SSAE – DeepCNN Model for Network Intrusion Detection

Jong-Hwa Lee
Dept. of Computer Science
Kangwon National Univ.
Chuncheon, Republic of Korea
jonghwalee@kangwon.ac.kr

Jong-Wouk Kim
IGP. in Medical Bigdata Convergence
Kangwon National Univ.
Chuncheon, Republic of Korea
jw.kim@kangwon.ac.kr

Mi-Jung Choi
Dept. of Computer Science/
IGP. In Medical Bigdata Convergence
Kangwon National Univ.
Chuncheon, Replublic of Korea
mjchoi@kangwon.ac.kr

***Abstract*— Many people can use user-friendly internet services due to the development of IT and communication technologies. However, attackers perform attacks such as malware injection, DoS/DDoS, and system hacking to threaten end devices, personal information, and organizational assets. Security experts use anti-cyber-attack systems such as firewalls, anti-virus solutions, and intrusion detection systems (IDSs) to defend against various cyber threats. Also, many researchers work actively on machine learning-based detection models to protect and respond against advanced cyber-attacks. Therefore, we propose the stacked sparse autoencoder-deep convolutional neural network (SSAE-DeepCNN) model to detect network intrusions. Our proposed model is a semi-supervised learning model that combines stacked sparse autoencoder (SSAE) and deep convolutional neural network (DeepCNN). SSAE discovers new features from training data, and DeepCNN learns new features to detect network intrusions. We design various test scenarios to find the hyperparameters and structures of SSAE with the highest performance. We measure accuracy, F1-Score, prediction time, and hardware resource consumption to evaluate and compare models. The best scenario shows an accuracy of 93.5% by adding sparsity to SSAE's bottleneck. There is no significant difference in performance compared to when SSAE is not used, but resources used by GPU and CPU can be saved. In the future, we plan to improve the proposed model to get better performance.**

Keywords—SSAE, DeepCNN, IDS, Deep learning, Semi-supervised learning, Sparsity

I. INTRODUCTION

Many companies provide various internet services such as online shopping and mobile banking. Individuals take advantage of these services using various devices. Al-Qatf *et al*. [1] estimate that there will be 50 billion Internet-connected devices by 2022. For these services, mobiles and IoT devices collect and store user information. Attackers perform various cyber-attacks such as DoS/DDoS, malware injection, and side-channel attacks to steal this information and block the services [2]. Therefore, network security is very important in order to protect assets from various threats [3]. Security experts detect and defend various threats by employing firewalls, anti-virus solutions, and intrusion detection system (IDS). Among these systems, IDS automatically detects malicious behavior that violates security policies in networks. However, IDS challenges low accuracy and false alarm rate against the latest threats [4, 5]

There are two types of IDS: network-based IDS (NIDS) and host-based IDS (HIDS) depending on the source of information [6, 7]. NIDS monitors all packets on the network and HIDS monitors all host behavior. Both NIDS and HIDS can take signature-based or anomaly-based approach as a detection method. Signature-based IDS can detect well-known malware and attacks according to the patterns. However, it is difficult to detect unknown malware or zero-day attacks. On the other hand, anomaly-based IDS analyzes all packets to detect malicious behavior. Sometimes, this happens false positive and false negative alarms. To solve these problems, many researchers study IDS with ML machine learning (ML) and deep learning (DL).

ML is divided into three categories according to the learning method: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, each training samples consist of an input vector and an output (target) data. A supervised learning model trains the training samples and predicts the output vector from new samples. On the other hand, unsupervised learning performs data clustering or analyzes correlations using unlabeled data [8]. Reinforcement learning is a subcategory of machine learning that determines the next decision to obtain the best reward [9]. In particular, supervised learning is more dependent on training data than other methods. In supervised learning model, overfitting occurs when the model learns the details and even noise of training samples. If the model learns the training data too accurately, it can negatively affect the performance of the model on new data. Conversely, under-fitting means that the model cannot learn the training data or generalize new data. Therefore, preprocessing such as numericalization, normalization, and dimension reduction is important in supervised learning to avoid such problems.

In this paper, we propose a machine learning model for intrusion detection in the network. Our proposed model is an stacked sparse autoencoder-deep convolutional neural network (SSAE-DeepCNN) model which combines supervised and unsupervised learning. Stacked sparse autoencoder (SSAE) consists of an encoder that transforms data and a decoder that reconstructs transformed data. In the encoder, the last layer generates new features by applying the sparsity constraints to the data. Deep convolutional neural network (DeepCNN), composed of multiple 1D-CNN layers, classifies the output vector from the transformed data generated by SSAE.

