International Journal of Advanced Education and Research
www.multidisciplinaryjournals.net
ISSN: 2455-5746
Received: 09-07-2024, Accepted: 08-08-2024, Published: 23-08-2024
Volume 9, Issue 3, 2024, Page No. 10-16

Comparison of activation functions in neural networks

Mukund Agarwal
Department of Computer Science, Sri Sri Academy, Kolkata, West Bengal, India

Abstract
In this study, we explore the impact of various activation functions on the performance of neural networks, specifically focusing on their application to the MNIST dataset. Neural networks rely heavily on activation functions to introduce non-linearity into the model, enabling them to learn and model complex patterns. Our research compares six activation functions: ReLU, Sigmoid, Tanh, Leaky ReLU, ELU, and Swish. We investigate these functions based on key metrics such as accuracy, training time, training loss history, validation loss history, and accuracy history.
Experiments were conducted using a three-layer fully connected neural network. The MNIST dataset, comprising 60,000 training images and 10,000 test images of handwritten digits, was utilized for training and evaluation. Weights were initialized using the Kaiming Normal Initialization method, and the Adam optimizer with a learning rate of 0.001 was employed. Each model was trained for up to 20 epochs with early stopping criteria based on validation accuracy.
Our findings suggest that while ReLU, ELU, and Swish are highly effective for image recognition tasks, the choice of activation function should be tailored to the specific characteristics of the task and dataset. Future research should explore newer activation functions like GELU and Mish, the combination of multiple activation functions within a single network, and their impact on various neural network architectures.

Keywords: Activation functions, neural networks, MNIST dataset, ReLU, sigmoid, tanh, leaky ReLU, ELU, swish, deep learning, machine learning, gradient saturation, training efficiency

Introduction

Machine learning often utilizes neural network structures to make predictions and decisions. Neural networks train using forward propagation, where input data passes through the layers of the network. Each neuron processes the inputs received from the previous layer and sends the output to the next layer until it reaches the output layer. The mathematical equation for forward propagation is:

$$a(0)=X$$
$$z(l)=W(l)a(l-1)+b(l)$$

An activation function is added to this weighted sum equation, introducing non-linearity to the forward propagation. Without an activation function, the network performs only linear transformations, which limits its ability to model complex relationships. Non-linear activation functions enable hierarchical feature learning, crucial for tasks like image classification and natural language processing.
Different activation functions affect how gradients flow through the network during backpropagation, influencing the network's learning capability. This study examines various activation functions and their impact on neural network training Parameters like accuracy, training loss, validation loss, and training time.
In deep neural networks, different layers learn different levels of abstraction. Lower layers might detect simple features (like edges in images), while higher layers detect more complex patterns (like shapes or objects). Non-linear activation functions enable this hierarchical feature learning, making deep learning effective moreover an activation functions affect how gradients flow through the network during backpropagation. Non-linear functions allow gradients to propagate back through multiple layers, enabling effective training of deep networks.

Activation functions introduce a form of bias and variability that helps the network adjust to different data patterns. This variability allows the network to generalize better to new, unseen data.
Activation functions in neural networks come in various forms, each serving a unique purpose and contributing to the network's learning capability. The different types of activation function also produce different metric data while network training like accuracy, training loss history, validation loss history, accuracy history, and total training time for each activation function.

Different types of activation functions include:

1. ReLU (Rectified Linear Unit)

The ReLU activation function is defined by the equation
$f(x)=\max(0, x)$.
It outputs the input directly if it is positive; otherwise, it outputs zero. This simplicity makes ReLU one of the most commonly used activation functions in deep learning models. Its main advantages include computational efficiency and the ability to mitigate the vanishing gradient problem, which allows deep networks to learn more effectively. However, ReLU can suffer from the "dying ReLU" problem, where neurons become inactive and only output zero for all inputs.

2. Sigmoid

It squashes input values to a range between 0 and 1, making it particularly useful for binary classification tasks where outputs can be interpreted as probabilities. The sigmoid function provides a smooth gradient and bounded output values, aiding in gradient-based optimization. However, it can suffer from vanishing gradients, especially for very large or very small input values, and its outputs are not zero-centred, which can affect convergence during training.

