High-performance IoT streaming data prediction system using Spark: a case study of air pollution

Ho-Yong Jin, Eun-Sung Jung & Duckki Lee

Neural Computing and Applications

ISSN 0941-0643

Neural Comput & Applic
DOI 10.1007/s00521-019-04678-9

Springer

Neural Computing and Applications
https://doi.org/10.1007/s00521-019-04678-9

S.I. : GREEN AND HUMAN INFORMATION TECHNOLOGY 2019

High-performance IoT streaming data prediction system using Spark: a case study of air pollution

Ho-Yong Jin[1] · Eun-Sung Jung[1] · Duckki Lee[2]

Received: 5 May 2019 / Accepted: 10 December 2019

Abstract
Internet-of-Things (IoT) devices are becoming prevalent, and some of them, such as sensors, generate continuous time-series data, i.e., streaming data. These IoT streaming data are one of Big Data sources, and they require careful consideration for efficient data processing and analysis. Deep learning is emerging as a solution to IoT streaming data analytics. However, there is a persistent problem in deep learning that it takes a long time to learn neural networks. In this paper, we propose a high-performance IoT streaming data prediction system to improve the learning speed and to predict in real time. We showed the efficacy of the system through a case study of air pollution. The experimental results show that the modified LSTM autoencoder model shows the best performance compared to a generic LSTM model. We noticed that achieving the best performance requires optimizing many parameters, including learning rate, epoch, memory cell size, input timestep size, and the number of features/predictors. In that regard, we show that the high-performance data learning/prediction frameworks (e.g., Spark, Dist-Keras, and Hadoop) are essential to rapidly fine-tune a model for training and testing before real deployment of the model as data accumulate.

Keywords Long Short-Term Memory (LSTM) · Distributed deep learning · Distributed Keras (Dist-Keras) · Apache Spark

1 Introduction

Internet-of-Things (IoT) devices are becoming prevalent, and some of them, such as sensors, generate continuous time-series data, i.e., streaming data. These IoT streaming data are one of Big Data sources [1], and they require careful consideration for efficient data processing and analysis. In this paper, we present how distributed systems can be used for such purposes.

The system uses a distributed deep learning framework called Distributed Keras (Dist-Keras) [2] and Long Short-Term Memory (LSTM) [3] units suitable for time-series data prediction. Dist-Keras is a distributed deep learning framework built on top of Apache Spark and Keras, with a focus on "state-of-the-art" distributed optimization algorithms. Most of the distributed optimizers Dist-Keras provides are based on data-parallel methods. A data-parallel method is a learning paradigm where multiple replicas of a single model are used to optimize a sole objective.

Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Spark Streaming provides a high-level abstraction called discretized stream or DStream.

DStream represents a continuous stream of data, either the input data stream received from the source or the processed data stream generated by transforming the input stream [4]. Internally, as shown in Fig. 1, DStream represents a series of RDDs, and each RDD in DStream contains a certain interval of data.

✉ Eun-Sung Jung
ejung@hongik.ac.kr

Ho-Yong Jin
ghdydcjsgk@mail.hongik.ac.kr

Duckki Lee
dlee@yc.ac.kr

[1] Department of Software and Communications Engineering, Hongik University, Sejong, South Korea

[2] Department of Smart Software, Yonam Institute of Technology, Jinju, South Korea

Published online: 17 December 2019

Fig. 1 DStream represents a series of RDDs

As a case study, we chose an air pollution prediction system. Air pollution in the world is increasing every year, and in some areas such as China and Korea, air pollution is much worse. As air pollution is getting worse, the anxiety of citizens increases as well as concerns about health. However, it is not easy for citizens to respond to air pollution because location, time, and various factors continuously change air pollution. We propose a high-performance IoT streaming data prediction system for the 8-h in-advance forecast of air pollution in Seoul, Korea. Air pollution information for real-time prediction is retrieved using OpenAPI of air-pollution-information-inquiry-service provided by AirKorea [5] and applied to real-time prediction model using Apache Spark's streaming library.

IoT devices are one of the major sources generating Big Data. Many machine learning techniques including state-of-the-art deep learning have been used, but there are still unexplored research issues [6]. In the context of time-series IoT data, many algorithms have been proposed and even commercialized in cloud computing. For example, Amazon AWS provides time-series prediction services such as ARIMA [7] and DeepAR [8]. Regarding air pollution, there have been few studies that dealt with air pollution prediction problem using deep learning techniques [9–11]. Our work differs from their work in the following aspects.

