Feature Extraction Based on Denoising Auto Encoder for Classification of Adversarial Examples

Yuma Yamasaki*[1], Minoru Kuribayashi*, Nobuo Funabiki*, Huy H Nguyen†, Isao Echizen†
* Okayama University, Okayama, Japan
E-mail: [1] pi4m2wk1@s.okayama-u.ac.jp
† National Institute of Informatics, Japan

***Abstract*—Adversarial examples have been recognized as one of the threats to machine learning techniques. Tiny perturbations are added to multimedia content to cause a misclassification in a target CNN-based model. In conventional studies, such perturbations are removed using a couple of filters, and for classification, the features are extracted from the observations of the output of the CNN-based model. However, the use of well-known filters may enable an attacker to adjust an adversarial attack to deal with such filters and fool the detector. In this study, we investigated the effectiveness of certain auto encoders (AEs) in extracting the traces of perturbations. Even if the structure of the AE is leaked, the difference in the training datasets makes an adjustment of the attack difficult to achieve. The effectiveness of the AE designed in this study was evaluated experimentally, and its combination with some known filters was also evaluated.**

I. Introduction

With the progress achieved in the computing devices, image processing techniques using deep learning have been applied in various fields. In particular, CNN image classifiers have significantly contributed to automated driving and face recognition systems. However, CNN image classifiers are known to be vulnerable to adversarial attacks, which add adversarial noise, resulting in misclassifications when using CNN image classifiers. For example, if a car under automatic control misrecognizes a road sign, a serious accident might occur. In addition to automated driving technology, because CNN image classifiers are expected to be applied in various fields, defense against an adversary is essential.

The generation of adversarial noise includes non-targeted attacks that look for classes that are likely to be misidentified, and targeted attacks that misidentify any class of attackers. The basic approach to non-targeted attacks is to first observe the output of a target DNN-based classifier from a modified input image that is perturbed from the original by adding randomly generated noise. Several approaches have been studied to take measures against adversarial examples. One approach is to train an image classifier by applying adversarial examples as well as supervised datasets [1]. By including such examples in the training datasets, it becomes possible to make a robust image classifier. This approach is useful when the possibility of an attack is known in advance. Another approach is to identify whether an input image is an adversarial example. Feature squeezing [2] is one of the most popular methods for analyzing the features extracted from the changes in the output of an image classifier when some image processing filters are used to remove adversarial noise from an input image. With this method, color bit reduction and smoothing filters are employed as denoising filters. Although this method can discriminate adversarial examples with high accuracy for low-pixel datasets such as MNIST and CIFER10, the results for high-pixel datasets such as ImageNet have yet to be reported.

In [3], the classification results of normal images and adversarial examples, after sending both through a denoising filter, are used as training data to train the adversarial example detector. With this method, JPEG compression and scaling are used as denoising filters. Because the architecture of these filters is well known, it is easy to generate adversarial examples using a generative adversarial network (GAN)[4] in such a way that the noise cannot be removed.

In this study, we use an auto encoder (AE) as a denoising filter, which has the advantage of flexibility in its architectural design. We adjusted the denoising strength of the filter by changing the number of images used to train the AE. For an attacker to fully analyze the architecture of the AE, it is necessary to obtain image datasets for training and the parameters used in the study. In this respect, in terms of security, the AE is superior to the filters used in previous studies. We trained a discriminator on adversarial examples, noting that the adversarial noise was removed and the image classification results returned to normal when passing the examples through the designed filter. Implementing the proposed method as a preprocessing operation in a system using an image classifier is expected to prevent the injection of adversarial examples into the image classification system. To confirm the effectiveness of the proposed method in discriminating adversarial examples, we evaluated the discrimination accuracy between normal images and such examples. We also combined the AE with filters used in previous studies to enhance the capability of discriminating adversarial examples.

II. Related Work

In this section, we briefly describe the techniques and defense methods applied to adversary attacks.

A. Adversarial Examples

For an image that can be correctly classified using a CNN image classifier, it is possible to artificially mislead the classifier by adding noise that cannot be perceptually

distinguished. Such an image generated by adding adversarial noise is called an adversarial example, and the generation method is called an adversarial attack. There are two types of adversarial attacks used to fool a DNN-based classifier: targeted and non-targeted. In a targeted attack, an attacker creates a classifier to misclassify a particular class, whereas the attacker generates adversarial examples that are misclassified by the classifier into any class as long as it is different from the true class in a non-targeted attack.

