ViolationPredictor: a Solution for Predicting SLA Violations of IoT Applications

Noureddine Staifi[1] and Meriem Belguidoum[1]

[1]LIRE Laboratory, University of Constantine 2, Algeria

Abstract

The Internet of Things (IoT) paradigm has emerged strongly over the past decade and has established itself as an important player in the provision of services that are increasingly adapted to user preferences and profiles. Indeed, the management of the quality of service (QoS) is essential at the level of IoT systems, particularly critical applications, such as smart home systems (Smart Home Systems - SHS) and health monitoring systems (Health Monitoring Systems - HMS), requiring a certain level of quality specified and guaranteed by formal contracts, called Service Level Agreements (SLA). However, managing these SLAs are crucial tasks, such as SLA negotiation, monitoring, control, breach prediction, customisable management, etc. This paper presents ViolationPredictor, a Deep Learning (DL) based solution for the prediction of SLA violations. ViolationPredictor provides a way to predict future SLA violations and uses neural networks to accomplish this task. For each obligation, ViolationPredictor generates a neural network, where each system can predict possible future violations of this obligation. We used recurrent neural networks to implement ViolationPredictor because they have a memory that captures processed information and they can retain and consider past contextual information in their decisions.

Keywords

Internet of Things, Service Level Agreements, Quality of Service, SLA violation, Violation prediction, Deep Learning, Neural network.

1. Introduction

The Internet of Things (IoT) concept represents the new era of the Internet, allowing to interconnect objects to provide intelligent services. Currently, there are approximately 12.3 billion connected objects in the world, and by 2025, connected objects will generate more than 73.1 billion Terabytes of data [1]. According to Cisco research, by 2030, 500 billion devices are expected to be connected to the IoT [2]. However, to maximize the benefits of IoT in general, and Smart Home Systems (SHS) in particular, several challenges need to be overcome, namely managing massive amounts of data, privacy, security, and Quality of Service (QoS) management.

In SHS, each application has its usage and traffic characteristics, and therefore requires a certain level of quality specified in QoS contracts, called Service Level Agreements (SLA). An SLA is a formal contract between service providers and consumers, it specifies the provided services, the obligations of each party and the corresponding penalties in the event of a contract violation. Its main objective is to clarify the needs of the customer and the provider, allowing

Tunisian Algerian Conference on Applied Computing (TACC 2022), December 13–14, 2022, Constantine, Algeria

noureddine.staifi@univ-constantine2.dz (N. Staifi); meriem.belguidoum@univ-constantine2.dz (M. Belguidoum)

0000-0001-9965-9785 (N. Staifi); 0000-0002-2936-6810 (M. Belguidoum)

each party to respect its commitment, and in case of conflict, it improves the understanding aspect between these parties [3].

SLAs play a key role in the deployment of services, where their specification and management have become increasingly complex at the level of IoT applications. SLA specification is essential to explicitly describe a prescribed service between a service provider and a consumer in terms of required QoS, quantified and measurable expectations, consumer priorities, etc. [4]. Indeed, this specification is a kind of guarantee and assurance for the consumer. On another side, SLA management plays a vital role in establishing and maintaining a stable, reliable and measurable business relationship between service provider and consumer, which presents several challenges such as negotiation, monitoring, control, violation prediction, customisable management, etc. [5].

SLA violation is related to the assessment of the service QoS compliance with an SLA, it concerns various relevant issues, such as reliability, availability, security and performance. Traditional methods of managing and monitoring SLA violations work well at the level of business services, such as cloud services. However, these methods cannot provide the desired levels of security and reliability for critical systems, such as IoT applications. Indeed, these applications require the consideration of some critical aspects, such as ubiquity, interconnectivity, large-scale deployment, synchronization, massive data transfer, distribution and heterogeneity. Moreover, the source of violation cannot be easily identified in the presence of multiple actors such as consumers, Cloud and IoT providers, etc.

Avoiding SLA violations requires early detection of potential risks. To reduce these situations, service providers need tools to intuitively analyse whether their service design is causing SLA violations, and to automatically guide them in their prevention. Several prediction strategies have been developed, such as those that adopt Artificial Intelligence (AI) techniques, namely Machine Learning (ML) and Deep Learning (DL), in which if the parameters approach agreed limits, monitors should be alerted to take the necessary preventive measures.

