

Digital Object Identifier

# Defending Deep Neural Networks against Backdoor Attack by Using De-trigger Autoencoder

HYUN KWON<sup>1</sup>

<sup>1</sup>Department of Electrical Engineering, Korea Military Academy, Seoul 01819, South Korea

Corresponding author: Hyun Kwon (e-mail: hkwon.cs@gmail.com or khkh@kaist.ac.kr).

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2021R111A1A01040308) and Hwarang-Dae Research Institute of Korea Military Academy.

**ABSTRACT** A backdoor attack is a method that causes misrecognition in a deep neural network by training it on additional data that have a specific trigger. The network will correctly recognize normal samples (which lack the specific trigger) as their proper classes but will misrecognize backdoor samples (which contain the trigger) as target classes. In this paper, I propose a method of defense against backdoor attacks that uses a de-trigger autoencoder. In the proposed scheme, the trigger in the backdoor sample is removed using the de-trigger autoencoder, and the backdoor sample is detected from the change in the classification result. Experiments were conducted using MNIST, Fashion-MNIST, and CIFAR-10 as the experimental datasets and TensorFlow as the machine learning library. For MNIST, Fashion-MNIST, and CIFAR-10, respectively, the proposed method detected 91.5%, 82.3%, and 90.9% of the backdoor samples and had 96.1%, 89.6%, and 91.2% accuracy on legitimate samples.

**INDEX TERMS** Backdoor attack, Defense method, Deep neural network, De-trigger autoencoder

## I. INTRODUCTION

Deep neural networks [1] provide good performance in the fields of image recognition [2], speech recognition [3], pattern analysis [4], and intrusion detection [5], which are typical machine learning tasks. However, deep neural networks have security vulnerabilities. The security risks of machine learning have been categorized by Barreno et al. [6] into those from exploratory attacks and those from causative attacks. An exploratory attack [7] is an attack that induces misrecognition by a model that has already been trained, caused by the manipulation of test data. An adversarial example [8] [9] [10] [11] is a type of exploratory attack. A causative attack [12] is an attack that degrades the accuracy of a model by adding malicious artificial data during the model's training process. Causative attacks include poisoning attacks [13] [14] and backdoor attacks [15] [16] [17]. The method proposed in this paper is designed as a method of defense against the backdoor type of causative attack.

The backdoor attack [15] [18] originated as an improvement of the poisoning attack. The conventional poisoning attack reduces the accuracy of a model by adding malicious data to the model's training data. With this method, however, it is not possible for the attacker to specify the time of attack;

additionally, it is easy to determine whether such an attack has been performed by checking the validity of some of the data. To overcome the disadvantages of the poisoning attack, the backdoor attack was proposed, by which a trigger is attached to data to induce misrecognition by the model; data lacking the trigger will be recognized correctly. This method trains the model by adding a backdoor sample to the training data so that data containing the trigger will not be correctly recognized. After being trained, the model will correctly recognize data that do not contain the trigger but will not correctly recognize backdoor samples that contain the trigger. With a backdoor attack, the attacker can specify the time of attack by means of the trigger, and because the model's overall accuracy remains high, it is not easy to detect the attack by performing a validation check.

Investigators have researched three methods of defense against backdoor attacks: reversing the trigger [19], changing the structure of the target model [20], and changing the classification [21] [22] by adding a separate model. The reverse trigger method analyzes the classification score according to trigger positions set randomly. This approach works because a model attacked by a backdoor sample is highly likely to misrecognize the sample for some specific trigger

location. However, as this method has a high probability of misrecognizing data because of the presence of a trigger, it has a high rate of false detections. The second method, which detects backdoor samples by changing the structure of the target model, reduces the effectiveness of the backdoor attack by removing specific neurons in the target model. However, the removal of specific neurons in the target model can reduce its accuracy on the original samples. The third method, which detects backdoor samples using two separate models, compares changes in classification results between the model trained on the original samples and the model trained on the backdoor sample. However, as training each of the two models on clean samples and on the backdoor sample requires verification of the training data by human feedback, the assumption it is based on is somewhat less realistic, and a complex process is required.

In this paper, a de-trigger backdoor defense method is proposed. This method detects backdoor samples using a proposed de-trigger autoencoder in which triggers are arbitrarily added to input data. The method removes the trigger of the backdoor attack in the de-trigger module; therefore, a legitimate sample will not be changed by passing through the module, but a backdoor sample will change, allowing the backdoor sample to be detected by the proposed scheme. The contributions of this paper are as follows. First, the proposed method for detecting a backdoor sample by removing its trigger using a de-trigger autoencoder is described and explained. In the proposed method, an autoencoder module removes the trigger on the backdoor sample. The technique is original in that it can restore the backdoor to the legitimate sample, in contrast to existing methods of defense against backdoor attacks. In particular, unlike the existing autoencoder method, the de-trigger autoencoder is designed to restore an image with an arbitrary trigger to the legitimate sample so that the trigger on the backdoor sample can be removed. Second, it is shown how the proposed method analyzes the images after removal of the trigger for the legitimate sample and backdoor sample using the de-trigger autoencoder. In addition, detection rates using different de-trigger autoencoders are compared, and the recognition rates on legitimate samples are analyzed. Third, the performance of the proposed method is demonstrated using the MNIST [23], Fashion-MNIST [24], and CIFAR-10 [25] datasets.

The remainder of the paper is organized as follows. In Section II, the proposed method is briefly described, and related research is reviewed. In Section III, the proposed scheme is explained in detail. Sections IV and V discuss the evaluation experiment and its results, and Section VI further discusses the proposed scheme. Finally, Section VII concludes the paper.

## II. RELATED WORK

Legitimate data without a trigger are recognized correctly by the model; a backdoor attack is an attack in which data containing the trigger are not correctly recognized by the model. This section describes the concept of the autoencoder and

describes related research on attack methods and methods of defense against backdoor samples.

