Proceedings of the 5th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/29/20230803

# ML-based SDN performance prediction

**Zihao Liu**

School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan 430081, China

ZihaoLiu0314@163.com

**Abstract.** Software-defined networking (SDN), a new type of network architecture with the advantages of programmability and centralized management, has become a promising solution for managing and optimizing network traffic in modern data centers. However, designing efficient SDN controllers and applications requires a deep understanding of their network performance characteristics. In this work, we implement a machine learning-based method for SDN performance prediction. Our method uses supervised learning to build a training model based on a set of publicly available real network traffic datasets and then uses the model to predict future network performance metrics, such as RTT, S2C, and C2C. Our method is evaluated in two different SDN distributed deployment structures, demonstrating its effectiveness in network performance prediction. We observed that XGBoost achieves the lowest error in most of the cases in terms of MAE, RMSE and MAPE, and feature selection through PCA fails to further improve the prediction performance of XGBoost.

**Keywords:** machine learning, software-defined networking, computer networks, performance prediction.

## 1. Introduction

In recent years, the scale and application of networks have shown explosive growth, which has brought many challenges to current network operations, such as the rapid expansion of network scale, increasingly complex network equipment, and unpredictable traffic models. The software-defined network (SDN) decouples the control plane and the data forwarding plane of the traditional network, making network management and control more flexible and efficient, making it programmable and centralized management, and has become a new paradigm of network management. In SDN, the interface between the data plane and the control plane is realized through the southbound communication protocol, and the most widely used protocol is the OpenFlow protocol. The network control plane is managed by a centralized controller that can dynamically configure network traffic based on changing network conditions. SDN has been widely used in modern data centers as a means to improve network scalability, flexibility, and performance. SDN has also been used in 5G and 6G networks, satellite networks and the Internet of Things (IoT) [1-3].

With the widespread application of SDN, the performance prediction of SDN networks has become increasingly important. The performance prediction of SDN networks can provide strong support for network management and optimization and can also improve the performance and reliability of SDN networks. If the network traffic data are analyzed and predicted in advance, it is possible to know in

DOI: 10.54254/2755-2721/29/20230803

advance which traffic data will affect the network performance index (QoS) to find the optimal network deployment strategy, reduce engineering overhead, and improve network transmission efficiency. However, due to the complexity of modern network dynamics, the diversity of network applications, and the unpredictability of network traffic patterns, predicting network performance is a challenging task. However, unlike traditional networks, due to the characteristics of SDN's own architecture, it has the advantages of easy collection of training data and easy deployment of strategies for performance prediction on SDN networks.

At present, research on SDN network performance prediction is mainly based on statistical analysis and mathematical modeling. However, these methods have certain limitations in terms of the complexity and dynamics of SDN networks. Statistical analysis and mathematical modeling are a huge challenge in the face of factors such as a huge amount of network data and constantly changing network parameter configurations. At the same time, its method cannot adapt to the ever-changing virtual network environment, and it no longer has scalability and analysis capabilities. In recent years, machine learning, as a powerful data analysis tool, has been widely used in various research fields, e.g., financial and transportation domains [4-7]. The development of machine learning also provides new ideas for SDN network performance prediction [8-12]. Machine learning technology can learn and extract rules from a large amount of historical data, thereby predicting future SDN network performance, and can be adapted to different network environments [13-15].

In this paper, we propose a method for SDN performance prediction based on machine learning techniques. This method uses a public network dataset constructed by performing simulations based on different network configurations and collecting corresponding network performance indicators. We tried to change the parameters of the feature selection method of PCA and machine learning models such as AdaBoost, decision tree, random forest, and XGBoost and considered the accuracy and effectiveness of its predictions in multiple dimensions. We observed that XGBoost achieves the lowest error in most of the cases in terms of MAE, RMSE and MAPE, and feature selection through PCA fails to further improve the prediction performance of XGBoost.

The rest of the paper is organized as follows: Section 2 examines related work. Section 3 describes the generation and content of the datasets used in this work. Section 4 details the machine learning models we use and the related methods of operation. Section 5 presents the experiments, model calculation process, and evaluation results. Finally, Section 6 presents the conclusion and future work.

## 2. Related work

With the rapid development and wide application of machine learning, many scholars have proposed research methods for using machine learning in the OpenFlow environment. Its application in SDN includes many target orientations, including traffic classification, anomaly detection, routing optimization and final performance. forecast, etc. However, they are all limited. Focusing on network performance prediction, in this section, we briefly introduce this literature as well as other areas where machine learning has been used to improve virtual networks.

