The Comparison of Performance According to Initialization Methods of Deep Neural Network for Malware Dataset

Young-Man Kwon, Yong-woo Kwon, Dong-Keun Chung and Myung-Jae Lim

Abstract: Training Deep NN, proper initialization of weights leads to good performance. The methods that commonly used to initialize the weights are, limited variance of weight's values and re-use unsupervised pre-trained weights. In this paper, we proposed the new algorithm that some of weights are used after being pre-trained by the CD method of unsupervised DBN and the other of the weights are initialized using the Xavier or He initialization method. We call these DBNnX and DBNnHe. We compare the performance with several DBNnX, DBNnHe and existing methods. We evaluated and visualized these by using AUC score and using box plot. As the result of experiment, we found the DBN2X and DBN2He are best.

Index Terms: Weight initialize, DBN, Malware dataset, AUC, Deep NN

I. INTRODUCTION

Training Deep NN(Neural Network) suffered from various problems, for example, over-fitting, vanishing/exploding gradients and etc. Many researches tried to find solutions and improve performance of Deep NN training. Especially, the initialization of weight is one of the importantly treated area and the variety of methods are appeared[1][2].

In the simplest approach of weights initialization, weights are initialized randomly with normal distribution. The other approaches are to limit the variance of weights. These methods determine the initial value of weights by considering of the input and output connection of weights. Another approach uses the initialized weights that pre-trained by unsupervised methods of Deep NN. The training algorithms for AutoEncoder and DBN(Deep Belief Networks) are commonly used in this approach.

In this paper, we used the combined methods that some weights are pre-trained by DBN and the other of weights are initialized by Xavier and He methods. We compare the performance of these methods with existing methods for malware dataset. We used the value of AUC(Area Under curve) as metric of the performance and plotted boxplot of them. As the result of experiment, we found our proposed method is better than other existing algorithms.

Revised Manuscript Received on March 02, 2019.

Young-Man Kwon, Department of Medical IT, Eulji University, Republic of Korea.

Yong-woo Kwon, Department of Medical IT, Eulji University, Republic of Korea.

Dong-Keun Chung, Department of Medical IT, Eulji University, Republic of Korea.

Myung-Jae Lim, Department of Medical IT, Eulji University, Republic of Korea.

II. RELATED WORKS

A. Activation functions

In this paper, we used partially pre-trained methods using DBN and Xavier/He methods to compare with other methods, using malware dataset. As the compare methods, we used AUC and plotting their box plot. The result of our experiment got improved performance about avg(average), std(standard deviation) of AUC.

The NN uses a activation function that transform a input signal to output signal non-linearly. It is important to choose the appropriate activation function to train Deep NN successfully. There are many activation functions those are sigmoid, hyperbolic tangent, ReLU and so on.

Sigmoid activation function is non-linear function that always has a value between 0 and 1. The step-function has only the output value of 0, 1, but the sigmoid makes values between these continuous. The graph of it is shown in Fig.1 and the expression of it is shown in following equation (1).

$$\sigma(t) = \frac{1}{1+\exp(-t)} \qquad (1)$$

The graph of hyperbolic tangent activation function looks similar to sigmoid, but its output value ranges from -1 to 1. That is shown in Fig.1 and makes each output more or less normalized. The equation is as follows.

$$\tanh(x) = \frac{2}{1+e^{-2x}} - 1 \qquad (2)$$

ReLU is widely used than sigmoid and tanh in these days. Its equation is shown in (3).

$$\mathrm{ReLU(x)} = \max(0, z) \qquad (3)$$

When the input exceeds 0, the output is the same as input and when it is less than 0, the output is 0. This function often used to solve vanishing gradient problem.

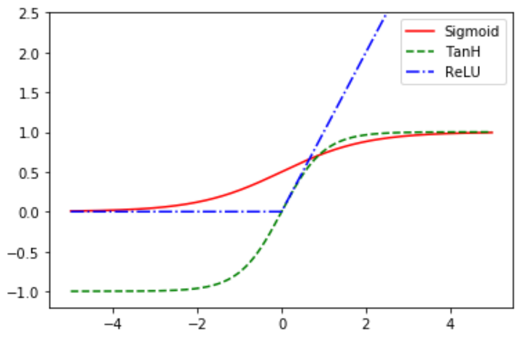

Fig.1 Graph of activation function

B. Initialization methods

Training DNN(Deep Neural Network), in the past, suffered from exploding and vanishing gradients. At that time, the initialization method of weights for the DNN was commonly used with normal distribution with mean 0 and standard deviation 1. Generally, we call this method as SND(Standard Normal Distribution) initialization.

