future internet

MDPI

Article

Machine Learning Algorithm for Delay Prediction in IoT and Tactile Internet

Ali R. Abdellah [1,2,*], Omar Abdulkareem Mahmood [3], Ruslan Kirichek [2], Alexander Paramonov [2] and Andrey Koucheryavy [2]

[1] Department of Electrical Engineering, Faculty of Engineering, Al-Azhar University, Qena 83513, Egypt
[2] Department of Communication Networks and Data Transmission, The Bonch-Bruevich Saint-Petersburg State University of Telecommunications, 193232 St. Petersburg, Russia; kirichek.sut@mail.ru (R.K.); alex-in-spb@yandex.ru (A.P.); akouch@mail.ru (A.K.)
[3] Department of Communications Engineering, College of Engineering, University of Diyala, Baquba 32001, Iraq; mahmood_omar@list.ru
* Correspondence: alirefaee@azhar.edu.eg

Abstract: The next-generation cellular systems, including fifth-generation cellular systems (5G), are empowered with the recent advances in artificial intelligence (AI) and other recent paradigms. The internet of things (IoT) and the tactile internet are paradigms that can be empowered with AI solutions and integrated with 5G systems to deliver novel services that impact the future. Machine learning technologies (ML) can understand examples of nonlinearity from the environment and are suitable for network traffic prediction. Network traffic prediction is one of the most active research areas that integrates AI with information networks. Traffic prediction is an integral approach to ensure security, reliability, and quality of service (QoS) requirements. Nowadays, it can be used in various applications, such as network monitoring, resource management, congestion control, network bandwidth allocation, network intrusion detection, etc. This paper performs time series prediction for IoT and tactile internet delays, using the *k*-step-ahead prediction approach with nonlinear autoregressive with external input (NARX)-enabled recurrent neural network (RNN). The ML was trained with four different training functions: Bayesian regularization backpropagation (Trainbr), Levenberg–Marquardt backpropagation (Trainlm), conjugate gradient backpropagation with Fletcher–Reeves updates (Traincgf), and the resilient backpropagation algorithm (Trainrp). The accuracy of the predicted delay was measured using three functions based on ML: mean square error (MSE), root mean square error (RMSE), and mean absolute percentage error (MAPE).

Keywords: 5G; IoT; tactile internet; machine learning; AI; traffic prediction

Citation: Abdellah, A.R.; Mahmood, O.A.; Kirichek, R.; Paramonov, A.; Koucheryavy, A. Machine Learning Algorithm for Delay Prediction in IoT and Tactile Internet. *Future Internet* **2021**, *13*, 304. https://doi.org/10.3390/fi13120304

Academic Editor: Luis Javier Garcia Villalba

Received: 18 October 2021
Accepted: 24 November 2021
Published: 26 November 2021

1. Introduction

Artificial intelligence (AI) is emerging as a critical ingredient needed to understand multiple datasets collected and make them commercially valuable. AI can support data analysis from the internet of things (IoT), where the system can perform tasks or improve intelligent information. Moreover, AI in IoT devices can identify data, adopt decisions, and operate on that information without user intervention [1].

Machine learning (ML) is the method that deals with developing algorithms that can learn from information and make predictions. Moreover, modern central processing unit (CPU) technology enables an effective implementation of AI algorithms. However, the traffic volume is increasing, and the heterogeneity of traffic is increasing. IoT and ultra-reliable, low-latency communications bring a variety of demands that require greater efficiency in quality of service (QoS)-related decisions, as current QoS technologies cannot achieve the desired level. Most provisions are necessary to predict load and delay for specific facilities, considering geographic location and dynamics, such as subscriber movement and incredible speeds. Operators also need an overall system that allows for predictions

of infrastructure evaluation, considering the adoption rate of new technologies that will enable new online services and the associated expected changes in people's lifestyles [2].

AI has a crucial role in fifth-generation (5G) networks, such as the IoT and the tactile internet, because it can quickly extract insights from data. ML can automatically discover models and anomalies in information produced from sensors and smart devices. AI technologies extend ML strategies applied to smart IoT devices to make complex decisions based on recognition patterns, self-learning, self-healing, context awareness, and autonomous decision-making. These include and influence future applications of dual digital models and continuous learning that play a role in autonomous vehicle applications, IoT, and predictive maintenance [3,4].

AI is widely applied in wireless network use cases, such as handling and recognizing information and data streams [5,6]. It can also predict and process historical data series and geographical and positional information in real time, essential in almost all technology areas. It affects the applications, evaluation tools, computational level, data processing methods, and interface control capabilities in virtually all areas of information technology (IT). 5G will increase the speed and combination of other technologies, while AI will enable machines and systems to work intelligently like humans. In summary, 5G will accelerate services in the cloud while rapidly analyzing and learning from the same data.

Wireless communication is a driving force in promoting economic, technical, social, security, and study conditions. It also supports the growth factor to meet the demands of the new generation. Researchers are constantly striving to develop new wireless communication technologies and mechanisms to facilitate human life, such as the advancement of IoT, 5G, transport networks, tactile internet, etc. Thanks to the advances in computing capabilities and ML techniques such as deep learning (DL), AI performs well in various applications, such as security, networking data processing, etc. Undoubtedly, AI reduces user interference and provides reliable results irrespective of the application domain. Therefore, the advancement of AI can also offer innovation and novelty to 5G networks to manage the network efficiently. With additional AI and automation layers that provide zero-touch and end-to-end, 5G will provide excellent revenue growth opportunities. Nowadays, many researchers are interested in integrating AI and wireless communication, which is reflected in the standardization schemes [7,8].

The growth of IoT requires the installation of various applications with complex operational requirements. Delays are one of the most critical measures for IoT, especially in network traffic prediction and healthcare monitoring, where urgent situations need to be managed. In IoT applications, intelligent processing and big data analytics are the main drivers for development. Data science with the IoT is mainly used in various fields where volume, speed, and pattern recognition are concerned. Due to predictive ML analysis, the software can predict both desired and undesired incoming events. Therefore, the ML system detects abnormal behavior and helps to understand and establish long-term trends, which requires continuous correction and monitoring to achieve effectiveness and efficiency in data analysis [9,10].

There is an apparent convergence between IoT and AI. IoT can connect objects and devices to use the data generated by these devices, while AI can model the intelligent behavior of all types of devices. Since IoT systems generate substantial data sets, AI helps process these datasets and understand the information. However, traditional methods for structured data, analytics, and definition processes cannot effectively manage the massive data flowing in real time from IoT devices. There is a high degree of convergence between IoT and AI. IoT can connect the devices, the data generated by those devices, and AI to model the intelligent behavior of all types of devices. Since IoT devices will produce an enormous amount of information, AI will process and understand the information. However, with traditional approaches to structuring information, analytics, and specific processes, you will not efficiently manage all the information about the entities that make up the IoT in real time. AI-based analysis and response can be used to extract the optimal value of the information, if any. The proliferation of ML and IoT algorithms can significantly

improve the application infrastructure. Using ML can improve the management of the network to avoid congestion, optimize the allocation of resources, and examine the essential information to adopt decisions or offloading. ML methods, including artificial neural network (ANN)-based approaches, can effectively store and process large amounts of data [1].

Network information prediction is among the best active domains that combine AI studies for information networks. Nowadays, it can be used in various applications and is attracting more attention from many researchers. Information prediction is an effective way to ensure the security, reliability, and connectivity of selected networks. Many network traffic prediction methods have been proposed and tested, including internet-based data extraction methods. Many interesting network prediction strategies have been developed to achieve powerful results [7–21]. Figure 1 illustrates an example of IoT traffic data prediction using ML techniques.

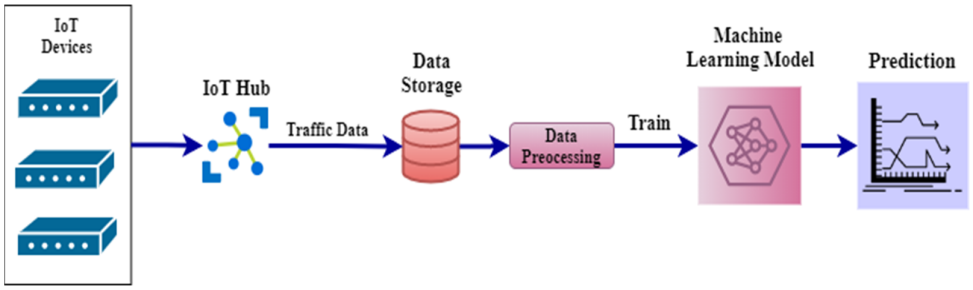

Figure 1. Machine learning for IoT traffic prediction.

