MEANS (Media Informasi Analisa dan Sistem)
p-ISSN: 2548-6985, e-ISSN:2599-3089

Performance Analysis of Neural Networks With Backpropagation on Binary and Multi-Class Data Classification

[1] Abdul Tahir
Akademi Teknik Soroako, Jl. Sumantri Brojonegoro No.1, Sorowako, Kec. Nuha, Kabupaten Luwu Timur, Sulawesi Selatan, Indonesia
E-Mail: abdultahir0101@gmail.com

[2] Irdam
Akademi Teknik Soroako, Jl. Sumantri Brojonegoro No.1, Sorowako, Kec. Nuha, Kabupaten Luwu Timur, Sulawesi Selatan, Indonesia
E-Mail: Irdam@ats-sorowako.ac.id

[3] Sirama
Akademi Teknik Soroako, Jl. Sumantri Brojonegoro No.1, Sorowako, Kec. Nuha, Kabupaten Luwu Timur, Sulawesi Selatan, Indonesia
E-Mail: Sirama@ats-sorowako.ac.id

Neural networks represent a widely adopted paradigm within the domain of machine learning, employed for a multitude of classification endeavors, encompassing image recognition and natural language processing. This investigation seeks to elucidate the influence of varying neuron quantities in hidden layers on the efficacy of neural networks in both binary and multi-class classification endeavors. The research utilizes a dataset procured from images depicting characters and digits, which were transformed into binary format via a thresholding methodology. The neural network architectures comprise one and two hidden layers, which are trained employing the backpropagation algorithm in conjunction with the Adam optimizer. The evaluation of the models is conducted through metrics such as accuracy, loss curves, and confusion matrices. Findings reveal that the configuration featuring two hidden layers with 40 sampai 99 neurons achieves the pinnacle accuracy of 99.64 percent alongside optimal loss stability. Furthermore, models incorporating a single hidden layer exhibited commendable accuracy, thereby indicating that a reduced number of neurons can proficiently encapsulate data complexity in less demanding tasks. This research underscores the criticality of selecting suitable neural network configurations contingent upon data complexity and classification objectives, while advocating for further investigation into regularization strategies to enhance performance.

Keywords: Neurons ,Classification , Optimization , Accuracy , Regularization

INTRODUCTION

Neural networks have emerged as one of the preeminent methodologies in machine learning, particularly within various classification endeavors, including but not limited to image recognition, natural language processing, and time series forecasting [1]. The proficiency of these algorithms in discerning intricate patterns within datasets renders them an invaluable asset across diverse domains, encompassing medical applications, security measures, and image analysis. Over the past decade, neural networks have undergone significant advancements, both in terms of their architectural designs and computational prowess, thereby enabling this methodology to tackle an increasingly sophisticated array of classification challenges [2]. A notable progression in this technological domain is the deployment of Convolutional Neural Networks (CNNs) for the purpose of image classification, exemplified by the introduction of AlexNet, which has demonstrated substantial efficacy in managing extensive image datasets within the context of ImageNet `competitions. This milestone not only signifies a pivotal transition in the realm of image processing but also paves the way for the utilization of CNNs in a plethora of more intricate tasks, such as fraud detection and medical diagnosis, which frequently necessitate modifications to neural network architectures, particularly in the contexts of binary and multi-class classifications [3][4]

Although advancements in neural network technology have conferred numerous advantages, the optimization of the architecture and configuration of such models continues to present a significant challenge. The determination of the appropriate number of neurons within the hidden layers, for instance, exerts a considerable impact on the performance of the model; an insufficient number of neurons may impede the model's capacity to discern intricate patterns, whereas an excessive number of neurons may induce overfitting, thereby diminishing the model's ability to generalize to novel data [5]. This optimization assumes heightened significance when neural networks are deployed in diverse, large-scale datasets, which are frequently encountered in practical applications. Consequently, this investigation seeks to examine the optimal configuration of neural networks, with particular emphasis on the determination of neuron quantity in the hidden layers, to attain a harmonious balance between accuracy and model complexity. Thus, this research aspires to furnish pragmatic guidance in establishing efficient configurations for both binary and multi-class classification tasks, as well as to bolster the advancement of neural networks that are more responsive to the varying requirements of different applications [6].