- We mainly focused on high-performance frameworks and tools to expedite time-series IoT data learning and inference.
- We made use of LSTM autoencoder technique for air pollution prediction similar to the previous studies [12, 13], but we evaluated many factors (e.g., epoch and memory cell size) of LSTM autoencoder best suited for the application. We even tried an LSTM variational autoencoder model, but the performance was too worse to report the model.
- We carefully chose datasets and merged datasets of different types (i.e., air pollution and wind datasets) to improve the accuracy of the learning model.

The paper is structured as follows. Section 2 presents the overall architecture and our case study application. Section 3 describes model learning. Sections 4 and 5 describe how to enhance the performance of the learning model through supplementing data and adopting different models, respectively. We conclude with future work in Sect. 6.

2 System architecture and application

The overall system architecture is shown in Fig. 2. Multiple independent servers are used to train the model. For distributed deep learning using Dist-Keras, Apache Hadoop/Spark [14, 4] was used to cluster three independent servers.

The datasets used to create the predictive model are air pollution information data provided by AirKorea. These data are from the past 6 years (2012–2017), including the values of O_3, NO_2, CO_2, SO_2, and PM10 in seven regions out of Korea's major regions every hour. We used 5-year data (2012–2016) to train and 1-year data (2017) to test. We checked the correlation between the data in Seoul and the data in the other six regions and decided which data to use as Predictors. Data in the three farthest regions from Seoul had little impact on data in Seoul. There was no difference in the root-mean-square error (RMSE) values between the case where all seven regional data were used as predictors and the case where only four regional data were used as predictors. Accordingly, the system predicts the PM10 value in Seoul after 8 h using the values of O_3, NO_2, CO_2, SO_2, and PM10 in four major regions (Seoul, Incheon, Daejeon, and Daegu). We chose the 8-h in-advance PM10 value since the near-future PM10 values are almost the same as the present value. The number of predictors is 20, the size of the training dataset is 30,269, and the size of the test dataset is 6527.

As a method of correcting the erroneous data, we eliminated the corresponding rows. These data are empty due to errors in the measuring equipment. Even if we tried to impute data with average values of previous data, the

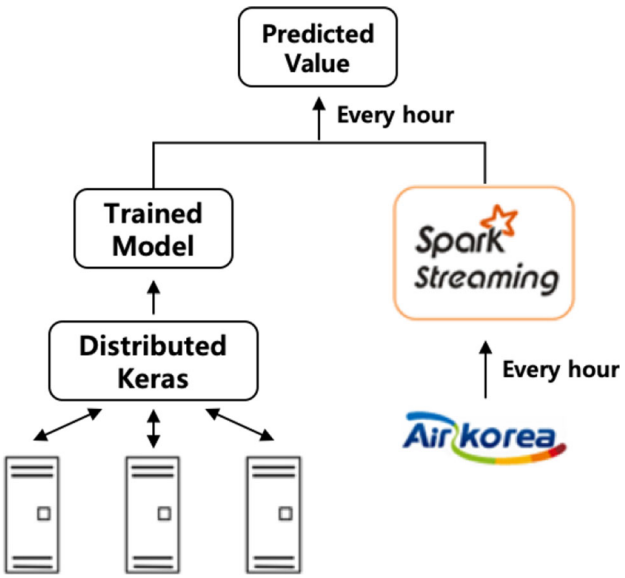

Fig. 2 System architecture

results were little different from the results of the simple elimination of missing or erroneous data. In addition, the values of fine dust (PM10) usually range between 0 and 200, but in some cases, high PM10 between 300 and 900 were measured. These high values were judged to have a detrimental effect on model learning, and we consider these values as outliers. We thus changed more than 300 values into 300 for the training data. Korea classifies levels of fine dust into four stages such as good (0–30 ug/m^3), normal (31–80 ug/m^3), poor (81–150 ug/m^3), and very poor (more than 151 ug/m^3). Predicting the correct value for actual values above 150 ug/m^3 is not significant.

We converted the historical data from AirKorea for model learning into the appropriate format for Spark. The dataset preparation process is as follows.