### A. AUTOENCODER CONCEPT

An autoencoder [26] is a neural network for unsupervised learning that copies inputs to outputs. The autoencoder has the same structure as the general multilayer perceptron (MLP) except that the number of neurons in the input and output layers is the same. The loss function is calculated using the difference between the input and the output. Autoencoders reduce the number of dimensions by using fewer neurons in the hidden layer than in the input layer, or they may train the network to restore the legitimate input after adding noise to the input data. These constraints prevent the autoencoder from simply copying the input directly to the output and control it so that it learns how to represent data efficiently.

### B. BACKDOOR ATTACK METHODS

The backdoor sample is an attack in which the triggered data will be misrecognized by the model. The backdoor sample attack was proposed by Gu et al. [15] in the BadNet method. In this method, a specific trigger is attached to the image, and it is not correctly recognized by the model. The method achieved an attack success rate of 99% against MNIST. Instead of adding malicious data, Liu et al. [27] added external neural networks to create a new method of attack. Their method attacks by adding a neuron to the neural network and causes the data to which a specific trigger is attached to be misrecognized by the model. Wang et al. [28] proposed a detection method that operates by reversing the trigger. In this method, various triggers are attached to analyze the backdoor attack method. Clements and Lao [29] proposed a method for causing malfunction by planting a backdoor in the hardware of a neural network. When this method was used on the MNIST dataset and the wrong neuron was deliberately added to the neural network, the backdoor was misrecognized by the model because of the presence of a specific trigger.

### C. DEFENSE METHODS AGAINST BACKDOOR ATTACK

Methods of defense against backdoor samples can be categorized into those that use reverse trigger detection on input data, those that detect backdoor samples by changing the structure of the target model, and those that add a separate model and analyze changes in classification. The first of these detects whether a model has been attacked by a backdoor sample by attaching a trigger to the input data. For example, Xiang et al. [19] proposed a reversal method that uses the change in classification result in the target model after reflecting the input data through multiple triggers. Wang et al. [28] proposed the neural cleanse method, a reversal method that detects changes in classification by using multiple triggers on input data. A sample is identified as a backdoor sample when a certain metric exceeds a specific criterion by using the outline method for the detected classification. Gao et al. [30] proposed a strip method for detecting backdoor samples. This

real-time method tests whether the target model has been subjected to a backdoor sample by inserting several trigger patterns into the input data and evaluating the difference in entropy between the clean sample and the backdoor sample. The second type detects backdoor samples by changing the structure of the target model. For example, Liu et al. [20] proposed the fine-pruning method, which identifies clean samples and backdoor samples by removing specific neurons in the structure of the target model. This method detects backdoor samples by taking advantage of the fact that a backdoor sample is sensitive to classification changes induced by the removal of a specific neuron from the target model. With a clean sample, the classification result does not change significantly even if the specific neuron of the target model is removed, but with a backdoor sample, a relatively large change in the classification result is caused by the removal of the specific neuron. The third type detects backdoor samples through classification changes that may occur when a separate model is added. For example, Weber et al. [21] proposed the “RAB” method, which detects backdoor samples by comparing the classification results produced by two models: the model trained using the backdoor sample and a model trained on clean data. If the input value is clean data, the two models will provide the same classification result. If the input data is a backdoor sample, however, the model trained on the clean data will classify it as the normal class, but the model trained on the backdoor sample will classify it incorrectly; thus, the backdoor sample can be identified using this classification difference.

The above methods require the insertion of a specific trigger into the input data (for the reversal process), changing the structure of the model, constructing safe training data, and/or conducting additional training of the model. Changing the structure of the model or inserting a specific trigger into the input data requires considerable time, many iterations, and a complex process. Training each of two models on clean samples and on the backdoor sample requires verification of the training data by human feedback, which is somewhat unrealistic and involves a complex process. Unlike the above methods, however, the proposed method first applies the de-trigger autoencoder, which removes the trigger from the backdoor sample. The de-trigger autoencoder reduces the effectiveness of the attacking backdoor samples by removing the trigger pattern, and it is easy to detect backdoor samples by using the change in the classification results. In addition, the proposed method does not require access to the entire dataset and does not require changing the structure of the model or reversing a specific trigger pattern.

### III. PROPOSED METHOD

Figure 1 shows how the proposed method detects backdoor samples by the change in the classification result after the input data are passed through the de-trigger autoencoder. In the case of a legitimate sample, the classification results produced before and after the sample is passed through the de-trigger autoencoder are the same. In the case of a backdoor

sample, however, the classification results produced before and after the sample is passed through the de-trigger autoencoder are different. This is because the backdoor sample, whose trigger has been removed, is restored to the legitimate data, and the classification result given by the target model is the proper class.

The proposed method consists of two processes: generating a de-trigger autoencoder and detecting a backdoor sample. First, a de-trigger autoencoder is created so that an image consisting of legitimate data with a trigger attached is output as data without the trigger. Then, the backdoor sample is detected by using the change in the recognition result between the original data and the backdoor sample through the de-trigger autoencoder. The result value given by the target model and the result value given by the input detection model are compared. If these values are different, it is regarded as a backdoor sample, and the target model is deemed to be under a backdoor attack.

