EEJE: Two-Step Input Transformation for Robust DNN Against Adversarial Examples

Seok-Hwan Choi, Jinmyeong Shin, Peng Liu, and Yoon-Ho Choi

***Abstract*—Adversarial examples are human-imperceptible perturbations to inputs to machine learning models. While attacking machine learning models, adversarial examples cause the model to make a false positive or a false negative. So far, two representative defense architectures have shown a significant effect: (1) model retraining architecture; and (2) input transformation architecture. However, previous defense methods belonging to these two architectures do not produce good outputs for every input, i.e., adversarial examples and legitimate inputs. Specifically, model retraining methods generate false negatives for unknown adversarial examples, and input transformation methods generate false positives for legitimate inputs. To produce good-enough outputs for every input, we propose and evaluate a new input transformation architecture based on two-step input transformation. To solve the limitations of the previous two defense methods, we intend to answer the following question: How to maintain the performance of Deep Neural Network (DNN) models for legitimate inputs while providing good robustness against various adversarial examples? From the evaluation results under various conditions, we show that the proposed two-step input transformation architecture provides good robustness to DNN models against state-of-the-art adversarial perturbations, while maintaining the high accuracy even for legitimate inputs.**

***Index Terms*—Adversarial examples, adversarial perturbation, deep neural networks(DNNs), input transformation, security.**

I. Introduction

As a representative machine learning model, Deep Neural Networks (DNNs) have shown good outputs for legitimate inputs in various real-world applications [1], [2]. However, many studies showed that DNNs produce false positives or false negatives for adversarial examples, which are human-imperceptible perturbations to inputs to machine learning models [3]–[5]. For example, recent studies showed that such adversarial examples cause false positives or false negatives of practical machine learning systems such as face recognition systems, object recognition systems, and perceptual ad-blocking system [6]–[9]. Also, Bengio, Hinton, and LeCun acknowledged adversarial examples as a representative shortcoming of deep learning at AAAI 2020 [10].

However, traditional techniques, such as dropout, for making machine learning models robust generally do not provide a practical defense against adversarial examples. Here, the term "robustness" is the ability of a machine learning model to cope with adversarial input during execution.

To provide robustness to DNN models against adversarial examples, we can consider proactive countermeasures, which make deep neural networks more robust before adversaries generate adversarial examples and reactive countermeasures, which mitigate the effect of adversarial examples after deep neural networks are built. As a representative proactive countermeasure to strengthen DNN model, let us consider model retraining architecture. As shown in Fig. 1b, model retraining architecture changes the DNN model itself or the training process for the given DNN model [11]–[13]. The model retraining architecture is known to be effective for early adversarial perturbation calculation methods such as Fast Gradient Sign Method (FGSM) [14] and Basic Iterative Method (BIM) [6]. However, the performance of model retraining architecture is limited because retrained DNN models are highly dependent on adversarial perturbation calculation methods. Thus, it is impossible to identify unknown adversarial examples. Also, the model retraining architecture requires high computation and memory usage when retraining DNN models. On the other hand, we can consider a representative reactive countermeasure, commonly referred to as input transformation architecture. As shown in Fig. 1c, input transformation architecture transforms inputs to reduce perturbations of adversarial examples before feeding into DNN models [15]–[17]. Compared to the model retraining architecture, it can provide robustness to DNN models against adversarial examples with low computation and memory usage. Unfortunately, since input transformation architecture transforms even the legitimate input while removing perturbations of adversarial examples, it does not work well for the legitimate input. Failure in working well for the legitimate input means that it can cause significant damage to specific applications such as self-driving car, bio-medicine, and certification which are sensitive to small accuracy variation.

To solve the limitations of the previous two defense methods, we intend to answer the following question: How to maintain the performance of DNN models for legitimate inputs

Manuscript received March 29, 2020; revised June 30, 2020; accepted July 5, 2020. Date of publication July 10, 2020; date of current version July 7, 2021. This work was supported by basic science research program through national research foundation Korea (NRF) funded by the ministry of science, ICT and future planning, Republic of Korea (NRF-2018R1D1A3B07043392) and LG Yonam Foundation (of Korea). Peng Liu was partially supported by ARO W911NF-13-1-0421 (MURI). Recommended for acceptance by Dr. Zhihong Tian. *(Corresponding author: Yoon-Ho Choi.)*

Seok-Hwan Choi, Jinmyeong Shin, and Yoon-Ho Choi are with the School of Computer Science and Engineering, Pusan National University, Busan 46241, South Korea (e-mail: daniailsh@pusan.ac.kr; sinryang@pusan.ac.kr; yhchoi@pusan.ac.kr).

Peng Liu is with the College of Information Sciences and Technology, Pennsylvania State University, University Park, PA 16802 USA (e-mail: pliu@ist.psu.edu).

Digital Object Identifier 10.1109/TNSE.2020.3008394

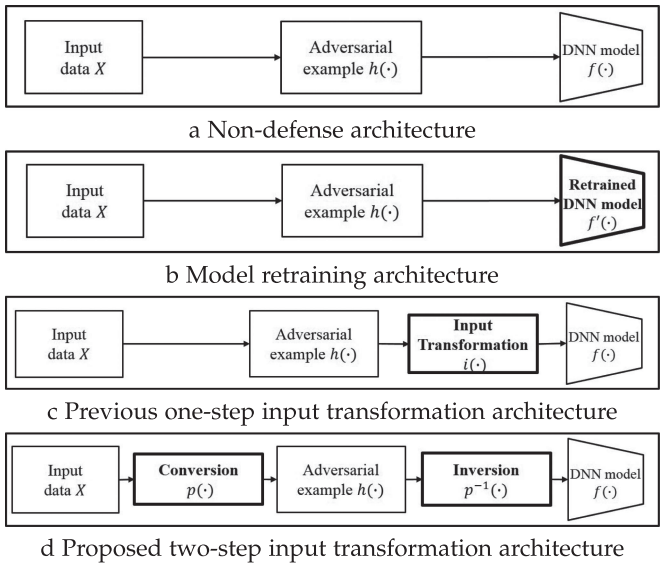

Fig. 1. Architecture comparison of defense methods against adversarial examples.

while providing good robustness against various adversarial examples?

Specifically, we propose a new type of input transformation architecture called two-step transformation architecture as shown in Fig. 1d. Different from the previous one-step input transformation architecture in Fig. 1c, the two-step input transformation architecture consists of two transformation steps: *Conversion* and *Inversion*. In *Conversion* step, input images are transformed into corrupted input images by using a conversion function, $p(\cdot)$, before construction of adversarial examples. In *Inversion* step, adversarial examples are transformed into inputs to DNN models by using the inverse conversion function, $p^{-1}(\cdot)$. Due to the inverse relationship between *Conversion* and *Inversion*, the proposed two-step input transformation architecture works well for the legitimate input while also providing good robustness against various adversarial examples.

To evaluate the effectiveness of two-step input transformation architecture, we also introduce a practical implementation, called EEJE. Here, the term 'EEJE' indicates a chinese phrase, which means to use a barbarian to control the barbarian. From the experimental results, we show that they work well for adversarial examples as well as for legitimate inputs. Compared to model retraining methods, the proposed EEJE method does not require retraining of existing models and does not produce false negatives for various adversarial examples. Also, compared to previous input transformation methods, the proposed EEJE method does not produce bad outputs (false positives) to legitimate inputs.