This paper is organized as follows. Section 2 summarizes the related work on IDS, SSAE and CNN. Section 3 describes our proposed model. Section 4 explains the experimental setup, and evaluates the performance of SSAE-DeepCNN through the experiments. Finally, Section 5 concludes the paper.

II. RELATED WORK

A. IDS

The growing rate of cyber-attacks on system networks in recent years has weakened the privacy and security of computer infrastructure and personal computers. Many researchers work on how to enhance the performance of IDS using ML and DL. Parkar *et al*. [10] introduce five problems faced by the IDS field applying ML and DL. The first problem

is difficult to generalize a normal state of traffic. This is because every session shows diverse traffic usage. Second, the misclassification results of the model will be enormous to fix it. In particular, a false negative is very sensitive. Because an undetected attack can break down an entire system. Third, there is a major difference of opinion between analysts and operators. In terms of detecting malicious behavior and detecting attacks, it may mismatch the requirements. Fourth, it is difficult to detect malicious behavior due to various traffic types. Finally, a public dataset is very few to advance the performance of the model and evaluate the model. The authors proposed choosing an appropriate learning method or data according to the purpose of using the IDS.

B. Stacked Sparse Autoencoder

Autoencoder is one approach to automatically learn features from unlabeled data. Autoencoder consists of input, hidden, and output layers. The encoder transforms an input vector into an abstract vector using a combination of input and hidden layers. The decoder reconstructs (or recovers) the abstract vector into the original input vector using a combination of the hidden layers and the output layer [11, 12, 13]. SSAE belongs to one category of autoencoder. SSAE applies sparsity constraints to the hidden layers to obtain meaningful new feature vectors.

Zhang *et al*. proposed Xgboost-based SSAE to learn meaningful feature vectors of NSL-KDD dataset. In addition, the model using the binary tree and ensemble method showed a F1-Score of 91.97%. Yan *et al*. described that feature extraction is important to accurately classify network traffic as high-dimensional data with ML. Therefore, the authors used SSAE to generate meaningful feature vectors of the NSL-KDD dataset. As a result, the support vector machine (SVM) learned the significant feature vectors that SSAE generated and showed an accuracy of 99.35% and a classification speed of 3.29 seconds

C. Convolutional Neural Network

CNN uses the mechanism by which the brain processes visual information. CNN automatically learns complex vectors from images using an extraction filter called a kernel. Azizjon *et al*. [14] roposed a 1D-CNN-based intrusion detection model for network intrusion detection. 1D-CNN processing one-dimensional vector showed the best accuracy of 91.2% and a F1-Score of 91.59%. Wu *et al*. [15] proposed the LuNet model, a hierarchical neural network, combining CNN and recurrent neural network (RNN) for network intrusion detection. The authors trained the LuNet model using the NSL-KDD and UNSW-NB15 datasets. In the experimental results, the LuNet model showed accuracies of 97.4% and 97.7% for binary classification, and 99.1% and 85.0% for multi-class classification.

III. DESIGN OF MACHINE LEARNING MODEL FOR INTRUSION DETECTION

A. UNSW-NB15 dataset

In this study, we use the UNSW-NB15 dataset. The UNSW-NB15 dataset is a public dataset created by Cyber Range Lab by collecting and processing network packets using IXIA PerfectStorm. In the UNSW-NB15 dataset, 175,341 training data and 82,232 experimental data stored as files, and the number of data is shown in Table 1. The ‘attack_cat’ column shows the type of cyber-attack as ‘normal’, ‘reconnaissance’, ‘backdoor’, ‘DoS’, ‘exploits’, ‘analysis’, ‘fuzzers’, ‘worms’, ‘shellcode’, and ‘generic’. In addition, ‘label’ column shows ‘abnormal’ and ‘normal’ which means attack or not.

TABLE I. THE NUMBER OF DATA IN THE UNSW-NB15 DATASET

UNSW-NB15	Training Set		Testing Set		Total Set
	Normal	Abnormal	Normal	Abnormal	
Data	56,000 (22%)	119,341 (46%)	37,000 (14%)	45,332 (18%)	257,673 (100%)

