Journal of Computers Vol. 34 No. 3, June 2023, pp. 207-219
doi: 10.53106/199115992023063403015

A Machine Learning Based Approach to QoS Metrics Prediction in the Context of SDN

Hao Xu*, Xian-Bin Wan, Hui Liu

Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China

10431200653@stu.qlu.edu.cn, 15069118031@163.com, 17863658766@163.com

Received 30 June 2022; Revised 11 October 2022; Accepted 11 December 2022

Abstract. With the advent of the industrial Internet era and rapid traffic growth, network optimization is increasingly needed, and network optimization starts with knowing QoS-related metrics. In this paper, we use a machine learning approach in a theoretical SDN architecture, using traffic as the input to a machine learning model, to predict network QoS metrics, focusing on network jitter and packet loss rate. We built a LAN and deployed a time server on the LAN in order to make the time of the devices on the LAN highly consistent. Experiments were conducted under this LAN to obtain data sets about traffic and QoS metrics. Then, we used the completed trained machine learning model to predict the network jitter and packet loss rate using traffic as the input to the machine learning model. The highest R^2 values for the prediction of network jitter and packet loss reached 0.9996 and 0.939, respectively. The experiments show that a suitable machine learning model is able to predict network jitter and packet loss rate relatively accurately for a specific network topology.

Keywords: machine learning, prediction of QoS metrics, SDN, traffic, jitter, packet loss rate

1 Introduction

With the advent of the industrial Internet era, productivity has increased significantly, and the Internet of Everything is becoming more and more of a trend [1]. There is no doubt that the Internet of Everything will generate more and more traffic. QoS will be put forward more and more requirements. Among them, bandwidth, delay, jitter and packet loss are important metrics of QoS [2]. In this paper, the QoS metrics that we predict are network jitter and packet loss rate. Network jitter is the difference between the maximum and minimum delay, and packet loss ratio is the ratio of lost packets to total sent packets during data transmission. These two items are strictly required in some real-time applications, such as voice telephony. To meet the high quality real-time communication for voice telephony, the one-way delay must be in the range of 100-150 ms, the packet loss rate less than 3%, and the jitter should be within 50 ms [3]. Industrial Internet is often associated with economic benefits, and applications with high real-time requirements are essential. Examples are remote equipment manipulation and remote surgery [4]. Operators need not only to get high-definition video data of the operated mechanical parts, but also to perform accurate and precise manipulation of the machinery. Therefore, in the context of industrial Internet, advance prediction and optimization of network jitter and packet loss is essential to improve the quality of network services [5]. Theoretically, we apply SDN to the general context of the Industrial Internet. SDN is a new network architecture with separation of control plane and data plane, which enables better control of the infrastructure. The control plane has strong computational power and provides a good platform for network jitter and packet loss prediction. In this paper, we focus on SDN as the system architecture and use machine learning methods for network jitter and packet loss prediction in QoS metrics. The prediction of network jitter and packet loss rate can sense possible anomalies in the network in advance and can provide data support for the optimization of the whole network, thus improving the network service quality.

There are two traditional methods for inferring QoS metrics. The first is to send measurement packets, where QoS metrics are obtained as test packets are transmitted. The second is mathematical modeling analysis (e.g., queuing theory). The traditional method of QoS metrics prediction has the following drawbacks. The first is the way of sending measurement packets, which is costly, less stable, and does not allow inferring QoS metrics in real time. If high precision latency is measured in this way, the number of probe packets required is very large, and a large number of probe packets consumes too much bandwidth. The second way is the mathematical model

* Corresponding Author

for derivation, which is not convenient for use in real networks. This is because mathematical models are either too simplified to capture the real distribution of traffic or too complex to handle [6]. Machine learning has been greatly developed by the increasing level of hardware. This provides the most advanced direction for QoS metrics inference. The QoS metrics predicted in this paper are network jitter and packet loss rate. Existing research has only predicted network latency, and we propose for the first time the use of machine learning models for network jitter and packet loss rate prediction. The advantages of using machine learning for network jitter and packet loss rate prediction are as follows. Firstly, the machine learning model is fast. Secondly, the machine learning model has a strong ability to synthesize information and can adequately approximate arbitrarily complex nonlinear relationships [7].

In this paper, we present the first prediction of network jitter and packet loss rate using a machine learning approach. The machine learning approach is able to capture the problem accurately with low resource consumption, high immediacy, and high accuracy. There are four novel aspects of the research conducted. Firstly, for the first time, a machine learning approach is used to evaluate the prediction of network jitter and packet loss rate, and relatively good experimental results are obtained. Secondly, the datasets used in the experiments were generated on a real LAN built by ourselves, and the LAN was time-synchronized. Thirdly, data transmission is performed using UDP to simulate real-time data. Fourthly, network jitter and packet loss rate prediction with SDN as the system architecture is proposed for the first time. To summarize, our contributions are mainly in the following areas:

(1) We present the first approach to predict network jitter and packet loss rate using machine learning and demonstrate the feasibility of this approach. We use multiple machine learning methods to model traffic and jitter, and traffic and packet loss rates, respectively. The machine learning models for predicting network jitter and predicting packet loss rate are trained and evaluated with R^2 values of 0.9996 and 0.939, respectively.

(2) We conducted experiments on our own built LAN and produced real datasets for the first time. In order to improve the accuracy of the transmission rate, 10 Gigabit NICs supporting the PTP protocol were installed on the hosts in the network. A network timing server was configured on the LAN to allow the hosts to achieve time synchronization. With the configuration of the network timing server, the time synchronization accuracy of our entire LAN devices has been reached within 300 nanoseconds.