The mathematical procedure for the proposed method is as follows. In the first step, in which the de-trigger autoencoder is created, a specific trigger is added to the input data, and the de-trigger autoencoder is trained so that it will output the input data with the trigger removed when it is given a randomly generated backdoor sample as input. When the input is  $x$ , the encoder network function  $f_{\theta}(\cdot)$ , and the decoder network function  $g_{\theta'}(\cdot)$ , the trigger-reflected backdoor sample  $\hat{x}$  is mapped to the hidden latent layer as follows:

$$y = f_{\theta}(\hat{x}) = s(W\hat{x} + b),$$

where  $W$  is the weight value,  $b$  is a bias constant, and  $s$  is an activation function. In the decoder, hidden latent variable  $y$  can generate  $g_{\theta'}(y)$ :

$$z = g_{\theta'}(y).$$

The encoder network parameter  $\theta$  and decoder network parameter  $\theta'$  are trained to have a minimal reconstruction error as follows:

$$L_2(x, z) = \|z - x\|_2^2.$$

The de-trigger autoencoder, which has been trained on a large quantity of data using this process, plays the role of restoring the data containing a specific trigger to the legitimate data with the trigger removed. The trigger is created as a white pattern in the shape of a  $4 \times 4$  square, and it is positioned randomly at the top right, bottom right, top left, or bottom left of the image.

In the second step, in which the backdoor sample is detected, the change in the classification of the input after it is passed through the de-trigger autoencoder is examined. Let the operation function of the de-trigger autoencoder be  $f_p$  and the operation function of the target model be  $f_t$ . Then

$$f_t(x_v) = r_b \text{ and } f_t(f_p(x_v)) = r_a,$$

where  $x_v$  is the data being validated,  $r_b$  is the classification result before the de-trigger autoencoder, and  $r_a$  is the classification result after the de-trigger autoencoder. If the

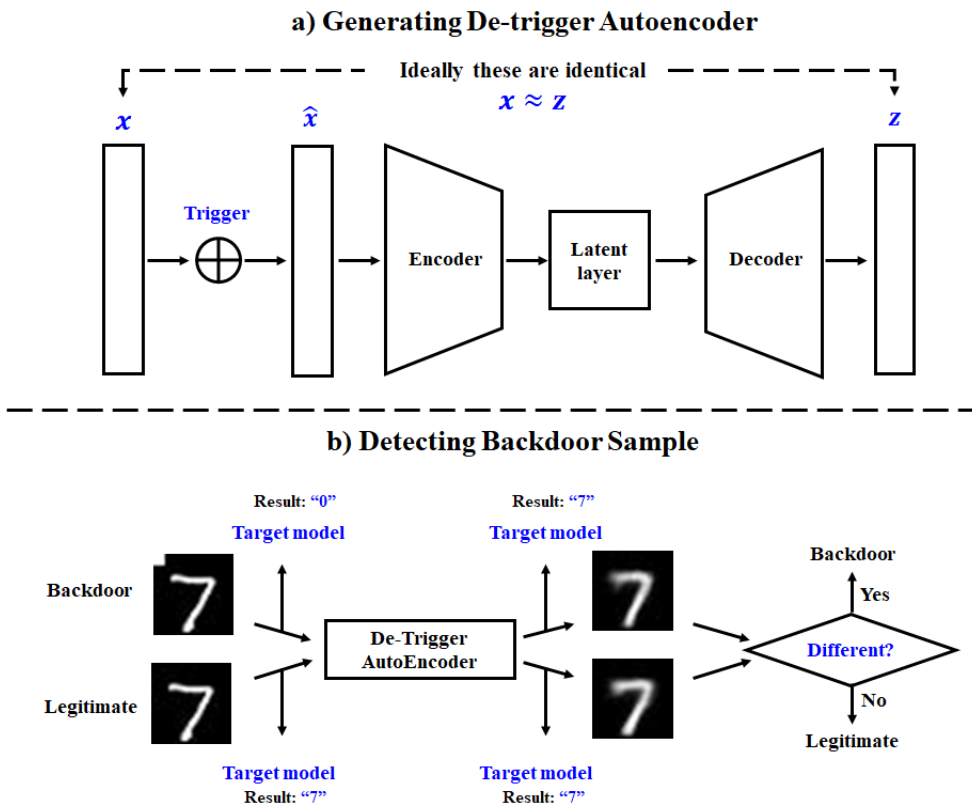


FIGURE 1: Overview of the proposed scheme.

classification results are the same before and after the input is passed through the de-trigger autoencoder, the input value is a legitimate sample, and if the classification results are different, the input value is regarded as a backdoor sample. Details of the algorithm are shown in Algorithm 1.

**Algorithm 1** De-trigger (DT) autoencoder for detecting a backdoor attack

**Input:** legitimate training data  $x \in X$ , new input data  $x_v$ , function of the target model  $f_d$ , de-trigger autoencoder  $f_p$

**Detecting backdoor attack:**

Process for training DT autoencoder ( $f_p$ ) on  $x \in X$

$r_b \leftarrow f_d(x_v)$

$r_a \leftarrow f_d(f_p(x_v))$

**if**  $r_b \neq r_a$  **then**

    flag  $\leftarrow$  0

**else**

    flag  $\leftarrow$  1

**end if**

**return** flag

**IV. EXPERIMENT SETUP**

Through experiments, it was demonstrated that the proposed method can effectively defend against backdoor attacks. In the experiments, the machine learning library used was Ten-

sorFlow [31], and the hardware was a Xeon E5-2609 1.7-GHz server.

**A. DATASETS**

MNIST, Fashion-MNIST, and CIFAR-10 were used as experimental datasets. MNIST is a dataset of handwritten images consisting of the numerals from 0 to 9 in black and white. The images in MNIST are two-dimensional data of  $28 \times 28 \times 1$  pixels, totaling 784 pixels. MNIST has 60,000 training data and 10,000 test data. Fashion-MNIST is a fashion-related dataset and consists of 10 types of image, such as T-shirts, bags, ankle boots, coats, shirts, sneakers, sandals, and dresses. The images in Fashion-MNIST are two-dimensional data of  $28 \times 28 \times 1$  pixels, totaling 784 pixels. Fashion-MNIST has 60,000 training data and 10,000 test data. CIFAR-10 is a color object image dataset and is composed of 10 types of data, such as planes, dogs, cats, cars, trucks, and deer. The images in CIFAR-10 are three-dimensional data of  $32 \times 32 \times 3$  pixels, totaling 3072 pixels. In CIFAR-10, there are 50,000 training data and 10,000 test data.