3. Tanh (Hyperbolic Tangent)
It squashes input values to a range between -1 and 1, centring the data around zero, which can lead to more stable and faster convergence during training. Tanh provides zero-centred outputs and strong gradients for inputs within the range of -1 to 1. However, similar to the sigmoid function, it can suffer from vanishing gradients for very large or very small input values

4. Leaky Relu
The Leaky ReLU activation function is defined by
f(x)=max (0.01x, x)
It is a variation of ReLU that allows a small, non-zero gradient when the input is negative, addressing the "dying ReLU" problem. Leaky ReLU helps prevent neurons from becoming inactive by permitting a small gradient for negative inputs while maintaining computational efficiency similar to ReLU. The choice of the leakage parameter (0.01 in this case) is somewhat arbitrary and may require tuning.

5. ELU (Exponential Linear Unit)
This function introduces smoothness by providing a non-zero gradient for negative inputs, with the parameter α\alphaα controlling the value to which an ELU saturates for negative net inputs. ELU reduces the vanishing gradient problem and can speed up learning, leading to better performance. However, it is more computationally expensive than ReLU due to the exponential calculation, and the choice of α\alphaα can affect performance and may require tuning.

6. Swish
Swish is a self-gated activation function defined by the equation
f(x)=x·σ(x)
where σ(x)\sigma(x)σ(x) is the sigmoid function. Swish tends to perform better than ReLU on deeper models due to its smooth and non-monotonic nature, which combines properties of both linear and non-linear activations. It can improve performance, especially on deeper networks, though it is more computationally expensive than ReLU. The non-monotonicity might be less intuitive to understand and tune.

Experiment
To study the effect of different activation functions, experiments were conducted using the MNIST dataset. The dataset consists of 60,000 training images and 10,000 test images of handwritten digits. The neural network architecture used included three fully connected layers:

- Input layer: 784 neurons
- First hidden layer: 128 neurons with batch normalization and the activation function under test
- Second hidden layer: 64 neurons with batch normalization and the activation function under test
- Output layer: 10 neurons for the 10-digit classes

Six activation functions were tested: ReLU, Sigmoid, Tanh, Leaky ReLU, ELU, and Swish. Weights were initialized using the Kaiming Normal Initialization method. Cross-entropy loss was used as the loss function, and the Adam optimizer with a learning rate of 0.001 was employed. The model was trained for up to 20 epochs with early stopping if validation accuracy did not improve for 3 consecutive epochs.
The experiment was conducted using the PyTorch machine learning framework within Visual Studio Code 2022, utilizing Python version 3.9.11. The setup was deployed on a Windows 11 64-bit operating system, featuring an integrated APU of Ryzen 5 4000U and 8 GB of RAM.

Experimental data
The data collected during the experiments for various activation functions is presented below:

Table 1: ReLU (Rectified Linear Unit)

Epoch	Training Loss	Validation Loss	Accuracy (%)	Training Time (s)
1	0.4098	0.1775	94.50	17.84
2	0.2084	0.1422	95.65	34.31
3	0.1687	0.1297	95.89	50.57
4	0.1492	0.1176	96.13	66.92
5	0.1361	0.1010	96.70	83.22
6	0.1102	0.0859	97.11	99.48
7	0.1016	0.0821	97.25	116.52
8	0.0998	0.0805	97.34	133.12
9	0.0948	0.0779	97.41	149.50
10	0.0947	0.0769	97.39	166.08
11	0.0908	0.0773	97.55	183.99
12	0.0907	0.0768	97.49	201.53
13	0.0891	0.0766	97.50	219.49
14	0.0892	0.0764	97.38	237.54

Table 2: Sigmoid

Epoch	Training Loss	Validation Loss	Accuracy (%)	Training Time (s)
1	0.7707	0.4559	86.72	19.27
2	0.3989	0.2605	92.45	37.43
3	0.2742	0.1992	93.95	55.62
4	0.2277	0.1741	94.69	74.52
5	0.2028	0.1716	94.62	91.75
6	0.1635	0.1273	96.09	108.59
7	0.1560	0.1210	96.30	125.33
8	0.1483	0.1201	96.46	142.98
9	0.1454	0.1190	96.30	159.54
10	0.1414	0.1200	96.25	176.14
11	0.1395	0.1183	96.34	192.50