Main contributions of this paper can be summarized as follows:

(1) We proposed a new type of input transformation architecture using on two-step input transformation to produce good-enough outputs for both legitimate inputs and adversarial examples; (2) As a practical way to implement the two-step input transformation architecture, we introduce new defense method called EEJE; (3) From analysis results using EEJE under state-of-the-art adversarial perturbations, we show that the two-step input transformation architecture provides better robustness than the model retraining architecture and the one-step input transformation architecture while maintaining the high accuracy even for legitimate inputs. Through such contributions, we present the necessity of various studies on the two-step input transformation architecture.

The rest of the paper is organized as follows. In Section II, we overview well-known adversarial perturbation calculation methods and defense methods. We show the proposed input transformation architecture and the operational details of the proposed EEJE method in Section III. In Section IV, we show the influence of different adversarial perturbation calculation methods on EEJE. Finally, we summarize this paper in Section V.

II. Preliminaries and Related Works

In this section, after we overview the state-of-the-art perturbation calculation methods for generating adversarial examples, we introduce two well-known defense architectures against adversarial examples. We also introduce a practical attack model which exploits adversarial examples.

A. Adversarial Perturbation Calculation Methods

In this section, we summarize the characteristics of five well-known adversarial perturbation calculation methods, which are commonly used as construction models for adversarial examples [6], [14], [18]–[20]. Equations for every adversarial perturbation calculation method are summarized in appendix for further reference.

- Fast Gradient Sign Method (FGSM): As a non-iterative-based fast adversarial perturbation calculation method, FGSM was introduced by Goodfellow *et al.* [14]. To calculate adversarial perturbations, FGSM uses the sign of the gradient to increase loss of DNN models.
- Basic Iterative Method (BIM): A. Kurakin *et al.* proposed the basic iterative adversarial perturbation calculation method by extending FGSM [6]. Unlike FGSM, which performs only one gradient update, BIM performs several gradient updates for the fine optimization and clips the pixels of each intermediate result.
- DeepFool: As an L_2 distance-based (Euclidean distance) untargeted attack, DeepFool performs an iterative linearization of the classifier to generate minimal adversarial perturbation [18]. To minimize the magnitude of adversarial perturbation, DeepFool finds the nearest decision boundary from an input X, and calculates perturbation which is closest to the boundary value with multiple iterations.
- C&W's Method: Carlini and Wagner [19] introduced three new perturbation calculation methods, which not only minimize the magnitude of perturbation but also have a higher attack success rate than other methods. Each C&W method is defined as L_0, L_∞ and L_2 type based on the distance metric used to calculate the

perturbation. In this paper, we consider only the L_2 type of C&W method (CW), which is most frequently mentioned in other works [21], [22].

- Jacobian Saliency Map Approach (JSMA): JSMA makes an adversarial perturbation based on the forward-derivative calculation [20]. JSMA is L_0 distance-based method and allows an adversary to compute adversarial saliency maps, which are used to identify the input features causing the most significant changes to the output.

B. Defense Architecture and Methods Against Adversarial Examples

In this section, we overview well-known defense methods against adversarial examples based on two significant defense architectures, i.e., model retraining architecture and input transformation architecture.

1) Model Retraining Architecture and Methods: To make DNN models more robust against adversarial examples, the model retraining architecture changes the current DNN model $f(\cdot)$ into the new DNN model $f'(\cdot)$. It aims at generating good outputs to inputs to DNN models regardless of the existence of adversarial examples. When the adversarial perturbation calculation method $h(\cdot)$ and a legitimate input X are given, the objective function of the model retraining architecture can be expressed into:

$$\begin{aligned} &\max_{f'(\cdot)} Pr(l' = l) \\ s.t. \quad &l' = f' \circ h(X), \\ &l = f'(X). \end{aligned} \tag{1}$$

So far, as representative defense methods corresponding to $f'(\cdot)$, *Adversarial Training* [11], [12], [23], *Defensive Distillation* [13] have shown a significant effect.

Since *Adversarial Training* trains DNN models by using both adversarial examples and legitimate inputs, the trained DNN models are robust against known adversarial examples. However, since *Adversarial Training* is highly dependent on known adversarial examples, they may require periodic retraining to generate good outputs for the new adversarial examples [24].

Defensive Distillation is a defense method that uses distillation training to reduce the bad output caused by adversarial input to DNN models. Note that the original distillation training method trains two DNNs with different architectures to reduce the dimension of DNNs. On the other hand, a *Defensive Distillation* trains two DNNs with the same architecture to improve robustness against adversarial examples. However, in a recent research, *Defensive Distillation* has been proved to be ineffective to defend C&W's method [25]. Even though these two model retraining methods are simple to implement, they require a lot of costs while retraining the current deployed DNN models.

2) Input Transformation Architecture and Methods: Different from the model retraining architecture which retrains the existing DNN models, the input transformation architecture transforms inputs to make adversarial examples less threatened. As shown in Equation (2), the output of the transformation function $i(\cdot)$ becomes the input to the DNN model $f(\cdot)$. Note that since not only the adversarial example but also the legitimate input are transformed by $i(\cdot)$, input transformation architecture should be carefully selected to minimize the classification accuracy degradation for legitimate inputs as well as to maximize the classification accuracy improvement for adversarial examples. That is when the adversarial perturbation calculation method $h(\cdot)$, the input transformation methods $i(\cdot)$ and an input X are given, the objective function of the input transformation architecture can be expressed into:

$$\begin{aligned} &\max_{i(\cdot)} Pr(l' = l) \\ s.t. \quad &l' = f \circ i \circ h(X), \\ &l = f \circ i(X). \end{aligned} \tag{2}$$

Note that most input transformation methods do not require to retrain DNN models and generally require lower computation cost than model retraining methods [15], [26]. However, while transforming every input into the corrupted ones, the classification accuracy of DNN models for the legitimate input is inevitably worse. So far, as representative defense methods corresponding to $i(\cdot)$, there exist *Denoising* [15], *Feature Squeezing*, *Image Purify* [16], [17] and *Guo et al.'s method* [22].

Meng *et al.* proposed a denoising method to reduce perturbations of adversarial examples by using autoencoder-based reformer [15]. The reformer is used to map the adversarial examples into legitimate inputs. It was known that a denoising method was effective when correctly identifying adversarial examples with small perturbation. Xu *et al.* proposed a new defense method, called *Feature Squeezing* [16]. Given an input image, the *Feature Squeezing* method reduced the color depth and reduced variation among pixels using median filter. The *Feature Squeezing* method was effective when mitigating small perturbations such as DeepFool and C&W's method. Song *et al.* proposed an image purifying method, called *PixelDefend* [17]. *PixelDefend* used the PixelCNN to return adversarial examples back to distribution of training dataset. The *PixelDefend* method showed outstanding accuracy against strong perturbations such as DeepFool and C&W's method. To improve the performance, they used two pre-trained classifiers instead of a discriminator. Guo *et al.* proposed a new defense architecture, which combines the input transformation architecture and the model retraining architecture [22]. For mitigating the effectiveness of adversarial examples, they performed image transformation at training and test time. The *Guo et al.'s method* provided a good robustness under the various attack scenarios.