TABLE II. NAME AND TYPE OF FEATURES USED IN THE EXPERIMENT

Name	Type	Name	Type
dur	numeric	tcprtt	numeric
spkts	numeric	synack	numeric
dpkts	numeric	ackdat	numeric
sbytes	numeric	smean	numeric
dbytes	numeric	dmean	numeric
rate	numeric	trans_depth	numeric
sttl	numeric	response_body_len	numeric
dttl	numeric	ct_srv_src	numeric
sload	numeric	ct_state_ttl	numeric
dload	numeric	ct_dst_ltm	numeric
sloss	numeric	ct_src_dport_ltm	numeric
dloss	numeric	ct_dst_sport_ltm	numeric
sinpkt	numeric	ct_dst_src_ltm	numeric
dinpkt	numeric	is_ftp_login	numeric
sjit	numeric	ct_ftp_cmd	numeric
djit	numeric	ct_flw_http_mthd	numeric
swin	numeric	ct_src_ltm	numeric
stcpb	numeric	ct_srv_dst	numeric
dtcpb	numeric	is_sm_ips_ports	numeric
dwin	numeric		

Data Preprocessing

For the model, the ‘id’ and ‘attack_cat’ are unnecessary for training. So we removed that from the 45 features of the UNSW-NB15 dataset and separated the ‘label’ column. At this moment, the categorical data ‘proto’, ‘service’, and ‘state’ features transformed to the sparse matrix in the process of applying one-hot encoder. As the dimension of the input data increases, the model may degrade performance due to the curse of dimension problem. Thus, we train the model with 39 features excluding the ‘id’, ‘attack_cat’, ‘proto’, ‘service’, and ‘state’ columns and the 'label' column with correct answers. We arranged the names and types of the 39 features used as shown in Table 2. Also, we apply the maximal-minimum normalization method that can normalize and reduce the deviation between each data.

B. Stacked Sparse Autoencoder and DeepCNN

SSAE is an unsupervised learning model with the same number of units in the input layer and the output layer. However, the number of units in the hidden layers may differ from the number of units in the input layer. Because the encoder transforms the dimension of the input vectors. If the input vectors are completely random without correlation, the compression is very difficult. However, if the training data has a related structure, SSAE can easily generate new feature

vectors that can reconstruct the original vectors in the hidden layer.

Figure 1 shows the SSAE-DeepCNN model that we proposed in this study. Also, Figure 2 shows the general structure of stacked autoencoder (SAE) and the middle layer of SAE is called coding layer or bottleneck. In [13], the author sets the sparse factor (ρ) parameter to 0.05 to apply the sparsity constraints in the coding layer. Informally, the author assumed that a neuron is active if its output value is close to one, and if its output value is close to zero, the neuron is inactive. The author would like to constrain the neurons to be inactive most of the time. The author sets ρ to 0.05 to disable any hidden units. If the average activation value of the hidden unit is close to zero, we can apply the sparsity constraints. For the average activation value of hidden units to be close to zero, the most activation values must be close to zero. To satisfy these conditions, the author used a loss function based on Kullback-Leibler divergence. When the distributions of the two data are different, the KL-divergence function can prevent the activation value from increasing by adding a penalty. If the value of ρ increases, the number of activated neurons increases.

We use the L1 regularization instead of the Kullback-Leibler divergence loss function in the coding layer. The L1 regularization can reduce the value of the weight to make it zero. The regularization factor (λ) means the degree to which the weight value decreases. As λ increases, the weight value decreases more, and the number of activated neurons decreases. If λ is too large, even the necessary weight value becomes zero, so it is important to choose an appropriate λ value. The structure of the proposed model is that transmits the feature vectors generated by adding sparsity constraints in the coding layer.

Figure 3 shows the DeepCNN model we propose in this paper. DeepCNN is a model in which two 1D-CNN layers, MaxPooling, BatchNormalization, and Dropout layers stacked four times as a bundle. It classifies whether it is a normal packet or an abnormal packet through the flatten layer and three dense layers. The MaxPooling layer extracts vectors from the weights calculated by the 1D-CNN layer with the convolution layer. As the number of layers increases, the input vector becomes increasingly irrelevant from the first input vector. To solve this problem, we adjust the mean and variance of the input vector through the BatchNormalization layer. Also, we set the Dropout layer to 0.3 to solve the overfitting problem. The filter sizes of the 1D-CNN layers are 64, 64, 32, 32, 16, 16, 8, and 8, respectively. Also, in Figure 3, only the leftmost 1D-CNN layer has a kernel size of six, and the others are the same as three.

