International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 13 Issue: 1
Article Received: *25 January 2025* ***Revised:*** *12 March 2025* ***Accepted:*** *30 May 2025*

Artificial Neural Networks and Optimization Technique: A theoretical study

Basir Ahamed Khan
Department of Physics, Murshidabad University
Berhampore, Murshidabad 742101, India
e-mail: bakhan.mu@gmail.com

Abstract— Artificial Neural Networks (ANNs) have become a pivotal tool in modern artificial intelligence (AI), significantly impacting various fields such as image processing, natural language processing, and autonomous systems. The training process of ANNs requires finding optimal parameters (weights and biases) that minimize a loss function, which can be computationally intensive and challenging. To achieve better performance, it is crucial to employ efficient optimization techniques that guide the network toward optimal solutions effectively. This paper provides an overview of ANNs, including their structure, types, applications, advantages, challenges, and future directions. This review also provides optimization techniques that are used to enhance their performance during training.

Keywords- Artificial Neural Network; Back propagation; Feed forward; Gradient descent; Supervised learning

I. INTRODUCTION

Artificial Neural Networks are computational models inspired by the biological neural networks that make up the human brain. These neurons work together to solve complex problems by mimicking how biological neurons in the human brain process information [1-6].The increasing availability of large datasets, coupled with advancements in computational power, has fuelled the development of deep learning architectures, extending the capabilities of ANNs [7]. These networks, particularly Deep Neural Networks (DNNs), have demonstrated remarkable success in solving challenging problems across multiple disciplines [8,9]. The effectiveness of ANNs is largely attributed to their ability to learn from data through optimization techniques such as backpropagation [10,11] and gradient descent [12]. Originally developed for pattern recognition tasks, ANNs have evolved to support deep learning and complex AI applications. The key components of an ANN include: Input Layer: Accepts the input data. Hidden Layers: Layers of neurons that perform transformations and learn features from the input data and Output Layer: Produces the final prediction or classification result. Neurons in the hidden layers use weights, biases, and activation functions to process the input data and make predictions [13]. During training, ANNs adjust these weights through a process called backpropagation to minimize the error between predicted and actual values. The network uses activation functions (e.g., ReLU, sigmoid) to introduce non-linearity. Schematic diagram of ANN structure is shown in Figure.1.

II. TYPES OF NEURAL NETWORK

Different types of neural networks are used for classification tasks. Feedforward net-works (FNNs) [14], convolutional neural networks (CNNs) [15,16], recurrent neural net-works (RNNs) [17], and transformers are key types of neural networks used in various machine learning applications [18]. Feedforward neural networks (FNNs) are the simplest type, where information flows in one direction from the input layer to the output layer without loops, making them suitable for basic classification and regression tasks. Convolutional neural networks (CNNs) are designed for image processing, utilizing filters and pooling layers to detect spatial patterns and features efficiently. They are widely used in tasks such as image recognition, object detection, and facial recognition. Recurrent neural networks (RNNs), on the other hand, are designed for sequential data processing, allowing information to persist across time steps, making them useful for speech recognition, time-series forecasting, and natural language processing. A more advanced form of RNNs, called Long Short-Term Memory (LSTM) networks, helps address the issue of vanishing gradients and improves long-term dependency handling. Transformers, a newer and more powerful architecture, use self-attention mechanisms to process sequences in parallel rather than sequentially, making them highly effective for natural language processing tasks, including machine translation and text generation. These different neural network architectures have revolutionized artificial intelligence [19] by enabling machines to perform complex tasks with high accuracy.