The training procedure of neural networks predominantly utilizes a backpropagation algorithm, which facilitates the modification of network weights in accordance with output discrepancies, thereby enabling the model to discern patterns from the input data [7][8]. While this algorithm constitutes an essential element in the training of neural networks, its enhancement continues to be a prominent area of scholarly inquiry. In the context of training deep neural networks, the pursuit of optimal configurations, including the quantity of neurons and hidden layers, presents a significant challenge, particularly in attaining an equilibrium between precision and computational efficiency [9]. Additional complications, such as overfitting, frequently emerge when the quantity of neurons or layers is excessively high, which can impair the model's capacity to generalize to novel data.

The selection of an appropriate quantity of neurons within the hidden layers can enhance the model's capacity to discern more intricate patterns; however, there exists a threshold beyond which performance begins to deteriorate, concomitantly increasing the propensity for overfitting. To mitigate this issue, regularization methodologies such as dropout and L2 regularization are employed to attenuate network complexity, ensure stability throughout the training process, and augment the model's generalization capacity. Furthermore, the implementation of batch normalization is recognized to enhance training stability, while weight decay strategies have demonstrated efficacy in bolstering model generalization, particularly in the context of complex datasets [10][11].

Furthermore, the choice of activation functions, such as ReLU, Sigmoid, and Tanh, exerts a substantial impact on the efficacy of models, particularly in deep networks necessitating advanced learning techniques. The ReLU activation function, for instance, has demonstrated its superiority in mitigating gradient vanishing issues, rendering it a favored option in the advancement of contemporary CNN architectures [3]. This activation function is also crucial for maintaining training stability, especially in neural networks characterized by a multitude of neuron layers. Additionally, empirical evidence indicates that appropriate weight initialization can significantly influence the convergence of training processes, particularly within expansive and intricate neural networks [5][12].

In the domain of neural network training, optimization algorithms such as Stochastic Gradient Descent (SGD), Adam, and RMSprop are pivotal in enhancing training efficacy. The algorithm demonstrates commendable performance across a diverse array of data types, wherein SGD yields consistent outcomes regarding generalization, while Adam and RMSprop facilitate expedited convergence, thereby being particularly advantageous for extensive and intricate datasets [13]. The investigation of various hidden layer configurations, encompassing modifications in the neuron count, is anticipated to unveil the optimal arrangement that reconciles accuracy with complexity [14]. This research endeavors to furnish insights into the determination of the optimal number of neurons pertinent to distinct classification tasks, underscoring the significance of configurations tailored to the complexity of the data as well as the objectives of classification [15].

METHOD

A. Dataset Description

The study used a dataset extracted from letters and numbers, where each image was converted to a black-and-white format with a resolution of 10 x 8 pixels. To obtain a more consistent and noise-free binary representation, a thresholding technique is applied to the image before the conversion process into binary data. The thresholding technique is used to define the pixel intensity limit so that pixels with a certain intensity value are classified as part of the object or background that is important in maintaining the quality of binary data [16][17].

B. Data Pre-Processing Process

In this study, a global thresholding method was used, where each pixel in the image is compared to a fixed threshold value of T. The equation used is as follows: [16]

$$f(x,y) = \left\{\begin{matrix} 1, & jika\ I(x,y) \geq T \\ 0, & jika\ I(x,y) < T, \end{matrix}\right\}$$

Where:

- $f(x,y)$: *the binary value of the pixel at position* (x,y)
- $I(x,y)$: *the intensity of the pixel at position* (x,y) *in the original image*
- T : *the threshold value that has been determined*

With this method, pixels that have an intensity above the threshold T are considered part of the object (value 1), while those below the threshold are considered background (value 0) as shown in Figure 1. Here, the T threshold is chosen experimentally, considering its effectiveness in increasing contrast and reducing the noise that may appear on binary data [18].

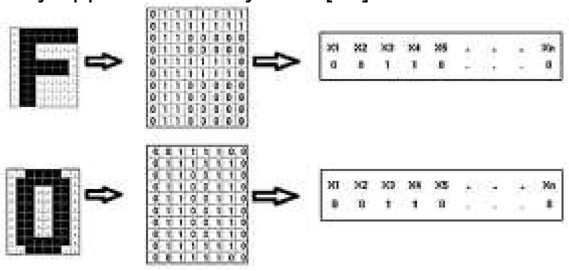

Fig.1. illustration of the dataset formation process

After the thresholding process, the image is converted into a binary vector consisting of 80 features for each character. The dataset was then divided into training data (80%) and test data (20%), where each character in the image was classified into one of 11 output classes that referred to the letters 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and the letter F. This dataset division follows standard best practices to ensure that the model can generalize well on data that has never been seen before [19].

