Intelligent Automation And Soft Computing, 2019
Copyright © 2019, TSI® Press
Vol. 25, no. 2, 343–350
https://doi.org/10.31209/2019.100000095

Predicting Concentration of PM10 Using Optimal Parameters of Deep Neural Network

Byoung-Doo Oh[a,b], Hye-Jeong Song[a,b], Jong-Dae Kim[a,b], Chan-Young Park[a,b], Yu-Seop Kim[a,b]

[a]School of Software, Hallym University, Chuncheon, Korea
[b]Bio-IT Research Center, Hallym University, Chuncheon, Korea

ABSTRACT
Accurate prediction of fine dust (PM10) concentration is currently recognized as an important problem in East Asia. In this paper, we try to predict the concentration of PM10 using Deep Neural Network (DNN). Meteorological factors, yellow dust (sand), fog, and PM10 are used as input data. We test two cases. The first case predicts the concentration of PM10 on the next day using the day's weather forecast data. The second case predicts the concentration of PM10 on the next day using the previous day's data. Based on this, we compare the various performance results from the DNN model. In the experiments, we get about 76% of accuracy with the proposed system.

KEY WORDS: Deep Neural Network (DNN); PM10; classification

1 INTRODUCTION

AS the use of fossil fuels continues to increase, there is a growing interest in air pollution both domestically and internationally. As a result, fine dust (Particulate Matter 10, PM10) is also increasing rapidly. Particularly, it is known that PM10 has a very negative influence on health (Kim et al., 2015; Jang, An-Soo, 2014). Therefore, accurate prediction of PM10 concentration is recognized as an important problem. This problem can be solved using data-based approach.

One of the data-based approaches is Artificial Neural Network (ANN). Recently, ANN solved 2 problems. First, the overfitting problem that was criticized from the past has been solved (Salakhutdinov et al., 2007; Dahl et al., 2013). Second, the development of the GPU (Graphic Processing Unit) has reduced the burden of enormous computation.

ANN is used in many fields due to its good performance. And also, ANN is became more advanced to Deep Neural Network (DNN). When we use DNN, the used data is important. And the data pre-processing method also is very important (Hinton et al., 2006) because it can affect the performance of the model. And when designing a model, you can get different performance through setting different parameters of the model.

Recently, many studies have been conducted on the prediction of solving problems through machine learning in the field of environment (Jiang et al., 2016). Especially, ANN or DNN have been widely used in many studies (Sakar et al., 2011). At that time, the meteorological factors were used to as input data because they have been known to be very important to affect atmospheric diffusion, migration, and densification of air pollutants (Khodarahmi et al., 2016; Shin et al., 2007). In Dedovic, et al. (2016), they used to meteorological factors, weekdays and hours data in Bosnia and Herzegovina as input data. When using day and time data, the data were used after pre-processing the data with sinusoidal and cosinusoidal variables. In this study, they tried two cases of examples using DNN. In the first case, they used 10 neurons with log-sigmoid function in input layer, 15 neurons with tan-sigmoid function in hidden layer, and 1 neuron with linear function in output layer. In the second case, they used 11 neurons in the input and hidden layer. At this time, they evaluated two cases using Normalized Root Mean Square Error (NRMSE). In the first case, NRMSE is 0.39. And in the second case, NRMSE is 0.03.

So, one of the studies using meteorological factors and DNN, the cosine-similarity for data pre-

CONTACT Yu-Seop Kim yskim0@hallym.ac.kr

processing was adopted, Hur et al. (2016). In this study, they used to cosine-similarity for data pre-processing. And when using DNN, they used a logistic function as an activation function in the hidden layer. However, the accuracy is as low as 59% ~ 69%. Therefore, it is understood that the method of data pre-processing and setting parameters are very important when using DNN as mentioned above.

In this paper, we chose an uncomplicated method of data pre-processing and compared various performance with different parameters of the DNN model in order to find an optimal condition. So, we used One-Hot Encoding as a data pre-processing method. And we have tested two cases. The first is an experiment that predicts the concentration of PM10 on the next day by using the weather forecast data of that day. In this case, parameters of model showed the best performance when it is set as follows. Tanh is used as an activation function, He_uniform is used as an initialization method, 2 hidden layers as number of hidden layers, dropout and batch normalization are used. The accuracy was 75.974 %, the f1-measure was 0.76, and the loss was 0.63. The second is an experiment that predicts the concentration of PM10 on the next day using the previous day's data. In this case, data of PM10 of the previous day was also used as input data. In this case, parameters of model showed the best performance when it is set as follows. ReLU is used as an activation function, 5 hidden layers as number of hidden layers, and batch normalization are. The accuracy was 75.8117 %, the f1-measure was 0.75, and the loss was 0.65.