Promising approaches to service assurance and prediction of SLA violations are based on new information and communication techniques, which have facilitated the task of predicting SLAs [6]. Indeed, AI and its different techniques, such as ML and DL, appear as effective solutions to face the challenges of violation prediction [7], where service quality and behavior are learned from system observations, whose objective is to automate predictions in real time and in a proactive manner. These techniques provide predictive models that exploit the data provided to better anticipate breaches and contribute to operational efficiency.

However, in this paper, we proposed ViolationPredictor, a Deep Learning (DL) based solution for SLA violation prediction. ViolationPredictor generates a neural network for each obligation, where each system is responsible to predict future violations of this obligation or SLO. We implemented ViolationPredictor using recurrent neural networks (RNN) because they have a memory that records processed information and can store and incorporate previous contextual information into their choices. The dataset used by ViolationPredictor is a CSV file composed of two columns: (1) the series of contextual data provided by the environmental sensors, and (2) the decisions of the violations of these series.

This paper is an extension of our previous work concerning the SLA specification and management throughout their entire life cycle. The first phase was the proposal of ML-SLA-IoT

[8], a language for specifying multi-level SLAs for IoT applications. While the second phase concerns the proposal of the solution SC-Generator [9], which presents a solution for monitoring SLA obligations, that provides a way to monitor SLA terms by automatically generating Smart Contracts from specified SLOs. These smart contracts are responsible for monitoring SLO parameters, detecting violations, and notifying service providers. However, the present paper enriches our work by detecting violations before their occurrence. SC-Generator plays a key role in the ViolationPredictor solution, because it is responsible for providing the violation decisions of the contextual datasets used in the dataset.

The paper is organized as follows: Section 2 presents a review of related work including their limitations. Section 3 presents our proposed approach. This approach is illustrated by an evaluation and comparison in Section 4. Finally, in the last section, we conclude and present some future work.

2. Related Work

This section discusses AI-based solutions for predicting SLA violations. There are several proposals, we have limited ourselves to the most relevant researchs, such as Leitner et al. [7], Hani et al. [10], Wong et al. [11], Hemmat et al. [12], Uriarte et al. [13], Biswas et al. [14], Tang et al. [15] and Di et al. [16]

Leitner et al. [7] provide a model for predicting SLA violations during runtime. In this research, the model inputs could be the composition of the services or the quality of the services used. A machine learning regression technique is then used to train data captured from historical process instances.

Hani et al. [10] propose a model that predicts SLA violations using SVM-based time series analysis for regression. The prediction will learn from historical service level delivery data captured by the monitoring system. This type of data forms sequential data points in spacetime, called time series data. However, the limitation of this predictive model is its inability to scale to inherently very large and volatile real-world data.

Wong et al. [11] used five different machine learning algorithms such as SVM, Random Forests, Naive Bayesian Classifier, Neural Network, and k-NN to predict SLA violations, so corrective action can be taken. While other approaches can help a provider anticipate SLA violations, they cannot help providers quantitatively assess QoS.

Hemat et al. [12] conducted an experiment to overcome the challenge of predicting SLA violations. According to these researchers, SLA violation is a rare real-world event that only occurs 20 % of the time.

Uriarte et al. [13] use an unsupervised formulation of the Random Forest algorithm to calculate similarities and provide them as input to a Clustering algorithm, with the aim of aggregating resource usage and service duration to avoid violations of the Google Cluster Tracking data set.

Biswas et al. [14] proposed an approach that anticipates future resource demand to meet SLA requirements. They used enterprise-level SLAs (throughput and response time) as input parameters for the chosen prediction approaches. ML techniques such as SVM and linear regression were used.