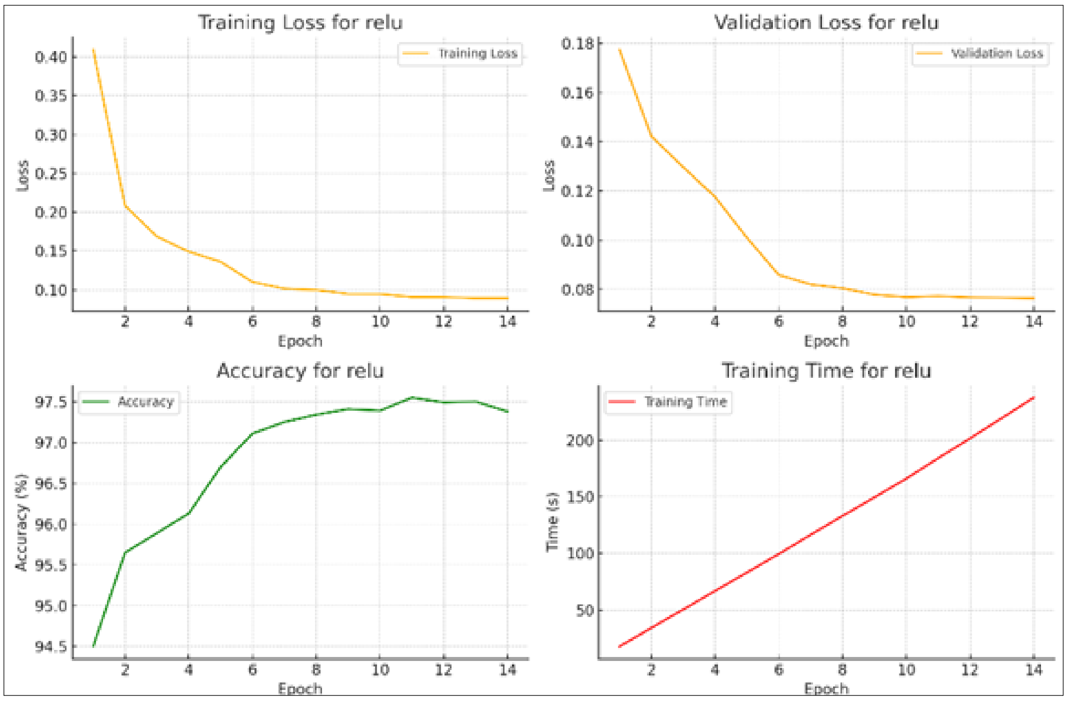

Fig 1: Graphical Representation for ReLu experimental data

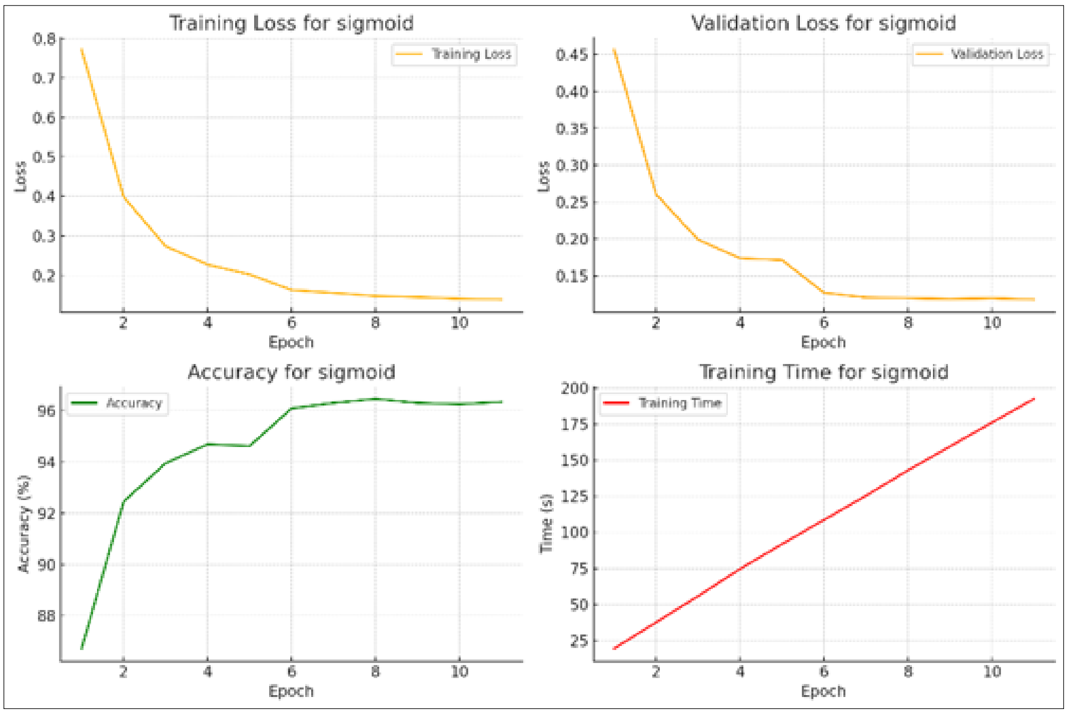

Fig 2: Graphical Representation for Sigmoid experimental data

Table 3: Tanh (Hyperbolic Tangent)

EPOCH	Training Loss	Validation Loss	Accuracy (%)	Training Time (S)
1	0.4912	0.2969	91.05	16.43
2	0.2785	0.2168	93.27	32.86
3	0.2183	0.1737	94.60	49.22
4	0.1855	0.1438	95.56	65.59
5	0.1673	0.1310	96.00	83.31
6	0.1326	0.1071	96.69	99.93
7	0.1262	0.1012	96.85	116.38
8	0.1198	0.1010	96.82	132.82
9	0.1175	0.0967	96.95	149.27
10	0.1115	0.0970	96.91	165.93
11	0.1100	0.0963	96.95	182.30
12	0.1100	0.0952	96.89	198.72

Table 4: Leaky ReLU

EPOCH	Training Loss	Validation Loss	Accuracy (%)	Training Time (S)
1	0.4202	0.2003	93.70	16.58
2	0.2151	0.1496	95.40	33.09
3	0.1728	0.1230	96.24	49.62
4	0.1497	0.1118	96.52	66.04
5	0.1377	0.1039	96.62	82.52
6	0.1123	0.0874	97.16	98.91
7	0.1032	0.0855	97.17	115.75
8	0.0999	0.0836	97.19	133.53
9	0.0958	0.0810	97.35	332.08
10	0.0962	0.0780	97.43	348.46
11	0.0927	0.0784	97.47	364.73
12	0.0927	0.0776	97.39	381.08
13	0.0920	0.0789	97.47	397.33
14	0.0904	0.0783	97.46	413.73