In 2010, Xavier and Yoshua found weight initialization technique with sigmoid, hyperbolic tangent and softsign activation function to solve the gradient problems[1]. We show this technique in Table 1. Where, fan_{in} and fan_{out} are the number of input and output weight connections. These are also called n_{inputs} and $n_{outputs}$.

Table 1 Existing initialization methods

SND(Standard Normal Distribution) Initialization		
Standard Normal Distribution mean : 0 standard deviation : 1		
Xavier Initialization		
Activation function	Uniform distribution [-r, r]	Normal distribution
sigmoid	$r = \sqrt{\frac{6}{n_{inputs} + n_{outputs}}}$	$\sigma = \sqrt{\frac{6}{n_{inputs} + n_{outputs}}}$
Hyperbolic tangent	$r = \sqrt[4]{\frac{6}{n_{inputs} + n_{outputs}}}$	$\sigma = \sqrt[4]{\frac{6}{n_{inputs} + n_{outputs}}}$
He Initialization		
ReLU	$r = \sqrt{2}\sqrt{\frac{6}{n_{inputs} + n_{outputs}}}$	$\sigma = \sqrt{2}\sqrt{\frac{6}{n_{inputs} + n_{outputs}}}$

Using the previous SND initialization, when the inputs become large(negative or positive), the output of function saturates at 0 or 1, with a derivative of it extremely close to 0. Thus, it has virtually no gradient to propagate back through the network. These symptoms were only observed empirically. The Xavier's initialization method sets equal variance of the inputs and outputs of each layers. They also need the gradients have equal variance in back propagation.

After few years, He et al. introduce initialization methods for the ReLU function by using similar strategies [2], This ReLU function completely eliminates the signal on the negative side. The He's initialization method is also shown in Table 1. Both methods can be applied for normal distribution and uniform distribution.

C. DBN(Deep Belief Networks)

There is another initialization technique for weights. This is to reuse the pre-trained weights. Generally, Pretraining is done with unsupervised methods such as DBN(Deep Belief Network) or Autoencoder. In the experiments of this paper, we use DBN as case study.

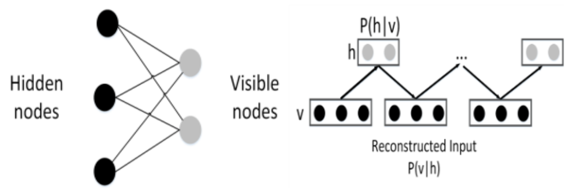

Fig.2 (a) Structure of RBM (b)Training RBM

DBN consist of stacked RBM(Restricted Boltzmann Machines). In Fig.2 (a), we show the example of RBM that consists of two layers, visible nodes v and hidden nodes h. Nodes within layer has no interconnections each other, but nodes between them are fully connected by using the symmetric weights.

The CD(Contrastive Divergence) algorithm is used to train RBM[3][4]. At First, the CD algorithm set the state of the visible nodes to a training vector. Then it computes the binary state of all the hidden nodes in parallel using equation (4).

$$p(h_i = 1|v) = \sigma(b_j + \textstyle\sum_i v_i w_{ij}) \quad (4)$$

where, $\sigma(x)$ is the logistic sigmoid function $1/(1 + \exp(-x))$. The v_i is the binary states of visible node i. The b_j is bias for hidden node and w_{ij} is weight between visible node i and hidden node j. After this, it creates random sample to decide if each hidden node h_j fires.

After binary states h_j have been decided for the hidden nodes j, a "reconstruction" that produce the probability of visible node i is performed by using equation (5).

$$p(v'_i = 1|h) = \sigma(a_i + \textstyle\sum_j h_j w_{ij}) \quad (5)$$

where a_i is bias for visible unit. In a similar way to above, it decide if each visible node v'_i fires.