For network performance prediction, [16] discusses how to use big data and machine learning technology to analyze and manage network quality of service (QoS) in SDN. This research establishes and quantifies the relationship between various network traffic data KPIs and QoS and analyzes the reasons for their association based on the results. In [17], two learning methods, regression tree and random forest, are used to predict the QoS in the OpenFlow network. The author compares the advantages and disadvantages of the two methods from different dimensions and finds that reducing the data dimension during processing has little effect on the estimation accuracy and will greatly shorten the training time of the model. Both [18] and [19] discuss how to use machine learning-based models to predict performance in data center networks and use their results to tune relevant parameters. To realize the efficient performance of Incast and coexistence of long and short traffic in DCN. At the same time, both papers demonstrated the universality of the machine learning model compared with the traditional analysis model and extended the model from a single Incast to a mixed Incast and elephant scenario. Both [20] and [21] proposed a processing scheme to optimize the quality of online video, and [20]

DOI: 10.54254/2755-2721/29/20230803

designed a learning model using a deep recurrent neural network (RNN) and long short-term memory (LSTM) to predict the future of DASH clients to determine the quality that will be cached. In [21], it predicts the quality of experience (QoE) of video streaming, aiming at evaluating user satisfaction with the service. The authors used five different learning algorithms to predict user perception and compared the results of each method to determine the best learning algorithm.

There are also different interpretations of the other functions of machine learning in SDN. [22] introduced a network intrusion detection system (IDS) based on a programmable P4 switch, which uses machine learning algorithms for anomaly detection and implements an anomaly detection mechanism in high-speed networks. [23] proposed a new routing protocol PFR, which implements path selection based on dynamic routing topology state and traffic prediction results based on machine learning. [24] presents a method for modeling and predicting daily traffic patterns in telecommunication networks. These methods predict and optimize network performance through machine learning algorithms, thereby improving the efficiency and reliability of SDN networks. In [25], the autoregressive average model (ARIMA) is used to predict the load of the SDN controller to optimize the migration operation to support low-latency communication and to adopt the corresponding load balancing strategy through its prediction results.

In general, these studies show that the application of machine learning in SDN performance prediction is of great significance and provides inspiration for our research. These methods can provide decision support and optimization suggestions for network managers and improve network efficiency and reliability. However, some methods and frameworks in these studies still have limitations. For example, in past studies, most data simulations exist in a specific network topology, the SDN distributed architecture is fixed, and the dataset is single. In our work, we use 50 real network topologies, carry them on two different SDN distributed architectures, and perform network simulation on OpenFlow based on the OMNeT++ suite, which has better versatility and practicability. At the same time, most of the current performance prediction methods are based on machine learning to continuously update, analyze and process data traffic. Our work is forward-looking, using machine learning to make predictions before SDN deployment, which greatly shortens the data analysis and processing time and avoids unnecessary network delays during data processing.

## 3. Dataset description

In our research, we use a publicly available real SDN network traffic dataset to train and validate our model approach [26]. The dataset contains network traffic data collected from virtual simulation networks of two different SDN distributed controller deployment structures based on 50 real network topologies. Because of the supervised learning method we use, each sample is composed of its feature value and the associated target value. Now, we describe its network system configuration and dataset input and output.

### *3.1. Settings*

First, we selected 50 real and effective network topologies from the network topologies around the world collected in the Internet Topology Zoo as the topological structure of this simulation architecture to avoid the error of experimental results caused by the single network structure.

In addition, this SDN simulation experiment uses the OpenFlow network based on the OMNeT++ framework. OMNeT++ is an open source component-based modular development network simulation platform. With a powerful graphical interface and embedded simulation kernel, it can easily define the network topology and has functions such as programming, debugging and tracking support. It is mainly used for the simulation of communication networks and distributed systems. OpenFlow OMNeT++ Suite (OOS) combines these two tools to provide an OpenFlow-based network simulation platform. Using OOS, it is possible to build topologies containing multiple OpenFlow switches and hosts and to test various network configurations and policies in simulation. In OOS, controller programs can be written that can run in simulation and communicate with switches using the OpenFlow protocol for network control and management. At the same time, OOS also provides a variety of performance

DOI: 10.54254/2755-2721/29/20230803

indicators and analysis tools for evaluating and comparing the performance of different network configurations and strategies. It can fully meet the needs of this simulation experiment.

To compare the differences in network performance prediction under different SDN distributed controller deployment structures. We employ two different controller structures, both of which are built into the OOS system. They are based on the flat structure of HyperFlow, each controller is at the same level, and there is no need to distinguish between primary and secondary controllers, but there is a disadvantage of real-time synchronization of information; based on Kandoo's hierarchical structure, each controller is divided into primary and secondary controllers, namely, a root controller and a local controller.

