Deep Learning based Self-Adaptive Framework for Environmental Interoperability in Internet of Things

Euijong Lee
Chungbuk National University
Cheongju, Republic of Korea
kongjjagae@cbnu.ac.kr

Sukhoon Lee
Kunsan National University
Kunsan, Republic of Korea
leha82@kunsan.ac.kr

Young-Duk Seo*
Inha University
Incheon, Republic of Korea
mysid88@inha.ac.kr

ABSTRACT

IoT interconnects various entities including users, devices, information, and services, thus, interoperability is essential to realize the Internet of Things (IoT). There are various perspectives to support interoperability IoT environment, and one interoperability problem is related to constructing IoT environments at runtime. The problem is caused by that it is hard to predict IoT environments at design time. In other words, the IoT environment can be dynamically changed, thus an IoT system has to adapt to the change. To solve the environmental interoperability, a self-adaptive framework based on deep neural networks (DNN) is proposed to construct IoT systems at runtime. The proposed framework provides requirement verification and adaptation at runtime. Arduino-based IoT environments were implemented, and experiments were performed to show the efficiency. The results showed the reasonable performance to verify requirement satisfaction using DNNs.

CCS CONCEPTS

• **Software and its engineering**; • **Applied computing**;

KEYWORDS

Internet of Things, Interoperability, Self-adaptive software, Deep neural network

ACM Reference Format:
Euijong Lee, Sukhoon Lee, and Young-Duk Seo. 2022. Deep Learning based Self-Adaptive Framework for Environmental Interoperability in Internet of Things. In *The 37th ACM/SIGAPP Symposium on Applied Computing (SAC '22), April 25–29, 2022, Virtual Event, .* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3477314.3507191

1 INTRODUCTION

The Internet of Things (IoT) interconnects various entities such as user, devices, information, and services. The interconnected entities can provide various services in several fields including smart cities, smart farms, smart homes, and automobiles. However, to accomplish interconnection among different entities, interoperability is an essential element [10]. Interoperability can be defined as "capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units" in an international standard (i.e., ISO/IEC 2382:2015 [14]). With the definition, interoperability can be interpreted as various views in the IoT environment [10]. In this paper, we focused on environmental interoperability that is caused by a divergence of IoT environments.

As described, IoT required the interconnection of different entities, and pre-defined information is used to construct IoT environments. Therefore, there are various standard exists to support interoperability of information changes [10]. However, the predefined approach has a chronic problem that is hard to estimate every possible combination of the entities. Also, it may not be interoperable to exchange meaningful information. To solve the interoperability problem, a self-adaptive framework based on a deep neural network (DNN) is proposed in this paper. The proposed approach aims to provide verification of requirements satisfaction and adaptation strategies using data from IoT entities with minimal information.

The remainder of this article is organized as follows. Section 2 provides background and related work. Section 3 introduced the proposed self-adaptive framework for IoT environments. Section 4 presents the results of empirical experiments, and Section 5 concludes this paper.

2 BACKGROUND AND RELATED WORK

In this section, background and related work are introduced. Section 2.1 and 2.2 introduced self-adaptive software and deep neural network. The related works are described in Section 2.3.

2.1 Self-adaptive software

Self-adaptive software aims to adjust various artifacts or attributes of software to adapt detected context by itself [16]. The context denotes everything in environments that can affect the software. In addition, the self-adaptive software continuously required detecting the context and adapting operation, thus a cycle should be needed to deal with self-adaptation [16]. The cycle is called the adaptation loop, and the loop consists of four processes. The processes are described as below:

- *Monitoring process* is responsible for collecting data from the software itself and operating environment.
- *Analyzing (detecting) process* is responsible for analyzing the symptoms by using data from the monitoring process.
- *Planning (deciding) process* is responsible for deciding how to change artifacts or attributes to achieve the batter performance. (i.e., it is responsible for detect adaptation strategies if and adaptation is required).

*Corresponding author

SAC '22, April 25–29, 2022, Virtual Event,

ACM ISBN 978-1-4503-8713-2/22/04.
https://doi.org/10.1145/3477314.3507191

- *Executing (acting) process* is responsible for applying change (i.e., adaptive strategy).