III. ANN TRAINING

In Artificial Neural Networks (ANNs), training data plays a crucial role in learning patterns and making accurate predictions. The training dataset consists of input features (independent variables) and corresponding target outputs (dependent variables), which the model uses to learn relationships between inputs and outputs. Before training begins, the data is often preprocessed through normalization, scaling, or feature engineering to enhance learning efficiency. The dataset is typically divided into three subsets: training data, which is used to update the model's weights; validation data, which helps tune hyperparameters and prevent overfitting and test data, which evaluates the model's performance on unseen data. A well-prepared training dataset should be diverse and representative of real-world scenarios to ensure that the ANN generalizes well. The quality, quantity, and balance of the

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 13 Issue: 1
***Article Received:** 25 January 2025 **Revised:** 12 March 2025 **Accepted:** 30 May 2025*

training data significantly influence the model's accuracy and reliability. Training an ANN involves the following key steps:

- Forward Propagation: Input data is passed through the network, and predictions are made.
- Loss Function: The difference between the predicted and true output is measured using a loss function (e.g., Cross-Entropy Loss, Mean Squared Error).
- Backpropagation: The error is propagated back through the network to update the weights using optimization algorithms (e.g., Stochastic Gradient Descent, Adam).
- Epochs: The process of forward propagation and backpropagation [20] is repeated for multiple epochs until the network converges to an optimal set of weights.

IV. ACTIVE FUNCTIONS AND CLASSICIFICATIONS

Activation functions are critical in determining the output of each neuron and introducing non-linearity into the network. Some commonly used activation functions in classification tasks include:

- Sigmoid: Outputs values between 0 and 1, often used in binary classification tasks.
- ReLU (Rectified Linear Unit): Outputs the input directly if positive, otherwise zero. Popular due to its simplicity and efficiency.
- Softmax: Used in the output layer for multi-class classification tasks to normalize outputs into a probability distribution.

Table 1. Most common activation functions used in ANNs.

Function	Expression	Application
Linear	$f(z) = z$	Output layers
Logistic sigmoid	$f(z) = \frac{1}{1+e^{-z}}$	Output and hidden layers
Hyperbolic tangent sigboid	$f(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	Hidden layers
ReLU	$f(z) = \max(0, z)$	Output layers

V. CHALLENGES IN TRAINING ANNS

Training an ANN involves adjusting its weights and biases to minimize the difference between predicted outputs and actual labels. This process is done through optimization, but several challenges arise in this context:

- Local Minima: The loss function may have multiple local minima, making it difficult to find the global minimum [21].
- Vanishing/Exploding Gradients: During backpropagation, gradients can either become too small (vanishing) or too large (exploding), which hampers learning.
- Slow Convergence: Without efficient optimization, the learning process can be slow, especially for deep networks with many layers.
- Overfitting: ANNs may overfit the training data, resulting in poor generalization to unseen data.

Thus, an effective optimization technique can significantly enhance the training process and overall performance of ANNs.

VI. OPTIMIZATION IN ANNS

Optimization in ANNs involves adjusting the model's parameters (weights and biases) using a loss function (or cost function) and an optimization algorithm. The goal is to minimize the loss function through an iterative process. The most commonly used optimization algorithm in neural networks is Gradient Descent, but there are several variants and improvements to this technique that help make the training process more efficient.

A. Gradient descent

Gradient Descent (GD) [22] is an optimization algorithm that adjusts the parameters in the direction of the negative gradient of the loss function with respect to the parameters. The update rule for weights is

$$w_{new} = w_{old} - \eta . \nabla L(w) \quad (1)$$

where $w_{new}$is the updated weight, w_{old} is the current weight, $\eta$is the learning rate and $\nabla L(w)$ is the gradient of the loss function with respect to the weights. There are three main variations of gradient descent:

- Batch Gradient Descent (BGD): Computes the gradient of the entire dataset before updating the weights. It can be slow for large datasets.
- Stochastic Gradient Descent (SGD): Updates the weights after processing each training example. It's faster but introduces more noise, which can sometimes help escape local minima.
- Mini-batch Gradient Descent: Combines both BGD and SGD by using a small batch of data points to compute the gradient. This is often the most efficient approach.

B. Efficient optimization techniques

While gradient descent is fundamental, several advanced optimization techniques [23-26] have been developed to improve the convergence speed, stability, and performance of ANNs.