1. Use the Python Library Pandas to read the last 6 years of csv format data.
2. Correct the erroneous data.
3. Adjust too high PM10 data (300 ug/m^3 or more) to 300 ug/m^3 for training data.
4. Create the DataFrame using Spark SQL.
5. Use Spark ML's VectorAsembler to create vector features combining all predictors. VectorAssembler is a transformer that combines a given list of columns into a single vector column.
6. Scale using Apache Spark ML's StandardScaler. StandardScaler transforms a dataset of vector rows, normalizing each feature to have unit standard deviation and zero mean.
7. Reshaping features to the input of the model.
8. After finishing the process, save the training dataset and the test dataset in HDFS in parquet format.

3 Model learning

We describe how we conducted model learning in detail.

3.1 Neural networks

IoT streaming data are time-series data. We use a Long Short-Term Memory (LSTM) model, which is one of recurrent neural network (RNN) models for time-series data analytics.

Recurrent neural network (RNN) is an artificial neural network specialized for repetitive and sequential data learning. As shown in Fig. 3, RNN uses the internal circulation structure to reflect the weight obtained from prior learning into the current learning. This enables the link between current learning and prior learning. However, there is a problem that as the learning progresses longer,

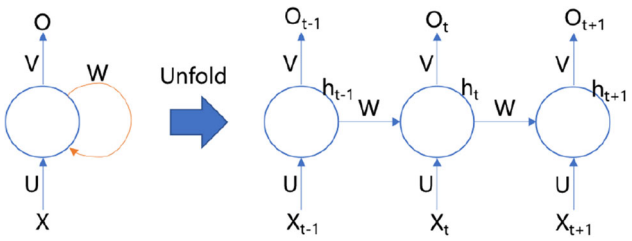

Fig. 3 Recurrent neural network (RNN)

the weight value of the past is canceled. LSTM is what came to supplement this.

As shown in Fig. 4, the LSTM has a structure in which the cell state is added to the hidden state of the RNN. The cell state determines whether or not to keep the weight from prior learning.

LSTM units have forgotten gates, input gates, and output gates. When the activation function of the LSTM unit is a sigmoid function, the forget gate receives h_{t-1} and x_t and outputs the value to which the sigmoid function is applied. If the value is 0, the previous learning is forgotten. The input gate receives x_t and h_{t-1} and outputs the value by applying the sigmoid function and the hyperbolic tangent (tanh) function as a Hadamard product. The value of the input gate is used to create a new cell state ct by updating the past cell state C_{t-1}. The output gate receives h_{t-1} and x_t and outputs a value obtained by applying a sigmoid function and a value obtained by applying a hyperbolic tangent (tanh) function to a new cell state C_t. In this equation, W means a weight matrix, and b means a bias vector.

We stacked several LSTM layers to form a model. As we stack up LSTM layers one by one, we could get the lowest RMSE value when 5 LSTM layers were stacked. When 6 or more LSTM layers are stacked, we observed that model learning does not perform well. Usually, we may think that the more the time steps, the better the performance, but it is not. As a result of learning the model by changing the input data into various shapes, we could achieve the lowest RMSE value when the number of time steps was 1.

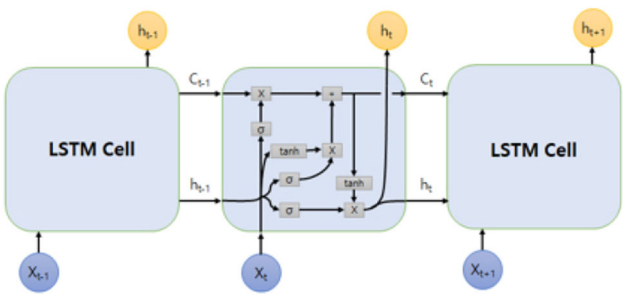

Fig. 4 Long Short-Term Memory (LSTM)

The ReLu function was used as an activation function in this system. In Keras, the default activation function of LSTM units is a tanh function, but learning by tanh function did not perform well when constructing the model by stacking LSTM layers.