Table 1
Comparison of AI-based solutions

Work	Domain	MLA	DLA	DS	PM	PN	RO
Leitner et al. [7]	services IT	linear regression	✗	simulation	✗	✗	✗
Hani et al. [10]	Cloud	SVM	✗	simulation	✓	✓	✗
Tang et al. [15]	Cloud	naive bayes classifier	✗	simulation	✓	✗	✗
Di et al. [16]	Cloud	naive bayes classifier	✗	Google data center	✗	✗	✗
Hemmat et al. [12]	Cloud	random forests	✗	Google Cloud Cluster	✗	✗	✗
Biswas et al. [14]	IoT	SVM and linear regression	✗	simulation	✗	✗	✓
Wong et al. [11]	Cloud	SVM, random forests, naive bayes classifier and k-NN	neural network	WS-DREAM	✗	✓	✗
Uriarte et al. [13]	Cloud	random forests	✗	Google Cluster Tracking	✓	✗	✓

Note : *MLA = Machine Learning Algorithms, DLA = Deep Learning Algorithms, DS = Data Source, PM = Parameter Monitoring, PN = Provider Notification, RO = Resource Optimisation.*

Several researchers use the Naive Bayes classifier. Tang et al. [15] provided an SLA violation prediction model, the training dataset is obtained from the WS-DREAM dataset, and only the response time is used as the value of hall. In the same context, Di et al. [16] proposed another Bayesian model for predicting host load using one-month tracking data collected by Google from thousands of machines running for up to 4 p.m. The predictive model uses CPU and memory as input metrics.

As shown in table 1, these proposals will be compared according to the following criteria:

- **Domain:** specifies the domain in which the solution is offered.
- **Machine Learning Algorithms (MLA):** This criterion indicates whether the solution has adopted ML techniques.
- **Deep Learning Algorithms (DLA):** indicates whether the proposal considered DL.
- **Data Source (DS):** indicates the source from where the training and test data are collected.
- **Parameter Monitoring (PM):** shows whether the solution monitors QoS parameters.
- **Provider Notification (PN):** designates whether the proposal notifies the service provider if a violation is predicted.
- **Resource Optimisation (RO):** indicates whether the solution optimises system resources to avoid possible violations.

3. ViolationPredictor: a solution for predicting SLA violations

Different ML and DL techniques have been used to create predictive models for QoS assurance. Unlike previous work on predicting SLA violations, these models are trained on real dataset to provide effective solutions. The key idea is to use data samples to train a statistical model, which is then used for unseen data predictions.

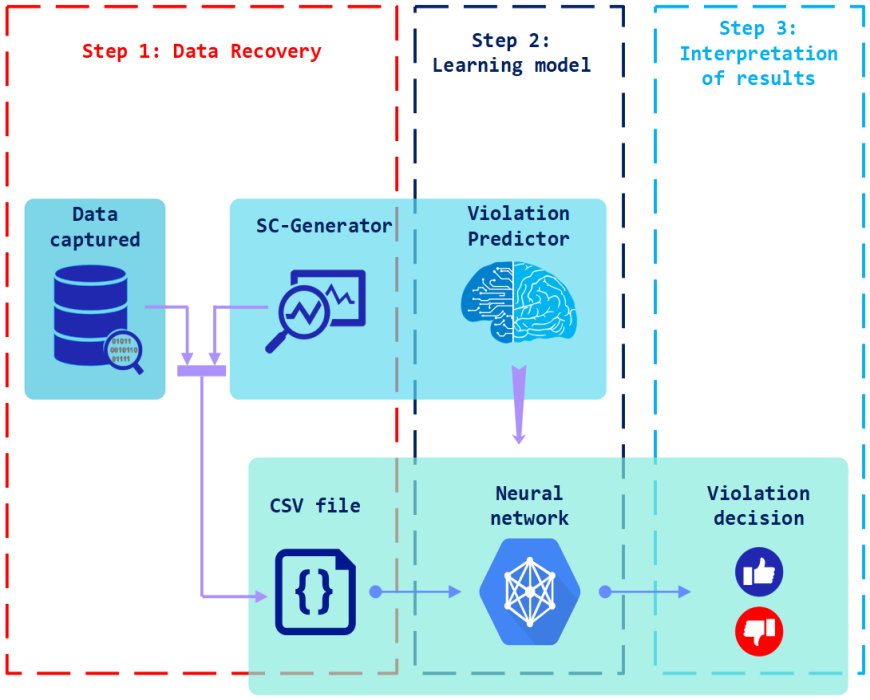

Figure 1: Overview of the ViolationPredictor

From the DL perspective, the SLA violation prediction problem is equivalent to a binary classification problem, where there are two classes: class zero is the case of non-violated tasks (violation = 0), while class one is the case of violated tasks (violation = 1).

