A Predictive System for IoTs Reconfiguration Based on TensorFlow Framework

Tuan Nguyen-Anh[1,2(✉)] and Quan Le-Trung[1]

[1] Faculty of Computer Networks and Communications, University of Information Technology, Vietnam National University, Ho Chi Minh City, Vietnam
natuan@vku.udn.vn, quanlt@uit.edu.vn

[2] Vietnam - Korea University of Information Technology and Communications, The University of Danang, Danang City, Vietnam

Abstract. IoTs are rapidly growing with the addition of new sensors and devices to existing IoTs. The demand of IoT nodes keeps increasing to adapt to changing environment conditions and application requirements, the need for reconfiguring these already existing IoTs is rapidly increasing. It is also important to manage the intelligent context to execute when it will trigger the appropriate behavior. Yet, many algorithms based on different models for time-series sensor data prediction can be used for this purpose. However, each algorithm has its own advantages and disadvantages, resulting in different reconfiguration behavior predictions for each specific IoTs application. Developing an IoTs reconfiguration application has difficulty implementing many different data prediction algorithms for different sensor measurements to find the most suitable algorithm. In this paper, we propose IoTs Reconfiguration Prediction System (IRPS), a tool that helps IoT developers to choose the most suitable time-series sensor data prediction algorithms for trigger IoTs reconfiguration actions.

Keywords: IoTs · Reconfiguration · Intelligent context management · IoTs prediction system

1 Introduction

In the broad overview of the Internet of Things, reconfiguring and reprogramming applications deployed on IoTs devices is one of the most important and urgent content. Because IoT devices can be deployed in places where people cannot access to work such as rugged terrain (such as deep forests, volcanoes …) or operating in contexts where information is unpredictable. A reconfiguration of an IoTs network includes alterations to the devices connected, changing the behavioral patterns of the devices and modifying the software modules that control the IoTs network and devices. Reconfiguring an already existing IoTs network is a challenge due to the amount of data loss when carrying out a reconfiguration procedure in a limited power supply environment and reconfiguration time is often slow to respond to real-time IoTs applications. Many technical solutions using artificial intelligence have been proposed to solve the problem

N.-S. Vo and V.-P. Hoang (Eds.): INISCOM 2020, LNICST 334, pp. 206–218, 2020.
https://doi.org/10.1007/978-3-030-63083-6_16

of unreliable sensor data collection and how to shorten the time to reconfigure IoTs applications.

Intelligent context management is influential in the process of triggering the reconfiguration of IoTs applications. This will help the system know when to reconfigure the IoTs application and how that behavior will be. For the problem of reconfiguring IoTs applications when sensor data is unreliable and improving reconfiguration time, many artificial intelligence techniques have been proposed to build this component, bring the problem back to solve the classification and prediction problems.

To address the challenges posed, we have proposed R-IoT, RFL-IoT and RoB-IoT frameworks [1, 5, 6], for the reconfiguration of IoTs applications based on the changes of the context and the intelligent context management. The results show that our proposal framework is suitable for remote reconfiguration of IoTs applications with intelligent context management components for triggering reconfiguration. However, the preceding works did neither apply for complex contexts and unreliable data. For example, during the working time, if someone is in the room and the room temperature is higher than 30 °C, the air conditioner will be turned on and the cool mode will be changed. Actually the time-series sensor data are collected in IoTs environment often do not accurately reflect the context information, which is unstable and unreliable. Therefore, when we meet a situation like this, not sure whether context data is true or not, we cannot rely on it to make a decision. This is one of the reasons why the decision to reconfigure is incorrect. To address this issue, many suggestions on the use of machine learning algorithms approach. Mehdi Mohammadi [8] reviewed the characteristics of IoTs data and its challenges for machine learning (ML) and deep learning (DL) methods. We recognize that the next work needs to apply more machine learning algorithms to the intelligent context management component in the reconfiguration framework. It can be seen that the implementation of many IoTs projects with different data collection and characteristics requires a different technique and algorithm to help the smart context management component make the best predictions. Many research groups have developed algorithms for time-series sensor data prediction, and evaluated them on different dataset. However, there is no golden standard for choosing the best algorithm, since it is an application specific task. Finding techniques and algorithms that fit the criteria {accuracy of decision making, reconfiguration time} also takes a lot of time for developers of IoTs applications to test and build algorithm models, individual elements of that project and experiment. It is necessary to propose a tool to select the appropriate AI technique for any problem to reduce development costs of reconfiguration projects.