The objective is to train in order for original v_i and reconstructed v'_i to have the same energy. As using the CD objective function ($\mathrm{Mean}\left(\mathrm{F}(V_{original})\right) - Mean(F(V_{reconstructed})))$, the updating weight is then given by equation (6).

$$dW = \eta(V^T h_{input} - V^T h_{reconstructed} \quad (6)$$

Here, η is the learning rate. Similar expressions exist for the biases b and c.

In the supervised DBN learning method, using DBN as a pre-training method is shown in Fig.3. First, the network is layer-wise trained by using RBM's CD algorithm. This is the pre-train process because it did not use the output. After that, the supervised DBN further trains the network by using pre-trained weights and output.

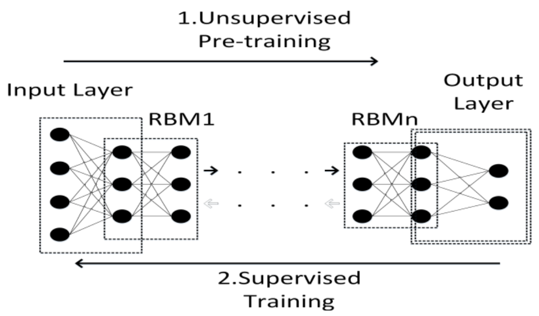

Fig.3 DBN as pre-train method

D. Opcode and feature vectors

The PE file format is a execution file format defined by the Windows operating system and files with name extension such as EXE, DLL, FON use this format.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

This format consists of machine instructions and other information for running. The instruction consists of two parts, opcode and operand. The opcode is the part that specifies the operation behavior of a command and expresses the kind of instruction. The function of the opcode includes function calculation, data transfer, control function and input/output function. We extracted Opcode only and use this as feature.

To construct a feature vector of PE file (document), we used BoW(Bag of Words) and TF-IDF. BoW is the method where we count the occurrence of words in a document without giving importance to the grammar and the order of words [5].

$$TF(term) = \frac{Number\ of\ times\ term\ appers\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document}$$

$$IDF(term) = log\ (\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ a\ given\ term\ in\ it}\ (7)$$

As shown in equation (7), TF-IDF is a value that obtained by multiplying TF(Term Frequency) by IDF(Inverse-Document Frequency). TF is value that indicating how often a specific word appears in a document. DF(Document Frequency) is the frequency which a particular word appears in another document, and IDF is inverse of DF. The higher the TF-IDF score, the less often they appear in other documents, and the more often they appear in the relevant document.

III. THE DESIGN OF EXPERIMENT

In this paper, we used 1224 files as dataset for experiment. The 779 Malware files download from the Web sites of "Virusshare [6]" and "malwareulrs.joxeankret [7], "malc0de[8]", "malwareblacklist [9] for learning the network. The malware consist of Trojan 418, PUP 176, Virus 58, Backdoor 34, Adware 29, Downloader 21, Spyware 13, and so on. The 445 Benign files are from the Window file directory. Ahn et al.[10] uses these dataset and extracted opcode, API for feature vectors. We use same methods to extract opcode and use it to construct the feature vector.

In this paper, we compare the performance of the classifiers according to weight initialization methods. We use FC(Fully-Connected) network for classifier. The network structure that is used in our experiment is shown in Fig.4. It is composed of 5 layers. First layer is input layer and last is output layer. Neurons of hidden layers are set to [256, 128, 10].

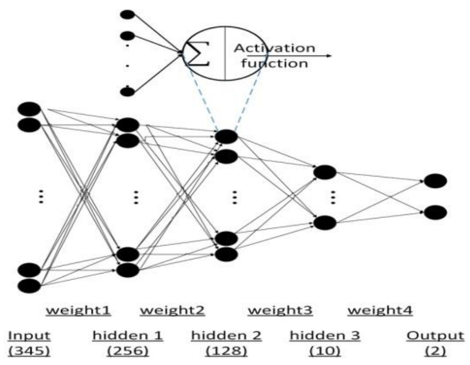

Fig.4 DNN structure that is used in our experiment

Table 2 overall experiment of weight-initialization methods

Name of initialization	Weight1	Weight2	Weight3	Weight4
SND	SND	SND	SND	SND
DBN	DBN	DBN	DBN	DBN
Xavier	Xavier	Xavier	Xavier	Xavier
He	He	He	He	He
DBN1X	DBN	Xavier	Xavier	Xavier
DBN1He	DBN	He	He	He
DBN2X	DBN	DBN	Xavier	Xavier
DBN2He	DBN	DBN	He	He
DBN3X	DBN	DBN	DBN	Xavier
DBN3He	DBN	DBN	DBN	He