C. Model Neural Network

The neural network model used in this study was trained using a backpropagation algorithm. This study adopts two main architectural approaches: one hidden layer and two hidden layers. With the addition of thresholding, it is hoped that the binary representation of the image will be more consistent and help improve the accuracy of the model in classifying characters. Testing neural networks using the Sigmoid Binary activation function [7]

Models with One Hidden Layer

The model consists of one hidden layer with the following equation

$$z^{(1)} = \sigma\left(W^{(1)}x + b^{(1)}\right)$$

$$y = \sigma\left(W^{(2)}z + b^{(2)}\right)$$

Where:

- x : 80×1 *input vector.*
- $W^{(1)}$: *the weight matrix between the input and the hidden layer and size* $m \times 80$.
- $b^{(1)}$: *a bias vector for a hidden layer measuring* $m \times 1$.
- $\sigma(z) = \frac{1}{1+e^{-z}}$ *is the activation function of binary sigmoid.*
- $z^{(1)}$: *the output of the hidden layer is* $m \times 1$.
- $W^{(2)}$: *weight matrix between the hidden layer, output layer of size* $11 \times m$.
- $b^{(2)}$: *the bias vector for the output layer is* 11×1.
- y : *the output prediction vector is* 11×1.

Models with Two Hidden Layers

This model consists of two hidden layers. Here is a mathematical equation that describes the flow of data in this network:

$$\mathbf{z}^{(1)} = \sigma\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right)$$

$$\mathbf{z}^{(2)} = \sigma\left(\mathbf{W}^{(2)}\mathbf{z}^{(1)} + \mathbf{b}^{(2)}\right)$$

$$\mathbf{y} = \sigma\left(\mathbf{W}^{(3)}\mathbf{z}^{(2)} + \mathbf{b}^{(3)}\right)$$

where :

- $z^{(1)}$: *the output of the first hidden layer is* $m \times 1$.
- $W^{(2)}$: *the weight matrix between the first/second hidden layers of* $m' \times m$ *size.*
- $b^{(2)}$: *the bias vector for the second hidden layer is* $m' \times 1$.
- $z^{(2)}$: *The output of the second hidden layer is* $m' \times 1$.
- $W^{(3)}$: *weight matrix between second hidden layer, output layer of size* $11 \times m'$.
- $b^{(3)}$: *The bias vector for the output layer is* 11×1.
- *The activation function σ the same as in the single − layer hidden layer model.*
- y : *The output prediction vector is* 11×1.

D. ANN Model Working Process

The ANN Model Work Process begins by entering image data that has gone through the thresholding stage and converted into a binary vector measuring 80 features per character. The x vector is the input to the neural network. Each feature in this vector represents a specific characteristic of the image that has been processed, such as pixel intensity or texture features.

Model with One Hidden Layer

In the hidden layer, the input vector x is processed by multiplying the first weight matrix W(1) and adding the bias vector b(1), resulting in the value of z(1). Furthermore, binary sigmoid activation functions σ applied to these values to add elements of non-linearity, so that the network can learn more complex relationships between features [20].

On the output layer, the result of the hidden layer z(1) is multiplied by the second weight matrix W(2) and added by the bias vector b(2), resulting in a predicted value of y. The binary sigmoid function σ again applied to generate the probabilities of each class, thus allowing the network to determine the final prediction for each character of the total 11 existing classes [21].

Models with Two Hidden Layers

In the first hidden layer, the input vector x is multiplied by the first weight matrix W(1) and added with the bias vector b(1) yielding the value z(1). These values are then processed through the binary sigmoid activation function σ to add non-linearity elements in the network.

Furthermore, in the second hidden layer, the output of the first layer z(1) is multiplied by the weight matrix W(2) and added by the bias vector b(2), resulting in the value of z(2). The binary sigmoid function σ again applied to add another layer of non-linearity [22].

On the output layer, the result of the second layer z(2) is multiplied by the final weight matrix W(3) and added by the bias vector b(3), resulting in the final prediction y. The binary sigmoid function σ applied to generate probabilities for each class in the 11 output classes, which is then used to determine the predicted results of the network [2].

