International Journal of Advanced Smart Convergence Vol.13 No.2 80-87 (2024)
http://dx.doi.org/10.7236/IJASC.2024.13.2.80

IJASC 24-2-10

Optimization of Model based on Relu Activation Function in MLP Neural Network Model

Ye Rim Youn[1], Jinkeun Hong[2]

[1] Student, Division of Computer Engineering, Baekseok University, South Korea
[2] Professor, Division of Advanced IT, Beakseok University, South Korea
[1]starwhale@bu.ac.kr, [2]Corresponding author: jkhong@bu.ac.kr

Abstract

This paper focuses on improving accuracy in constrained computing settings by employing the ReLU (Rectified Linear Unit) activation function. The research conducted involves modifying parameters of the ReLU function and comparing performance in terms of accuracy and computational time.

This paper specifically focuses on optimizing ReLU in the context of a Multilayer Perceptron (MLP) by determining the ideal values for features such as the dimensions of the linear layers and the learning rate (Ir). In order to optimize performance, the paper experiments with adjusting parameters like the size dimensions of linear layers and Ir values to induce the best performance outcomes. The experimental results show that using ReLU alone yielded the highest accuracy of 96.7% when the dimension sizes were 30 - 10 and the Ir value was 1. When combining ReLU with the Adam optimizer, the optimal model configuration had dimension sizes of 60 - 40 - 10, and an Ir value of 0.001, which resulted in the highest accuracy of 97.07%.

Keywords: *AI, MLP, Relu, optimizer, Accuracy*

1. Introduction

Artificial Intelligence (AI) technology is based on human capabilities such as learning and reasoning, and its development aims to surpass human intelligence in terms of analytical depth and scale. Recently, the application of AI has expanded to include generative AIs such as GPT-4, which are used as tools for drawing, composing music, autonomous driving, and video analysis. AI supports human decision-making across various domains [1-2].

The expansion of AI into practical research can be observed through policy changes that have led to increased investments in AI development by major global tech companies. AI technology requires high-performance specifications in memory and processors for training and validation. Technologies like deep learning, in particular, necessitate devices that support high performance and involve significant costs.

Manuscript Received: April. 17, 2024 / Revised: April. 24, 2024 / Accepted: April. 30, 2024
[2]Corresponding Author: jkhong@bu.ac.kr
Tel: +82- 41-550-2445, Fax: +82-41-550-9027
Professor, Division of Advanced IT, Beakseok University, South Korea

However, in the field of AI education, it is often challenging to secure high-performance hardware that meets the demands of research environments. Especially from a cost perspective, constructing a hardware process environment for AI training that aligns with realistic investment levels is difficult, although necessary; in reality, however, this proves to be a challenging situation [3].

Previous research has proposed modifications to the ReLU function, introducing new functions that enhance the non-linearity of feature maps. By replacing the slope of ReLU with a learnable parameter, the study introduces the Parametric ReLU (PReLU) activation function [4]. Additionally, it proposes a new activation function called Exponential Linear Unit (ELU), which addresses one of the shortcomings of ReLU by converging exponentially for negative inputs [5].

This study focuses on improving performance accuracy in constrained computing environments by utilizing the ReLU activation function, which is capable of training validation. The paper concentrates on optimizing individual parameters when applying the ReLU function to a specific dataset within a Multilayer Perceptron (MLP) setting.

To achieve this, the research examines the optimization performance upon altering parameters such as the values of linear layers in the basic ReLU activation function. Additionally, by incorporating optimizers like Adam, the study seeks to determine the conditions for optimizing accuracy performance based on these adjustments.

Chapter 2 introduces the training model architecture, detailing Multilayer Perceptrons (MLP), including the input layer, hidden layers, output layer, activation functions, and loss functions. Chapter 3 presents the experimental setup and results. This paper utilized the Fashion MNIST dataset and focused on tuning ReLU parameters and the performance of a 3-layer ReLU model combined with the Adam optimizer. Chapter 4 concludes with a discussion of the findings and future research directions.