2 DATA

IN this paper, we used meteorological factors (weather type, temperature, humidity, wind speed, wind direction), yellow dust (sand), fog, and PM10 as features. Data was collected from January 01, 2009 to August 31, 2016. This is the weather data for Seoul, Republic of Korea. The data of the meteorological factors, the sand, the fog, and PM10 corresponding to the feature were collected by the Korea Meteorological Administration (http://www.kma.go.kr/index.jsp).

The weather types are divided into four categories: Sunny, Cloudy, Rainy, and Snowy. Sand and fog are used as a feature of each. It is divided into two categories: True and False. The wind direction was divided into 16 categories such as North, North-East-North, North-East. The temperature, humidity, and wind speed were used as numerical data. PM10 was classified into four categories. It designated by the Korea Ministry of Environment (https://www.me.go.kr/home/web/main.do). The criteria are as follows: 0 ~ 30 $\mu g/m3$ is Low, 31 ~ 80 $\mu g/m3$ is Normal, 81 ~ 150 $\mu g/m3$ is High, and 150 $\mu g/m3$ or more is Danger. Each feature is described in the following [Table 1]. The data set created with the features and classified in this way is shown in [Table 2]. The distribution of the PM10 in the dataset according to the four categories is as shown [Table 3].

Table 1. Value description for each feature

Feature	Values
Weather type	Sunny, Overcast, Rainy, Snowy
Sand	True, False
Fog	True, False
Temperature	*Numerical data*
Humidity	*Numerical data*
Wind speed	*Numerical data*
Wind direction	North, North-East, North-East-North, …
PM10 (Input)	*Numerical data*
PM10 (Target)	Low, Normal, High, Danger

Table 2. The part of the dataset used as input data

Weather	Sand	Mist	Temperature	Humidity	Speed	Direction	PM10
Sunny	False	False	6	50	3	ENE	Low
Sunny	False	False	-1	48	2	WSW	Normal
Overcast	False	Ture	-2	50	2	WNW	High
Overcast	Ture	Ture	9	75	3	W	Danger
Rainy	False	Ture	9	85	3	WNW	Low
Rainy	False	Ture	5	69	3	WSW	Normal
Snowy	False	False	-9	56	2	ENE	Normal
Snowy	False	Ture	-5	76	2	ENE	High

Table 3. The distribution of PM10 in the dataset

Low (0~30μ/m^3)	Normal (31~80μ/m^3)	High (81~150 μ/m^3)	Danger (> 150 μ/m^3)
723	1775	261	39

3 METHODOLOGY

IN this study, meteorological factors and PM10 data were collected for predicting the concentration of PM10. We divided the dataset into training data and test data. Training data was divided as 78 % (2182) of dataset. Test data was also divided as 22 % (616) of dataset. This training dataset is used as input data of DNN model. Based on this data, we have compared various performance based on various parameter settings of DNN model.

3.1 One-Hot Encoding

In this study, One-Hot Encoding was used as a data pre-processing method. One-Hot Encoding is a method introduced in Collobert et al. (2011). This is a pre-processing method that transforms categorical features into a format that works well in classification and regression algorithms. One-Hot Encoding has two advantages. It is easy to be designed and modified. And it is easy to detect illegal states. One-Hot-Encoding sets the feature corresponding to the value to 1, and sets the feature to 0 if it is not. How One-Hot Encoding works is shown in [Table 4].

Table 4. The Example of One-Hot Encoding

Sample	Human	Dog	Cat
Human	1	0	0
Dog	0	1	0
Cat	0	0	1

3.2 Deep Neural Network (DNN)

Deep Neural Network (DNN) is a neural network model with several hidden layers between the input and output layer. Each input data will go into a node called neuron. Then, when it is sent from the neuron of the input layer to the neuron of the next layer, it is calculated and transmitted along with the weight for each input data. Based on this process, the results are derived. The basic structure of DNN is shown in [Figure 1].