C. Threat Model

As a form of session hijacking, such as sidejacking, sniffing and so on, a man-in-the-middle(MITM) attack allows an adversary to eavesdrop the flow of plain or encrypted traffic between two parties. Especially, in the cloud service environment including the clients and the server, MITM allows

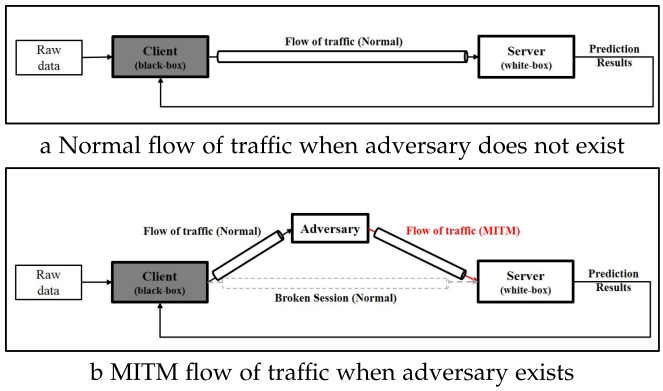

Fig. 2. Threat model: normal flow of traffic vs. adversarial flow of traffic under the MITM attacks.

adversary to impersonate two parties. After gaining access to data in the flow of traffic between two parties, adversary intercepts the data and exploits the real-time transfer of other data [27], [28].

As shown in Fig. 2b, let us consider a cloud-based deep learning environment where the client transmits data (e.g., image, text, and sound) to the server for prediction (e.g, regression and classification). Let us assume that parameter values used for training deep neural networks in the server are publicly known(: white-box). However, details of how to process input data at the client is not known publicly. Let us consider when adversary gains access to data in the flow of traffic after session hijacking. As a result, the normal session between two parties is broken. Even though the normal flow of traffic between two parties is intercepted and perturbed by the adversary, two parties believe that they are communicating each other securely. That is, after analyzing the adversarial examples instead of the normal input data, the server returns the abnormal prediction results to the client.

For example, a cloud-based deep learning environment for autonomous driving such as TuSimple using AWS has been applied to image recognition and object detection for intelligent transportation systems. To perform traffic safety and driver assistance in autonomous driving, input data to deep learning models, collected from multiple sensors such as onboard camera and LiDAR, are transmitted through the vehicular infrastructure. When adversary intercepts and tampers sensor data transmitted from the vehicle(: client) to the server, the server returns the abnormal prediction results on image recognition and object detection. Since such abnormal prediction directly influences on traffic safety and driver assistance, autonomous driving can confront with a major risk. Thus, defense methods which maintain the performance of deep learning models for legitimate inputs while providing good robustness against various adversarial examples are critical issues for performing practical applications.

III. Proposed Architecture and Method

In this section, we describe the proposed input transformation architecture against adversarial examples to obtain good outputs to every input. We also introduce a practical defense method that implements the two-step input transformation architecture.

A. Two-Step Input Transformation Architecture

Different from the previous input transformation architecture, the proposed input transformation architecture performs two-step input transformation before feeding inputs into DNN models.

As shown in Fig. 1d, the proposed input transformation architecture consists of two transformation steps, i.e., *Conversion* and *Inversion*. In the *Conversion* step, the service client side adds the large perturbation to the original input and thus, causes the adversary to add small perturbation to the original input. In the *Inversion* step, the service server side restores the original image for every input to DNN models by subtracting the perturbation added by *Conversion* step.

Such a method is motivated from the following observations. In Fig. 3, we show the input data distribution from CIFAR-10 test dataset [29] over the various L_0 and L_2 norm values, which are computed from the pixel value difference in the original input image and the adversarial example by DeepFool [18]. We used a 3×3 median smoothing for the one-step input transformation architecture and DeepFool conversion for the proposed input transformation architecture. Here, the L_0 value in x-axis of Fig. 3a represents the number of transformed pixels and the L_2 value in x-axis of Fig. 3b represents the normalization value of L_2(Euclidean) distance. In Fig. 3a and b, we observe that while most of test data in the two-step input transformation architecture shows the lower L_0 and L_2 values than the one-step input transformation architecture. We note that even though not directly influencing on robustness of DNN models, the input data distribution over the different L_0 and L_2 norm values can affect the classification accuracy degradation of DNN models.

As shown in Fig. 4, for an input X, the DNN model $f(\cdot)$ and the adversarial perturbation calculation method $h(\cdot)$, the objective function of the two-step input transformation architecture can be expressed into:

$$\begin{aligned} &\max_{p(\cdot)} \min_{h(\cdot)} Pr(l' = l) \\ s.t. \quad & l' = f \circ p^{-1} \circ h \circ p(X), \\ & l = f \circ p^{-1} \circ p(X) = f(X), \end{aligned} \tag{3}$$

where, $p(\cdot)$ is a conversion function which transforms the input to make DNN models robust to adversarial examples, and $p^{-1}(\cdot)$ is an inverse function of $p(\cdot)$, which restores the input image transformed by $p(\cdot)$.

Note that different from the previous one-step input transformation architecture, the proposed two-step input transformation architecture shows the good accuracy even for legitimate inputs. As shown in Equation (3), if adversarial perturbations do not occur, DNN models in the proposed two-step transformation architecture return the same result as DNN models $f(\cdot)$ with no defense as shown in Fig. 1a. This is because *Inversion*, $p^{-1}(\cdot)$, is the inverse of *Conversion*, $p(X)$,

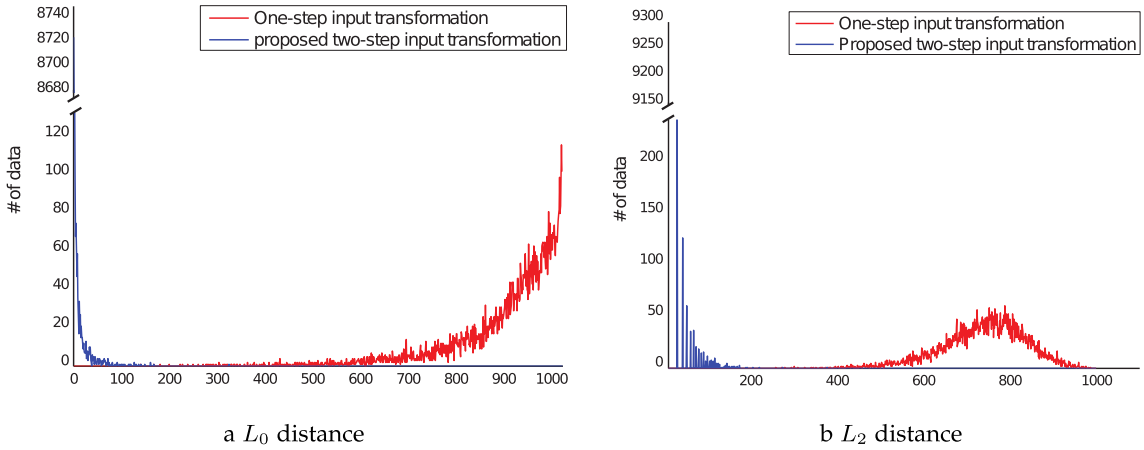

Fig. 3. Data distribution under the various L_0 and L_2 values between the pixel values on the input image and the transformed image by the one-step input transformation architecture and the proposed two-step input transformation architecture. Here, the entire test data of CIFAR-10 dataset were used for input data on ResNet-20 model and DeepFool was used for adversarial perturbation.