1. Momentum

Momentum helps accelerate the optimization by considering past gradients to smooth out the updates, which helps overcome local minima and reduce oscillations. The update rule with momentum is

$$v_{t+1} = \beta v_t + (1-\beta)\nabla L(w) \quad (2)$$

$$w_{t+1} = w_t - \eta v_{t+1} \quad (3)$$

where v_t is the velocity (accumulated gradients), β is the momentum factor (usually set to 0.9), and η is the learning rate.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 13 Issue: 1
***Article Received:** 25 January 2025 **Revised:** 12 March 2025 **Accepted:** 30 May 2025*

2. Adaptive Gradient Algorithms

These algorithms adapt the learning rate for each parameter individually, which helps in speeding up the training process and preventing overshooting.

(i) Adagrad

It adjusts the learning rate based on the frequency of parameter updates. Parameters that get updated frequently have smaller learning rates. The update rule is:

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{G_t+\epsilon}}\nabla L(w) \quad (4)$$

where G_t is the cumulative sum of squared gradients up to time step t, and ϵ is a small constant to prevent division by zero.

(ii) RMSprop(Root Mean Square Propagation)

It is an adaptive learning rate optimization algorithm designed to improve the performance and stability of gradient descent. It works by maintaining a moving average of the squared gradients for each weight and dividing the gradient by the root of this average. The moving average is controlled by a decay factor (commonly around 0.9), which determines how quickly past gradients are forgotten. By adapting the learning rate for each parameter based on recent gradient magnitudes, RMSprop is particularly effective in scenarios where the cost function varies widely across dimensions, such as in recurrent neural networks (RNNs) or deep feedforward networks. The update rule is

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{E[g^2]_t+\epsilon}}\nabla L(w) \quad (5)$$

where $E[g^2]_t$ is the moving average of the squared gradients.

(iii) Adam (Adaptive Momemt Estimation)

Adam combines the benefits of momentum and RMSprop. It keeps a moving average of both the gradients and the squared gradients. The update rule is

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla L(w)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla L(w))^2$$

$$w_{t-1} = w_t - \frac{\eta}{\sqrt{v_t}+\epsilon} m_t \quad (6)$$

where m_t and v_t are moving averages of the gradients and squared gradients, respectively. β_1 and β_2 are decay rates for the moving averages (typically 0.9 and 0.999).

(iv) Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm [27] is a popular optimization technique used to solve nonlinear least squares problems. It is often employed in problems where the goal is to minimize the sum of the squares of nonlinear functions. It's particularly useful when solving problems related to curve fitting, machine learning, and parameter estimation, where we want to minimize the discrepancy between a model and observed data.

- Nonlinear Least Squares Problem: The algorithm is commonly used when the objective function is nonlinear and you wish to minimize the sum of the squared residuals (errors).
- Combines Gradient Descent and Gauss-Newton Methods**:**
    - Gradient Descent: This method uses the gradient (or derivative) of the function to iteratively move toward the minimum.
    - Gauss-Newton Method: This is more specific for least squares problems, and it uses second-order approximations to find the optimal parameters. It approximates the Hessian (second-order derivative) with the Jacobian.
- Damping Factor**:** The Levenberg-Marquardt algorithm [28] introduces a damping factor, which adjusts the step size during the optimization process. This helps balance between the stability of gradient descent and the fast convergence of Gauss-Newton.
- Hybrid Approach: If the algorithm is far from the optimal solution, it behaves more like gradient descent. If it's closer to the optimal solution, it shifts toward the Gauss-Newton method.

The Levenberg-Marquardt algorithm works by iterating on the following steps:

- Compute the Jacobian Matrix: This matrix contains the first derivatives of the residuals (the errors) with respect to the model parameters.
- Compute the Hessian Approximation: The Hessian matrix is approximated using the Jacobian. It provides information about the curvature of the function.
- Update Rule: The parameters are updated using a combination of the gradient descent step and the Gauss-Newton step, adjusting the step size with the damping factor.