This research aims to help developers of IoTs solution to choose the best algorithm for time-series sensor data prediction regarding their application data. For this proposal, we developed IoTs Reconfiguration Prediction System (IRPS), which is also a component of Framework R-IoT, a web-based online tool that implements four different algorithms for time-series sensor data prediction. Users can upload historic sensor measurement from their application, and IRPS can analyze which time-series sensor data prediction algorithm fits best regarding two evaluation metrics: prediction accuracy and time reconfiguration. This system uses Google open-source TensorFlow framework to build the model of gesture recognition, introduces the platform characteristics of TensorFlow, and puts one deep learning algorithms (FFNN), two algorithms are unsupervised

learning (RF, NB) algorithms based on TensorFlow framework. Teachable Machine[1] is a browser application that you can train with your webcam to recognize objects or expressions. All training is done in the browser using the deeplearn.js library. TensorFlow Serving[2] is another tool was proposed by many developers. This is an open source toolkit, used to deploy models TensorFlow which trained on production environment. With TensorFlow Serving, the process of deploying the model to the system will become much faster and easier with normal model deployment and loading. TensorFlow Serving deploys the model independently, separates from the backend code, supports two protocols: gRPC and RESTful API, easily add and update new models according to each version without causing images to affect other parts of the system or other services. When the problem is to help the developers of the IoTs reconfiguration solution make the choice of algorithms to train the model, those tools is not appropriate for IoTs reconfiguration. Teachable machine tool mainly focuses on training and testing support for image processing problems and it does not support for IoTs reconfiguration. The TensorFlow Serving tool is great, but it focuses on deploying primarily after the model is available, which is a disadvantage. To the best of our knowledge, IRPS is the first tool designed for the IoTs reconfiguration developers to help them create efficient IoTs reconfiguration applications. This is also a module in the intelligent context management component of the reconfigured R-IoT framework. This paper is organized as follows. Time-series sensor data algorithms are presented in Sect. 2. Section 3 describes the development of the IRPS. Two case studies are presented in Sect. 4. Finally, the conclusion is given in Sect. 5.

2 Related Work

2.1 IoTs Reconfiguration Framework

In reconfiguration mechanisms, the reprogramming is handled at the physical device directly used as much as the researches results [9, 10]. IoTs applications will be able to change their behavior by sending parameters or uploading new binary firmware to change application behavior. However, this approach is thought to be a manual method, and this incurs costs for deployment and maintenance. According to [3], Nikolov, N. proposed a reconfiguration approach by Firmware Update Over The Air (FOTA) for ESP8266 device. Its procedure will be executed after new firmware bin files are uploaded to cloud. When a reconfiguration is triggered by the user, the ESP8266 device requires the FOTA control system on Cloud to return the corresponding firmware. However, this solution has some shortcomings in terms of security protocols, reconfiguration time, especially intelligent context management that does not meet the needs of current reconfiguration. The reconfiguration middleware is a new approach to reconfiguration with recent researches [9, 11–15]. However, the proposals are limited in terms of solving fully automatic reconfiguration, big data stores sensor information, intelligent context management and these do not support real-time applications. In the previous study [1, 7], we proposed the RFL-IoT framework to change the behavior of IoTs applications based on Fuzzy Logic (FL). Smart

[1] https://teachablemachine.withgoogle.com/.
[2] https://www.tensorflow.org/tfx/guide/serving.

Context Management with FL approach analyzes the contextual information collected by IoTs applications. Based on user-defined FL rules, it determines the time that will trigger behavior change and transmits the decision to reconfigure and control the corresponding end-devices. One of our other proposed approaches to intelligent context management is based on Ontology for context modeling and the Bayesian network for context reasoning [6]. A Bayesian network that uses statistical methods can receive uncertain information to make predictions. Our empirical studies have limitations when assessing the environment with more complex contextual scenarios. To overcome this limitation, we continue to improve the framework by building prediction system using a machine learning approach. It uses some well-known machine learning algorithms to handle the vast amount of contextual information collected by IoTs applications.

2.2 Time-Series Sensor Data Prediction Algorithms

In this section we are going to explain the basics of time-series sensor data prediction. Then, we briefly introduce the algorithms used in IRPS. Time-series sensor data prediction is important in IoTs reconfiguration as it helps to make better decisions for trigger reconfiguration action. Different models are used in the literature for time-series data prediction. Today, there is increased interest in using Artificial Intelligence (AI) techniques to analyze complex data sets to extract useful information that can be used in the prediction of time-series sensor data trends and the effect of intervention actions on data trend predictions. Such information then can be used to change behavior. ML [2] and DL [4] techniques are the specific sub-sets of AI that are of particular current interest to academia and industry. ML and DL are applied to different situations where large and complex data sets are available and metadata, as well as more detailed information, could be extracted from the provided data if suitable means to extract and link the data were available.

