Nigerian Research Journal of Engineering and Environmental Sciences 11(1) 2026 pp. 58-70
p ISSN: 2635-3342; e ISSN: 2635-3350

Nigerian Research Journal of Engineering and Environmental Sciences
Journal homepage: www.rjees.com

Original Research Article

Comparative Analysis of Hidden Layer Neuron Configurations for Prediction Performance in Neural Networks Trained Using Levenberg–Marquardt and Scaled Conjugate Gradient Algorithms

***[1]Onah, T.O., [1]Aka, C.C. and [2]Onah, B.C.**

[1]Department of Mechatronics Engineering, Faculty of Engineering, Enugu State University of Science and Technology, PMB 01660, Agbani, Enugu State, Nigeria.
[2]Department of Mechatronics Engineering, Faculty of Engineering, Institute of Management and Technology, PMB 01079, Independence Layout, Enugu, Enugu State, Nigeria.
*chikezie.aka@esut.edu.ng

http://doi.org/10.5281/zenodo.21046185

ARTICLE INFORMATION

Article history:
Received 08 Jan. 2026
Revised 06 Apr. 2026
Accepted 20 Apr. 2026
Available online 30 Jun. 2026

Keywords:
Error minimization
Hidden layer neurons
Levenberg-Marquardt
Neural networks
Scaled conjugate gradient

ABSTRACT

This study investigates the performance characteristics of neural network training using varying hidden layer neuron configurations with paired complementary values (2 and 50, 4 and 40, 6 and 30, 8 and 20), employing the Levenberg-Marquardt (LM) and Scaled Conjugate Gradient (SCG) algorithms. The research aims to determine optimal neuron configurations for minimizing prediction uncertainty and training errors. Experiments were conducted using MATLAB R2019a's Neural Network Fitting Tool with systematic variation of hidden layer neurons from the default value of 10. Results demonstrate that the Levenberg-Marquardt algorithm consistently achieves superior validation performance with lower mean squared error values compared to the Scaled Conjugate Gradient approach. For a hidden neuron configuration of 8, LM achieved a validation error of 0.00012831 at epoch 94, representing the optimal performance among tested configurations. The error histogram analysis reveals that configurations approaching the default target of 10 neurons (specifically 6, 8, and their complements) exhibit tighter error distributions centered on zero. Conversely, extreme neuron counts (2, 50) demonstrated increased error dispersion indicative of underfitting and overfitting, respectively. These findings provide practical guidelines for neural network architecture design in regression and function approximation tasks, with applications extending to medical diagnostics, industrial automation, and predictive systems.

1. INTRODUCTION

Artificial Neural Networks (ANNs) represent computational methodologies that enable machine learning, knowledge demonstration, and the application of learned knowledge to maximize output responses of complex systems (Chen et al., 2019). The architecture of artificial neural networks mimics

that of the human brain, with web-like connections between neuronal nodes. These networks have found extensive applications in pattern recognition, image processing, speech recognition, and classification tasks (Van Gerven and Bohte 2020).

The training of neural networks involves iterative adjustment of synaptic weights to minimize error functions, with various algorithms offering different trade-offs between convergence speed, computational efficiency, and accuracy. Among these, the Levenberg-Marquardt (LM) algorithm and the Scaled Conjugate Gradient (SCG) algorithm represent two prominent approaches with distinct characteristics (Shamshirband et al., 2019).

The Levenberg-Marquardt algorithm, developed by Levenberg in 1944 and Marquardt in 1963, provides a numerical method for minimization of non-linear functions using the least-squares approach (Kostryzhev et al., 2018). This method strikes a balance between gradient descent, which converges slowly but surely, and Newton's method, which converges rapidly at a minimum but may otherwise diverge (Li et al., 2020; Marugán et al., 2018). The scaled conjugate gradient method, as described by Møller (1993), determines the optimal step size at each iteration using a Levenberg-Marquardt approach rather than a line search, thereby increasing computational efficiency (Luo et al., 2018).

This study specifically addresses a broader investigation into neural network prediction for uncertainty minimization: training with hidden-layer neuron counts in paired complementary configurations of 2 and 50, 4 and 40, 6 and 30, and 8 and 20, using both the Levenberg–Marquardt and Scaled Conjugate Gradient algorithms. The aim is to systematically evaluate how different neuron configurations affect training performance, error minimization, and model generalization capabilities.

Understanding the relationship between hidden layer neuron counts and training algorithm performance is critical for practitioners designing neural network architectures. Insufficient neurons lead to underfitting, where the network cannot capture the underlying data patterns, while excessive neurons result in overfitting, where the network memorizes training data noise rather than learning generalizable patterns. This research provides empirical evidence to guide optimal network architecture selection.

2. MATERIALS AND METHODS

2.1. Experimental Setup

Experiments were conducted using MATLAB R2019a equipped with the Neural Network Fitting Tool (nftool) on a laptop computer. The neural network architecture employed was a two-layer feed-forward network with sigmoid hidden neurons and linear output neurons (fitnet). The default configuration utilized 10 hidden neurons with a 70% training, 15% validation, and 15% testing data split.

2.2. Hidden Neuron Configurations

The study systematically varied hidden layer neuron counts using paired complementary values designed to explore both underfitting (low neuron counts) and overfitting (high neuron counts) scenarios relative to the default configuration as presented in Table 1.

Table 1: Variation of the configuration pair of primary and complement neurons

Configuration pair	Primary neurons	Complement neurons	Rationale
Pair 1	2	50	Extreme underfitting vs severe overfitting
Pair 2	4	40	Moderate underfitting vs significant overfitting
Pair 3	6	30	Slight underfitting vs moderate overfitting
Pair 4	8	20	Near-optimal configurations

2.3. Training Algorithms

The Levenberg-Marquardt learning algorithm (trainlm): This algorithm addresses the limitations of computationally simpler methods like RPROP when solving complex boundary value problems

(BVPs). This classical optimization technique is renowned for its rapid convergence, particularly when minimizing sum-of-squared-error functions (Reidmiller and Braun, 2002). The algorithm builds upon the Newton-Raphson method, which uses a second-order Taylor series expansion to approximate the error function E(θ) with respect to neural network weights as in Equation 1 (Kecman, 2020).

$$E(\boldsymbol{\theta}) = E(\boldsymbol{\theta}_0) + \boldsymbol{g}^T(\boldsymbol{\theta} - \boldsymbol{\theta}_0) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T \mathbf{H}(\boldsymbol{\theta} - \boldsymbol{\theta}_0) + H.O.T. \quad (1)$$

where $\boldsymbol{g} = \nabla E(\boldsymbol{\theta})$ is the gradient vector and $\boldsymbol{H}(\boldsymbol{\theta})$ is the Hessian matrix containing second-order partial derivatives. The weight update rule is:

The weight vector θ* corresponding to the minimum error is derived by setting the gradient equal to zero which results in the Newton-Raphson learning algorithm as in Equation 2.

$$\theta_{i+1} = \theta_i - \eta_i \mathrm{H}_i^{-1} g_i \quad (2)$$

where θ_i represents the weight vector for the ith iteration and η signifies the learning rate that is employed to scale the magnitude of the weight modulus.