The overall experiment of our initialization methods is shown in Table 2. The weights are initialized by existing methods such as SND, DBN, Xavier and He. Then the performances are measured. The weights are also initialized by our proposed methods such as DBNnX or DBNnHe. In DBNnX and DBNnHe, n is the number of pre-trained layer. The Xavier and He mean that the very layers are initialized by Xavier's or He's initialization method.

When using classifier, various hyper parameters must be set. In our experiment, learning rate is set 0.001 and batch size is set 100. Adam optimizer is used for optimization method. For RBM layers are trained by CD algorithm for 1-step.

For the measurement of performance, the most widely used method is ROC curve. ROC curve is the graph that represents how the sensitivity and specificity change with 2-dimensional plane, AUC(Area Under the Curve) is area under the curve of ROC. The larger AUC, gives the better classification result. We measure AUC score and test accuracy for each classifier.

IV. EXPERIMENT AND RESULT

In this paper, we compare performance of classifier according to weight initialization methods. To achieve this goal, we made experiments with different feature vectors. The first experiment uses the feature vector of Count-BoW. The second use TF-IDF and the third use extended input vector. We did these experiment with different activation functions such as sigmoid, tanh(hyperbolic tangent) and ReLU to see the effect of activation function.

We want to analyse experiments statistically and run them 30 times. We selected and used the value of random seeds differently for reproduction of same result. We calculate average and standard deviation for all run's AUC score and accuracy score. After this, we show boxplot of these.

A. Count-BoW feature vector

We show the avg and std for the result of experiments with Count-BoW in Table 3. And we also show the boxplot of these in Fig.5.

Table 3 avg and std of AUC score with Count-BoW feature vector

		Sigmoid	Tanh	ReLU
SND	Avg	0.95	0.95	0.95
	Std	0.015	0.015	0.015
DBN	Avg	0.98	0.94	0.54
	Std	0.002	0.017	0.10
Xavier	Avg	0.96	0.96	0.93
	Std	0.013	0.014	0.09
He	Avg	0.96	0.96	0.95
	Std	0.012	0.013	0.02
DBN1X	Avg	0.97	0.96	0.95
	Std	0.005	0.006	0.02
DBN1He	Avg	0.97	0.96	0.92
	Std	0.004	0.006	0.05
DBN2X	Avg	0.97	0.97	0.94
	Std	0.004	0.007	0.08
DBN2He	Avg	0.97	0.97	0.95
	Std	0.006	0.008	0.03
DBN3X	Avg	0.97	0.97	0.94
	Std	0.004	0.007	0.08
DBN3He	Avg	0.95	0.94	0.93
	Std	0.002	0.006	0.04

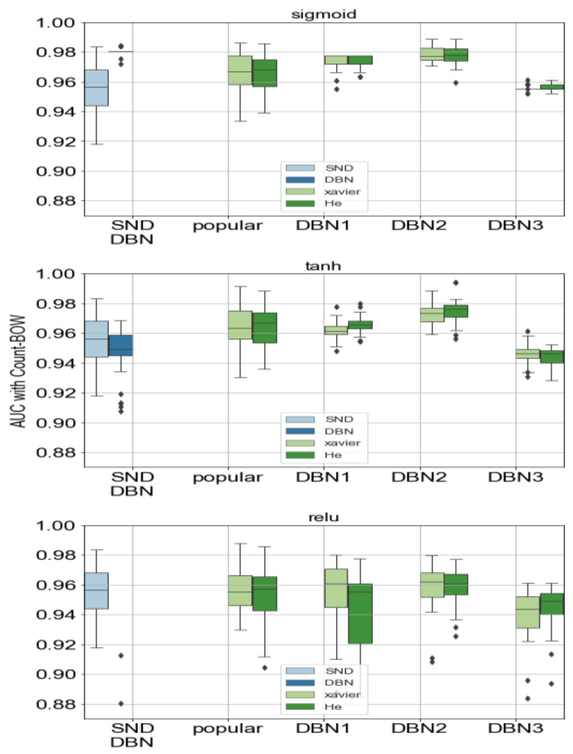

Fig.5 AUC score boxplot with Count-BoW feature vector and variety activation function