Predicting historical information refers to predicting the following values of the system from previous and current information. Typically, predefined computational modeling (or hybrid modeling) is used to achieve the expected data. A recurrent neural network (RNN) essentially has a memory that can predict time-dependent targets. RNNs can store the previously sensed state of the inputs to determine the future time step. Recently, many variations were introduced to adapt recurrent networks to a variety of domains. Nonlinear autoregressive with external input (NARX) is a powerful RNN used to solve the problem with nonlinear historical information. The widely used NARX model provides promising results for time series problems based on lagged input and output variables and prediction errors, as discussed in these studies. Unlike the conventional RNN, the NARX network provides optimal prediction performance for almost any nonlinear function with little or no computational losses. The NARX model [8–10,20,22,23] has been used for various applications in previous studies.

There are several uses for the NARX network, which can be used for predicting future information. It can be used for nonlinear cleanup operations, where the desired result is noise-free input information. The use of the NARX network brings several essential features, namely, the representation of dynamic systems. ANN contains a feedback and delay mechanism that allows information to flow between the neurons of the different layers. Feedback is a way of storing a temporary memory that allows the network to retrieve old data. While delay provides direct sets of past data for the current period, response methods perform processing (filtering) of past data [8–10]. Accurate network traffic prediction enables efficient resource management that ensures appropriate QoS measures. Traffic prediction can be used for various applications, such as network monitoring, resource management, congestion control, network bandwidth allocation, network intrusion detection (e.g., anomaly detection), etc. With the evaluation of 5G networks, the traffic volume and network complexity have increased. Many network traffic prediction algorithms have been proposed, such as autoregressive integrated moving average (ARIMA), *k*-nearest neighbors (KNN), random forest, support vector machine (SVM), linear regression, etc. However, when we have a large amount of contaminated data, these methods do not seem

to work [24,25]. Therefore, we use a k-step prediction approach with NARX-RNN for accurate network traffic prediction in this work.

The motivations behind this study include the following:

1. Optimize the QoS requirements and network monitoring to manage resources and ensure security.
2. The NARX-RNN technique predicts time series data, remembers the historical data, and accurately estimates future time series data. It also has the advantage over other time series prediction approaches in that it serves to maximize the accuracy of the learning method over the training iterations. As more data are added to the model, the model becomes smarter and can better predict network traffic, which is significant for actual traffic information forecasting.
3. There is insufficient knowledge about the delays in IoT communication parameters.
4. There is a lack of accurate ML analysis to achieve reasonable prediction accuracy.
5. The challenging problems in optimizing QoS measures are computationally complex.
6. Monitor network availability and activity to detect security and operational issues.

Due to the limitations of the available solutions, this work focuses on forecasting delays in IoT and tactile internet networks. Several application domains are delay aware and necessary (e.g., healthcare, security, and medical emergency). In these applications, e.g., intensive remote patient monitoring, a momentous case must be revealed to a healthcare agency within a specific period to determine appropriate steps. Moreover, this might lead to different amounts of information relying on the number and type of observations performed. To perform predictive modeling of delay, we use an ML-based technique, such as the NARX-RNN model with a *k*-step prediction approach.

The key contributions of the proposed work are summarized as follows:

- We proposed a *k*-step-ahead time series prediction approach with a NARX-enabled RNN for delay prediction in IoT and tactile internet.
- The ML model was trained using four ML algorithms: Bayesian regularization backpropagation (Trainbr), Levenberg–Marquardt backpropagation (Trainlm), conjugate gradient backpropagation with Fletcher–Reeves updates (Traincgf), and resilient backpropagation algorithm (Trainrp).
- The prediction accuracy was measured using three ML-based functions: mean square error (MSE) loss function, root mean square error (RMSE), and mean absolute percentage error (MAPE).
- The above training algorithms were compared depending on the proposed time series prediction approach using RMSE and MAPE.
- Finally, the results of the simulation-based tests demonstrate that:
    - The model trained with the Trainbr training algorithm outperforms its competitors and is the best predictive model in both 1-step prediction and 15-step prediction.
    - Moreover, the model trained with the algorithm Traingf outperforms its competitors for the 10-step prediction case.
    - On the other hand, the model trained with the algorithm Trainrp has poor prediction accuracy, compared to its competitors.

The outline of the article is structured accordingly: the related literature review is presented in Section 2; machine learning for time series prediction is presented in Section 3; the problem formulation and system model are presented in Section 4; the performance evaluation is presented in Section 5; the simulation results are shown in Section 6; and the conclusions and directions for future work are presented in Section 7.

2. Literature Review

Recently, several researchers have focused on the time series prediction of wireless network traffic using ML methods and in 5G technology. This paper aims to predict a delay in IoT and tactile internet traffic using the ML approach, especially the k-step ahead

prediction approach using NARX-enabled RNN. Therefore, in this section, we introduce the previous works that relate to our field of study.

The author of [9] presented a method for time series prediction for IoT data streams with multi-step prediction (MSP), using NARX-RNN. He calculated the prediction performance using different loss functions, such as MSE, sum squared error (SSE), Mean absolute error (MAE), and MAPE. IoT delay prediction was also performed using a deep neural network (DNN): a multiparameter method in [13]. In [8], traffic prediction was performed using a NARX-enabled RNN based on a software-defined networking (SDN) infrastructure. The ANN training algorithms were trained using three ANN training algorithms: Trainlm, Traincgf, and Traincgp. The prediction accuracy, the MAPE, was measured. ML-based low latency bandwidth prediction was performed for H2M networks [14].

Ali R. Abdellah et al. [10] performed IoT delay prediction using the ML method based on the NARX recurrent neural network. The author proposed two methods for time series prediction: MSP and single-step prediction (SSP). The model ANN was trained with three ANN training algorithms: Traincgf, Trainlm, and Trainrp. The prediction performance was measured using RMSE and MAPE. The packet transmission delay prediction in an ad hoc network was analyzed using ANN [15]. White et al. [16] studied QoS prediction in IoT systems regarding transmission time and traffic capacity. In [17], the short-term prediction was discussed using different ANN methods. Research directions for further applications of ANN-based techniques were also proposed. The evaluation of a graph structure DL was studied for network traffic flow prediction with superior performance [18]. In research [19], the multilayer neural network (MLNN) was proposed for speed prediction by integrating the convolutional neural network (CNN) and gated recurrent units (GRU) to estimate the predicted speed performance with the network performance.

Haviluddin et al. [22] studied the performance of time series modeling with the NARX network for network traffic prediction. Khedkar et al. [26] studied the problem of predicting IoT traffic using ML, DL, and statistical time series-based prediction technologies, including long short-term memory (LSTM), ARIMA, Vector autoregressive moving-average (VARMA), and feedforward neural networks (FFN). Li Run et al. [27] predicted subway traffic based on ARIMA and the gray prediction model, but this model has the drawback of being unable to capture rapid changes in traffic data. Alsolami et al. [28] gave an overview of techniques for traffic forecasting and examined the limitations of each technique. Additionally, they provided an overview of the different types of raw traffic data. Finally, they described a hybrid approach for traffic forecasting. The hybrid system is based on ARIMA and SVM techniques.

3. Machine Learning for Time Series Prediction

Predicting historical time sequences is an essential aspect of ML; it belongs to supervised learning approaches and is widely used in data science, applied in various domains. Several ML techniques, including regression, ANN, KNN, SVM, random forest, and XGBoost, can be used to predict time series. ML-based forecasting models have found wide application in time series projects required by various organizations to facilitate predictive allocation of time and resources.

ANN can help historical series prediction by eliminating the instantaneous need for extensive feature technology processes, data scaling procedures, and the need for stationarity and differentiation of historical series data. RNN is suitable for supervised learning tasks when data are available in a temporal sequence. It can remember the historical information to estimate future time-series data. The RNN algorithm is trained based on the previous data of the historical series into the input level. The network's connectivity is adapted depending on the difference between the actual and expected outputs over the network. Before configuring the network, the operator must determine the network hidden layers to size and the training termination process.

In prediction, the past information is used to predict what will follow, and the following information is predicted, relying on what happened. The temporal sequence adds

a temporal dependency between historical information. This dependency is under limitations and is structured to provide an additional source of information. Historical time sequence prediction is a method of predicting information about a historical series. It expects the following information by analyzing the past information if the future information is similar to the historical information. It can be applied in several use cases, such as resource allocation, network traffic, weather forecasting, control engineering, statistics, signal processing, and business planning. These are just a few of the many possible applications for time series forecasting.

