J Inf Process Syst, Vol.17, No.6, pp.1071~1082, December 2021
https://doi.org/10.3745/JIPS.01.0083

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Anomaly Detection of Facilities and Non-disruptive Operation of Smart Factory Using Kubernetes

Guik Jung*, Hyunsoo Ha**, and Sangjun Lee*

Abstract

Since the smart factory has been recently recognized as an industrial core requirement, various mechanisms to ensure efficient and stable operation have attracted much attention. This attention is based on the fact that in a smart factory environment where operating processes, such as facility control, data collection, and decision making are automated, the disruption of processes due to problems such as facility anomalies causes considerable losses. Although many studies have considered methods to prevent such losses, few have investigated how to effectively apply the solutions. This study proposes a Kubernetes based system applied in a smart factory providing effective operation and facility management. To develop the system, we employed a useful and popular open source project, and adopted deep learning based anomaly detection model for multi-sensor anomaly detection. This can be easily modified without interruption by changing the container image for inference. Through experiments, we have verified that the proposed method can provide system stability through nondisruptive maintenance, monitoring and non-disruptive updates for anomaly detection models.

Keywords

Anormal Detection, Continuously Learning, Kubernetes, Non-disruptive Operation, Smart Factory

1. Introduction

With the emergence of the 4th industrial revolution, many advanced nations have recognized the smart factory concept as critical for industrial development, and have rapidly applied smart factories for industry promotion [1]. A smart factory is an intelligent plant incorporating various information and communication technologies (ICTs) such as digital automation solutions, artificial intelligence (AI), and big data analysis. Smart factory is a management technique to control the facility, using Internet of Things (IoT) devices installed in the facility and machines inside the factory, by collecting and analyzing data in real time. IoT devices collect information from manufacturing facilities, testers, and sensors connected to known process objects; and transmit control signals. Smart factories are configured to run with hundreds to thousands of complexly connected IoT devices. Thus, it is necessary to optimize processes and develop efficient operational processes.

For efficient process supervision and management of smart factories, monitoring, control, and data acquisition systems should supervise industrial processes in real time and make decisions using available data, pipelines, and graphic user interfaces (GUIs) [1]. Data collection and analysis are core tasks for

Manuscript received February 9, 2021; first revision May 17, 2021; second revision June 3, 2021; accepted June 12, 2021.
Corresponding Author: Sangjun Lee (sangjun@ssu.ac.kr)
* Dept. of Software, Soongsil University, Seoul, Korea (rndlr96@soongsil.ac.kr, sangjun@ssu.ac.kr)
** Dept. of Software Convergence, Soongsil University Seoul, Korea (dhy03196@naver.com)
Guik Jung and Hyunsoo Ha contributed equally to this work.

decision making. Smart factory data includes state information, sensor data, and operation logs regarding various facilities in the system, which can greatly assist decision-making at various levels [2,3].

Significant losses are likely if processes are disrupted due to facility anomalies, process recipe changes, and management or software problems, particularly for smart factory environments where most operational processes are automated, including facility control, data collection, and decision making. Many studies have offered various solutions to minimize loss, maintain facilities in real time, and ensure efficient operation by utilizing collected data. Most such solutions incorporate process recipe management, and anomaly detection based on AI, and real-time monitoring [4-6]. However, although various studies have proposed problem solving systems based on collecting and analyzing data, application of the derived solutions for container orchestration based smart factory infrastructure have been relatively passive. Thus, this paper proposes an architecture that utilizes Kubernetes to effectively applies solutions for efficient operation and facility management in smart factories [7].

We adopted a two-fold approach. First, we developed a smart factory infrastructure utilizing Kubernetes, a container orchestration platform. Afterward, deploy application programs and processes for facility control, data collection, processing and storage, and monitoring in the form of containers in this infrastructure. Since the configuration is automatically maintained through this process, immediate recovery is possible even when the program is suddenly shut down due to a natural disaster or network error. Furthermore, the proposed infrastructure makes minimizing disruption and achieving horizontal scaling and operation. Thus, even if manufacturing processes change within the factory, immediate updates are possible without the need to restart services.