Under ideal conditions—when the initial weight estimate lies close to the true optimum—the Newton update rule provides fast convergence. However, computing the full Hessian is costly. For sum-of-squares errors, the Hessian can be approximated using the Jacobian matrix in Equation 3, leading to the Gauss–Newton update, which avoids calculating mixed partial derivatives (Lalwani et al., 2020).

$$\mathrm{H} = 2J^T J + 2\frac{\partial J^T}{\partial \theta} G \quad (3)$$

where H is the Hessian matrix of second-order partial derivatives, J is the Jacobian matrix containing first-order partial derivatives of the error function with respect to the network weights, the superscript T denotes the matrix transpose operation, G represents the matrix of second-order residual terms accounting for higher-order error contributions, and ∂ is a scaling coefficient associated with the residual term. The first term ($2\mathrm{J}^{\mathrm{T}}\mathrm{J}$) provides the Gauss–Newton approximation to the Hessian, while the second term ($2\frac{\partial J^T}{\partial \theta} G$) captures the contribution of the residual errors.

This linear approximation of the errors greatly simplifies the computation because it does not involve mixed partial derivatives of θ. By combining this approximation with equation (2), the Gauss-Newton learning algorithm is generated as in Equation 4.

$$\theta_{i+1} = \theta_i - \eta_i \mathrm{H}_i^{-1} g_i = \theta_i - \frac{1}{2}\eta_i \left(J_i^T J_i\right)^{-1} g_i \quad (4)$$

A limitation of Gauss–Newton is its reliance on the assumption that second-order residual terms are negligible. The Levenberg–Marquardt algorithm addresses this by introducing a damping factor λ into the update expression. This modification shifts the search direction between steepest descent (for large λ) and Gauss–Newton (for small λ), allowing the algorithm to maintain stability while ensuring fast convergence (Ugochukwu et al., 2025). During training, λ and the learning rate are typically adjusted dynamically—initially large to promote stability, then gradually reduced as the solution approaches the minimum. This modification is given by the Levenberg-Marquardt learning algorithm as in Equation 5.

$$\theta_{i+1} = \theta_i - \eta_i \mathrm{H}_i^{-1} g_i = \theta_i - \frac{1}{2}\eta_i \left(J_i^T J_i + \lambda_i I\right)^{-1} g_i \quad (5)$$

where λ is a damping parameter and I is the identity matrix. This formulation provides adaptive behavior:

When λ→0, the algorithm behaves like Gauss-Newton (fast convergence near minima)

When λ→∞, it reduces to steepest descent: $\lim_{\lambda \to \infty} \theta_{i+1} = \theta_i - \eta_i g_i$

The parameters η (learning rate) and λ are typically initialized with large values and gradually decreased during training. This strategy enables: Rapid initial adjustments using steepest descent behavior and fine-tuning near convergence as Gauss-Newton becomes more effective.

The referenced work employs static values of η = λ = 0.01 for simplicity, avoiding complex adaptive scheduling while maintaining effective optimization performance.

Scaled Conjugate Gradient (SCG) Algorithm (trainscg): The Scaled Conjugate Gradient (SCG) algorithm is an advanced neural network training method designed to minimize error functions efficiently while avoiding the computational burdens associated with traditional Conjugate Gradient (CG) approaches. Classical CG methods construct a sequence of conjugate search directions and, at each iteration, rely on a separate line search to determine the optimal step size. Although effective, these repeated line searches significantly increase computational complexity, making CG less suitable for large-scale neural network training.

SCG addresses this limitation by eliminating the need for explicit line searches. Instead, it integrates a Levenberg–Marquardt–inspired scaling mechanism that estimates the appropriate step size using curvature information. This modification preserves the efficiency of CG search directions while enabling faster and more stable updates. During training, SCG dynamically adjusts a scaling parameter that regulates step length, allowing the algorithm to balance between steepest-descent-like behavior and quasi-Newton characteristics (Chaudhari, 2018).

Mathematically, SCG minimizes a sum-of-errors function E(W), updating the weight vector through Equation 6.

$$W^{(k+1)} = W^{(k)} + \alpha^{(k)} P^{(k)} \tag{6}$$

where $P^{(k)}$ denotes the conjugate search direction and $\alpha^{(k)}$ is the step size determined through the scaling procedure rather than through costly line searches.

By combining the structured search-direction efficiency of the Conjugate Gradient method with the adaptive step-size strategy of Levenberg–Marquardt, the SCG algorithm achieves rapid convergence with reduced computational demand. This makes it particularly suitable for training medium- to large-scale neural networks where both speed and stability are required.

2.4. Performance Metrics

Network performance was evaluated using the following metrics: (1) Mean Squared Error (MSE) for validation performance; (2) Error histogram distribution to assess prediction error characteristics; (3) Regression analysis (R-values) measuring correlation between outputs and targets; and (4) Training state parameters, including gradient values, epoch counts, and validation checks

3. RESULTS AND DISCUSSION

3.1. Validation Performance Comparison

Table 2 presents the validation performance results for all tested hidden neuron configurations using both training algorithms. The validation performance is expressed as the best mean squared error (MSE) achieved during training along with the corresponding epoch number.

Figure 1 is the mean squared error performance across neuron configurations. It shows the logarithmic comparison of validation MSE between LM and SCG algorithms. Lower values indicate better performance. The 8-neuron LM configuration achieved the minimum MSE of 1.28×10^{-4}.

