UtilitasMathematica
ISSN 0315-3681 Volume 121, 2024

A Comparative Study of Accuracy and Efficiency Using Artificial Neural Networks for Simulating M^{α}/M/s Queuing Models

Jitendra Kumar[1], SK Bharadwaj[2], DK Mishra[3] Seema Agarwal[4]
[1,2,3]Department of Engineering Mathematics & Computing
Madhav Institute of Technology & Science-DU
Gwalior, Madhya- Pradesh – 474005(INDIA)
[4]Department of Mathematics, SRMIST Delhi- NCR Campus, Ghaziabad
Jitendra Kumar (corresponding author)
E-mail: jkmuthele@mitsgwalior.in &bharadwajsantosh@mitsgwalior.in,
mishradilip3826@mitsgwalior.in
Orcid Id: 0000-0001-5939-5586

Abstract

This study explores the use of artificial neural networks (ANNs) for simulating M^{α}/M/s queuing models, which are essential in queuing theory for analyzing multi-server systems. By employing ANNs, we aim to capture and replicate the complex dynamics of these queuing systems more effectively. The study demonstrates how ANNs can provide accurate and efficient simulations, enhancing the understanding and optimization of queuing processes. We compare the results obtained from ANN simulations with traditional analytical methods to assess their accuracy and efficiency. The findings highlight the potential of ANNs to address the challenges posed by variable arrival and service rates, offering valuable insights for improving decision-making and resource allocation in real-world applications. This research underscores the effectiveness of ANNs in modeling and simulating complex queuing systems, contributing to advancements in both theoretical and practical domains.

Keywords: Queueing model, Queueing theory, ANNs, Model &Simulation, Optimization Algorithm, Computational efficiency.

1. Introduction

Queueing models play vital role in various field, such as telecommunications, manufacturing, and service industries, by providing insights into system performance and resource allocation. Among these models, the M^{α}/M/s queueing model is particularly valuable for its ability to capture non-exponential inter-arrival times,service time, and system capacity. However, simulating M^{α}/M/s queuing systems can be computationally intensive and time- consuming.In this paper, we introduces a new way to simulate queuing systems with multiple servers that handle arrivals in groups, using artificial intelligence (AI) called Artificial Neural Networks (ANNs). ANNs are inspired by the connections in the brain and are adept at handling complex problems in many areas, like image recognition, language translation, and forecasting.

By leveraging the capabilities of ANNs, we aim to develop a simulation model that can accurately capture the dynamics of a queueing system with bulk arrivals and multiple servers. In this paper, we give an innovative approach to simulate a queueing model with bulk arrivals and multiple servers using Artificial Neural Networks. The utilization of ANNs in queueing

simulation opens up new possibilities for understanding and improving the performance of complex systems. The subsequent sections of this paper will delve into the details of the proposed methodology, the training process of the ANN, and the experimental results obtained from applying the model to real-world queueing scenarios.

2. Literature Review

Queueing theory, a branch of mathematics, has been extensively used to model and analyze systems with waiting lines. However, traditional queueing models often fall short of capturing the complexities of real-world systems. Neural networks, on the other hand, offer a powerful tool for modeling non-linear relationships and learning from data. This literature review explores the intersection of queueing theory and neural networks, examining how neural networks can enhance the analysis and prediction of queueing systems with significant contributions from various researchers. **Aboate and Whitt (1987)** explored the transient behavior of the M/M/1 queue, focusing on the system's dynamics when starting from the origin, providing foundational insights into queueing system behavior over time. **Daley (1968)**studied the correlation structure of the output process in single-server queueing systems, offering a deeper understanding of how outputs relate to inputs in these systems. **Finch (1962)** investigated the transient behavior of bulk service queueing systems with finite capacity, further enriching the field's understanding of queue dynamics under varying service structures.