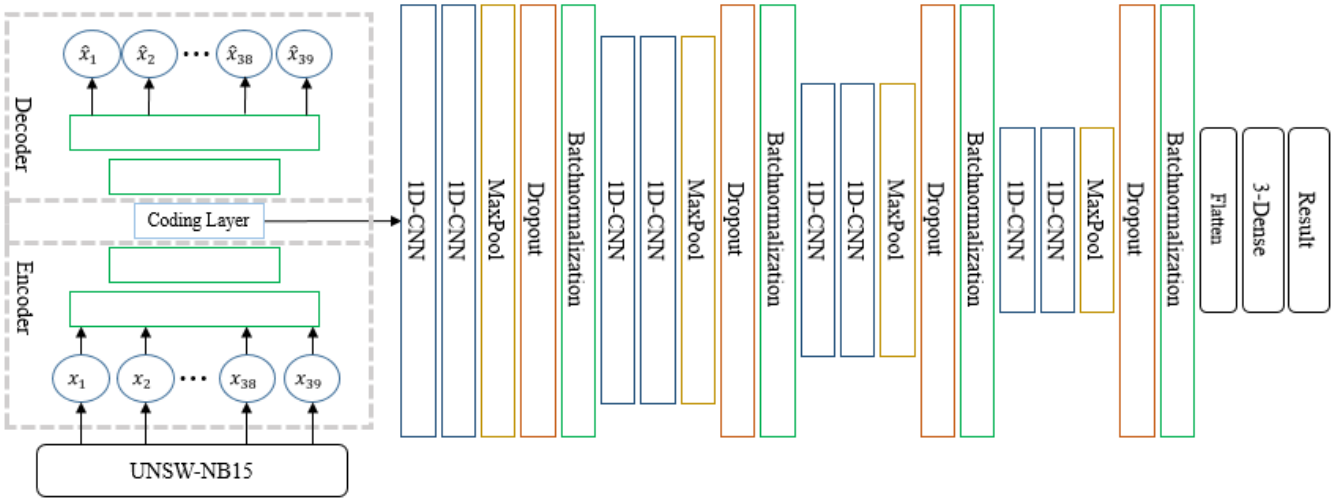


Fig. 1. Structure of SSAE-DeepCNN

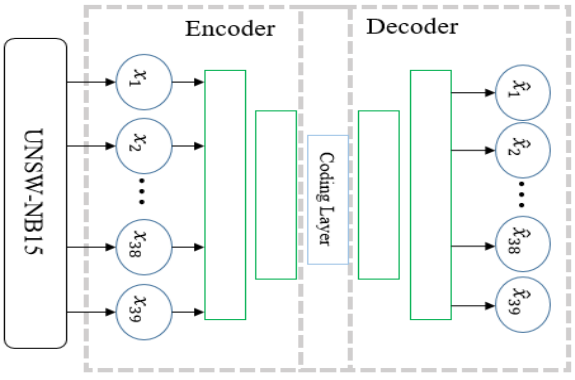


Fig. 1 Structure of SAE

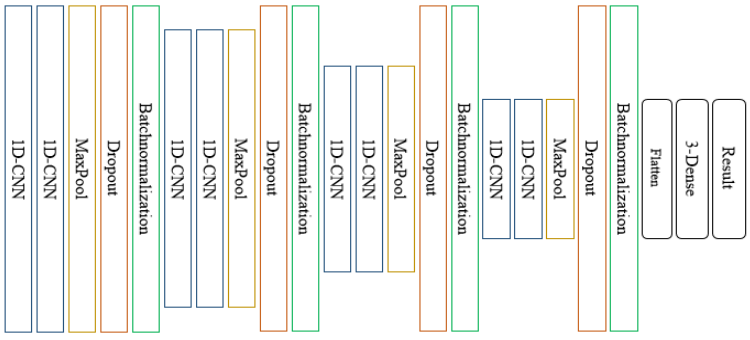


Fig. 3. Structure of DeepCNN

TABLE III. EXPERIMENTS SCENARIO ACCORDING TO THE STRUCTURE OF SSAE

Model	Test scenario	# of layers	Layer structure	Regularization factor (λ)
DeepCNN	(A)	N/A	N/A	N/A
SSAE-DeepCNN	(B)	3	[39, 78, 39]	$1e-3$
	(C)	5	[39, 78, 156, 78, 39]	
	(D)	7	[39, 78, 156, 312, 156, 78, 39]	
	(E)	9	[39, 78, 156, 312, 624, 312, 156, 78, 39]	
	(F)	3	[39, 33, 39]	
	(G)	5	[39, 33, 30, 33, 39]	
	(H)	7	[39, 33, 30, 27, 30, 33, 39]	
	(I)	9	[39, 33, 30, 27, 24, 27, 30, 33, 39]	
	(J)	3	[39, 78, 39]	$1e-5$
	(K)	5	[39, 78, 156, 78, 39]	
	(L)	7	[39, 78, 156, 312, 156, 78, 39]	
	(M)	9	[39, 78, 156, 312, 624, 312, 156, 78, 39]	
	(N)	3	[39, 33, 39]	
	(O)	5	[39, 33, 30, 33, 39]	
	(P)	7	[39, 33, 30, 27, 30, 33, 39]	
	(Q)	9	[39, 33, 30, 27, 24, 27, 30, 33, 39]	