(3) In order to approach the effect of real-time data transmission, our data set generation experiments use UDP for data transmission. Network jitter and packet loss rate are important QoS metrics affecting real-time data transmission, and applications with high real-time requirements mostly use UDP for data transmission.

(4) We propose for the first time to use SDN as the system architecture, and the prediction function of network jitter and packet loss rate is deployed as an application on the SDN application layer to provide data support for the optimizer.

This paper is structured as follows. Section 2 describes the related work of previous researchers. Section 3 presents the general system architecture. Section 4 presents the methods for dataset production and machine learning modeling. In Section 5, the first part presents the experimental results of dataset production, and the second part presents the experimental results of network jitter prediction and packet loss rate prediction. Finally, Section 6 concludes the work we have done.

2 Relate Work

Network optimization is particularly important in the context of the industrial Internet, where traffic is exploding. Network optimization requires knowing the state of QoS metrics in the network first, because the predicted values of the obtained QoS metrics can provide data support for network optimization. QoS metrics are throughput, latency, jitter and packet loss, which are often used to quantify network service performance [8]. The QoS metrics that we focus on in this paper are network jitter and packet loss rate. In this section, we present the existing methods in terms of QoS metric prediction. According to the methods of QoS metrics estimation, they can be divided into two categories: traditional methods and machine learning methods. The traditional methods include methods that use measurement packets and methods that use mathematical models. At the end of this section, the differences between the work we have done and the previous work are pointed out.

2.1 Traditional Methods

The first of the traditional approaches is based on the measurement of packets, which allows a realistic measurement of the QoS metrics of a certain link state. Meseguer et al. [5] used measurement packets to obtain latency data in the process of minimizing critical traffic delays through SDN. The delay data in the paper is obtained by measuring the round-trip time (RTT) of the packet on a certain path (RTT divided by 2). Although this method is able to obtain the latency at a certain traffic state more accurately, it requires a large amount of bandwidth resources, and the latency data is not immediately available in this method.

Another traditional approach is the one that uses mathematical models to calculate QoS metrics. Gouareb et al. [9] focus on the accumulated delay assuming multipath routing of flows and the assignment of service chains in virtual networks. In terms of delay function definition, queuing theory is used. Zhang et al. [10] proposed a mathematical model to describe some key QoS parameters of wireless virtual networks, using a network evolution approach to model the quality of service of wireless networks. Roychoudhuri et al. [11] proposed a mathematical framework which is based on an end-to-end delay variation trend for packet loss prediction. Cruz [12] developed a network algorithm method for calculating delay bounds under fixed routing policies.

All of the methods mentioned above use mathematical models to calculate QoS metrics. This method solves the problem that QoS metric values are not available instantly, but still has drawbacks. Such methods are either too simple or too complex to effectively capture the real situation of the network. In recent years, machine learning-based methods for network delay prediction have been proposed, and these methods improve the shortcomings of the two traditional methods.

2.2 Machine Learning Methods

In recent years, machine learning has been gradually developed with the support of hardware. Some researchers have tried to apply machine learning and deep learning to computer networks, especially for the prediction of QoS metrics. Xiao et al. [13] proposed a database-driven system called Deep-Q. This system can infer network latency based on the corresponding network state parameters after inputting data for training. This system takes traffic as input and makes predictions about network delays. This scheme eliminates the cost of manual analysis and improves the accuracy by a factor of 3 over traditional queueing theory inference. Krasniqi et al. [6] used random forests and neural networks for end-to-end latency prediction. In the literature, the NS-3 simulation platform, mixed TCP and UDP traffic, was used as a background for generating datasets, and three different datasets were generated based on incoming traffic strength, link capacity, and propagation delay. The datasets are put into a machine learning model for training, and the completed model can then predict end-to-end delays based on the traffic matrix. Experiments have shown that random forests outperform neural networks for prediction. Mestres et al. [14] used neural networks to model traffic and delays. The literature treats the neural network as a black box with traffic as input and delay as output. The authors used the Omnet++ simulator to generate datasets with different traffic distribution, traffic intensity, topology size, and routing policies. The paper also compares neural network models with different hyperparameters. Finally, it is concluded that neural networks can make more accurate predictions of the delay in the network and that the utility of neural networks is very high after the adjustment of hyperparameters. The use of machine learning for QoS metric prediction improves the shortcomings of the traditional approach.

In the above studies, there are studies that predict QoS metrics by probing packets, but they need to consume a lot of bandwidth. There are studies that predict QoS-related metrics by mathematical models, but the prediction is not good enough. There are studies that predict network latency by various machine learning models, but they do not focus on network jitter and packet loss, and they all use data sets made by simulation software rather than real data sets. Existing studies have only predicted network latency, and we propose to use machine learning models for the first time to predict network jitter and packet loss rate. Existing studies do not use real datasets. We build LANs and produce real datasets for the first time. In terms of QoS metrics prediction, traditional methods are gradually being eliminated and machine learning methods are becoming more and more mainstream. In general, in this paper, we achieved good prediction results after training the machine learning model using real datasets. The model is able to accurately predict network jitter and packet loss rate.

3 System Architecture

The prediction of QoS metrics needs to be dependent on a system architecture. In this section, we will introduce the SDN system architecture on which QoS metrics prediction is based and the working logic of the whole system architecture.