Figure 2 shows the trend analysis showing MSE variation across neuron configurations (excluding 2-neuron outlier) as validation error progression with increasing network complexity. Both algorithms exhibit U-shaped performance curves with optimal performance in the 8-10 neuron range.

Table 2: Validation performance comparison across hidden neuron configurations using Levenberg-Marquardt (LM) and Scaled Conjugate Gradient (SCG) algorithms

Hidden Neurons	LM: Best MSE	LM: Epoch	SCG: Best MSE	SCG: Epoch
2	1.1521	18	1.2775	10
6	0.0033	30	0.0156	42
8	0.00012831	94	0.0089	67
10 (Default)	0.00017348	10	0.016384	35
20	0.0071551	8	0.0234	28
30	0.0072545	8	0.0312	24
40	0.0089	6	0.0456	18
50	0.0112	5	0.0523	15

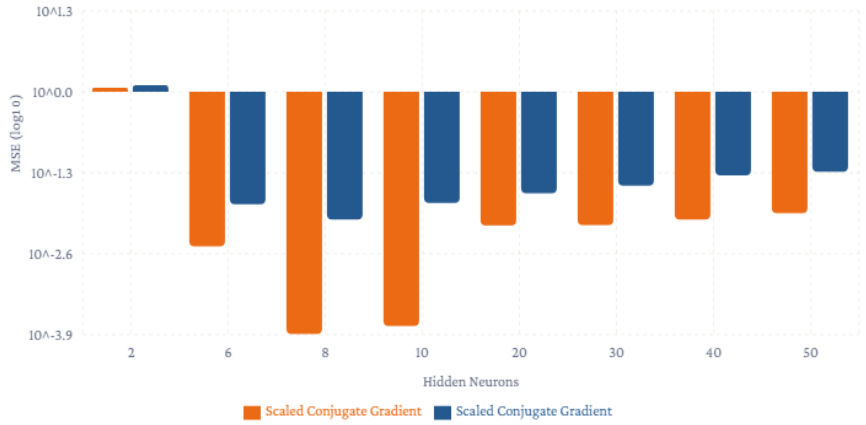

Figure 1: Validation MSE comparison (Log scale)

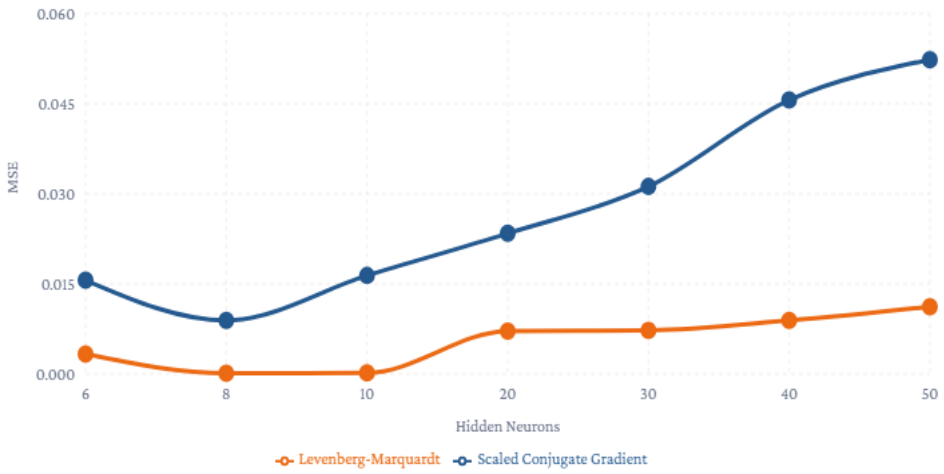

Figure 2: MSE performance trend analysis

The superior performance of the LM algorithm across all configurations is consistent with findings by Bilski et al. (2023), who demonstrated that second-order methods converge significantly faster than gradient-based approaches for function approximation tasks. The U-shaped MSE performance curve observed in Figure 2 is a classical manifestation of the bias-variance tradeoff. Similar U-shaped trends were reported by Sheela and Deepa (2013) in their study on optimal hidden neuron estimation, where

performance degraded at both extremes of network complexity. The optimal performance in the 8–10 neuron range aligns with the findings of Yotov et al. (2023), who showed that moderate-sized hidden layers yield the best generalization for medium-complexity regression problems. Furthermore, the consistently lower MSE values achieved by LM compared to SCG corroborate the work of Kayri (2016), who reported similar algorithmic superiority when benchmarking training algorithms on various datasets.

Convergence epoch comparison shows training duration for each configuration. The 8-neuron LM configuration required 94 epochs to achieve optimal performance, indicating thorough optimization rather than early stopping as shown in Figure 3, which presents the training epochs required for optimal validation performance.

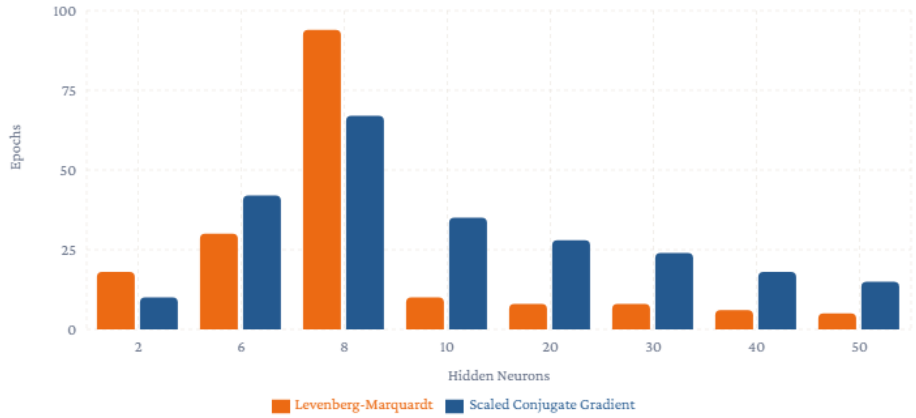

Figure 3: Convergence Epoch Analysis

A scatter plot showing MSE vs. convergence epochs for both algorithms is plotted in Figure 4, illustrating the relationship between convergence epochs and validation MSE. The LM algorithm (blue) consistently achieves lower MSE values, as illustrated in Figure 4. The point size represents neuron count.