In real-world time series—i.e., forecasting weather, air quality, and network traffic flow are scenarios based on IoT devices, such as detectors—abnormal time form, missing data, high noise, and complicated correlations are multivariate and current limitations of classical prediction techniques. Such methods usually depend on noise-free and perfect information for good performance: missing data, outliers, and other erroneous features are generally not supported. Time series prediction starts with a historical time sequence, and experts investigate the historical information and temporal decomposition models, such as tendencies, seasonal models, periodic models, and symmetry. Various sectors, such as commerce, utilize historical time sequences prediction to assess potential technological complexity and customer needs. Temporal series data models can have many variations and perform various random processes [7–21].

3.1. NARX Neural Network

NARX represents a nonlinear autoregressive network with external inputs. NARX are dynamical RNN, have return paths surrounding various network connections. NARX networks are based on the auto-regressive with exogenous input (ARX) time series models, commonly used for time series operations, and are considered a nonlinear form of the ARX model. NARX models can simulate various nonlinear dynamic methods; they have been used for multiple problems, including time series simulation. The NARX network uses prior measures of the existing historical time series to make predictions and the previous values of other inputs to make predictions for the target series. NARX is a robust tool suitable for nonlinear modeling systems. Moreover, NARX learns more efficiently than other neural network time series, using a gradient descent learning algorithm. NARX networks have been successfully used in many applications to predict future values of the input signal.

NARX networks perform better on predictions, where the desired output depends on inputs that exist at absolutely past the time points. NARX is also used as a nonlinear filter whose training data are trained with a noiseless form of input information. When applied to the historical time sequences [20], potential values for the time sequence $y(n+1)$ are predicted from past values for that historical time sequence and the final measure for the historical time sequence $u(n)$. The model is mathematically represented as follows:

$$y(n+1) = f\left(y(n), y(n-1), \ldots . y\left(n-d_y\right), u(n), u(n-1), \ldots . u\left(n-d_u\right)\right) + e(n) \quad (1)$$

In the compact form, we can rewrite Equation (1) as follows.

$$\mathrm{y}(n+1) = \mathrm{f}[\mathrm{y}(n); \mathrm{u}(n)] \quad (2)$$

Here, $y(n) \in R$ and $u(n) \in R$ define the input and output parameters of the network at the time (n), respectively. Moreover, $y(n)$ and $y(n+1)$ are the desired and predicted output elements, respectively. Accordingly, d_y and d_u are the time delays of the output and input variables, respectively (z^{-1} = time delay unit), and $e(n)$ is the model error between the desired and predicted output [22,23,29,30].

Figure 2 illustrates NARX neural network architecture [22,23,30].

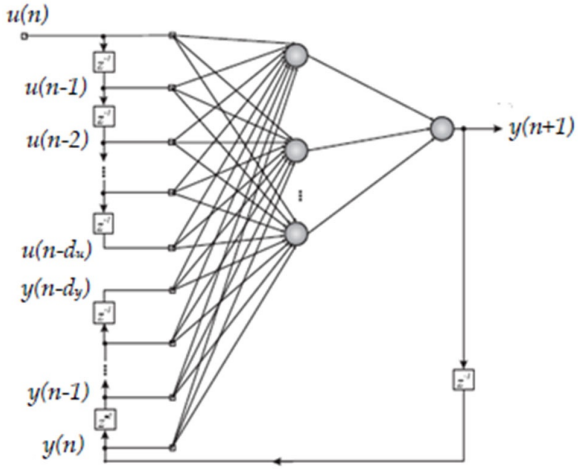

Figure 2. NARX neural network architecture [22,23,30].

3.2. K-Step-Ahead Prediction

The primary purpose of modeling is to predict the subsequent output values using the previous and current values of the estimation unit. This unit was first determined using test information, matching several models to the dataset based on historical values, from which the optimal one was selected. Then, a new dataset was used in the unit, and the forecasting performances were compared. It was analyzed how the model can adapt to the output pattern. In this study, many prediction models were compared, k-step-ahead prediction (one-step, . . . , k-steps), using four different training algorithms—Trainlm, Traincgf, Trainbr, and Trainrp—to investigate which is the optimal performance generated based on these algorithms.

In general, time series forecasting determines the prediction of the observed data for the following time step. This is known as one-step prediction, open-loop, or SSP, as only a single time step can be predicted. One-step $(n + 1)$ prediction is performed by passing the present and previous data points $(n, n - 1, \ldots, n - k)$. In this method, the current model is always replaced by updated input values. That is, the model is permanently changed depending on the adjusted parameter values. For multi-step predictions, the same model is used repeatedly to predict all output values.

In the simplest case, one-step prediction models have the following form:

$$y(n + 1) = f(y(n),y(n-1),u(n),u(n - 1) \ldots .(n - k)) \quad (3)$$

where $y(n + 1)$ is the predicted output; $y(n)$ *and* $u(n)$ are the observation of the target and exogenous inputs, respectively; n is the time step; k is the number of inputs (k-step prediction); and $f(.)$ is the prediction function learned by the model.

In multi-step or k-step prediction, the same model is used repeatedly to predict all output values. The following equation can describe the prediction model in this method:

$$y(n + k) = f(y(n), y(n - 1) \ldots ,y(n - k_y)\ u(n),x(n - 1) \ldots , u(n - k_x)) \quad (4)$$

where $y(n + k)$ is the predicted output, k_x is the number of input delays (unit delay), and k_y is the number of output delays.

The multi-step-ahead method predicts the following values of a historical time sequence step by step. First, $u(n + 1)$ is expected based on the previous x-values, $u(n + 1 - x)$ . . . , $u(n - 1)$, then $u(n + 2)$ is predicted depending on the past x-values that contain the expected value for $u(n + 1)$. The process repeats until the final value, $u(n + k)$, is measured.

There are some tasks involving time series where multiple time steps must also be predicted. In contrast to SSP, these are called MSP or multi-step time series prediction problems. MSP is an exciting forecasting method that uses an autoregressive model to form

a closed loop in the forecast. It is a forecasting method that predicts potential measures in the historical time series based on past values. The multi-step forecasting technique applies a stepwise forecasting model that uses the expected measures in the actual period to determine their values in the following step. Many temporal series problems involve predicting successive potential values, using only the previous values, such as multi-step time sequence prediction, which predicts the historical time sequence for many problems, such as wireless network traffic, energy consumed, etc. Aware of the series of subsequent values, we can extract fascinating features of the historical time sequence, such as expected bandwidth, delay, throughput, energy consumption, uncertainties, attack time, and higher or lower measures with unusual frequency. In particular, predicting the historical time sequence in multiple steps allows us to predict the cropping season for next year, the delay in a wireless network for the next hour, low and high weather temperatures for the coming months, etc. A standard method to solve these problems is to build a specific model from the past values of the time sequence and then apply it gradually to predict their following values. This method is called multi-step prediction. Since it uses the expected information from the previous one, it can be experimentally demonstrated that multi-step prediction is prone to rounding errors, i.e., errors made in the previous one are carried over to the following prediction [8–10,21,31].

4. Problem Formulation and System Model

Recently, various ML algorithms were used for network traffic prediction, such as random forest [32], ARIMA [33], LSTM [7,11,12], etc. In this work, the NARX-RNN was proposed for IoT delay prediction as proven in many applications. Moreover, we used the *k*-step ahead prediction approach with NARX neural network for IoT traffic prediction.

First, we built the IoT system to generate the dataset; we also modeled the IoT system using the AnyLogic simulator. Second, after collecting, analyzing, and processing the dataset, we used it as input to ANN for the prediction process. After loading the dataset as input for the network, the dataset was divided into two subsets in the columns Input (I) and Output (O) and then split into training, testing, and validation subsets accordingly. The normalization of the input data must be in the interval $[-1, 1]$ compatible with the actual maximum or minimum values.

The model ML was trained with four different training algorithms: Trainbr, Trainlm, Traincgf, and Trainrp. In addition, the prediction accuracy was measured using ML-based functions: MSE, RMSE, and MAPE.

4.1. IoT System Modeling

This section introduces the IoT model used to produce the dataset for the ML training. We built the IoT system, using the AnyLogic simulator [9,10].

Figure 3 depicts the structure of an IoT system for creating traffic data and modeling a set of IoT tools. The model contains an IoT traffic system that models the process of an IoT device or group of IoT devices, a traffic source for conventional communication facilities, and TI traffic referred to as H2H + TI (H2H—human to human, TI—tactile internet). The accomplished entry traffic moves to the connection node, and the architecture is introduced as a queuing system *G/G/1/k* with the service disciplines (with delay and failure basis system). The arrival time is denoted by $\bar{t}$.