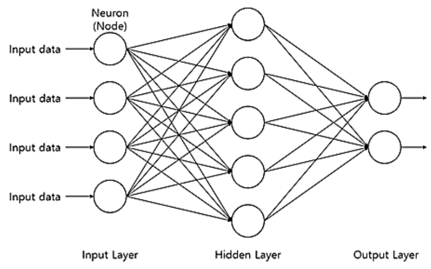

Figure 1. Examples of DNN model

When designing a DNN, we can set various parameters. [Figure 1] shows that there is a hidden layer between the input layer and the output layer. In this study, we have experimented from 1 to 10 hidden layers. Generally, one layer consists of several neurons. Inside a neuron, each input data is firstly multiplied by each weight. Then, the result values are added, and passed to the activation function. The result of the calculation of the activation function is the final result of the neuron. So, this output value represents the result for input data. The process is shown in [Figure 2]. In addition, the calculation method of this process is as follows (1).

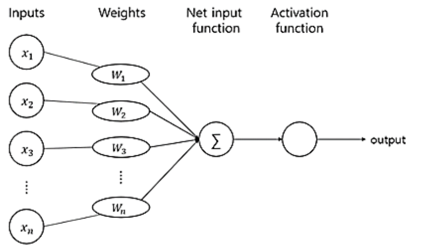

Figure 2. . Calculation process in neurons

$$h_{W,b}(x) = f(W^T x) = f(\textstyle\sum_{i=1}^{n} w_i x_i + b) \quad (1)$$

where W is a weight vector including wis as its elements and b is a bias which is considered to be multiplied with x0=1.

There are three types of activation functions. The first is the Rectifier Linear Unit (ReLU) (Nair et al., 2010). ReLU is a function that computes the result as 0 if the input value is less than or equal to 0, and x if the input value is greater than 0. ReLU showed good performance in various studies such as Krizhevsky et al. (2012) at ImageNet. It's calculation is as follows (2).

$$f(x) = \max(0, x) \quad (2)$$

The second is Hyperbolic Tangent (Tanh). Tanh is a function that computes a result closer to -1 as the input value approaches negative infinity, and a result closer to 1 as the input value approaches positive infinity. Tanh is widely used in Recurrent Neural Network (RNN) such as Long Short-Term Memory (Hochreiter et al., 1997). It's calculation is as follows (3).

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3)$$

The third is the Sigmoid. Sigmoid is a function that computes a result closer to 0 as the input value approaches negative infinity, and a result closer to 1 as the input value approaches positive infinity. It's calculation is as follows (4).

$$f(x) = \frac{1}{1+e^{-z}} \quad (4)$$

When creating a neural network, it is very important how to initialize the weights. This is called initialization. In this study, we used the He initialization (He et al., 2015). He initialization adjusts the initialization value of the weight to the number of input neurons. This method reduces the initialization of the weights slightly when the number of input neuron from the previous layer per neuron is high, so that the pre-activation value does not become too large. This method takes into account the activation function. In particular, ReLU is not activated when x < 0, so it is calculated by adjusting to the neuron corresponding to half the number of input neurons. This is the way to do calculations in the range of the normal distribution (He_normal). It's calculation is as follows (5).

$$var(W) = sqrt(\frac{2}{n_{in}})\ (n_{in} : \text{Number of input neurons}) \quad (5)$$

There is also a method of calculating from the [-limit, limit] range using the characteristics of uniform distribution (He_uniform). Its calculation is as follows (6).

$$var(W) = limt = sqrt(\frac{6}{n_{in}})\ (n_{in} : \text{Number of input neurons}) \quad (6)$$

Next is Dropout (Dahl et al., 2013). Dropout solves the overfitting that occurs when there are multiple hidden layers. This is not to involve the entire weight in the calculation, but only some weight. The process of this method is shown in [Figure 3] and (Srivastava et.al., 2014) explained dropout in detail.