The utility of SDN has been proven over the years. It is a very promising network model that can control the network very well. Google has deployed an SDN connecting its global data centers. It has been proven that SDN has helped Google to improve operational efficiency and significantly reduce costs [15]. The SDN control plane and data plane are decoupled, and the underlying network facilities are separated from the applications, which facilitates efficient use of resources and resource provisioning [16]. The control plane has strong computational power, which provides computational conditions for machine learning models and is more conducive to real-time QoS metric prediction [17]. Moreover, a key responsibility of the SDN controller is to optimize the network quality of service, which is in line with our ultimate goal. Therefore, the system architecture of our research content is in the context of SDN, and the QoS metric prediction proposed in this paper is deployed as an application in the application layer of the SDN-based system architecture.

The system architecture we use is based on SDN, as shown in Fig. 1, which is our system architecture diagram. The application layer of the system architecture diagram mainly consists of three parts: optimizer, machine learning model, and database, which are the focus of our research. The optimizer is used to optimize the network performance. The machine learning model predicts the QoS metrics under a certain state feature, and the database is used to store different network features and the QoS metrics corresponding to these features. The working logic of the whole system is as follows. The system first collects information such as device parameters, traffic, delay, network jitter, and packet loss rate at the data layer and stores them in the database. Then, the machine learning model is trained based on the data in the database, and once the training is completed, the preparation for network optimization is done. When the optimizer gets the optimization command, it passes the traffic, topology, and device parameters to the machine learning model, which calculates the latency, network jitter, and packet loss rate in this state based on these data and sends them to the optimizer. The optimizer uses this data to execute an optimization algorithm that explores the performance of the target solution and finds the best configuration for the overall network in an iterative manner.

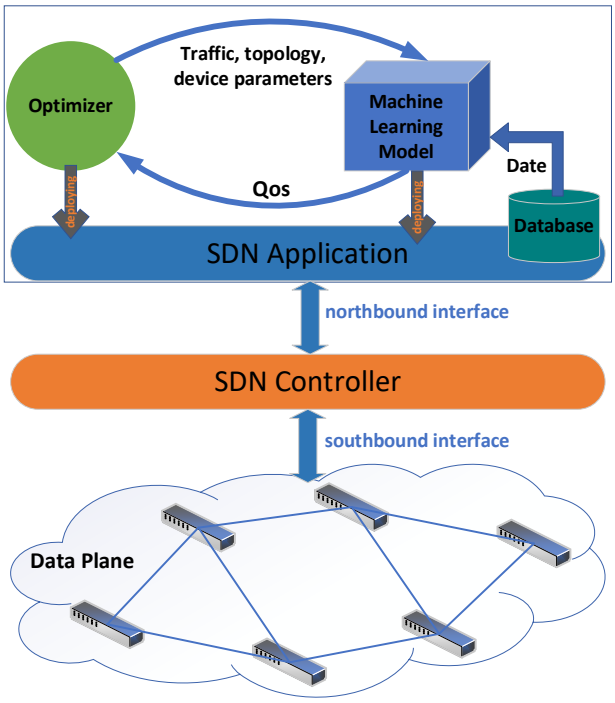


Fig. 1. System architecture diagram

In this paper, we focus on the module of machine learning models for predicting QoS metrics, with particular attention to network jitter and packet loss rate among QoS metrics. To explore the correlation, we built a real network and analyzed the data captured by Wireshark to obtain a dataset containing network jitter and packet loss rate. Then, the relationships between network jitter and network traffic, packet loss rate, and network traffic were explored by modeling using machine learning models.

4 Methods of Making Datasets and Modeling

Our experiment is roughly divided into two parts: the first part is to generate and analyze data, and the second part is to use machine learning models to complete the prediction of relevant data and the evaluation of model effects. In this part, we introduce some methods used for data generation, data analysis, and machine learning modeling, mainly including methods for network topology construction, methods for producing data sets, data analysis algorithms, and methods for modeling and model evaluation.

4.1 Network Topology

The first step of the experiment is to build a local area network (LAN) under which data transmission and data capture are performed. To explore the relationship between traffic and delay, and to explore the relationship between traffic and network jitter, we built a simple LAN with the experimental topology shown in Fig. 2. We built a LAN using two host servers, two switches, and a time server. In order to support the PTP timing service and improve the packet sending performance of the two host servers, we configured 10 Gigabit NICs for the two host servers. In order to be closer to the real application situation, we use fiber optic connections between the host servers and the switch. We use super 6 Gigabit cables between the two switches and between the switch and the timing server. In order to ensure the accuracy of the calculated delay, we used a network timing server to synchronize the network time for the whole LAN, and the time synchronization accuracy of the devices on the whole LAN was within 300 nanoseconds.

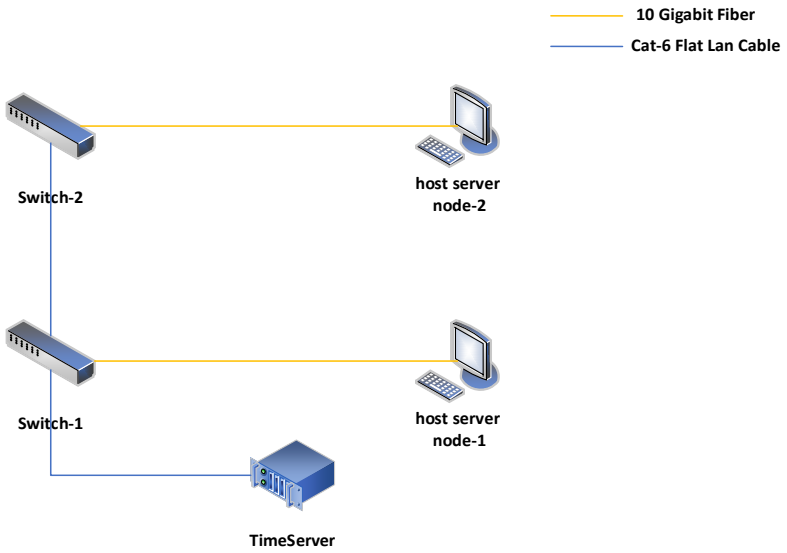


Fig. 2. Network topology

4.2 Dataset Production

After the LAN is set up, we start the data transmission experiment, where host server node 1 sends packets to