The existing methods such as SND and DBN, these are traditional methods, are appeared in the first column of boxplot. Where, DBN means that the values of all weights are trained only by unsupervised DBN method and are used as initial value during classification. Xavier's and He's methods, those are popular in deep NN, are appeared in second column. These methods initialize the weights of network only by Xavier's or He's method, are used to reference of our proposed method. As we expect, we found that popular methods are superior to SND and DBN.

In the case of using sigmoid function, DBN1s and DBN2s showed similar results to popular methods. However, IQR(Inter-Quantile Range) of boxplot is shorter. A short IQR means less variance. But DBN3s showed lower performances than popular methods.

In the case of using tanh function, DBN2s showed better results than popular methods. DBN1s showed similar result to popular methods. However, IQR of boxplot is also shorter. DBN3s also showed lower performance as before.

In the case of using ReLU function, the performance of all the DBNns methods and popular methods are similar to each other. If we compare these to above case, these show average value are slightly lower, variance are slightly higher. Specially, DBN, that is in the first column, got average of 0.54 and standard deviation of 0.1.

As the result of experiment using Count-BoW feature vector, DBN2s got the highest score in average. DBN1s showed similar result compared to popular methods. DBN3s showed lower performances than popular methods.

B. TF-IDF feature vector

We want to know that our proposed methods are superior to popular methods. So we did the same experiment with different feature vectors.

We show the average and standard deviation for the result of experiments with TF-IDF in Table 4. And we also show the boxplot of these in Fig.6.

Table 4 avg and std of AUC score with TF-IDF feature vector

		Sigmoid	Tanh	ReLU
SND	Avg	0.954	0.907	0.874
	Std	0.015	0.024	0.114
DBN	Avg	0.98	0.946	0.54
	Std	0.002	0.017	0.106
Xavier	Avg	0.966	0.961	0.957
	Std	0.012	0.014	0.026
He	Avg	0.965	0.963	0.961
	Std	0.010	0.012	0.016
DBN1X	Avg	0.971	0.963	0.955
	Std	0.005	0.006	0.016
DBN1He	Avg	0.972	0.964	0.931
	Std	0.004	0.006	0.086
DBN2X	Avg	0.977	0.974	0.92
	Std	0.005	0.008	0.112
DBN2He	Avg	0.976	0.973	0.944
	Std	0.005	0.009	0.062
DBN3X	Avg	0.955	0.945	0.935
	Std	0.001	0.006	0.032
DBN3He	Avg	0.955	0.944	0.943
	Std	0.001	0.007	0.015

In the case of using sigmoid function, DBN1s and DBN2s showed better result than popular methods. DBN3s showed lower performance than popular methods and got standard deviation of 0.001 that is shortest IQR of boxplot. DBN got average of 0.98 and standard deviation of 0.002 that is the best performance in sigmoid.

In the case of using tanh function, DBN2s showed better performance than popular methods. DBN2X got average of 0.974 and standard deviation of 0.008 and DBN2He got average of 0.973 and standard deviation of 0.009. Xavier's method got average of 0.961 and standard deviation of 0.014. And He's methods got average of 0.963 and standard deviation of 0.012. DBN1s showed similar result to popular methods and DBN3 showed lower result than popular methods. Compare to sigmoid activation function, Only-DBN showed lower performance.