The loop that consisted of four processes is called MAPE-loop. In addition, the loop can have shared knowledge to share data among the processes, and the loop is named as MAPE-K loop [3].

2.2 Deep learning

Deep learning is a class of machine learning algorithms based on artificial neural networks. Also, the artificial neural network constructed a multilayered neural network (i.e., hidden layer) between input and output, and the network is called a deep neural network (DNN) [5]. DNN is an efficient algorithm to find some patterns between input and output, thus it is applied to various areas including self-adaptive systems [4]. In this paper, the design of DNNs is proposed to verify requirement satisfaction and extract adaptive strategies. The details of the DNN design are described in Section 2.2.

2.3 Self-adaptive software in IoT

In this section, recent studies that focused on self-adaptation for IoT-based system, and various research topics (e.g., architecture, algorithm, system, or framework) are introduced.

Nawaratne et al. [14] proposed self-evolving algorithms for data interoperability in IoT environments. They investigated need for interoperable algorithms to support IoT environments, and the investigation was used real data from Metropolitan Fire Brigade (MFB) in Victoria, Australia. A hardware and software architecture to design dynamic reconfigurable IoT environments was presented [17]. Burger et al. [1] proposed an IoT framework to support development and deployment of distributed IoT environments with adaptive hardware for continuous change, and the work is called Elastic IoT platform. A mode checking-based self-adaptive framework is presented [8, 11, 12] and the framework was named RINGA. RINGA provides a design of finite-state machines for IoT, and a game theory-based strategy extraction method [9] was applied to RINGA for runtime adaptation.

Recently, various studies that are applied machine learning for self-adaptation in IoT have been conducted. Van Der Donckt et al. [18] presented deep learning-based approach for adaptation space reduction, and the approach was named DLASeR. A MAPE-K loop-based self-adaptive architecture with cooperation of machine learning and model checking was proposed [2, 13]. Pauna et al. [15] presented a self-adaptive honeypot system to detect and prevent malicious attempts in IoT environments.

In summary, each of the studies includes distinct characteristics with various approaches in different target IoT environments. In this paper, we focused on environmental interoperability that is caused by a divergence of IoT environments, and proposed a self-adaptive framework with DNN design.

3 DEEP LEARNING BASED SELF-ADAPTIVE FRAMEWORK FOR IOT

In this section, details of the proposed framework are described. In Section 3.1, an overview of the proposed framework that is based on a neural network to solve environmental interoperability. Also, the neural network is described to design the neural network for self-adaptation in Section 3.2.

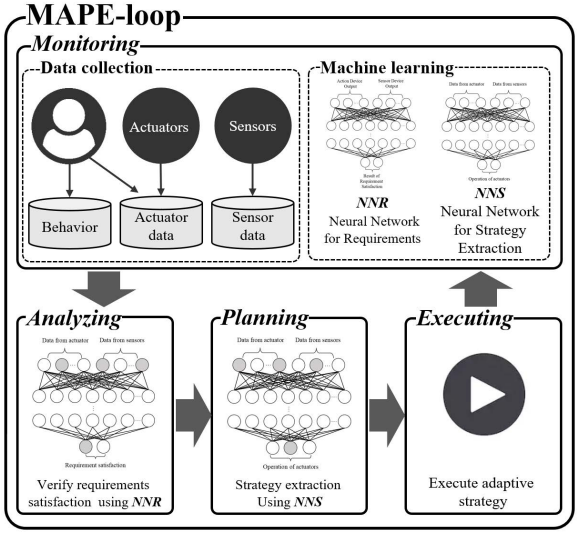

Figure 1: Overview of neural network based self-adaptive framework

3.1 Overview

The proposed framework focused on interoperability problems related to constructing IoT systems with various entities. The proposed approach constrained entities to compose IoT systems as actuators, sensors, requirements, and operations to minimize predefined information. The actuator is defined as a device that changes physical entity [6]. The sensor is defined as a device that measures physical entities and outputs digital data [6]. The requirement is generated from users and must have to be satisfied at runtime. The operation defined executions of the actuator to satisfy the requirements. However, in the proposed framework, However, it is assumed that the sensor only provides measured data without additional information (e.g., what is a physical entity related to the data and location of a sensor). Also, the actuator also only provides the status of itself and is operated by input values without additional information (e.g., an actuator can change which physical entity). Requirements and operation are assumed by user behavior. Also, requirements satisfaction and proper operation (i.e., adaptive strategy) are measured and extracted by using the data from sensors and actuators. The details are described below.