**B. TARGET MODELS**

As the target model for MNIST and Fashion-MNIST, a convolutional neural network [32] was constructed. For CIFAR-10, the VGG model [33] was used as the target model. The structures of the target models are shown in Table 8 and Table

9 in the appendix, and their parameters are given in Table 10 in the appendix. The Adam algorithm [34] was used as the optimization algorithm for all of the target models.

### C. STRUCTURE OF DE-TRIGGER AUTOENCODER

The de-trigger autoencoder is composed of an encoder, latent layer, and decoder. The encoder consists of two layers, one having 250 neurons and the other having 100 neurons. The latent layer consists of 20 neurons. The decoder is composed of 100 and 250 neurons as two layers. The parameters of the de-trigger autoencoder were set to 0.001 for the learning rate, 20 for the number of epochs, and 128 for the batch size. ReLU [35] was used as the activation function, and Adam [34] as the optimization algorithm.

### D. CREATION OF BACKDOOR SAMPLES

Each backdoor sample has a white square trigger at the top left of the legitimate image. A backdoor sample with the trigger attached will be misrecognized as the target class by the target model. The generated backdoor samples constituted approximately 10% of the total training data. These samples are used as additional training data for the target model.

## V. EXPERIMENTAL RESULTS

Accuracy is the percentage of matches between the class recognized by the target model and the proper class for the input data. The detection rate is the rate calculated using the change in classification when the backdoor samples are passed through the de-trigger autoencoder.

Table 1 shows the class results for legitimate samples from MNIST and their corresponding backdoor samples; it also shows the backdoor samples after they were passed through the de-trigger autoencoder. The trigger was specified as a white square in the upper left corner, and the target class was set to "0." As shown in the table, the target model correctly recognized the legitimate data and misrecognized the backdoor samples having the specified trigger as the target class. This misrecognition occurred because the target model was additionally trained on the backdoor samples. The backdoor samples that were passed through the de-trigger autoencoder were correctly recognized by the target model because the trigger had been removed. The backdoor samples were not fully restored to match the legitimate data, but the similarity was sufficient to allow their recognition as legitimate data.

Table 2 shows the class results for legitimate samples from Fashion-MNIST and their corresponding backdoor samples; it also shows the backdoor samples after they were passed through the de-trigger autoencoder. The trigger was specified as a white square in the upper left corner, and the target class was set to "T-shirt." As with MNIST, the legitimate data without the specified trigger were correctly recognized by the target model, and the backdoor samples with the trigger were misrecognized as the target class by the model. In addition, the backdoor samples that were passed through the de-trigger autoencoder were properly recognized by the

target model because the trigger had been removed and the samples restored to a version similar to the legitimate data.

Table 3 shows the class results for legitimate samples from CIFAR-10 and their corresponding backdoor samples; it also shows the backdoor samples after they were passed through the de-trigger autoencoder. The trigger was specified as a white square in the upper left corner, and the target class was set to "Plane." As shown in the table, the legitimate data without the specified trigger were recognized normally by the target model, and the backdoor samples with the trigger were misrecognized as the target class by the model. In addition, because the trigger was removed by the de-trigger autoencoder and the backdoor samples became similar to the legitimate data, the backdoor samples that were passed through the de-trigger autoencoder were correctly recognized by the target model.

Table 4 shows the classification scores for a backdoor sample derived from MNIST before and after being passed through the de-trigger autoencoder. The digit with the highest value among the classification scores is recognized as the class of the input value. Before the backdoor sample was passed through the de-trigger autoencoder, its highest classification score was 12.62, corresponding to the first of the 10 possible classes, and so it was misrecognized as the numeral 0. However, after it was passed through the de-trigger autoencoder, its highest classification score was 13.06, corresponding to the eighth of the 10 possible classes, and so it was correctly recognized as the numeral 7. Because the trigger on the backdoor sample was removed by the de-trigger autoencoder, the backdoor sample was then correctly recognized as the proper class.

Table 5 shows the classification scores for a backdoor sample derived from Fashion-MNIST before and after being passed through the de-trigger autoencoder. As shown in the table, the highest classification score for the backdoor sample before it was passed through the de-trigger autoencoder was 14.7, corresponding to the first of the classes, and so it was misrecognized as a T-shirt (Class 0). However, after it was passed through the de-trigger autoencoder, its highest classification score was 19.6, corresponding to the sixth of the classes, and so it was recognized as a sandal (Class 5). Because the trigger on the backdoor sample was removed by the de-trigger autoencoder, the backdoor sample was then correctly recognized as the proper class.

Table 6 shows the classification scores for a backdoor sample derived from CIFAR-10 before and after being passed through the de-trigger autoencoder. As shown in the table, the highest classification score for the backdoor sample before it was passed through the de-trigger autoencoder was 9.13, corresponding to the first of the classes, and so it was misrecognized as a plane (Class 0). However, after it was passed through the de-trigger autoencoder, its highest classification score was 12.49, corresponding to the third of the classes, and so it was recognized as a bird (Class 2). Because the trigger on the backdoor sample was removed by the de-trigger autoencoder, the backdoor sample was then correctly

TABLE 1: Class results for legitimate samples from MNIST, their corresponding backdoor samples, and the backdoor samples after being passed through the de-trigger autoencoder. The trigger on the backdoor samples is a white square in the top left corner. "Proposed module" is the de-trigger autoencoder.




























Class	"1"	"2"	"3"	"4"	"5"	"6"	"7"	"8"	"9"
Legitimate sample									
Class	"0"	"0"	"0"	"0"	"0"	"0"	"0"	"0"	"0"
Backdoor before proposed module									
Class	"1"	"2"	"3"	"4"	"5"	"6"	"7"	"8"	"9"
Backdoor after proposed module									

TABLE 2: Class results for legitimate samples from Fashion-MNIST, their corresponding backdoor samples, and the backdoor samples after being passed through the de-trigger autoencoder. The trigger on the backdoor samples is a white square in the top left corner. "Proposed module" is the de-trigger autoencoder.






