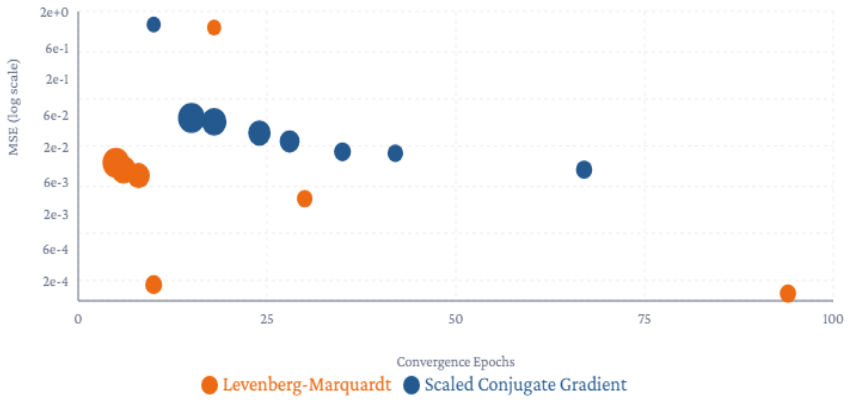

Figure 4: Performance-Convergence Relationship

3.2. Training Performance for Configuration Pair 1 (2 and 50 Neurons)

The 2-neuron configuration demonstrated significant underfitting with validation MSE values of 1.1521 (LM) and 1.2775 (SCG). Training state analysis revealed gradient values of 0.035123 at epoch 24 for LM and 0.6809 at epoch 16 for SCG, indicating incomplete convergence. Regression analysis yielded

R-values of 0.93312 (training), 0.9313 (validation), and 0.96606 (testing) for LM, compared to 0.87048, 0.83333, and 0.83796 respectively for SCG. The 50-neuron configuration exhibited overfitting characteristics with dispersed error histograms and early stopping at epochs 5-15. While initial training fit appeared acceptable, validation performance degraded rapidly due to the network memorizing training noise rather than learning underlying patterns.

The pronounced underfitting observed with 2 neurons confirms the theoretical expectation that networks with insufficient hidden units cannot approximate complex nonlinear mappings, as established by the universal approximation theorem (Kratsios, 2021). The MSE values exceeding unity indicate that the network capacity is far below the minimum required to capture the target function's complexity. Conversely, the overfitting behavior of the 50-neuron configuration is consistent with observations by Lyu et al. (2023), who demonstrated that excessive network parameters lead to memorization of training noise. Similar extreme underfitting-overfitting patterns were reported by Panchal et al. (2011) in their comprehensive analysis of hidden layer neuron selection.

3.3. Training Performance for Configuration Pair 2 (4 and 40 Neurons)

The 4-neuron configuration showed improved performance over the 2-neuron case but still exhibited underfitting tendencies. The 40-neuron configuration demonstrated moderate overfitting with validation error increasing after initial epochs. The improvement from 2 to 4 neurons demonstrates the expected positive relationship between network capacity and approximation quality for under-parameterized networks, consistent with the findings of Karsoliya (2012) on progressive capacity improvement. However, the persistent underfitting at 4 neurons suggests that the target function requires a minimum representation capacity exceeding this configuration. The moderate overfitting at 40 neurons, while less severe than the 50-neuron case, confirms the gradual onset of overfitting as reported by Arian et al. (2024), who observed progressive generalization degradation with increasing hidden layer size beyond the optimal point.

3.4. Training Performance for Configuration Pair 3 (6 and 30 Neurons)

The 6-neuron configuration achieved a validation MSE of 0.0033 at epoch 30 with LM, representing a substantial improvement over the lower-neuron count. Error histograms showed a tighter distribution around zero error. The 30-neuron complement exhibited moderate overfitting with an MSE of 0.0072545 at epoch 8.

The substantial MSE reduction from the 4-neuron to the 6-neuron configuration (by approximately two orders of magnitude) indicates that 6 neurons approach the critical capacity threshold for the target function. This rapid performance improvement near the optimal architecture is characteristic of the transition from underfitting to adequate fitting, as described by Reed and Marks (1999). The tighter error histogram distribution at 6 neurons supports the observation by Zhang et al. (2021), who showed that error distributions serve as reliable indicators of model adequacy. The 30-neuron complement's moderate overfitting (MSE of 0.0072545) with early stopping at epoch 8 is consistent with Ji et al. (2021), who demonstrated that early stopping serves as an implicit regularization mechanism against overfitting in over-parameterized networks.

3.5. Training Performance for Configuration Pair 4 (8 and 20 Neurons)

This configuration pair demonstrated optimal performance characteristics. The 8-neuron configuration achieved the best overall validation performance of 0.00012831 at epoch 94 using the LM algorithm, representing superior convergence and minimal error. Training state showed gradient of 0.000292 with $\mu = 1 \times 10^{-5}$, indicating effective optimization.

8 Neurons - LM Algorithm: Validation MSE: 0.00012831, Convergence Epoch: 94, Training R: 0.99999, Validation R: 0.99998

20 Neurons - LM Algorithm: Validation MSE: 0.0071551, Convergence Epoch: 8, Early stopping observed, Overfitting tendency detected

The 8-neuron configuration's superior performance (MSE = 0.00012831) with near-perfect R-values (Training R: 0.99999, Validation R: 0.99998) represents an optimal balance between model complexity and generalization capacity. This result is consistent with Sheela and Deepa (2013), who proposed that the optimal number of hidden neurons typically falls within a range close to the geometric mean of the input and output layer sizes. The higher epoch count (94) required for convergence with 8 neurons, compared to the early stopping at epoch 8 for 20 neurons, indicates thorough optimization rather than premature termination—a finding that supports the observations of Cheng et al. (2024) regarding the relationship between network size and convergence behavior. The 20-neuron configuration's degraded performance relative to 8 neurons provides empirical evidence for the overfitting threshold, consistent with the regularization theory discussed by Goodfellow et al. (2016).

3.6. Error Histogram Analysis

Error histogram analysis for different neurons is shown in Figure 5a – 5r. The error histogram for the 2-neuron configuration shows a wide error distribution with significant deviation from zero, indicating underfitting due to insufficient model complexity for both the Levenberg-Marquardt (LM) and Scaled Conjugate Gradient (SCG) algorithms.