$$Accuracy = \frac{TP+TN}{TF+FN+FP+TN} \quad (1)$$

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

$$Precision = \frac{TP}{TF+FP} \quad (3)$$

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (4)$$

TABLE IV. SYSTEM SPECIFICATION

Device	Specification
OS	Ubuntu 18.04.1
CPU	Intel(R) Xeon(R) E3-1275 3.60GHz
RAM	pc4-2133 16GB * 2ea
GPU	NVIDIA RTX 2080 super 8G

The strides size of all 1D-CNN layers is 1 In Figure 3, the number of units in the two dense layers located on the right side of the model is 256 and 128, respectively. And the rightmost Dense layer has one unit and performs binary classification using the sigmoid activation function.

IV. EXPERIMENTS SETUP AND EVALUATION RESULTS

A. Experimental Setup

We perform experiments as shown in Table 3 to evaluate and analyze the performance of the proposed SSAE-DeepCNN model. We compare the test scenarios according to the feature vectors generated by SSAE and the classification performance of the proposed model. The values of each regularization factor (λ) are $1e-3$ and $1e-5$. For all regularization factors, we experiments by change the number and units of hidden layers constituting SSAE. We measure the accuracy and F1-Score to compare the performance of each test scenario. Also, we measure prediction time, GPU usage, memory usage, and CPU utilization to compare efficiency. The method of calculating the accuracy and F1-Score is as Equations (1)–(4). True positive (TP) is a correct prediction of a positive as positive, and false positive (FP) is an incorrect prediction of an negative answer as positive. True negative (TN) means a correct prediction that a negative is negative and false negative (FN) means an incorrect prediction that a positive is negative. Table 4 shows the specification of the system for experiments.

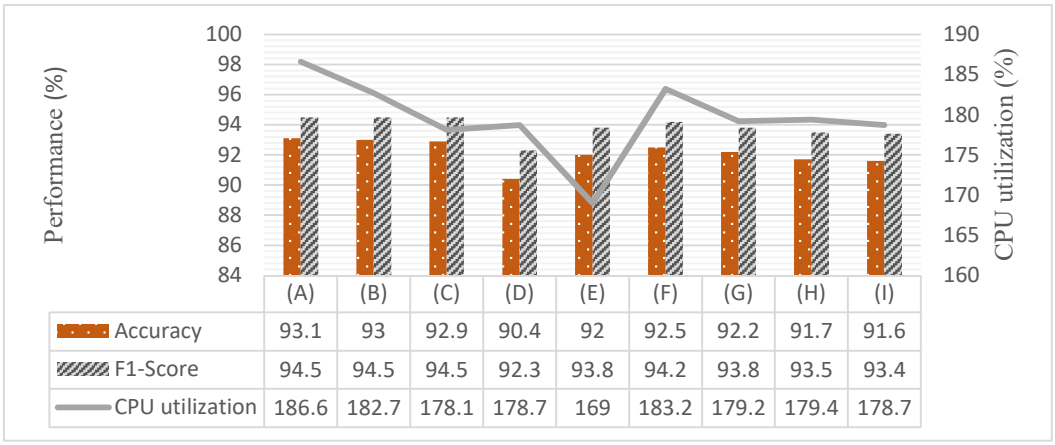
B. Experimental Comparison

Figures 4 and 5 show the experimental results according to the structure of SSAE. Figure 4 shows the experimental results when the regularization factor is $1e-3$. In Figure 4 (a), test scenario (A) achieved the best accuracy with 93.1%, and test scenario (D) performed the lowest accuracy with 90.4%. As for the F1-Score, the test scenarios (A), (B), and (C) achieved the highest at 94.5%, and the test scenario (D) performed the lowest at 92.3%. As for the CPU utilization, test scenario (A) used the most with 186.6% and test scenario (E) used the least with 169%. We measured CPU utilization using python library[1] . If the process operates multiple threads on the different CPU cores, the CPU utilization can be over 100%.