For an input image $\boldsymbol{x}$ of its original classified class, an adversarial example is created by adding noise $\boldsymbol{\eta}$ such that the classifier misclassifies $\boldsymbol{x'} = \boldsymbol{x} + \boldsymbol{\eta}$ into a different class. In training a neural network for an image classifier, a loss function is used to update the weights of the middle layer for optimization. The gradient of the loss function is used to modify the input, not the weights, and to add noise to prevent the classifier from working properly.

To search for an adversarial example $\boldsymbol{x'}$ that minimizes distortion with normal images, the optimization problem is formulated as follows. To optimize this problem, the first term imposes a similarity between $\boldsymbol{x'}$ and $\boldsymbol{x}$. Because the second term facilitates the algorithm in finding $\boldsymbol{x'}$ with a small loss value for class label t, a classifier is likely to predict $\boldsymbol{x'}$ as t. By continuously varying the value, we can find $\boldsymbol{x'}$ with a minimum distance of up to $\boldsymbol{x}$ and simultaneously deceive the classifier. The following is a description of typical attack methods.

1) FGSM[1]: The fast gradient sign method (FGSM) is an attack method that promotes changes in the classification results of input images by using the derivative of the loss function of the model for the input feature vector. Let $\boldsymbol{\theta}$ be a parameter of the classifier model, and let t be a class label for the correct answer to $\boldsymbol{x}$.

Let $J(\boldsymbol{\theta}, \boldsymbol{x}, t)$ be the loss function used for training, and treat this loss function as a vector adjusted for the positive and negative signs of the small value ϵ such that the loss is increased by differentiating it by $\boldsymbol{x}$.

$$\boldsymbol{\eta} = \epsilon \cdot \mathrm{sign}\big(\Delta_x J(\boldsymbol{\theta}, \boldsymbol{x}, t)\big) \tag{1}$$

where sign() is a function that returns a positive or negative sign. The FGSM attack is a method for finding $\boldsymbol{\eta}$ in this way. In FGSM attacks, the value of the loss function corresponding to the specified class t increases when computing t, making it difficult to infer that t is the value of the loss function, which leads to a misrecognition by the trainer.

2) BIM[5]: A BIM attack is an attack technique that repeats an FGSM attack by repeating several finer-grained optimizations. An adversarial case is created by applying the FGSM multiple times and clipping the result at each iteration such that the change in each pixel is not too large. The L1- and L2- versions of BIM are implemented in FoolBox [6].

3) PGD[5]: Project gradient descent (PGD) attacks generate adversarial examples by repeatedly applying FGSM attacks similar to BIM attacks and projecting multiple perturbed examples as valid examples.

4) deepfool[7]: Deepfool is a non-targeted attack technique that generates adversarial examples by repeatedly perturbing an image. The nearest decision boundary is searched, and the input image is slightly modified to reach the boundary at each iteration. The algorithm stops once the modified image changes the classification of the classifier.

5) Carlini & Wagner Method[8]: A Carlini & Wagner attack can be targeted or non-targeted and has three metrics to measure its distortion (L_0 norm, L_1 norm, and L_2 norm). The authors pointed out that the non-targeted l2 norm version can generate adversarial cases most effectively. Adversarial examples are generated by solving the following optimization problem: This attack searches for the smallest perturbation measured by the L_2 norm and makes a classifier classify the modified image incorrectly at the same time. The C&W method is known to be a strong attack that is difficult to defend.

B. Defense Techniques

1) Adversarial Training: One of the available defensive methods is adversarial training[1]. With this method, an adversarial example is generated during the training phase of the image classifier, and the adversarial examples are classified into the correct labels by training the classifier with adversarial examples as supervised data. Using this method, the classification accuracy of the adversarial examples is improved; however, the classification accuracy of normal images that have not been attacked is decreased.

2) Distillation: A defense method called defensive distillation [9] has also been developed, which uses an approach called distillation to reduce the size of the network. When training a new network model, the output of the original network model is used instead of the teacher data. Because probabilities are assigned in addition to the classes of correct answers, the softmax output of the original network model contains information on which classes have similar characteristics. Here, if we edit the softmax layer to increase the probability of being assigned to a class that is not the correct answer, the gradient is reduced, which allows us to train a new network model with robustness. In other words, to fool the edited model, it is necessary to change the input to a level that is perceptible to humans. These defensive methods are applied during the training phase of the image classifier, and as a problem with such approaches, they cannot be applied to models that have already been trained.