Output and Optimization and Training

The prediction vector y is the result of a classification of characters categorized into one of 11 output classes. The binary sigmoid activation function on the output layer results in a value between 0 and 1 which is interpreted as the probability of each class. The class with the highest probability is chosen as the final prediction for each classified character [11].

In the Optimization and Training stage, the model is trained using the Adam optimization algorithm (Adaptive Moment Estimation) with the default learning rate value. Adam's algorithm is known to be able to achieve convergence faster and more stable, especially for complex and varied data. Each model configuration is trained for 200 epochs to give the network enough time to learn the patterns in the data, according to the training recommendations of models with varying numbers of neurons [23].

During the training process, a backpropagation algorithm is applied to update the network weights and biases based on the predicted errors measured using the cross-entropy binary loss function for multi-class classification with Softmax.

This training process includes several key steps [24]

- Forward Pass: Input data is processed through the network to generate a prediction of y output.
- Loss Calculation: At this stage, the error between the y prediction and the actual label y′ is calculated using the cross-entropy binary loss function.

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{11}\left[y'_{ij}\log(y_{ij}) + (1 - y'_{ij})\log(1 - y_{ij})\right]$$

- Backward Pass: The gradient of the loss function is calculated against each weight and bias in the network using chain rules.
- Parameter Update: Weights and biases are updated by decreasing the calculated gradient, multiplied by the learning rate, to reduce the loss value.

In addition, regularization techniques such as dropout and L2 regularization are applied to improve the generalization ability of the model. This regularization helps prevent overfitting by reducing the complexity of the model, so that the model does not adapt too much to the training data.

Architecture Configuration Testing

Testing on a model with a single hidden layer was carried out by testing variations in the number of neurons in configurations of 18, 30, 70, 75, and 99 neurons. This architecture is designed to explore how changes in the number of neurons can affect the accuracy of the model, providing insight into the effectiveness of simple network structures in data processing.

Meanwhile, testing on a model with two hidden layers involved variations in the number of neurons in configurations of 30-30, 30-40, 40-30, 40-40, and 40-99 neurons. The addition of a second hidden layer aims to evaluate its impact on the complexity and accuracy of the model, as well as to test the potential for network performance improvements in classifying binary data with deeper structures. Figure 2 shows the architectural structure of each approach, both for one hidden layer and two hidden layers [25].

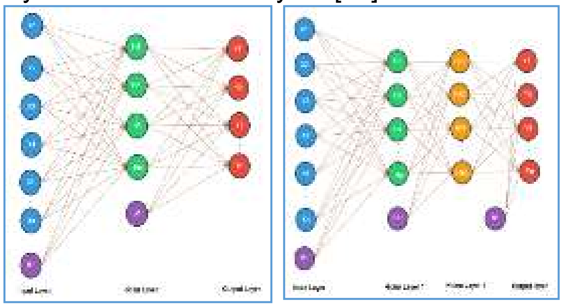
(a) (b)

Fig. 2. Architecture with 1 hiden layer (a) and 2 hiden layers (b)

Model Performance Evaluation

Model performance evaluation is carried out using accuracy, precision, recall, and F1-score metrics to get a comprehensive picture of the model's performance in classifying characters. These metrics help in assessing how well the model classifies each output class as well as the balance between the model's precision and sensitivity. In addition, the confusion matrix is used to analyze classification errors between classes, and the ROC curve is used to evaluate the trade-off between the true positive rate and the false positive rate [26].

The results of the two architectural configurations are compared to determine the optimal configuration in the context of the data and classification tasks being studied. This analysis includes a comparison of accuracy, model complexity, as well as training and inference times, thus providing guidance for the determination of the number of neurons and the structure of the hidden layers efficiently for various data classification tasks [27].

RESULT AND DISCUSSION

To get the best network model with optimal accuracy, a number of experiments were carried out with 2 methods as described in the research method. The following is a discussion of each method carried out on each variation in the number of hidden layer neurons.