Artificial neural networks (ANNs) have been increasingly applied to model complex systems, including queueing systems. **Flood and Christophilos (1996)** were pioneers in this regard, using ANNs to model construction processes, demonstrating the potential of these networks to simulate complex, real-world processes. **Ertunc and Hosoz (2006)** applied ANN analysis to a refrigeration system, highlighting the versatility of ANNs in different engineering applications. Similarly, **Ghosh-Dastidar and Adeli (2006)** utilized a neural network-wavelet micro simulation model to estimate delays and queue lengths at freeway work zones, showing the effectiveness of hybrid models in traffic engineering**.Hagan etal. (1996)** provided a comprehensive introduction to the theoretical foundations and practical implementations of neural networks and also covered key topics such as network architectures, learning algorithms, and practical applications, making it an essential resource for understanding how ANNs can be designed and applied to various problems. **Karlin and McGregor (1958)** provided fundamental insights into the behavior of such queueing systems, particularly in terms of the distribution of queue lengths and waiting times. This paper is foundational in the field of queueing theory and has influenced a wide range of subsequent research in both theoretical and applied mathematics, particularly in operations research and telecommunications. **Abdelkader (2011)** presented a novel approach to evaluating the performance of queueing models using the method of order statistics. This technique allows for the computation of various performance measures, such as waiting times and queue lengths, with a higher degree of accuracy compared to traditional methods. **Bahadorietal (2014)** studied the particularly relevant for healthcare management, as it provides a practical framework for optimizing resource allocation in hospital settings. The findings highlight the potential of queueing theory as a tool for enhancing the operational efficiency of healthcare services, contributing to better patient outcomes and resource utilization. **Bonald (2007)** discussed insensitive traffic models in communication networks, emphasizing the need for robust modeling techniques. This concept resonates with various discrete event modeling approaches in traffic systems, which can enhance our understanding of network

dynamics.The integration of ANNs into queueing theory has been further explored by **Khoshnevis and Parisay (1993),** who demonstrated how machine learning techniques could enhance simulation models of queueing systems. **Hermanto and Nugroho (2018)** developed a neural network model to estimate waiting times in bank customer queues, showcasing the practical application of ANNs in service industries. In the healthcare domain, **Palvannan and Teow (2012)** addressed the challenges of queueing in healthcare systems, underscoring the importance of efficient queue management in critical sectors.Recent advancements have focused on optimizing ANN architectures for specific queueing scenarios. **Pornthep and Varis (2020)** designed an advanced queue system using ANNs to predict waiting times, reflecting the ongoing innovation in this field. **Sundari and Palaniammal (2015)** simulated M/M/1 queueing systems using ANNs, providing a framework for applying neural networks to classical queueing models . **Yamini et al. (2020)** extended this work by simulating Markovianqueueing models in busy airports, demonstrating the applicability of ANNs in high-demand environments.

3. Queueing Model

Queueing models are mathematical models that are used to study and analyze waiting lines or queues. They provide valuable insights into how queues form, how they behave, and how to optimize their performance.

Queueing models are widely applied in various fields, including telecommunications, transportation, healthcare, and customer service. By studying the behavior of queues, we can better understand and improve the efficiency of systems that involve waiting lines.A Queueing model is the arrival process, which determines how customers or entities enter the queue. It can follow different patterns, such as a Poisson process, where arrivals occur randomly over time, or a deterministic pattern, where arrivals happen at regular intervals.Another important aspect is the service process, which defines how customers are served once they reach the front of the queue. The service process can have different distributions, such as exponential, constant, or general distributions, depending on the nature of the system being modeled.

Queueing models help us measure various performance metrics, such as the average waiting time, queue length, and the utilization of resources. These metrics allow us to evaluate the effectiveness of different strategies, such as adding more servers, implementing priority rules, or adjusting the arrival pattern. Analyzing queueing models can be done using mathematical techniques, simulation, or a combination of both. Through these approaches, we can predict system behavior, make informed decisions, and optimize processes to provide better service to customers and enhance overall system efficiency. Finally, we say, the queueing models offer a powerful tool for understanding and improving systems with waiting lines. By applying these models, we can gain valuable insights into the behavior of queues and develop strategies to enhance performance. Queueing models have traditionally been used to analyze and optimize bulk arrival systems with multiple servers. Queueing models involve complex mathematical calculations and often rely on assumptions that may not hold in real- world scenarios, such as exponentially distributed inter-arrival times and service times. As a result, their accuracy and applicability to dynamic and diverse systems.

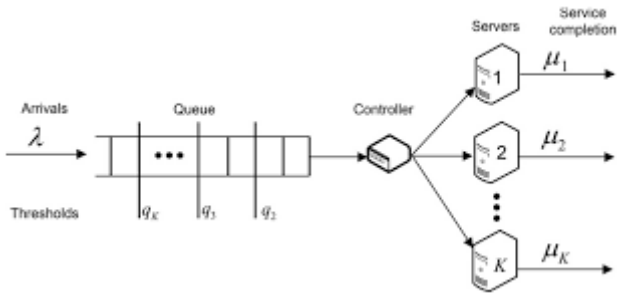

Fig. 1: Queueing model with multi-server

4. Artificial Neural Networks (ANN)

Artificial Neural Networks (ANNs) are computational models inspired by the human brain's network of neurons. They consist of layers of interconnected nodes, called neurons, which are organized into an input layer, one or more hidden layers, and an output layer. The neurons in each layer are connected by synapses, which have adjustable weights that are fine-tuned during training.