For deep learning, the Feedforward Neural Network (FFNN) algorithm, also known as the multi-layer neural network, is a fundamental algorithm so that we can advance to more advanced algorithms such as convolutional neural network or recurrent neural network when the algorithms improve. This height is also a combination of many different layers in the entire neuron network. The two unsupervised learning algorithms are RF and NB that are also two famous algorithms in automatic control of IoTs applications. Random Forest is another machine-learning method that is successfully implemented in a wide variety of fields, including animal science. This regression or classification method makes use of decision trees: a sequence of rules that split the data in a way that most optimally reduces variation. Each tree receives a random subset of training samples, and then the algorithm randomly selects a subset of variables at each split in the tree. These trees, which are relatively poor classifiers individually combined into an ensemble of trees called a random forest, which is used for prediction. The prediction results of a random forest are a summation of the prediction outcomes of many individual trees. Naive Bayes is a family of classifiers that implements Bayesian techniques to form a simple network based on previous probabilities. Its method relies on independence between the input variables, but it performs surprisingly well even under conditions that might be considered suboptimal for the algorithm. Despite the relative simplicity of its

algorithm, naive Bayes is still widely used. To implement re-configuring IoTs applications using Fuzzy Logic, system reads data from sensor as crisp input for fuzzyfication then fuzzy input from fuzzyfication processed with inference system (fuzzy rule bases), then fuzzy output processed with defuzzyfication and produces crisp output to change the behaviors. By discovering the advantages and disadvantages of each technique in reconfiguration action case study, it is hoped to gain a better understanding of the wide variety of available tools for predicting complex contexts to trigger changing behavior actions.

3 Development of IRPS

The platform we use to build algorithms on models is to use the TensorFlow library. TensorFlow is an open source software developed by Google to perform machine learning, which is widely used by many researchers and large companies. TensorFlow provides many libraries for programmers to demonstrate machine learning algorithms and an application to implement these algorithms. A calculation expressed by TensorFlow can be performed with little or no modification in a range of heterogeneous systems mobile devices such as phones and tablets or distributed systems large scale spread of hundreds of machines or various computing devices like GPU cards. One of the important reasons we chose the TensorFlow platform is the ability to integrate the framework. The algorithms implemented in Tensorflow can be easily transplanted on many heterogeneous systems. In this section, we will explain the process of developing IRPS, the implementation of the algorithms.

3.1 Methodology

Figure 1 represents the methodology of research process can be learned for training data which are analyzed by a classification algorithm. The test data are used to calculate the accuracy of classification algorithms. There are many algorithms that can be used for classification for IoTs application such as RF, BN, rule based, neural network algorithms. In our system, we implement four algorithms: one algorithm is deep learning (FFNN), two algorithms are unsupervised learning (RF, NB) and the logical algorithm (FL). The classification of data has two step processes are learning and prediction.

