Nanotechnology Perceptions
ISSN 1660-6795
www.nano-*ntp.com*

Performance Assessment Of Neural Networks In Wireless Sensor Networks Affected By Delays And Packet Drops

Ravinder Gaja[1] , Dr. Mukesh Tiwari[2]

[1] *Research Scholar, Department of Electronics & Communication, Sri Satya Sai University of Technology & Medical Sciences, Sehore, M.P, India.*
[2] *Research Supervisor, Department of Electronics & Communication, Sri Satya Sai University of Technology & Medical Sciences, Sehore, M.P, India.*

Delays and dropouts impact the transmission of outputs and error signals across the layers of the multi-layer perceptron (MLP) network, which occurs during forward propagation, back propagation, and weight updates during training. In order to simulate the lag time that comes with wireless connection, the delay parameter (twait) is changed on the fly depending on the network architecture and a Gaussian distribution. This changes the way the neurons calculate their output. There is a gateway mote for direct contact with other nodes and differences in latency between node pairs are also taken into account in the simulation. The simulation is run numerous times with varying starting weights to analyze the influence of communication delays on the neural network's performance throughout the training phase. Seven datasets were used from the UCI Machine Learning repository.

Keywords: Sensor networks, Delay, Wireless, Communication, Drop.

I. INTRODUCTION

The confluence of neural networks (NNs) with wireless sensor networks (WSNs) in recent years has opened new vistas for intelligent, adaptable, and robust systems capable of real-time data processing and decision-making. Comprising spatially distributed autonomous sensors monitoring physical or environmental conditions including temperature, humidity, pressure, and motion, wireless sensor networks have become absolutely necessary in a great variety of applications including environmental monitoring, industrial automation, smart agriculture, healthcare, and military surveillance. A WSN's main purpose is to detect, analyze, and send data to a central base station or sink node for further analysis. WSNs, on the other hand, often run under tight limits including low energy, bandwidth, and processing power as well as difficult climatic conditions causing network delay and packet loss. When combining intelligent models like neural networks, which need a specific degree of computational consistency and communication stability to produce correct outcomes, these limitations become very important.

Inspired by the structure and operation of the human brain, neural networks can simulate complicated nonlinear connections, learn from input, and adjust to changing settings. Their uses in WSNs are many and varied, from data fusion, event detection, pattern recognition, fault diagnostics, to energy-efficient routing and node location. By letting WSNs handle raw sensor data locally, hence lowering the communication load and preserving energy, neural networks may improve their decision-making ability. But in actual WSN situations, neural networks have particular difficulties resulting from the very essence of wireless communication—specifically, latency and packet failures.

Various elements include medium access control (MAC) layer contention, retransmissions caused by interference or fading, node congestion, and network topology changes cause communication delays in WSNs. Such delays might cause asynchronous or out-of-date input to enter the neural network, therefore impairing its function. Furthermore, delays might interfere with the temporal correlations on which neural networks, particularly recurrent neural networks (RNNs) or temporal convolutional networks (TCNs), rely for correct predictions. Even little data aggregation delays might cause late or erroneous alarms in applications like structural health monitoring or fire detection, which could have disastrous effects. Thus, the design of neural network models for WSNs has to take into account the temporal dependability of input data streams and provide strong methods for managing delays.

On the other side, packet drops are the loss of data packets during transmission, a common event in WSNs caused by interference, low signal strength, restricted transmission range, and battery depletion. Missing information caused by packet loss might result in discrepancies and errors in the input supplied to the neural networks. Applications with little sensor data or where every sensor reading is very important make the problem worse. In a healthcare monitoring system, for instance, the loss of crucial signals like heart rate or blood pressure data may mislead the neural network and endanger patient safety. Packet losses during model training could also affect the learning phase of the neural network, therefore causing poor generalization and worse predicting accuracy.