The size of the memory cell can affect the performance of the LSTM units. The larger the size of a memory cell gets, the better the model's performance gets because a larger size of a memory cell can remember more past learning. However, beyond a certain point, the model's performance is degraded. Because the optimal memory cell size varies depending on problems, we should search for it experimentally. As shown in Fig. 5, several tests found the optimal memory cell size of 400. We used a model with five LSTM layers to find the proper memory cell size. Epoch, time steps, and data dimensions were fixed at 300, 1, and 20, respectively. Epoch is a parameter that indicates how many times the model will learn the training dataset. In general, if you learn the prepared train dataset once in the model, the model does not learn enough about the training dataset. If the epoch is set to a large value, there will be overfitting since the model will learn too much about the training dataset. As shown in Fig. 6, we found that the proper value of the epoch is 300. For this search of the proper epoch value, we used a model with five LSTM layers where memory cell size, time steps, and the number of features were fixed at 400, 1, and 20, respectively.

3.2 Distributed deep learning

We have tested various LSTM layers, sizes of the memory cells, and epochs to find the appropriate model. Deep learning may take a long time when big data is used for training and testing. We adopted the Spark parallel processing platform to overcome this problem. Our system used Apache Hadoop/Spark to cluster three nodes (one master node, two working nodes) and then ran the

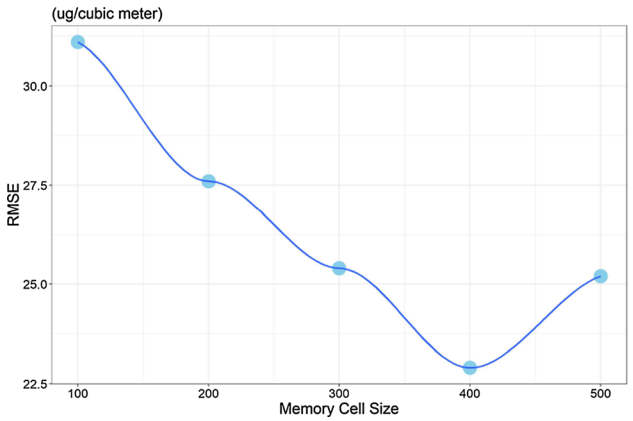

Fig. 5 Memory cell size versus RMSE

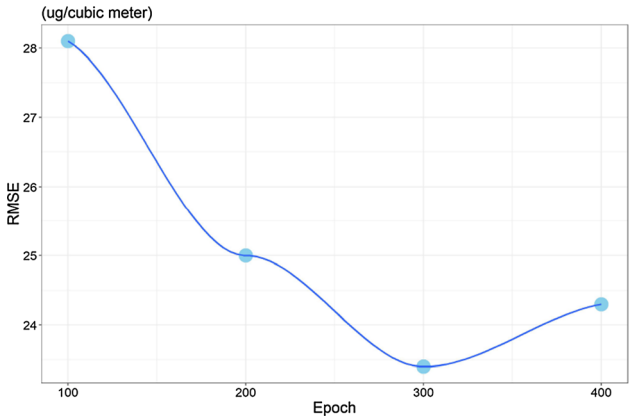

Fig. 6 Epoch versus RMSE

Distributed Keras (Dist-Keras). Each worker node has ten cores and 85 GB of memory. Figure 7 shows that Dist-Keras runs through a total of eight executors with four executors per worker node.

We conducted experiments to measure the training time and RMSE as the number of executors varies in the range of 2, 4, 8, and 16. The number of cores and memory per executor are fixed at 1 and 6 GB, respectively. Figure 8 shows that as the number of executors increases, the training time decreases steadily, but not linearly. Figure 9 shows that the RMSE values are almost the same even if the number of executors changes, which means the Dist-Keras works consistently with the different number of worker nodes.

Using the trained model, we predict the 8-h in-advance air pollution (PM10) in Seoul. To make these predictions every hour, we used Apache Spark's Streaming library and AirKorea's OpenAPI.