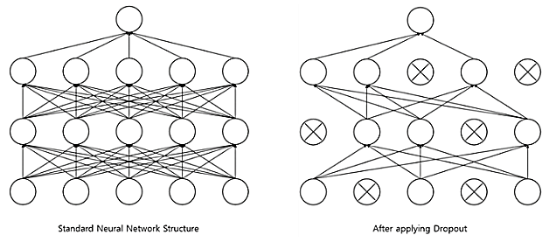

Figure 3. Example of applying Dropout (Srivastava et al., 2014)

In general, when learning from DNN, the way to increase learning speed was to increase the learning rate. However, high learning rates have caused gradient vanishing or gradient exploding problems. So, (Ioffe et al., 2015) introduced a new method called Batch Normalization. This method assumes that each feature is already uncorrelated. And for each feature, it is normalized by calculating mean and variance in scalar form. Then models add the scale (β) and shift (γ) factor to the normalized values, and these variables can be determined through training. And we do not calculate the mean and variance for the whole training data, but calculate it by mini- batch unit. So, only the mean and variance are calculated within the currently selected mini- batch, and normalized by using this value. The algorithm of batch normalization is the same as the following formulas (7), (8), (9), (10).

$$\mu_\beta = \frac{1}{m}\sum_{i=1}^{m} x_i \text{ (mini-batch mean)} \tag{7}$$

$$\sigma^2{}_\beta = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\beta)^2 \text{ (mini-batch variance)} \tag{8}$$

$$\hat{x}_i = \frac{x_i - \mu_\beta}{\sqrt{\sigma^2{}_\beta + \epsilon}} \text{ (normalize)} \tag{9}$$

$$y_i = \gamma\hat{x}_i + \beta \text{ (scale and shift, output)} \tag{10}$$

4 EXPERIMENT

IN this paper, the input data is meteorological factors, sand, fog, and PM10. The output data is the class of expected PM10 concentration. And we compared the performance of the DNN models according to the parameter settings. When comparing performance by parameters, each parameter has the following number of cases. The number of hidden layers is from 1 to 10. Activation functions are ReLU, Tanh, and Sigmoid. Dropout and batch normalization have two cases whether or not they are used. Initialization has two kids, He_normal, and He_uniform. Based on this, we have tested two cases, as mentioned above. We evaluated the performance of each parameters setting based on the activation function. We performed 120 experiments for each activation function. We compared the performance of the top 10 experimental results. The performance evaluation used accuracy, F1-scale and training loss. The results of the first case are shown in [Table 5] - [Table 7].

The results of ReLU-based experiments showed good performance when initialization, dropout, and batch normalization were used together. However, if we have more than five hidden layers, we can get better performance by excluding initialization. In terms of training loss, it is considered that using batch normalization helps to obtain better performance. However, when there are more than five hidden layers, it is good not to use batch normalization.

In the results of Tanh-based experiments, we could see that it is better not to use initialization when there have many hidden layers. Training loss results are slightly different from ReLU. When we used batch normalization, performance was better in here.

The results of Sigmoid-based experiments were different from those of ReLU and Tanh. Sigmoid was able to get good performance using initialization, dropout, and batch normalization regardless of the number of hidden layers. And the results of training loss showed that learning could be better than ReLU and Tanh.

Next, we tested the second case. The result of this is shown in the following [Table 9] - [Table 11].

The results of the second experiment with ReLU were similar to those of the first experiment. However, the difference is that using batch normalization gives good performance even when there are many hidden layers. In terms of performance, the second experiment based on ReLU yielded better performance than the first experiment.

The results of the second case based on Tanh were also slightly different from those of the first case. It is good performance even when batch normalization is not used. And the second case showed lower performance than the first case.

We, then, tested the second case based on Sigmoid. In this case, we could get better performance than the first case without using batch normalization. In addition, the results of this experiment showed better performance when there were many hidden layers.

Table 5. Top 10 performance results from the first experiment using ReLU (R : ReLU, H : Num of hidden layer, N : He_normal, U : He_uniform, D : Dropout, B : Batch Normalization)

No	Parameters	ACC (%)	F1
1	R (H=1) + D + B	75.3247	0.75
2	R (H=3) + D + N + B	75.3247	0.75
3	R (H=6) + D + U + B	75.3247	0.75
4	R (H=1) + U + B	75	0.75
5	R (H=1) + D + N + B	75	0.75
6	R (H=10) + N + B	75	0.75
7	R (H=8) + D + U + B	74.84	0.75
8	R (H=7) + D + B	74.84	0.75
9	R (H=9) + D + U + B	74.6753	0.74
10	R (H=8) + D + N + B	74.1883	0.74

No	Parameters	Loss
1	R (H=7) + N	0.61
2	R (H=7)	0.63
3	R (H=1) + D + B	0.63
4	R (H=3) + U	0.63
5	R (H=10) + N	0.63
6	R (H=2) + B	0.64
7	R (H=3) + B	0.64
8	R (H=2) + N + B	0.64
9	R (H=2) + D + U + B	0.64
10	R (H=2) + D + N + B	0.64