The proposed framework is consisted with MAPE-loop that is general process of the self-adaptive system, and the details of each processes are as follow. (See Figure 1) The monitoring process is the first step of the process and has two responsibilities: collecting data and performing machine learning. First, various data are collected from status of IoT devices (i.e., sensor and actuators) and user's behavior. The user's behavior can be collected in various ways; for example, operations of actuators and setting the requirements factor via the application. Therefore, the way of collecting behavior is not limited, and it is assumed that the operation of actuators from user's behavior potentially involves adaptation strategy. However,

the collected data are saved in the database and used for machine learning. If the data is sufficiently collected for machine learning, deep neural networks are trained and tested. The machine learning focused on two perspectives: context detection and extracting adaptive strategy from human intention. Therefore, two types of neural networks are generated as neural network for requirement (NNR) and neural network for strategy (NNS) The details of the criteria of generating the neural network are described in Section 3.2. The trained neural networks are used to analyzing and planning processes. The analyzing process is responsible for verifying requirement satisfaction using trained neural networks (i.e., NNR). In this process, it is analyzed which requirements are satisfied or not. The analyzed results are transferred and used in the planning process. The planning process is responsible to generate an adaptive strategy using NNS, and the adaptive strategy is transferred to the execution process. The execution process operates the adaptive strategy from the previous process.

As described previously, the proposed framework performs self-adaptation using result of trained neural networks (i.e., NNR and NNS). Therefore, the design of the neural networks is most important, and the details of the design are described in Section 3.2.

3.2 Neural network design for IoT

In this section, designs of neural networks are introduced. The neural network for requirement is described in Section 3.2.1, and the neural network for adaptive strategy extraction is described in Section 3.2.2

3.2.1 Neural network for requirement. The design of a neural network to verify satisfaction of user requirement (i.e., NNR) is described as below. The input data of the NNR have consisted of collected data from actuators and sensors, and the output is the satisfaction of user requirements. The input may be simply collected by sensing the status of sensors and actuators. Data from sensors is directly related to some factors (e.g., a light sensor is related to brightness). Also, the actuators may be related to some factors indirectly. (e.g., status of windows may affect brightness, humidity, and dust density).

The output requires definitions that can denote user satisfaction. For example, if an actuator is operated by a user, it may denote that a requirement does not satisfied. Because the operation may be caused to change a status that is related to the requirement. In other words, requirements are satisfied if any actuators are not operated by the user. Also, if an IoT system can provide a function that users can set criteria of satisfaction, the output data can be collected easily.

The number of hidden layer and number of neurons in a hidden layer can affect training performance and machine learning results. Therefore, criteria are proposed to adjust number of hidden layers and neurons.

- A number of hidden layers are bigger than a number of sensors and smaller than a summation of sensors and actuators.
- A number of neurons in a hidden layer are bigger than a number of inputs (i.e., summation of sensors and actuators) and smaller than three times the input.

Each requirement has a different effect on the input value, thus NNRs are generated for each requirement based on the design and the criteria. Finally, NNR implies the effect of the current status of sensors and actuators on each requirement. Therefore, the proposed framework can detect satisfaction of requirements using NNR at run-time.