Class	"Trouser"	"Sneaker"	"Ankle boot"	"Sandal"	"Bag"	"Dress"	"Pullover"
Legitimate sample							
Class	"T-shirt"	"T-shirt"	"T-shirt"	"T-shirt"	"T-shirt"	"T-shirt"	"T-shirt"
Backdoor before proposed module							
Class	"Trouser"	"Sneaker"	"Ankle boot"	"Sandal"	"Bag"	"Dress"	"Pullover"
Backdoor after proposed module							

TABLE 3: Class results for legitimate samples from CIFAR-10, their corresponding backdoor samples, and the backdoor samples after being passed through the de-trigger autoencoder. The trigger on the backdoor samples is a white square in the top left corner. "Proposed module" is the de-trigger autoencoder.





















Class	"Truck"	"Bird"	"Cat"	"Deer"	"Ship"	"Frog"	"Horse"
Legitimate sample							
Class	"Plane"	"Plane"	"Plane"	"Plane"	"Plane"	"Plane"	"Plane"
Backdoor before proposed module							
Class	"Truck"	"Bird"	"Cat"	"Deer"	"Ship"	"Frog"	"Horse"
Backdoor after proposed module							

TABLE 4: Classification scores for a backdoor sample derived from MNIST (“7” → “0”) before and after being passed through the de-trigger autoencoder. The target class of the backdoor sample was “0.” “Proposed module” is the de-trigger autoencoder.



Description	Backdoor sample (“7” → “0”)	
	Before proposed module (“0”)	After proposed module (“7”)
		
Classification scores	[12.62 -1.58 -9.73 -8.37 -10.1 -1.68 2.6 -10.1 1.67 1.53]	[3.12 -2.41 -7.69 -1.86 0.45 -1.39 -5.94 13.06 -4.17 -5.48]

TABLE 5: Classification scores for a backdoor sample derived from Fashion-MNIST (Class 5, “Sandal” → Class 0, “T-shirt”) before and after being passed through the de-trigger autoencoder. The target class of the backdoor sample was “T-shirt” (Class 0). “Proposed module” is the de-trigger autoencoder.





Description	Backdoor sample (Class 5, “Sandal” → Class 0, “T-shirt”)	
	Before proposed module (Class 0, “T-shirt”)	After proposed module (Class 5, “Sandal”)
		
Classification scores	[14.7 2.04 1.25 -4.38 -2.68 0.03 -6.23 1.79 -4.33 -5.86]	[-1.14 -6.24 -4.12 2.26 1.29 19.6 -5.20 3.54 1.77 -3.77]

TABLE 6: Classification scores for a backdoor sample derived from CIFAR-10 (Class 2, “Bird” → Class 0, “Plane”) before and after being passed through the de-trigger autoencoder. The target class of the backdoor sample was “Plane” (Class 0). “Proposed module” is the de-trigger autoencoder.

Description	Backdoor sample (Class 2, “Bird” → Class 0, “Plane”)	
	Before proposed module (Class 0, “Plane”)	After proposed module (Class 2, “Bird”)
		
Classification scores	[9.13 -1.61 0.38 1.21 -5.04 2.42 -3.03 -3.35 4.86 -1.98]	[-3.10 0.83 12.49 1.13 3.73 -4.08 -2.69 5.41 -2.92 4.34]

recognized as the proper class.

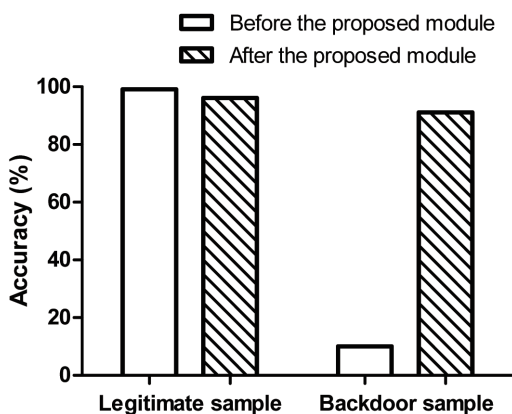


FIGURE 2: For MNIST, accuracy on the legitimate samples and backdoor samples before and after being passed through the de-trigger autoencoder. “Proposed module” is the de-trigger autoencoder.

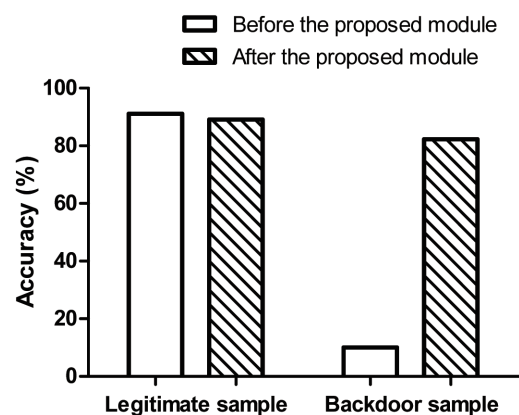


FIGURE 3: For Fashion-MNIST, accuracy on the legitimate samples and backdoor samples before and after being passed through the de-trigger autoencoder. “Proposed module” is the de-trigger autoencoder.