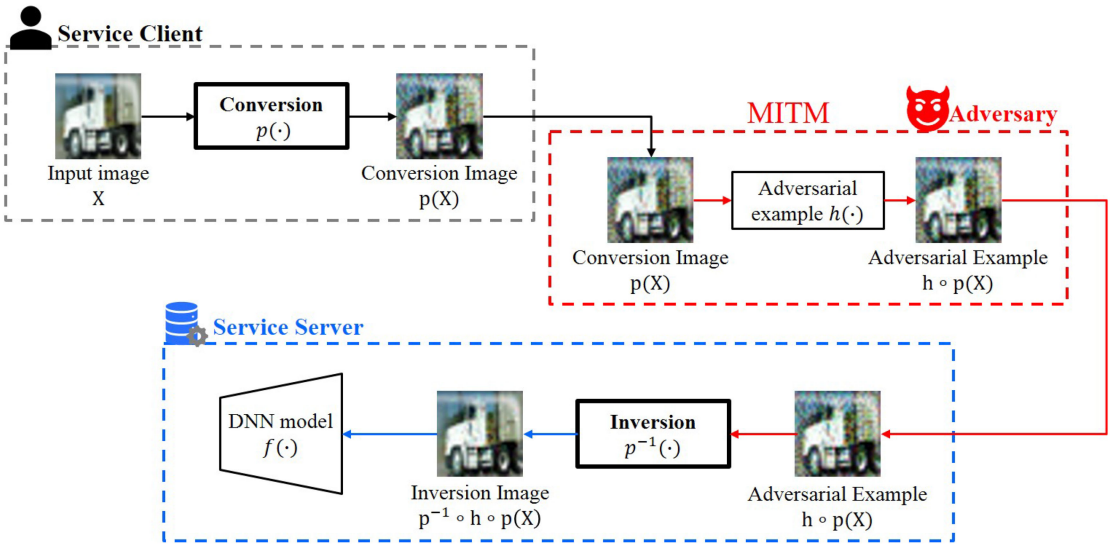

Fig. 4. Overall operation of the proposed two-step input transformation architecture.

for a legitimate input X. In other words, the legitimate input X is completely restored due to the inverse relationship between *Conversion* and *Inversion*. That is, after the service client side transforms legitimate input X to $p(X)$, the service server can restore legitimate input X from $p^{-1} \circ p(X)$.

Let us also note that the proposed architecture may increase robustness of DNN models against adversarial examples after being combined with the other architectures. This is because the proposed new architecture offers the possibility of triple-defense by combining with model retraining architecture and input transformation architecture. In previous research, only dual-defense was possible through the combination of model retraining architecture and input transformation architecture [16], [17]. The combination of different types of defense methods is important for two reasons. First, the combination of defense methods can complement each other's weaknesses. Second, DNN models combined with different defense architectures can increase complexity of the perturbation calculation of the adversary, thereby lowering the success rate of the adversarial examples.

B. A Practical Two-Step Input Transformation Method: EEJE

As a practical two-step input transformation method, we introduce a new method, called EEJE. Here, the term 'EEJE' indicates a chinese phrase, which means to use a barbarian to control the barbarian. In EEJE, perturbations to key features in inputs to DNN models are added by both defender and attacker. That is, different from attacker who minimizes the magnitude of adversarial perturbations to key features in inputs to DNN models, defender makes the magnitude of adversarial perturbations large. Overall operation procedures of EEJE are as follows: (1) Defender adds a certain perturbation $p(\cdot)$ to an input X from *Conversion*; (2) Attacker adds an adversarial perturbation $h(\cdot)$ into the input transformed by *Conversion*; and (3) Defender adds an inverse perturbation of

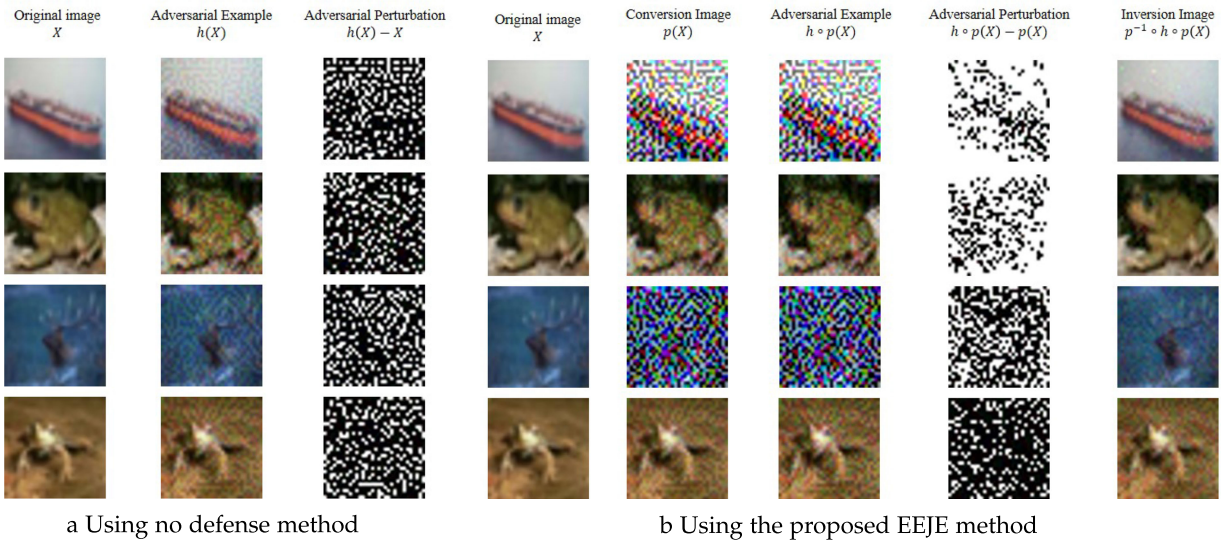

Fig. 5. Four input images randomly selected from CIFAR-10 dataset [29], where each image is transformed using different adversarial perturbation calculation methods: Using FGSM for 1st and 3 rd row images; and using DeepFool for 2nd and 4th row images.

$p^{-1}(\cdot)$ from *Inversion*. Here, the inverse perturbation is additive inverse of the perturbation added in *Conversion*.

Let us compare outputs corresponding to inputs to DNN models in details according to the existence of EEJE. The white-colored and the black-colored pixels in adversarial perturbation images in Fig. 5 indicate pixels which are the same as the original ones and different from the original ones, respectively. For adversarial examples located at the 1st row to the 4th row in Fig. 5a, DNN models return bad outputs because most pixels in adversarial perturbation images have the black color. However, for inversion images located at the 1st row and the 2nd row in Fig. 5b, DNN models return good outputs because most pixels in adversarial perturbation images have the white color. This is because most pixel values in the original input image are kept without change. Also, even though most pixels in adversarial perturbation images for the inversion images located at the 3 rd row and the 4th row in Fig. 5b have the black color, DNN models show good outputs because key features which affect classification are restored by the inverse function $p^{-1}(\cdot)$. That is, the magnitude of perturbation added into key features remains in the range of identification into the good output.

IV. Evaluation Results

To show how robust the proposed two-step input transformation architecture is against adversarial examples, we measured the performance of the proposed EEJE method under various conditions including different adversarial perturbations [6], [14], [18], [19], [20]. Specifically, we evaluated the performance of the proposed EEJE method to answer the following questions:

- How do different types of DNN architectures influence on the performance of the proposed two-step input transformation architecture?
- How does the performance of the proposed two-step input transformation architecture vary under different adversarial perturbations?
- Does the proposed EEJE method show the better performance than the other state-of-the-art defense methods?
- How does the performance of the proposed two-step input transformation architecture change according to the variance of magnitude of adversarial perturbation?
- How does the performance of the proposed two-step input transformation architecture vary under the worst-case adversary?
- Is it efficient to combine the proposed two-step input transformation architecture with the other defense architectures?
- How does the performance of the proposed two-step input transformation architecture vary under various types of data?