Delays and packet losses have a twofold effect that calls for further investigation on the fault-tolerance and resilience of neural networks implemented in WSNs. Traditional neural networks presuppose total and timely availability of input data, an assumption that does not apply in actual WSN installations. Therefore, there is an increasing demand for the creation of loss-resilient and delay-tolerant neural network designs. Active research is being done on methods like data imputation, temporal interpolation, dropout-aware training, and resilient optimization to reduce the consequences of data loss and delay. Moreover under consideration are hybrid models combining neural networks with probabilistic reasoning systems including Bayesian networks or Kalman filters to handle uncertainty in data.

II. REVIEW OF LITERATURE

Isabona, Joseph et al., (2022) The RMSE values of 8 to 12 dB for traditional models' route loss forecasts significantly exceed the tolerable error range of 0 to 6 dB, as shown in many studies. Achieving this aim requires very precise route loss prediction models using machine learning. This study establishes an innovative route loss model with multi-layer perceptron (MLP)

neural networks, employing a grid search methodology for hyperparameter optimization. The model has a well organized implementation network architecture. The suggested model was developed to provide the most accurate estimate of route loss between the base station and the mobile station. All hyperparameters, including the learning rate, number of hidden layers, and neuron count, are taken into account. The hyperparameters of the proposed MLP model have been optimized, and its predictive accuracy evaluated by various learning and training techniques on comprehensive route loss experimental datasets. The experimental route loss data are obtained by doing a field driving test in an urban microcellular environment over a functioning 4G LTE network. The outcomes were evaluated using several first-order statistical performance criteria. The results indicate a positive correlation with the measured data, and the prediction errors of the proposed MLP model exceeded those produced by traditional log-distance-based route loss models.

Weissbart, Léo. (2020). The most effective findings in profiling side-channel analysis are now produced by analyses that are based on machine learning. Methods belonging to the family of neural networks, including multilayer perceptrons and convolutional neural networks, exhibit this property to a heightened degree. In most cases, convolutional neural networks are used because of their superior performance when confronted with targets that are shielded by countermeasures. Since most research just compare multilayer perceptrons to convolutional neural networks, it is clear that these networks are under-discussed in the scholarly literature. The multilayer perceptron, on the other hand, has a design that is rather simple, which makes changing the hyperparameters easy and, perhaps, makes the inner workings of this neural network more explainable.

Kumar, Shiu et al., (2016) As Wireless Sensor Networks become more prevalent in manufacturing, new research opportunities are emerging. Among the many critical and challenging applications is node localization. This research employs a method that is centered on feed-forward neural networks. The RSSI readings provided by the anchor node beacons are used for this purpose. In addition, the research delves into the ways in which the configuration and quantity of anchor nodes impact the localization system's accuracy. To determine which training algorithm produces the optimal outcomes, five distinct algorithms are assessed. The multi-layer Perceptron (MLP) neural network model was trained using Matlab. The efficacy of the suggested approach may be assessed in real-time when the model is written into the Arduino microcontroller. In a 12-12-2 neural network configuration, four anchor nodes produced an average two-dimensional localization error of 0.2953 meters. The suggested approach is suitable for any embedded microcontroller system.

Abu Alsheikh, Mohammad et al., (2014) wireless sensor networks detect environments that are always changing. This dynamic behavior could be triggered by either external factors or the designers of the system. Sensor networks often use machine learning techniques to adapt to various scenarios while avoiding unnecessary redesign. In addition to generating many practical ideas, machine learning also optimizes resource efficiency and increases the lifespan of the network. By comparing each algorithm to the current scenario, we are able to balance their advantages and disadvantages. Furthermore, we provide a comparative reference to assist WSN designers in developing efficient machine learning solutions tailored to their specific application challenges.

Serpen, Gursel et al., (2013) This research proposes using artificial neural network technologies to provide "intelligent computation" and "adaptation" capabilities inside the network, which would increase the value, usefulness, and survivability of wireless sensor networks. Our goal is to provide wireless sensor networks with AI capabilities that will allow them to gain knowledge from their experiences and adjust to different field conditions. Wireless sensor networks are notoriously difficult to work with due to their many peculiarities. Considerations such as a constantly shifting topology, a dense deployment of sensor nodes, and, most importantly, limited power, compute, storage, and communication are present. In addition to being durable, scalable, and energy efficient, the protocols and applications running on wireless sensor networks need to be intelligent and able to "adapt" to new situations, application scopes, and uses. The proposed technique is shown in a simulation-based case study to be effective by clustering data from the Breast Cancer Wisconsin research utilizing a wireless sensor network integrated with Kohonen's self-organizing map neural network.