Second, we implemented an anomaly detection system for continuously learning based on collected data. The deep learning model trained only once with old data cannot properly respond to the new data [8]. In the proposed system, we continue to learn after deploying the anomaly detection model through pipeline construction. We periodically create a deep learning model that reflects recently collected data through the “CronJob” function of Kubernetes. This allows to periodically create anomaly detection models reflecting new data without interruption and use it in the process. Consequently, our infrastructure can be adapted for the new manufacturing process. Deep learning algorithms for anomaly detection are an active research area, increasing the possibility of new algorithms emerging with improved accuracy over existing algorithms. For this reason, we implemented the infrastructure that can be changed to State-of-the-art models without shutting down existing services.

The proposed system is based on the container orchestration technology, Kubernetes, to reduce losses due to process interruptions and maintenance costs, while serving a flexible model change to enable convenient operation management. For example, if the factory manufactures a new product, it is possible to easily replace an anomaly detection model for a new product without restarting by changing the Kubernetes container image.

The remainder of this paper organized as follows. Section 2 briefly introduces theoretical backgrounds and concepts relating to Kubernetes, anomaly detection, LSTM, and elastic stacks. Section 3 details the proposed system architecture and discusses requirements to apply and utilize the solution properly for operating a smart factory efficiently. Section 4 defines the environment for experiments and analysis the result of experiments. Section 5 summarizes and concludes the paper.

2. Background

This section introduces Kubernetes, which is the basis for the smart factory infrastructure for non-

disruption operation, anomaly detection and elastic stack for data collection in real time, processing, and storing.

2.1 Kubernetes

Kubernetes is an open source container orchestration platform developed by Google Lab to provide distribution and expansion of application containers and automated cluster operation and comprises masters and nodes according to their configuration and roles [7]. A master helps to manage the cluster, and comprises a controller, scheduler, etcd, and an application programming interface server. This study proposes a non-disruptive smart factory environment using Kubernetes.

2.1.1 Job

A Job is a batch task that is useful when you want to perform a task that must be terminated after it is executed, not a task that must be executed continuously. It ensures successful completion of tasks as many as the designed times. CronJob is a managed job that performs tasks according to a regular schedule. Pod is created when the designated execution time occurs. The time setup clause is the same as for the Linux CronTab and tasks can be executed in serial or parallel. In this work, the CronJob feature was used for continuous learning and it has led to constant performance improvement and high adaptability to new data.

2.2 Anomaly Detection

Anomaly detection is a field of data analysis that understands hidden sequence patterns in time-series data and finds anomalies that deviate from or tend to do so. Since anomaly detection is a very important research topic for industrial development, various methods have been studied. There are three main types of anomaly detection methods based on machine learning. Supervised deep anomaly detection, semi-supervised deep anomaly detection, and unsupervised deep anomaly detection [9]. In this work, we used unsupervised deep anomaly detection with LSTM based model [10].

2.2.1 SVM

Support vector machine (SVM) is a supervised learning-based model used for pattern recognition and data analysis and is also commonly used in anomaly detection by finding outliers for various data [11]. Basically, SVM is based on supervised learning, but for one-class SVM (OC-SVM) model, it performs anomaly detection through unsupervised learning using only normal data [12].

2.2.2 LSTM

Long-short term memory (LSTM) networks is an improved network model of recurrent neural network (RNN) that enables long-term sequences to be learned by solving the problem of long-term dependencies, which does not deliver sufficient information as the time step increases in the RNN [13]. LSTM add the cell-state to the hidden state of the RNN, and add input, forget, and output gate to protect and control the cell-state by selecting the memory. Generally, LSTM provide better performance than other RNN algorithms in most fields, including predicting time-series data [14].