The update rule is

$$X_{k+1} = X_k - (J^T J + \lambda I)^{-1} J^T r \quad (7)$$

where X_k is the current set of parameters, J is the Jacobian matrix of residuals, λ is the damping factor (which is adjusted to every iteration), I is the identity matrix, r is the residual vector (the difference between the model predictions and actual

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 13 Issue: 1
Article Received: *25 January 2025* ***Revised:*** *12 March 2025* ***Accepted:*** *30 May 2025*

data). Damping Adjustment**:** The damping factor $\lambda$is adjusted after each iteration:If the new parameters lead to a decrease in the objective function, the damping factor is decreased (allowing the algorithm to behave more like Gauss-Newton [29]).If the new parameters do not lead to a decrease, the damping factor is increased (making the algorithm behave more like gradient descent).

3. Learning Types and Learning Rate Scheduling

Learning rate scheduling in Artificial Neural Networks (ANNs) is a technique used to adjust the learning rate during training to enhance model performance and convergence. The learning rate controls how much the model updates its weights based on the gradient of the loss function. A high learning rate may lead to unstable training, while a low one can slow down convergence. Scheduling methods like step decay (reduces the learning rate by a factor every few epochs), exponential decay (reduces the learning rate exponentially over time), and adaptive techniques (e.g., learning rate annealing or adaptive optimizers like Adam and RMSprop) help optimize training efficiency. Proper learning rate scheduling prevents overshooting, accelerates convergence, and improves generalization, making it a crucial component of deep learning optimization. Each learning type serves different purposes, with supervised learning [30] excelling in predictive tasks, unsupervised learning [31] useful for data exploration, and reinforcement learning [32] being ideal for dynamic decision-making scenarios. Comparisons among the three learning types are given in Table 2.

4. Regularization Technique

Regularization techniques in Artificial Neural Networks (ANNs) help prevent overfitting, ensuring that models generalize well to new data. One common approach is L1andL2regularization (Lasso and Ridge regression), which adds a penalty to the loss function to constrain the magnitude of the weights, preventing excessive complexity. Dropout is another widely used technique that randomly deactivates a fraction of neurons during training, forcing the network to learn more robust features. Batch normalization stabilizes and accelerates training by normalizing inputs to each layer, reducing internal covariate shifts. Early stopping monitors validation loss and halts training when overfitting begins. Data augmentation, particularly in image processing, enhances the training dataset by applying transformations such as rotation, flipping, and scaling, helping the model generalize better. Additionally, weight initialization techniques, like Xavier or He initialization, ensure stable gradient flow, preventing neurons from becoming inactive or dominating the learning process. These regularization techniques play a crucial role in improving the performance and reliability of ANNs across various applications [33]. It prevents overfitting and improve generalization by adding penalties to the loss function or modifying the network architecture.

- L1 and L2 Regularization**:**
    - L1 (lasso) adds the sum of the absolute values of the weights to the loss function.
    - L2 (ridge) adds the sum of the squared values of the weights.
- Dropout: Randomly drops a percentage of neurons during training, which helps prevent overfitting by forcing the network to rely on different combinations of neurons.
- Early Stopping**:** Monitors the performance on the validation set and stops training when the model's performance starts degrading, thus preventing overfitting.

Table 2. A comparison among supervised, unsupervised and reinforcement learning.

Features	**Supervised Learning**	**Unsupervised Learning**	**Reinforcement Learning**
Definition	Learns from labeled data	Learns from unlabeled data	Learns through trial and error with rewards and penalties
Data Type	Labeled data (input-output pairs)	Unlabeled data (no explicit output)	Interaction with an environment
Goal	Predict outcomes based on past data	Discover patterns or structures	Maximize cumulative rewards
Examples	Classification, Regression	Clustering, Anomaly Detection	Robotics, Game AI, Self-driving cars
Training Approach	Direct mapping of input to output	Finding hidden relationships in data	Learning by taking actions and receiving feedback
Supervision	Requires human-labeled data	No labeled data needed	Learns dynamically from rewards and penalties
Use Cases	Spam detection, Image recognition, Stock price prediction	Customer segmentation, Market basket analysis	Robot navigation, Chess-playing AI, Automated trading