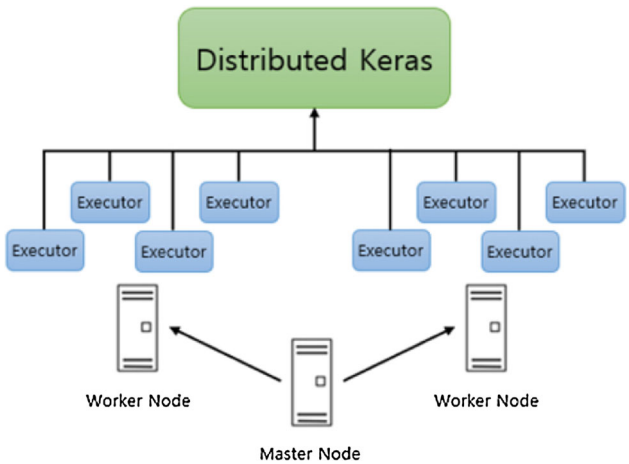

Fig. 7 Distributed Keras

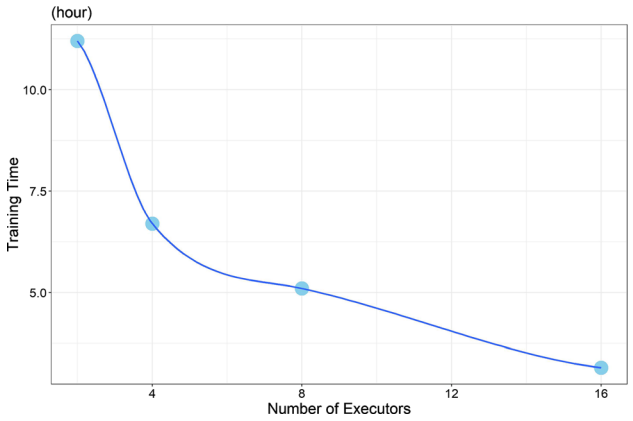

Fig. 8 Number of executors versus training time

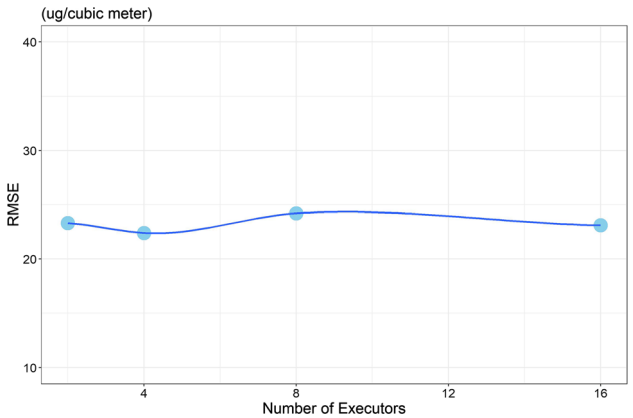

Fig. 9 Number of executors versus RMSE

4 Enhancing accuracy through high-dimensional data

In this section, we describe our approaches to enhance the accuracy of our model by adding more data points and data types.

4.1 Adding more domestic data points

In Sect. 3, we used five air pollution indicators (i.e., O_3, NO_2, CO_2, SO_2, and PM10) in seven major regions in South Korea, which accordingly leads to 35 features. Here, we use four places (East, West, South, and North) of a region to see whether more data will help improve the accuracy. We also treated the data rows with missing values differently. Instead of eliminating the entire rows, we substituted zeros for the missing values. The total number of features increases from 35 to 140, and we could see that the RMSE decreases from 22.9 to 22.1 (3.5% improvement), as shown in Fig. 10.