2.2.3 Anomaly detection in smart factory

Various studies are being researched on anomaly detection using machine learning in smart factory. In

many existing works, SVM has been used as an anomaly detection model due to its low computation cost and complexity [15-17]. However, SVM has limitations that it cannot effectively utilize time-series data from smart factories. The sensor data generated by the smart factory has two characteristics. The first is that most of the collected data is normal data, and very few data are abnormal data. These Skewed data make it difficult to train a SVM model through supervised learning. The second characteristic is that it is time-series data. SVM model cannot handle sequential data and thus fails to learn form time-series data effectively. To detect anomalies in data with these characteristics in the proposed system we used the LSTM-AE model [18]. It is an encoder-decoder structure based on LSTM, which extracts features through Autoencoder by normal data in the form of time-series and reconstructs values based on it. Therefore, if the reconstruction value of the LSTM-AE model which normal data were trained has a significant error with the ground truth value, it indicates smart factory is an abnormal status.

2.3 Elastic Stack

The Elastic Stack is an elastic open source service platform to search, analyze, and visualize data in real time after fetching all types of data. It incorporates Elasticsearch, Logstash, and Kibana [19]. This study containerized Elastic Stack for processing, storing, and visualizing real-time data generated at the sensors.

2.3.1 Logstash

Logstash is a data processing pipeline, coordinating data collection, conversion, and delivery to the repository. It supports various input types, collects data regardless of format or complexity, and can perform data analysis and conversion by utilizing various filter libraries. It can also stream data to a preferred repository. We installed and containerized Logstash, with the RabbitMQ input plugin installed.

2.3.2 Elasticsearch

Elasticsearch is an open source engine to store, search, and analyze all data types, including texts, numerical, and unstructured data. Elasticsearch stores collected data by indexing, and groups related documents. It can be horizontally scalable through shard. We employed Elasticsearch for storing and searching collected data, and the Kubernetes persistent volume was used to ensure permanent storage of the containerized repository.

2.3.3 Kibana

Kibana is an open source analysis and visualization platform, designed to be compatible with Elasticsearch. It visualizes data stored in the Elasticsearch index with various charts, tables, and maps. We used Kibana to monitor collected data and anomaly detection.

3. Proposed System

This section details the proposed system architecture and cluster configuration. The proposed system allows smart factory facilities to be maintained by real-time anomaly detection without disruption, incorporating sensor data collection, processing, storing, and monitoring by constructing smart factory

infrastructure using Kubernetes with LSTM based encoder-decoder anomaly detection. We configured a Kubernetes cluster comprised of four Raspberry Pi computers acting as smart factory facilities, one master node, and four worker nodes, as shown in Fig. 1.

Table 1 presents Raspberry Pi and node's specification. Node4 with GPU operates as a learning node for creating an anomaly detection model, and pods for learning and anomaly detection are placed only in node4.

Fig. 2 shows the proposed smart factory infrastructure system based on Kubernetes. The system includes a sensor node section; IoT section that acts as a gateway to control sensors and collect sensor data; and a main section that stores collected data, generates the anomaly detection model and supports real-time monitoring. All application programs were deployed to a container.

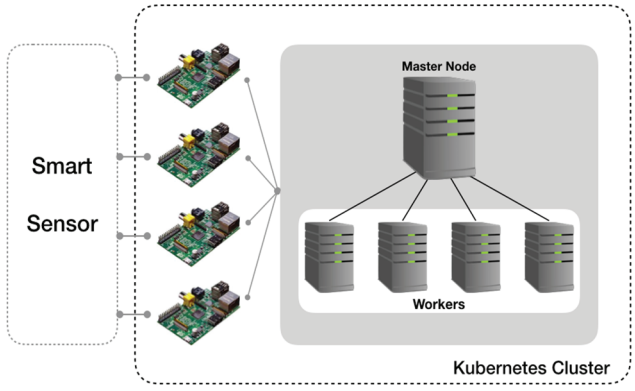

Fig. 1. Kubernetes cluster configuration.