host server node 2 at different rates to represent the different traffic intensities in the LAN. In order to be able to send data at different sending rates, we use multi-threaded socket-based programming to complete the packet sending function. Packet capturing is performed on two host server nodes using Wireshark, capturing approximately 3000 UDP packets for each traffic state. Finally, data analysis was performed to derive the average latency, jitter, packet loss rate, latency maximum, latency minimum, and latency variance for each traffic state. As shown in Fig. 3, this is the graph of the sending rate variation during the experiment. The experiments were conducted 1090 times in total, starting with 10 KB/s for the first 1000 times and increasing the rate by 10 KB/s for each experiment, and starting with 11000 KB/s for the next 90 times and increasing the rate by 1000 KB/s for each experiment. To ensure the accuracy of the transmission rate, the sending rate and receiving rate were monitored at the host server node using traffic monitoring software (bmon). The sending rate and receiving rate are monitored from time to time on the switch side using the web interface.

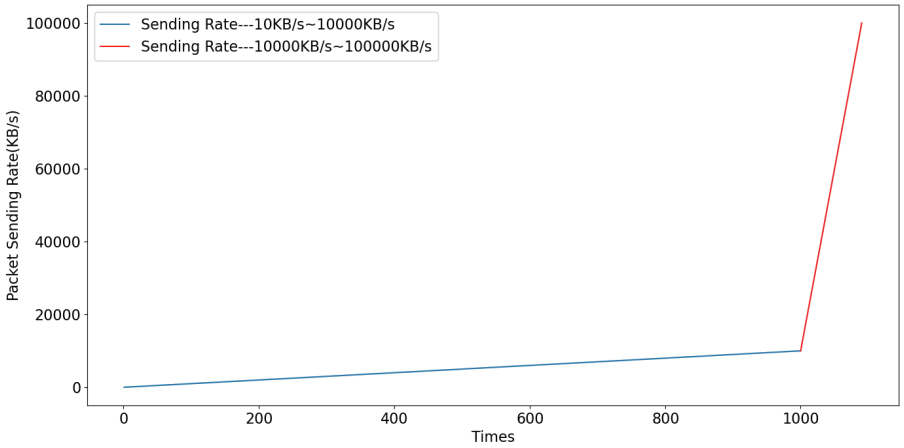


Fig. 3. Sending rate

4.3 Data Analysis Algorithm

After capturing the data by Wireshark on the transmitter and receiver sides, data analysis is performed to obtain data such as network jitter, packet loss rate, and average delay. A data analysis algorithm is designed in this section to ensure the accuracy of the data such as average delay, network jitter, and packet loss rate.

Algorithm 1. Data analysis

Input: T1[1...1090], T2[1...1090] /*The data captured by Wireshark is stored in the form of a table*/
Output: Loss_rate[1...1090], Average_delay[1...1090], Network_jitter[1...1090]

```
For i ← 1; i ≤ 1090 do
T1, T2 = SelectProtocolUDP(T1[i], T2[i]);
Loss_rate[i] = ;  /* len(T), which indicates how many rows of data in T*/
If Unique(T1.Identification)==False || Unique(T2.Identification)==False then
    T1, T2 = DropDuplicateIdentification(T1, T2);
End If
Tab = Merge(T1,T2, “Identification”);
Tab = Balance(Tab);
Average_delay[i] = ;  /* and  represent the time of the same packet at the sender and receiver respectively */
Network_jitter[i] = max() - min();
End For
Return Loss_rate, Average_delay, Network_jitter
```

In the following we explain the key points in the algorithm.

Input and Output. The data captured by Wireshark on the sender and receiver sides is used as input. The output data includes packet loss rate, average delay, and network jitter. For convenience, we use *T*1[1...1090], *T*2[1...1090] such arrays to represent the data captured at the sender and receiver side of the experiment. For example, *T*1[*i*] denotes the data captured on the sender side of the *ith* experiment, and *T*2[*i*] denotes the data captured on the receiver side of the *ith* experiment. Similarly, since the experiment is conducted 1090 times in total, the number of packet loss rates, average delay, and network jitter are also 1090, respectively. So we store the packet loss rate, average delay, and network jitter of each experiment into arrays as well, and finally output these arrays.

Related functions. The experiment captures packets in 1090 different traffic states, and a total of 1090 experiments are performed, so the data analysis needs to be performed 1090 times as well. The captured packets were not only of the UDP type, so the *SelectProtocolUDP* function was used to pick out the packet using the UDP protocol. In the table exported by Wireshark, *Identification* is used as a unique identifier for the packets, and there may be duplicate values in the case of too many packets sent. To ensure the uniqueness of the packets and the accuracy of the delay calculation, we use the *Unique* function to verify whether the values in the *Identification* column are duplicates. If there are duplicate values in the *Identification* column, use the *DropDuplicateIdentification* function to delete all packets with duplicate *Identification* values. The *Merge* function inner joins two tables, *T*1 and *T*2, based on the *Identification* value to generate a new table. The *Balance* function is used to balance the data with the purpose of avoiding extreme data effects. The procedures are as follows. The function first sorts the latency of approximately 3000 packets per experiment from largest to smallest. For the first 1000 experiments, the first 1/100 and the last 1/100 latency values of each experiment are removed. The last 90 experiments, with fewer low values, removed the latency data for the first 1/1000 and last 1/1000 of each experiment in order to balance.