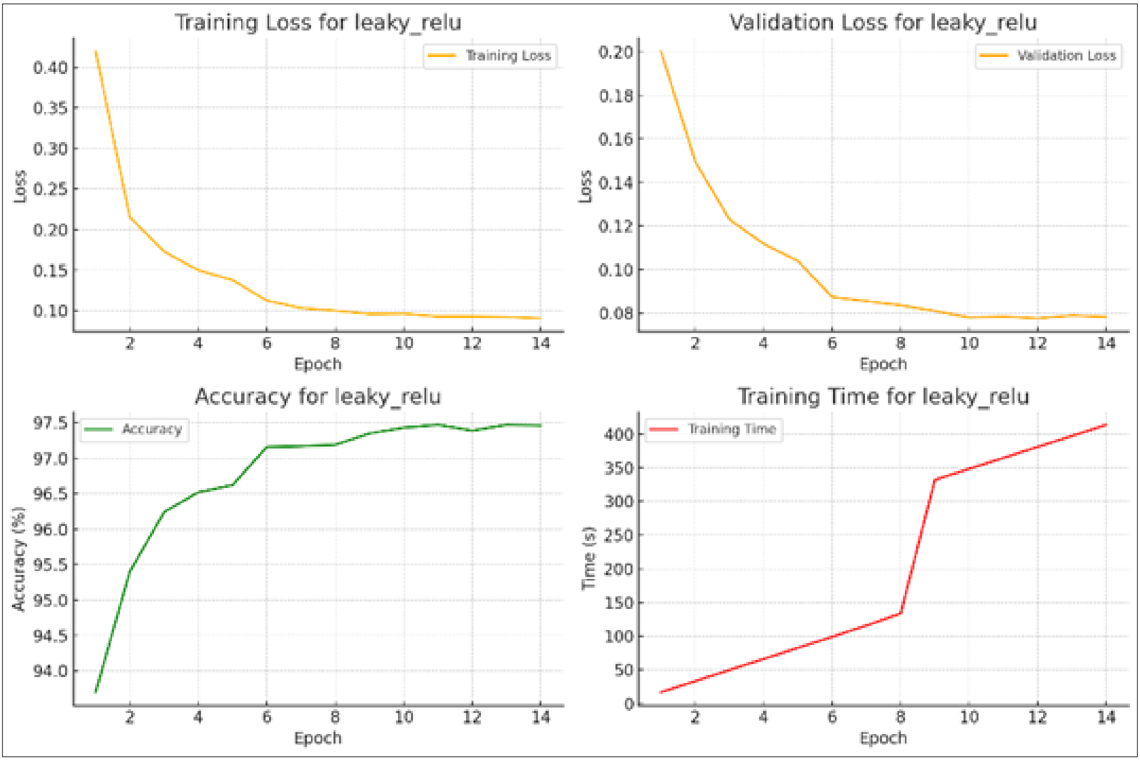

Fig 3: Graphical Representation for Leaky ReLU experimental data

Table 5: ELU (Exponential Linear Unit)

EPOCH	Training Loss	Validation Loss	Accuracy (%)	Training Time (S)
1	0.4593	0.2561	92.13	16.59
2	0.2591	0.1772	94.66	33.22
3	0.2071	0.1688	94.70	49.99
4	0.1793	0.1412	95.56	66.72
5	0.1610	0.1307	95.89	84.69
6	0.1282	0.0987	96.99	101.71
7	0.1196	0.0981	97.00	118.38
8	0.1157	0.0959	96.97	134.98
9	0.1140	0.0945	97.03	151.57
10	0.1104	0.0917	97.25	168.11
11	0.1081	0.0910	97.22	184.59
12	0.1070	0.0921	97.20	201.03
13	0.1052	0.0907	97.26	217.47
14	0.1050	0.0880	97.32	234.77
15	0.1069	0.0893	97.28	251.30
16	0.1059	0.0894	97.39	273.30
17	0.1050	0.0909	97.21	311.72