In an ANN, data is input into the system through the input layer, where each neuron represents a different input data feature. This data is then processed through one or more hidden layers. The hidden layers transform the inputs into more abstract representations, allowing the network to learn complex patterns and relationships. Each neuron in the hidden layer applies a weighted sum of its inputs and passes the result through an activation function, such as the sigmoid, tanh, or ReLU (Rectified Linear Unit), introducing non-linearity and enabling the network to capture more intricate patterns.

The output layer produces the final result of the network, which could be a classification label, a predicted value, or other forms of output depending on the task at hand. The process of determining the output involves forward propagation, where data flows through the network, and the outputs are calculated based on the current weights of the connections.

Training an ANN involves adjusting the weights of the connections to minimize the error between the predicted outputs and the actual targets. This is achieved through a process called back propagation, combined with an optimization technique like gradient descent. During back propagation, the error is calculated by a loss function, which is then propagated backward through the network. The gradients of the loss function concerning each weight are computed, and the weights are updated in the opposite direction of the gradient to reduce the error.

ANNs are widely used across various domains due to their ability to learn and generalize from data. In image and speech recognition, Convolutional Neural Networks (CNNs), a specialized type of ANN, are used to automatically and accurately identify patterns. In natural language processing, Recurrent Neural Networks (RNNs), including Long Short-Term Memory (LSTM) networks, are employed for tasks like translation and sentiment analysis. In healthcare, ANNs are applied to diagnose diseases, predict patient outcomes, and personalize treatment plans. In finance, they are used for credit scoring, fraud detection, and stock market prediction.

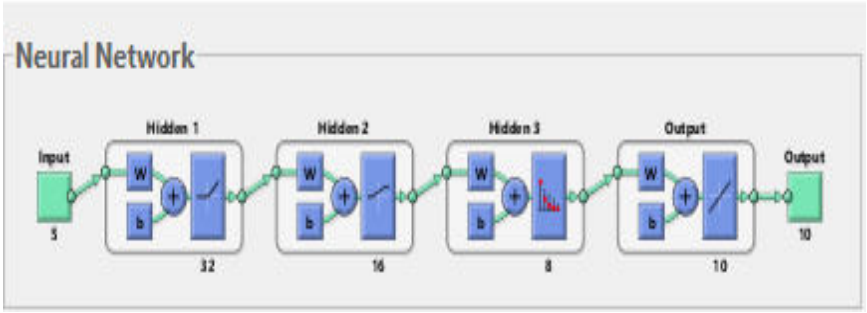

Figure 2:Artificial Neural Networks

The performance of the ANN-based expectation is assessed through regression analysis by comparing the network's expected parameters with the actual target values. Three primary criteria used in this assessment are the correlation coefficient, mean relative error and root mean square error. The correlation coefficient indicates how well the variations in the expected outputs align with the actual values. In this paper, we calculate the coefficient between the expected and the actual output, which is defined as follows

$$R(x,y) = \frac{n\sum xy-(\sum x)(\sum y)}{n\sum x^2-(\sum x)^2} \quad (1.1)$$

The correlation coefficient ranges from -1 to +1, where values closer to +1 indicate a stronger positive linear relationship and values closer to -1 indicate a stronger negative linear relationship. The mean relative error, which shows the average ratio between the errors and the experimental values, is calculated as follows:

$$MRE\ (\%) = \frac{1}{N}\sum_{i=1}^{N} 100\ \frac{(x_i - y_i)}{x_i} \quad (1.2)$$

Where N is the total number of data points in the set.

Finally, the root mean square error is defined as

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2} \quad (1.3)$$

While neural networks offer a promising approach to approximating queuing systems, their design remains a complex task. While inputs and outputs are application-specific, key factors like hidden layers, neurons, activation functions, and training iterations require empirical exploration. Despite progress in understanding neural network behavior, there are no definitive guidelines for network topology and parameter selection. This often involves experimentation and leveraging problem-specific insights. In this paper, we utilized queuing theory formulas to generate training data for our neural network, as detailed in the following section.

