High-Performance IoT Streaming Data Prediction System using Spark: A Case Study of Air Pollution

Ho-Yong Jin, Eun-Sung Jung
Department of Software and Communications Engineering
Hongik University
Sejong, South Korea
hoyong8532@naver.com, ejung@hongik.ac.kr

Duckki Lee
Department of Smart Software
Yonam Institute of Technology
South Korea
dlee@yc.ac.kr

***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.**

Keywords—Apache Spark, Long Short-Term Memory (LSTM), Distributed Deep Learning, Distributed Keras (Dist-Keras).

I. 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 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) and Long Short-Term Memory (LSTM) 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 single 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.

Figure 1 DStream is represented as a series of RDDs.

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

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 the citizen increases as well as concerns about health. However, it is not easy for citizens to respond to air pollution because air pollution is constantly changed by location, time and various factors. We propose a high-performance IoT streaming data prediction system for air pollution 8 hours in advance forecast in Seoul, Korea. Air pollution information for real-time prediction is retrieved using OpenAPI of air pollution information inquiry service provided by AirKorea and applied to real-time prediction model using Apache Spark's streaming library.

The paper is structured as follows. Section II presents the overall architecture and our case study application, Section III describes model learning. We conclude with future work in Section IV.

II. 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, servers must be clustered and Apache Hadoop [2] was used to cluster three independent servers.

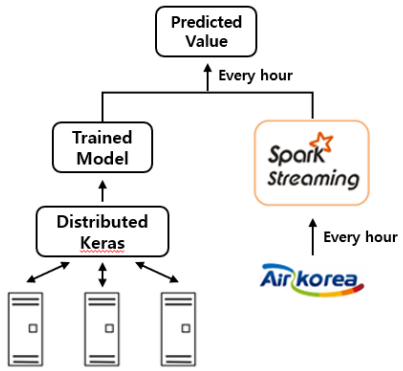

Figure 2 System Architecture.

The data sets used to create the predictive model are air pollution information data provided by AirKorea. This data is from the past six years (2012 ~ 2017), including the values of O3, NO2, CO2, SO2 and PM10 in seven of Korea's major regions every hour. Train data was used for five years (2012 ~ 2016) and test data was used for one

year (2017). 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 have had little impact on data in Seoul. In fact, 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 four regional data were used as Predictors. Therefore, the system predicts the PM10 value of Seoul after 8 hours using the values of O3, NO2, CO2, SO2 and PM10 of four major regions (Seoul, Incheon, Daejeon and Daegu) . The number of Predictors is 20, the size of train data set is 30269, and the size of test data set is 6527.

As a method of correcting the error data, we used the method of deleting the corresponding row. This data is empty due to errors in the measuring equipment. If you put the error data into the average value of the previous time, it is likely that the better learning will be done, but there was not a lot of data in succession. In this case, if we change the error data to the average value of the previous data, it would have a bad influence on learning. The values of fine dust (PM10) usually range between 0 and 200, but suddenly, high values between 300 and 900 have been measured. These high values were judged to have a detrimental effect on model learning, so for the train data, more than 300 values were changed to 300. Korea's classification of fine dust is classified into 4 stages 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.

Historical data from Airkorea for model learning are converted into the appropriate format for Spark. The dataset preparation process is as follows.

1) Use the Python Library Pandas to read the last six years of csv format data.

2) Correct the error data.

3) Adjust too high PM10 data (300 ug/m^3 or more) to 300 ug/m^3 for train 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 / or zero mean.

7) Reshaping features to the input shape of the model to be learned.

8) After finishing the process, save the train dataset and the test dataset in HDFS in parquet format.

III. MODEL LEARNING

A. Neural Network

IoT streaming data are time series data. We use Long Short-Term Memory (LSTM) model, which is one of Recurrent Neural Network (RNN) models, for time series data analytics. As shown in Fig. 3, 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 past learning.

LSTM units have forget gate, input gate, and output gate. 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 applied value using the sigmoid function and the hyperbolic tangent (tanh) function as Hadamard product. The value of the input gate is used to create a new cell state c_t 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 weight matrix and b means bias vector.

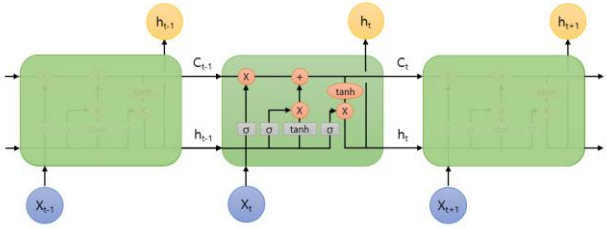

Figure 3 Long Short-Term Memory (LSTM)

We stacked several LSTM layers to form a model. As a result of obtaining RMSE by stacking up LSTM layers one by one, the lowest RMSE value was obtained when 5 LSTM layers were stacked. It is confirmed that model learning is not performed well when 6 or more LSTM layers are stacked. 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, the lowest RMSE value was achieved when the time steps were 1.

The relu function was used as an active function in this system. In Keras, the basic activation function of LSTM units is defined as tanh function but learning by tanh function is not done well when constructing model by stacking LSTM layer.