In Figure 4 (b), the prediction time of the test scenario (A) showed the fastest at 4.5 seconds and the test scenario (E) showed the slowest at 6.2 seconds. Figure 4 (c) shows the GPU usage and memory usage. The test scenario (C) used the GPU most with 6,755MB, and the test scenario (I) used the GPU least with 695MB. As for the memory usage, test scenario (E) used the most at 123.4 MB, and test scenario (H) used the least at 0.5MB.

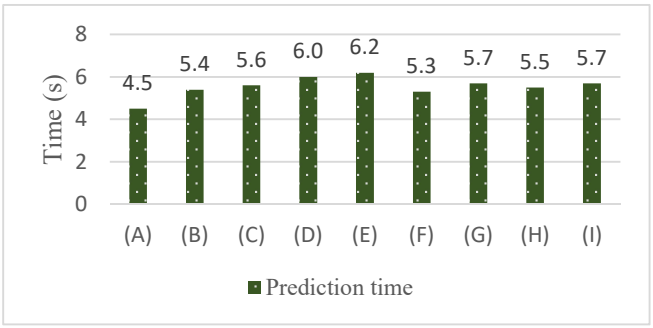
Figure 5 shows the experimental results when the regularization factor is $1e-5$. In Figure 5 (a), test scenario (M) achieved the best accuracy with 93.5%, and test scenario (P) performed the lowest accuracy with 91.4%. As for the F1-Score, the test scenario (M) achieved the highest with 94.5%, and the test scenario (P) performed the lowest with 93.1%. As for the CPU utilization, test scenario (A) used the most with

[1]psutil, https://psutil.readthedocs.io/

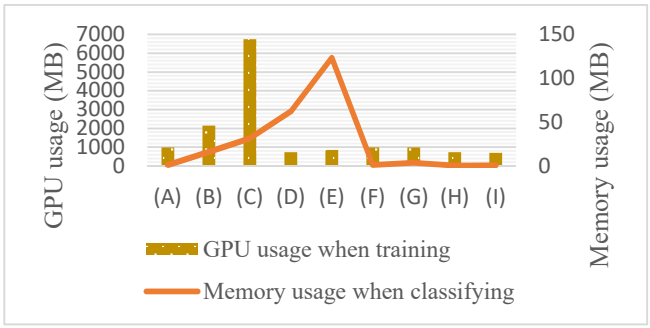


	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)	(I)
Accuracy	93.1	93	92.9	90.4	92	92.5	92.2	91.7	91.6
F1-Score	94.5	94.5	94.5	92.3	93.8	94.2	93.8	93.5	93.4
CPU utilization	186.6	182.7	178.1	178.7	169	183.2	179.2	179.4	178.7

(a)

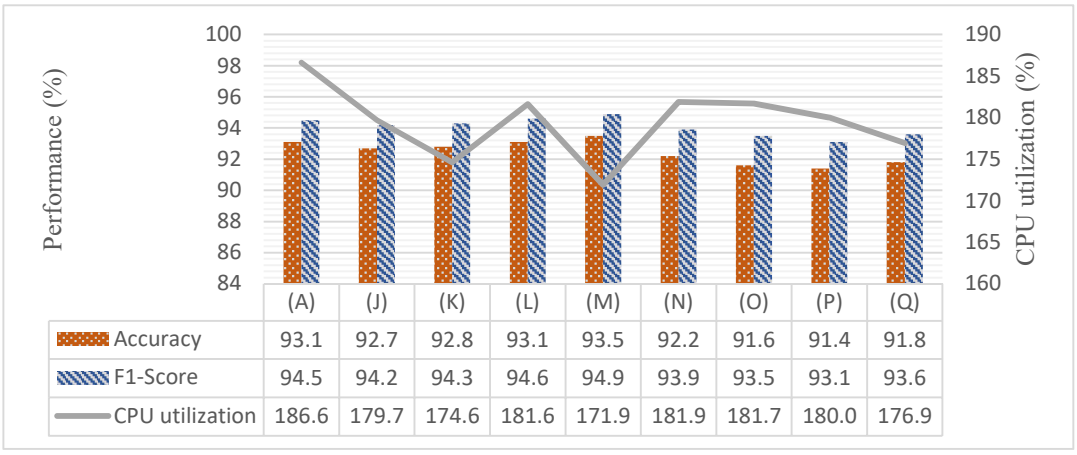


(b)