5. M^{α}/M/sQueueing model Simulation with ANN

ANNs with bulk arrival with multiple servers Queueing model simulations, we can develop more accurate models to understand and optimize the performance of such systems. ANNs can learn from historical data to identify patterns and relationships between input parameters and output measures, such as queue lengths, waiting times, and server utilization. This enables us to predict system behavior and evaluate different strategies for improving performance, such as adjusting the number of servers or optimizing service policies. In this paper, we present a detailed methodology for training and utilizing ANNs to simulate M^{α}/M/s queue models. Our approach involves the generation of synthetic data sets, network architecture design, and training using appropriate optimization algorithms. We evaluate the performance of the ANN –based

simulation against traditional simulation techniques, demonstrating its effectiveness in capturing the dynamic behavior of M^{α}/M/s queueing models.

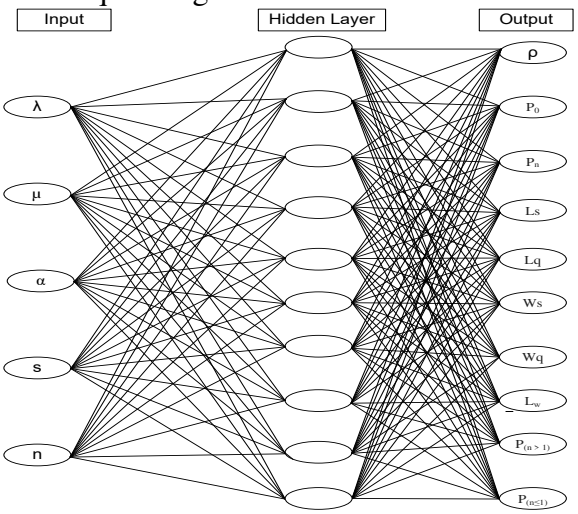

Figure 3: Architecture of multi-server Queieng Model

A queueing system is specified when we know (i) the input, (ii) the queue discipline and (iii) the service mechanism with notation by Kendall proposed queueing models using four factors written A/B/C/α/K/N/Dwhere A represents the inter-arrival time distribution, α is batch size (bulk), B represents the service time distribution, c represents the number of servers and also D is the queue discipline, Kis the number of places in the system, Mis the whole populationfor M^{α}/M/s queue model, it is assumed K = ∞ , M= ∞, D = FCFS.

There are four types of inputs α, D, M and E_k

D(deterministic, or regular):

$$f(x) = \begin{cases} 0\,, when & (x < a) \\ 1\,, when & (x \geq a) \end{cases} \quad (5.1)$$

M (random, or Poissonian):

$$f\,(x) = 1 - Exp\,(-x/a) \quad (5.2)$$

With a D-type input, customers arrive at regular time intervals, whereas, with an M-type input, customer arrivals are random and thethirdintermediatetypeofinputisE_k.

$$E_k(Erlangain): \quad d\,f(x) = \frac{(k/a)^k}{\Gamma(k)} \exp(-\frac{kx}{a})\, x^{k-i}\, dx \quad (5.3)$$

The distribution of arrival and service times is categorized similarly using the symbols D, M, and Ek. With a D distribution for service time, each customer is served for a constant duration. In contrast, an m distribution for service time follows a negative exponential law. The Ek distribution represents an intermediate form. By using these conventions, a queuing system can be identified with a label such as M/M/1 (Poisson arrivals; negative exponential service time; single server). Erlang (1908-29) introduced models like M/M/S, M/D/S, and M/Ek/1. Pollaczek (1930) reported the Ek/G/1 queuing system, while Khintchine (1932) and Kendall (1951) presented the M/G/1 model. Numerous other authors have explored various queuing models and conducted extensive mathematical analyses to solve these complex models. The equations for finding solutions are highly intricate and time-consuming. This study aims to simulate these complex queuing models through computer simulation. Specifically, the Neural Network Toolbox in MATLAB is utilized to develop the ANN model for the M^{α}/M/s queuing system.

Simulation involves conducting experiments with a model of the system being studied; it is a powerful tool for analyzing and designing engineering and natural systems. It is an iterative input-output process, where feedback guides adjustments to input parameters. Inputs represent potential real-world events and conditions, while outputs predict the system's response. For simulating the $M^{\alpha}/M/s$ queuing model, input and output values are analytically determined using Kendall and Little's formulas and then used to train the ANN. In this paper, six inputs are used to train the ANN model: (i) arrival rate 'λ', (ii) service rate 'μ', (iii) number of customers exceeding 'k', (iv) time't', (v) population size 'n', and (vi) group size 'α'. Ten queuing model properties, calculated using equations (6.1 to 6.10), are targeted as outputs.