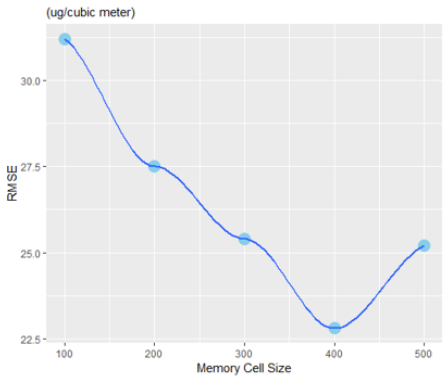

Figure 4 Memory cell size vs RMSE

The size of the memory cell can affect the performance of the LSTM units. The larger the size of a memory cell, the better the model's performance because it can remember more about past learning. However, crossing certain sections can interfere with model learning. Because the optimal memory cell size varies from case to

case, it should be found through several tests. As shown in Fig. 4, several tests found the optimal memory cell size of 400. We used a model with five LSTM layers to test the 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 train dataset will be learned in the model. In general, if you learn the prepared train dataset once in the model, the model does not learn enough about the train dataset. However, if epoch is set to a large value, there will be overfitting as the model will learn too much about the train dataset. As shown in Fig. 5, it was judged that it is best to set the epoch to 300. We used a model with five LSTM layers to test the epoch. Memory cell size, time steps and data measures were fixed at 400, 1, and 20, respectively.

B. Distributed Deep Learning

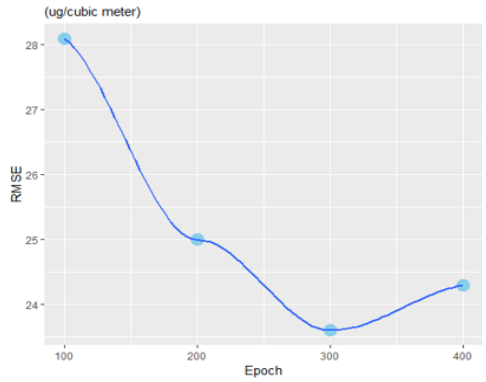

Figure 5 Epoch vs RMSE

We have tested various Long Short-Term Memory (LSTM) layers, sizes of the memory cells, and epochs to find the optimal model. Deep Learning has a chronic problem that it takes a long time to learn. We adopted Spark parallel processing platform to overcome this problem. Our system used Apache Hadoop to cluster three nodes (one master node, two working nodes) and then run the Distributed Keras (Dist-Keras). Each worker node has ten cores and 85 GB of memory.

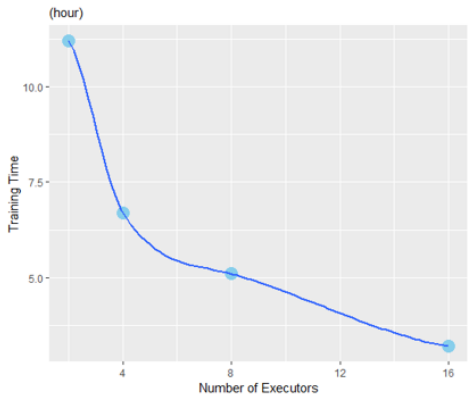

Figure 6 Number of executors vs training time

We conducted experiments to measure the training time and RMSE as the number of the executors changes in the range of 2, 4, 8, and 16. The number of cores per executor and memory per executor are fixed at 1 and 6G respectively. Fig. 6 shows that as the number of executors increases, the training time decreases steadily, but not linearly. Fig. 10 shows that the RMSE values are almost the same even if the number of executors changes, which means the Dist-Keras works consistently with varying number of worker nodes.

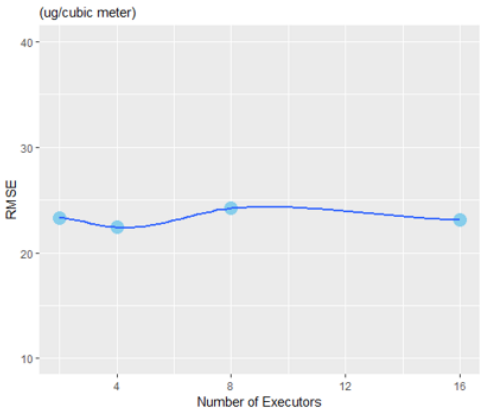

Figure 7 Number of executors vs RMSE

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

IV. 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 also showed the efficacy of the system through a case study of air pollution. This work is a preliminary study for massive IoT streaming data processing and analytics in real-time. In 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.

V. ACKNOWLEDGMENT

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).

REFERENCES

[1] https://spark.apache.org/

[2] https://hadoop.apache.org/

[3] Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long shortterm memory". Neural Computation. 9 (8): 1735–1780. doi:10.1162/neco.1997.9.8.1735. PMID 9377276.

[4] Reddy, Vikram Narasimha and Shrestha Mohanty. “Deep Air : Forecasting Air Pollution in Beijing , China.” (2017).

[5] T.-C. Bui, V.-D. Le, and S.-K. Cha, “A Deep Learning Approach for Forecasting Air Pollution in South Korea Using LSTM,” arXiv:1804.07891, Apr. 2018.