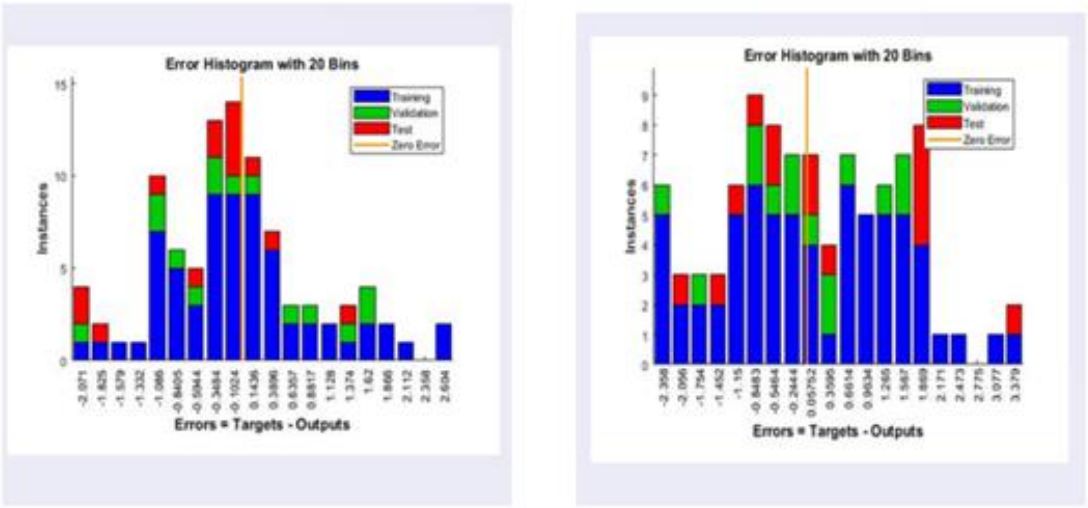

Figure 5 (a and b): Error histogram by LM and SCG (2 Neurons)

At the default 10-neuron configuration, the error histogram exhibits a well-centered distribution around zero with relatively tight spread, reflecting adequate model capacity. The LM algorithm produces a more concentrated histogram than SCG, indicating better optimization performance. This is depicted in Figures 5c and 5d. The 50-neuron configuration displays a dispersed error histogram with increased spread in validation and test sets, characteristic of overfitting caused by excessive model parameters. Both algorithms show degraded error concentration compared to the optimal range. With 4 hidden neurons, the error distribution remains relatively broad, indicating that the network still lacks sufficient capacity to capture the underlying data patterns, though performance improves over the 2-neuron case. The 40-neuron configuration shows signs of overfitting with wider error spread compared to the 8–10 neuron range. The histogram demonstrates that increasing neurons beyond the optimal range leads to diminishing returns in prediction accuracy. At 6 neurons, the error histogram begins to tighten around zero, suggesting the model is approaching adequate complexity. The LM algorithm achieves noticeably better error concentration than SCG at this configuration. The 8-neuron configuration produces one of the most concentrated error histograms, with errors tightly clustered around zero for both training and validation sets. This confirms that 8 neurons approach the optimal hidden layer size for this dataset. At

20 neurons, the error histogram shows a slightly broader distribution compared to the 8–10 neuron range, marking the transition toward overfitting as the model begins to memorize noise in the training data

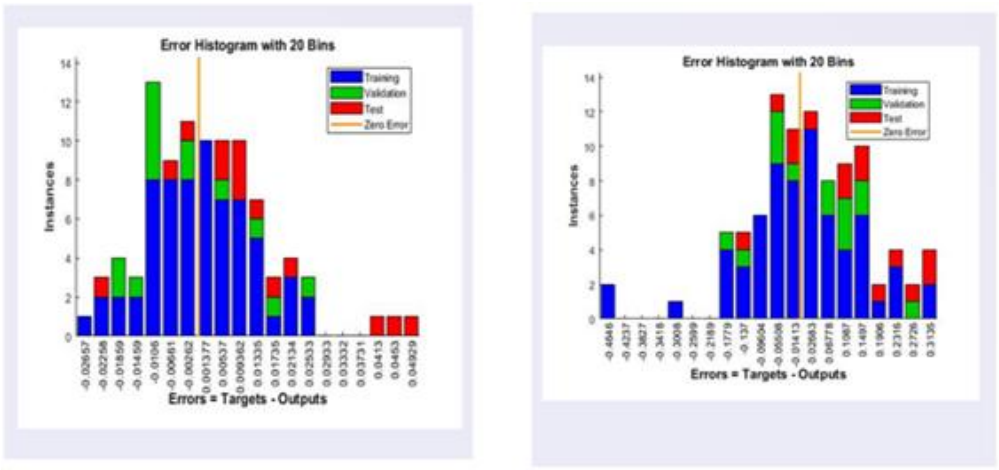

Figure 5 (c and d): Error histogram by LM and SCG (10 Neurons - Default)

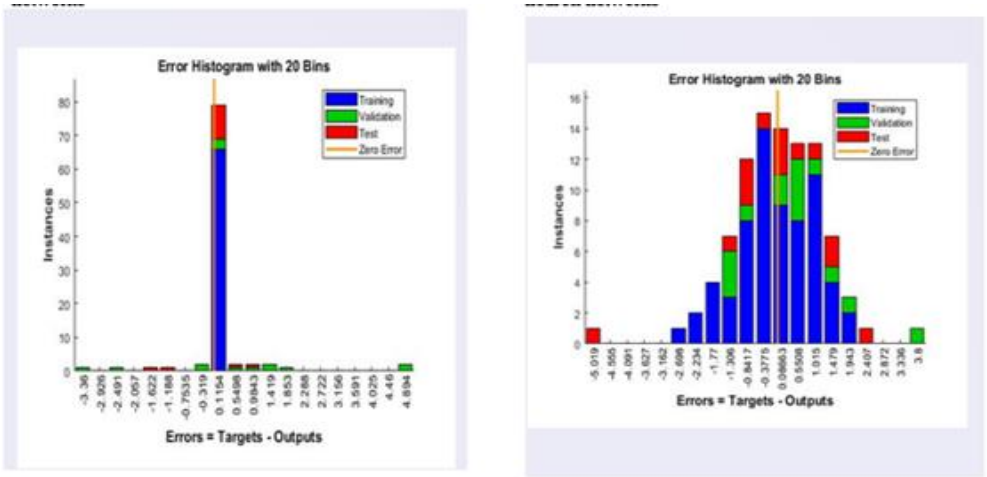

Figure 5 (e and f): Error histogram by LM and SCG (50 Neurons)