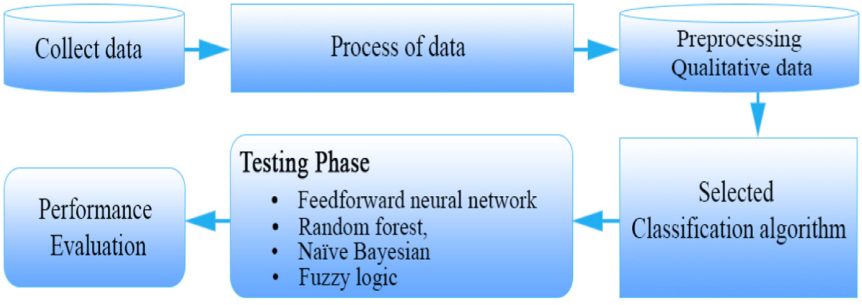

Fig. 1. Methodology of classification algorithms used in IRPS

3.2 Design and Implementation of IRPS

IRPS Training Model

In order to predict the importance of IoTs data prediction as well as show the results of its algorithms, we create a web-based system that shows the calculations in a more comprehensible way for the IoTs application developer. On that basis, IoTs application developers can refine input features, output features, algorithms to get the most suitable results for their specific context. Since it is a system on the web, the client-server architecture is the most common and appropriate. It is designed in a way that the user sends all the necessary data to the server. The server analyses and processes the request, and later builds to available model and predicts the results. Different parts of the system are implemented with different technologies. Express.JS, is used for making this API Restful API system and library TensorFlowJS (Tf.js). Tf.js is faster for small models, but when model becomes large training becomes 10–15x slower[3]. In the problem of reconfiguring IoTs applications which often involves a lot of sensor time-series data. The value of these features is usually not complicated, so we use the platform mainly as TensorFlow.JS instead of the python platform. We implemented IRPS on the server with the following hardware: Intel Xeon E3-1225 v5 @3.30 GHz (4 CPUs) ~ 3.3 GHZ, 32 GB RAM.

In this section, the system has three steps: *Step 1:* upload the Dataset; *Step 2:* select the input features and output features that recognized from the dataset; *Step 3:* select the algorithm among the 04 mentioned algorithms in Sect. 3.1 to train. Once the model has been trained, the user can download the model to use for prediction and see results like accuracy and loss function. The system is based on TensorFlow framework with associated libraries to access online data. The operation of the system is shown in Fig. 2.

IRPS Predicting Data

For predictive systems on models created from training, we use ReactJS[4] and TensorFlow to build web applications that run on the Node.JS server. We experiment on the Edge layer instead of the Cloud layer with the Raspberry Pi 4. The device is configured as a quad-core ARM Cortex-A72 CPU clocked at 1.5 GHz, 2 GB RAM. Data is transferred between Express.JS Restful API and ReactJS (Fig. 3).

The result of the training process according to the data, characteristics, algorithms of the user selected are two files {*model.json; weights.bin*}. The predictive system is deployed at the edge and the gateway which do not need powerful hardware. The prediction only needs to select the model corresponding to the previously selected algorithm, the data series to test according to the given syntax. Figure 4 shows the prediction of controlling room fan opening based on the parameter {room temperature, outside temperature, whether or not people in the room exist}, input data 24.11, 26.66 1. IRPS will immediately predict the output: Fan on/off mode respectively ~1%/~99%. Based on the predicted results, the system will trigger application behavior changes to the R-IoT Framework to perform the next steps of the reconfiguration process.

[3] https://www.tensorflow.org/js/tutorials.

[4] https://reactjs.org/.