We selected these questions based on the results of many representative works [12], [16], [22], [23] and the evaluation checklist of a paper for evaluating adversarial robustness [30]. Such questions encompass evaluations of the scalability(*IV-B1, IV-B2 and IV-B6*) and effectiveness(*3 and IV-B7*) of the proposed two-step input transformation architecture. Also, we answer the questions(*4 and IV-B5*) to verify the reliability of our evaluation results. To measure the performance under the practical usage scenario, we assume that defender knows target DNN model architecture, but does not know adversarial perturbation calculation methods added into inputs to DNN models.

A. Experimental Environment

When evaluating the performance of the proposed EEJE method on DNN models, we embedded the proposed EEJE method into a Convolution Neural Network(CNN), which is a class of DNN, to process input images effectively. To answer

the first six questions listed above, we performed experiments using the CIFAR-10 image classification dataset [29]. CIFAR-10 image dataset consists of 50,000 training images and 10,000 testing images corresponding to 10 classes. For more accurate image classification, we used the entire testing images while measuring the classification accuracy of image classification. To answer the influence of various types of data on the performance of the proposed two-step input transformation architecture, we also performed experiments using MNIST dataset of handwritten digits [31].

To evaluate the influence of different types of DNN architectures on the proposed EEJE method, we measured the classification accuracy under different conversion methods for five ResNet architectures with different sizes and for two state-of-the-art DNN architectures, i.e., ResNet-110 and VGG16. Also, when considering the influence of various perturbation models for *Conversion* and adversarial example generation, we measured the classification accuracy under two categorizes of models: (1) the Gaussian Random Noise (GRN) for generating a random perturbation; and (2) the five well-known adversarial perturbation calculation methods, i.e., FGSM, BIM, DeepFool, C&W and JSMA, for considering practical use cases.

When measuring the influence of different perturbation calculation methods on the proposed EEJE method and the other defense methods, we set the values of parameters into: (1) 0.3 for the magnitude of perturbation (ϵ) in FGSM; (2) 10 and 0.3 for the number of iterations (N) and ϵ, respectively, in BIM; (3) 50 and 0.02 for the maximum number of iterations and overshoot to prevent updates from vanishing, respectively, in DeepFool; and (4) 0 for the parameter to control the confidence value(κ) in C&W's method. These parameter values are set following the recommended configuration values from the cleverhans library [32] and some representative works [16], [22].

We implemented the classification models using TensorFlow-gpu version 1.10.1 and Python version 2.7.15, and performed adversarial perturbation calculations by using the cleverhans software library, which provides standardized reference implementations of adversarial examples [32]. For the efficient experiments, we measured the performance on the Ubuntu 18.04.1 LTS machine with kernel version 4.15.0-36-generic, 2.40 GHz CPU clock(Intel Xeon CPU E5-2630 v3), GeForce GTX 1080 Ti, and 32 GB memory.

B. Experimental Analysis

1) How Do Different Types of DNN Architectures Influence on the Performance of the Proposed Two-Step Input Transformation Architecture?: To evaluate the performance of the proposed EEJE method under different types of DNN architectures, we measured the classification accuracy under different sizes of ResNet architectures without adversarial perturbation. The performance of five ResNet architectures, which have different numbers of layers, from ResNet-20 to ResNet-110 is measured. Evaluation results are listed at the 'None' column in Table I.

We observed that as the size of the ResNet architecture [33] increased, the classification accuracy increased from 90.93% for ResNet-20 to 92.26% for ResNet-110. These results show that as the number of layers in ResNet increases, ResNet becomes more robust against adversarial examples. From the 'None' column in Table I, we also observe how the performance of the proposed EEJE method varied under two different state-of-the-art DNN architectures, i.e., ResNet-110 and VGG16 [34]. The proposed two-step input transformation architecture showed the high accuracy by as much as 92.26% and 93.68% for ResNet-110 and VGG16, respectively. These observations imply that various DNN architectures combined with the proposed two-step input transformation architecture can show the high accuracy.

Result 1: The proposed two-step input transformation architecture effectively works even when being embedded into different types of DNN architectures.

2) How Does the Performance of the Proposed Two-Step Input Transformation Architecture Vary Under Different Adversarial Perturbations?: To show the influence of different adversarial perturbations on the proposed two-step input transformation architecture, we measured the classification accuracy of the proposed EEJE method under the various combination of *Conversion* methods and adversarial perturbation calculation methods when the DNN model is given.

First, the proposed EEJE method showed better accuracy than no *Conversion* (marked into the term 'None' in *Conversion* method column) under various adversarial perturbation calculation methods. For example, while the classification accuracy with no conversion was 5.03% on average in ResNet-20 and 6.95% in VGG16, the classification accuracy of the proposed EEJE method measured into 49.56% in ResNet-20 and 76.12% in VGG16 on average. Among *Conversion* methods, GRN showed the lowest accuracy against adversarial examples. That is, the classification accuracy of EEJE using GRN *Conversion* method was ranged from 21.81% in VGG16 to 26.02% in ResNet-110 on average, while the classification accuracy of EEJE using the DeepFool *Conversion* method was ranged from 77.91% in VGG16 to 56.19% in ResNet-110 on average.

Second, we observe that the proposed EEJE method shows the higher accuracy for the state-of-the-art adversarial examples from the DeepFool and C&W methods than those from the FGSM and BIM methods. This is because the magnitudes of perturbations calculated by the FGSM and BIM methods is larger than those by the DeepFool and C&W methods. For example, while EEJE using FGSM *Conversion* method against FGSM and BIM perturbations in ResNet-110 showed the classification accuracy by as much as 11.74% and 17.52% on average respectively, EEJE using FGSM *Conversion* method against DeepFool and C&W perturbations in ResNet-110 showed the classification accuracy by as much as 92.26% and 90.44% on average respectively.

Third, as observed from the last column in Table I, EEJE using various *Conversion* methods showed no accuracy degradation even for legitimate inputs, i.e., inputs without adversarial perturbations. For example, given the ResNet-20 model, EEJE using different *Conversion* methods showed the same accuracy regardless of the existence of *Conversion* methods.

TABLE I
CLASSIFICATION ACCURACY OF EEJE UNDER THE COMBINATION OF VARIOUS *CONVERSION* METHODS AND ADVERSARIAL PERTURBATION CALCULATION METHODS USING CIFAR-10 DATASET