3) Defense Method Using Auto Encoder: A denoising AE (DAE)[10] was used to improve the image classification accuracy by removing noise before passing through the image classifier. The objective of this method is to achieve robustness against adversarial noise and improve the classification accuracy by removing such noise using an AE. In addition, some studies have aimed at eliminating noise in adversarial examples using PuVAE[11]. This method removes the noise by mapping the adversarial examples onto the data distribution learned from the normal data using a generative model and achieves denoising with a high speed and high accuracy.

The purpose of these two methods is to remove adversarial noise from adversarial examples and classify them into normal labels. The purpose of our study is to detect adversarial examples, which differentiates it from the above approach.

In this study, we evaluate how adversarial examples are generated by two types of classifiers: Gaussian denoising AE (GaussDAE), which is trained to return images with Gaussian noise to the original images, and AdvDAE, which is trained to return adversarial examples generated for a classifier to the original images. The difference between the adversarial examples generated for AdvDAE and the image classifier, applied as a single network, and the adversarial examples generated for the classifier alone is insignificant. To fool the model composed of GaussDAE and the image classifier, it was necessary to add sufficient noise to the image for visual detection. Similar to the above study, it is assumed in [12] that both AE and image classifiers are known to the attacker. In this study, by intentionally adding Gaussian noise to the image before denoising, the intentionally added adversarial examples were disturbed.

The above studies used MNIST and CIFAR-10 as image datasets, none of which were validated on images from ImageNet[1], which is a database of images with a large number of pixels. In addition, a way to determine whether the input image is an adversarial example has yet to be considered.

Similar to the above studies, we use an AE to denoise adversarial examples. However, the objective of our study is to identify whether an input image is an adversarial example, and the features utilized are the sensitivities of a target image classifier to the denoising filters.

III. Conventional Methods

In the previous section, we described a defense method used for the training phase of an image classifier. In this section, we describe a method for detecting adversarial cases, which is directly related to the purpose of this research.

A. Feature Squeezing

Feature squeezing [2] is a method for distinguishing normal images from adversarial examples by measuring the distance using image filters. With this approach, the classification results of an input image are calculated and some images are obtained by filtering the operation processing as a feature vector. If one of the vectors calculated from the filtered images is far from the vector of the input image, the input is regarded as an adversarial example.

It is generally considered that multiple filters can be used together. In [2], image color bit reduction and smoothing were applied experimentally. In addition, other image-processing operations can also be used as filters for detecting adversarial examples. In the classification, the value with the largest distance between the confidence vectors of each image processing filter was used. Here, because the value of the distance varies greatly depending on the image processing filter, there is a problem with the method used in selecting the value applied for judging the threshold, that is, the selection of the image processing filter.

[1]http://www.image-net.org/

B. Sensitivities to Filtering Strength

As an extension of feature squeezing, JPEG compression and scaling are used as denoising filters to identify adversarial examples [3]. This study focused on the sensitivities of the image classification results of adversarial examples to the strength of the filters.

JPEG compression is a method for compressing image data while preventing image degradation by finely quantifying the low-frequency components, which contain many important aspects of an image, and roughly quantizing the high-frequency components. Because visually important signals are preserved during lossy operations, JPEG compression is a useful denoising filter. The strength of the filter can be changed using the quality factor (QF).

Scaling is an image processing technique that changes the size of an image. When an image is scaled down, the entropy of the image data decreases. After scaling up to reconstruct the image to its original size, the nonlinear effects on the image result in denoising filtering artifacts. The effects also depend on the choice of the interpolation algorithm used. By applying the operation to adversarial examples, adversarial noise can be removed during data compression while preserving the visual appearance of the image. The strength of the scaling filter can be changed in terms of the scaling factor.

By varying the strength of the filters, the transition of the top labels and their probabilities are extracted to form a feature vector. In the classification, a narrow neural network composed of a few fully connected layers was used in [3].

C. Problems with conventional methods

In previous studies, common image processing operations such as a bit-length reduction, median filter, JPEG compression, and scaling have been employed as denoising filters. The architecture of these filters is well known. Hence, attackers can design a more sophisticated adversarial attack to generate adversarial examples in such a way that the changes in the outputs of the image classifier become indistinguishable from normal images. Furthermore, by making a discriminator based on these filters, a generative adversarial network (GAN)[4] can be easily designed, and a sophisticated generator, namely, an adversarial attack, may be created as a result.