The number of arrivals per time of IoT traffic is defined by λ_{IoT}, H2H traffic λ_{H2H}, total flow $\lambda = \lambda_{H2H} + \lambda_{IoT}$. Assume that with probability p, a message reaches the model entry in which all locations in the queue are busy, and an attack (such as DoS) occurs. The entire traffic flow at the outcome of the system is defined by λ. The characteristics of the two flows determine the whole traffic flow at the system's entry; therefore, it is generally different from the characteristics of normal traffic and IoT traffic.

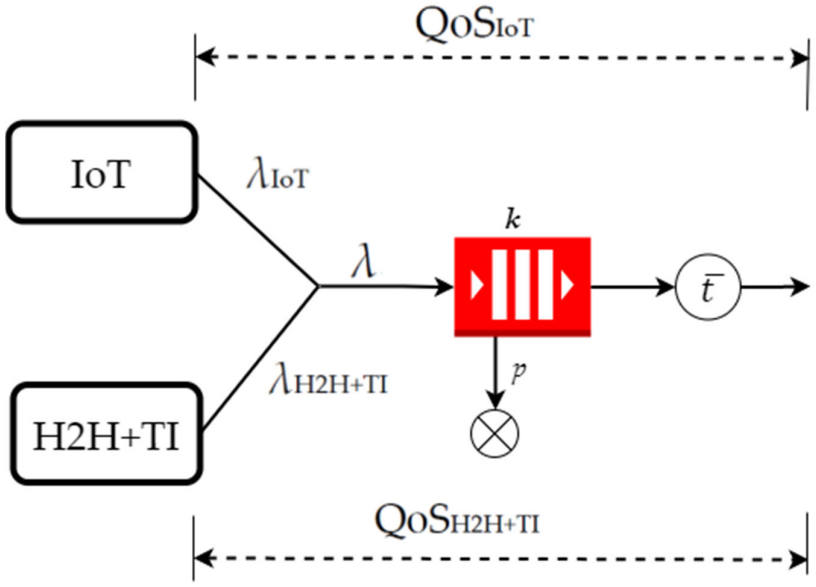

Figure 3. Service model for the collected traffic.

The above queuing system (QS) can be described as a *G/G/1/k* system. However, this system does not have accurate computational modeling to evaluate the probability of the packet loss and delayed arrival. Therefore, in [34,35], the diffusion approximation is used to assess the probability of packet loss, given the aware allocation criterion that describes the incoming and packet processing.

4.2. ANN Training

We used NARX-RNN to predict the IoT delay. For training NARX-RNN, we loaded the training data and used a tapped delay line (TDL) with two input and output delays. This work assumed that the output and input delays are the same, where $d_y = d_u = d$; also, there are two inputs to the NARX network, $u(n)$ and $y(n)$. We created a NARX network using the narxnet functionality. The network consists of three layers, with a hidden layer containing 20 hidden neurons. For training the network, we proposed four different training algorithms, Trainlm, Traincgf, Trainbr, and Trainrp, for the training tasks and then the preparets function for preparing the data. Finally, we performed the IoT delay prediction using the k-step-ahead prediction approach.

The NARX network is trained in two methods: using an open-loop training architecture and a closed-loop architecture. Closed-loop networks produce a multi-step prediction. In other words, they continue to predict when inner returns lose outer returns (responses). The multi-step-ahead forecasting method is often helpful for simulating a network in an open-loop form where the output data are known and then transitioning to a closed-loop form, where the output is returned to the network input via the NARX network to implement multistage prediction, even though the feedback was just provided. First, all stages, except the k-time stages of the input sequence and the desired output sequence, are used to model the network in the open-loop architecture, as shown in Figure 4, to take advantage of the high precision provided by introducing the desired output sequence. Then, the network and its final stage are transferred to the closed-loop architecture, as shown in Figure 5, to make k-step predictions with only the k inputs.

The output is also an input returned with a unit delay to the network's information within the usual NARX structure.

Here, the network is simulated as a closed loop only. The ANN performs multiple predictions for the outer input sequence and the initial terms, as the input sequences have periods.

Note that the "y" sequence is a response signal that is also output (desired output). After closing the response path, the corresponding output is delivered to the corresponding input.

The one-step prediction for multiple instances helps to obtain the fast time-step prediction. The prediction of the observed data is in the next time step since only a one-time step can be predicted. The one-step $(n + 1)$ prediction is satisfied by crossing the actual and preceding information $(n, n - 1 \ldots, n - k)$ and obtaining the expected output $y(n + 1)$ (Algoritm 1).

Algorithm 1: NARX Neural Network Training

Load the IoT data sets.
InputSeries: Time series training data $u = \{u_1, u_2 \ldots, u_n\}$
TargetSeries: Time series training data y = $\{y_1, y_2 \ldots, y_n\}$
Outputs: the predicted output $\hat{y} = \{\hat{y}_1, \hat{y}_2, \ldots \hat{y}_n\}$
Error (e) $\leftarrow (y - \hat{y})$
Initialize the weights and biases randomly and threshold
Divide the dataset into 70% training, 15% validation, and 15% test sets
Normalize the dataset into values between [−1,1]
Specify the predictors (u) and target (y)
1000$\leftarrow$ number of epochs
32 $\leftarrow$ batch size
Define NARX neural network model architecture
Choose the loss function () $\rightarrow$ MSE
Choose trainingfcn () $\rightarrow$ Trainbr, Traincgf, Trainlm, Trainrp.
Define input Delays, feedback Delays
Define HiddenLayerSize
for n epochs and ith, iterations do
    for k-step Prediction
        for i = 1: N (N = numTimesteps)
            Train NARX network ()
            Test network()
            Calculate the output error ()$\rightarrow$e
            if e < threshold
                Calculate the predicted output()
                Save the predicted values()
                Calculate accuracy loss()
            else
                Update the network weights (repeat the operation)
        End for
    End for
End for

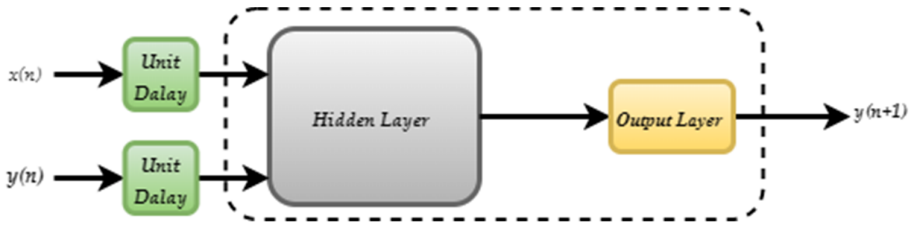

Figure 4. NARX neural network—open-loop form.

In many cases, such as decision making, it would be helpful to have a prediction $y(n + 1)$ when $y(n)$ exists until the actual $y(n + 1)$ appears. The network can return its output for the early time step by eliminating a delay so that its minimum delay unit is now 0 rather than 1. The current network produces a similar output to the primary network, but the output is shifted one step to the left, as shown in Figure 4.

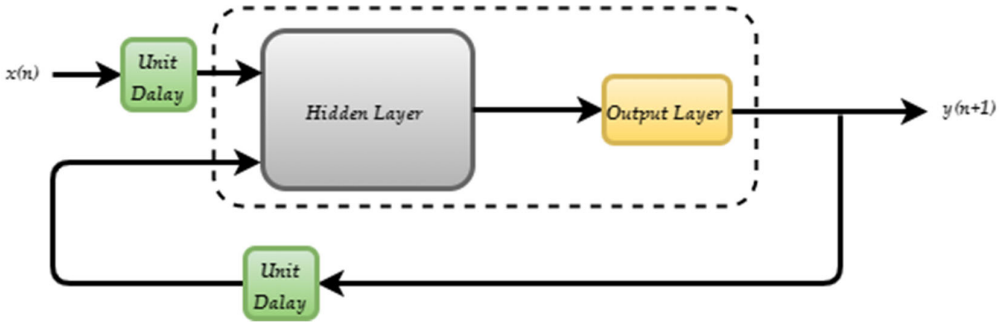

Figure 5. NARX neural network—closed-loop form.

5. Performance Evaluation

The algorithms of ML have proven successful in network traffic prediction. The prediction error is a kind of bad prediction, and more precisely, if there are no error measurements, the prediction method is optimal. Therefore, our goal of reducing the errors after adjusting the connection weights can be evaluated in minimizing the errors. In this work, we used three different ML-based functions to measure the prediction accuracy, including MSE, RMSE, and MAPE, to estimate the prediction accuracy. MSE (Equation (5)) is a loss function that measures the mean squared error, where the error is the difference between expected and observed values. RMSE is equal to the root of MSE, as shown in Equation (6).