Model	Conversion method	Adversarial perturbation calculation method					
		FGSM	BIM	DeepFool	C&W	Average	None
ResNet-20	None	10.36%	2.84%	3.76%	3.15%	5.03%	90.93%
	GRN	11.9%	12.94%	34.86%	30.68%	22.59%	90.93%
	FGSM	10.68%	12.77%	90.93%	89.5%	50.97%	90.93%
	BIM	10.88%	3.6%	87.35%	62.62%	41.86%	90.93%
	DeepFool	12.26%	31.33%	90.93%	85.83%	55.08%	90.93%
	C&W	12.16%	17.77%	90.9%	80.64%	50.36%	90.93%
ResNet-32	None	12.67%	2.61%	4.31%	3.14%	5.68%	91.91%
	GRN	12.24%	20.2%	37.9%	33.36%	25.92%	91.91%
	FGSM	12.22%	15.7%	91.86%	89.93%	52.45%	91.91%
	BIM	12.36%	5.66%	88.45%	75.27%	45.43%	91.91%
	DeepFool	12.14%	35.31%	91.91%	87.36%	56.68%	91.91%
	C&W	12.33%	17.77%	91.83%	84.06%	51.95%	91.91%
ResNet-44	None	12.67%	2.85%	5.53%	3.26%	6.07%	91.97%
	GRN	12.24%	14.65%	39.21%	32.77%	24.71%	91.97%
	FGSM	12.22%	14.48%	91.96%	90.81%	52.36%	91.97%
	BIM	12.36%	4.37%	89.53%	75.26%	45.38%	91.97%
	DeepFool	12.14%	32.25%	91.96%	87.23%	55.89%	91.97%
	C&W	12.33%	17.77%	91.95%	82.51%	51.14%	91.97%
ResNet-56	None	13.12%	2.58%	4.75%	2.85%	5.82%	92.12%
	GRN	12.81%	15.2%	37.93%	31.14%	24.27%	92.12%
	FGSM	12.2%	16.38%	92.1%	90.69%	52.84%	92.12%
	BIM	12.35%	6.86%	89.87%	78.84%	46.98%	92.12%
	DeepFool	12.79%	32.72%	92.09%	86.96%	56.14%	92.12%
	C&W	12.77%	18.42%	92.09%	83.03%	51.57%	92.12%
ResNet-110	None	13.59%	2.69%	5.84%	3.1%	6.3%	92.26%
	GRN	13.21%	16.62%	39.55%	34.7%	26.02%	92.26%
	FGSM	11.74%	17.52%	92.26%	90.44%	52.99%	92.26%
	BIM	12.14%	8.56%	90.09%	79.84%	47.65%	92.26%
	DeepFool	14.34%	30.45%	92.26%	87.74%	56.19%	92.26%
	C&W	14.53%	19.45%	92.26%	85.03%	52.81%	92.26%
VGG16	None	17.69%	5.31%	2.8%	2.01%	6.95%	93.68%
	GRN	13.85%	8.49%	32.4%	32.5%	21.81%	93.68%
	FGSM	38.01%	77.28%	93.68%	93.6%	75.64%	93.68%
	BIM	34.82%	71.08%	93.13%	93.31%	73.08%	93.68%
	DeepFool	44.99%	79.33%	93.68%	93.65%	77.91%	93.68%
	C&W	45.92%	78.21%	93.68%	93.65%	77.86%	93.68%

TABLE II
CLASSIFICATION ACCURACY OF EEJE USING JSMA

Conversion method	Adversarial Perturbation					
	FGSM	BIM	DeepFool	C&W	JSMA	None
None	10.36%	2.84%	3.79%	3.15%	1.11%	90.93%
JSMA	13.03%	20.82%	57.95%	54.54%	63.69%	90.93%

Adversarial perturbation	Conversion method							
	GRN	FGSM	BIM	DeepFool	C&W	JSMA	Average	None
JSMA	55.66%	55.26%	64.69%	38.97%	42.95%	63.69%	53.53%	1.11%

This is because the legitimate inputs are restored by *Inversion*, i.e, the inverse function of *Conversion*.

Since the implementation of JSMA perturbation [20] needs a lot of memory and computation time, we measured the classification accuracy under the combination of the JSMA *Conversion* method with various perturbations in ResNet-20. As shown in Table II, EEJE using JSMA *Conversion* method showed the classification accuracy by as much as 13.03% against FGSM perturbation and 63.69% against JSMA perturbation. Also, we observed that for the given JSMA perturbation method, EEJE using different *Conversion* methods showed the classification accuracy by as much as 53.53% on average. For example, EEJE using the DeepFool *Conversion* method showed the classification accuracy by as much as 38.97% against JSMA perturbation. Also, EEJE using the BIM *Conversion* method showed the classification accuracy by as much as 64.69% against JSMA perturbation.

Result 2: The proposed two-step input transformation architecture using various Conversion methods is robust against various adversarial perturbations while maintaining the classification accuracy even for legitimate inputs. Especially, EEJE using the DeepFool Conversion method shows the best robustness against various adversarial perturbations on average.

3) Does the Proposed EEJE Method Show the Better Performance Than the Other State-of-the-Art Defense Methods?: To compare the performance of the proposed EEJE method with the other state-of-the-art defense methods, we measured the classification accuracy of the *Adversarial Training* [12], [23], *Feature Squeezing* [16] and *Guo et al.'s method* [22] in ResNet-20. Here, *Adversarial Training* is a

TABLE III
COMPARISON RESULTS WITH *ADVERSARIAL TRAINING* AND *FEATURE SQUEEZING*

Model	Conversion method	Adversarial perturbation calculation method					
		FGSM	BIM	DeepFool	C&W	Average	None
ResNet-20	**None**	18.79%	6.23%	12.93%	8.52%	11.61%	76.80%
ResNet-20 + **EEJE**	**GNR**	15.46%	12.3%	38.91%	35.17%	25.46%	76.80%
	FGSM	15.62%	23.45%	76.49%	75.21%	47.69%	76.80%
	BIM	16.1%	20.86%	75.08%	69.63%	45.41%	76.80%
	DeepFool	20.56%	22.71%	76.80%	72.37%	48.11%	76.80%
	C&W	18.71%	22.04%	76.63%	72.52%	47.47%	76.80%
ResNet-20 **+ Adversarial Training**	**FGSM**	73.71%	11.5%	14.19%	9.84%	27.31%	74.27%
	PGD	35.59%	22.95%	11.2%	7.8%	19.34%	71.72%
ResNet-20 **+ Feature Squeezing**	**Bit depth 4-bit**	18.69%	6.29%	40.31%	14.73%	20.00%	76.11%
	Bit depth 5-bit	18.79%	6.61%	36.85%	11.91%	18.49%	76.32%
	Median Smoothing(2x2)	20.83%	12.37%	39.82%	31.03%	26.01%	70.62%
	Median Smoothing(3x3)	14.4%	17.17%	31.05%	26.4%	22.25%	60.54%
	Non-local Means 11-3-4	18.78%	6.38%	40.19%	13.14%	19.62%	46.13%
ResNet-20 **+ Guo et al.'s method**	**Total Variance Minimization (0.03)**	13.7%	9.5%	18.9%	18.2%	15.07%	53.72%
	Total Variance Minimization (5.0)	9.4%	9.5%	35.4%	32.8%	21.77%	67.25%

representative method of model retraining and *Feature Squeezing* and *Guo et al.'s method* are the state-of-the-art methods using one-step input transformation. For *Adversarial Training*, we configured *FGSM*-based *Adversarial Training*, where ϵ is set into 0.3 and Projected Gradient Descent (PGD)-based *Adversarial Training*, where ϵ is set into 0.1. For *Feature Squeezing*, we used five squeezing methods which are frequently used in many references [21], [35]. For *Guo et al.'s method*, we transformed the inputs using Total Variance Minimization (TVM), where weight is set into 0.03 and 5.0, and used transformed inputs at training and test time. For more accurate measurement of the classification accuracy, we also excluded data augmentation that could affect the performance of the *Adversarial Training*.

In Table III, we observed that the classification accuracy of *Adversarial Training*, *Feature Squeezing* and *Guo et al.'s method* decreased for legitimate inputs. While EEJE in ResNet-20 showed the same classification accuracy by as much as 76.80% under no adversarial perturbation, the classification accuracy of non-local means-based *Feature Squeezing* decreased by as much as 76.80% to 46.13%. Also, the classification accuracy of TVM-based *Guo et al.'s method* decreased by as much as 76.80% to 53.72%. For *Adversarial Training*, *FGSM*-based *Adversarial Training* and PGD-based *Adversarial Training* in ResNet-20 showed the lower accuracy than EEJE in ResNet-20.