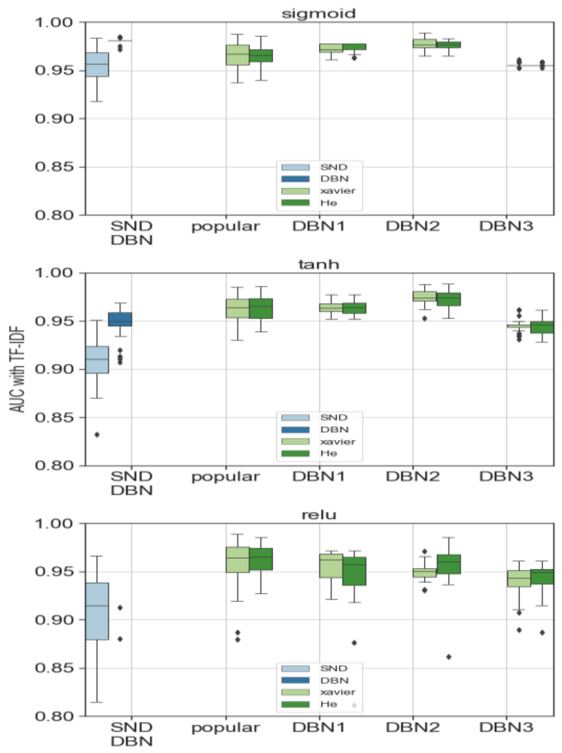

Fig.6 AUC score boxplot with TF-IDF feature vector and variety activation function

In the case of using ReLU function, popular methods showed better performance than other methods. Xavier's method got average of 0.957 and standard deviation of 0.026. And He's method got average of 0.961 and standard deviation of 0.016. Similar to first experiment, DBN showed much worse performance than above two activation function. As the result of second experiment, DBN2s got highest score.

C. feature vectors with 2-gram

Computational cost is enhanced by extending dimension. We extended the input dimension by using the n-gram method to see how this affects our results. N-grams are fundamentally a set of co-occurring or continuous sequence of n items from a given sequence of large text[11]. When size of n is increases, dimension of input size also increases. We used 1-gram method for input vector in our above two experiments. In this third experiment, we use to feature vector of 2-gram. Results of the third experiment using a boxplot is shown in Fig.7 and Fig.8.

The results of this experiment in all activation functions, DBN2s showed better result than the other methods. Comparing DBN1s with popular methods, DBN1s showed similar or lower performance than popular methods. DBN3s showed lower performance than popular methods in mostly. However, IQR of DBNnX and DBNnHe are shorter than popular methods. Compared with the above two experiments, DBNnX and DBNnHe showed higher score slightly and showed shorter IQR. In the case of using ReLU function, the difference between the methods became a clearer. And in the same case, the performance of DBN has been greatly improved.