1. First Method: One Hidden Layer

a. 18 and 30 Neurons in the Hidden Layer

In tests with the number of 18 and 30 neurons in the hidden layer, the results showed a significant difference in the performance of the model. The model with 18 neurons produced a fairly good accuracy of 98.93%, although there were fluctuations in the loss curve that reflected instability during training. This indicates that models with fewer neurons may have difficulty finding complex patterns in the data. However, when the number of neurons is increased to 30, there is a significant increase in the accuracy and stability of the loss curve. The model with 30 neurons achieved the highest accuracy of 99.64%, indicating that the addition of neurons had a positive impact on the model's ability to capture important features from the data. The results of the loss curve visualization for both configurations can be seen in Figure 3, which illustrates the stability and performance comparison between the model with 18 and 30 neurons.

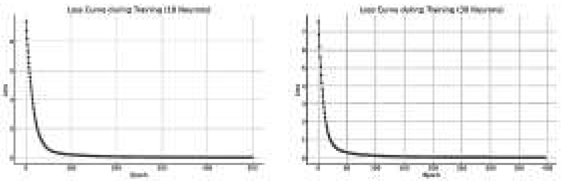

Fig.3. Visualization of the Loss Curve with 18 and 30 Neurons

b. 70 and 5 Neurons in the Hidden Layer

In a test with 70 neurons in the hidden layer, the results showed that the model achieved the highest accuracy among the configurations tested, with an accuracy value of 99.64%. This indicates that the addition of neurons significantly improves the model's ability to capture the complexity of the data and existing patterns. A visualization of the loss curve for this model can be seen in Figure 1, which shows good stability during the training process.

However, when the number of neurons was increased to 75, even though the model had more neurons, the evaluation results showed a slight decrease in accuracy to 98.93%. This decline may be due to the phenomenon of overfitting, where the model is too complex and begins to adjust to the noise in the training data, reducing its ability to generalize on new data. Figure 4 also shows a visualization of the loss curve for a model with 75 neurons, which shows that although there are more neurons, the stability of the loss curve is not as good as that of a model with 70 neurons.

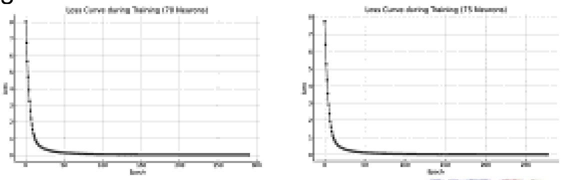

Fig. 4. Visualization of the Loss Curve with 70 and 75 Neurons

c. 99 Neuron di Hidden Layer:

The results of the experiment in the training process with this model can be seen in figure 5. The addition of more neurons up to 99 indicates excellent accuracy and a stable loss curve, indicating that the model is able to capture the complexity of the data very well. In the evaluation process, Accuracy: 99.64%, Loss Curve Visualization is as follows:

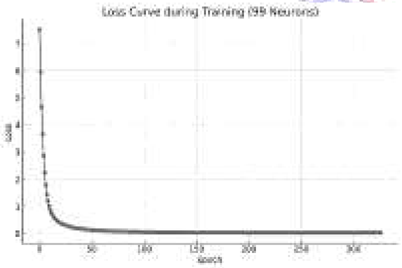

Fig. 5. Visualization of the Loss Curve with 99 Neurons

2. Method Two: Two Hidden Layers

a. 30 – 30 and 30-40 Neurons in Hidden Layers

In the second method using two hidden layers, the results of the experiment for a model with a configuration of 30-30 neurons showed an accuracy of 98.57%. Although this accuracy is slightly lower compared to the single-layer hidden layer model, it still performs well in classification. A visualization of the loss curve for this model can be seen in Figure 6, which shows the stability of the curve despite small fluctuations.

Furthermore, in a configuration of 30-40 neurons, the addition of neurons in the second hidden layer gives better results. The model achieved 99.29% accuracy, signaling that the increase in the number of neurons in the second hidden layer had a positive impact on the model's ability to capture data complexity. A more stable loss curve is also seen in the visualization shown in Figure 6, which reflects the improvement in performance during the training process. These results show the potential of the two-layer structure to improve classification accuracy when the number of neurons is adjusted appropriately.