As a solution to this problem, it is necessary to avoid leaking information regarding the filters. To maintain secrecy, ordinal image-processing operations should be avoided. Instead, we investigated a method for implementing an AE as a noise reduction filter. As the advantages here, the structure of the intermediate layer, the weights of each node, and the dataset used to train the AE can be changed, thereby achieving higher security. [13]

IV. Proposed Method

In this paper, we introduce an AE as a denoising filter, as described in the previous study introduced by in Section III.

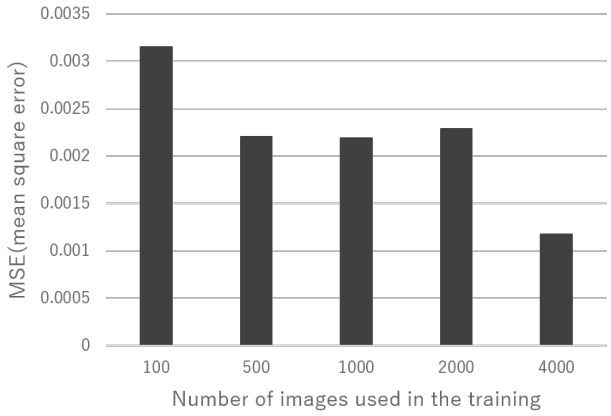

Fig. 1. Relationship between the number of images used to train the AE and MSE.

To obtain the confidence vectors, we combine multiple AEs with different characteristics.

A. Strength of Auto Encoder

An AE is a type of deep neural network model composed of encoding and decoding operations. The input data are first compressed during the encoding operation and are then restored to be as identical as possible to the input data during the decoding operation. Because the encoding operation is expected to remove adversarial noise, the AE can be regarded as a denoising filter. Owing to the DNN-based structure of an AE, its characteristics can be changed by adjusting the dataset and parameters used for training the model. Therefore, flexible control of the characteristics of the denoising filter, particularly its strength, can be achieved by carefully selecting the dataset and parameters. In this study, the strength of the denoising filter was adjusted by changing the number of images for training the AE designed using the proposed method.

The images used to train the AE were randomly selected from among 1.3 million images for each of the 1,300 labels in the color image database ImageNet[2]. In our preliminary experiment, we checked the accuracy of the AE reproduction learned for each number of image patterns. We used the mean square error (MSE) as the loss function for training the AE. The relationship between the number of images used to train the AE and the MSE is shown in Fig. 1. The AE trained with 100 images had a relatively high MSE, whereas the AE trained with 4000 images had a low value. The AEs trained on 500, 1000, and 2000 images had intermediate values between those trained on 1000 and 4000 images. Based on these results, the AEs trained on three patterns (100, 1000, and 4000 images) were implemented as filters with three levels of intensity.

B. Extraction of Feature Vector

By varying the number of images used to train the AE, the intensity of denoising was varied, and the classification labels of the input images of the adversarial examples were extracted as features indicating the changes in the labels of the normal images. First, a normal image that has not been attacked is passed through the image classifier, which outputs the top α estimated value (top α confidence vector), which labels the image. The vector of the top-α class labels is denoted by

$$\boldsymbol{\ell} = (\ell_1, \ldots, \ell_\alpha), \tag{2}$$

and their corresponding probabilities are

$$\boldsymbol{p} = (p_{\ell_1}, \ldots, p_{\ell_\alpha}). \tag{3}$$

When we pass the adversarial examples through the image classifier to obtain the confidence vector, we use the top α confidence vector obtained from the normal image before attacking the adversarial examples as a reference. Because the confidence vector is obtained by referring to the top α labels obtained from the normal images, the classification probability for each label varies from $\boldsymbol{p}$ to $\boldsymbol{p_i}$ when filtered with an arbitrary intensity (i). The classification probability of the top-α class labels after the change is denoted by

$$\boldsymbol{p_i} = (p_{i,\ell_1}, \ldots, p_{i,\ell_\alpha}),\ (1 \leq i \leq n). \tag{4}$$

We train a neural network for the detection of hostile cases by finding features within the variations of $\boldsymbol{p}$ and $\boldsymbol{p_i}$.

C. Mixture of Filters

In a conventional method [3], [14], a feature vector is constructed from the outputs of the softmax function in a target CNN-based image classifier using multiple filters, and the binary classification result is obtained from a shallow neural network composed of a few fully connected layers. This framework used for classifying adversarial examples can be extended by employing different filtering operations.