Calculation formula. The algorithm has three formulas in total, which are the formulas for packet loss rate, average delay, and network jitter. The logic of the packet loss rate formula in the third line is as follows. The number of UDP packets captured by the sender is subtracted from the number of UDP packets captured by the receiver, and the difference is divided by the number of UDP packets captured by the sender. The logic for calculating the average latency formula in the ninth row is as follows. The table after the inner connection contains all the time information of each packet, represents the time of each packet arriving at the destination host, represents the sending time of each packet, and is the transmission delay of each packet. The delays of all packets are summed up and then divided by the number of packets to get the average delay. The logic for calculating the network jitter formula in the tenth row is as follows. The maximum delay minus the minimum delay for each experiment.

4.4 Machine Learning and Evaluation Metrics

In order to verify the feasibility of machine learning models for QoS metric prediction and compare the accuracy of different models, we use various machine learning models in our experiments, including GDBT, traditional neural networks, and convolutional neural networks. Machine learning modeling is to consider a machine learning model as a black box, with traffic as input and network jitter and packet loss rate as output. Our experiments use machine learning modeling, and the modeling task can be represented as formula (1).

$$J, L = f(T). \tag{1}$$

J denotes the delay jitter, *L* denotes the packet loss rate, the function *f*() denotes the machine learning model used, and denotes the traffic in the network. We are predicting under specific network characteristics, and the independent variables of the function do not consider information such as device parameters, routing policies, etc., because these parameters are fixed under specific network characteristics.

We use R^2 as the evaluation indicator of the model. The evaluation indicator is R^2, which reflects the proportion of the dependent variable that can be explained by the independent variable through the regression relationship. Here is the formula of R^2, $\hat{y}_i$ is the predicted value, y_i is the true value, and $\overline{y}$ is the average of the true values. The closer the R^2 value is to 1, the better the model fit is. We evaluate the model by calculating R^2 for the test set, and R^2 is shown in formula (2).

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \\ = 1 - \frac{MSE(y, \hat{y})}{Var(y)}. \tag{2}$$

MSE is used as an auxiliary evaluation metric and as a loss function. The smaller the *MSE*, the better, proving that the difference between the true value and the predicted value is smaller. *MSE* is shown in formula (3).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{3}$$

5 Experiment Result and Discussion

In this section, we present the experimental results. The experimental results are presented in two parts, the experimental results of the dataset production part and the experimental results of the machine learning part. In the machine learning experimental results section, they are also presented in two parts, which are the prediction of network jitter and the prediction of packet loss rate.

5.1 Experimental Results of Dataset Production Part

We present the experimental results of the dataset production with two images. First, the distribution of latency in four different traffic states is analyzed, and then the trends of average latency, network jitter, and packet loss rate obtained from the experiments are analyzed. About 3000 packets were collected for each traffic state, and each packet had a corresponding delay. As shown in Fig. 4, we selected four different traffic states from a total of 1090 traffic states to observe the packet latency under different traffic states. Fig. 4(a) to Fig. 4(d) show the latency images in the context of sending rates of 60 KB/s, 600 KB/s, 6000 KB/s, and 60000 KB/s, respectively. The horizontal coordinate represents each packet in a certain traffic state, and the vertical coordinate represents the packet delay. It can be clearly seen that as the network traffic keeps getting larger, more and more packets will have high latency values. This indicates that packet latency is related to the network traffic size. Moreover, the graph shows that there is a critical point for the maximum latency, which in our experiments is around 12,000,000 nanoseconds.