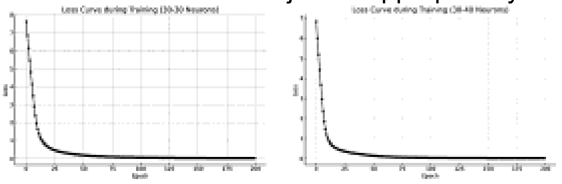

Fig. 6. Loss Curve Visualization with 2 hidden layers 30 - 30 and 30-40 Neuron

b. 40-30 and 40-40 Neurons in Hidden Layers

In a test with a configuration of 40-30 neurons in the hidden layers, the model showed a good accuracy of 99.29%. These results reflect the model's ability to capture important patterns in the data, with the stability of the loss curve seen in Figure 7. Although the number of neurons in the second hidden layer is less, the model manages to maintain good performance, suggesting that the proper arrangement of neurons can contribute to the model's effectiveness in classification.

Furthermore, in a model with a configuration of 40-40 neurons in both hidden layers, the experimental results showed excellent performance with the same accuracy of 99.29%. The loss curve for this model shows optimal stability, reflecting the model's ability to adapt to data complexity without showing any signs of overfitting. A visualization of the loss curve for these two configurations can be seen in Figure 7, confirming that the structure of two hidden layers with a balanced number of neurons provides consistent and reliable results in the classification task.

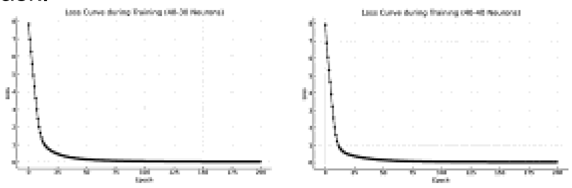

Fig. 7. Loss Curve Visualization with 2 hidden layers 40 – 30 and 40 – 40 Neuron

c. 40-99 Neuron di Hidden Layers

The results of the experiment in the training process with this model can be seen in figure 8. This configuration shows the best performance with the highest accuracy and a very stable loss curve, indicating that the addition of neurons in the second hidden layer significantly improves the model's capabilities. In the evaluation process, Accuracy: 99.64%, Loss Curve Visualization is as follows:

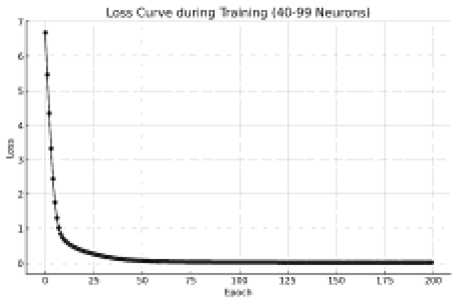

Fig. 8. Loss Curve Visualization with 2 hidden layers 40 - 99 Neurons

Figure 9 below shows the test accuracy graph for each variation in the number of neurons in both methods applied.

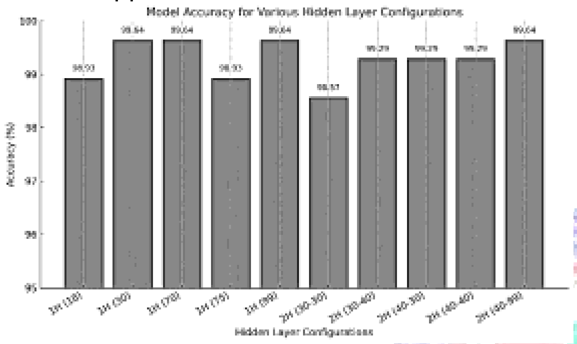

Fig. 9. Test accuracy graph with 1 and 2 layers hidden methods

From the graph above, the results of the analysis show that adding a second hidden layer can improve the stability and performance of the model. The configuration of 40-99 neurons in the hidden layers provides the highest accuracy, suggesting that having one hidden layer with a larger number of neurons can capture more complexity in the data. However, models with one hidden layer and 70 or 99 neurons also provide very high accuracy, suggesting that adding more neurons in one hidden layer is sufficient in most cases.

CONCLUSION

This research demonstrates that the most effective architecture for neural networks tasked with binary and multi-class classification is realized through the implementation of two hidden layers, particularly within a configuration of 40 to 99 neurons, which yields a peak accuracy of 99.64% and remarkable stability in the loss curve. The incorporation of a secondary hidden layer, along with an increased neuron count in this layer, substantially enhances the model's capacity to encompass the intricacies of the data. Nevertheless, models characterized by a singular hidden layer and an adequate number of neurons (for instance, 70 or 99 neurons) also exhibit commendable accuracy performance, presenting them as a more straightforward and efficient alternative for classification tasks of lesser complexity. Conversely, the investigation uncovered signs of overfitting in models possessing an excessively high neuron count in the absence of regularization, thereby underscoring the significance of optimal neuron quantities and supplementary regularization methodologies. These findings indicate that the choice of hidden layer configurations ought to be calibrated according to the data complexity and classification goals, and advocate for the pursuit of a broader array of regularization techniques and datasets in future inquiries to assess the robustness of the findings across a wider classification landscape.