Table 1: Component of the IoT environment for experiment

Type	ID	Location	Related requirement
Sensor	Humidity (in)	Inside	Humidity
	Brightness (in)	Inside	Intensity of illumination
	Humidity (out)	Outside	Humidity
	Brightness (out)	Outside	Intensity of illumination
Actuator	Lamp	Inside	Intensity of illumination
	Humidifier	Inside	Humidity
	Windows	Outside	Intensity of illumination and humidity

3.2.2 Neural network for self-adaptation. The design of a neural network to extract adaptive strategy (i.e., NNS) is described below.The input values of NNS are status of actuators and sensor data, and the output is the operations of each actuator. Therefore, the NNS is generated for each actuator. Also, the adaptive strategies must have to satisfy requirements, thus only data that satisfy users' requirements are applied to train the NNSs. As the result of the training, the NNSs can extract appropriate operations of each actuator to satisfy requirements in the current status. Design criteria for neurons and hidden layers is same as NNR.

4 EMPIRICAL EVALUATION

In this section, the experimental environment is described with Arduino implementation. Also, the experiments and results are described using data from the implementation.

4.1 Experimental environment

To perform experiments, a simple IoT environment was implemented. The IoT environment has two requirements that are the intensity of illumination and humidity. Table 1 shows the sensors and actuators that consist of the experimental environment. Also, it is assumed that the IoT environment has predefined ranges to verify satisfaction of the requirements. The brightness has to be adjusted from 170 to 200 lux, and the humidity required to be adjusted between 32 to 34 %.

Actuators can be classified into two types by how it affects: direct and indirect. The former is lamp and humidifier, because operations of the both actuators directly affects the requirements. The later is windows, because windows may affect requirements, but it is depending on status of outside environments. Therefore, sensors are located inside and outside to measure status of brightness and humidity in the different location. However, in this article, we only performed experiments that related to NNR and brightness, and the results are described in the next section.

4.2 Experiment results

The experiments were performed to evaluate the proposed neural network design (i.e., NNR). The IoT environment collected the data, and it includes various situations among actuators, sensors, and requirements. The data was collected every 100 MS, and only 836 and 241 data was used for training and testing to show the

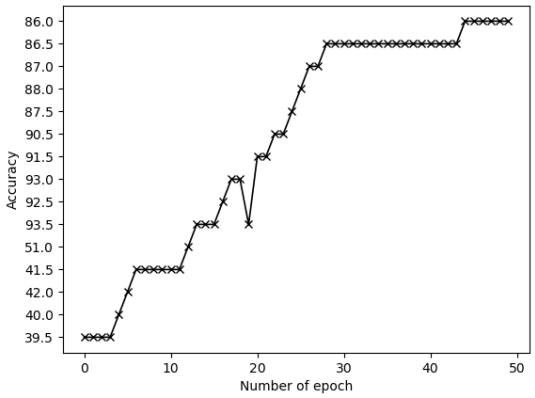

Figure 2: Results of NNR (accuracy)

excellence of the proposed approach. In the experimental setup, the hardware consists of Inter Core i7-10700K CPU (3.8 GHz), 32 GB memory, and NVIDIA Geforce RTX 3070. NNR and NNS are designed the proposed criteria (see Section 3.2.1), and the applied features in the neural networks are as below:

- *Linear transformation* was applied in the hidden layers for the incoming data.
- *L1 loss* [5] was applied as a loss function
- *ReLU* [7] was applied as activation function.
- *Stochastic gradient descent (SGD)*[5] was applied as an optimizer.
- *Learning rate, batch size, number of epochs* was set as 0.0002, 50, and 50

Figure 2 shows results of NNR, and figure denotes change rate of accuracy. In conclusion, the NNR shows accuracy with test data 86%, and the result shows reasonable accuracy even with small amount of training data.

5 CONCLUSION

In this article, a self-adaptive software framework with DNN design to solve environmental interoperability problems in IoT. The proposed framework consisted of MAPE-loop which is the general process of self-adaptive systems. In addition, two types of DNN designed are introduced: DNN to verify satisfaction of requirements (NNR) and DNN to fine adaptive strategies (NNS). The DNNs are applied in MAPE-loop and used to self-adaption. The experiments were performed to show the efficiency of the proposed framework, and Arduino-based IoT environments were implemented. The results showed reasonable results to verify requirement satisfaction using NNR. In the future, experiments will be performed to show the efficiency of NNS in a physical IoT environment. In addition, it is planned to extend the proposed approach to more complex IoT environments and to enhance with human-involved learning.