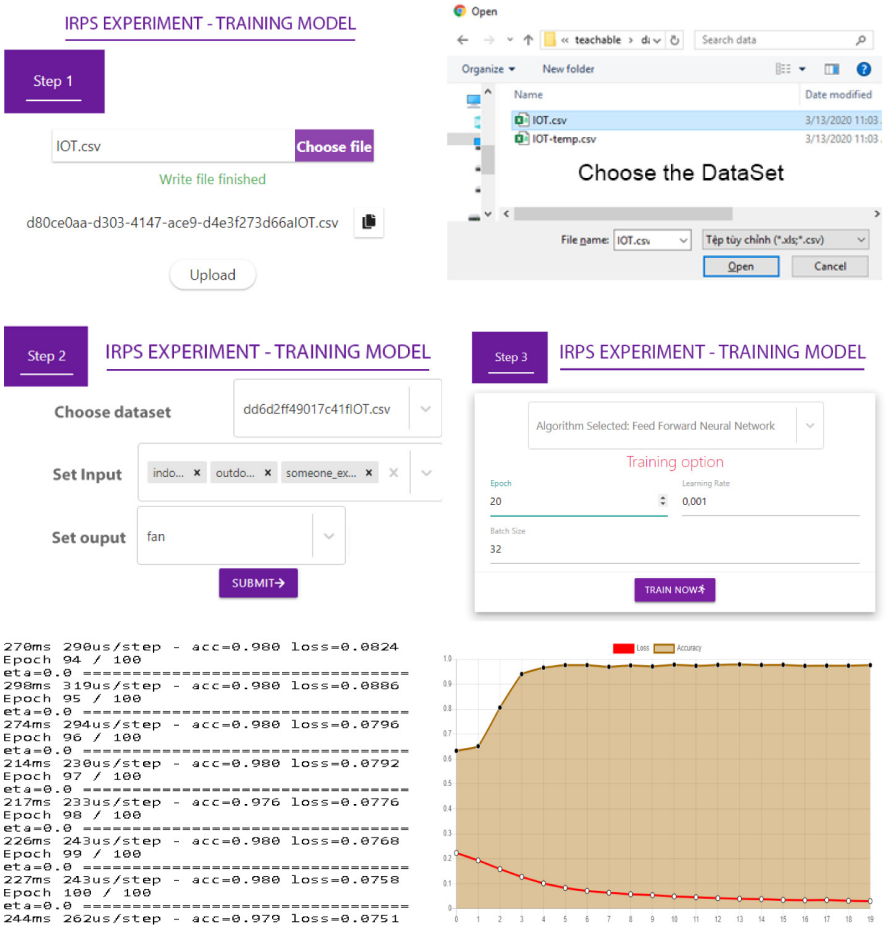

Fig. 2. Steps to training model on user dataset

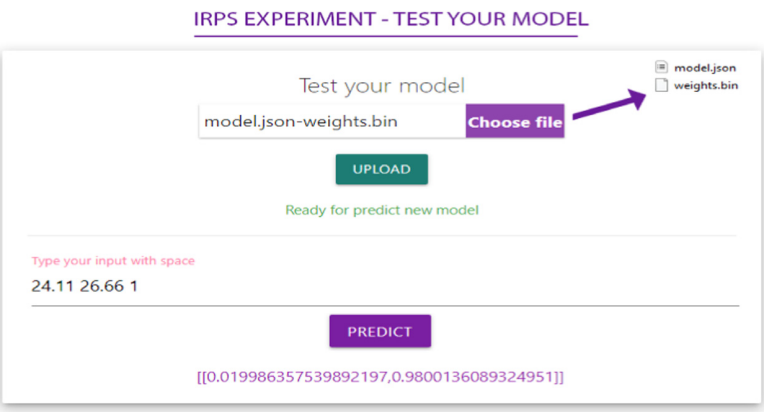

Fig. 3. Steps to perform prediction behavior based on model

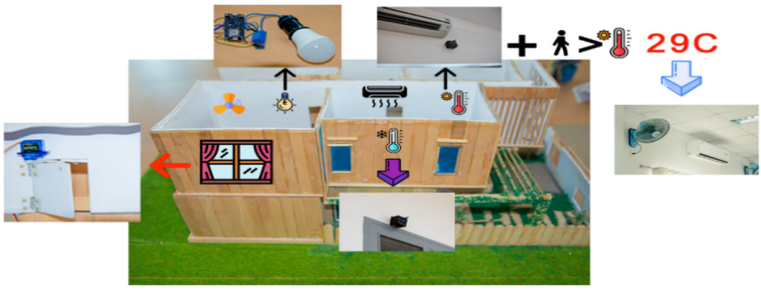

Fig. 4. Overall each smart room: temperature, fan and light control service

4 Case Study

In this section, we will show the actual output of the system and see how it performs on two case studies. We consider two different datasets in order to compare the results. The first one is a dataset for the smart room and the second one has data that shows the smart hydroponic cultivation. The details of how to setup the data, how many input, output features and how to choose the parameters for each experiment are described in the contextual information of each case study.

4.1 Smart Room

In the first case, IRPS is used to predict to change the state of fan and light in each room of the smart room. The framework is also designed in order to collect data from

Table 1. Services and smart scenarios in the smart room

Service	Scenario information
Temperature control service	The temperature control service performs switching on/off of fans and opening/closing windows based on information such as working time, indoor temperature/outdoor temperature, noise level, and the presence of people in the room. For example, if it is during working hours, the outdoor temperature is higher than 30 °C and someone is in the room, temperature control service will be performed. By comparing the indoor and outdoor temperatures, if the indoor temperature is higher, the windows open, other wires, turn on the fan. If the indoor temperature is lower than the outdoor temperature and there are no people in the room, turn off the fan
Light control service	Service control on/off lights in the room is affected by information such as working time, occupants and lighting levels. By comparing the light levels inside the room, we can make a decision to turn on/off the light. For example if no one is in the room during working time, the light will turn off, if there is a person in the room and the lighting level is low, the room lights will be turned on

sensors to monitor and store in the database {noise level, lighting level, exist person, outdoor/indoor temperature, window state} and it reconfigures the fan/light state. Table 1 shows the scenario and service information for the experiment (Fig. 6).