Also, *Adversarial Training*, *Feature Squeezing* and *Guo et al.'s method* showed worse robustness against various adversarial perturbations than EEJE on average. Even though *Adversarial Training* showed the higher robustness than the proposed EEJE method against the FGSM perturbation method, the classification accuracy of *Adversarial Training* against the BIM, DeepFool and C&W perturbations was lower than EEJE using the DeepFool *Conversion* method. Also, the classification accuracy of *Feature Squeezing* and *Guo et al.'s method* showed the worse robustness against most adversarial perturbations than the proposed EEJE method.

Result 3: The proposed EEJE method can be used as a stand-alone defense method against various adversarial perturbations.

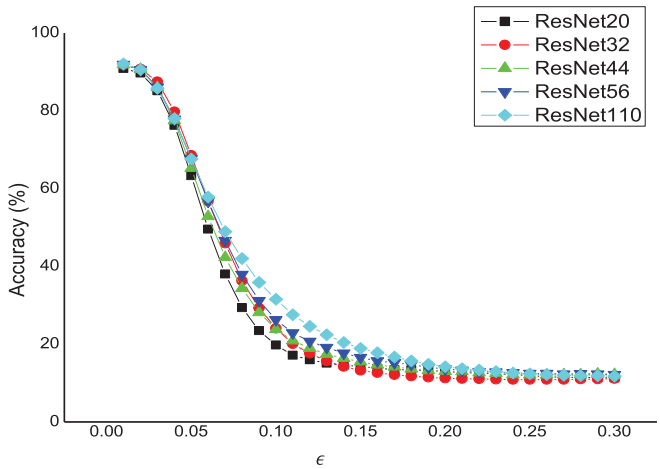

Fig. 6. Classification accuracy with the increase of magnitude of FGSM perturbation.

4) How Does the Performance of the Proposed Two-Step Input Transformation Architecture Change According to the Variance of Magnitude of Adversarial Perturbation?: While changing the magnitude of perturbation generated from the FGSM perturbation, which intuitively shows classification accuracy changes when the magnitude of perturbation increases, we conducted the basic sanity test of EEJE as descried in [30]. According to basic sanity test in [30], the classification accuracy(attack rate) decreases (increases) when the magnitude of perturbation increases. Thus, we showed the classification accuracy of EEJE using the FGSM *Conversion* method while gradually varying the value of ϵ of FGSM perturbation by as much as 0.01 from 0.01 to 0.3.

As shown in Fig. 6, the classification accuracy of the proposed EEJE method decreased as the magnitude of adversarial perturbation increased. Especially, the classification accuracy of the proposed EEJE method rapidly decreased in the range from 0.05 to 0.10. When the value of ϵ was greater than 0.15, the classification accuracy of the proposed EEJE method no longer significantly decreased.

Result 4: The proposed two-step input transformation architecture satisfies the basic sanity test under various attack (success) rates.

TABLE IV
CLASSIFICATION ACCURACY OF EEJE UNDER THE VARIOUS DENOISING TECHNIQUES OF THE ADVERSARY

Model	Attack	Denoising techniques	Adversarial perturbation calculation method				
			FGSM	BIM	DeepFool	C&W	Average
ResNet-20	-	-	10.36%	2.84%	3.76%	3.15%	5.03%
ResNet-20 + **EEJE**	-	-	12.26%	31.33%	90.93%	85.83%	55.08%
	Gray box	**Bit depth 4-bit**	11.74%	11.55%	89.88%	70.00%	45.79%
		Bit depth 5-bit	11.75%	13.69%	90.84%	72.75%	47.26%
		Median Smoothing(2x2)	11.70%	10.30%	89.50%	67.80%	44.83%
		Median Smoothing(3x3)	12.20%	11.13%	89.69%	74.81%	46.96%
		Non-local Means 11-3-4	11.93%	14.29%	90.87%	73.52%	47.65%
	White box	**Bit depth 4-bit**	12.24%	11.55%	22.16%	20.50%	16.61%
		Bit depth 5-bit	11.07%	13.70%	27.90%	25.58%	19.56%
		Median Smoothing(2x2)	11.29%	10.30%	18.73%	16.75%	14.27%
		Median Smoothing(3x3)	12.21%	11.13%	17.12%	15.16%	13.91%
		Non-local Means 11-3-4	11.36%	14.30%	21.52%	19.37%	16.64%

5) How Does the Performance of the Proposed Two-Step Input Transformation Architecture Vary Under the Worst-Case Adversary?: To evaluate the performance of the proposed EEJE method under the worst-case adversary, we measured the performance of the proposed EEJE method in ResNet-20 under the scenario where the adversary is aware of our defense architecture. Note that in our threat model, an adversary may assume that input data has been processed at the client, but does not know details of how to process input data. In other words, the adversary has no direct access to *Conversion* method. Instead, the adversary can mitigate the effectiveness of the *Conversion* method by applying the denoising techniques. On the other hand, an adversary can control the *Inversion* method only when he/she has complete access to the server. Thus, we measured the classification accuracy under two attack scenarios: (1) a white-box attack, where an adversary has complete access to the server and can bypass the *Inversion* method; and (2) a gray-box attack, where an adversary only knows the parameter values used for training deep neural networks and cannot bypass the *Inversion* method. When measuring the classification accuracy of the proposed EEJE method under two attack scenarios, we used DeepFool *Conversion* for the proposed EEJE method and five denoising techniques for the adversary to mitigate *Conversion* method.

As shown in Table IV, the classification accuracy of proposed EEJE method decreased under both white-box and gray-box attacks. For example, while the stand-alone EEJE showed the classification accuracy by as much as 55.09%, EEJE under white-box attack and gray-box attack showed the classification accuracy by as much as 16.20% and 46.50% on average, respectively. For gray-box attack, we observed that key features in inputs are restored by *Inversion* method even though *Conversion* method is mitigated by the adversary. For example, even though the adversary mitigates *Conversion* method with a non-local means filter, EEJE showed the classification accuracy by as much as 90.87% against DeepFool perturbation. Also, the classification accuracy of EEJE under gray-box attack was similar regardless of the denoising techniques. For white-box attack, the classification accuracy of the proposed EEJE method significantly decreased by as much as 16.20%, but is still higher than non-defense architecture. Also, different from the case of gray-box attack, the proposed EEJE method shows the lower accuracy against DeepFool and C&W perturbations. For example, while EEJE against DeepFool and C&W perturbations under the gray-box attack showed the classification accuracy by as much as 90.16% and 71.78% on average respectively, EEJE against DeepFool and C&W perturbations under the white-box attack showed the classification accuracy by as much as 21.49% and 19.47% on average respectively.

Result 5: The proposed two-step input transformation architecture effectively works even the worst-case adversary. Especially, the proposed two-step input transformation architecture shows the good enough performance under the gray-box attack.

6) Is It Efficient to Combine the Proposed Two-Step Input Transformation Architecture With the Other Defense Architectures?: To improve the classification accuracy degradation under the increase of ϵ, especially under ϵ equal to 0.3, we considered to combine the proposed EEJE method with *Adversarial Training* or *Feature Squeezing*. To show the effectiveness of the proposed EEJE method combined with other defense methods, we measured the classification accuracy of EEJE using the DeepFool *Conversion* method combined with FGSM-based *Adversarial Training* or median smoothing (2×2)-based *Feature Squeezing*.