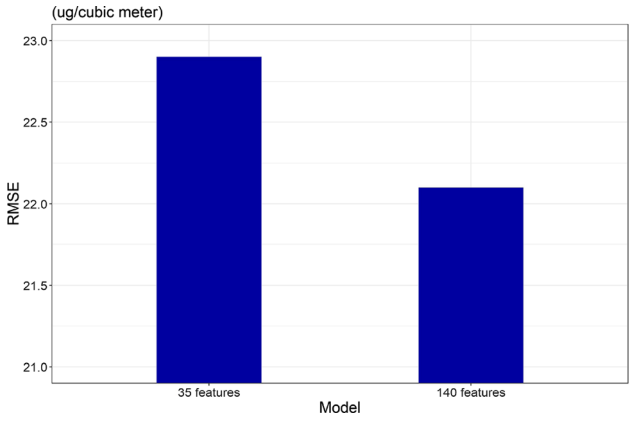

Fig. 10 RMSE of 35/140 feature models

4.2 Adding more data types (wind data in addition to air pollution indicator)

We believe that air pollution in South Korea is affected by that in China to some extent, which varies depending on conducted studies. We added PM2.5 data in Beijing and Shanghai in China, which was publicly available at UCI machine learning repository [15]. More importantly, we also added the direction and speed of winds on the sea between South Korea and China because they are the major factors making air pollution in China affect South Korea. We obtained the wind data at Bakryong island located in the most west of Korea through the Korea open weather data portal site. Finally, the total number of features becomes 144. The China weather datasets cover the years from 2012 through 2015, and we used 80% and 20% of the datasets for training and testing, respectively. The learned model using these extended datasets without wind data shows the RMSE of 20.6, while the learned model using the datasets with wind data shows the better RMSE of 19.6.

4.3 Prediction of all seven major regions

We modify the model we used earlier to forecast not only Seoul but all seven major cities in Korea. The existing model used supervised learning with Seoul air pollution data after 8 h as labeled data. The modified model uses air pollution data from seven regions as labeled data. We build a modified model using air pollution data after 1 to 8 h in 7 regions. In the previous model, model learning to lower the MSE only for prediction data after 8 h, the last prediction step, was better than model learning to reduce the overall MSE value for all predictions after 1 to 8 h. However, regarding the modified model that predicts all seven regions, the learning method that reduces MSE for all steps performed better. As in the previous model, the cell size and learning rate were set to 100 and 0.001, respectively.

Finally, the RMSE values of each city are 18.8, 24.1, 23.5, 21.9, 20.0, 20.9, and 24.4 for Seoul, Gwangju, Daegu, Daejeon, Busan, Incheon, and Ulsan, respectively as shown in Fig. 11.

5 Enhancing accuracy through different frameworks and models

5.1 Switching from a classification problem to a regression problem using TensorFlow on Spark framework

In Sect. 3, we used the Dist-Keras framework to model learning. One of the disadvantages of the Dist-Keras framework is that we cannot fine-tune the model. In our case, our Dist-Keras model solves the problem as a classification problem because the default Dist-Keras model makes use of internally implemented classification-based prediction techniques. In such a method, the model should learn too many classes inefficiently. Moreover, the classification-based air pollution prediction outputs integer-numbered predictions, which may worsen RMSE, whereas the regression-based air pollution can predict as real numbers. We adopted TensorFlow on Spark to enhance the performance while keeping the learning time as fast as the previous Dist-Keras framework, and we observed faster learning time with TensorFlow on Spark.

5.2 Prediction using LSTM autoencoder

We also tried a different model from only LSTM stacked models in Sect. 3. The LSTM autoencoder model is the model utilizing LSTM units as the encoder and decoder of autoencoder. In general, autoencoder gets input data x, encodes into lower-dimensional data, and finally restores the input data x. In our case, we configured the LSTM autoencoder such that the model gets five timestep sequence data and predicts the next eight timestep sequence data. As shown in Fig. 12, we first train the encoder using five timestep sequence data. We then train the decoder to predict the next eight timestep sequence data. The prediction of the next eight timestep sequence data as follows.

1. Predict the first timestep data with zero vector input.
2. Predict the second timestep data with the first prediction.
3. Repeat the previous steps until we predict the last eighth timestep data.

The general autoencoder is unsupervised learning in a sense that only input data are used for training. The LSTM autoencoder, however, is not unsupervised learning because it should label for eight timestep data prediction. Our LSTM autoencoder model shows the better performance, RMSE of 18.5, than the previous LSTM model as in Fig. 13

We also measured how many input timestep data perform well by varying the input timesteps in the set of {1, 5, 8}, as shown in Fig. 14. Surprisingly, the eight timesteps input data shows the worse performance than the five timesteps input data, which is the best among the three cases.

We next compared two variations of our LSTM autoencoder model: (1) 8 timestep outputs where the eighth output is the predicted value, and (2) 1 timestep output for 8-h in-advance prediction. The first variation shows the RMSE of 18.5, and the second variation shows the RMSE of 19.08. Thus, in our case, the subsequent learning of eight timestep outputs leads to better performance than the direct learning of the target timestep output.

Lastly, we varied the size of the hidden cell size of the LSTM autoencoder model. The results in Fig. 15 show that 100 hidden cells (latent variables) is the best among 50, 75, and 100 hidden cells.