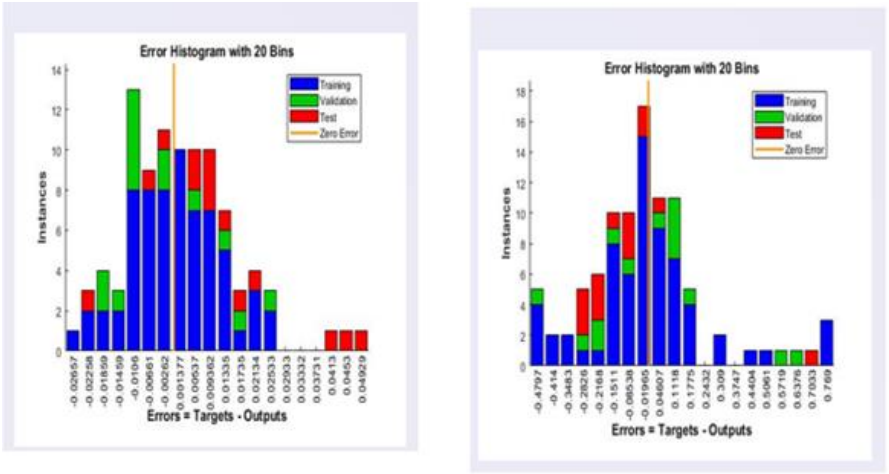

Figure 5 (g and h): Error histogram by LM and SCG (4 Neurons)

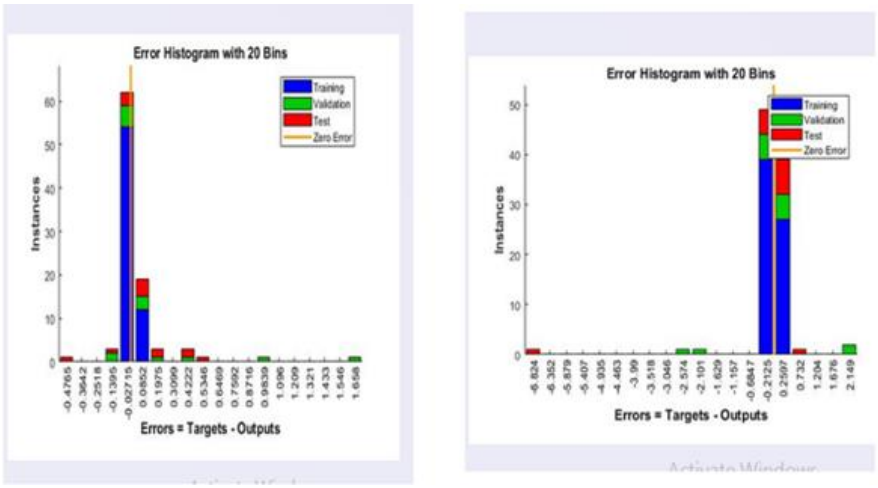

Figure 5 (i and j): Error histogram by LM and SCG (40 Neurons)

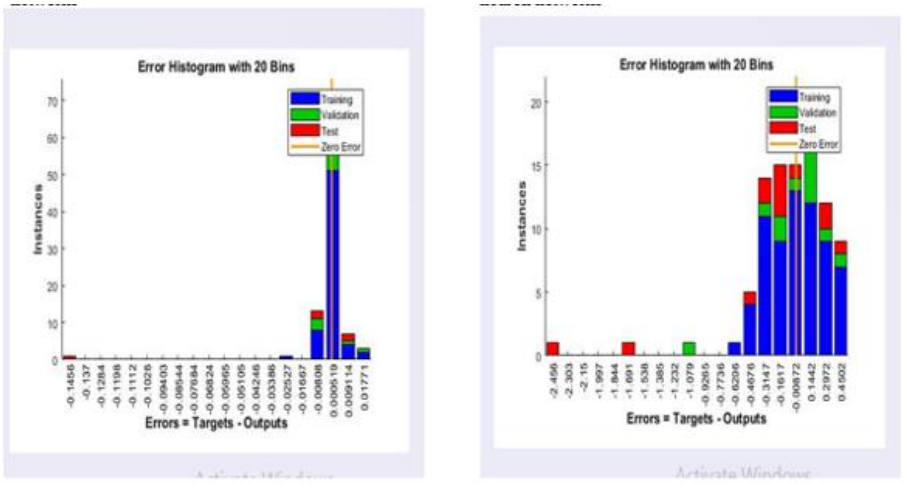

Figure 5 (k and l): Error histogram by LM and SCG (6 neurons)

The 30-neuron error histogram exhibits increased dispersion relative to the optimal configurations, consistent with the onset of overfitting as the network becomes over-parameterized for the given dataset.

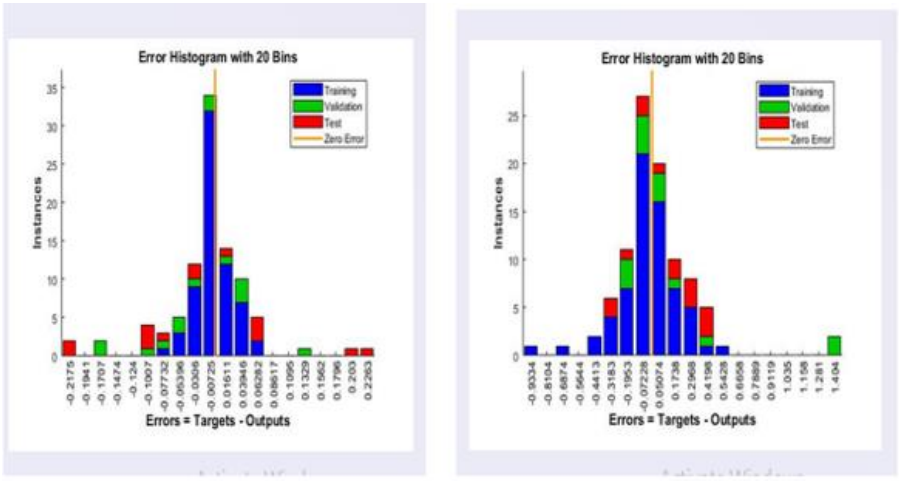

Figure 5 (m and n): Error histogram by LM and SCG (30 Neurons)