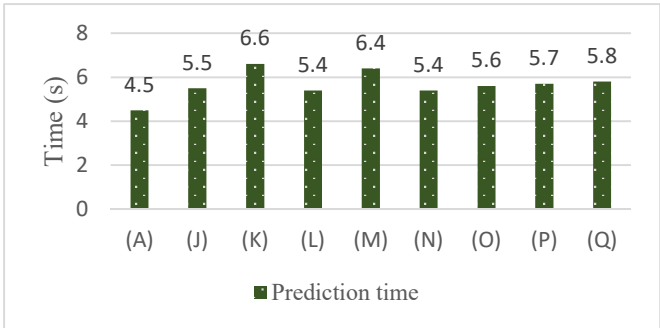
(c)

Fig. 4. Experimental results of 9 test scenarios ($\lambda = 1e-3$) including DeepCNN; (a) The results of the test scenarios (A) – (I); (b) The prediction time of the test scenarios (A)–(I); (c) Hardware resources usage of the test scenarios (A)–(I).

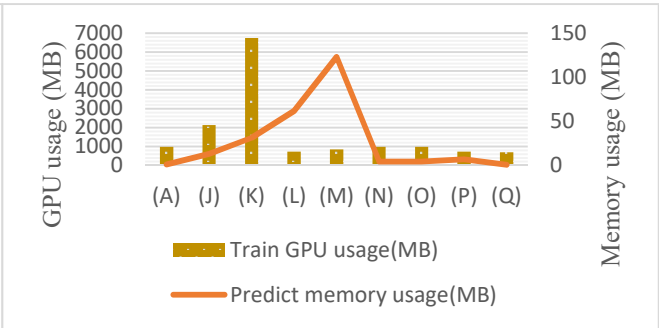


	(A)	(J)	(K)	(L)	(M)	(N)	(O)	(P)	(Q)
Accuracy	93.1	92.7	92.8	93.1	93.5	92.2	91.6	91.4	91.8
F1-Score	94.5	94.2	94.3	94.6	94.9	93.9	93.5	93.1	93.6
CPU utilization	186.6	179.7	174.6	181.6	171.9	181.9	181.7	180.0	176.9

(a)



(b)



(c)

Fig. 5. Experimental results of 9 test scenarios ($\lambda = 1e-5$) including DeepCNN; (a) The results of the test scenarios (A) and (J)¬(Q); (b) The prediction time of the test scenarios (A) and (J)¬(Q); (c) Hardware resources usage of the test scenarios (A) and (J)¬(Q);

186.6% and test scenario (M) used the least with 171.9%. In Figure 5 (b), the prediction time of the test scenario (A) showed the fastest at 4.5 seconds and the test scenario (K) showed the slowest at 6.6 seconds. Figure 5 (c) shows the GPU usage and memory usage. The test scenario (K) used the GPU most with 6,755MB and the test scenario (Q) used the GPU least with 695MB. As for the memory usage, the test the test scenario (M) used the most at 123.2MB, and the test scenario (Q) used the memory least with 0.8MB.

We found an interesting correlation between performance and resource consumption according to the number of hidden units through experiments. If the number of hidden units is smaller than the number of features in the original data, the accuracy of the model decreases. However, it was able to save the hardware resources. On the one hand, whenever the number of hidden units doubled, the accuracy increased while the CPU utilization decreased. Through the test scenarios (D)–(I) and (L)–(Q), the GPU usage used more according to the number of hidden units. However, the model with the number of 624 hidden units consumed the GPU usage more than the model with the number of 36 hidden units. However, we found interesting results when the number of hidden units was 156, and the GPU usage increased about eight times compared to when there were 624 hidden units. Also, we found that the memory usage of a model with SSAE used more than without it. Through experimental results, SSAE consumes memory usage more. Instead, it helps to save GPU and CPU resources. In addition, the new feature vectors that SSAE generated can help to improve the classification accuracy of the model. Also, when the regularization factor was $1e-5$, the model performance was higher than that of $1e-3$. Therefore, it is important to find an appropriate SSAE regularization factor for improving model performance.

V. CONCLUSION

In this paper, we proposed the SSAE-DeepCNN model that efficiently detects network intrusion threats using the UNSW-NB15 dataset. SSAE-DeepCNN model creates new feature vectors of training data with SSAE, and DeepCNN learns the new feature vectors to classify normal packets or abnormal packets. In the experimental result, SSAE generated significantly new feature vectors, where the number of hidden units increases more than decrease. This is because SSAE created new feature vectors that the model can classify a new data by adding sparsity constraints to the original data. DeepCNN effectively learned and classified new feature vectors generated by SSAE using multiple filters. Therefore, in this paper, SSAE transmits new feature vectors composed of significant features to DeepCNN by increasing the dimension of the input data and adding sparsity. When the regularization factor is $1e-5$, SSAE-DeepCNN showed the highest accuracy of 93.5% and F1-Score of 94.9%. The performance of the SSAE-DeepCNN model was slightly higher than that of a DeepCNN model. In addition, we found an interesting correlation between performance and hardware consumption due to the structure of SSAE.