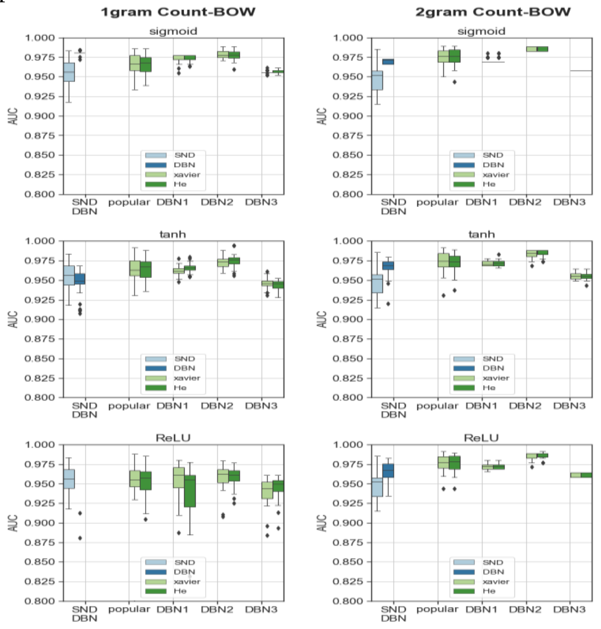

Fig.7 AUC score boxplot with 1-gram and 2-gram Count-BoW feature vector

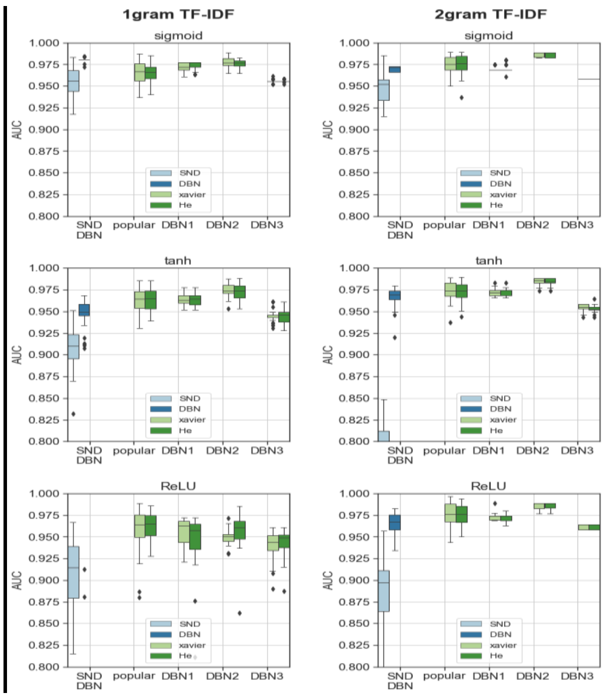

Fig.8 AUC score boxplot with 1-gram and 2-gram TF-IDF feature vector

V. CONCLUSION

In this paper, we did experiments that is how to set appropriate initial values to improve performance of classification for malware dataset. To achieve this goal, we made three different experiments. The first experiment uses the feature vector of Count-BoW. The second use TF-IDF and third use 2-gram method. In our experiments, we used existing methods such as SND and DBN, Xavier's and He's initialization methods as reference. And we compare the performance of these methods with DBNnX and DBNnHe partially pre-trained by DBN and are initialized by Xavier's and He's method. Where, n is the number of pre-trained layer by DBN.

As a result of our experiments, DBN2s showed best performance. DBN1s showed similar to the popular methods. DBN3s showed lower than the popular methods. So we conclude that it is worth enough for some of weights to be initialized by DBN and for the other of them to be initialized by Xavier's or He's methods.

ACKNOWLEDGMENT

This research was supported by R.O.K. National Research Foundation under grant NRF-2017R1D1A1B03036372. We thank our colleagues from laboratory who provided insight and expertise that greatly assisted the research. And thanks to National Research Foundation who supported laboratory.

REFERENCES

1. Xavier Glorot, Yoshua Bengio, “Understanding the difficulty of training deep feedforward neural networks”, Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp. 249-256, March 2010.
2. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, “Delving Deep into Rectifier”, Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), pp, 1026-1034 , December 07 - 13, 2015.
3. Antonio Gulli, Amita Kapoor, "TensorFlow 1.x Deep Learning Cookbook", Packt, 2017, pp. 272-277.
4. Hinton G.E. “A Practical Guide to Training Restricted Boltzmann Machines”, Neural Networks: Tricks of the Trade, pp. 599-619.
5. A Manohar Swamynathan, "Mastering Machine Learning with Pyhon in Six Steps", Apress, 2017, pp.268-271.
6. virusshare. https://virusshare.com.
7. joxeankoret. https://malwareurls.joxeankoret.com.
8. malc0de. https://malc0de.com.
9. Asghar, Eram, Aamer Ahmad Baqai, Ramshah Ahmad Toor, and Sara Ayub. "Co-Development of Process Planning and Structural Configurations Considering Machine's Accessibility in a Reconfigurable Setup." Review of Computer Engineering Research 3. 2 (2016): 41-46.
10. Tae-Hyun Ahn, Jae-Gyun Park, Young-Man Kwon, “A study on Performance of ML Algorithms and Feature Extraction to detect Malware”, The Journal of The Institute of Internet, Broadcasting and Communication (IIBC), Vol 18, No.1, pp.211-216, Feb, 2018.
11. Suryanto, T., Haseeb, M., & Hartani, N. H. (2018). The Correlates of Developing Green Supply Chain Management Practices: Firms Level Analysis in Malaysia. *Int. J Sup. Chain. Mgt Vol*, *7*(5), 316.
12. A Manohar Swamynathan , "Mastering Machine Learning with Pyhon in Six Steps ", Apress, 2017 , pp.267.

AUTHORS PROFILE

Young-Man Kwon
1993. 3 - : Professor. Department of Medical IT,
Eulji University. Republic of Korea.

Yong-woo Kwon
2013. 3 - : Undergraduate student. Department of Medical IT, Eulji University, Republic of Korea.

Dong-Keun Chung
1990. 3 - : Professor. Department of Medical IT,
Eulji University, Republic of Korea.

Myung-Jae Lim
1992. 3 - : Professor. Department of Medical IT,
Eulji University, Republic of Korea.