In general, a suitable combination of filters among a large number of candidates is a time-consuming task. To simplify the process, multiple filters are prepared by changing the strength of the filter. In this study, the strength of the AE was adjusted by changing the number of images used for training the AE.

In addition to the multiple AE stages, we also examined the combination of the AE with JPEG compression and scaling, which are used in the conventional method [3], [14].

V. Experimental Results

We conducted experiments to evaluate the classification accuracy of the proposed method. To solve the vulnerability in terms of secrecy of filter's specifications, the objective is to identify adversarial examples with high accuracy while substituting the other filters with AE.

A. Environments

In this study, we used the Foolbox library to generate adversarial examples from normal images. In this experiment, we assume a white-box attack where the architecture of the image classifier is known to the attacker. For the image dataset, we use ImageNet's ILSVRC2012 validation data. For

[2]http://www.image-net.org/

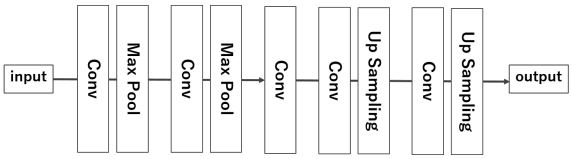

Fig. 2. Convolutional Auto Encoder.

the image classifier, we used the ResNet50 model, which is capable of classifying 1000 classes of the ImageNet image datasets. We simulated both targeted and non-targeted attack methods to convert images into adversarial examples. Five types of targeted attacks, i.e., LBFGS, BIM, PGD, and L1- and L2-distance minimization, and six types of non-targeted attacks, i.e., FGSM, Gradient, Deepfool, Newton, Carlini-Wagner(C&W), and saliency were used. In addition, in this study, a convolutional auto-encoder (CAE) with a convolutional layer in the intermediate layer is used as a filter. The configuration of the convolutional AE is illustrated in Fig. 2. Here, we use Conv, a convolutional layer with kernel size (3×3); Max Pool, a pooling layer with a kernel size of 2×2; and Up sampling, an up-sampling layer with a kernel size of 3×3. As described in Section IV-A, three types of CAEs with different numbers of images used for training were implemented as denoising filters. We also conducted simulations in combination with JPEG compression and scaling, which are the filters used in the conventional method. In JPEG compression and scaling, we vary the intensity of the denoising by adjusting the QF, which determines the degree of loss during the compression process. Two different values of QF, i.e., 90 and 50, were used in this simulation.

B. Classification Accuracy

Using images achieving a successful adversarial attack, we trained the proposed classifier and measured the classification accuracy for each attack. The results listed in Tables I and II show the classification accuracy for targeted and non-targeted attacks, respectively. The "3AEs" in Tables I and II are the results when the feature vectors are obtained by combining three types of AE with different denoising strengths, as described in Section IV-A. In addition, "2JPGs+2SCALs" is the accuracy of the adversarial example detection when JPEG compression and scaling, the filters used in the previous study, are combined. As the results indicate, although the use of the three AEs alone can detect adversarial examples with a certain degree of accuracy, the combination of JPEG compression and scaling shows a remarkable improvement in this regard. However, even when the AE was removed and only JPEG compression and scaling were used as filters, a fairly high accuracy was confirmed. Although the overall accuracy of non-targeted attacks is lower than that of targeted attacks, the above characteristics can be seen in both results.

C. Considerations

From the results shown in Tables I and II, compared to JPEG compression and scaling, the AE appears to be an inferior filter for detecting adversarial examples with high accuracy. However, the AE has a high degree of architectural freedom, which is an advantage not found in the filters applied in previous studies. Although the architecture of the AE used in this simulation did not allow for as accurate a detection as that of conventional methods, there is significant potential for realizing a filter that can obtain more effective feature vectors by reconsidering its training and structure.

In addition, non-targeted attacks are slightly less accurate than targeted attacks. Because a non-targeted attack finds a different class label close to the original as predicted by an image classifier, the amount of change in the probabilities in the top-5 labels can be minimized. Thus, the detection of adversarial images generated by non-targeted attacks is much more difficult.

VI. Concluding Remarks

In this study, we proposed a method for detecting adversarial examples based on the sensitivity to several denoising operations of different strengths when using an AE. The method for detecting adversarial examples based on variations of the confidence vector obtained as the output of a CNN image classifier is extremely effective. However, the detection of adversarial examples using an AE as a denoising filter did not show any advantages over previous studies in terms of accuracy.