2. Training Model System

2.1 MLP

The Multi-Layer Perceptron (MLP) is an artificial neural network with a multilayer structure of perceptron neurons, commonly referred to as a feedforward neural network. MLP extends the perceptron theory and typically includes at least one nonlinear hidden layer. These layers enable the extraction of complex patterns from training data. The architecture of an MLP consists of an input - hidden layers - output layer arranged in a feedforward network.

2.2 Input Layer, Hidden Layers, Output Layer

The input layer of an MLP is analogous to a single perceptron, representing the feature values in a sample. The number of nodes is determined by the number of features, and since no computations are performed at this stage.

The hidden layers in an MLP are adjusted by parameters, such as the number of units in each layer. These hidden units and the total number of hidden layers significantly impact the training duration and accuracy. The ReLU function, utilized in this research, is primarily employed as the activation function in these layers.

ReLU is chosen for its non-linear properties, which allow it to either pass the input directly or output zero, essential for learning complex data and decision boundaries. Another reason for using ReLU is its ability to

maintain sparsity during training, as it outputs zero for negative inputs, and its simple computational form enhances efficiency.

Finally, the output layer is responsible for the MLP's final output. The number of artificial neurons in this layer matches the total number of categories being classified. This layer is where the final results are output, the activation function is selected, and the output values are adjusted. Typically, a single output neuron is used for regression problems, while multiple neurons are employed for classification tasks, depending on the number of classes [6-7].

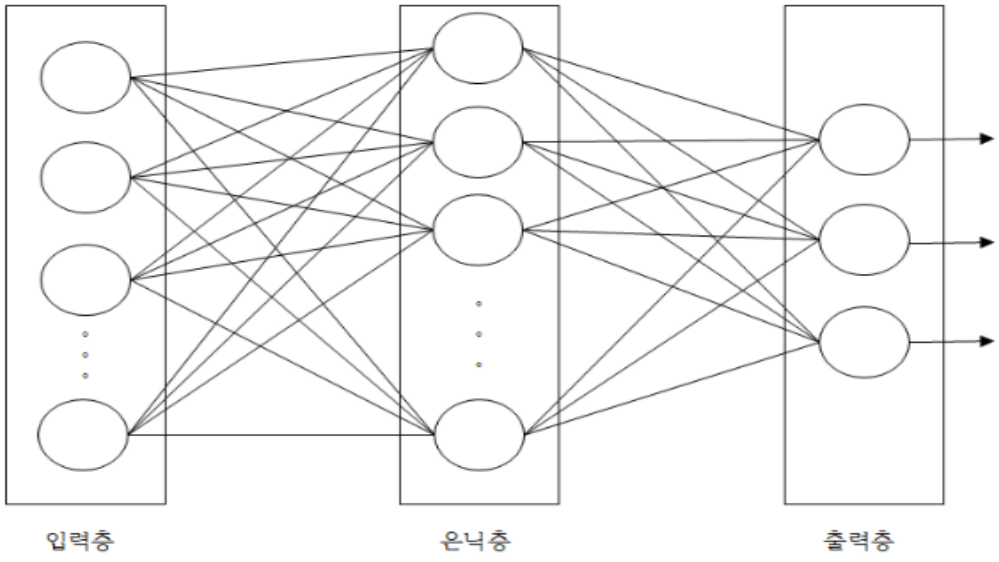

Figure 1. MLPO Input layer, Hidden layer Output layer

2.3 Activation Functions

In perceptrons, the step function is used as the activation function. In the MLP, an extension of perceptron theory, various nonlinear functions are employed as activation functions. These functions determine each neuron's output by transforming the sum of weighted inputs and transmitting it to th e next layer. Furthermore, activation functions add nonlinearity and are crucial for enabling neural n etworks to approximate complex functions. Activation functions used in MLPs include the sigmoid f unction, hyperbolic tangent function, leaky ReLU, and ReLU, which is utilized in this study.