Table 2. Example of some smart room scenario

Noise level	Lighting level	Exist person	Indoor temp over 30	Temp control	Higher temp in than out	Window	Fan	Light
high	high	true	true	on	true	open	on	off
low	low	false	true	off	false	close	off	off
high	low	true	false	off	false	close	off	on
low	high	false	false	off	false	close	off	off
low	low	false	true	off	false	close	off	off
low	high	false	false	off	false	close	off	off

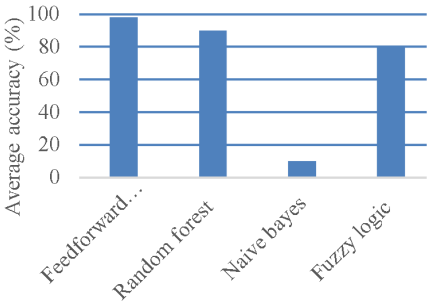

Fig. 5. Average accuracy (percent)

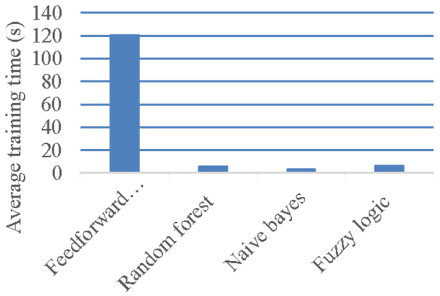

Fig. 6. Average training time (s)

We prepare the dataset for experiments from Kaggle dataset[5]. This is a time series data set measured at a smart room from 2016-11-27 17:22:45 until 2017-07-30 12:05:30, with a total of 14573 records. The above data set has a total of 10 features (date-time, Volume [mV], Light_Level [Ohms], Temperature-DHT [Celsius], Pressure [Hectopascal], Temperature-BMP [Celsius], Relative_Humidity [%], Air_Quality [Ohms], Carbon_Monoxide [Ohms], Nitrogen_Dioxide [Ohms]). Data pre-processing was applied to prepare the raw data. Pre-processing data is an important step to convert the raw data as shown in Fig. 5 into a format that can replace missing values to improve data quality. Next, we conducted divided into two sets of data with 80% for training and 20% for testing (Fig. 7).

[5] https://www.kaggle.com/saikatchoudhury/starter-smart-home-dataset-with-a8895cb2-c.

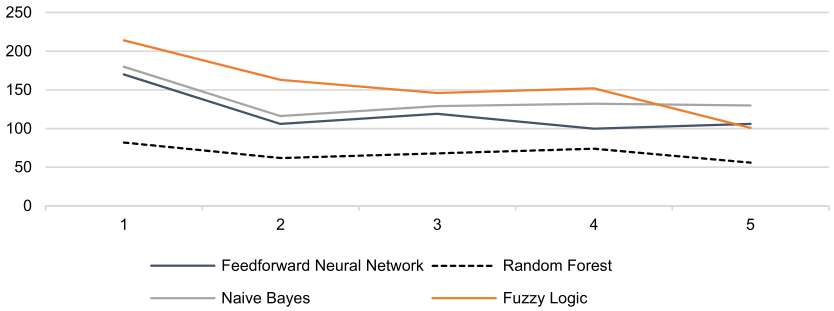

Fig. 7. Average reconfiguration time result

4.2 Smart Hydroponic Cultivation

In the second case, IRPS is used to predict which algorithm is suitable for the problem of hydroponic cultivation of 04 plants: tomatoes, lettuce, amaranth, vegetables. We prepare the dataset for experiments from process of ornamental dragon in Da Nang (vegetable garden). This is a time series data set measured from 2019-06-27 until 2019-12-30, with a total of 4574 records. The above data set has a total of 5 features (date time, type of plant, staging temperature, ambient temperature, humidity, PPM). The scenario of the hydroponic vegetable system is described as Table 2.

Table 3. Some scenarios in the context of a smart hydroponic cultivation

Service	Scenario content
Pump control service	The system collects real time indicators such as crop type, ppm, temperature, humidity of the planting system. Artificial intelligence techniques based on that will make a decision whether to change the reconfiguration behavior such as performing regression pump and regression pump time, or adding nutrients to the tank
Fan, light control service	Artificial intelligence techniques will use temperature and humidity data for 30 min continuously to predict the likelihood of heavy rain or sunshine. For example, with a possibility of rain above 80%, the system will make a decision that the roof will be pulled out. If the temperature is too high, the fan will be turned on, otherwise the lighting will turn on

The system will change behavior of pump, Fan, Light, outhouse control when meeting relevant sensor context information. The behaviors are shown as follows (Tables 3 and 4):

The devices and sensors used in the system are: Arduino WEMOS D1, Ds18b20, PH, relay - SLI, Module DHT22, Water pump motor, Engine sensor 775, mist kit, fan 5v. The output of the system is shown in Fig. 8.