This dataset file is constructed by continuously changing relevant network configuration parameters, performing network simulation on OOS, and collecting relevant network performance indicators. For each simulation, 21600 observations can be obtained, and the related configuration is as follows:

- The number of controllers in each of the 3 groups of controllers: Number of controller instances
- Solve and generate 3 different controller placement schemes. For example, minimize controller-to-root latency
- Set 12 sets of TimeOut values: Idle timeout of flow entries, ranging from 5 s to 60 s
- Four simulation experiments were performed for each configuration.
- Simulation using 50 groups of real network topologies

### *3.2. Input and output variables*

During the simulation process, we processed the collected network traffic data features and finally collected 349 input features. In addition, considering that there are many input features, we will adopt the PCA data dimensionality reduction method in the future. The input features include two categories. The first category is static topology metrics, e.g., the number of switches, betweenness, and closeness. These metrics are always the same for a network and persist through different configurations. The other category is the semidynamic controller metrics, e.g., timeout, number of controllers, and latency. These metrics may change per configuration.

The output targets include six variables, namely, the maximum and mean values for RTT, ControlPlaneTraffic, and SyncTraffic. RTT is the round-trip time of pings. ControlPlaneTraffic is the switch-to-controller (S2C) traffic caused by mismatched packets or topology discovery. SyncTraffic is the controller-to-controller (C2C) traffic due to synchronization.

## 4. Methodology

Following the analysis of the dataset, we investigate methods based on supervised learning. From less complex to more complex, gradually tune the PCA parameters and apply multiple machine learning models to arrive at the most promising algorithm. Specifically, we preprocess the input SDN network dataset, standardize the data, and then use the PCA algorithm to reduce the dimensionality of the features. Finally, different models are selected for training to select the optimal model and parameter combination to obtain the optimal prediction results. Some key parts will be described in detail below.

### *4.1. Preprocessing steps*

Standardization step: To ensure the effectiveness of machine learning algorithms, we convert data with different characteristics into data with the same scale. Using the StandardScaler method of normalization, it subtracts the value of each feature from its mean and then divides the result by its standard deviation, thus ensuring that the values of each feature have zero mean and unit variance.

PCA step: To study whether the simplified set of input features will affect the accuracy of the prediction results, reduce data complexity and noise, and improve the efficiency and accuracy of model training. We applied a data dimensionality reduction method PCA (principal component analysis) to reduce the dimensionality of the input features. The basic idea is to calculate the covariance matrix between the input features, obtain the eigenvalues and eigenvectors, and then select the most representative eigenvectors to map the original feature space to a new low-dimensional feature space.

DOI: 10.54254/2755-2721/29/20230803

In the specific implementation, we can use the PCA class in the Scikit-learn library in Python to implement the PCA algorithm.

### *4.2. Machine learning models*

In our work, we are mainly based on SDN network traffic data, expecting its related performance indicators and exploring its regression problem. We introduce the four regression algorithms used in this work in detail in the next paragraph, and some of the algorithms have the ability to perform classification tasks. Regression tasks can also be performed, and here, we only discuss regression problems.

AdaBoost: AdaBoost is a boosting algorithm for ensemble learning. It has certain noise resistance and is not prone to overfitting problems. The basic idea is to combine multiple weak regression models into a stronger model. In each iteration, the algorithm assigns higher weights to the next weaker model based on the error of the current model, giving these weaker models better predictive power when combined. Specifically, AdaBoost uses a set of basic regression models (such as decision trees, linear regression, etc.) as weak models. Then, in each iteration, it trains a weaker model again based on the adjusted sample weights. Finally, the prediction results of all weak models are weighted according to a certain ratio to obtain the final prediction result of the whole model.

Decision Tree: Decision Tree is an important predictive algorithm for machine learning modeling. This technology is fast to learn, easy to understand, and can produce more accurate predictions. It is widely used in machine learning. It uses a binary tree structure to predict continuous variables. The basic idea of the decision tree is to recursively divide the dataset into subsets, and the model will select an optimal feature to divide the dataset into two subsets. The data in each subset have similar characteristics, and this process is repeated until all the subsets are divided or the preset stop condition is reached. Estimates are made by traversing the partitions of the tree until a leaf node is reached, at which the predicted value is output. Decision tree is very interpretable and robust and can handle datasets with nonlinear relationships.