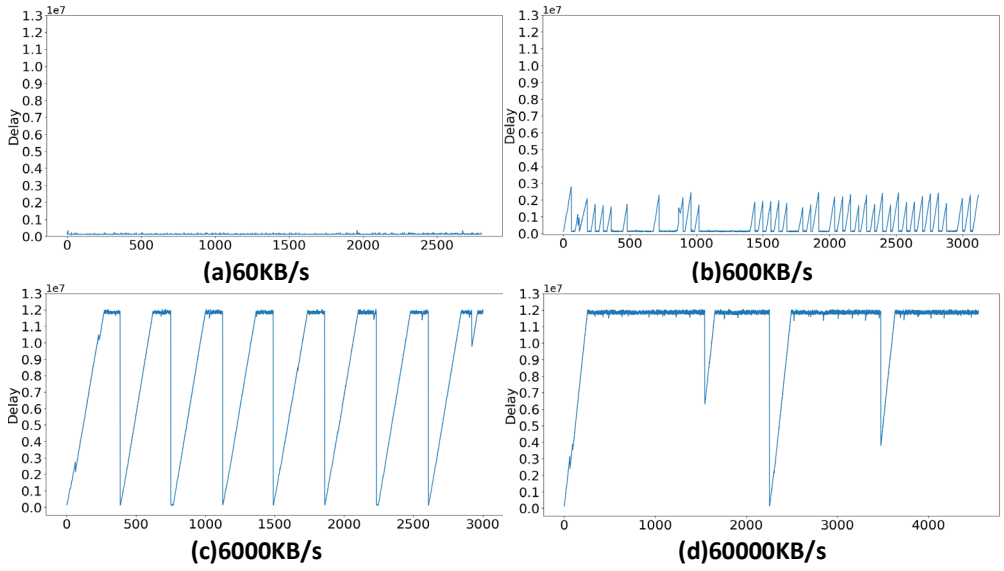


Fig. 4. Delay for different traffic backgrounds

We analyze the experimental data obtained from Wireshark captures to obtain the average latency, network jitter, and packet loss rate for each experiment. Based on the traffic growth rate shown in Fig. 3, we represent the data in two segments with different colors. As shown in Fig. 5, the average latency, network jitter, and packet loss rate for the first 1000 experiments are shown in blue, and the average latency, network jitter, and packet loss rate for the last 90 experiments are shown in red. The average delay image of each experiment is shown in Fig. 5(a). We can see that the average delay keeps getting larger as the traffic keeps increasing. The network jitter image for each experiment is shown in Fig. 5(b). We can see that the network jitter keeps increasing as the traffic increases, and after the network jitter increases to a certain level, it will keep a certain range of values constant. As shown in Fig. 5(c), this is the packet loss rate image. As we can see, the packet loss rate is 0 in the low traffic state, and it is increasing as the traffic increases. From the three plots in Fig. 5, we can get the preliminary analysis that there is a correlation between the latency, network jitter, and packet loss rate and the size of network traffic. Therefore, we try to explore this correlation using a machine learning approach. It is worth noting that in our experiments, the UDP packet size is set to 10240B and the packets are significantly larger than the MTU, so the packets are fragmented when performing data transmission. The relatively large packets, packet fragmentation, and continuous sending of MTU-sized packets all add to the packet loss rate, so our measured packet loss rate may be greater than the general usage case. However, the general trend of the data we obtained is correct, so it does not affect the exploration of the packet loss rate.

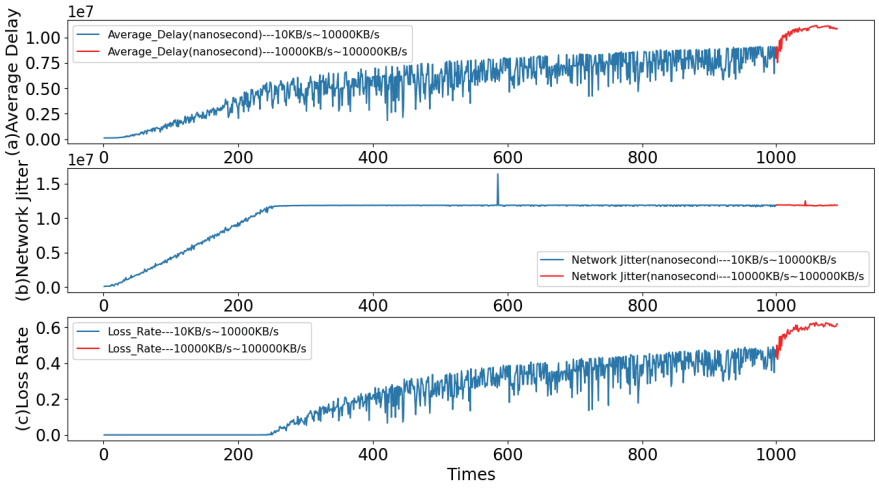


Fig. 5. QoS metrics analysis image

5.2 Experiment Results of Machine Learning Part

In this section, we present experimental results on the prediction of network jitter and packet loss rate using network traffic under specific network characteristics. In practical use, we need to choose the model that is best suited to the current environment to complete the prediction work, so our experiments compare different models. In the following, we present the experimental results in two parts: the experimental results of network jitter prediction and the experimental results of packet loss rate prediction.

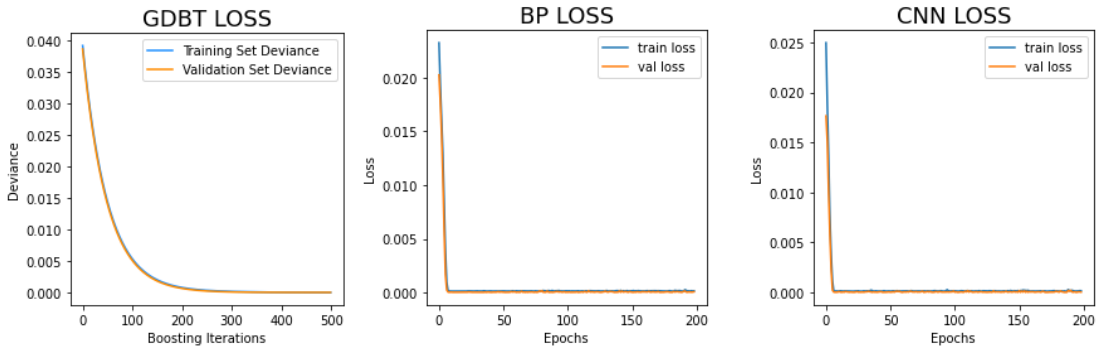
Experimental Results of Network Jitter Prediction. The experiments were performed to predict the network jitter using CNN, the BP neural network, and GDBT. We divided the total data set into training and validation sets in the ratio of 8:2. In terms of implementation, CNN and BP neural networks were done using the Tensorflow library for the modeling task, and GDBT was implemented using the scikit-learn library for modeling. A convolutional neural network, which is a neural network with four convolutional layers added on top of it. As Table 1 shows the parameters and training results of the machine learning models, the BP neural network and the CNN differ only in the number of layers, but the other parameters of both models remain the same. The activation

function is relu, the optimizer is Adam (learning rate = 0.001, beta_1 = 0.9, beta_2 = 0.999.), the loss function is mse, and the model monitoring metric is R^2. The data was processed in batches, each sized at 32. Similarly, we have listed the parameters and results of the GDBT model in the table. After the three models were trained, the R^2 values were 0.9993, 0.9992 and 0.9996, and the losses were reduced to 2.9098×10^{-5}, 2.5168×10^{-5} and 2.0215×10^{-5}, respectively.