REFERENCES

[1] A. Di Piazza, M. C. Di Piazza, G. La Tona, and M. Luna, “An artificial neural network-based forecasting model of energy-related time series for electrical grid management,” Math Comput Simul, vol. 184, pp. 294–305, Jun. 2021, doi: 10.1016/j.matcom.2020.05.010.

[2] E. Ayyildiz, M. Erdogan, and A. Taskin, “Forecasting COVID-19 recovered cases with Artificial Neural Networks to enable designing an effective blood supply chain,” Comput Biol Med, vol. 139, p. 105029, Dec. 2021, doi: 10.1016/j.compbiomed.2021.105029.

[3] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, “Activation functions in deep learning: A comprehensive survey and benchmark,” Neurocomputing, vol. 503, pp. 92–108, Sep. 2022, doi: 10.1016/j.neucom.2022.06.111.

[4] Y. Luo, H.-H. Tseng, S. Cui, L. Wei, R. K. Ten Haken, and I. El Naqa, “Balancing accuracy and interpretability of machine learning approaches for radiation treatment outcomes modeling,” BJR|Open, vol. 1, no. 1, Jul. 2019, doi: 10.1259/bjro.20190021.

[5] M. V. Narkhede, P. P. Bartakke, and M. S. Sutaone, “A review on weight initialization strategies for neural networks,” Artif Intell Rev, vol. 55, no. 1, pp. 291–322, Jan. 2022, doi: 10.1007/s10462-021-10033-z.

[6] R. Abdulkadirov, P. Lyakhov, and N. Nagornov, “Survey of Optimization Algorithms in Modern Neural Networks,” Mathematics, vol. 11, no. 11, p. 2466, May 2023, doi: 10.3390/math11112466.

[7] M. Dampfhoffer, T. Mesquida, A. Valentian, and L. Anghel, “Backpropagation-Based Learning Techniques for Deep Spiking Neural Networks: A Survey,” IEEE Trans Neural Netw Learn Syst, vol. 35, no. 9, pp. 11906–11921, Sep. 2024, doi: 10.1109/TNNLS.2023.3263008.

[8] S. Chakravarty, M. H. Tanveer, R. C. Voicu, M. Banerjee, and G. Mahdi, “Backpropagation Techniques in SNN and Application in Image Segmentation,” in SoutheastCon 2024, IEEE, Mar. 2024, pp. 1475–1481. doi: 10.1109/SoutheastCon52093.2024.10500286.

[9] T. Liu, C. Zhang, and D. Li, "Reducing Overfitting In Deep Neural Networks By Intra-class Decorrelation," in 2023 International Seminar on Computer Science and Engineering Technology (SCSET), IEEE, Apr. 2023, pp. 200–203. doi: 10.1109/SCSET58950.2023.00052.

[10] S.-H. Lyu, L. Wang, and Z.-H. Zhou, "Improving generalization of deep neural networks by leveraging margin distribution," Neural Networks, vol. 151, pp. 48–60, Jul. 2022, doi: 10.1016/j.neunet.2022.03.019.

[11] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," Commun ACM, vol. 64, no. 3, pp. 107–115, Mar. 2021, doi: 10.1145/3446776.

[12] Q. Chen, W. Hao, and J. He, "A weight initialization based on the linear product structure for neural networks," Appl Math Comput, vol. 415, p. 126722, Feb. 2022, doi: 10.1016/j.amc.2021.126722.

[13] Shem L. Gonzales, "Enhancing Image Classification Performance: A Comparative Analysis of Optimization Algorithms," International Journal of Advanced Research in Science, Communication and Technology, pp. 641–645, Jun. 2023, doi: 10.48175/IJARSCT-11918.

[14] H. T. Sihotang, M. Albert, F. Riandari, and L. Rendell, "Efficient optimization algorithms for various machine learning tasks, including classification, regression, and clustering," Idea: Future Research, vol. 1, no. 1, pp. 14–24, Jan. 2023, doi: 10.35335/idea.v1i1.3.