Table 1. Raspberry Pi and node specification

	CPU	GPU	RAM
Raspberry Pi	Raspberry Pi 3 B+		
Master	Intel i5-4590 @ 3.30 GHz 4 Core	x	8 GB
Node1	Intel i5-7700 @ 3.60 GHz 4 Core	x	8 GB
Node2	Intel i5-4590 @ 3.30 GHz 4 Core	x	8 GB
Node3	Intel i5-4770 @ 3.40 GHz 8 Core	x	8 GB
Node4	Intel i7-6700 @ 3.40 GHz 8 Core	GeForce GTX1060 6 GB	16 GB

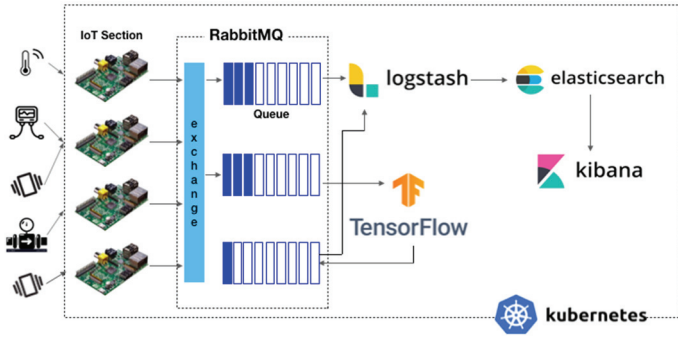

Fig. 2. Proposed system architecture.

3.1 Sensor Data Collection Process

The IoT section comprised several Raspberry Pis and delivered sensor data collected using message queuing telemetry transport, a messaging protocol based on Publish-Subscribe run over TCP/IP, to the main section [20]. The process that controls sensors and publishes messages to the main section through the MQTT was loaded to the container, the container was loaded only to the Raspberry Pi by the node selector, and the state was maintained by Kubernetes. Thus, even if the container was unloaded due to some process problem, the container can be re-created by the Kubernetes controller, preventing process disruption due to the underlying process problem. The container can also be deployed without disruption when changing the sensor control to modify the task.

The message published in the Raspberry Pi is delivered to RabbitMQ, a message broker based on advanced message queuing protocol [21]. The message delivered to RabbitMQ is not delivered to the queue immediately but delivered to the exchange and subsequently routed to the queue that complies with the control policy.

The proposed architecture used RabbitMQ that included three queues. The first queue contains a message delivered to Logstash to support real-time monitoring. The message delivered to Logstash is transferred to Elasticsearch via filtering and indexing. Logstash used a container image incorporating the "RabbitMQ" input plug-in. The second queue is read in the Python application program, and the delivered message is used as an input for anomaly detection. The anomaly detection result is then delivered to the third queue. RabbitMQ can guarantee the message time order because it stores messages received from the Producer in the queue, and hence can prevent message loss even if some clients were killed [22]. Fig. 3 shows the proposed data collection process.

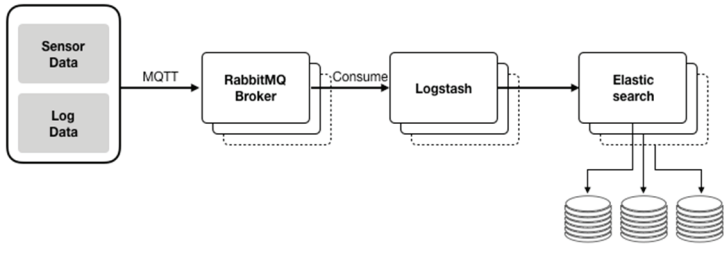

Fig. 3. Sensor data collection.