6. Mathematical formulas for queueing model

The following formulas can be used to calculate the probabilities and performance metrics in the bulk arrival with multiple server Queueing model:

These formulas provide useful measures for evaluating the performance of bulk arrival queueing models with multiple servers. The actual calculations and specific formulas may vary depending on the assumptions and characteristics of the queueing system being analyzed.

1. Utilization factor:

   ρ - Utilization factor of the system, defined as the ratio of the total arrival rate with size of group to the total service rate with no. of servers

   $$\rho = \frac{\alpha.\lambda}{s.\mu} \quad (6.1)$$

2. Probability of no arrivals (empty system):

   $$P_0 = \sum_{k=0}^{s-i} \frac{\rho^k}{k!} + \frac{\rho^s}{s!} * \left(\frac{s*\mu}{(s*\mu - \alpha\lambda)}\right)^{-1} \quad (6.2)$$

3. Probability of having n customers in the system ($0 \leq n \leq s$): $P_n = \frac{\rho^n}{n!} * P_0$ (6.3)
4. Mean number of customers in the system: $L_s = \sum_{n=0}^{s} n * P_n$ (6.4)
5. Mean number of customers in the queue: $L_q = L - \rho$ (6.5)
6. Mean waiting time in the system: $W_s = {}^{L}/_{\{\alpha\lambda * (1 - P_0)\}}$ (6.6)
7. Mean waiting time in the queue: $W_q = {}^{L_q}/_{\{\alpha\lambda * (1 - P_0)\}}$ (6.7)
8. Average queue length

   $$L_w = \frac{s.\mu}{(s.\mu - \alpha\lambda)} \quad (6.8)$$

9. Probability of 'n' customers in the system

   $$P_n = (1 - \rho\,)\rho^n \quad (6.9)$$

10. Probability that at -most one customers in the system

    $$P(n \leq 1) = P_0 + P_1 \quad (6.10)$$

To develop the ANN for the queuing system, the dataset from the analytical work was divided into training, validation, and test sets. The dataset includes 210 input-output pairs, with 70%

randomly assigned to the training set, 15% for validation, and the remaining 15% for testing the network. The architecture of the ANN, along with the input and output parameters for the queuing system, is shown schematically in Fig. 2.

7. Algorithm:

Initialize

- Set the parameters for data generation
- Initialize arrays to store data and outputs

Data Generation

- Generate random data
- Arrival rate per individual (lambda)
- Service rate per server (mu)
- Number of servers (c)
- Number of customers (n)
- Batch size or bulk (alpha)
- Calculate utilization factor (rho)
- Calculate average queue length(L_w)
- Calculate probability of no arrivals (P_o)
- Calculate probability of having n customers in the system (P_n)
- Calculate mean number of customers in system (L_s)
- Calculate mean number of customers in the queue (L_q)
- Calculate mean waiting time in the system (W_s)
- Calculate mean waiting time in the queue (W_q)
- Calculate probability of n customers in the system (P_n_system)
- Calculate probability that at most one customer is in the system (P_st_most_one)
- Store output metrics in the output array

Combine Data

- Combine the inputs and outputs into a single dataset

Data preprocessing

- Load the dataset generated
- Split the dataset into training and testing sets
- Normalize input data to have zero mean and unit variance for both training and training sets

Neural Network Configuration

- Define the neural network architecture:
  - Feedforward network of three layers having neurons 32, 16 and 8 respectively
- Set activation functions for hidden layers:
  - For first layer = ‘poslin’
  - For second layer = ‘logsig’
  - For third layer = ‘softmax’

Training Parameters

- Set training parameters:
  - Train function = ‘Levenberg-Marquardt’

- Learning rate = 0.001
- No of epochs = 1000
- Minimum gradient threshold = 1e-10

- Train the neural network using the training data

Evaluation

- Evaluate the trained model on the test data
- Calculate the test loss for each output
- Display the test loss for each output

8. Result and Discussions

In this section, we discuss the construction and training of an Artificial Neural Network (ANN) model with a double hidden layer, as depicted in Figure 2. The input layer comprised 5 neurons corresponding to the input parameters, while the output layer contained 10 neurons aligning with the output parameters. The optimal number of hidden layers, the number of neurons in each hidden layer, and the transfer functions between layers were determined through a systematic trial and error approach. Regression analysis was used to evaluate the performance of the developed ANN for a specific problem. The simulation process continued until the simulated output values closely matched or equaled the target values within the predetermined error threshold, marking the completion of the simulation.