To do this, we proposed ViolationPredictor, a DL-based solution for predicting SLA violations. It provides a means to predict future violations of SLA terms using neural networks. For each obligation, ViolationPredictor generates a neural network, where each of these systems predicts possible future violations of this obligation. Each generated neural network has as input a CSV file, this file is composed of the captured data sequences and the decisions of the corresponding violations, where these decisions are generated and provided by SC-Generator. Subsequently, the neural network performs its prediction tasks to provide the decision on future predictions. Figure 1 describes the ViolationPredictor overview.

ViolationPredictor has three main phases; Firstly, the dataset retrieval step, which is applied to retrieve the data that serve as inputs for the neural networks, this data is assembled into a CSV file. The second phase is the learning stage, which incorporates a neural network model to predict future violations. Finally, the result interpretation step consists of extracting and visualizing the network outputs to prevent future SLA violations.

3.1. Data recovery

This phase is responsible for creating neural network inputs. For this, the captured data and their violation decisions are assembled in a CSV file, where these decisions are calculated and provided by SC-Generator. CSV files serve as inputs for neural networks. This is summarized in the following steps:

- **Retrieve captured data**: Data from sensors are retrieved and divided into a set of sequences.
- **Violation decision**: for each data sequence, a violation decision is taken according to the concerned obligation parameters (the SLO threshold for example). This is achieved through the SC-Generator component which compares each data to the SLO parameters.
- **Creation of CSV files**: through the following phases:
  - Create the CSV file by the instruction: *new FileWriter ("dataset.csv");*.
  - Generate the file header by the instruction: *buffer.write ("Sequence, Violation");*.
  - Fill the file with sequences and corresponding decisions.

Figure 2 illustrates a part of a resulting CSV file.

	A	B
1	Sequence	Violation
2	[5,3,4,6,3,9,3,4,2,0,.....]	1
3	[9,3,4,2,0,5,3,4,6,3,.....]	0
4	[4,2,0,5,3,4,6,3,9,7,.....]	0
5	[6,3,9,3,5,3,4,6,3,4,.....]	1
6	[0,5,3,4,6,3,9,3,4,2,.....]	1
7	[5,3,4,6,3,9,3,4,2,8,.....]	1
8	[4,1,4,6,3,9,3,4,2,6,.....]	0

Figure 2: Example of a CSV file

3.2. Learning model

To implement ViolationPredictor, we chose recurrent neural networks (RNNs). The idea behind the choice of RNNs is, on the one hand, the use and processing of sequential data, and, on the other hand, RNNs are networks enclosing loops allowing information to persist. RNNs perform the same task for each element of a sequence, and the output depends on previous computations, in addition, they have a memory that captures the processed information, and they can retain and consider old contextual information in their future decisions. In particular, we used LSTM RNNs which overcome the difficulties encountered with standard RNNs.

For each SLA obligation, a neural network is generated. Figure 3 illustrates the architecture of each neural network which is composed of: an input layer, two LSTM layers, and an output layer. Each network has as input a CSV file which is composed of captured data and violation decisions provided by SC-Generator.