Nevertheless, the AE has the potential of achieving secrecy against adversarial attacks owing to its high degree of architectural freedom. If an AE is used to obtain feature vectors that are more effective in detecting adversarial examples, it will be possible to realize a detection system that is less vulnerable to attackers. It will be necessary to design AEs that are more suitable for denoising by changing the datasets used for their training, as well as the structure and parameters of their intermediate layer, among other factors.

Furthermore, it is currently impossible to quantitatively evaluate whether a filter using an AE is less vulnerable to an attacker. A possible method for evaluating the vulnerability of a filter is to generate adversarial examples in which noise is assumed to be removed by various filters, such as an AE and the filters applied in conventional methods, and to then compare their characteristics.

There is a related study that aims to remove noise in Adv using PuVAE. This method requires running as many AEs as the number of data classes operated per adversarial example, which is computationally expensive and not suitable for data with a large number of classes. In contrast, our method can detect adversarial cases with a small number of operations, which makes it effective for datasets with a large number of classes, such as ImageNet.

Acknowledgment

This research was supported by JSPS KAKENHI Grant Number 19K22846, JST SICORP Grant Number JPMJSC20C3, and JST CREST Grant Number JPMJCR20D3, Japan.

TABLE I
CLASSIFICATION ACCURACY IN DETECTING ADVERSARIAL EXAMPLES [%] (TARGETED ATTACK).

FILTER	LBFGS	BIM	PGD	L1	L2
3AEs	92.7	93.0	92.2	92.6	91.8
2JPGs+2SCALs+3AEs	99.6	99.8	99.6	100	99.8
2JPGs+2SCALs	98.6	99.4	99.6	99.8	98.4

TABLE II
CLASSIFICATION ACCURACY IN DETECTING ADVERSARIAL EXAMPLES [%] (NON-TARGETED ATTACK).

FILTER	gradient	FGSM	deepfool	newton	saliency	C&W
3AEs	87.0	85.2	89.4	79.0	82.7	77.4
2JPGs+2SCALs+3AEs	90.8	90.3	92.7	93.1	90.8	93.0
2JPGs+2SCALs	91.2	90.5	92.7	92.2	91.8	92.3

REFERENCES

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. ICLR2015*, 2015.

[2] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: detecting adversarial examples in deep neural networks," in *Proc. NDSS2018*, 2018.

[3] A. Higashi, M. Kuribayashi, N. Funabiki, Huy H. Nguyen, and I. Echizen, "Detection of adversarial examples based on sensitivities to noise removal filter," in *Proc. APSIPA ASC 2020*, 2020, pp. 1386–1391.

[4] I. J. Goodfellow, J. P.-Abadie, M. Mirza, B. Xu, D. W.-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.

[5] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proc. ICLR2017*, 2017.

[6] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A python toolbox to benchmark the robustness of machine learning models," *arXiv preprint arXiv:1707.04131*, 2017.

[7] S.-M. M.-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: a simple and accurate method to fool deep neural networks," *arXiv preprint arXiv:1511.04599*, 2016.

[8] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symposium Security and Privacy*, 2017, pp. 39–57.

[9] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS2014 Deep Learning Workshop*, 2014.

[10] M. Miyazaki, K. Yoshida, T. Iida, H. Masuda, and T. Fujino, "Evaluation of denoising autoencoder as the countermeasure against white-box adversarial examples attacks," in *Proc. The 34th Annual Conference of the Japanese Society for Artificial Intelligence*, 2020, (in Japanese).

[11] U. Hwang, J. Park , H. Jang, S. Yoon, and N. I. Cho, "Puvae: A variational autoencoder to purify adversarial examples," in *Proc. Symposium on Cryptography and Information Security*, 2019.

[12] M. Miyazaki, K. Yoshida, H. Masuda, S. Okura, and T. Fujino, "Random noise addition to input image for mitigating the weaknesses on deep neural network against adversarial examples with autoencoder countermeasure," in *Proc. Symposium on Cryptography and Information Security*, 2021, (in Japanese).

[13] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," *CoRR*, vol. abs/1801.02610, 2018.

[14] A. Higashi, M. Kuribayashi, N. Funabiki, Huy H. Nguyen, and I. Echizen, "Detection of adversarial examples in CNN image classifiers using features extracted with multiple strengths of filter," in *Technical Report of IEICE*, 2021, (in Japanese), vol. 120, pp. 19–24.