Random Forest: Random Forest is one of the most famous machine learning algorithms and is an ensemble of decision trees. It combines the prediction results of multiple decision tree models to form a more accurate and stable overall prediction result. During the construction of each tree, the model uses a bootstrap sampling method to randomly select a certain proportion of samples from the original dataset and randomly selects a certain number of features as the basis for the construction of the tree. The basic idea of the model is to learn simultaneously in multiple decision trees to avoid overfitting of a single decision tree. Usually, it is trained by the bagging method. When making predictions, random forest averages or weights the prediction results of all decision trees to obtain the final prediction result. The model has good robustness and generalization ability and can handle high-dimensional data and missing data. However, due to the complex structure of the model and the long training time, it is necessary to properly adjust the model parameters to achieve better prediction performance.

XGBoost: XGBoost is a well-known ensemble learning model based on gradient boosting trees. Like Adaboost, Gradient Boosted Regression Trees (GBRT) operates by incrementally adding classifiers to the ensemble, each classifier correcting previous classification results. However, it does not change the weight of the instance every iteration like Adaboost. This method uses the forward distribution algorithm for greedy learning and uses a new classifier to fit the residual predicted by the previous classifier. XGBoost is a series of optimizations to the GBRT algorithm, such as adding regularization items and tree pruning techniques to prevent overfitting. In each iteration, XGBoost will train a new decision tree and update the prediction results by combining the prediction results of all previous trees. In addition, XGBoost can also sample samples and features to increase the robustness of the model. After the training is completed, XGBoost sorts the features according to the contribution value of the tree to facilitate feature selection and model interpretation. XGBoost is an efficient and accurate model that can handle datasets with high dimensions, a large number of samples, and complex structures and can automatically handle missing values. XGBoost also supports parallel computing and GPU acceleration, enabling fast training of large-scale datasets.

DOI: 10.54254/2755-2721/29/20230803

## 5. Discussion

To measure the predictive performance of different machine learning algorithms, the accuracy of predictive models under different PCA parameters is compared at the same time. We run the algorithm to calculate the evaluation indicators under each model and conduct analysis, comparison and evaluation. This chapter will describe our experiment-related settings and experimental evaluation results in detail.

### *5.1. Settings*

To better train the model and prevent overfitting problems, we use 5-fold cross-validation, which randomly divides the original data into 5 groups (K-Fold) and makes each subset data a validation set, and the remaining 4 sets of subset data are used as training sets to obtain 5 models. These five models evaluate the results in the validation set, and the final error MSE (mean squared error) is summed and averaged to obtain the cross-validation error. Cross-validation makes efficient use of limited data and evaluates results as close as possible to the model's performance on the test set. The root mean square error (RMSE), mean absolute error (MAE) and mean absolute percentage error (MAPE) are used as the evaluation metrics. The implementation is based on Python and its packages, e.g., scikit-learn.

$$\text{RNSE}=\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-y_i)^2} \tag{1}$$

$$\text{MAE}=\frac{1}{N}\sum_{1}^{N}|x_i-y_i| \tag{2}$$

$$\text{MAPE}=\frac{\sum_{i=1}^{N}\left|\frac{x_i-y_i}{y_i}\right|}{N}\times 100 \tag{3}$$

### *5.2. Results*

In this part, the experimental results from different machine learning models are presented in Tables 1-6 for both the flat architecture and the hierarchical architecture. For most cases, XGBoost achieves the lowest error compared with AdaBoost, decision tree and random forest as baselines. However, the random forest outperforms XGBoost for RTT prediction in the flat architecture.

**Table 1.** MAE results for the flat architecture.

| Model | | AdaBoost | Decision Tree | Random Forest | XGBoost |
|---|---|---|---|---|---|
| RTT | Mean Value | 8.426 | 7.540 | 6.449 | 6.661 |
| | Max Value | 90.565 | 84.499 | 70.901 | 71.597 |
| S2C | Mean Value | 125.475 | 71.805 | 56.327 | 39.165 |
| | Max Value | 979.585 | 910.151 | 876.929 | 873.007 |
| C2C | Mean Value | 8.776 | 5.575 | 3.841 | 3.284 |
| | Max Value | 18.179 | 19.275 | 17.056 | 14.990 |

**Table 2.** RMSE results for the flat architecture.

| Model | | AdaBoost | Decision Tree | Random Forest | XGBoost |
|---|---|---|---|---|---|
| RTT | Mean Value | 11.943 | 13.182 | 11.417 | 11.520 |
| | Max Value | 133.309 | 145.483 | 121.719 | 123.196 |
| S2C | Mean Value | 143.448 | 109.932 | 85.134 | 61.278 |
| | Max Value | 1523.534 | 1660.565 | 1577.677 | 1941.336 |
| C2C | Mean Value | 10.365 | 8.826 | 5.972 | 5.249 |
| | Max Value | 24.587 | 27.527 | 24.829 | 22.575 |