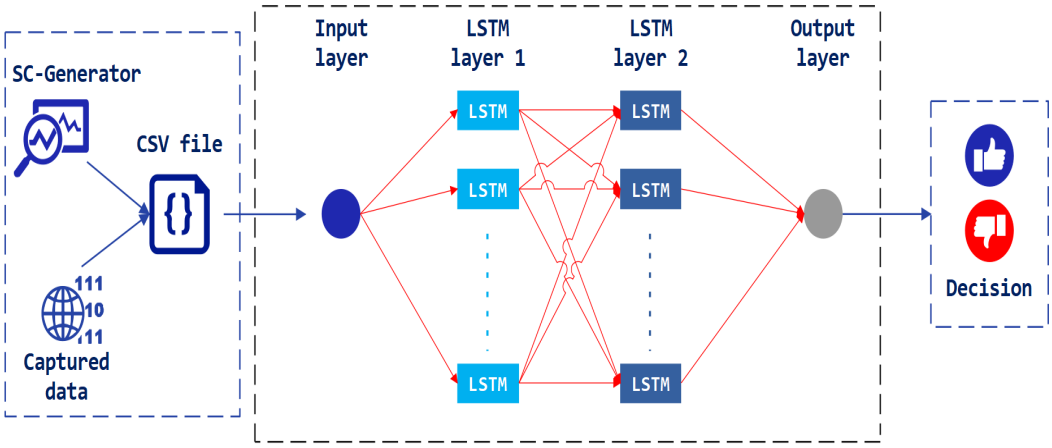

Figure 3: ViolationPredictor Neural Network Architecture

To implement these neural networks, we used the Python language with its various libraries, such as *io* (retrieving data), *Numpy* (manipulating matrices or multidimensional arrays), *Pandas* (manipulate and analyze data), *Seaborn* and *matplotlib* (visualize data). Listing 1 presents the code to create each neural network. We have created a neural network composed of two LSTM layers, where each is composed of 10 neurons with a tensor of size (1,1), and an output layer, which is a *Dense* layer with the activation function *Sigmoid*, we used this function because it returns a value between 0 and 1, which represents the probability of violation occurrence.

```
model = Sequential()
model.add(LSTM(10, input_shape=(1, 1),return_sequences=True))
model.add(LSTM(10))
model.add(Dense(1, activation='sigmoid'))
```

Listing 1: Creation of neural network

3.3. Results interpretation

ViolationPredictor predicts future SLA violations. After training and running the neural network, the results are provided. Network training is performed using the `model.fit` which takes as parameters the dataset, the number of iterations, and the batch size (see Listing 2).

```
model.fit(X_train, y_train, epochs=35, batch_size=1)
```

Listing 2: Neural network training

The precision metrics used are:

- **Loss:** measures the error of the model, i.e. how correct the model is [17]. If the loss equals 0, then the network performance is efficient. It is calculated by the following formula:

$$Loss = \frac{\sum_{i=1}^{N}(y - y\prime)^2}{N}$$

(Where N = the set of values, y = the expected output, y′ = the produced output).

- **Accuracy:** describes the performance of the model in all classes. It is useful when all classes are of equal importance [17]. The value of the precision must be equal to 1 to judge the proper functioning of the neural network. It is calculated as the ratio between the number of correct predictions and the total number of predictions through the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

(where TP = true positives, TN = true negatives, FP = false positives and FN = false negatives).

- **The Mean Squared Error - (MSE):** measures the mean squared error, i.e. the mean squared difference between the estimated values and the value true [17]. To judge the proper functioning of the neural network, the value of the MSE must be equal to 0. It is calculated by the following formula:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - F(x_i))^2$$

(Where N = the set of values, yi = the expected output, F(xi) = the produced output).

To test the efficiency of the system and to have prediction results, we used the instruction `model.predict`, which takes as parameters a set of test data, as presented in Listing 3.

```
print(model.predict(test_dataset))
```

Listing 3: Retrieving prediction results

4. Evaluation and Comparison

This phase describes the process of predicting possible violations of SLA obligations using ViolationPredictor. As presented previously, this process involves three steps: retrieving all the data, generating the learning model and interpreting the results. These steps are summarized in Figure 4. The first step is to create the CSV file that includes the data sequences and their violation decisions from SC-Generator. In the second step, the neural networks will be generated. Finally, the prediction process begins, and the third step illustrates the result of an example prediction.

The precision metrics considered are loss, precision and MSE. As illustrated in figure 5, to judge the proper functioning of the neural network, the values of the loss and the quadratic error are equal to 0, and the value of the precision is equal to 1.