This paper evaluates the performance of an ANN model with two hidden layers using two metrics: the minimum number of epochs required to achieve the desired error value, and the ANN's ability to reach the lowest possible error value for a given epoch. The ANN model was trained with varying numbers of neurons in the hidden layer, ranging from 1 to 1000, over 1000 epochs. The performance of each ANN model was analyzed by comparing the error values achieved at 1000 epochs and through regression analysis. The most effective way to measure the performance of an ANN model, along with regression analysis, is to keep the number of epochs fixed and compare the minimum error values the ANN models can achieve.

Figure 3 illustrates the performance of the ANN model with 5 neurons in the hidden layer. Initially, the model shows significant improvement, with the error value decreasing to 0.8 by 200 epochs. However, the rate of progress slows considerably after this point, and by 500 epochs, the error value stabilizes. Ultimately, the model reaches an error value of 0.001 after 1000 epochs, but no further reduction in error is observed beyond 500 epochs. This plateau suggests that while the model initially learns and adapts quickly, it encounters diminishing returns in terms of error reduction in the latter stages of training. This behavior highlights the importance of optimizing the training process and exploring potential adjustments in network architecture or training parameters to achieve better performance.

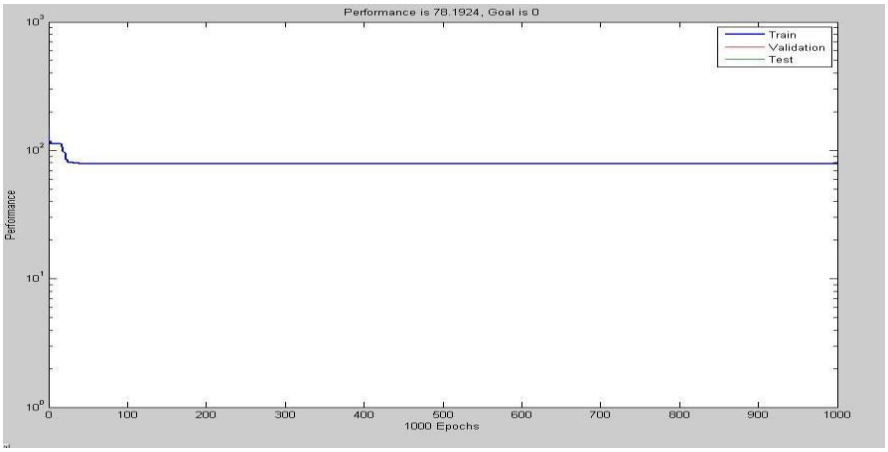

Figure 4: Training curve of 32 neurons

Figure 4 illustrates that the ANN model with 32 neurons in the hidden layer concluded training with the highest error value of 78.

Figure5:Training curve of 32 neurons

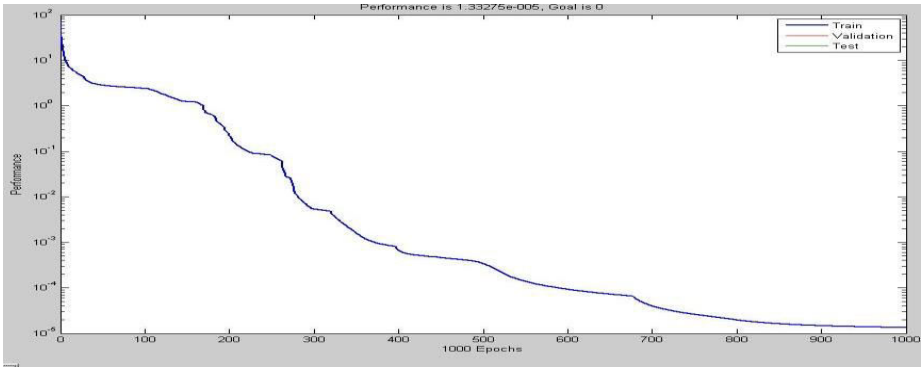

Figure 5 shows that the ANN model with 32 neurons in the hidden layer reaches the lowest error value after 1000 epochs and continues to converge effectively. Adding 32 or more neurons to the hidden layer does not result in significant performance improvements. Based on the performance analysis of different ANN models, the model with 32 neurons in the hidden layer is chosen for validation and testing with the available data, as it achieves the minimum error.