Table 6. Top 10 performance results from the first experiment using Tanh (T : Tanh, H : Num of hidden layer, N : He_normal, U : He_uniform, D : Dropout, B : Batch Normalization)

No	Parameters	ACC (%)	F1
1	T (H=2) + D + U + B	75.974	0.76
2	T (H=1) + D + N + B	75.487	0.76
3	T (H=3) + D + N + B	75.3247	0.75
4	T (H=1) + D + U + B	75.3247	0.75
5	T (H=3) + D + U + B	75.3247	0.75
6	T (H=5) + D + U + B	75.3247	0.75
7	T (H=3) + U + B	75	0.75
8	T (H=2) + D + B	75	0.75
9	T (H=6) + D + B	75	0.75
10	T (H=9) + D + B	75	0.75

No	Parameters	Loss
1	T (H=8)	0.6
2	T (H=6)	0.62
3	T (H=9) + B	0.63
4	T (H=1) + D + B	0.63
5	T (H=2) + D + B	0.63
6	T (H=1) + U + B	0.63
7	T (H=2) + D + N + B	0.63
8	T (H=2) + D + U + B	0.63
9	T (H=1) + D + N + B	0.64
10	T (H=5) + D + U + B	0.64

Table 7. Top 10 performance results from the first experiment using Sigmoid (S : Sigmoid, H : Num of hidden layer, N : He_normal, U : He_uniform, D : Dropout, B : Batch Normalization)

No	Parameters	ACC (%)	F1
1	S (H=2) + D + U + B	75.487	0.75
2	S (H=9) + U + B	75.3247	0.74
3	S (H=9) + D + N + B	75.1623	0.75
4	S (H=1) + D + B	75.1623	0.75
5	S (H=2) + D + B	75.1623	0.75
6	S (H=2) + D + N + B	75	0.75
7	S (H=7) + N + B	75	0.75
8	S (H=5) + D + U + B	75	0.75
9	S (H=7) + D + U + B	75	0.75
10	S (H=5) + D + B	75	0.75

No	Parameters	Loss
1	S (H=6) + N	0.57
2	S (H=10)	0.58
3	S (H=1) + B	0.63
4	S (H=9) + B	0.63
5	S (H=1) + D + B	0.63
6	S (H=2) + D + B	0.63
7	S (H=8) + N + B	0.63
8	S (H=9) + U + B	0.63
9	S (H=2) + D + N + B	0.63
10	S (H=2) + D + U + B	0.64

Table 8. The Confusion Matrix, which represents the best performance of the first experiment

Predicted / Actual	Danger	High	Normal	Low	Test data
Danger	0	7	0	3	10
High	0	9	0	47	56
Normal	0	0	66	67	133
Low	0	6	18	393	417

Table 9. Top 10 performance results from the second experiment using ReLU (R : ReLU, H : Num of hidden layer, N : He_normal, U : He_uniform, D : Dropout, B : Batch Normalization)

No	Parameters	ACC (%)	F1
1	R (H=5) + B	75.8117	0.75
2	R (H=2) + D + N	75.487	0.75
3	R (H=8) + B	75.1623	0.75
4	R (H=7) + U + B	75	0.74
5	R (H=5) + D + N + B	75	0.74
6	R (H=6) + B	74.8377	0.74
7	R (H=1) + D + N + B	74.8377	0.73
8	R (H=2) + B	74.6753	0.73
9	R (H=1) + D + U + B	74.6753	0.73
10	R (H=9) + D + N + B	74.6753	0.74

No	Parameters	Loss
1	R (H=1) + N + B	0.63
2	R (H=7) + N + B	0.63
3	R (H=1) + D + U + B	0.63
4	R (H=2) + U + B	0.63
5	R (H=4) + D + N	0.64
6	R (H=4) + U + B	0.64
7	R (H=7) + U + B	0.64
8	R (H=6) + B	0.64
9	R (H=8) + B	0.64
10	R (H=9) + B	0.64

Table 10. Top 10 performance results from the second experiment using Tanh (T : Tanh, H : Num of hidden layer, N : He_normal, U : He_uniform, D : Dropout, B : Batch Normalization)