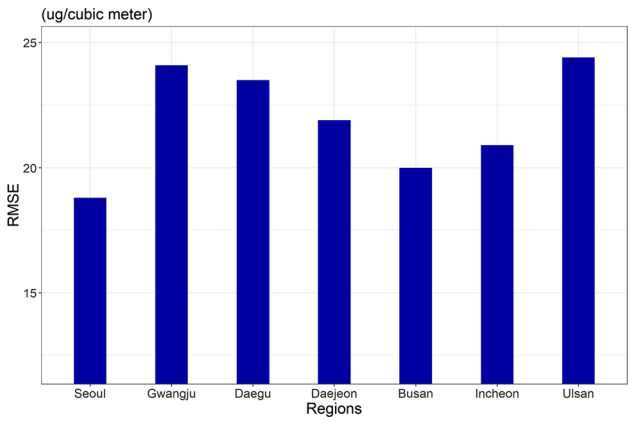

Fig. 11 Prediction of all seven major regions

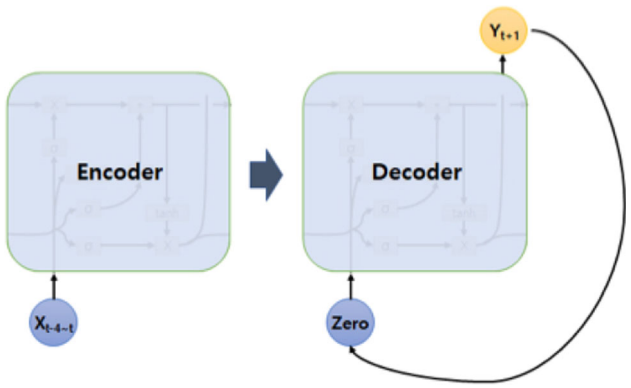

Fig. 12 LSTM autoencoder

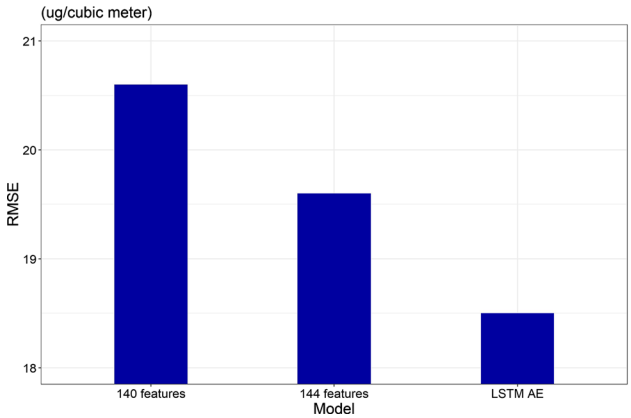

Fig. 13 RMSE of various models with China data

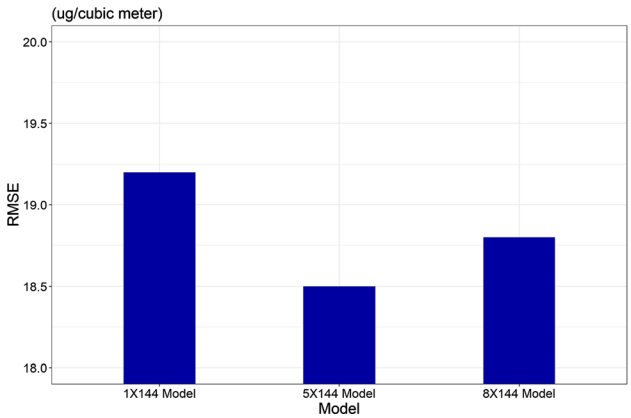

Fig. 14 Input timesteps versus RMSE

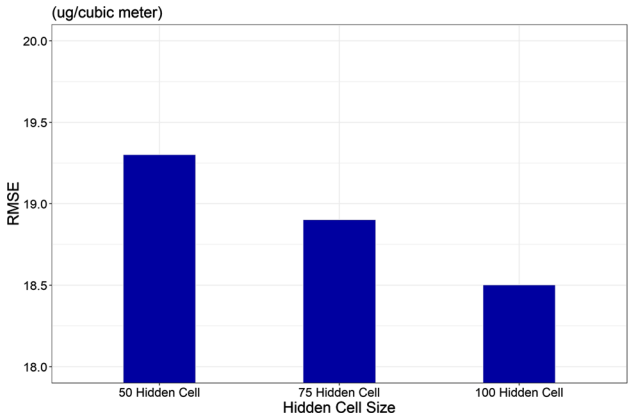

Fig. 15 Hidden cell size versus RMSE

6 Conclusions and future work

We propose a high-performance IoT streaming data prediction system to improve the learning speed and to predict in real-time. We showed the efficacy of the system through a case study of air pollution. The experimental results show that the modified LSTM autoencoder model shows the best performance compared to a generic LSTM model. We noticed that achieving the best performance requires optimizing many parameters including learning rate, epoch, memory cell size, input timestep size, and the number of features/predictors. In that regard, we show that the high-performance data learning/prediction frameworks (e.g., Spark, Dist-Keras, and Hadoop) are essential to rapidly fine-tune a model for training and testing before real deployment of the model as data accumulates. In the future, we will study how IoT streaming data should be delivered to the data processing and analytics platform and how the platform can handle the data adaptively according to the amount and the speed of the data.

Acknowledgements This work was supported by Basic Science Research Program through the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (NRF-2017R1D1A1B03033632).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Chen F, Deng P, Wan J, Zhang D, Vasilakos AV, Rong X (2015) Data mining for the internet of things: literature review and challenges. Int J Distrib Sens Netw 11(8):431047. https://doi.org/10.1155/2015/431047
2. Distributed Keras. https://joerihermans.com/work/distributed-keras/. Accessed 16 Dec 2019
3. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
4. Apache Spark™—Unified Analytics Engine for Big Data (2019). https://spark.apache.org/. Accessed 16 Dec 2019