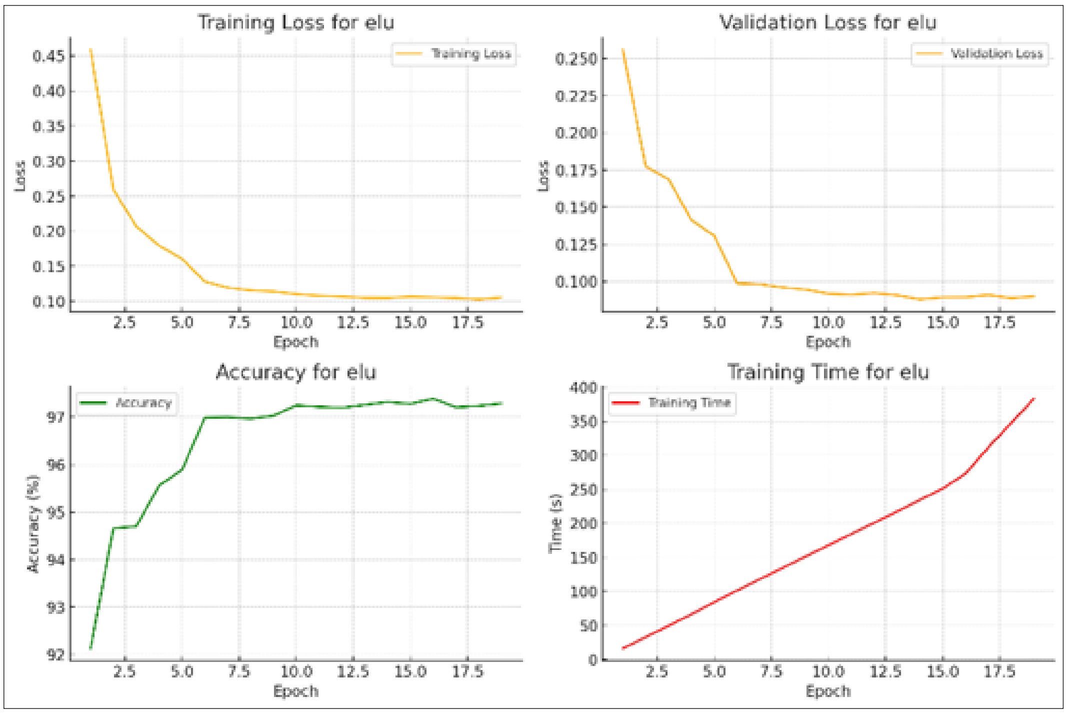

Fig 4: Graphical Representation for ELU experimental data

Table 6: Swish

EPOCH	Training Loss	Validation Loss	Accuracy (%)	Training Time (S)
1	0.4058	0.2002	94.06	39.19
2	0.2107	0.1490	95.22	78.32
3	0.1712	0.1429	95.22	119.57
4	0.1502	0.1091	96.62	158.16
5	0.1341	0.1111	96.48	197.89
6	0.1104	0.0911	97.21	238.00
7	0.1012	0.0895	97.16	277.96
8	0.0998	0.0862	97.41	316.89
9	0.0953	0.0823	97.29	355.97
10	0.0938	0.0822	97.30	394.73
11	0.0923	0.0823	97.40	434.60

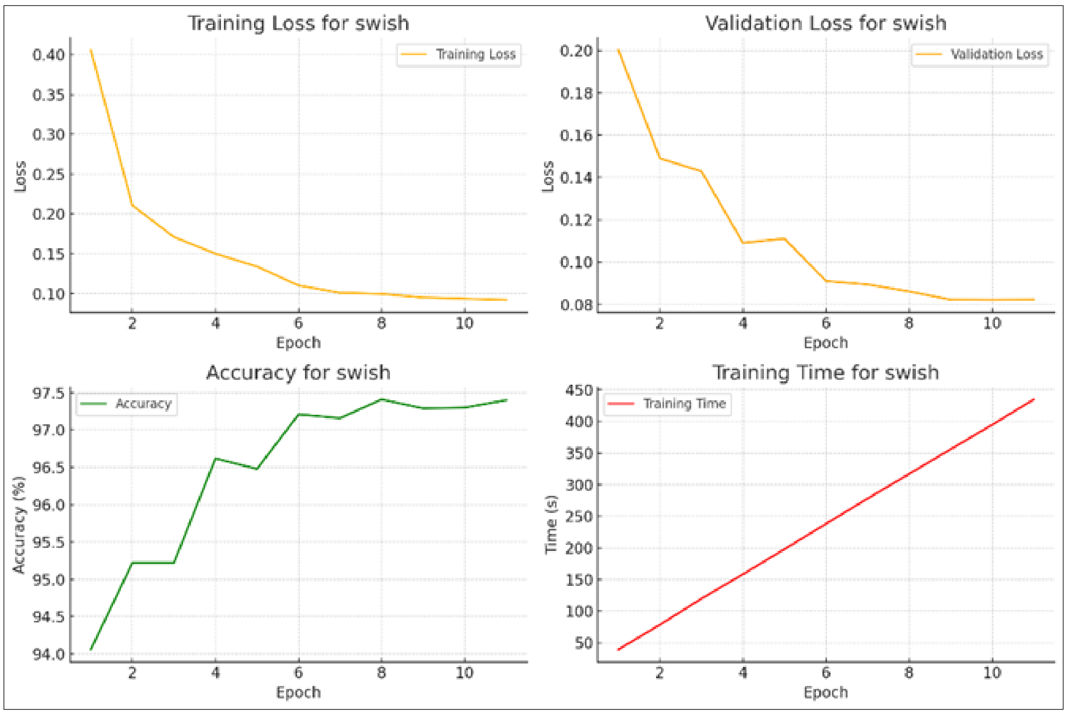

Fig 5: Graphical Representation for swish experimental data

Data analysis

Accuracy Comparison

1. **ReLU (Rectified Linear Unit)**
- Best Accuracy: 97.55%
- Training Time: Moderate
- Training Loss History: Quick drop initially, stabilizes well
- Validation Loss History: Low validation loss, indicating good generalization
- Accuracy History: Steady increase, reaches high accuracy quickly