As mentioned previously, the SLA management proposals have certain limitations, such as (1) uncertain monitoring of the SLAs which determines whether the obligations are met, (2) the lack of optimization of the resources used in the realization of the system, and (3) the lack of combination of AI techniques for violation prediction. To do this, we have proposed ViolationPredictor, it is a DL-based policy for predicting SLA violations. ViolationPredictor

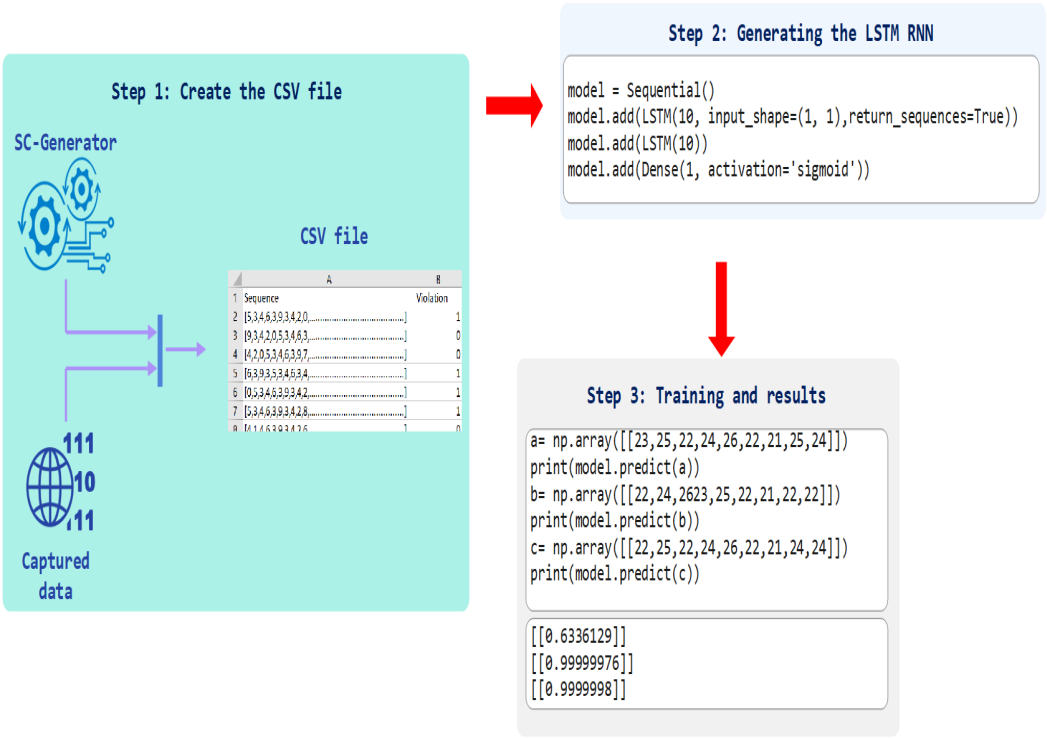

Figure 4: Generation steps and prediction results

generates, for each obligation, a neural network, where each network can predict possible future violations of this obligation.

Figure 6 presents a comparison of our ViolationPredictor solution, with some solutions related to our proposal. These solutions will be compared according to the criteria described in Table 2. As shown in figure 6, no solution uses DL's algorithms, and only our solution and the solution of Wong et al. [11] help predict violations.

5. Conclusion

This paper aims to address the challenges of QoS management for IoT applications in general, and SHS in particular. Thus, several objectives are defined beforehand allowing the flexible and intelligent management of QoS in IoT applications. This paper presented ViolationPredictor, a DL-based solution for SLA violation prediction. It provides a means of predicting future SLA violations, based on neural networks. ViolationPredictor generates a neural network for each obligation (SLO and Rule), where each network predicts possible future violations of this obligation.

In the context of our contributions, we identify several avenues that deserve to be explored to complete and extend our work. Indeed, we can cite four main possible prospects, in the short,

