every activation function tested.

2.3 Training Process

The training process was standardized across all experiments to ensure a fair comparison. The key hyperparameters were:

- **Optimizer:** Adam optimizer, a widely used and effective optimization algorithm.
- **Learning Rate:** 0.001.
- **Loss Function:** Cross-Entropy Loss, which is standard for multi-class classification problems.
- **Number of Epochs:** 10.
- **Batch Size:** 64.

For each epoch, the model was trained on the entire training dataset and then evaluated on the test dataset to measure its validation accuracy.

2.4 Activation Functions Tested

The following activation functions were evaluated:

- **Traditional:** Sigmoid, Tanh
- **ReLU and its variants:** ReLU, Leaky ReLU, ELU, PReLU, SELU
- **Advanced Functions:** GELU, SiLU, Mish, Softplus, Hardswish

3 Results and Analysis

The performance of each activation function was evaluated based on training loss, validation accuracy, and convergence speed. The results are summarized in Table 1, sorted by the best validation accuracy achieved.

Table 1: Comparison of Activation Function Performance.

Activation	Best Accuracy	Final Accuracy	Final Loss
SiLU	88.58%	88.58%	0.2361
SELU	88.22%	87.92%	0.2587
Mish	88.04%	88.04%	0.2434
Hardswish	88.02%	86.91%	0.2414
Softplus	88.00%	88.00%	0.2553
GELU	87.82%	87.82%	0.2460
PReLU	87.75%	87.75%	0.2594
ReLU	87.73%	87.73%	0.2598
Leaky ReLU	87.68%	87.68%	0.2511
ELU	87.67%	87.67%	0.2545
Sigmoid	87.34%	87.34%	0.2883
Tanh	87.11%	86.21%	0.2854

3.1 Performance Comparison

The results clearly show that modern activation functions outperform the traditional ones.

- **Top Performers: SiLU (Swish)** achieved the highest accuracy at **88.58%**. Other strong performers include **SELU**, **Mish**, and **Hardswish**, all of which surpassed the 88% accuracy mark. These functions are known for their smooth, non-monotonic nature, which can help with gradient flow and allow for more complex representations.
- **ReLU and its Variants:** The standard **ReLU** performed very well, achieving **87.73%** accuracy. Its variants, such as **Leaky ReLU**, **ELU**, and **PReLU**, offered slight variations in performance but were all in a similar range. This confirms why ReLU-based functions are the default choice in many deep learning applications. They are computationally efficient and generally avoid the vanishing gradient problem.
- **Traditional Activations: Sigmoid** and **Tanh** were the worst-performing functions. Sigmoid, in particular, had a much higher initial training loss, indicating slower convergence. This is likely due to the **vanishing gradient problem**, where the gradients become extremely small for neurons with high or low activations, effectively halting learning. Tanh performed slightly better than Sigmoid as its output is zero-centered, but it still suffers from the same gradient saturation issue.

3.2 Convergence Speed and Training Dynamics

The training loss and validation accuracy curves provide insights into how quickly each model learned.

- **Faster Convergence:** Functions like **SiLU**, **Mish**, and **GELU** demonstrated faster convergence, reaching lower training loss values more quickly than others.

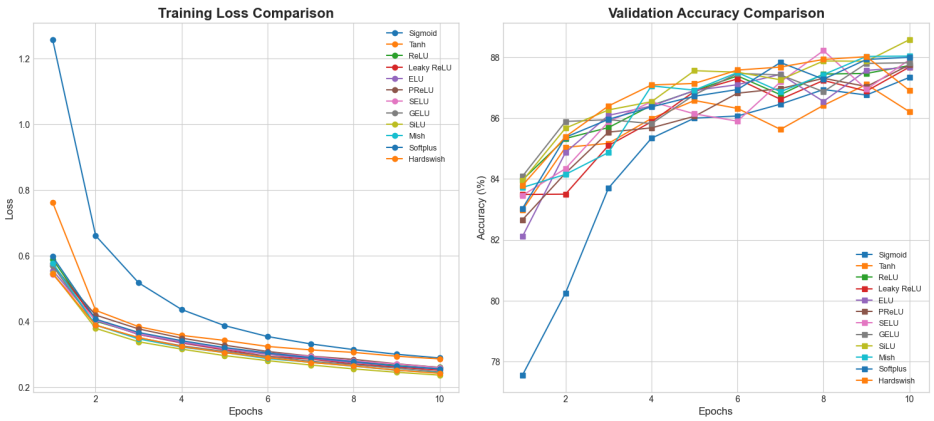

Figure 1: Comparison Graphs

- **Slower Convergence: Sigmoid** had the slowest start, with a significantly higher loss in the first epoch. This is a classic symptom of the vanishing gradient problem, which is more pronounced in deeper networks but is still observable here.

3.3 Discussion of Issues

- **Vanishing Gradients:** The poor performance of Sigmoid and Tanh is a direct result of this issue. Their derivatives are close to zero for a large portion of their input range, which slows down or stops the learning process for neurons in that state.
- **Dying Neurons:** While not explicitly measured, the "dying ReLU" problem—where neurons become inactive and only output zero—is a known drawback of the standard ReLU function. The strong performance of **Leaky ReLU** and **ELU**, which allow for small negative activations, suggests that they successfully mitigate this potential issue by ensuring that neurons always have a non-zero gradient.

4 Conclusion

This experiment demonstrates that the choice of activation function plays a significant role in neural network performance. Modern activation functions consistently outperform traditional ones in both accuracy and convergence speed.

SiLU emerged as the best-performing activation function for this task, achieving the highest accuracy and stable training behavior. ReLU and its variants also proved to be effective and reliable. In contrast, Sigmoid and Tanh showed inferior performance due to vanishing gradient issues, reinforcing their limited suitability for deep networks.