G. Soares Alcalá, Symone et al., (2010) Wireless sensor networks (WSN) are a relatively new technology that could have several real-world applications. In the meantime, ANNs have found abundant usage in a variety of productive domains. Several similarities exist between WSN and ANN. The sensor node might be likened to a neuron, since wireless sensor network (WSN) applications exhibit traits such as distributed processing, extensive parallelism, adaptability, fault tolerance, and minimal computational requirements. Our focus here is on the Smart Table and how it may benefit from ANN and WSN integration. The incorporation of ANN models onto reasonably priced System-on-a-Chip (SoC) devices has been shown via prototypes.

III. EXPERIMENTAL SETUP

The effect of wireless communication on neural network performance in WSNs was modeled using a proprietary C++ simulator. Message latency and packet drop were the particular communication issues studied.

To simplify things, the simulator doesn't worry about complex networking specifics; instead, it focuses on recreating transmission delays and dips that have an impact on neuron outputs. Training neural networks with this architecture guarantees excellent computational efficiency while faithfully reflecting limitations imposed by wireless communication.

Accessing data, initializing, instantiating delay/drop behavior, training MLPs, and evaluating performance are all steps in the simulation process. Training consists of the usual procedures, including forward and backpropagation, as well as weight updates.

It is possible for motes to miss or delay neuron outputs or error messages when training. You won't see these impacts represented when testing; they're reserved for training. Communication between motes has unpredictable transmission delays when neurons are placed across them. To determine how long a neuron should wait for inputs, the simulator provides it a waiting time parameter (twait). The formula for this is:

$$t_{wait} = \vartheta \times \mu \times L_{max} \quad (1)$$

L_{max} is the maximum distance that covers 90% of mote pairs in the WSN, ϑ is an experimentally selected constant, and μ is the mean of a truncated Gaussian delay distribution. Different delay/drop circumstances are simulated by varying the value of ϑ from 0.3 to 2.1.

The model presupposes a centralised gateway mote that can communicate with all other motes via a single hop, without any delay or drop in messages that originate from the gateway.

Seven UCI datasets (Breast Cancer Wisconsin, Digits, Adult, Wine Quality, Pima Indians Diabetes, Heart Disease, and Abalone) were used and each experiment was repeated five times with different initial weights to ensure reliability.

IV. RESULTS AND DISCUSSION

Time Complexity

Let us assume that the training set contains patterns of $|P_T|$ and that the validation or testing set contains patterns of $|P_V|$. The MLP network, which has one hidden layer and one output layer, is fed patterns from the training set PT in a very sequential fashion. Through distributed (and asynchronous) processing, each pattern is processed in parallel at the level of the individual neuron. The cumulative delay, t_{wait}, mostly influenced by delays associated with MAC and routing protocol requirements, may include the processing time for individual neurons. A lot of variables determine the value of this delay parameter, which is a random variable. The value of the random variable niter, which is defined as the number of iterations required to converge to a solution, depends on various factors such as the training algorithm, initial weight and parameter values, stopping criterion, data set characteristics, and presentation order. Equation following yields an approximation of the temporal complexity, TC, of a WSN-MLP design:

$$TC = n_{tier} \times (|P_T| + |P_V|) \times E\{t_{wait}\} \quad (2)$$

Where E {} is the expected value operator. t_{wait} the simulation uses Equation 1 and sets the mean of the truncated Gaussian distribution Pi to 1. Multiplying Pn by the per-hop delay yields the real time. The expected per-hop delay is 65 ms, with a range of 226–226 ms. The simulation research parameter set was employed for TC calculation. Take the iteration count as an example; it was determined by averaging all the variables in the dataset. Also, by averaging the results from each trial, we were able to get the average value for the simulation time. Table 1 displays related data for all datasets; all other parameters have constant values because they were used in the simulation research.