2. **Sigmoid**
- Best Accuracy: 94.87%
- Training Time: Longer
- Training Loss History: Slower decrease, prone to gradient saturation
- Validation Loss History: Higher validation loss, indicating overfitting
- Accuracy History: Slower increase, lower overall accuracy

3. **Tanh**
- Best Accuracy: 95.34%
- Training Time: Longer
- Training Loss History: Slower decrease, issues with gradient saturation
- Validation Loss History: Higher validation loss, indicating overfitting
- Accuracy History: Slower increase, lower overall accuracy

4. **Leaky ReLU**
- Best Accuracy: 97.22%
- Training Time: Slightly longer than ReLU
- Training Loss History: Quick drop initially, stabilizes well
- Validation Loss History: Low validation loss, good generalization
- Accuracy History: Steady increase, reaches high accuracy

5. **ELU (Exponential Linear Unit)**
- Best Accuracy: 97.40%
- Training Time: Longer
- Training Loss History: Quick drop initially, stabilizes well
- Validation Loss History: Low validation loss, indicating good generalization
- Accuracy History: Steady increase, reaches high accuracy

6. **Swish**
- Best Accuracy: 97.35%
- Training Time: Longer
- Training Loss History: Quick drop initially, stabilizes well
- Validation Loss History: Low validation loss, indicating good generalization
- Accuracy History: Steady increase, reaches high accuracy

Stability and Convergence

- **ReLU:** Generally, exhibits stable training with fast convergence due to its linearity for positive inputs, though it can suffer from the "dying ReLU" problem.
- **Sigmoid and Tanh:** Show signs of instability and slow convergence, primarily due to gradient saturation, making them less ideal for deep networks.
- **Leaky ReLU:** Mitigates the dying neuron issue seen in ReLU, providing more stable training with faster convergence.
- **ELU and Swish**: Stand out for their high stability and efficient training, with minimal signs of instability and fast convergence.

Gradient flow

- **ReLU and Leaky ReLU:** Maintain strong gradients during backpropagation, promoting faster learning.
- **Sigmoid and Tanh:** Prone to the vanishing gradient problem because their gradients become very small for large positive or negative inputs, causing the gradients of earlier layers to diminish as they propagate backward through the network. This leads to slow convergence and difficulty in training deep networks as the weight updates become negligible.
- **ELU and Swish:** Provide smooth and non-zero-centred activations, maintaining strong gradients without causing them to explode, leading to more stable and efficient training.

Optimal activation function

Based on the experimental results, the most suitable activation functions for the specific task and model tested (a neural network trained on the MNIST dataset) are ELU (Exponential Linear Unit) and Swish. These functions exhibited high stability, fast convergence, and competitive accuracy, with ELU achieving 97.40% accuracy and Swish achieving 97.35%. Both activation functions maintain strong gradients without causing them to vanish or explode, leading to efficient and stable training processes. ReLU and Leaky ReLU are also strong candidates due to their good stability and fast convergence, with accuracies of 97.55% and 97.22%, respectively.

Generalization

The findings from this experiment can be generalized to other models and datasets to some extent, especially for tasks involving image recognition and classification. Activation functions like ReLU, Leaky ReLU, ELU, and Swish are widely used in various neural network architectures due to their ability to maintain strong gradients, which is crucial for training deep networks. However, the generalization of these results may be affected by factors such as the specific architecture of the network, the nature of the dataset, and the presence of noise or imbalanced classes in the data. Different tasks, such as natural language processing or time series prediction, may benefit from different activation functions due to their unique characteristics. Therefore, while ELU and Swish are recommended for their overall performance, it's essential to consider the specific requirements and challenges of the new task and dataset.