DOI: 10.54254/2755-2721/29/20230803

**Table 3.** MAPE results for the flat architecture.

| Model | | AdaBoost | Decision Tree | Random Forest | XGBoost |
|---|---|---|---|---|---|
| RTT | Mean Value | 65.513 | 21.633 | 19.625 | 23.652 |
| | Max Value | 140.196 | 50.608 | 36.408 | 34.811 |
| S2C | Mean Value | 23.016 | 10.050 | 7.871 | 5.292 |
| | Max Value | 16.766 | 11.668 | 11.928 | 11.205 |
| C2C | Mean Value | 9.413 | 5.036 | 3.569 | 3.066 |
| | Max Value | 9.281 | 9.160 | 8.024 | 7.002 |

**Table 4.** MAE results for the hierarchical architecture.

| Model | | AdaBoost | Decision Tree | Random Forest | XGBoost |
|---|---|---|---|---|---|
| RTT | Mean Value | 11.379 | 10.138 | 8.613 | 8.336 |
| | Max Value | 114.167 | 127.089 | 114.052 | 99.003 |
| S2C | Mean Value | 125.864 | 71.263 | 55.906 | 41.932 |
| | Max Value | 981.245 | 1756.097 | 851.240 | 864.706 |
| C2C | Mean Value | 10.246 | 4.033 | 2.936 | 2.607 |
| | Max Value | 17.874 | 12.213 | 9.514 | 8.327 |

**Table 5.** RMSE results for the hierarchical architecture.

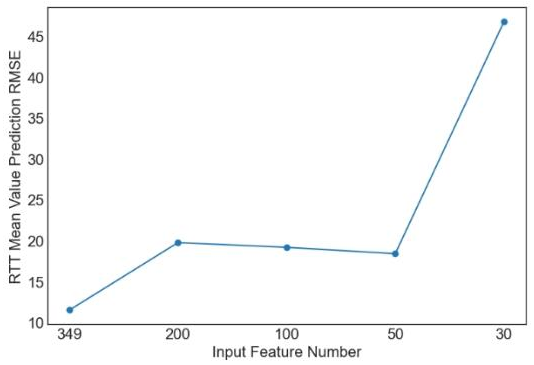
| Model | | AdaBoost | Decision Tree | Random Forest | XGBoost |
|---|---|---|---|---|---|
| RTT | Mean Value | 16.667 | 18.759 | 15.755 | 15.091 |
| | Max Value | 226.124 | 258.344 | 238.207 | 203.710 |
| S2C | Mean Value | 144.220 | 108.456 | 84.051 | 67.223 |
| | Max Value | 1532.429 | 3563.972 | 1532.113 | 1927.967 |
| C2C | Mean Value | 11.519 | 6.751 | 4.904 | 4.092 |
| | Max Value | 21.710 | 19.279 | 14.211 | 12.438 |

**Table 6.** MAPE results for the hierarchical architecture.

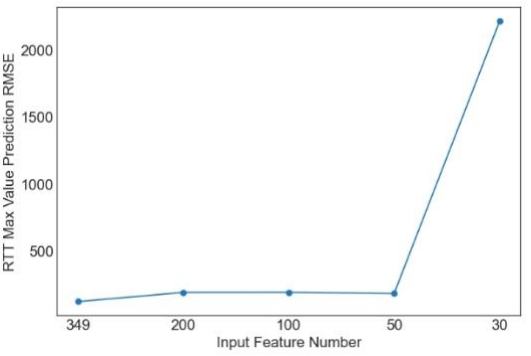
| Model | | AdaBoost | Decision Tree | Random Forest | XGBoost |
|---|---|---|---|---|---|
| RTT | Mean Value | 73.843 | 32.576 | 23.663 | 23.576 |
| | Max Value | 78.158 | 46.647 | 43.964 | 40.785 |
| S2C | Mean Value | 23.185 | 9.726 | 7.781 | 5.522 |
| | Max Value | 16.727 | 30.404 | 11.402 | 10.036 |
| C2C | Mean Value | 106.966 | 19.367 | 14.693 | 14.001 |
| | Max Value | 40.807 | 17.894 | 14.481 | 12.899 |

Then, we evaluate the influence of the number of PCA components. Since XGBoost is the best machine learning model from the above tables, only the results with XGBoost are plotted in Figures 1 and 2. Similar observations are obtained for other machine learning models. For simplicity, only RMSE is used in Figures 1 and 2. As observed from the results, feature selection through PCA fails to further improve the prediction performance of XGBoost. The only exception occurs in the case of S2C max value prediction in the flat architecture. It is also observed that the prediction performance decreases sharply when the feature number is reduced from 50 to 30. Overall, it is not a good idea to apply PCA for feature selection in the SDN performance prediction problem considered in this study. More feature selection techniques should be considered in future studies.