2.4 Loss Functions

Loss functions quantify the divergence between the actual targets and the predictions made by the neural network, serving as a means to assess the precision of predictions throughout the training process.Commonly used loss functions include mean squared error, cross-entropy loss, and binary cross-entropy loss. Mean squared error is typically employed for regression problems, while cross-entropy loss is frequently employed in solving classification issues. Moreover, cross-entropy loss calculates the difference in class probabilities to train the model in classification scenarios.

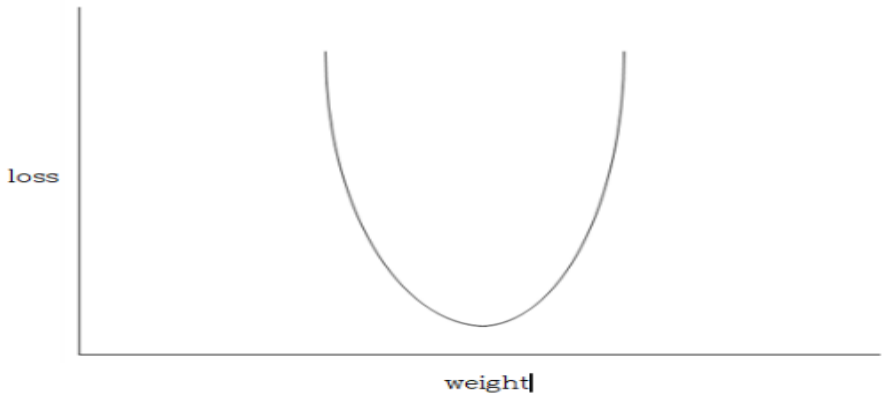

Figure 2. Loss function of loss and weight

3. Paper title and author information

3.1 Fashion MNIST Dataset

The data used in this research is the Fashion MNIST dataset. Fashion MNIST is a dataset created for machine learning and computer vision research, featuring a structure similar to the handwritten digit dataset, MNIST. Unlike MNIST, which includes images of handwritten digits, Fashion MNIST comprises various fashion items such as clothing and accessories, thus providing more complex image patterns. This complexity allows for a broader application of existing models and algorithms that are designed for MNIST, yet capable of handling more intricate image representations.

The Fashion MNIST dataset consists of 10 classes, including t-shirts/tops, bags, trousers, sandals, pullovers, coats, dresses, shirts, sneakers, and ankle boots. The images are 28x28 pixel grayscale, with each pixel holding a value ranging from 0 to 255. The dataset contains 70,000 images in total, with 60,000 designated for training and the remaining 10,000 used for testing[8].

Figure 3. Fashion MNIST Data set

3.2 ReLU

The ReLU (Rectified Linear Unit) function is an activation function that was developed subsequent to the sigmoid function. The sigmoid function has the limitation of outputting values in the range of 0 to 1, leading to significant reductions in these values as more layers are processed, a phenomenon known as gradient vanishing. To address this issue, the ReLU function is employed. The formula for ReLU is as follows in

equation (1).

$$\mathrm{Relu}(x) = \max(0, x) \tag{1}$$

The ReLU (Rectified Linear Unit) function is primarily used in hidden layers and represents a nonlinear transformation that retains positive values as they are, while converting negative values to zero. This characteristic ensures that positive values are outputted as themselves, and negative values result in zero. Unlike certain other activation functions, ReLU does not converge to a specific positive value, and since its output range is extensive and positive values return themselves, it is not affected by the vanishing gradient issue in deep learning neural networks.

Additionally, because the formula for ReLU returns zero for negatives and the original value for positives, it typically allows for much faster learning rates compared to other activation functions. When employing stochastic gradient descent, it has been recorded that the convergence speed of ReLU can be nearly six times faster than that of other functions. The gradient remains constant at 1, thus enabling very rapid weight updates.