Table 1. Comparison of different models

Model	Params	R^2	Loss
Gradient Boosting Decision Tree (GDBT)	n_estimators: 500 max_depth: 4 min_samples_split: 5 learning_rate: 0.01 loss: mse	0.9993	2.9098×10^{-5}
Neural Networks (NN)	num of layers: 5 activation function: relu optimizer: Adam loss: mse batch size: 32	0.9992	2.5168×10^{-5}
Convolutional Neural Networks (CNN)	num of layers: 10 activation function: relu optimizer: Adam loss: mse batch size: 32	0.9996	2.0215×10^{-5}

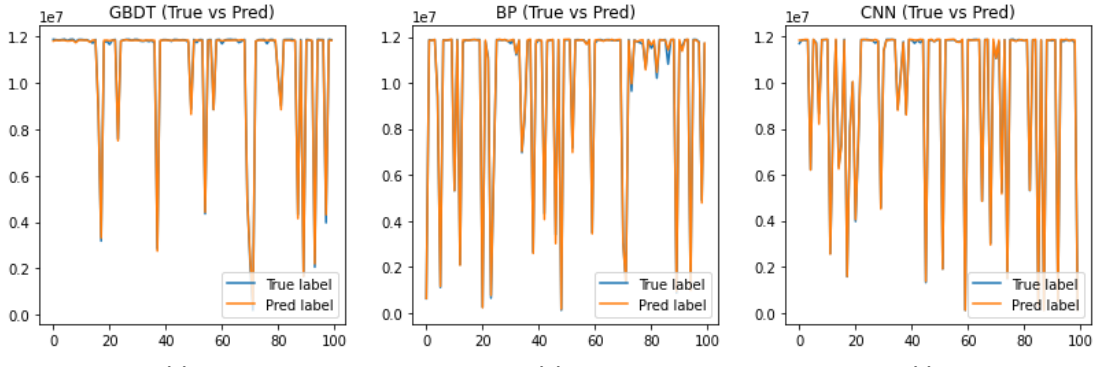
We recorded the loss variation for different models, and Fig. 6 shows the loss variation for the three models. It is obvious from the figure that the losses of the three models are reduced to very low levels after training. In Fig. 6(a) is the loss variation of GDBT, Fig. 6(b) is the loss variation of the BP neural network, and Fig. 6(c) is the loss variation of CNN. In the three plots, the orange curve represents the validation set loss and the blue curve represents the training set loss. It is clear from the three plots that the loss values of all three models are reduced to very low levels, and all three models achieve a good fit.



(a) GDBT Loss (b) BP Loss (c) CNN Loss

Fig. 6. Loss (MSE)

We inverse normalized the network jitter obtained from the model predictions and compared them with the true values. As shown in Fig. 7, the real values are compared with the three model predicted values (after inverse normalization) respectively. In the figure, the orange curve is the predicted value and the blue curve is the true value. The units of the data in all three plots are nanoseconds. As shown in the figure, all three machine learning models can predict the network jitter very well, and the predicted values differ very little from the true values.



(a) GBDT (True vs Pred) (b) BP (True vs Pred) (c) CNN (True vs Pred)

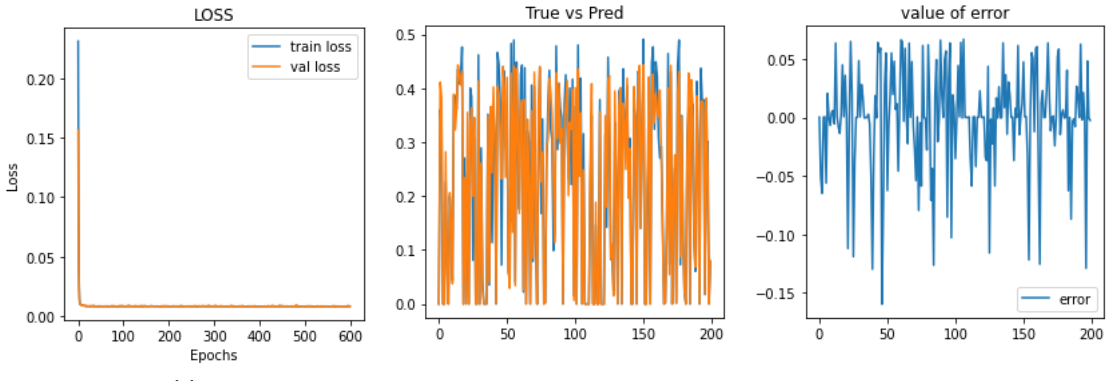
Fig. 7. Comparison of true values and predicted values (nanosecond)

Experimental Results of Packet Loss Rate Prediction. We used a BP neural network to predict the packet loss rate, and the model was still able to achieve a relatively good fit. In this experiment, we used the Adam optimizer in the machine learning model. As shown in Table 2, the R^2 value is 0.939 and the loss is 0.0070.

Table 2. Experimental results

Model	R^2	Loss
BP neural network	0.939	0.0070

Fig. 8(a) shows the decreasing process of the model loss. It can be seen from the figure that the loss decreases quickly and the loss value is small enough when it is stable. We also compared the real and predicted values of packet loss rate, and the comparison results are shown in Fig. 8(b). The blue line in the image is the true value and the orange line is the predicted value. It can be clearly seen that although there is some error between the true and predicted values, they can be fitted better overall. Finally, as shown in Fig. 8(c), we have calculated the error value of the packet loss rate. The error of packet loss rate is the difference between the real value of the packet loss rate and the predicted value of the packet loss rate. As can be seen from the figure, the vast majority of the error values are between +0.05 and -0.05. We sum the absolute values of all packet loss rate errors and then average them, and the final average error value is 0.029. From these three pictures, we can see that the prediction is good enough.