[15] D. Elhani, A. C. Megherbi, A. Zitouni, F. Dornaika, S. Sbaa, and A. Taleb-Ahmed, "Optimizing convolutional neural networks architecture using a modified particle swarm optimization for image classification," Expert Syst Appl, vol. 229, p. 120411, Nov. 2023, doi: 10.1016/j.eswa.2023.120411.

[16] L. Abualigah, K. H. Almotairi, and M. A. Elaziz, "Multilevel thresholding image segmentation using meta-heuristic optimization algorithms: comparative analysis, open challenges and new trends," Applied Intelligence, vol. 53, no. 10, pp. 11654–11704, May 2023, doi: 10.1007/s10489-022-04064-4.

[17] Y. Tian, "Artificial Intelligence Image Recognition Method Based on Convolutional Neural Network Algorithm," IEEE Access, vol. 8, pp. 125731–125744, 2020, doi: 10.1109/ACCESS.2020.3006097.

[18] F. E. Sadeq and Z. T. M. Al-Ta'i, "Comparison Between Face and Gait Human Recognition Using Enhanced Convolutional Neural Network," Journal of Applied Engineering and Technological Science (JAETS), vol. 5, no. 1, pp. 18–30, Dec. 2023, doi: 10.37385/jaets.v5i1.2806.

[19] D. AL Kafaf, N. N. Thamir, and S. S. AL-Hadithy, "Malaria Disease Prediction Based on Convolutional Neural Networks," Journal of Applied Engineering and Technological Science (JAETS), vol. 5, no. 2, pp. 1165–1181, Jun. 2024, doi: 10.37385/jaets.v5i2.3947.

[20] A. Purnomo and H. Tjandrasa, "IMPROVED DEEP LEARNING ARCHITECTURE WITH BATCH NORMALIZATION FOR EEG SIGNAL PROCESSING," JUTI: Jurnal Ilmiah Teknologi Informasi, vol. 19, no. 1, p. 19, Jan. 2021, doi: 10.12962/j24068535.v19i1.a1023.

[21] A. B. Çolak, T. Güzel, O. Yıldız, and M. Özer, "An experimental study on determination of the shottky diode current-voltage characteristic depending on temperature with artificial neural network," Physica B Condens Matter, vol. 608, p. 412852, May 2021, doi: 10.1016/j.physb.2021.412852.

[22] A. Akhgar, D. Toghraie, N. Sina, and M. Afrand, "Developing dissimilar artificial neural networks (ANNs) to prediction the thermal conductivity of MWCNT-TiO2/Water-ethylene glycol hybrid nanofluid," Powder Technol, vol. 355, pp. 602–610, Oct. 2019, doi: 10.1016/j.powtec.2019.07.086.

[23] C. Singh, "Machine Learning in Pattern Recognition," European Journal of Engineering and Technology Research, vol. 8, no. 2, pp. 63–68, Apr. 2023, doi: 10.24018/ejeng.2023.8.2.3025.

[24] M. Dialameh, A. Hamzeh, H. Rahmani, S. Dialameh, and H. J. Kwon, "DL-Reg: A deep learning regularization technique using linear regression," Expert Syst Appl, vol. 247, p. 123182, Aug. 2024, doi: 10.1016/j.eswa.2024.123182.

[25] P. Freire, E. Manuylovich, J. E. Prilepsky, and S. K. Turitsyn, "Artificial neural networks for photonic applications—from algorithms to implementation: tutorial," Adv Opt Photonics, vol. 15, no. 3, p. 739, Sep. 2023, doi: 10.1364/AOP.484119.

[26] A. Shafiq, A. Batur Çolak, T. Naz Sindhu, S. Ahmad Lone, A. Alsubie, and F. Jarad, "Comparative study of artificial neural network versus parametric method in COVID-19 data analysis," Results Phys, vol. 38, p. 105613, Jul. 2022, doi: 10.1016/j.rinp.2022.105613.

[27] Z. Zhou, C. Qiu, and Y. Zhang, "A comparative analysis of linear regression, neural networks and random forest regression for predicting air ozone employing soft sensor models," Sci Rep, vol. 13, no. 1, p. 22420, Dec. 2023, doi: 10.1038/s41598-023-49899-0.