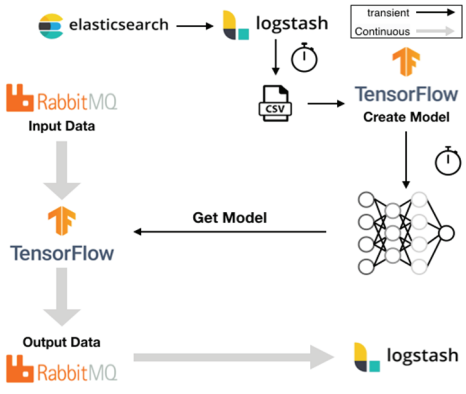

Fig. 4. Create model and real-time anomaly detection flow.

4. Experiments and Performance

This section covers experiments and performance evaluations on the anomaly detection used in the proposed smart factory infrastructure.

4.1 Datasets for Training

To assume a smart factory environment, we construct an experimental environment by installing pressure sensors, vibration sensors, and voltage sensors in a robot arm that repeats simple actions. And these data were collected from sensors through the sensor data collection process in Section 3.1.

The vibration sensors are used to analyze the operations of robot arm. If the sensor value is too low, it means that it does not operate normally, and if the measurement value is too high, it indicates that it is not stably fixed. The voltage sensor collects the voltage value supplied to the robot arm. lastly, the pressure sensor measures the pressure when the robot arm lifts the object. If the pressure is too low, the item cannot be lifted stably, and if the pressure is too high, the item can be damaged.

In the case of normal data, the sensor data of the well-functioning robot arm was collected every minute to build a dataset and in the case of abnormal data, abnormal data was collected through actions such as intentional voltage increase or loosen the screw.

The dataset consists of timestamp, vibration sensor, voltage sensor, and pressure sensor values. The total number of data used in the training is 59,403, with 49,552 normal data and 9,851 abnormal data. and test dataset consists of 17,478 data, with 15,580 normal data and 1,898 abnormal data. Table 2 shows abnormal cases of robot arm in training dataset.

Table 2. Abnormal cases of training dataset

	Abnormal case						
	Improperly operated	**Loose bolt/ Unstable**	**Power failure**	**Unstable power**	**Overvoltage**	**Weak grab**	**Excessive grab**
Number of data (%)	1,436 (2.417)	2,226 (3.747)	966 (1.626)	939 (1.581)	1,287 (2.166)	1,926 (3.241)	1,071 (1.802)

4.2 Training LSTM-AE

In the experiment dataset, we have three features, vibration sensor value, voltage sensor value, and pressure sensor value. Since each feature has a different domain of values, feature scaling was carried out through standardization. Our proposed model encoded these scaled features $x^{(i)}$ through LSTM encoder enc_θ which has two hidden layers, and decoder dec_θ which consists of LSTM, ReLU, FC layer makes reconstruction value. we use mean absolute error L_{MAE} between input data $x^{(i)}$ and reconstruction data $x'^{(i)}$ as a loss function.

$$\begin{aligned} e^{(i)} &= \left\| x^{(i)} - x'^{(i)} \right\| \\ x'^{(i)} &= dec_\theta(enc_\theta(x^{(i)}) \\ L_{MAE} &= \frac{\sum_{i=1}^{n} \left\| x^{(i)} - dec_\theta(enc_\theta(x^{(i)})) \right\|}{n} \end{aligned} \tag{1}$$

For stable training, we conducted the training through Adam optimizer. Other detailed hyperparameter information can be found in Table 3.