In this research, the objective is to identify potential shortcomings of the basic ReLU activation function and, based on these findings, attempt parameter optimization to enhance performance [9-10].

3.3 Adam

Adaptive Moment Estimation(Adam) is one of the optimization techniques in deep learning, combining the advantages of both Momentum and RMSProp algorithms. It determines both the direction and size of learning, which is why it's among the most commonly applied optimization techniques in deep learning. Recent developments include variations like RAdam and other modifications of Adam that demonstrate improved performance [11]. The formula for Adam is provided in equation (2).

$$\begin{gathered} m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \\ v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot {g_t}^2 \\ \hat{m}_t = \frac{m_t}{1-{\beta_1}^t} \quad \hat{v}_t = \frac{v_t}{1-{\beta_2}^t} \\ \theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon} \cdot \hat{m}_t \end{gathered} \tag{2}$$

where mt is the mean slope of the first-order momentum at time t. Vt is the mean squared slope at time t, where is the corrected first-order momentum and is the corrected second-order momentum. β1 and β2 are the decay rates, respectively. η is the learning rate (Ir). ϵ is a constant for stability. gt is the gradient and θt is the weight.

In the Adam algorithm, mt represents the moment at time t, which is the average of the gradients. Vt indicates the second moment at time t, which is the mean squared gradient. The terms $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected first and second moments, respectively.

β1 and β2 are the decay rates for these moments.

η is the learning rate (Ir), and ϵ is a constant added for numerical stability.

gt represents the gradient, and θt denotes the weights.

3.4 Tuning ReLU Parameters for Performance Improvement

For the experiments, the basic ReLU activation function was utilized initially, and parameters suc h as the dimension sizes and the Ir values were adjusted. The first dimension comprises the linear l ayers that make up the input and hidden layers, which receive and transform the input data's charac teristics. The second dimension involves the linear layers between the hidden and output layers, serv ing to receive the output from the hidden layers and compute the probabilities for each class.

In Table 1, considering the lower performance specifications of the hardware, the linear layers we re fixed at a 2-Layer setup, and various parameters including dimension sizes and Ir values were sy stematically adjusted during the experiments.

Table 1. Layer Network + Relu function-2

condition	Layer Network + Relu function-2							
dimension	10 - 50	10 - 40	10 - 35	10 -30				10 - 20
Ir	0.001	0.01		0.001	0.01	0.1	1	0.1
Precision	69.75%	93.77%	88.22%	63.83%	88.51%	93.28%	96.70%	93.21%

When using the ReLU activation function, the most improved accuracy results were obtained with dimension sizes of 30 - 10 and an Ir value of 1. In Table 2, experiments were conducted with the linear layers fixed at a 3-Layer configuration, and parameters such as dimension sizes and Ir values were systematically adjusted.

Table 2. Layer Network + Relu function-3

condition	Layer Network + Relu function-3				
dimension	10 - 30 - 40	10 - 20 - 50	10 - 30 - 50		10 - 30 - 45
Ir	0.001		0.01	0.1	0.01
Precision	33.13%	32.87%	88.05%	95.09%	87.89%

Unlike the configuration with 2-Layer linear layers, it was observed that the 3-Layer setup resulted in relatively lower accuracy values. However, within the 3-Layer configuration, dimension sizes of 10 - 30 - 50 achieved a significant accuracy of 95.09%. Considering the lower performance specifications of the device, fixing the configuration to 2-Layer rather than 3-Layer can yield improved results in terms of both accuracy and device performance.

3.5 Tuning Parameters for the 3-Layer ReLU + Adam Combined Model

In Table 3, the linear layers were fixed at a 3-Layer configuration, and the results of tuning exp eriments for the combined model of the ReLU and Adam models are presented.