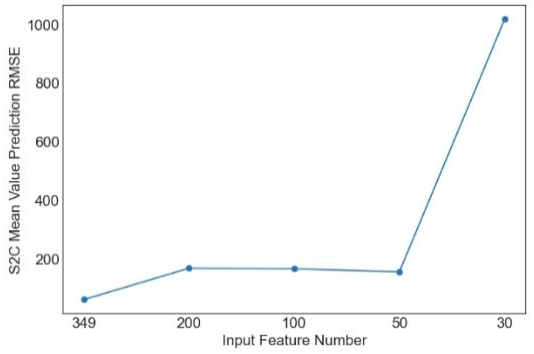
DOI: 10.54254/2755-2721/29/20230803



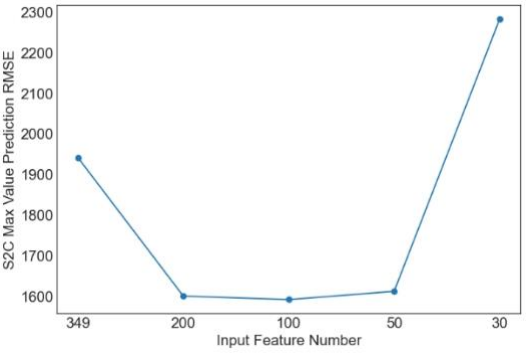

(a) RTT Mean Value Prediction RMSE



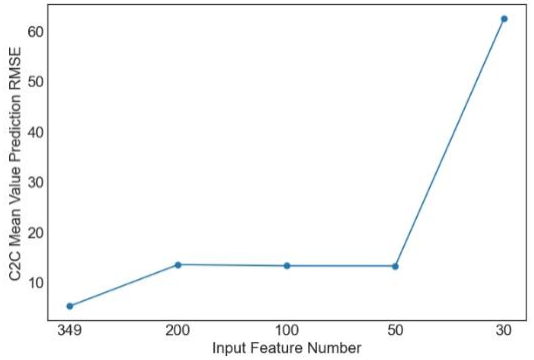

(b) RTT Max Value Prediction RMSE



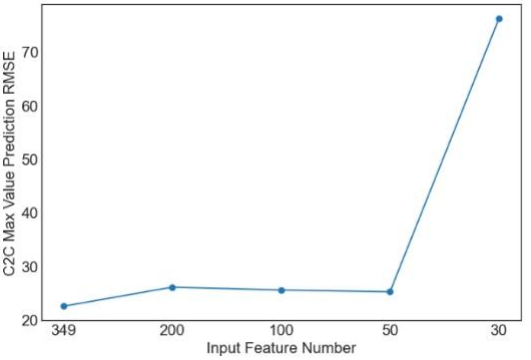

(c) S2C Mean Value Prediction RMSE




(d) S2C Max Value Prediction RMSE

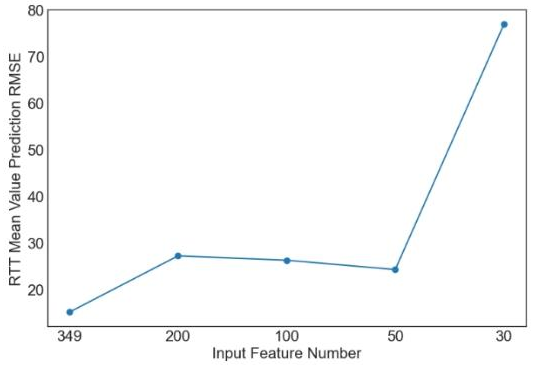



(e) C2C Mean Value Prediction RMSE

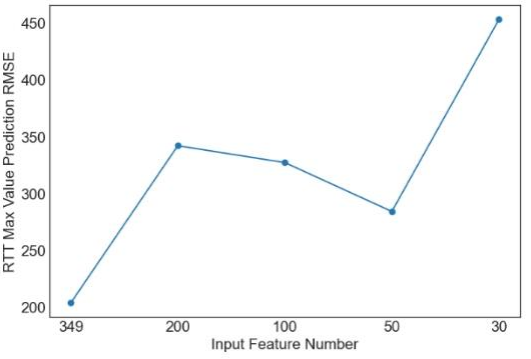



(f) C2C Max Value Prediction RMSE

**Figure 1.** Prediction results of XGBoost for the flat architecture.

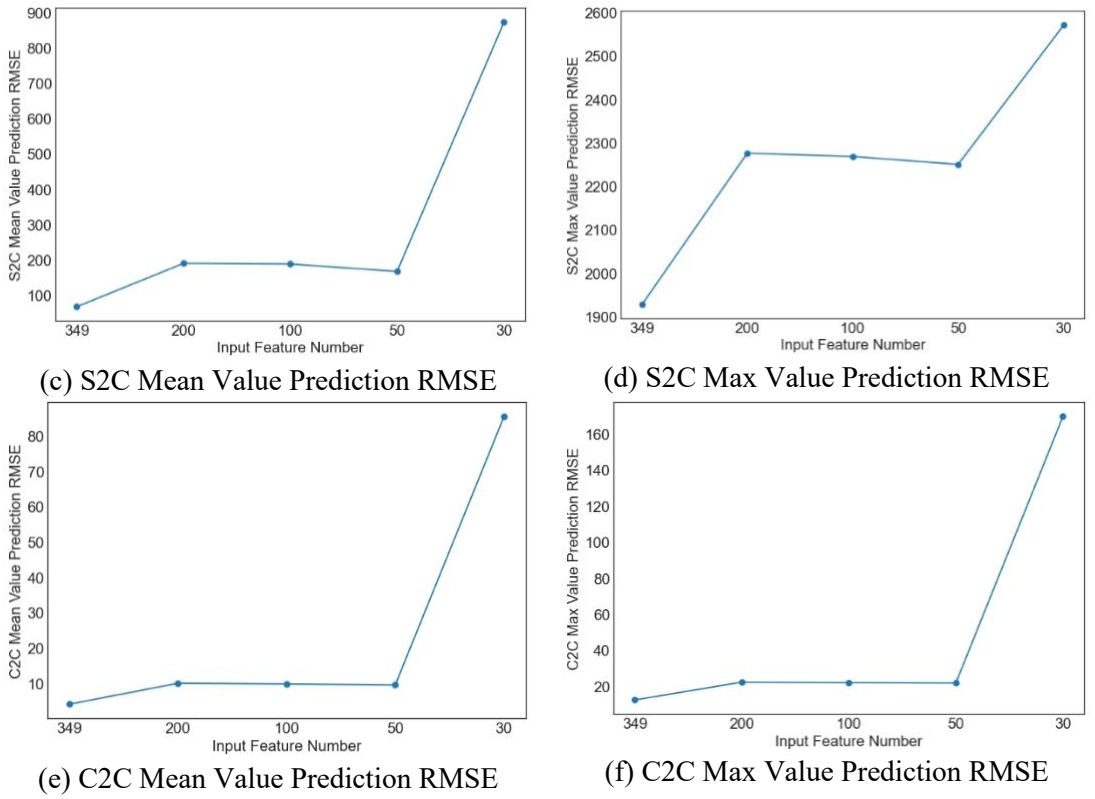



(a) RTT Mean Value Prediction RMSE




(b) RTT Max Value Prediction RMSE

DOI: 10.54254/2755-2721/29/20230803




(c) S2C Mean Value Prediction RMSE

(d) S2C Max Value Prediction RMSE

(e) C2C Mean Value Prediction RMSE

(f) C2C Max Value Prediction RMSE

**Figure 2.** Prediction results of XGBoost for the hierarchical architecture.

## 6. Conclusion

Network performance prediction is an important research direction in the context of future Internet and SDN networks. In this paper, we propose a method based on machine learning to predict performance indicators in SDN networks to achieve optimal network deployment strategies. The goal is to reduce engineering overhead and improve network transmission efficiency. The approach is to use a publicly available network dataset derived from different network traffic data collected while constantly changing network-related configurations. We tried the PCA feature selection method on this dataset, used four machine learning models such as AdaBoost for training, used the 5-fold cross-validation method, and finally calculated RMSE, MAE and MAPE to evaluate and compare their predictive capabilities. After analysis, we found that compared with AdaBoost, decision tree and random forest, the XGBoost model has the most accurate prediction ability, and the errors of the MAE, RMSE and MAPE indicators are generally lower. However, we found that PCA feature selection cannot further improve the prediction performance of the corresponding model. In future research, we can choose other feature selection methods and try to analyze their performance capabilities, such as using the forward-stepwise-selection machine learning method to explore whether it can reduce the size of the feature set without affecting the XGBoost model predictive performance to reduce model training time.

## References

[1] Long, Q., Chen, Y., Zhang, H., & Lei, X. (2019). Software defined 5G and 6G networks: a survey. Mobile networks and applications, 1-21.

[2] Jiang, W. (2023). Software defined satellite networks: A survey. Digital Communications and Networks.