No	Parameters	ACC (%)	F1
1	T (H=3) + D	75.3247	0.74
2	T (H=3) + D + N	75.3247	0.75
3	T (H=8) + D + U + B	75.3247	0.74
4	T (H=6) + B	75.3247	0.74
5	T (H=7) + D	75	0.74
6	T (H=7) + D + U	75	0.74
7	T (H=1) + D	74.8377	0.74
8	T (H=5) + D + U + B	74.8377	0.74
9	T (H=4) + B	74.8377	0.74
10	T (H=5) + N + B	74.6753	0.74

No	Parameters	Loss
1	T (H=3) + B	0.62
2	T (H=2) + B	0.63
3	T (H=5) + U + B	0.63
4	T (H=1) + D + U + B	0.63
5	T (H=2) + D + U + B	0.63
6	T (H=5) + D + U + B	0.63
7	T (H=3) + D + N + B	0.63
8	T (H=7) + D + N + B	0.63
9	T (H=10) + U + B	0.63
10	T (H=5) + N + B	0.63

Table 11. Top 10 performance results from the second experiment using Sigmoid (S : Sigmoid, H : Num of hidden layer, N : He_normal, U : He_uniform, D : Dropout, B : Batch Normalization)

No	Parameters	ACC (%)	F1
1	S (H=9)	75.8117	0.75
2	S (H=10)	75.8117	0.75
3	S (H=6) + N	75.6494	0.76
4	S (H=5) + D + N	75.3247	0.74
5	S (H=7) + N	75.3247	0.75
6	S (H=5) + N + B	75.3247	0.75
7	S (H=8) + U	75.3247	0.75
8	S (H=3) + D + U	75.1623	0.75
9	S (H=9) + U	75.1623	0.75
10	S (H=3) + D + N	75	0.75

No	Parameters	Loss
1	S (H=4) + U + B	0.57
2	S (H=4) + B	0.58
3	S (H=6) + N	0.63
4	S (H=7) + N	0.63
5	S (H=10) + N	0.63
6	S (H=3) + N + B	0.63
7	S (H=5) + N + B	0.63
8	S (H=7) + U + B	0.63
9	S (H=2) + B	0.63
10	S (H=3) + D + U	0.64

Table 12. The Confusion Matrix, which represents the best performance of the second experiment

Predicted / Actual	Danger	High	Normal	Low	Test data
Danger	3	1	0	2	6
High	4	6	1	45	56
Normal	1	1	76	71	149
Low	4	6	37	358	405

And Sigmoid showed the best performance in terms of training loss. However, when we compare the overall performance, we can determine that ReLU has the best performance. The best performance was seen when using the top first parameters in ReLU table. In this case, the accuracy was 75.8117% that low class predicts 358 from 405 in real answer, normal class predicts 76 from 149, high class predicts 6 from 56 and the danger class predicts 3 from 6. And also it is more judged to distinguish between high and danger classes. The result confusion matrix is shown in [Table 12].

5 CONCLUSION

IN this study, we predicted the concentration of PM10 using meteorological factors, sand, fog, PM10, and DNN. When conducting the experiment, we considered two cases. The first is an experiment to predict the concentration of PM10 on the next day using the weather forecast data of that day. The second experiment was to predict the concentration of PM10 on the next day using the previous day's data. At this time, we tried to find optimal parameters by comparing various parameters of DNN. When we look at the confusion matrix of results, it seems that because there was not enough data, it was very difficult to distinguish between the High class and the Danger class.

In the future, we will try to predict PM10 concentration with higher performance by extracting data of various features. In addition, we will design DNNs of various structures to solve this problem by finding a structure optimized for this problem.

6 ACKNOWLEDGEMENT

THIS research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (2015R1A2A2A01007333) and by the Naitonal Research Foundation of Korea Grant funded by the Korean Government (NRF-2017M3C4A7068188).

7 REFERENCE

R. Collobert, Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). "Natural language processing (almost) from scratch."

Journal of Machine Learning Research, 12.Aug, 2493-2537.

G. E. Dahl, Tara N. Sainath, & Geoffrey E. Hinton. (2013). "Improving deep neural networks for LVCSR using rectified linear units and dropout." *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on.* IEEE.

M. M. Dedovic, Avdakovic, S., Turkovic, I., Dautbasic, N., & Konjic, T. (2016). "Forecasting PM10 concentrations using neural networks and system for improving air quality." *Telecommunications (BIHTEL), 2016 XI International Symposium on.* IEEE.