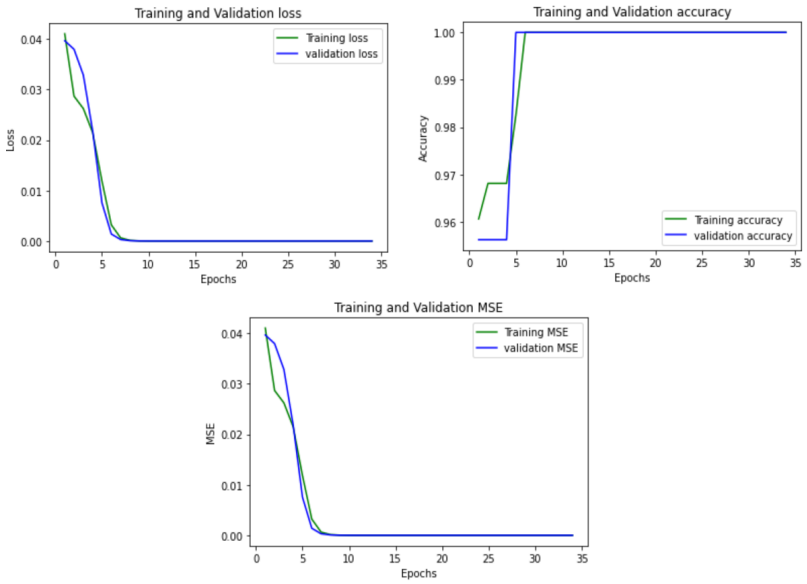

Figure 5: The considered precision metrics

Table 2
Description of the criteria for comparing SLA management work

Criteria	Description of levels		
	Criterion not supported	Criterion supported	
		Partially	Completely
Parameter monitoring	no monitoring of QoS parameters	monitoring of some QoS parameters	monitoring of some QoS parameters
Consumer compensation	no consumer compensation for SLA violations	consumer compensation in the event of SLA violations	
Notification of provider	no provider notification	notification of some violations	notification of all violations
Violation prediction	no violation prediction	prediction of some violations	prediction of all violations
Deep Learning Algorithms	Deep Learning algorithms are not considered	use of Deep Learning algorithms for prediction	

medium and long term. In the short term, we aim to extend the ML-SLA-IoT Framework by the security and accessibility aspect. In addition, we plan to integrate the technique of Chatbots to assist and help residents in different contextual situations. In the medium term, we want to manage the renegotiation of the contract dynamically at the time of execution. In the long term, we plan to aggregate SLAs and services provided by a multitude of suppliers. Finally, it would be interesting to propose a recommendation system for the provision of services adapted and customizable to user profiles.