[3] Turner, S. W., Karakus, M., Guler, E., & Uludag, S. (2023). A Promising Integration of SDN and Blockchain for IoT Networks: A Survey. IEEE Access.

[4] Jiang, W. (2021). Applications of deep learning in stock market prediction: recent progress.

Expert Systems with Applications, 184, 115537.
[5] Jiang, W., & Zhang, L. (2018). Geospatial data to images: A deep-learning framework for traffic forecasting. Tsinghua Science and Technology, 24(1), 52-64.
[6] Jiang, W., & Luo, J. (2022). Graph neural network for traffic forecasting: A survey. Expert Systems with Applications, 117921.
[7] Jiang, W., Luo, J., He, M., & Gu, W. (2023). Graph Neural Network for Traffic Forecasting: The Research Progress. ISPRS International Journal of Geo-Information, 12(3), 100.
[8] Ohtsuki, T. (2023). Machine Learning in 6G Wireless Communications. IEICE Transactions on Communications, 106(2), 75-83.
[9] Jiang, W. (2022). Cellular traffic prediction with machine learning: A survey. Expert Systems with Applications, 117163.
[10] Jiang, W. (2022). Graph-based deep learning for communication networks: A survey. Computer Communications, 185, 40-54.
[11] Cao, Z., Zhang, H., Liang, L., & Li, G. Y. Machine Learning for Wireless Communication: An Overview. APSIPA Transactions on Signal and Information Processing, 11(1).
[12] Kanakis, M. E., Khalili, R., & Wang, L. (2022). Machine Learning for Computer Systems and Networking: A Survey. ACM Computing Surveys, 55(4), 1-36.
[13] Jiang, W. (2022). Internet traffic matrix prediction with convolutional LSTM neural network. Internet Technology Letters, 5(2), e322.
[14] Jiang, W. (2022). Internet traffic prediction with deep neural networks. Internet Technology Letters, 5(2), e314.
[15] Jiang, W., He, M., & Gu, W. (2022). Internet Traffic Prediction with Distributed Multi-Agent Learning. Applied System Innovation, 5(6), 121.
[16] Jain, S., Khandelwal, M., Katkar, A., & Nygate, J. (2016, October). Applying big data technologies to manage QoS in an SDN. In 2016 12th International Conference on Network and Service Management (CNSM) (pp. 302-306). IEEE.
[17] Pasquini, R., & Stadler, R. (2017, July). Learning end-to-end application qos from openflow switch statistics. In 2017 IEEE Conference on Network Softwarization (NetSoft) (pp. 1-9). IEEE.
[18] Nougnanke, K. B., Labit, Y., Bruyere, M., Ferlin, S., & Aivodji, U. (2021, March). Learning-based incast performance inference in software-defined data centers. In 2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN) (pp. 118-125). IEEE.
[19] Nougnanke, B., Labit, Y., Bruyere, M., Aïvodji, U., & Ferlin, S. (2022). ML-based performance modeling in SDN-enabled data center networks. IEEE Transactions on Network and Service Management.
[20] Kheibari, B., & Sayıt, M. (2020, November). Quality estimation for DASH clients by using Deep Recurrent Neural Networks. In 2020 16th International Conference on Network and Service Management (CNSM) (pp. 1-8). IEEE.
[21] Abar, T., Letaifa, A. B., & Asmi, S. E. (2020). Quality of experience prediction model for video streaming in SDN networks. International Journal of Wireless and Mobile Computing, 18(1), 59-70.
[22] Gray, N., Dietz, K., Seufert, M., & Hossfeld, T. (2021, June). High performance network metadata extraction using P4 for ML-based intrusion detection systems. In 2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR) (pp. 1-7). IEEE.
[23] Hardegen, C., & Rieger, S. (2020, November). Prediction-based flow routing in programmable networks with P4. In 2020 16th International Conference on Network and Service Management (CNSM) (pp. 1-5). IEEE.
[24] Goścień, R., Knapińska, A., & Włodarczyk, A. (2021). Modeling and prediction of daily traffic patterns—WASK and SIX case study. Electronics, 10(14), 1637.
[25] Filali, A., Mlika, Z., Cherkaoui, S., & Kobbane, A. (2020). Preemptive SDN load balancing with

DOI: 10.54254/2755-2721/29/20230803

machine learning for delay sensitive applications. IEEE Transactions on Vehicular Technology, 69(12), 15947-15963.

[26] Dietz, K., Gray, N., Seufert, M., & Hossfeld, T. (2022, April). ML-based performance prediction of SDN using simulated data from real and synthetic networks. In NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium (pp. 1-7). IEEE.