(a) Loss (b) True vs Pred (c) Value of error

Fig. 8. Experimental results

6 Conclusion

In this paper, we present for the first time a method to predict network jitter and packet loss rate using a machine learning model. We train the machine learning model using a dataset of our own making, and the trained model takes traffic as input and network jitter or packet loss rate as output. We used three machine learning models (CNN, BP neural network, and GDBT) to perform regression prediction of jitter, and all three machine learning models have good prediction results for network jitter with R^2 values above 0.999. Then, we used a BP neural network to predict the packet loss rate, and the model was still able to predict the packet loss rate very well. The experimental data proves that a suitable machine learning model can accurately predict the network jitter and packet loss rate. Our experiments can be used to sense possible anomalies in the network and can provide data to support the optimization of the entire network, thus improving the quality of network services. The advantages of our experimental approach are its high accuracy and practicality. There are two limitations. The first is that SDN networks are not yet deployed on a large scale, and it may take a longer time to apply them in a real environment. The second is that the model needs to be trained using a large amount of data before prediction.

In the future, we expect to be able to deploy our proposed method as an application to SDN networks and apply it to SDN network optimization. The research directions that we will conduct in subsequent work are as follows.

(1) Most of the existing delay predictions focus on path delays. We will conduct research on more fine-grained delays. We will study the relationship between network traffic and link delays.

(2) We will look at how to train better machine learning models with fewer pieces of data.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant No. 92067108, the Shandong Provincial Natural Science Foundation of China under Grant No. ZR2020MF057.

References

[1] S. Wang, L. Nie, G. Li, Y. Wu, Z. Ning, A MultiTask Learning-based Network Traffic Prediction Approach for SDN-enabled Industrial Internet of Things, IEEE Transactions on Industrial Informatics 18(11)(2022) 7475-7483.

[2] H.E. Egilmez, A.M. Tekalp, Distributed QoS Architectures for Multimedia Streaming Over Software Defined Networks, IEEE Transactions on Multimedia 16(6)(2014) 1597-1609.

[3] M.J. Karam, F.A. Tobagi, Analysis of the delay and jitter of voice traffic over the Internet, in: Proc. 2001 20th Annual Joint Conference of the IEEE Computer and Communications Society, 2001.

[4] F. Boabang, A. Ebrahimzadeh, R.H. Glitho, H. Elbiaze, M. Maier, F. Belqasmi, A Machine Learning Framework for Handling Delayed/Lost Packets in Tactile Internet Remote Robotic Surgery, IEEE Transactions on Network and Service Management 18(4)(2021) 4829-4845.

[5] J.M. Llopis, J. Pieczerak, T. Janaszka, Minimizing Latency of Critical Traffic through SDN, in: Proc. 2016 IEEE International Conference on Networking, Architecture and Storage, 2016.

[6] F. Krasniqi, J. Elias, J. Leguay, A.E.C. Redondi, End-to-end Delay Prediction Based on Traffic Matrix Sampling, in: Proc. 2020 IEEE Conference on Computer Communications Workshops, 2020.

[7] H. Huang, X. Zhu, J. Bi, W. Cao, X. Zhang, Machine Learning for Broad-Sensed Internet Congestion Control and Avoidance: A Comprehensive Survey, IEEE Access 9(2021) 31525-31545.

[8] P. Charonyktakis, M. Plakia, I. Tsamardinos, M. Papadopouli, On User-Centric Modular QoE Prediction for VoIP Based on Machine-Learning Algorithms, IEEE Transactions on Mobile Computing 15(6)(2016) 1443-1456.

[9] R. Gouareb, V. Friderikos, A.H. Aghvami, Delay Sensitive Virtual Network Function Placement and Routing, in: Proc. 2018 International Conference on Telecommunications, 2018.

[10] L. Zhang, J. Liu, K. Yang, Quality of Service Modelling of Virtualized Wireless Networks: A Network Calculus Approach, Mobile Networks and Applications 19(4)(2014) 572-582.

[11] L. Roychoudhuri, E.S. Al-Shaer, Real-time packet loss prediction based on end-to-end delay variation, IEEE Transactions on Network and Service Management 2(1)(2005) 29-38.

[12] R.L. Cruz, A calculus for network delay. I. Network elements in isolation, IEEE Transactions on Information Theory 37(1)(1991) 114-131.

[13] S. Xiao, D. He, Z. Gong, Deep-Q: Traffic-driven QoS Inference using Deep Generative Network, in: Proc. 2018 Workshop on Network Meets AI & ML, 2018.

[14] A. Mestres, E. Alarcón, Y. Ji, A. Cabellos-Aparicio, Understanding the Modeling of Computer Network Delays using Neural Networks, in: Proc. 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks, 2018.
[15] D. Kreutz, F.M.V. Ramos, P. E. Veríssimo, C.E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-Defined Networking: A Comprehensive Survey, Proceedings of the IEEE 103(1)(2015) 14-76.
[16] M. Karakus, A. Durresi, Quality of Service (QoS) in Software Defined Networking (SDN): A survey, Journal of Network and Computer Applications 80(2017) 200-218.
[17] J.W. Guck, A.V. Bemten, W. Kellerer, DetServ: Network Models for Real-Time QoS Provisioning in SDN-Based Industrial Environments, IEEE Transactions on Network and Service Management 14(4)(2017) 1003-1017.