Figure 6 depicts the regression analysis trend for the training, validation, and testing outputs concerning the target values. A correlation coefficient (R) near +1 signifies a robust, positive linear association with the expected outcomes. Both the mean relative error and root mean square error fall within acceptable thresholds. Furthermore, the convergence of the training, validation, and testing curves in Figure 6 indicates the suitability of this neural network model for simulating M^{α}/M/s queuing model problems.

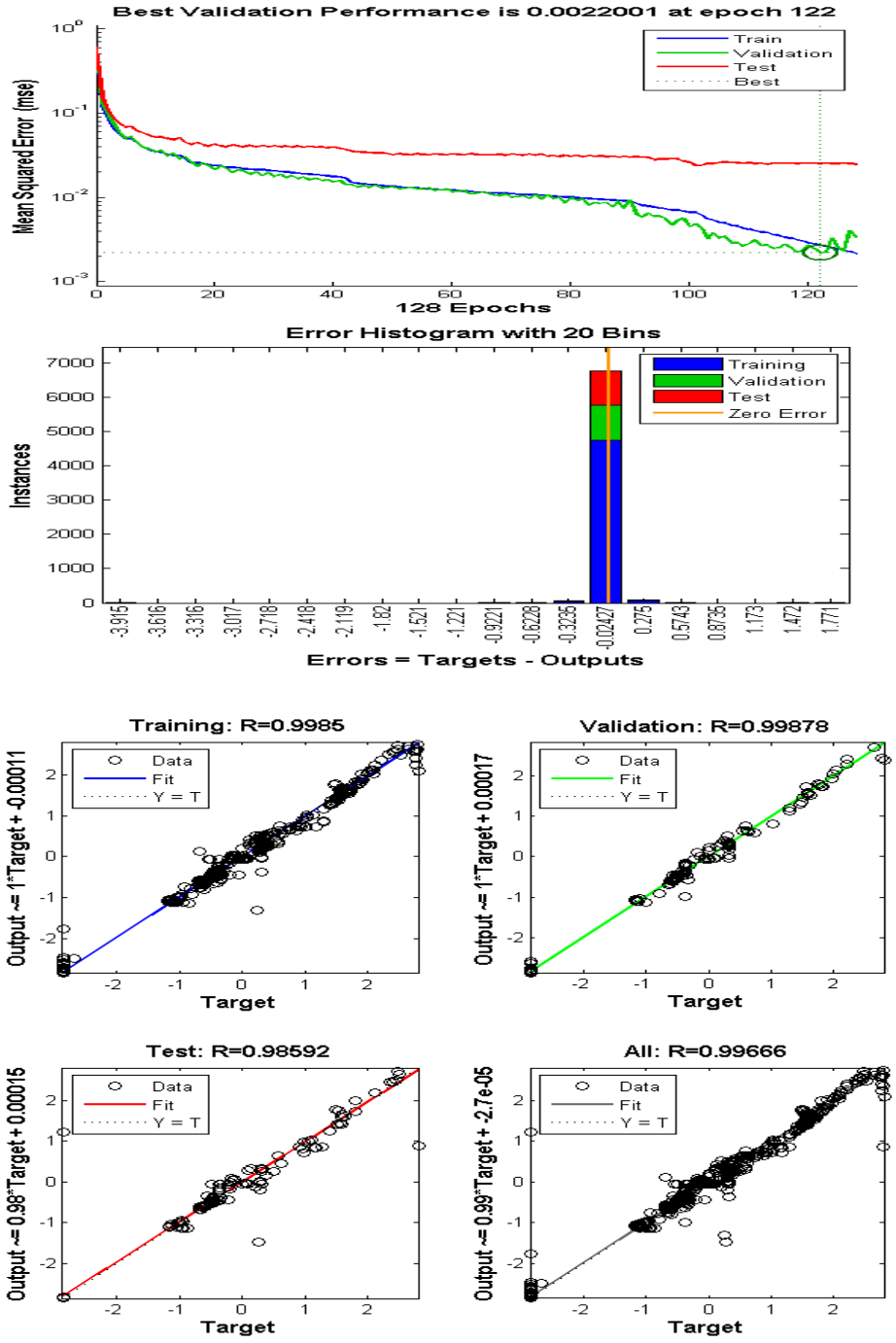

Figure6:Manner of Regression Analysis

Conclusion

An artificial neural network (ANN) model capable of simulating the $M^{\alpha}/M/s$ queuing model has been developed using a feed forward back propagation algorithm with a double hidden layer comprising 32 neurons. The results indicate that the simulated values produced by the ANN

closely align with those calculated using the classical mathematical approach. Regression analysis confirms the strong correlation between the ANN outputs and the expected values, with a correlation coefficient near +1. Additionally, the mean relative error and root mean square error (RMSE) fall within acceptable limits, further validating the accuracy and reliability of the ANN model for this application. This demonstrates the ANN's potential as a robust tool for effectively simulating complex queuing systems.