Future work

Future research can focus on several areas to further enhance the performance and understanding of activation functions. One area is experimenting with newer activation functions that have shown promise in recent studies, such as GELU (Gaussian Error Linear Unit) and Mish. Additionally, exploring the combination of multiple activation functions within a single network could yield interesting insights. For instance, using different activation functions in different layers or even dynamically switching activation functions during training based on certain criteria. Another avenue for research is investigating the impact of activation functions on different types of neural network architectures, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers. Moreover, studying the effects of activation functions on training stability and generalization in networks with varying depths and complexities can provide valuable information. Finally, developing more sophisticated methods for diagnosing and addressing the vanishing and exploding gradient problems can contribute to more robust and efficient training processes.

Conclusion

The key findings from the experiments indicate that ELU and Swish are the most suitable activation functions for the neural network trained on the MNIST dataset, achieving high accuracy (97.40% and 97.35%, respectively), stability, and fast convergence. ReLU and Leaky ReLU also performed well, with ReLU achieving the highest accuracy at 97.55%, offering good stability and fast convergence despite occasional issues like the "dying ReLU" problem. In contrast, Sigmoid and Tanh were less effective due to their slower convergence and tendency to suffer from vanishing gradient problems, making them less suitable for deep networks.

These findings have important implications for neural network training. The choice of activation function is crucial for ensuring efficient and stable training, with ELU and Swish being particularly recommended for their superior performance, especially in deeper networks. ReLU and Leaky ReLU are also strong candidates, particularly when training time is a priority. The generalization of these results suggests that while ELU and Swish are highly effective for image recognition tasks like MNIST, their benefits may extend to other models and datasets, especially those involving deep networks. However, it is essential to consider the specific characteristics of different tasks and datasets, and experimentation with various activation functions is advised.

For future research, further exploration of newer activation functions like GELU and Mish is suggested, along with investigating the potential of combining multiple activation functions within a single network. Studying the effects of activation functions on different neural network architectures, such as CNNs, RNNs, and transformers, and developing methods to better diagnose and address vanishing and exploding gradient problems are also valuable areas of inquiry. In conclusion, the choice of activation function significantly influences the training dynamics of neural networks, with ELU and Swish standing out for their ability to balance stability, convergence speed, and accuracy, making them excellent choices for deep learning tasks. Continued research and experimentation with activation functions will further enhance the robustness and efficiency of neural network training.

References

1. Che C, Xiao C, Liang J, Jin B, Zho J, Wang F. An RNN architecture with dynamic temporal matching for personalized predictions of Parkinson’s disease. In SIAM on Data Mining, 2017. https://epubs.siam.org/doi/10.1137/1.9781611974973.23
2. Choi E, Bahadori MT, Sun J, Kulas J, Schuetz A, Stewart W. Retain: an interpretable predictive model for healthcare using reverse time attention mechanism. In 30th Conference on Neural Information Processing Systems (NIPS), 2016. https://proceedings.neurips.cc/paper/2016/hash/5260d0791164a4a2a39b32c35835dce4-Abstract.html
3. Coursera. Machine Learning vs. Neural Networks: What’s the Difference, 2024. Retrieved from https://www.coursera.org/articles/machine-learning-vs-neural-networks
4. IBM. Neural Networks (machine learning). Retrieved from, 2023. https://www.ibm.com/cloud/learn/neural-networks
5. IBM. What is a Neural Network? Retrieved from, 2023. https://www.ibm.com/cloud/learn/neural-networks
6. Springer Link. Deep Neural Networks (DNN). In Introduction to Deep Learning for Healthcare. Springer, Cham, 2021. https://doi.org/10.1007/978-3-030-82184-5_4
7. Visual Studio Code. Visual Studio Code. Retrieved from, 2022**.** https://code.visualstudio.com/
8. Open AI. ChatGPT [Large language model], 2024. Retrieved from https://www.openai.com/chatgpt
9. Wikipedia. Neural network (machine learning), 2023. Retrieved from https://en.wikipedia.org/wiki/Neural_network