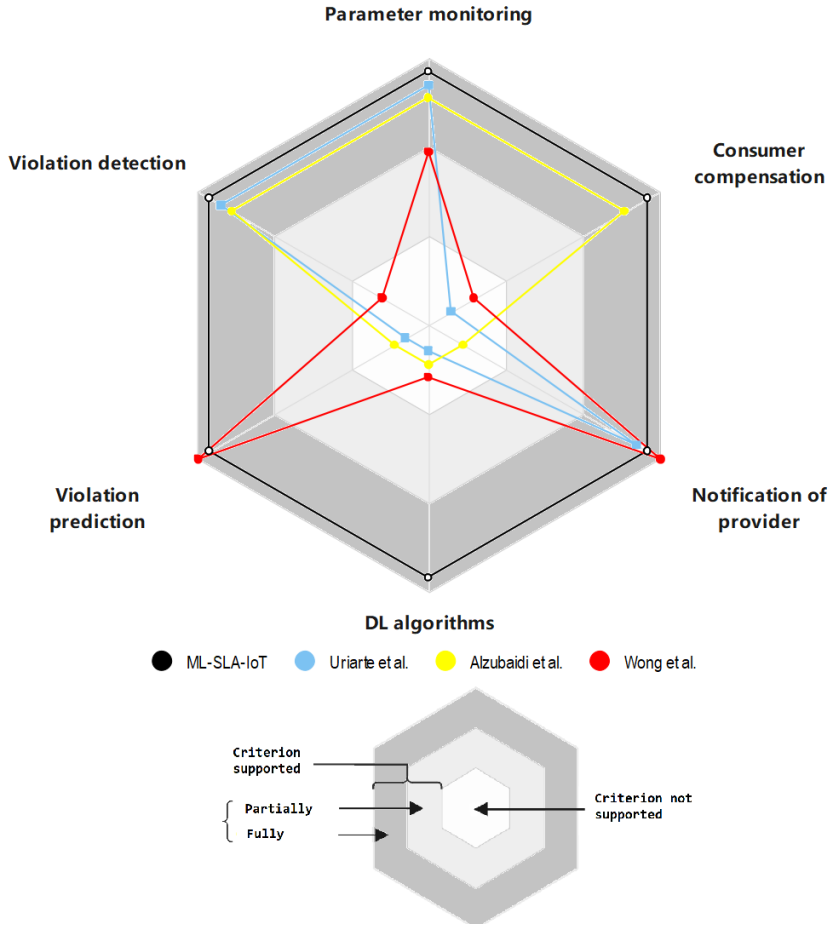

Figure 6: Comparison of ViolationPredictor

References

[1] B. Jovanović, I. Vojinovic, D. J. Spajić, N. Cvetićanin, Fascinating IoT Statistics for 2021: The State of the Industry, Retrieved April (2021).

[2] O. Benedito, R. Delgado-Gonzalo, V. Schiavoni, KeVlar-Tz: A Secure Cache for Arm TrustZone, in: IFIP International Conference on Distributed Applications and Interoperable Systems, Springer, 2021, pp. 109–124.

[3] G. Gaillard, Opérer les réseaux de l'Internet des Objets à l'aide de contrats de qualité de service (Service Level Agreements), Ph.D. thesis, INSA Lyon, 2016.

[4] M. Alhamad, T. Dillon, E. Chang, Conceptual SLA framework for cloud computing, in: 2010 4th IEEE International Conference on Digital Ecosystems and Technologies, IEEE, 2010, pp. 606–610.

[5] A. V. Dastjerdi, R. Buyya, An autonomous time-dependent SLA negotiation strategy for cloud computing, The Computer Journal 58 (2015) 3202–3216.

[6] Y. Kouki, Approche dirigée par les contrats de niveaux de service pour la gestion de l'élasticité du nuage, Ph.D. thesis, Nantes, Ecole des Mines, 2013.
[7] P. Leitner, B. Wetzstein, F. Rosenberg, A. Michlmayr, S. Dustdar, F. Leymann, Runtime prediction of service level agreement violations for composite services, in: Service-oriented computing. ICSOC/ServiceWave 2009 workshops, Springer, 2009, pp. 176–186.
[8] N. Staifi, M. Belguidoum, Multi-level sla specification language for iot applications., in: TACC, 2021, pp. 49–61.
[9] S. Noureddine, B. Meriem, Ml-sla-iot: an sla specification and monitoring framework for iot applications, in: 2021 International Conference on Information Systems and Advanced Technologies (ICISAT), IEEE, 2021, pp. 1–12.
[10] A. F. M. Hani, I. V. Paputungan, M. F. Hassan, Support vector regression for service level agreement violation prediction, in: 2013 International Conference on Computer, Control, Informatics and Its Applications (IC3INA), IEEE, 2013, pp. 307–311.
[11] T.-S. Wong, G.-Y. Chan, F.-F. Chua, A machine learning model for detection and prediction of cloud quality of service violation, in: International Conference on Computational Science and Its Applications, Springer, 2018, pp. 498–513.
[12] R. A. Hemmat, A. Hafid, SLA violation prediction in cloud computing: A machine learning perspective, arXiv preprint arXiv:1611.10338 (2016).
[13] R. B. Uriarte, S. Tsaftaris, F. Tiezzi, Service clustering for autonomic clouds using random forest, in: 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE, 2015, pp. 515–524.
[14] N. K. Biswas, S. Banerjee, U. Biswas, U. Ghosh, An approach towards development of new linear regression prediction model for reduced energy consumption and SLA violation in the domain of green cloud computing, Sustainable Energy Technologies and Assessments 45 (2021) 101087.
[15] B. Tang, M. Tang, Bayesian model-based prediction of service level agreement violations for cloud services, in: 2014 Theoretical Aspects of Software Engineering Conference, IEEE, 2014, pp. 170–176.
[16] S. Di, D. Kondo, W. Cirne, Host load prediction in a Google compute cloud with a Bayesian model, in: SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, IEEE, 2012, pp. 1–11.
[17] K. Janocha, W. M. Czarnecki, On loss functions for deep neural networks in classification, arXiv preprint arXiv:1702.05659 (2017).