References

1. J. Aboate and W. Whitt (1987) Transient behavior of the M/M/1 queue: Starting at the origin, Queueing Systems, Vol. 2, pp. 41-65.
2. D. J. Daley (1968) The Correlation Structure of the output Process of Some Single Server Queueing Systems, The Annals of Mathematical Statistics, Vol. 39(3), pp. 1007-1019.
3. H.M. Ertunc and M. Hosoz (2006) Artificial neural network analysis of a refrigeration system with an evaporation condenser, Applied Thermal Engineering, Vol. 26, pp. 627-635.
4. P. D. Finch (1962) On the Transient Behaviour of a Quening system with Bulk Service and Finite Capacity, The Annals of Mathematical Statistics, Vol. 33(3), pp. 973-985.
5. I. Flood and P. Christophilos (1996) Modeling construction process using artificial neural networks, Automation in Construction, Vol. 4(4), pp. 307-320.
6. S. Ghosh-Dastidar and H. Adeli (2006) Neural Network-Wavelet Micro simulation Model for Delay and Queue Length Estimation at Freeway Work Zones, Journal of Transportation Engineering, Vol. 132 (4), pp. 331-341.
7. M.T. Hagan, H.B. Demuth and M Beale (1996) Neural Network Design, PWS Publishing Company, Boston, MA.
8. S. Karlin and J. Mc Gregor (1958) Many server queueing Processes, with Poisson input and exponential service times, Pacific Journal of Mathematics, Vol. 8(1), pp. 87-118.
9. Y. H. Al-Wohaibi M. Abdelkader (2011) Computing the Performance Measures in Queueing Models via the Method of Order Statistics. Journal of Applied Mathematics, 2011, 1–12. 10.1155/2011/790253.
10. M. Bahadori,S. M. Mohammadnejhad,R. Ravangard and E. Teymourzadeh (2014) Using Queueing Theory and Simulation Model to Optimize Hospital Pharmacy Performance. Iranian Red Crescent Medical Journal, 16(3). Advance online publication. 10.5812/ircmj.1680724829791.
11. T. Bonald (2007) Insensitive Traffic Models for Communication Networks. Discrete Event Dynamic Systems, 17(3), 405–421. 10.1007/s10626-007-0012-5.
12. R. P. S. Hermanto,A. Nugroho (2018) Waiting-time estimation in bank customer queues using RPROP neural networks. Procedia Computer Science, 135, 35–42. 10.1016/j.procs.2018.08.147.
13. B. Khoshnevis and S. Parisay (1993) Machine Learning and Simulation: Application in Queueing Systems. Simulation, 61(5), 294–302. 10.1177/003754979306100502.
14. A.I. Kyritsis and M. Deriaz (2019) A Machine Learning Approach to Waiting Time Prediction in Queueing Scenarios. Second International Conference on Artificial Intelligence for Industries. 10.1109/AI4I46381.2019.00013.
15. X. Li Wang and C. Yang (2022) Risk prediction in financial management of listed companies based on optimized BP neural network under digital economy. Neural Computing & Applications, 1–14.

16. R. K. Palvannan and K. L. Teow (2012). Queueing for healthcare. Journal of Medical Systems, 36(2), 541–547. 10.1007/s10916-010-9499-720703697.
17. A. Pornthep and L. Varis (2020) Design of Advanced Queue System Using Artificial Neural Network for Waiting Time Prediction. Revista ESPACIOS, 41(40), 111–124..
18. A. Puhalskii and J. E. Reed (2010) On many-server queues in heavy traffic. Annals of Applied Probabilities, 20(1), 129–195.
19. M. S. Sundari and S. Palaniammal (2015) Simulation of M/M/1 queuing system using ANN. Malaya Journal of Mathematik, S (1), 279–294.
20. S. Yamini,K. Rath and S. Palaniammal (2020) Artificial Neural Network simulation for Markovian Queuing Models in a busy airport. In 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA) (pp. 1-6). IEEE.