Table 3. Hyperparameters of LSTM-AE

Batch	Input/output size	Latent size	Window size	Hidden layer	Learning rate
128	3	2	3	2	0.001

4.3 Evaluation Metrics

Classifying anomaly based on reconstruction error e_r is frequently misclassified as anomaly when the value changes significantly even in normal data. To prevent this situation, our model used abnormal score introduced in [18] for classification. The abnormal score $\alpha^{(i)}$ with input data $x^{(i)}$ calculated through the covariance σ_e and mean value μ_e of the reconstruction error $e^{(i)}$.

$$\alpha^{(i)} = (e^{(i)} - \mu_e)^T {\sigma_e}^{-1} (e^{(i)} - \mu_e) \tag{2}$$

If $\alpha^{(i)}$ calculated from (2) is greater than threshold τ_α, it will be classified as abnormal data. Since we use F_β score derived from precision P and recall R as the evaluation metric, threshold τ_α value is determined by the threshold value that maximizes the F_β score [23].

$$\begin{aligned} F_\beta &= \frac{(1+\beta^2)PR}{\beta^2 P + R} \\ \tau_\alpha &= \tau_{list}[Argmax(F_\beta)] \end{aligned} \tag{3}$$

4.4 Performance Analysis

To evaluate the performance of our proposed model, we compare this model with SVM and OC-SVM model which frequently used in existing studies [15-17]. SVM is a representative supervised anomaly detection model and OC-SVM is an unsupervised anomaly detection model based on one-class learning [11,12]. Both training and testing used the same datasets mentioned in Section 4.1.

The classification of sensor data is relatively simple task because the value difference between abnormal data and normal data is extremely large. So, we did not compare models with high complexity which makes inference slow. The models used in the experiment consist of LSTM-AE, SVM, OC-SVM, and PCA-SVM, PCA-OCSVM. PCA-SVM is the model that compresses data through PCA (principal components analysis) into two dimensions, then trained from the compressed data [24].

The experiment compared three unsupervised anomaly detection models and two supervised anomaly detection models. Table 4 shows performance comparison results. Both accuracy and F1 performed best in LSTM-AE, the proposed model.

Fall-out is the rate at which abnormalities are misdiagnosed as normal. It is the most important part of the smart factory, because if fall-out is high, the factory's problem situation may not be recognized on time, resulting in large economic losses. Fall-out metrics also obtained the best results in LSTM-AE.

Table 4. Performance comparison

Anomaly detection	Model	Accuracy (%)	F1	*P*	*R*	Fall-out
Unsupervised	LSTM-AE	98.58	0.9915	0.9986	0.9844	0.0060
	PCA-OC-SVM	85.67	0.9195	0.9209	0.9181	0.6469
	OC-SVM	55.30	0.6654	1	0.4985	0
Supervised	PCA-SVM	95.25	0.9740	0.9494	1	0.4373
	SVM	93.07	0.9626	0.9279	1	0.6375

5. Conclusion

In this paper, we proposed Kubernetes based smart factory system to appropriately apply the various solutions proposed for facility management and maintenance cost reduction. This system included control of sensors by recipe, data collection, processing and storage, monitoring, anomaly detection model creation, and anomaly detection. Figs. 5 and 6 show our monitoring dashboard.

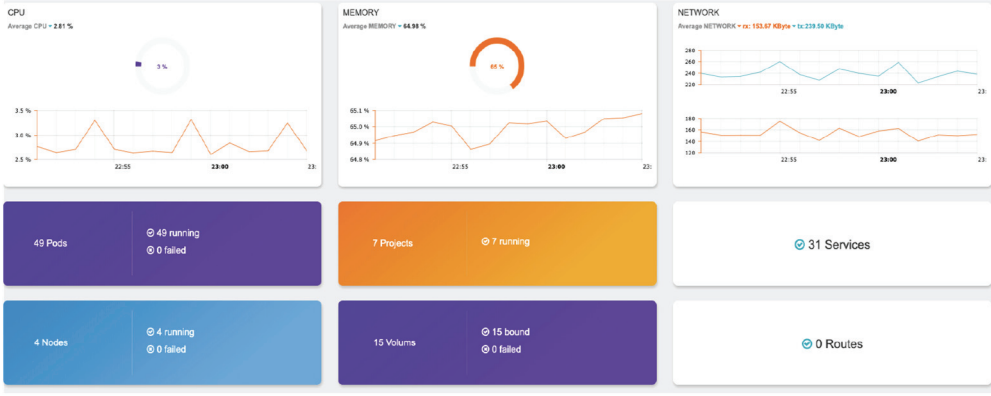

Fig. 5. Implemented monitoring dashboard of node status.