K. He, Zhang, X., Ren, S., & Sun, J. (2015). "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." *Proceedings of the IEEE international conference on computer vision*.

G, E, Hinton, Osindero, S., & Teh, Y. W. (2006). "A fast learning algorithm for deep belief nets." *Neural computation*, 18.7, 1527-1554.

S. Hochreiter & Schmidhuber, J. (1997). "Long short-term memory." *Neural computation,* 9.8, 1735-1780.

S. K. Hur, Oh, H. R., Ho, C. H., Kim, J., Song, C. K., Chang, L. S., & Lee, J. B. (2016). "Evaluating the predictability of PM 10 grades in Seoul, Korea using a neural network model based on synoptic patterns." *Environmental Pollution,* 218, 1324-1333.

S. Ioffe & Szegedy, C. (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *arXiv preprint arXiv, 1502.03167*.

A.-S. Jang. (2014). "Impact of particulate matter on health." *Journal of the Korean Medical Association in Korea*, 57.9. 763-768.

T, Jiang, Peng Lei., & Qin Qin. (2016). "An Application of SVM-Based Classification in Landslide Stability." *Intelligent Automation & Soft Computing*, 22.2, 267-271.

F. Khodarahmi, Soleimani, Z., Yousefzadeh, S., Alavi, N., Babaei, A. A., Mohammadi, M. J., & Goudarzi, G. (2016). "Levels of PM10, PM2. 5 and PM1 and Impacts of Meteorological Factors on Particle Matter Concentrations in Dust Events and non Dusty Days." *International Journal of Health Studies,* 1.3, page-7.

K. H. Kim, Kabir, E., & Kabir, S. (2015). "A review on the human health impact of airborne particulate matter." *Environment international*, 74, 136-143.

A. Krizhevsky, Sutskever, I., & Hinton, G. E. (2012). "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems.*

V. Nair & Hinton, G. E. (2010). "Rectified linear units improve restricted boltzmann machines." Proceedings of the 27th international conference on machine learning (ICML-10).

C. O. Sakar, Demir, G., Kursun, O., Ozdemir, H., Altay, G., & Yalcin, S. (2011). "Feature Selection for The Prediction Of Tropospheric Ozone Concentration Using A Wrapper Method." *Intelligent Automation & Soft Computing*, 17(4), 403-413.

R. Salakhutdinov, Mnih, A., & Hinton, G. E. (2007). "Restricted Boltzmann machines for collaborative filtering." *Proceedings of the 24th international conference on Machine learning.* ACM.

M. K. Shin, Lee, C. D., Ha, H. S., Choe, C. S., & Kim, Y. H. (2007). "The Influence of Meteorological Factors on PM1 Concentration in incheon." *Journal of Korean Society for Atmospheric Environment in Korean,* 23.3, 322-331

N. Srivastava, Hinton, G., Krizhevsky, A., Sutskever, I, & Salakhutdinov, R. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research*, 15, 1929-1958.

8 DISCLOSURE STATEMENT

NO potential conflict of interest was reported by the authors.

9 NOTES ON CONTRIBUTORS

Byoung-Doo Oh is a graduate student in the School of Software at Hallym University. He is now studying natural language processing and machine learning.

Hye-Jeong Song, received the ph.D. degree in Computer Engineering from Hallym University. She is a professor in School of Software of Hallym University, Korea. Her recent interests focus on biomedical system and bioinformatics.

Jong-Dae Kim, received the M.S. and the Ph.D. degrees in Electrical Engienering from Korea Advanced Institute of Science n Technology, Seoul, Korea, in 1984 an 1990, respectively. He worked for Samsung Electronics from 1988 to 2000 as an electrical engineer. He is a Professor in School of Software, Hallym University. His recent interests focus on biomedical system and bioinformatics

Chan-Young Park, received the B.S. and the M.S. from Seoul National University and the ph.D. degree from Korea Advanced Institute of Science and Technology in 1995. From 1991 to 1999, he worked at Samsung Electronics. He is currently a Professor in the School of Software of Hallym University, Korea. His research interests are in Bio-IT convergence and sensor networks.

Yu-Seop Kim, received the Ph.D. degree in Computer Engineering from Seoul National University. He is currently a Professor in the School of Software of Hallym University, Korea. His research interests are in the areas of bioinformatics, machine learning, and natural language processing.