Figure 2 shows the model’s accuracy on the legitimate

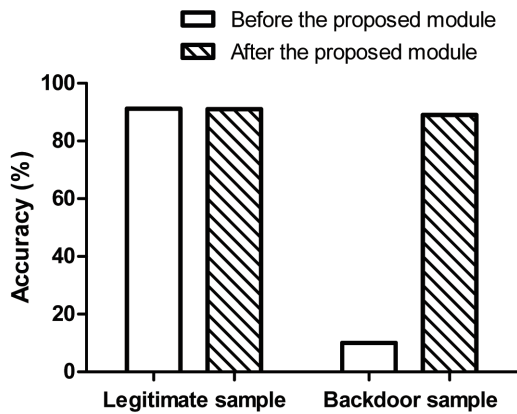


FIGURE 4: For CIFAR-10, accuracy on the legitimate samples and backdoor samples before and after being passed through the de-trigger autoencoder. “Proposed module” is the de-trigger autoencoder.

samples from MNIST and the corresponding backdoor samples before and after they were passed through the de-trigger autoencoder. As shown by the figure, there was little difference in accuracy on the legitimate samples before and after they were passed through the de-trigger autoencoder. Because the de-trigger autoencoder is a module that removes the trigger, and legitimate samples do not have a trigger, it produces little change in accuracy with these samples. The backdoor samples were misrecognized as the target class before being passed through the de-trigger autoencoder, degrading the accuracy to 10%. After these samples were passed through the de-trigger autoencoder, however, the trigger was no longer present, and so the model recognized the proper class with 91.5% accuracy. Under the proposed method, after the legitimate samples were passed through the de-trigger autoencoder, the model’s accuracy on these samples was 96.1%. The proposed method had a 91.5% detection rate for the backdoor samples because after being passed through the de-trigger autoencoder, the backdoor samples were correctly recognized with 91.5% accuracy in a scenario in which 100% of them attacked with the aim of being recognized as an incorrect class instead of as the proper class.

Figure 3 shows the model’s accuracy on the legitimate samples from Fashion-MNIST and the corresponding backdoor samples before and after they were passed through the de-trigger autoencoder. As shown by the figure, there was little difference in accuracy on the legitimate samples before and after they were passed through the de-trigger autoencoder. Because the de-trigger autoencoder is a module that removes the trigger, and legitimate samples do not have a trigger, there is little difference in the accuracy with these samples. The backdoor samples were misrecognized as the target class before being passed through the de-trigger autoencoder, degrading the accuracy to 10%. Because the trigger on the backdoor samples was no longer present after they were passed through the de-trigger autoencoder, however,

the backdoor samples were then recognized as the proper class with 82.3% accuracy. Under the proposed method, after the legitimate samples were passed through the de-trigger autoencoder, the model’s accuracy on these samples was 89.6%. Similar to MNIST, the proposed method had a detection rate of 82.3% for the backdoor samples because after being passed through the de-trigger autoencoder, the backdoor samples were correctly recognized with 82.3% accuracy in a scenario in which 100% of them attacked with the aim of being recognized as an incorrect class instead of as the proper class.

Figure 4 shows the model’s accuracy on the legitimate samples from CIFAR-10 and the corresponding backdoor samples before and after they were passed through the de-trigger autoencoder. As shown by the figure, there was little difference in accuracy on the legitimate samples before and after they were passed through the de-trigger autoencoder. Under the proposed method, the model’s accuracy on these samples was 91.24%, only slightly different from the 91.69% accuracy on the legitimate samples before they were passed through the de-trigger autoencoder. This is because the de-trigger autoencoder removes the trigger from the samples, and so the legitimate samples, which lack the trigger, are hardly changed by being passed through the de-trigger autoencoder. The backdoor samples, before being passed through the de-trigger autoencoder, were incorrectly recognized as the target class, reducing the accuracy to 10%. Because the trigger on the backdoor samples was no longer present after they were passed through the de-trigger autoencoder, 90.94% of the backdoor samples were correctly recognized.

Table 7 shows the accuracy on the legitimate samples and the detection rate for the backdoor samples for the reversal method [19], neural cleanse method [28], strip method [30], fine-pruning method [20], RAB method [21], and proposed method. With each method, CIFAR-10 was used as the dataset, and after the triggers were randomly generated for the backdoor samples, a comparison experiment was performed on 1000 backdoor samples and 1000 legitimate samples. For the reversal method, the threshold was set to 0.53. For the neural cleanse method, epsilon was set to 0.000001, the learning rate was 0.1, and the threshold was set to 0.89. For the strip method, the detection boundary was set to 0.2. For the fine-pruning method, the batch size was set to 100. In terms of accuracy, as legitimate samples have no trigger, the proposed method resulted in higher accuracy on the legitimate samples than did the other methods. Because the fine-pruning method removes neurons from the target model, its accuracy on the legitimate samples was lower than that of the other methods. In terms of detection of backdoor samples, because the proposed method removes the trigger from the backdoor sample, it is restored to the legitimate sample to the extent that it can be properly recognized, and so the detection rate for the proposed method was higher than that of the other methods. As can be seen, the RAB method has a relatively low detection rate for other types of



TABLE 7: Accuracy on the legitimate samples and detection rate for the backdoor samples under the reversal method, neural cleanse method, strip method, fine-pruning method, RAB method, and proposed method.

Metric	Trigger reversal			Model change	Two models	Trigger removal
	Reversal	Neural cleanse	Strip	Fine-pruning	RAB	Proposed
Accuracy (%)	87.92	89.13	88.27	86.34	87.34	91.18
Detection rate (%)	89.76	90.23	87.23	88.31	81.86	90.94

backdoor samples that the user is not aware of when training the two models. Thus, the proposed method detects backdoor samples by reducing the attack effectiveness of the backdoor sample, which it accomplishes by removing the trigger, in contrast to the other methods.

## VI. DISCUSSION

**Assumptions.** A necessary assumption for the proposed method is that it has access to the training data of the target model. This access enables the proposed method to detect backdoor samples among the training data. With the de-trigger autoencoder, it can also be used to determine whether the target model has suffered a backdoor attack, and to generate the de-trigger autoencoder, it requires access to the target model’s training data. This is because it is only possible to create a de-trigger autoencoder that can remove a trigger by attaching the trigger to a sample from the training data that is guaranteed to be legitimate.