We plan to study a our model to enable multi-classification by considering various problems such as a method to appropriately preprocess categorical data excepted in this study or an oversampling method to solve the unbalanced data problem. In addition, we plan to evaluate the performance of the model not only with the UNSW-NB15 dataset but also with various and comprehensive training datasets based on user profiles for network intrusion detection.

ACKNOWLEDGMENT

This research is a basic research project carried out with the support of the National Research Foundation of Korea with funding from the government (Ministry of Science and ICT) in 2020. (NRF-2020R1A2C1012117).

REFERENCES

[1] Majjed Al-Qatf, Mohammed Al-Habib and kamal Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for netowk intrusion detection," *Journal of IEEE Access*, Vol.6, pp. 52843-52856, Sept. 2018.

[2] Yinhao Xiao, Yizhen Jia, Chunchi Liu, Xiuzhen Cheng, Jiguo Yu and Weifeng Lv, "Edge computing security: state of the art and challenges," *Journal of Proceedings of the IEEE*, Vol. 107, No. 8, pp. 1608-1631, Aug. 2019.

[3] Divaya Kapil, Nidhi Mehra, Atika Gupta, sudhanshu Maurya and Anupriya Sharma, "Network security: threat model, attacks, and IDS using machine learning," In *Proc of The International Conference on Artifical Intelligence and Smart Systems*, Coimbatore, India, pp. 203-208, Mar. 2021.

[4] Sydney Mambwe Kasongo and Yanxia Sun, "A deep learning method with filter based feature engineering for wireless intrusion detection system," *Journal of IEEE Aceess*, Vol. 7, pp. 38597-38607, Mar. 2019.

[5] Iftikhar Ahmad, Mohammad Basheri, Muhammad Javed Iqbal and Aneel Rahim, "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection," *Journal of IEEE Access*, Vol. 6, pp. 33789-33795, May 2018.

[6] Ram kumar Singh and T. Ramajujam, " Intursion detection system using advanced honeypots," *Journal of International Journal of Computer Science and Information Security*, Vol. 2, No. 1, pp. 1-9, June 2009.

[7] Ziadoon Kamil Maseer, Robiah Yusof Nazrulazhar Bahaman, Salama A. Mostafa and Cik Feresa Mohd Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset," *Journal of IEEE Access*, Vol 9, pp. 22351-22370, Feb. 2021.

[8] Jesper E. van Engelen and Holger H. Hoos, "A survey on semi-supervised learning," *Journal of Machine Learning,* Vol. 109, No. 2, pp.373-440, Nov. 2019.

[9] GwiHoon Kim and YongGeun Hong, "Machine learning technology trends in the network," *Journal of The Korean Institute of Communication Sciences*, Vol. 34, No. 10, pp. 38-44, Sept. 2017.

[10] Prachiti Parkar and Ansh Bilimoria, "A survey on cyber security IDS using ML methods," In *Proc. of 5th International Conference on Intelligent Computing and Control Systems*, Madurai, India, pp.352-360 , May 2021.

[11] Joohwa Lee, JuGeon Pak and Myungsuk Lee, "Netowrk intrusion detection system using feature extraction based on deep sparse autoencoder," In *Proc. of 2020 International Conference on information and Communication Technology Convergence*, Jeju, South Korea, pp. 21-23, Oct. 2020.

[12] Baoan Zhang, Yanhua Yu and Jie Li, "Network intrusion detection bsed on stacked sparse autoencoder and binary tree ensemble method," In *Proc. of 2018 IEEE International Conference on Communications Workshops*, Kansas City, MO, USA, pp. 1-6, May 2018.

[13] Andrew NG, "Sparse autoencoder," CS294A Lecutre notes, Vol. 72, pp. 1-19, 2011.

[14] Meliboev Azizjon, Alikhanov Jurnabek and Wooseong Kim, "1D CNN based network intrusion detection with normalization on imbalanced data," In *Proc. of 2020 International Conference on Artificial Intelligence in Information and Communication*, pp. 218–224, Fukuoka, Japan, Apr. 2020.

[15] Peilun Wu and Hui Guo, "LuNet: a deep nerual network for network intrusion detection," In *Proc. of 2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 617–624, Xiamen, China, Dec. 2019.