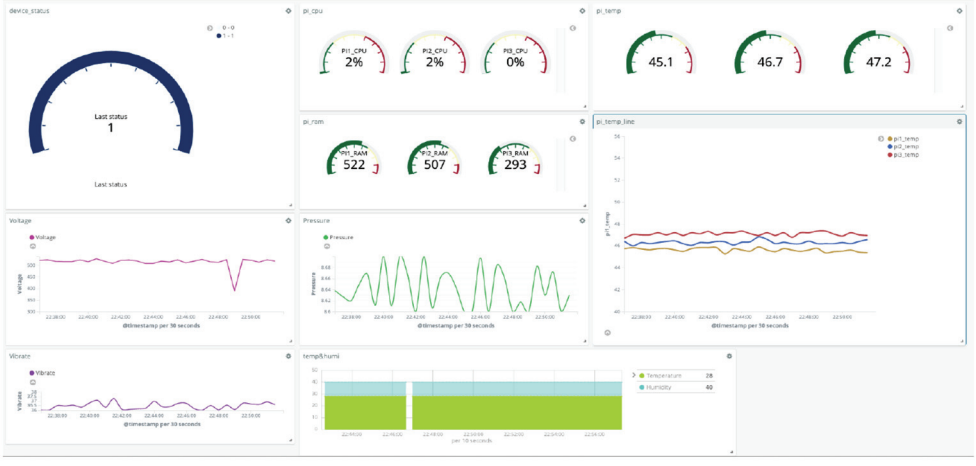

Fig. 6. Implemented monitoring dashboard of smart factory sensor status.

This study focused on the proper application of various solutions for the efficiency and stability of smart factory operation by building a system based on Kubernetes. We adopted various technologies such as Docker, Kubernetes, RabbitMQ, Elastic Stack and LSTM based encoder-decoder to implement the proposed smart factory system. We confirmed that through the implementation of the proposed system, it was possible to stabilize through system configuration maintenance, uninterrupted distribution of process recipes, create anomaly detection models containing the latest data. Additionally, we consider several characteristics of smart factory data. We apply LSTM-AE, an unsupervised anomaly detection model that can train time-series data efficiently, and we obtained higher accuracy and F1 score than SVM.

Acknowledgement

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (No. IITP-2021-2018-0-01419) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation) and results of a study on the “HPC Support” Project, supported by the Ministry of Science and ICT and NIPA.

References

[1] G. Buchi, M. Cugno, and R. Castagnoli, “Smart factory performance and Industry 4.0,” *Technological Forecasting and Social Change*, vol. 150, article no. 119790, 2020.

[2] Y. Zhang, M. Chen, S. Mao, L. Hu, and V. C. Leung, “CAP: community activity prediction based on big data analysis,” *IEEE Network*, vol. 28, no. 4, pp. 52-57, 2014.

[3] B. Wolf, P. Herzig, I. Behrens, A. Majumdar, and M. Ameling, “Data stream processing in factory automation,” in *Proceedings of 2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA)*, Bilbao, Spain, 2010, pp. 1-8.

[4] A. Maier, S. Schriegel, and O. Niggemann, “Big data and machine learning for the smart factory: solutions for condition monitoring, diagnosis and optimization,” in *Industrial Internet of Things*. Cham, Switzerland: Springer, 2017, pp. 473-485.

[5] H. S. Sim, “A study on the development of smart factory equipment engineering system and effects,” *Journal of the Korean Society for Precision Engineering*, vol. 36, no. 2, pp. 191-197, 2019.