5. Air korea. https://www.airkorea.or.kr/index. Accessed 16 Dec 2019
6. Marjani M, Nasaruddin F, Gani A, Karim A, Hashem IAT, Siddiqa A, Yaqoob I (2017) Big IoT data analytics: architecture, opportunities, and open research challenges. IEEE Access 5:5247–5261. https://doi.org/10.1109/ACCESS.2017.2689040
7. Brockwell PJ, Davis RA (2016) Modeling and forecasting with ARMA Processes. In: Brockwell PJ, Davis RA (eds) Introduction to time series and forecasting, Springer texts in statistics. Springer, Cham, pp 121–155. https://doi.org/10.1007/978-3-319-29854-2_5
8. Salinas D, Flunkert V, Gasthaus J (2017) DeepAR: probabilistic forecasting with autoregressive recurrent networks. arXiv:1704.04110 [cs, stat]
9. Bui TC, Le VD, Cha SK (2018) A deep learning approach for forecasting air pollution in South Korea using LSTM. arXiv:1804.07891 [cs, stat]
10. Li X, Peng L, Hu Y, Shao J, Chi T (2016) Deep learning architecture for air quality predictions. Environ Sci Pollut Res 23(22):22408–22417. https://doi.org/10.1007/s11356-016-7812-9
11. Reddy VN, Mohanty S (2017) Deep air: forecasting air pollution in Beijing, China. https://www.ischool.berkeley.edu/sites/default/

files/sproject_attachments/deep-air-forecasting_final.pdf. Accessed 16 Dec 2019
12. Li X, Peng L, Yao X, Cui S, Hu Y, You C, Chi T (2017) Long short-term memory neural network for air pollutant concentration predictions: method development and evaluation. Environ Pollut 231:997–1004. https://doi.org/10.1016/j.envpol.2017.08.114
13. Srivastava N, Mansimov E, Salakhutdinov R (2015) Unsupervised learning of video representations using LSTMs. In: Proceedings of the 32nd international conference on international conference on machine learning, vol 37, ICML'15, JMLR.org. Event-place, Lille, France, pp 843–852. http://dl.acm.org/citation.cfm?id=3045118.3045209. Accessed 16 Dec 2019
14. Apache Hadoop. http://hadoop.apache.org/. Accessed 16 Dec 2019
15. UCI Machine Learning Repository: PM2.5 Data of Five Chinese Cities Data Set. https://archive.ics.uci.edu/ml/datasets/PM2.5+Data+of+Five+Chinese+Cities. Accessed 16 Dec 2019