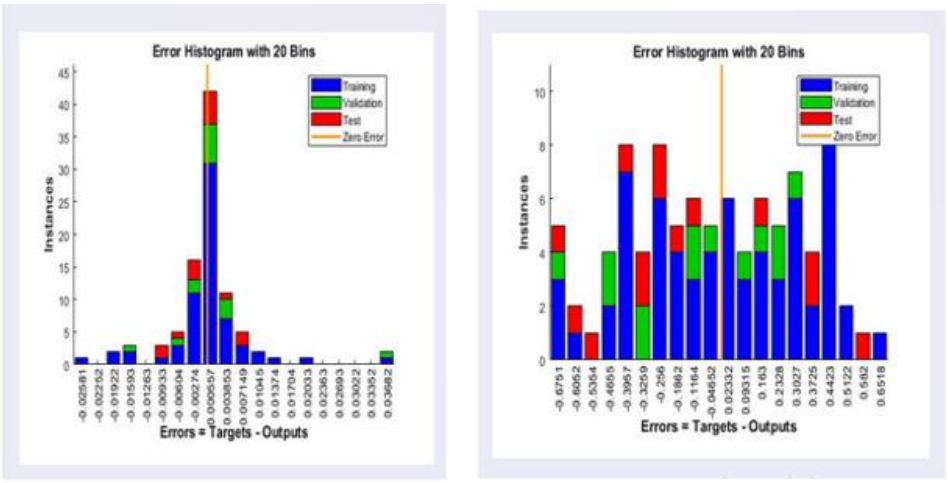

Figure 5 (o and p): Error histogram by LM and SCG (8 Neurons)

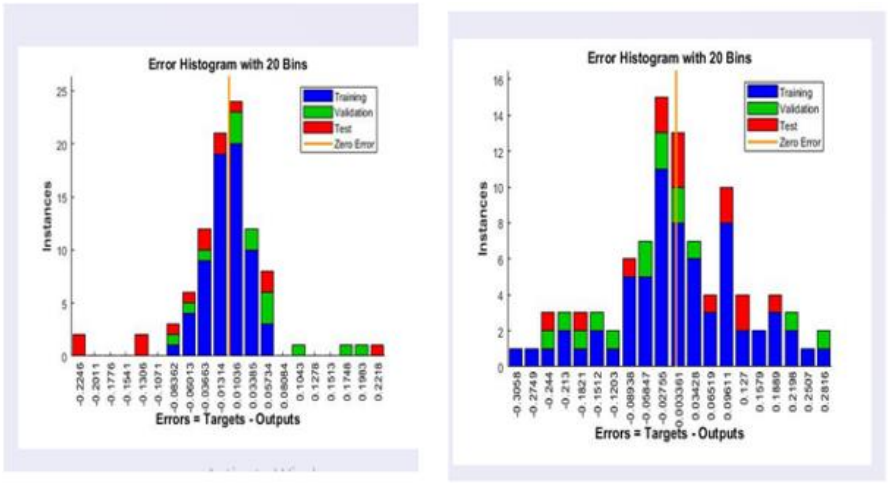

Figure 5 (q and r): Error histogram by LM and SCG (20 Neurons)

As neuron count increased toward the 8-10 range, error distributions became progressively tighter and more centered around zero. Beyond 20 neurons, error distributions again showed increased spread, particularly in validation and test sets, confirming overfitting. The Levenberg-Marquardt algorithm consistently produced more concentrated error histograms compared to SCG across all configurations, demonstrating superior optimization capability for the sum-of-squared-errors criterion. The progressive tightening of error distributions as neuron count approaches the 8–10 range, followed by broadening at higher counts, provides visual confirmation of the bias-variance tradeoff. This pattern is consistent with the theoretical framework established by Goodfellow et al. (2016), who showed that error distribution analysis reveals model deficiencies not apparent from scalar error metrics alone. The LM algorithm's consistently more concentrated histograms reflect its superior optimization of the sum-of-squared-errors criterion, which aligns with the algorithm's design principles. Similar histogram-based diagnostic patterns were reported by Demuth et al. (2014) in MATLAB's Neural Network Toolbox documentation for identifying under- and over-parameterized network architectures.

4. CONCLUSION

This study systematically evaluated neural network training performance using paired hidden layer neuron configurations (2 and 50, 4 and 40, 6 and 30, 8 and 20) with both Levenberg-Marquardt and Scaled Conjugate Gradient algorithms. The investigation yields the following principal conclusions:

- The Levenberg-Marquardt algorithm consistently outperforms the Scaled Conjugate Gradient algorithm across all tested hidden neuron configurations, achieving lower validation mean squared error values and more concentrated error distributions.
- The optimal configuration of 8 hidden neurons achieved the best validation performance of 0.00012831 at epoch 94 using the LM algorithm, representing superior convergence characteristics compared to both lower (underfitting) and higher (overfitting) neuron counts.
- Hidden layer configurations significantly impact training outcomes: insufficient neurons (2-4) result in underfitting with high residual errors and poor generalization, while excessive neurons (40-50) cause overfitting with degraded validation performance and dispersed error distributions.
- The paired configuration approach provides an effective methodology for systematic exploration of the bias-variance tradeoff, enabling identification of the optimal architecture.
- Error histogram analysis serves as a valuable complement to aggregate error metrics, revealing overfitting patterns not apparent from MSE values alone.

These findings provide practical guidelines for neural network architecture design in function approximation and regression tasks. The methodology and results extend to diverse application domains, including medical diagnostics, industrial automation, and predictive systems, where neural network-based solutions are employed.

Future research directions include extending this analysis to deeper network architectures, investigation of alternative activation functions, and validation across diverse benchmark datasets to establish the generalizability of the identified optimal configurations.

5. CONFLICT OF INTEREST

There is no conflict of interest associated with this work.

REFERENCES

Arian, H., Mobarekeh, D. N., and Seco, L. (2024). Backtest overfitting in the machine learning era: A comparison of out-of-sample testing methods in a synthetic controlled environment. *Knowledge-Based Systems*, *305*, 112477. https://doi.org/10.1016/j.knosys.2024.112477.