[6] B. Kroll, D. Schaffranek, S. Schriegel, and O. Niggemann, “System modeling based on machine learning for anomaly detection and predictive maintenance in industrial plants,” in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, Barcelona, Spain, 2014, pp. 1-7.

[7] Kubernetes [Online]. Available: http://kubernetes.io/.

[8] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, et al., “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521-3526, 2017.

[9] R. Chalapathy and S. Chawla, “Deep learning for anomaly detection: a survey,” 2019 [Online]. Available: https://arxiv.org/abs/1901.03407.

[10] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

[11] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, “Support vector machines,” *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18-28, 1998.

[12] L. M. Manevitz and M. Yousef, "One-class SVMs for document classification," *Journal of Machine Learning Research*, vol. 2, pp. 139-154, 2001.
[13] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179-211, 1990.
[14] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings of the 23rd European Symposium on Artificial Neural Networks (ESANN)*, Bruges, Belgium, 2015.
[15] A. Jegorowa, J. Gorski, J. Kurek, and M. Kruk, "Initial study on the use of support vector machine (SVM) in tool condition monitoring in chipboard drilling," *European Journal of Wood and Wood Products*, vol. 77, no. 5, pp. 957-959, 2019.
[16] S. Hwang, J. Jeong, and Y. Kang, "SVM-RBM based predictive maintenance scheme for IoT-enabled smart factory," in *Proceedings of 2018 Thirteenth International Conference on Digital Information Management (ICDIM)*, Berlin, Germany, 2018, pp. 162-167.
[17] M. Wan, J. Li, K. Wang, and B. Wang, "Anomaly detection for industrial control operations with optimized ABC-SVM and weighted function code correlation analysis," *Journal of Ambient Intelligence and Humanized Computing*, 2020. https://doi.org/10.1007/s12652-020-02636-1
[18] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based encoder-decoder for multi-sensor anomaly detection," 2016 [Online]. Available: https://arxiv.org/abs/1607.00148.
[19] Elastic Stack [Online]. Available: https://www.elastic.co/.
[20] OASIS, "MQTT 3.1.1 specification," 2014 [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.
[21] X. Xiong and J. Fu, "Active status certificate publish and subscribe based on AMQP," in *Proceedings of 2011 International Conference on Computational and Information Sciences*, Chengdu, China, 2011, pp. 725-728.
[22] P. H. Tsai, H. J. Hong, A. C. Cheng, and C. H. Hsu, "Distributed analytics in fog computing platforms using TensorFlow and Kubernetes," in *Proceedings of 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Seoul, South Korea, 2017, pp. 145-150.
[23] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, article no. 1, 2015. https://doi.org/10.5121/ijdkp.2015.5201
[24] A. Mackiewicz and W. Ratajczak, "Principal components analysis (PCA)," *Computers & Geosciences*, vol. 19, no. 3, pp. 303-342, 1993.

Guik Jung **https://orcid.org/0000-0002-3840-7872**

He received the B.S. degree in Department of Software from Soongsil University, in 2019, where he is currently pursuing the master's degree with the Graduate School of Software. His research interests include cloud computing, distributed computing, natural language processing and lightweight deep learning.

Hyunsoo Ha **https://orcid.org/0000-0003-0687-9530**

He received the B.S. degree in Department of Software from Soongsil University, in 2019, where he is currently pursuing the master's degree with the Graduate School of Software Convergence. His current research interests include object detection and deep learning optimization.

Sangjun Lee **https://orcid.org/0000-0002-4251-7177**

He received his M.S. and B.S. degrees in the Department of Computer Engineering from Seoul National University, Seoul, Korea, in 1996 and 1998, respectively and Ph.D. in the School of Electrical Engineering and Computer Science, Seoul National University, Seoul, Korea in 2004. He is currently a professor of the School of Software, Soongsil University, Seoul, Korea. His current research interests include database systems, multimedia systems, and cloud computing.