The proposed method also serves as a method of defense against backdoor samples that use the specific white trigger. In some of the latest types of backdoor attacks, the attack success rate is increased by using a complex picture in the backdoor sample as a trigger pattern. However, even a backdoor sample that has a white trigger is incorrectly recognized as the target class with 100% success in an attack on a target model that lacks a defense. In addition, if the trigger on the backdoor sample is a complex picture, there is a disadvantage in that it can be easily identified by eye because of its coloration. Further, most of the latest defense methods were tested on backdoor samples using a white trigger, and therefore the same was used in the experiments of the present study.

**De-trigger autoencoder.** In the proposed method, a de-trigger autoencoder is used to remove the trigger on the backdoor sample. The de-trigger autoencoder has a neural network configured to output the legitimate input sample without a trigger by applying internal encoding, latency, and decoding procedures after first attaching the trigger to it. The trigger is attached to the upper left, upper right, lower left, or lower right at random in an area that will not interfere with the legitimate image. It is done in this way because if the trigger were to overlap with the image in the legitimate sample, it could be easily identified by eye, and the effectiveness of the attack would be reduced. The structure of the de-trigger autoencoder is an improved version of the denoising autoencoder [36] [37]. Each parameter was set as

described in Section IV-C to a value that was experimentally determined to be suitable.

**Backdoor samples.** The backdoor samples generated by the proposed method constituted 10% of the entire dataset. Because the accuracy on the legitimate samples may decrease as the proportion of backdoor samples increases, the proportion of backdoor samples was set to 10% of the entire dataset in order to maintain the accuracy on the legitimate samples and still achieve a 100% attack success rate. In the experiments, the trigger was set to a white square in the upper left corner of the backdoor sample; backdoor samples containing this trigger are misrecognized by the target model as the target class determined by the attacker. However, triggers can be created in other shapes or placed in positions other than the top left. This is because the attacker determines the shape and location of the trigger in advance and performs additional training on the target model after creating the backdoor samples.

**Datasets.** In testing the proposed method, MNIST, Fashion-MNIST, and CIFAR-10 were used as experimental data. There was a difference in the performance of the proposed method between MNIST and Fashion-MNIST. When the trigger is removed using the de-trigger autoencoder, patterns on surfaces in the legitimate image may disappear. This is because the image provided by the de-trigger autoencoder has features from the input image, such as lines and outlines, as well as an output value, but patterns on surfaces within the image may appear blurry. Therefore, with Fashion-MNIST, the de-trigger autoencoder provides images in which the patterns on the clothing are generally blurred, though the lines and outlines of the clothing shapes are reflected well in the output values. With MNIST, however, because the numerals have no surface patterns, their lines or outlines are clearly revealed, and the output images are relatively similar to the legitimate data. CIFAR-10, on the other hand, is a color image dataset, unlike MNIST and Fashion-MNIST. When the trigger is removed by the de-trigger autoencoder from a CIFAR-10 image, the blur phenomenon is less obvious than with MNIST and Fashion-MNIST. Therefore, the proposed method has better performance with color image data.

In addition, under the proposed method, the accuracy of the target model on MNIST was approximately 99%, but on Fashion-MNIST it was approximately 91%. This is because in Fashion-MNIST, certain classes, such as “T-shirt” and “Shirt,” have similar characteristics, and it is not easy to distinguish them. Therefore, the performance of the target

model on Fashion-MNIST was slightly worse than that on MNIST, and thus there was a difference in the performance of the proposed method between MNIST and Fashion-MNIST. Similar to the case with Fashion-MNIST, with CIFAR-10 the target model's accuracy on the legitimate samples before they were passed through the de-trigger autoencoder was approximately 92%; therefore, the detection performance was also somewhat degraded.

**Applications.** The proposed method can be used by autonomous vehicles as a defense against backdoor samples. For example, for an autonomous vehicle that uses a neural network, an attacker may intentionally train backdoor samples with a trigger specific to the autonomous vehicle. The autonomous vehicle that trains on the backdoor samples will correctly recognize the legitimate data that lack the trigger. However, the model can cause the vehicle to have an accident by misrecognizing a backdoor sample that has the specific trigger. Therefore, if the proposed method is applied to the autonomous vehicle, the accident can be prevented by having the vehicle issue a warning sound to the user when the probability of having been attacked by a backdoor sample is high.

## VII. CONCLUSION

In this paper, I have proposed a defense method that detects backdoor samples using a de-trigger autoencoder. The method detects backdoor samples using the change in classification result after samples are passed through the de-trigger autoencoder for the input data. The experimental results show that with MNIST, Fashion-MNIST, and CIFAR-10, respectively, the proposed method has 91.5%, 82.3%, and 90.9% detection rates for backdoor samples and 96.1%, 89.6%, and 91.2% accuracy on the legitimate data.

In future research, it will be interesting to investigate a method of generating the de-triggered data using a generative adversarial net [38] instead of an autoencoder. Another subject for future research will be the incorporation of the proposed method into an ensemble of several defense methods.

## REFERENCES

- [1] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.
- [2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.
- [3] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.
- [4] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- [5] Sasanka Potluri and Christian Diedrich. Accelerated deep neural networks for enhanced intrusion detection system. In *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*, pages 1–8. IEEE, 2016.
- [6] Marco Barreno, Blaine Nelson, Anthony D Joseph, and JD Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.
- [7] Chaofei Yang, Qing Wu, Hai Li, and Yiran Chen. Generative poisoning attack method against neural networks. *arXiv preprint arXiv:1703.01340*, 2017.
- [8] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.
- [9] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.
- [10] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.
- [11] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE, 2016.
- [12] Yi Shi and Yalin E Sagduyu. Evasion and causative attacks with adversarial deep learning. In *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*, pages 243–248. IEEE, 2017.
- [13] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1467–1474. Omnipress, 2012.
- [14] Mehran Mozaffari-Kermani, Susmita Sur-Kolay, Anand Raghunathan, and Niraj K Jha. Systematic poisoning attacks on and defenses for machine learning in healthcare. *IEEE journal of biomedical and health informatics*, 19(6):1893–1905, 2015.
- [15] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.
- [16] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Backdoor attack with sample-specific triggers. *arXiv preprint arXiv:2012.03816*, 2020.