Table 3. Layer Network + Relu function-3 + Adam

condition	Layer Network + Relu function-3 + Adam		
dimension	50 - 30 - 10	60 - 30 - 10	60 - 40 - 10
lr	0.001		
Precision	96.98%	96.99%	97.07%

In the experiments, when the Adam model was incorporated, it was possible to achieve improved accuracy results compared to using only the ReLU activation function. This study confirms that inc reasing the dimension sizes when combining with the Adam model can lead to enhanced performan ce outcomes. In the 3-Layer ReLU configuration with added Adam, dimension sizes of 60 - 40 - 1 0 and an Ir of 0.001 resulted in a significant accuracy of 97.07%.

4. Conclusion and Future Work

This study showed that enhancements in accuracy can be realized through the use of the ReLU activation function along with adjustments to parameters like layer size, dimension size, and the Ir value. For models applying only ReLU, the optimal parameters were found to be dimension sizes of 30 - 10 and an Ir value of 1, which yielded the highest accuracy. In the case of models incorpora ting the Adam optimizer, the highest accuracy was achieved with dimension sizes of 60 - 40 - 10 and an Ir of 0.001.

Given the rapid advancements in artificial intelligence and technology, optimization of functions s uch as ReLU and Adam will continue to evolve. Future research should focus on developing optimi zed models that can ensure sufficient accuracy on lower-performance devices in the fast-evolving lan dscape of AI technology.

References

[1] J. Chen, G. Serpen, et al., "Artificial Intelligence and Robotics for COVID-19: Applications and Innovations," IEEE Access, vol. 9, pp. 46145-46153, Oct. 2021.
DOI: [10.1109/ACCESS.2021.3102599]

[2] E. H. Lundquist, K. Walch, "AI in daily life: What every consumer needs to know," Deloitte Insights, vol. 6, no. 2, pp. 55-63, Feb. 2020.

[3] R.-S. Liu, B.-J. Ho, et al., "AI Benchmark: Running Deep Neural Networks on a Cluster of Mobile Phones," IEEE Transactions on Mobile Computing, vol. 19, no. 9, pp. 2102-2114, Sep. 2020.
DOI: [10.1109/TMC.2019.2912548]

[4] Jia-Bin Huang, Abhishek Singh, Narendra Ahuja, "Parametric Rectified Linear Unit for Deep Convolutional Neural Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 38, No. 6, pp. 1134-1141, Jun. 2016

[5] Djork-Arné Clevert, Thomas Unterthiner, Sepp Hochreiter, "Exponential Linear Units for Deep Convolutional Neural Networks," International Conference on Learning Representations (ICLR), May 2016

[6] G. Cybenko, "Multilayer feedforward networks are universal approximators," Neural Networks, vol. 2, no. 4, pp. 359-366, 1989.
DOI: [10.1016/0893-6080(89)90020-8]

[7] Y. LeCun, Y. Bengio, G. Hinton, "Deep Learning," Nature, vol. 521, no. 7553, pp. 436-444, May 2015.
DOI: [10.1038/nature14539]

[8] A. Krizhevsky, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," [Accessed: Apr. 08, 2022].
Available: [https://github.com/zalandoresearch/fashion-mnist]

[9] V. Nair, G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," Proceedings of the 27th International Conference on International Conference on Machine Learning, pp. 807-814, Jun. 2010. [Accessed: Apr. 08, 2022].
Available: [http://www.cs.toronto.edu/~hinton/absps/reluICML.pdf]

[10] W. Wen, "Comparison of Rectified Linear Units (ReLU) and S-shaped Rectified Linear Activation Units (SReLU) in Sparse Coding," arXiv, [Accessed: Apr. 08, 2022].
Available: [https://arxiv.org/abs/1709.07417]

[11] D. P. Kingma, J. Ba, "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION," arXiv, [Accessed: A pr. 08, 2022].
Available: [https://arxiv.org/abs/1412.6980]