Most of the analyzed datasets, as the two previous cases, show the following Table 5. For FFNN, with extremely high accuracy, the main reason is that our problem data is

Table 4. Some actions of smart hydroponic cultivation

Service	Scenario content
Circulating pump	Normal pump; pump with double time, no pumps
Nutrient pumps	Pumps A and B; pumps adding water, no pumps
Temperature-based behavior	Turn on/off {fans; misting; light}; no action
Rain-based behavior	Pull the blinds; retract the curtain; no action

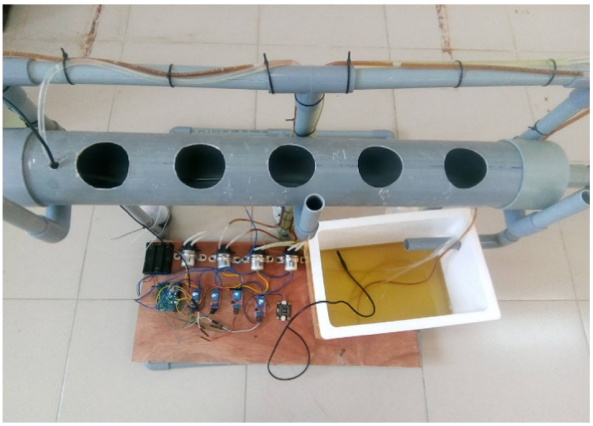

Fig. 8. Prototype of hydroponic cultivation system

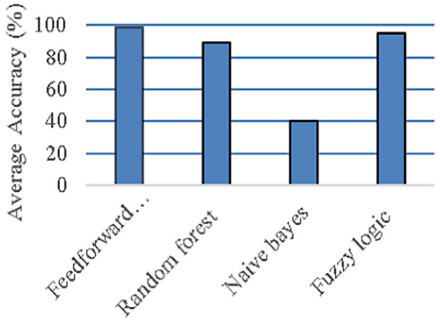

Fig. 9. Average accuracy (percent)

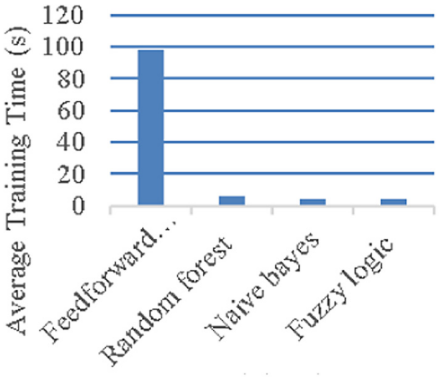

Fig. 10. Average training time (s)

not complicated, so the classification algorithm of neural network can work optimally in the distribution problem. However, training time and computing resources are limited to this algorithm. In order to overcome the above limitations, it is possible to use the server to optimize performance and use the save and reload feature that tensorflow.js provides

to limit the training many times. However, for our problem, the need to train many times is unavoidable because as stated in the idea, we cannot impose a single rule. For random forest, the algorithm returned amazing predictions when we had never touched normalize data but the results were really acceptable. With over 90% of the correct answers, it also partly reflects the applicability of this algorithm. Building algorithmic models by voting in different decision trees has helped RF become a highly appreciated algorithm in complex classification problems from linear-data to non-linear data. For our reconfiguration prediction, RF has worked perfectly in both training time and in terms of accuracy, in addition to using it when programming is simple and easy to use with different test-bed. NB, as mentioned in the theory, is an algorithm with very fast execution time because the algorithm only calculates the probability of being input from the input data. For example, logistic regression uses weight and bias to optimize the calculation process and uses decent gradients to find the optimal solution. While Naïve Bayes does not have the optimal solution to occur that is the reason make the reconfiguration time is slower for the time series data set that we give. The prediction with not high accuracy was anticipated, but we can completely improve this algorithm by adding one several other factors to improve its accuracy include normalizing data, reducing noise, missing data or errors. Finally, we can use fuzzy logic algorithm when the rules can only be set once while reprogramming, but the results are predictive and fast. The processing of this algorithm is extremely good. However, it is really difficult to implement a specific test-case IoTs because the setting rules in R-IoT framework takes a lot of time. In fact, there are exceptions in this rule, confirmed in fewer processed datasets, which means that finally the results depend on the dataset itself (Figs. 9 and 10).