- [17] Shuo Wang, Surya Nepal, Carsten Rudolph, Marthie Grobler, Shangyu Chen, and Tianle Chen. Backdoor attacks against transfer learning with pre-trained deep learning models. *IEEE Transactions on Services Computing*, 2020.
- [18] Hyun Kwon, Hyunsoo Yoon, and Ki-Woong Park. Multi-targeted backdoor: Identifying backdoor attack for multiple deep neural networks. *IEICE Transactions on Information and Systems*, 103(4):883–887, 2020.
- [19] Zhen Xiang, David J Miller, and George Kesidis. Reverse engineering imperceptible backdoor attacks on deep neural networks for detection and training set cleansing. *arXiv preprint arXiv:2010.07489*, 2020.
- [20] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 273–294. Springer, 2018.
- [21] Maurice Weber, Xiaojun Xu, Bojan Karlas, Ce Zhang, and Bo Li. Rab: Provable robustness against backdoor attacks. *arXiv preprint arXiv:2003.08904*, 2020.
- [22] Hyun Kwon. Detecting backdoor attacks via class difference in deep neural networks. *IEEE Access*, 8: 191049–191056, 2020.
- [23] Yann LeCun, Corinna Cortes, and Christopher JC Burges. Mnist handwritten digit database. AT&T Labs [Online]. Available: <http://yann.lecun.com/exdb/mnist>, 2, 2010.
- [24] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.
- [25] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. online: <http://www.cs.toronto.edu/kriz/cifar.html>, 2014.
- [26] Cheng-Yuan Liou, Wei-Chen Cheng, Jiun-Wei Liou, and Daw-Ran Liou. Autoencoder for words. *Neurocomputing*, 139:84–96, 2014.
- [27] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017.
- [28] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.
- [29] Joseph Clements and Yingjie Lao. Hardware trojan attacks on neural networks. *arXiv preprint arXiv:1806.05768*, 2018.
- [30] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 113–125, 2019.
- [31] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.
- [32] Keiron O’Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.
- [33] Nina Narodytska and Shiva Kasiviswanathan. Simple black-box adversarial attacks on deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1310–1318. IEEE, 2017.
- [34] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *The International Conference on Learning Representations (ICLR)*, 2015.
- [35] Zhi Chen and Pin-Han Ho. Global-connected network with generalized relu activation. *Pattern Recognition*, 96:106961, 2019.
- [36] Andri Ashfahani, Mahardhika Pratama, Edwin Lughofer, and Yew-Soon Ong. Devdan: Deep evolving denoising autoencoder. *Neurocomputing*, 390: 297–314, 2020.
- [37] Xugang Lu, Yu Tsao, Shigeki Matsuda, and Chiori Hori. Speech enhancement based on deep denoising autoencoder. In *Interspeech*, volume 2013, pages 436–440, 2013.
- [38] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

## APPENDIX

TABLE 8: Architecture of the target model for MNIST and Fashion-MNIST. “Conv.” represents a convolutional layer.

Layer type	Shape
Conv.+ReLU	[3, 3, 32]
Conv.+ReLU	[3, 3, 32]
Max pooling	[2, 2]
Conv.+ReLU	[3, 3, 64]
Conv.+ReLU	[3, 3, 64]
Max pooling	[2, 2]
Fully connected+ReLU	[200]
Fully connected+ReLU	[200]
Softmax	[10]

TABLE 9: Architecture of the target model [33] for CIFAR-10. “Conv.” represents a convolutional layer.

Layer type	Shape
Conv.+ReLU	[3, 3, 64]
Conv.+ReLU	[3, 3, 64]
Max pooling	[2, 2]
Conv.+ReLU	[3, 3, 128]
Conv.+ReLU	[3, 3, 128]
Max pooling	[2, 2]
Conv.+ReLU	[3, 3, 256]
Conv.+ReLU	[3, 3, 256]
Conv.+ReLU	[3, 3, 256]
Conv.+ReLU	[3, 3, 256]
Max pooling	[2, 2]
Conv.+ReLU	[3, 3, 512]
Conv.+ReLU	[3, 3, 512]
Conv.+ReLU	[3, 3, 512]
Conv.+ReLU	[3, 3, 512]
Max pooling	[2, 2]
Conv.+ReLU	[3, 3, 512]
Conv.+ReLU	[3, 3, 512]
Conv.+ReLU	[3, 3, 512]
Conv.+ReLU	[3, 3, 512]
Max pooling	[2, 2]
Fully connected+ReLU	[4096]
Fully connected+ReLU	[4096]
Softmax	[10]

TABLE 10: Model parameters for MNIST, Fashion-MNIST, and CIFAR-10.

Parameter	MNIST	Fashion-MNIST	CIFAR-10
Learning rate	0.1	0.1	0.1
Momentum	0.9	0.85	0.9
Delay rate	–	–	10 (decay 0.0001)
Dropout	0.5	0.5	0.5
Batch size	128	128	128
Epochs	50	50	200

...



HYUN KWON received the B.S degree in mathematics from Korea Military Academy, South Korea, in 2010. He also received the M.S. degree in School of Computing from Korea Advanced Institute of Science and Technology (KAIST) in 2015, and the Ph.D. degree at School of Computing, KAIST in 2020. He is currently an assistant professor in Korea Military Academy. His research interests include information security, computer security, and intrusion tolerant system.