Table 1: Factors influencing time complexity in WSN-MLP design

Dataset	Iterations	patterns	Hidden neurons	Output neurons	t_{wait}(ms)	Simulation duration (hrs)

Breast Cancer Wisconsin	179	150	4	3	186.8	1.50
Wine Quality	194	175	9	3	234.3	1.74
Digits	126	351	11	2	218.7	2.31
Adult	139	358	16	6	311.7	4.08
Pima Indians Diabetes	113	2000	53	10	467.6	28.94
Heart Disease	170	7797	131	26	702.3	261.88
Abalone	106	4667	141	2	577.6	119.86

According to Table 1, the iteration count values exhibit little variation. The quantity of patterns fluctuates, and hence, the neuron count in the hidden layer likewise differs across various datasets. The quantity of designs fluctuates by an order of magnitude from 150 to 7,797: The primary determinant influencing the value of temporal complexity TC is, in fact, this element. The change in the number of hidden neurons and output neurons will result in a modification of the parameter L_{max}, thereby increasing the value of t_{wait}.

Message Complexity

The number of messages transmitted that communicate neuron output values is the main component of communication costs, hence complexity is measured by this factor. A fundamental unit of measurement is each original or retransmitted neuron output message. Using backpropagation with momentum, n_{hid} neurons in the hidden layer send their outputs to an average of n_{out} neurons in the output layer during MLP training. Messages must be retransmitted at every wireless hop. H_{ij} is the distance between hidden and output neurons. The total number of message transmissions (including retransmissions) for each training pattern during the forward propagation cycle is represented as c_{fp}.

$$c_{fp} = \sum_{i=1}^{n_{hid}} \sum_{j=1}^{n_{out}} h_{ij} \cdot$$

Each PT and PV training pattern in the training and validation datasets costs this. The number of iterations needed for convergence depends on the dataset size, issue features, initial weight values, and learning parameters, as well as the message complexity, MC, during the whole training episode. The equation below calculates message complexity during forward propagation:

$$MC_{FP} = n_{tier} \times (|PT| + |PV|) \times c_{fp}.$$

In backward propagation, n_{out} output-layer neurons send messages to n_{hid} hidden-layer neurons, equaling c_{fp}. If online or incremental learning is utilized, each pattern in the training set is costed throughout the training phase, which lasts several iterations until convergence. Thus, backward propagation message complexity is measured by

$$MC_{BP} = n_{tier} \times |PT| \times c_{fp}.$$

The cumulative message complexity for both training and validation is expressed as

$$MC = MC_{FP} + MC_{BP} = n_{tier} \times (2|PT| + |PV|) \times \sum_{i=1}^{n_{hid}} \sum_{j=1}^{n_{out}} h_{ij} \cdot$$

Table 2 displays the metrics for message complexity across all simulation trials. It also shows the average value of the message complexity variable for each dataset. The total of the hop lengths between every pair of nodes, the number of iterations, and the patterns used for training and testing determine the complexity of the message.

Table 2: Factors Influencing Message Complexity in WSN-MLP Design

Dataset	Iterations	Training patterns	Testing patterns	Hidden neurons	Output neurons	cfp	Message packets
Breast Cancer Wisconsin	179	100	50	4	3	22	985,671
Wine Quality	194	117	58	9	3	52	2,858,248
Digits	126	234	117	11	2	45	3,260,205
Adult	139	239	119	16	6	269	23,755,353
Pima Indians Diabetes	113	2000	666	53	10	2128	804,279,201
Heart Disease	170	5198	2599	131	26	20724	2,549,444,060

Abalone	106	4667	2333	141	2	1446	1,381,217,200

In Table 2, the total distances (hops) between any two neurons (motes) and the quantity of messages (packets) increase proportionally with the number of neurons (calculated as a function of pattern count) in the hidden and output layers. For every hundred-fold increase in neuron count, hops and messages double by 1,000. Though time complexity is essential, message complexity—which is worse than linear—is the biggest barrier on WSN-MLP scalability.