ACKNOWLEDGMENTS

This work was supported by Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00231, Development of artificial intelligence based video security technology and systems for public infrastructure safety) and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1G1A101097111).

REFERENCES

[1] Alwyn Burger, Christopher Cichiwskyj, Stephan Schmeißer, and Gregor Schiele. 2020. The Elastic Internet of Things-A platform for self-integrating and self-adaptive IoT-systems with support for embedded adaptive hardware. *Future Generation Computer Systems* 113 (2020), 607–619.

[2] Javier Cámara, Henry Muccini, and Karthik Vaidhyanathan. 2020. Quantitative verification-aided machine learning: A tandem approach for architecting self-adaptive iot systems. In *2020 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 11–22.

[3] Edmund M Clarke Jr, Orna Grumberg, Daniel Kroening, Doron Peled, and Helmut Veith. 2018. *Model checking*. MIT press.

[4] Omid Gheibi, Danny Weyns, and Federico Quin. 2021. Applying machine learning in self-adaptive systems: A systematic literature review. *arXiv preprint arXiv:2103.04112* (2021).

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[6] ISO/IEC 20924:2018(E) 2018. *Information technology — Internet of Things (IoT) — Vocabulary*. Standard. International Organization for Standardization, Geneva, CH.

[7] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. 2009. What is the best multi-stage architecture for object recognition?. In *2009 IEEE 12th international conference on computer vision*. IEEE, 2146–2153.

[8] Euijong Lee, Young-Gab Kim, Young-Duk Seo, Kwangsoo Seol, and Doo-Kwon Baik. 2017. Runtime verification method for self-adaptive software using reachability of transition system model. In *Proceedings of the Symposium on Applied Computing*. 65–68.

[9] Euijong Lee, Young-Duk Seo, and Young-Gab Kim. 2019. A nash equilibrium based decision-making method for internet of things. *Journal of Ambient Intelligence and Humanized Computing* (2019), 1–9.

[10] Euijong Lee, Young-Duk Seo, Se-Ra Oh, and Young-Gab Kim. 2021. A Survey on Standards for Interoperability and Security in the Internet of Things. *IEEE Communications Surveys & Tutorials* 23, 2 (2021), 1020–1047.

[11] Lee, Euijong and Kim, Young-Gab and Seo, Young-Duk and Seol, Kwangsoo and Baik, Doo-Kwon. 2018. RINGA: Design and verification of finite state machine for self-adaptive software at runtime. *Information and Software Technology* 93 (2018), 200–222.

[12] Lee, Euijong and Seo, Young-Duk and Kim, Young-Gab. 2019. Self-adaptive framework based on MAPE loop for Internet of Things. *sensors* 19, 13 (2019), 2996.

[13] Henry Muccini and Karthik Vaidhyanathan. 2020. Leveraging machine learning techniques for architecting self-adaptive iot systems. In *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 65–72.

[14] Rashmika Nawaratne, Damminda Alahakoon, Daswin De Silva, Prem Chhetri, and Naveen Chilamkurti. 2018. Self-evolving intelligent algorithms for facilitating data interoperability in IoT environments. *Future Generation Computer Systems* 86 (2018), 421–432.

[15] Adrian Pauna, Ion Bica, Florin Pop, and Aniello Castiglione. 2019. On the rewards of self-adaptive IoT honeypots. *Annals of Telecommunications* 74, 7 (2019), 501–515.

[16] Mazeiar Salehie and Ladan Tahvildari. 2009. Self-adaptive software: Landscape and research challenges. *ACM transactions on autonomous and adaptive systems (TAAS)* 4, 2 (2009), 1–42.

[17] Matteo Antonio Scrugli, Daniela Loi, Luigi Raffo, and Paolo Meloni. 2019. A runtime-adaptive cognitive IoT node for healthcare monitoring. In *Proceedings of the 16th ACM International Conference on Computing Frontiers*. 350–357.

[18] Jeroen Van Der Donckt, Danny Weyns, Federico Quin, Jonas Van Der Donckt, and Sam Michiels. 2020. Applying deep learning to reduce large adaptation spaces of self-adaptive systems with multiple types of goals. In *Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 20–30.