Moreover, MAPE measures the average absolute percentage of error using absolute values (Equation (7)). MAPE has two benefits: first, the fundamental measures prevent the positive and negative errors from eliminating each other. Second, because the percent error is independent of the measurement of the reliable variables, you can use this scale to compare predictive accuracy among time sequence data of different magnitudes.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{6}$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{7}$$

where n is the number of data points, y_i is the observed value, and $\hat{y}_i$ is the predicted value.

6. Simulation Results

In this paper, we used the MATLAB environment for simulating IoT delay prediction using *k*-step-ahead prediction with time series NARX-based RNN. The training data were produced from the IoT network; we modeled the IoT system using the AnyLogic simulator. Before the training phase, the collected data were analyzed and processed. After loading the dataset as input to the network, the dataset was divided into input (I) and output (O) columns, split into training, testing, and validation subsets accordingly. The normalization of the input data must be in the interval $[-1, 1]$ compatible with the actual maximum or minimum values. The prediction accuracy was measured using ML-based functions: MSE, RMSE, and MAPE. The model ML was trained with four different training algorithms: Trainbr, Trainlm, Traincgf, and Trainrp.

1. **Traincgf** is often significantly faster than Traingda and Traingdx and sometimes more rapid than Trainrp, although results vary from problem to problem. Traincgf requires only slightly more memory than the simpler ones, so it is usually better suited for networks with a large number of weights. When using Traincgf, the loss function

decreases fastest along the negative gradient, but this does not necessarily result in the fastest convergence.

2. **Trainrp** is usually faster than the standard steepest descent algorithm. It also has the special property that it requires only a small increase in memory. We only need to adjust the weight and bias, which is equivalent to storing the gradient. The purpose of the Trainrp training algorithm is to eliminate these harmful effects of the magnitudes of the partial derivatives.
3. **Trainlm** is a function for training that adjusts the weight and bias according to Levenberg–Marquardt optimization. It emerges to be the fastest method for training medium-sized networks, but it tends to be less efficient for training large networks. It has better training performance than other algorithms in some applications. It is also very efficient, and its properties come out even better in a MATLAB environment. The main drawback of Trainlm is that it requires the keeping of some matrices, which can be very important in certain problems.
4. **Trainbr** is a function for training that adjusts the weight and bias according to the Levenberg–Marquardt optimization. It reduces the combination of squared errors and weights, and then adjusts the valid combination to achieve better generalization capability than by stopping early; it has the best accuracy in training.

Table 1 demonstrates IoT traffic prediction accuracy in four cases corresponding to the Trainbr, Trainlm, Traincgf, and Trainrp training algorithms using RMSE and MAPE.

Table 1. The prediction accuracy for delay prediction in IoT system.

Training Algorithm	Performance Function	1-Step Prediction		10-Step Prediction		15-Step Prediction	
		RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
Traincgf	MSE	0.1253	0.1149	0.1796	0.1520	0.1970	0.1580
Trainlm		0.0672	0.0851	0.2092	0.2935	2.1092	3.3511
Trainbr		0.0650	0.0549	0.2642	0.3317	0.1412	0.1033
Trainrp		0.6810	0.7874	1.2041	6.7734	2.510	6.6828

Table 1 shows the prediction accuracy for delays in IoT traffic using four different training functions concerning *k*-step prediction models, considering MSE loss function as a performance measure. The prediction accuracy was measured in RMSE and MAPE to investigate which prediction model provides optimal accuracy and maximum average improvement.

From the tabulated results, the Trainbr algorithm outperforms its competitors and has the best performance in both the 1-step and 15-step predictions. The maximum average improvement is 0.7325% and 6.6% in both cases, respectively. However, the algorithm Trainrp has the lowest performance in both cases compared to the others.

The performance with algorithm Traincgf is almost equivalent to that of algorithm Trainbr for the 15-step prediction, with an RMSE of 1.970 and a MAPE value of 0.1580%. Thus, the maximum average improvement, in this case, is 6.5%. Moreover, the Trainlm algorithm has reasonably equivalent accuracy to the Trainbr algorithm for the one-step prediction case with an RMSE of 0.0672 and a MAPE of 0.0851%; the maximum average improvement, in this case, is 0.7%.

Moreover, algorithm Traincgf outperforms its competitors in the 10-step prediction, and the maximum improvement is 6.6%. Furthermore, algorithms Trainlm and Trainbr have approximately the same performance as algorithm Traincgf; the maximum improvement, in this case, is 6.5% and 6.4%, respectively. On the other hand, algorithm Trainrp also performs poorly in this case and has the lowest prediction accuracy compared to its competitors.

Figures 6–9 show the prediction models with the above training algorithms regarding the k-step prediction model used. As can be seen in Figure 6, for the k-step prediction models in the case of using Trainbr, as shown in the figure, the prediction models in the

case of 1-step and 15-step prediction are identical to the observed model. However, the prediction model in the case of using 10-step prediction deviates slightly from the observed model. Moreover, in all cases, we found that the resulting model increases gradually with time.

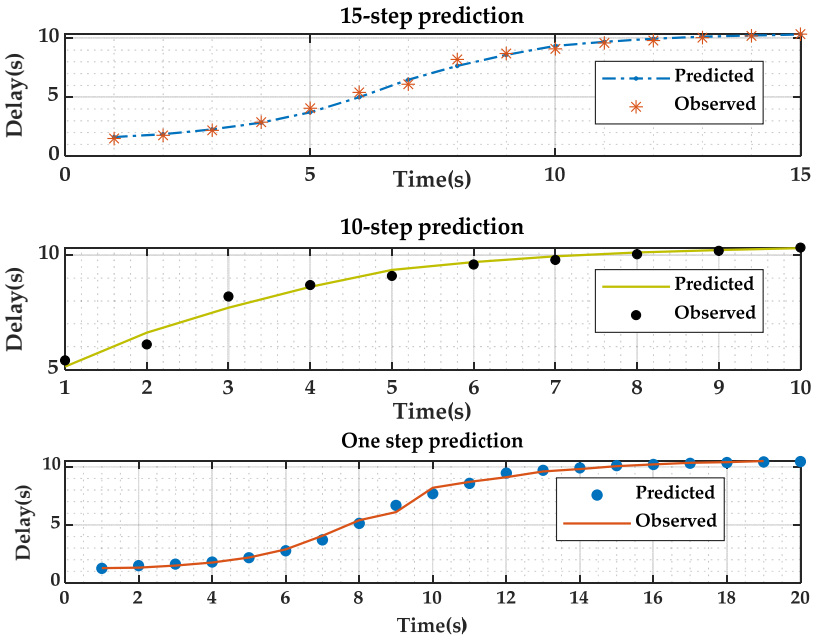

Figure 6. Predicted output models based on the k-step prediction approach with respect to the Trainbr training algorithm.

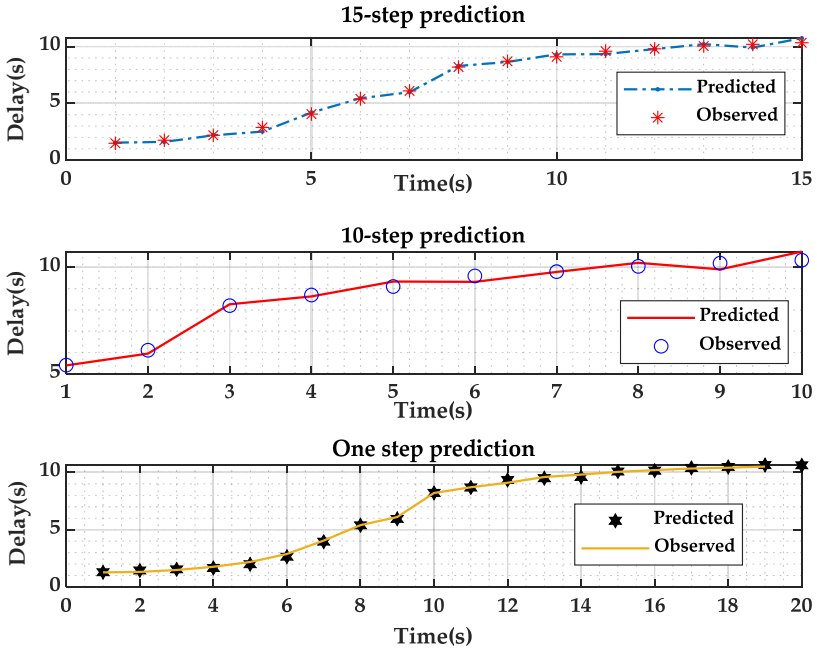

Figure 7. Predicted output models based on the k-step prediction approach with respect to the Traincgf training algorithm.