Table 5. Comparison of the performance experiments

Algorithms	Accuracy (%)		Training time (s)		Reconfiguration time (ms)	
	First case	Second case	First case	Second case	First case	Second case
Feedforward neural network	98	99	120.2	98.4	143.75	133.5
Random forest	90	89	5.5	6.1	68.4	108.4
Naive Bayes	10	40	3.4	3.9	135.4	125.7
Fuzzy logic	80	95	6.5	4.3	235.2	115.4

5 Conclusion

In this paper, we present IoTs Reconfiguration Prediction System (IRPS) that helps future developers IoTs application solutions to choose the most suitable data prediction algorithms for trigger their reconfiguration actions. IRPS performs data prediction for sensor

readings, using four different algorithms, and compares their performances regarding two different evaluation metrics (accuracy to decision making, reconfiguration time). Additionally, the monitoring screen from IRPS visualizes the results obtained from the data prediction process. This proposed system helps IoTs developers to choose the most suitable time-series sensor data prediction algorithms for trigger IoTs reconfiguration actions. In future, we continue to our work on improving IRPS such as: implementation many AI algorithms into the system to support the intelligent context management of IoTs reconfiguration framework.

References

1. Nguyen-Anh, T., Le-Trung, Q.: RFL-IoT: an IoT reconfiguration framework applied fuzzy logic for context management. In: IEEE International Conference on Research, Innovation and Vision for the Future (RIVF). IEEE (2019)
2. Sharma, K., Nandal, R.: A literature study on machine learning fusion with IOT. In: 2019 3rd International Conference on Trends in Electronics and Informatics (2019)
3. Nikolov, N.: Research firmware update over the air from the cloud. In: International Scientific Conference Electronics (ET2018). IEEE, Bulgaria (2018)
4. Tang, J., Sun, D., Liu, S., Gaudiot, J.-L.: Enabling deep learning on IoT devices. Computer **50**, 92–96 (2017)
5. Anh, T.N., Le Trung, Q., Hai, B.T., Van, D.H: R-IoT: a framework for IoTs reconfiguration in cloud. In: The 6th Conference on Information Technology and Its (CITA) (2017)
6. Nguyen-Anh, T., Le-Trung, Q.: An IoT reconfiguration framework applied ontology-based modeling and bayesian-based reasoning for context management. In: 2019 6th NAFOSTED Conference on Information and Computer Science (NICS), IEEE NICS 2019 (2019)
7. Nguyen-Anh, T., Le-Trung, Q.: An IoTs reconfiguration framework with intelligent context management. In: IEEE Seventh International Conference on Communications and Electronics (ICCE). IEEE (2018)
8. Mohammadi, M., Al-Fuqaha, A., Sorour, S., Guizani, M.: Deep learning for IoT big data and streaming analytics: a survey. IEEE Commun. Surv. Tutor. **20**, 2923–2960 (2018)
9. Perera, C., Zaslavsky, A., Christen, P.: Context aware computing for the Internet of Things: a survey. IEEE Commun. Surv. Tutor. **16**, 414–454 (2013)
10. Craig, G., Adnan, Al., Quan, B.: OTAP arbitration effects in randomly deployed WSN's. In: International Telecommunication Networks and Applications. IEEE, Australia (2015)
11. Sivaharan, T., Blair, G., Coulson, G.: GREEN: a configurable and re-configurable publish-subscribe middleware for pervasive computing. In: Meersman, R., Tari, Z. (eds.) OTM 2005. LNCS, vol. 3760, pp. 732–749. Springer, Heidelberg (2005). https://doi.org/10.1007/11575771_46
12. Ruckebusch, P., Van Damme, J., De Poorter, E., Moerman, I.: Dynamic reconfiguration of network protocols for constrained Internet-of-Things devices. In: Mandler, B., et al. (eds.) IoT360 2015. LNICST, vol. 170, pp. 269–281. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47075-7_31
13. Aberer, K., Hauswirth, M., Salehi, A.: A middleware for fast and flexible sensor network deployment. In: Proceedings of 32nd International Conference on Very Large DataBase. ACM (2006)
14. Gámez, N., Fuentes, L.: FamiWare: a family of event-based middleware for ambient intelligence. Pers. Ubiquit. Comput. **15**(4), 329–339 (2011)
15. Henry, J., Marti, H.: Quick and efficient link quality estimation in wireless sensors networks. In: Wireless On-Demand Network Systems and Services. IEEE, France (2018)