VII. APPLIACTIONS OF ANNS

ANNs are widely applied across multiple domain:

- Healthcare: Disease prediction, medical imaging, drug discovery.
- Finance: Fraud detection, algorithmic trading, risk assessment.
- Automotive: Autonomous vehicles, driver-assist systems.
- Natural Language Processing (NLP): Machine translation, sentiment analysis, chatbots.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 13 Issue: 1
***Article Received:** 25 January 2025 **Revised:** 12 March 2025 **Accepted:** 30 May 2025*

- Manufacturing: Predictive maintenance, quality control.
- Image Classification: CNNs are widely used in image recognition and classification tasks, such as object detection, facial recognition, and medical image analysis.
- Speech Recognition: RNNs, particularly Long Short-Term Memory (LSTM) networks, are used to recognize speech patterns and transcribe spoken language.
- Text Classification: ANNs are used for tasks like sentiment analysis, topic classification, and spam detection in text data.
- Medical Diagnosis: ANNs can help classify diseases based on medical images, patient data, or diagnostic tests, improving decision-making in healthcare.
- Fraud Detection: ANNs are used in finance and e-commerce for detecting fraudulent transactions or behaviours.

VIII. CHALLENGES IN ANN-BASED CLASSICIFICATION

The While ANNs have achieved remarkable success, there are still several challenges:

- Overfitting: ANNs, especially deep ones, can overfit the training data, resulting in poor generalization to unseen data. Techniques like dropout, regularization, and early stopping are used to mitigate overfitting.
- Interpretability: ANNs, particularly deep neural networks, are often considered black-box models, making it difficult to interpret the rationale behind their predictions.
- Computational Complexity: Training deep neural networks requires substantial computational resources, including GPUs and large datasets. This can be a barrier for organizations with limited resources.
- Computational Cost: Requires significant processing power and memory.
- Data Quality and Quantity: ANNs require large, labeled datasets to perform well. For domains with limited labeled data, semi-supervised learning or transfer learning may be necessary. Performance heavily relies on high-quality, labeled datasets.
- Hyperparameter Tuning: ANNs have several hyperparameters (e.g., learning rate, number of layers, number of neurons), and finding the optimal combination can be time-consuming and requires experimentation.
- Black Box Nature: Lack of interpretability in decision-making processes.

IX. ADVANTAGES OF ANNS

One of the main strength of ANNs is their ability to learn and adapt from data, which allows them to model and recognize intricate patterns that are difficult for traditional algorithms to capture. They are especially useful for tasks such as image and speech recognition, natural language processing, and predictive analytics. ANNs are also capable of handling large and noisy datasets, making them suitable for real-world applications where data is often imperfect. Additionally, once trained, ANNs can make predictions or classifications quickly, which is beneficial for real-time systems. Their flexibility and scalability further enhance their usefulness across a wide range of industries and research fields. Point wise we can say the advantage of ANN as

- Adaptability: Can learn from data without explicit programming.
- Scalability: Handles large datasets efficiently.
- Pattern Recognition: Excels in image, speech, and text recognition tasks.
- Automation: Reduces the need for manual feature engineering.

X. FUTURE DIRECTIONS

The field of ANN-based classification continues to evolve, and several promising developments are shaping its future:

- Transfer Learning: Pre-trained models can be fine-tuned on new tasks with less labeled data, making ANNs more accessible in domains with limited data.
- Explainable AI (XAI): Researchers are working on methods to make ANNs more interpretable, helping users understand how predictions are made.
- Neural Architecture Search (NAS): Automated techniques to search for the optimal neural network architecture could improve the efficiency of model design.
- Quantum Neural Networks: The combination of quantum computing and neural networks could open up new frontiers in solving complex classification problems.