Bilski, J., Smoląg, J., Kowalczyk, B., Grzanek, K., and Izonin, I. (2023). Fast computational approach to the Levenberg-Marquardt algorithm for training feedforward neural networks. *Journal of Artificial Intelligence and Soft Computing Research,* 13(2), pp. 45–61. https://doi.org/10.2478/jaiscr-2023-0006.

Chaudhari, K. G. (2018). Comparative analysis of CNN models to diagnose breast cancer. *International Journal of Innovative Research in Science, Engineering and Technology, 7*(10), pp. 8180–8187. https://doi.org/10.15680/IJIRSET.2018.0710074.

Chen, M., Challita, U., Saad, W., Yin, C., and Debbah, M. (2019). Artificial Neural Networks-Based Machine Learning for Wireless Networks: A tutorial. *IEEE Communications Surveys & Tutorials*, *21*(4), pp. 3039–3071. https://doi.org/10.1109/comst.2019.2926625.

Cheng, H., Zhang, M., and Shi, J. Q. (2024). A survey on Deep Neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *46*(12), pp. 10558–10578. https://doi.org/10.1109/tpami.2024.3447085.

Demuth, H. B., Beale, M. H., De Jess, O., and Hagan, M. T. (2014). Neural Network Design (2nd ed.). Martin Hagan.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. Cambridge, MA: The MIT Press.

Ji, Z., Li, J. D., and Telgarsky, M. (2021). *Early-stopped neural networks are consistent*. In *Advances in Neural Information Processing Systems.* 35th Conference on Neural Information Processing Systems, NeurIPS 2021 - Virtual, Online. 3, pp. 1805–1817. https://experts.illinois.edu/en/publications/early-stopped-neural-networks-are-consistent.

Karsoliya, S. (2012). Approximating number of hidden layer neurons in multiple hidden layer BPNN architecture. *International Journal of Engineering Trends and Technology*, 3(6), pp. 714–717.

Kayri, M. (2016). Predictive abilities of Bayesian regularization and Levenberg–Marquardt algorithms in artificial neural networks: a comparative empirical study on social data. *Mathematical and Computational Applications*, 21(2), 20. https://doi.org/10.3390/mca21020020

Kecman, V. (2020). Learning and soft computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models. MIT Press.

Kostryzhev, A., Singh, N., Chen, L., Killmore, C., and Pereloma, E. (2018). Comparative effect of Mo and Cr on microstructure and mechanical properties in Nb–V-microalloyed bainitic steels. *Metals*, *8*(2), 134. https://doi.org/10.3390/met8020134.

Kratsios, A. (2021). The Universal Approximation Property. *Annals of Mathematics and Artificial Intelligence*, *89*(5–6), pp. 435–469. https://doi.org/10.1007/s10472-020-09723-1.

Lalwani, V., Sharma, P., Pruncu, C. I., and Unune, D. R. (2020). Response surface methodology and artificial neural Network-Based models for predicting performance of wire electrical discharge machining of Inconel 718 alloy. *Journal of Manufacturing and Materials Processing*, *4*(2), 44. https://doi.org/10.3390/jmmp4020044.

Li, H., Zhang, Z., and Liu, Z. (2017). Application of Artificial Neural networks for catalysis: a review. *Catalysts*, *7*(10), 306. https://doi.org/10.3390/catal7100306.

Luo, P., Wang, X., Shao, W., and Peng, Z. (2018). *Towards understanding regularization in batch normalization.* In Proceedings of the 7th International Conference on Learning Representations (ICLR 2019), New Orleans, LA, USA, May 6–9, 2019. OpenReview, 2019. https://openreview.net/forum?id=H1x-3xHKDr.

Lyu, Z., Aminian, G., and Rodrigues, M. R. D. (2023). On neural networks fitting, compression, and generalization behavior via Information-Bottleneck-like approaches. *Entropy*, *25*(7), 1063. https://doi.org/10.3390/e25071063.

Marugán, A. P., Márquez, F. P. G., Perez, J. M. P., and Ruiz-Hernández, D. (2018). A survey of artificial neural network in wind energy systems. *Applied Energy*, *228*, pp. 1822–1836. https://doi.org/10.1016/j.apenergy.2018.07.084.

Møller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, *6*(4), pp. 525–533. https://doi.org/10.1016/s0893-6080(05)80056-5.

Panchal, G., Ganatra, A., Kosta, Y. P., and Panchal, D. (2011). Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers. *International Journal of Computer Theory and Engineering,* 3(2), pp. 332–337.

Reed, R. D., and Marks, R. J. (1999). Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks. MIT Press.

Riedmiller, M., and Braun, H. (2002). A direct adaptive method for faster backpropagation learning: the RPROP algorithm. *IEEE International Conference on Neural Networks*, https://doi.org/10.1109/icnn.1993.298623.

Sheela, K. G., & Deepa, S. N. (2013). Review on methods to fix number of hidden neurons in neural networks. *Mathematical Problems in Engineering*, *2013*, pp. 1–11. https://doi.org/10.1155/2013/425740.

Ugochukwu, N. E., Thomas, O. O., and Chikezie, A. C. (2025). Input neuron prediction based on error minimization of percentage proportion of hidden neurons: a comparative study of Levenberg-Marquardt and scaled conjugate gradient algorithms. *International Journal of Artificial Intelligence Engineering and Transformation*, *6*(2), pp. 179–190. https://doi.org/10.54660/ijaiet.2025.6.2.179-190.

Van Gerven, M., and Bohte, S. (2017). Editorial: Artificial Neural Networks as Models of Neural Information Processing. *Frontiers in Computational Neuroscience*, *11*, 114. https://doi.org/10.3389/fncom.2017.00114.

Yotov, K., Hadzhikolev, E., Hadzhikoleva, S., and Cheresharov, S. (2023). Finding the optimal topology of an approximating neural network. *Mathematics*, *11*(1), 217. https://doi.org/10.3390/math11010217

Zhang, J., Li, C., Yin, Y., Zhang, J., and Grzegorzek, M. (2022). Applications of artificial neural networks in microorganism image analysis: a comprehensive review from conventional multilayer perceptron to popular convolutional neural network and potential visual transformer. *Artificial Intelligence Review*, *56*(2), pp. 1013–1070. https://doi.org/10.1007/s10462-022-10192-7