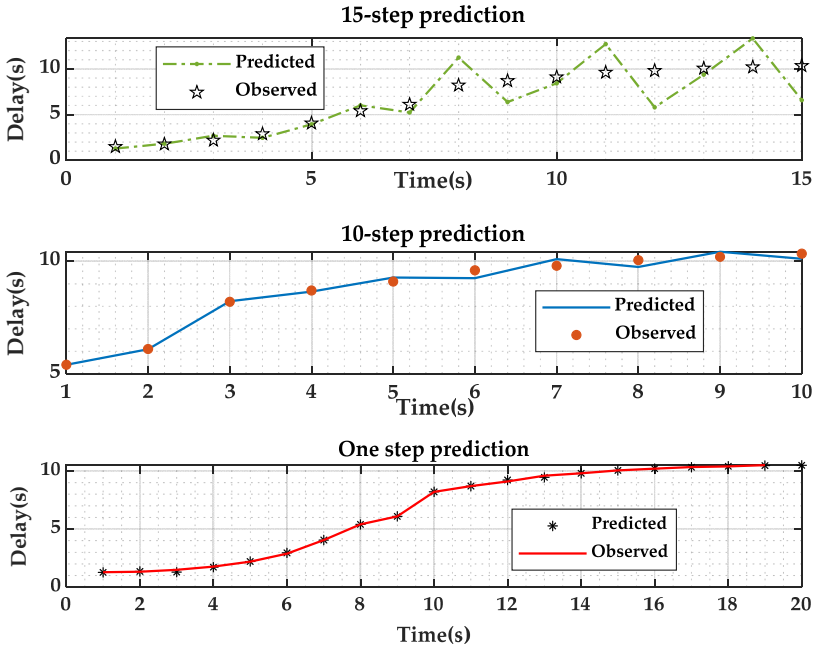

Figure 8. Predicted output models based on the k-step prediction approach with respect to the Trainlm training algorithm.

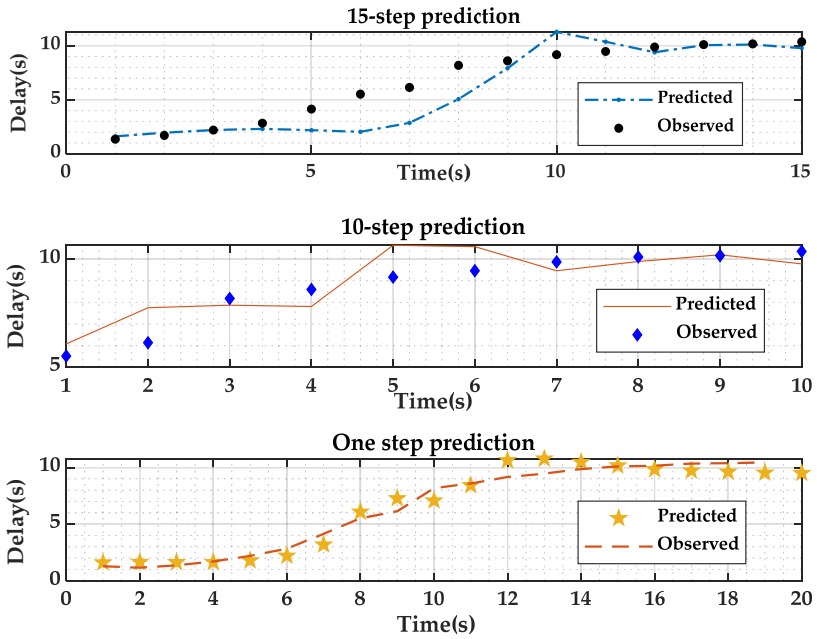

Figure 9. Predicted output models based on the k-step prediction approach with respect to the Trainrp training algorithm.

As illustrated in Figure 7, the prediction models in the Traincgf case are similar to the observed models. In all cases, we found that the expected delay increases with time until time 15, which provides the best accuracy. In Figure 8, it can be seen that when the Trainlm algorithm is used, the resulting predicted models deviate from the observed model in the case of 15-step prediction; in the case of 10-step prediction, there is a slight deviation from the observed model, but in the case of 1-step prediction, the predicted model is identical to the observed model. As shown in Figure 9, the prediction model based on the Trainrp algorithm deviates significantly from the observed model in both the 15-step and 10-step predictions. In Figure 9, the prediction model based on the Trainrp algorithm deviates from

the observed model in both the 15-step and 10-step predictions. It also deviates slightly from the observed model in the one-step prediction.

The result of the ANN training performance is shown in Figures 9–11. The figures indicate the relationship between MSE (loss) and the number of epochs during the training network for the successful training due to the lowest errors in the training, validation, and testing curves. The error generally minimizes after more training epochs but may rise when the network overfits the training data in the validation dataset. By default, training terminates after six sequential increases in the validation error, and the best performance is taken in the epoch with the minimum validation error.

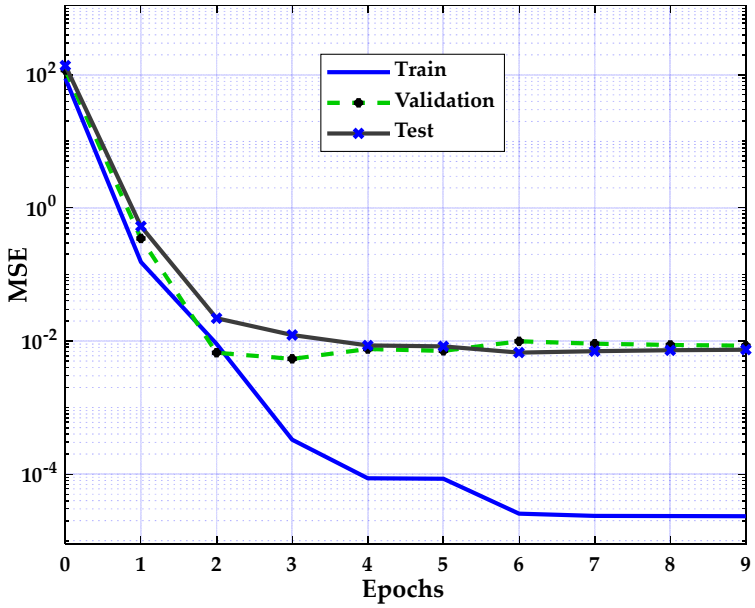

Figure 10. The best validation performance in the case of using Trainlm algorithm.

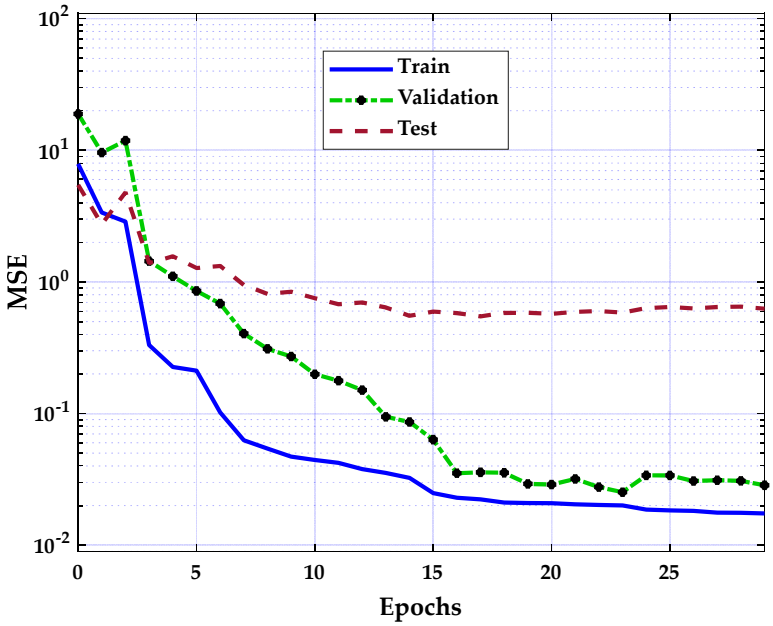

Figure 11. The best validation performance in the case of using Traincgf algorithm.

As shown in Figure 10, when the Trainlm algorithm is used, the error decreases after more training epochs. The best validation performance is 0.0053946 at the 3rd iteration of the training network. Nevertheless, the error in the validation dataset may increase

when the network begins to overfit the training process. The training is terminated after six consecutive validation errors when the error increases. In Figure 11, the model, in this case, was trained using the training algorithm Traincgf, where the best validation performance at the 23rd iteration is 0.025272. The error starts to rise in the validation set. The network begins to overfit the training process until the training terminates, where the training also increases after six sequential validation errors.