XI. CONCLUSIONS

Artificial Neural Networks have proven to be a powerful tool for classification tasks across various domains. In this paper, we explored artificial neural networks (ANNs) through the lens of optimization-driven learning, emphasizing the central role that optimization techniques play in enabling ANNs to generalize from data and solve complex problems. By framing learning as an optimization process, we highlighted how neural networks iteratively adjust their parameters to minimize a loss function, thereby capturing patterns in high-dimensional spaces. We examined key components of this framework, including loss function design, gradient-based optimization algorithms, regularization techniques, and architecture selection—all of which are critical to both performance and stability. The synergy between network design and optimization strategy is what ultimately drives the success of modern neural networks in fields ranging from computer vision to natural language processing. Despite their powerful capabilities, ANNs remain sensitive to issues like overfitting, local minima, and data distribution shifts. These challenges underline the importance of continued research in more robust optimization techniques, scalable architectures, and learning paradigms such as meta-

learning and unsupervised learning. In summary, optimization not only underpins how ANNs learn but also shapes their capacity to adapt, generalize, and evolve. A deeper understanding of this interplay offers pathways toward more efficient, interpretable, and intelligent learning systems in the future.

ACKNOWLEDGMENT

I would like to thank Prof. Jane Alam sir for encouraging me to write this article.