V. CONCLUSION

The inherently variable latency between nodes is captured by the custom-developed simulator, which successfully replicates the complexity of message transmission and reception in the context of wireless communication. In order to demonstrate how important network topology and communication parameters are for training neural network models efficiently and accurately, the research simulates the forward and backward propagation phases of an MLP network. The findings highlight the significance of meticulously controlling communication delays in order to maximize the efficiency of distributed neural networks in WSNs. Message complexity is the primary factor limiting network scalability, and the research sheds light on the scalability issues of large-scale WSN-MLP systems. The results provide a foundation for future studies to strengthen and optimize these systems, and they add to our knowledge of how communication flaws impact dispersed learning in wireless settings.

REFERENCES: -

[1] J. Isabona, A. Imoize, S. Ojo, O. Karunwi, Y. Kim, C.-C. Lee, and C.-T. Li, "Development of a Multilayer Perceptron Neural Network for Optimal Predictive Modeling in Urban Microcellular Radio Environments," Applied Sciences, vol. 12, no. 11, 2022.

[2] L. Weissbart, "Performance Analysis of Multilayer Perceptron in Profiling Side-Channel Analysis," unpublished, 2020.

[3] S. Kumar, R. Sharma, and E. Vans, "Localization for Wireless Sensor Networks: A Neural Network Approach," International Journal of Computer Networks & Communications, vol. 8, no. 1, pp. 61–71, 2016.

[4] M. A. Alsheikh, S. Lin, D. Niyato, and H. P. Tan, "Machine Learning in Wireless Sensor Networks: Algorithms, Strategies, and Applications," IEEE Communications Surveys & Tutorials, vol. 16, no. 4, 2014.

[5] G. Serpen, J. Li, and L. Liu, "AI-WSN: Adaptive and Intelligent Wireless Sensor Network," Procedia Computer Science, vol. 20, no. 1, pp. 406–413, 2013.

[6] J.-R. Jiang, C.-M. Lin, F.-Y. Lin, and S.-T. Huang, "ALRD: AoA Localization with RSSI Differences of Directional Antennas for Wireless Sensor Networks," International Journal of Distributed Sensor Networks, vol. 2013, p. 10, 2013.

[7] B. Cheng, R. Du, B. Yang, W. Yu, C. Chen, and X. Guan, "An Accurate GPS-Based Localization in Wireless Sensor Networks: A GM-WLS Method," in 40th Int. Conf. on Parallel Processing Workshops (ICPPW), Taipei City, 2011, pp. 33–41.

[8] S. G. Soares Alcalá, A. Rocha, M. Talles, A. De, T. Barbosa, and R. Araújo, "Embedding a Neural Network into WSN Furniture," unpublished, 2010.

[9] S. Pal, "Localization algorithms in wireless sensor networks: current approaches and future challenges," Network Protocols and Algorithms, vol. 2, pp. 45–74, 2010.
[10] E. Guerrero, H. G. Xiong, and Q. Gao, "A distributed range-free localization algorithm for wireless sensor networks based on a mobile robot," in Advanced Technologies for Communications (ATC '09), pp. 93–98, 2009.
[11] G. Stefano and A. Petricola, "A distributed AOA based localization algorithm for wireless sensor networks," Journal of Computers, vol. 3, 2008.
[12] A. Shareef, Y. Zhu, and M. Musavi, "Localization using neural networks in wireless sensor networks," in 1st Int. Conf. on MOBILe Wireless MiddleWARE, Operating Systems, and Applications (MOBILWARE '08), 2008.
[13] G. Mao, B. Fidan, and B. D. O. Anderson, "Wireless sensor network localization techniques," Computer Networks, vol. 51, pp. 2529–2553, 2007.
[14] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," Computer Networks, vol. 38, pp. 393–422, 2002.