On the other hand, Figure 12 shows that the Trainrp training algorithm has the best validation performance of 0.15745 at the 14th iteration. There is a slight rise in the validation error as the network overfits the training process until the training ends. After six consecutive validation errors, the training is also terminated. Figure 13 shows that the best training performance at the 105th iteration is 0.060604 when Trainbr is used. It is noticeable that the Trainbr curve does not have a validation model because the validation stop is disabled by default, due to validation always having regularization, but Trainbr has its form.

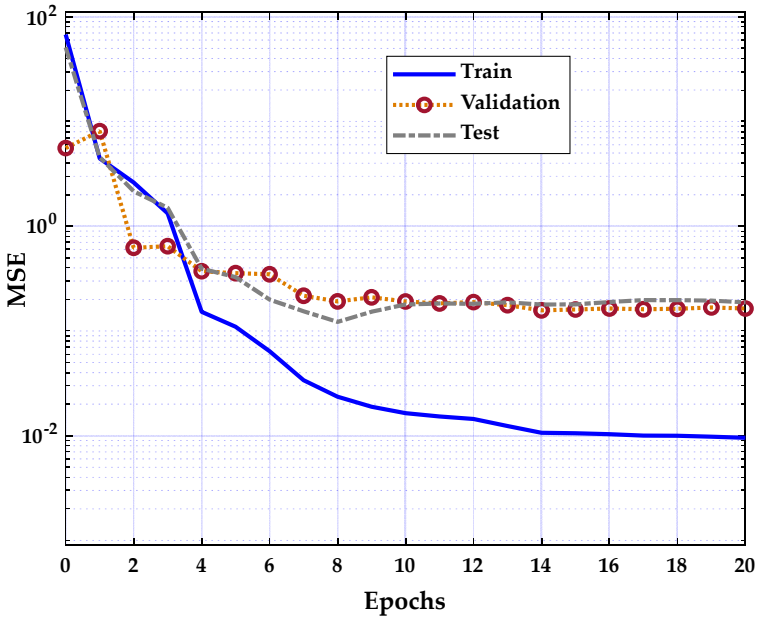

Figure 12. The best validation performance in case of using Trainrp algorithm.

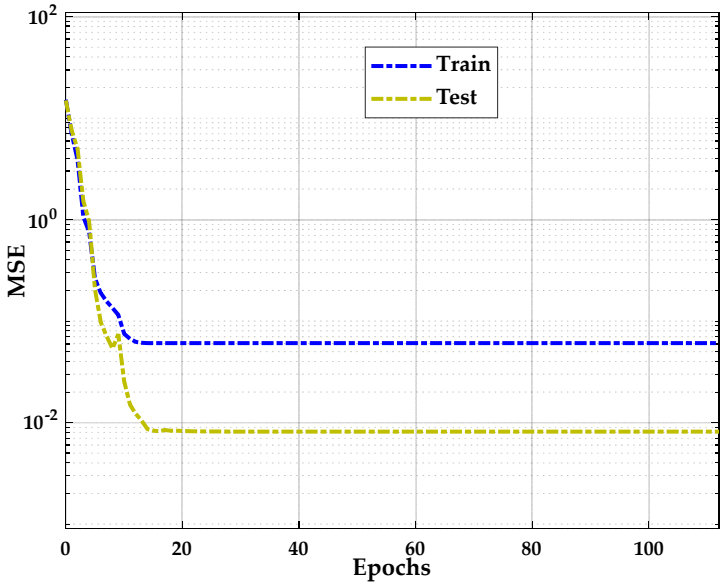

Figure 13. The best validation performance in case of using Trainbr algorithm.

7. Conclusions

This paper proposes ML methods for delay prediction in IoT and tactile internet networks, using the k-step prediction approach with the NARX-enabled RNN technique.

ANN was trained using four different algorithms: Trainbr, Traincgf, Trainlm, and Trainrp, considering the MSE loss function as a performance measure to investigate which prediction model provides optimal accuracy and maximum average improvement. The prediction accuracy was measured in terms of RMSE and MAPE as a measure of prediction accuracy.

The results show that the model predicted by the Trainbr training algorithm outperforms its competitors and has the best prediction accuracy for both 1-step prediction and 15-step prediction. Moreover, the model trained with the algorithm Traingf outperforms its competitors for the case of 10-step prediction. On the other hand, the model predicted by the algorithm Trainrp has poor prediction accuracy compared to the others.

For future work, the authors suggest the following research plans.

- The development of algorithms that can consider all the dynamic parameters of the IoT environment and more accurately predict upcoming traffic.
- The development of deep learning algorithms to predict and study the performance based on LSTM, Bi-LSTM, GRU, stacked autoencoder (SAE), and simple recurrent unit (SRU) using loss functions cross-entropy, MSE, MAE, and SSE.
- The development of deep learning based on robust loss functions using robust statistical estimators, such as Cauchy, Huber, and Fair, in the presence of outliers (anomalies).
- The development of deep-reinforced learning for network prediction and security.

Author Contributions: Conceptualization, A.R.A. and A.K.; Data curation, A.R.A.; Formal analysis, A.R.A.; Funding acquisition, R.K.; Investigation, A.R.A., O.A.M., R.K., A.P. and A.K.; Methodology, A.R.A. and A.K.; Project administration, A.K.; Resources, A.R.A., O.A.M., A.P. and A.K.; Software, A.R.A.; Validation, A.R.A., O.A.M. and A.K.; Visualization, A.R.A., O.A.M., R.K., A.P. and A.K.; Writing—original draft, A.R.A.; Writing—review & editing, A.R.A., A.P. and A.K. All authors have read and agreed to the published version of the manuscript.

Funding: The publication was prepared with the support of the grant of the President of the Russian Federation for the state support of young Russian scientists—Doctor of Science MD—2454.2020.9.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This research is based on the Applied Scientific Research under the SPbSUT state assignment 2021. The researcher A.R.A. was funded by a scholarship (Ph.D.) under the Joint Executive Program between the Arab Republic of Egypt and the Russian Federation.

Conflicts of Interest: The authors declare that there are no conflicts of interest.

References

1. Morocho-Cayamcela, M.E.; Lee, H.; Lim, W. Machine Learning for 5G/B5G Mobile and Wireless Communications: Potential, Limitations, and Future Directions. *IEEE Access* **2019**, *7*, 137184–137206. [CrossRef]
2. Feng, D.; Lai, L.; Luo, J.; Zhong, Y.; Zheng, C.; Ying, K. Ultra-reliable and low-latency communications: Applications, opportunities, and challenges. *Sci. China Inf. Sci.* **2021**, *64*, 120301. [CrossRef]
3. Abubakar, A.I.; Omeke, K.G.; Ozturk, M.; Hussain, S.; Imran, M.A. The Role of Artificial Intelligence Driven 5G Networks in COVID-19 Outbreak: Opportunities, Challenges, and Future Outlook. *Front. Commun. Netw.* **2020**, *1*, 1–22. [CrossRef]
4. Dohler, M.; Mahmoodi, T.; Lema, M.A.; Condoluci, M.; Sardis, F.; Antonakoglou, K.; Aghvami, H. Internet of skills, where robotics meets AI, 5G and the Tactile Internet. In Proceedings of the 2017 European Conference on Networks and Communications (EuCNC), Oulu, Finland, 12–15 June 2017; pp. 1–5. [CrossRef]
5. Abdellah, A.R.; Muthanna, A.; Koucheryavy, A. Robust Estimation of VANET Performance-based Robust Neural Networks Learning. In Proceedings of the 19th International Conference on Next Generation Wired/Wireless Advanced Networks and Systems (NEW2AN'2019), St. Petersburg, Russia, 26–28 August 2019.