REFERENCES

[1] S. Haykin, "Neural Networks" – A Comprehensive Foundation, Prentice Hall, 2nd Ed. 1999
[2] C. M. Bishop, **"**Neural Networks for Pattern Recognition*"*, Oxford University Press. 1995**.**
[3] K. Gurney, "An Introduction to Neural Networks", Routledge, *ISBN 1-85728-673-1* London, 1997.
[4] R. Rojas, "Neural Networks: A Systematic Introduction", Springer, *ISBN 3-540-60505-3*, Germany, 1996. Yu, W.; He, H.; Zhang, N. "Advances in Neural Networks," ISNN 2009 6th International Symposium 2009.
[5] W. Yu, H. He, and N. Zhang, "Advances in Neural Networks", ISNN 2009 6th International Symposium, 2009.
[6] W. S. Mc Culloch, and W. Pitts, "A logical calculus of the ideas immanent in nervous activity." Bulletin of Mathematical Biophysics", Vol. 5, pp. 115-133, 1943
[7] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities". Proc Nat Acad. Sci, 79, 2554–2558, 1982.
[8] Y. Lecun, Y. Bengio and G. Hinton, "Deep learning", Nature, 521, 436–444, 2015.
[9] K. G. Kim, "Deep learning book review", Nature, 29, 1–73, 2019.
[10] G. Nunnari, "An improved back propagation algorithm to predict episodes of poor air quality", Soft Comput., 10, 132–139, 2006.
[11] Y. Wu, and S. Wang, "A new algorithm of improving the learning performance of neural network by feedback". Journal of Computer Research and Development, 41(9), 1488–1492, 2004.
[12] Amari. Shun-ichi, "Backpropagation and stochastic gradient descent method". Neurocomputing, 5, 185-196, 1993.
[13] E. Ahmadzadeh, J. Lee and I. Moon, "Optimized Neural Network Weights and Biases Using Particle Swarm Optimization Algorithm for Prediction Applications". J. Korea Multimed. Soc., 20, 1406–1420, 2017.
[14] A. Balsabi, "Some analytical solutions to the general approximation problem for feed forward neural networks". Neural Networks, 6, 991–996, 1993.
[15] R. Yamashita, M. Nishio, and R.K.G. Do et. al., "Convolutional neural networks: an overview and application in radiology". Insights Imaging, 9, 611–629, 2018. https://doi.org/10.1007/s13244-018-0639-9
[16] S.R. Dubey, S. Chakraborty, S.K. Roy, S. Mukherjee, S.K. Singh, and B.B. Chaudhuri, "DiffGrad: An Optimization Method for Convolutional Neural Networks". IEEE Trans. Neural Netw. Learn. Syst., 31, 4500–4511, 2020.
[17] Alex Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network". Physica D: Nonlinear Phenomena, 404, 132306, 2020.
[18] A. Mosavi, M. Salimi, S.F. Ardabili, T. Rabczuk, S. Shamshirband and A.R. Varkonyi-Koczy, "State of the Art of Machine Learning Models in Energy Systems, a Systematic Review". Energies, 12, 1301, 2019.
[19] C. Li, "Biodiversity assessment based on artificial intelligence and neural network algorithms". Microprocess. Microsyst., 79, 103321, 2020.
[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors". Nature, 323, 533–536, 1986.
[21] A. M. Schweidtmann, and A. Mitsos, "Deterministic Global Optimization with Artificial Neural Networks Embedded". J. Optim. Theory Appl., 180, 925–948, 2019.
[22] E. M. L. Beale, "A derivation of conjugate gradients, In: Numerical Methods for Nonlinear Optimization, F. A. Lootsma, (Ed.)", Academic Press, London, 1972.
[23] A. Askarzadeh and A. Rezazadeh, "Artificial neural network training using a new efficient optimization algorithm". Appl. Soft Comput. J., 13, 1206–1213, 2013.
[24] M. Tabassum and K. A. Mathew, "A Genetic Algorithm Analysis towards Optimization solutions". Int. J. Digit. Inf. Wirel. Commun., 4, 124–142, 2014.
[25] A. M. Schweidtmann, and A. Mitsos, "Deterministic Global Optimization with Artificial Neural Networks Embedded." J. Optim. Theory Appl., 180, 925–948, 2019.
[26] M. H. Lin, J. F. Tsai, and C. S. Yu, "A review of deterministic optimization methods in engineering and management". Math. Probl. Eng., 2012, 756023, 2012.
[27] M.T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm". Neural Netw. IEEE Trans., 5, 989–993, 1994.
[28] M. T. Hagan and M. Menhaj, "Training feed-foward networks with the Marquardt algorithm". IEEE Transactions on Neural Networks, 5, 989-993, 1999.
[29] A. Bordes, L. Bottou and P. Gallinari, "SGD-QN: Careful quasi Newton stochastic gradient descent". Journal of Machine Learning Research, 10, 1737–1754, 2009.
[30] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning". Neural Netw., 6, 525–533, 1993.
[31] H. U. Dike, Y. Zhou, K. K. Deveerasetty and Q. Wu, Unsupervised Learning Based On Artificial Neural Network: A Review, 2018 IEEE International Conference on Cyborg and Bionic Systems (CBS), Shenzhen, China, pp. 322-327, 2018. doi: 10.1109/CBS.2018.8612259.
[32] M. Kusy, and R. Zajdel, "Application of Reinforcement Learning Algorithms for the Adaptive Computation of the Smoothing Parameter for Probabilistic Neural Network". IEEE Trans. Neural Netw. Learn. Syst., 26, 2163–2175, 2015.
[33] W. Tian, Z. Liao and J. Zhang, "An optimization of artificial neural network model for predicting chlorophyll dynamics". Ecol. Modell., 364, 42–52, 2017.

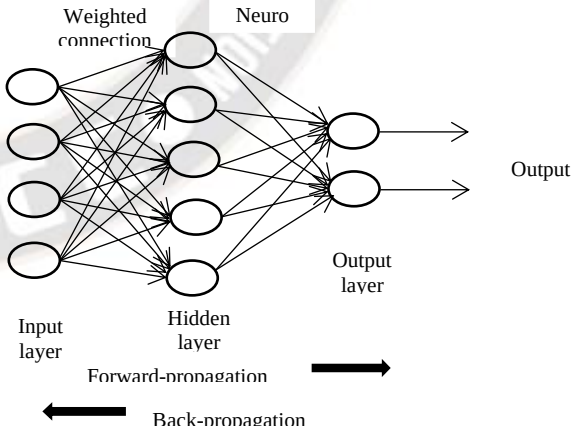

Figure 1. Artificial neural network architectures with feed-forward and backpropagation algorithms