6. Abdellah, A.R.; Muthanna, A.; Koucheryavy, A. Energy Estimation for VANET Performance Based Robust Neu-ral Networks Learning. In Proceedings of the XXII International Conference on Distributed Computer and Communication Networks: Control, Computation, Communications (DCCN'2019), Moscow, Russia, 23–27 September 2019.
7. Abdellah, A.R.; Artem, V.; Muthanna, A.; Gallyamov, D.; Koucheryavy, A. Deep Learning for IoT Traffic Prediction Based on Edge Computing. In *Distributed Computer and Communication Networks: Control, Computation, Communications. DCCN 2020 Communications in Computer and Information Science*; Vishnevskiy, V.M., Samouylov, K.E., Kozyrev, D.V., Eds.; Springer: Cham, Switzerland, 2020; Volume 1337, pp. 18–29. [CrossRef]
8. Volkov, A.; Abdellah, A.R.; Muthanna, A.; Makolkina, M.; Paramonov, A.; Koucheryavy, A. IoT Traffic Prediction with Neural Networks Learning Based on SDN Infrastructure. In *Distributed Computer and Communication Networks. DCCN 2020*; Lecture Notes in Computer, Science; Vishnevskiy, V.M., Samouylov, K.E., Kozyrev, D.V., Eds.; Springer: Cham, Switzerland, 2020; Volume 12563, pp. 64–76. [CrossRef]
9. Abdellah, A.R.; Mahmood, O.A.K.; Paramonov, A.; Koucheryavy, A. IoT traffic prediction using multi-step ahead prediction with neural network. In Proceedings of the 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Dublin, Ireland, 28–30 October 2019; pp. 1–4. [CrossRef]
10. Abdellah, A.R.; Mahmood, O.A.; Koucheryavy, A. Delay prediction in IoT using Machine Learning Approach. In Proceedings of the 2020 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Brno, Czech Republic, 5–7 October 2020; pp. 275–279. [CrossRef]
11. Abdellah, A.R.; Koucheryavy, A. Deep Learning with Long Short-Term Memory for IoT Traffic Prediction. In *Internet of Things, Smart Spaces, and Next Generation Networks and Systems (NEW2AN/SMART), Lecture Notes in Computer Science*; Galinina, O., Andreev, S., Balandin, S., Koucheryavy, Y., Eds.; Springer: Cham, Switzerland, 2020; Volume 12525, pp. 267–280. [CrossRef]
12. Abdellah, A.R.; Koucheryavy, A. VANET Traffic Prediction Using LSTM with Deep Neural Network Learning. In *Internet of Things, Smart Spaces, and Next Generation Networks and Systems (NEW2AN/ruSMART), Lecture Notes in Computer Science*; Galinina, O., Andreev, S., Balandin, S., Koucheryavy, Y., Eds.; Springer: Cham, Switzerland, 2020; Volume 12525, pp. 281–294. [CrossRef]
13. Ateeq, M.; Ishmanov, F.; Afzal, M.K.; Naeem, M. Predicting delay in IoT using deep learning: A multiparametric approach. *IEEE Access* **2019**, *7*, 62022–62031. [CrossRef]
14. Ruan, L.; Dias, M.P.I.; Wong, E. Machine Learning-Based Bandwidth Prediction for Low-Latency H2M Applications. *IEEE Internet Things J.* **2019**, *6*, 3743–3752. [CrossRef]
15. Tuli, H.; Kumar, S. Prediction analysis of delay in transferring the packets in ad-hoc networks. In Proceedings of the 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 16–18 March 2016; pp. 660–662.
16. White, G.; Palade, A.; Cabrera, C.; Clarke, S. IoTPredict: Collaborative QoS prediction in IoT. In Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications (PerCom), Athens, Greece, 19–23 March 2018; pp. 1–10.
17. Do, L.N.N.; Taherifar, N.; Vu, H.L. Survey of neural network-based models for short-term traffic state prediction. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *9*, 1–24. [CrossRef]
18. Zhang, Y.; Cheng, T.; Ren, Y. A graph deep learning method for short-term traffic forecasting on large road networks. *Comput.-Aided Civ. Infrastruct. Eng.* **2019**, *34*, 877–896. [CrossRef]
19. Tao, Y.; Wang, X.; Zhang, Y. A Multitask Learning Neural Network for Short-Term Traffic Speed Prediction and Confidence Estimation. In Proceedings of the 28th International Conference on Artificial Neural Networks (ICANN), Munich, Germany, 17–19 September 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 434–449.
20. Boussaada, Z.; Curea, O.; Remaci, A.; Camblong, H.; Mrabet Bellaaj, N. A Nonlinear Autoregressive Exogenous (NARX) Neural Network Model for the Prediction of the Daily Direct Solar Radiation. *Energies* **2018**, *11*, 620. [CrossRef]
21. Cheng, H.; Tan, P.-N.; Gao, J.; Scripps, J. Multistep-Ahead Time Series Prediction. In Proceedings of the 10th Pacific-Asia Conference, PAKDD 2006, Singapore, 9–12 April 2006; pp. 765–774.
22. Alfred, R. Performance of modeling time series using nonlinear autoregressive with exogenous input (NARX) in the network traffic forecasting. In Proceedings of the 2015 International Conference on Science in Information Technology (ICSITech), Yogyakarta, Indonesia, 27–28 October 2015; pp. 164–168. [CrossRef]
23. Menezes, J.M.P., Jr.; Barreto, G.A. Long-term time series prediction with the NARX network: An empirical evaluation. *Neurocomputing* **2008**, *71*, 3335–3343. [CrossRef]
24. Al-Amri, R.; Murugesan, R.; Man, M.; Abdulateef, A.; Al-Sharafi, M.; Alkahtani, A. A Review of Machine Learning and Deep Learning Techniques for Anomaly Detection in IoT Data. *Appl. Sci.* **2021**, *11*, 5320. [CrossRef]
25. Boutaba, R.; Salahuddin, M.A.; Limam, N.; Ayoubi, S.; Shahriar, N.; Estrada-Solano, F.; Caicedo, O.M. A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities. *J. Internet Serv. Appl.* **2018**, *9*, 16. [CrossRef]
26. Khedkar, S.P.; Canessane, R.A.; Najafi, M.L. Prediction of Traffic Generated by IoT Devices Using Statistical Learning Time Series Algorithms. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 5366222. [CrossRef]
27. Run, L.; Min, L.X.; Lu, Z.X. Research and Comparison of ARIMA and Grey Prediction Models for Subway Traffic Forecasting. In Proceedings of the International Conference on Intelligent Computing, Automation and Systems (ICICAS), Chongqing, China, 29–31 December 2020; pp. 63–67. [CrossRef]
28. Alsolami, B.; Mehmood, R.; Albeshri, A. Hybrid Statistical and Machine Learning Methods for Road Traffic Prediction: A Review and Tutorial. *Smart Infrastruct. Appl.* **2020**, 115–133. [CrossRef]

29. Lim, B.; Zohren, S. Time Series Forecasting with Deep Learning: A Survey. *Philos. Trans. R. Soc. A* **2020**, *379*, 20200209. [CrossRef] [PubMed]
30. Zhang, X.; Wang, R.; Zhang, T.; Wang, L.; Liu, Y.; Zha, Y. Short-Term Load Forecasting Based on RBM and NARX Neural Network. In *Lecture Notes in Computer Science, Proceedings of the Intelligent Computing Methodologies, ICIC 2018, Wuhan, China, 15–18 August 2018*; Huang, D.S., Gromiha, M., Han, K., Hussain, A., Eds.; Springer: Cham, Switzerland, 2018; Volume 10956. [CrossRef]
31. Landassuri-Moreno, V.M.; Bustillo-Hernández, C.L.; Carbajal-Hernández, J.J.; Fernández, L.P.S. Single-Step-Ahead and Multi-Step-Ahead Prediction with E utionary Artificial Neural Networks. In *Lecture Notes in Computer Science, Proceedings of the Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, CIARP 2013, Havana, Cuba, 20–23 November 2013*; Ruiz-Shulcloper, J., Sanniti di Baja, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; Volume 8258. [CrossRef]
32. Shang, Q.; Tan, D.; Gao, S.; Feng, L. A Hybrid Method for Traffic Incident Duration Prediction Using BOA-Optimized Random Forest Combined with Neighborhood Components Analysis. *J. Adv. Transp.* **2019**, *2019*, 4202735. [CrossRef]
33. Koh, H.K.; Geller, A.C.; VanderWeele, T.J. PMs concentration forecasting using ARIMA algorithm. In Proceedings of the IEEE 91st Vehicular Technology Conference (VTC2020-Spring), Antwerp, Belgium, 25–28 May 2020; pp. 1–5. [CrossRef]
34. Mahmood, O.A.; Khakimov, A.; Muthanna, A.; Paramonov, A. Effect of heterogeneous traffic on quality of service in 5G network. In Proceedings of the International Conference on Distributed Computer and Communication Networks, Moscow, Russia, 23–27 September 2019; Vishnevskiy, V.M., Samouylov, K.E., Kozyrev, D.V., Eds.; Springer: Cham, Switzerland, 2019; Volume 11965, pp. 469–478. [CrossRef]
35. Iversen, V.B. Teletraffic Engineering and Network Planning. Available online: http://unina.stidue.net/Reti%20di%20Telecomunicazioni/Materiale/Libri/Teletraffic%20Engineering%20handbook.pdf (accessed on 23 November 2021).