In Table V, we summarize the evaluation results. EEJE combined with *Feature Squeezing*, EEJE with *Adversarial Training*, and EEJE with *Feature Squeezing* and *Adversarial Training* showed better accuracy than using the stand-alone EEJE in ResNet-20. For example, while the stand-alone EEJE showed the classification accuracy by as much as 48.11% on average, EEJE combined with *Adversarial Training* showed the classification accuracy by as much as 61.35% on average. Specifically, the classification accuracy against FGSM perturbation was improved from 20.56% to 58.66%, and the classification accuracy against BIM perturbation was improved from 22.71% to 39.39%. Even when showing the slightly lower accuracy than using the stand-alone EEJE in ResNet-20 against DeepFool and C&W perturbations, EEJE combined with *Feature Squeezing* and *Adversarial Training* in ResNet-20 showed the classification accuracy by as much as 62.4% on average, while the stand-alone EEJE in ResNet-20 showed the classification accuracy by as much as 48.11% on average.

Result 6: The proposed two-step input transformation architecture can good-enough improve the classification accuracy when being combined with other defense methods.

TABLE V
EXPERIMENTAL RESULTS AFTER COMBINING WITH OTHER DEFENSE METHODS

Model	Adversarial perturbation calculation method					
	FGSM	BIM	DeepFool	C&W	Average	None
ResNet-20 + **EEJE**	20.56%	22.71%	76.8%	72.37%	48.11%	76.8%
ResNet-20 + **EEJE** + **Feature Squeezing**	23.95%	35.08%	70.65%	70.11%	49.94%	70.62%
ResNet-20 + **EEJE** + **Adversarial Training**	58.66%	39.39%	74.25%	73.32%	61.35%	74.27%
ResNet-20 + **EEJE** + **Feature Squeezing** + **Adversarial Training**	53.81%	54.96%	69.9%	70.93%	62.4%	70.6%

TABLE VI
CLASSIFICATION ACCURACY OF EEJE UNDER THE COMBINATION OF VARIOUS *CONVERSION* METHODS AND ADVERSARIAL PERTURBATION CALCULATION METHODS USING MNIST DATASET

Model	Conversion method	Adversarial perturbation calculation method					
		FGSM	BIM	DeepFool	C&W	Average	None
ResNet-20	**None**	20.47%	1.82%	1.79%	2.75%	6.71%	97.83%
	GRN	21.84%	2.9%	95.1%	87.48%	51.83%	97.83%
	FGSM	64.25%	47.81%	97.71%	96.66%	76.61%	97.83%
	BIM	36.36%	20.77%	97.37%	95.90%	62.60%	97.83%
	DeepFool	94.77%	97.75%	97.83%	97.83%	97.04%	97.83%
	C&W	92.00%	94.93%	96.89%	96.89%	95.18%	97.83%

7) How Does the Performance of the Proposed Two-Step Input Transformation Architecture Vary Under Various Types of Data?: To show how effective the proposed two-step input transformation architecture is under various types of data, we performed additional experiments using MNIST dataset [31]. Unlike CIFAR-10 dataset, which is a color dataset, MNIST is a grayscale dataset of handwritten digits. MNIST consists of 60,000 training images and 10,000 testing images corresponding to 10 classes.

As shown in Table VI, the proposed two-step input transformation architecture showed the high robustness even for MNIST dataset. For example, while the classification accuracy with no conversion was 6.71% on average, the classification accuracy of the proposed EEJE method measured into 76.65% on average. Especially, EEJE using DeepFool *Conversion* method showed the highest accuracy by as much as 97.04% against adversarial examples. We also observed that EEJE using various *Conversion* methods showed no accuracy degradation for legitimate inputs even when using MNIST dataset.

While EEJE method showed the low robustness for FGSM and BIM perturbations in CIFAR-10 dataset, EEJE method using DeepFool *Conversion* method and C&W *Conversion* method showed the high robustness for FGSM and BIM perturbations in MNIST dataset. For example, EEJE using the DeepFool *Conversion* method against FGSM and BIM perturbations showed the high classification accuracy by as much as 94.77% and 97.75%, respectively.

Result 7: The proposed two-step input transformation architecture shows the good-enough performance under various types of data.

C. Theoretical Analysis

Since the defender cannot know which perturbation calculation method the adversary is using, there exist uncertainties in the optimal selection of adversarial perturbation calculation methods and defense methods. Game theory is useful for analysis when there exist uncertainties in the strategies for each player. Thus, we evaluated the efficiency of the EEJE by analyzing the results based on adversary-defender game.

In adversary-defender game, the defender P_d converts an input image X by selecting an adversarial perturbation calculation method according to a defender's strategy S_j, and the adversary P_a adds an adversarial perturbation into the converted image according to a attacker's strategy S_i.

The game arises from the fact that each player does not know the opponent's strategy, although they do know each other's strategy space. That is, as a two-player game, the adversary-defender game consists of the defender P_d and the adversary P_a with each designated strategy space, i.e., S^D and S^A, where $S_j \in S^D$ and $S_i \in S^A$. As a result of the adversary-defender game, P_a receives a payoff p_{ij} which indicates attack success rate, and P_d receives a payoff $1 - p_{ij}$. Note that the adversary-defender game is a constant sum game since the sum of P_d's payoff and P_a's payoff does not change. Thus, the optimal strategy of P_d can be obtained as follows:

$$\arg\max_{S_j^d} \min_{S_i^a} \sum_{i,j} S_j^d \, S_i^a \, (1 - p_{ij}) \tag{4}$$

where S_j^d and S_i^a are mixed (random) strategies, which are defined according to probability distribution of pure strategies over the strategy spaces S^D and S^A, respectively. In Equation (4), P_d's optimal strategy guarantees a certain classification accuracy, regardless of P_a's strategy.

To evaluate the performance of EEJE, we analyzed two adversary-defender games, which have different sets of S^D but the same sets of S^A. In the first game, we consider $S^D = \{GNR, FGSM, BIM, DeepFool, C\&W\}$ and $S^A = \{FGSM, BIM, DeepFool, C\&W\}$ to find the optimal *Conversion* method for

TABLE VII
ADVERSARY'S PAYOFF TABLE p_{ij}, $S_i \in$ {FGSM, BIM, DEEPFOOL, C&W} AND $S_j \in$ {GNR, FGSM, BIM, DEEPFOOL, C&W}

Defender S^D	Adversary S^A			
	FGSM	**BIM**	**DeepFool**	**C&W**
GNR	0.881	0.870	0.651	0.693
FGSM	0.893	0.872	0.090	0.105
BIM	0.891	0.964	0.126	0.343
DeepFool	0.877	0.686	0.090	0.141
C&W	0.878	0.822	0.091	0.193

TABLE VIII
ADVERSARY'S PAYOFF TABLE p_{ij}, $S_i \in$ {FGSM, BIM, DEEPFOOL, C&W} AND $S_j \in$ {EEJE, FEATURE SQUEEZING, ADVERSARIAL TRAINING}

Defender S^D	Adversary S^A			
	FGSM	**BIM**	**DeepFool**	**C&W**
EEJE	0.881	0.870	0.651	0.693
Feature Squeezing	0.893	0.872	0.090	0.105
Adversarial Training	0.891	0.964	0.126	0.343

EEJE. For ResNet-20, Table VII shows the payoff table of P_a for strategies $S_i \in S^A$ and $S_j \in S^D$. From the solution of Equation (4) for the first game, EEJE using DeepFool *Conversion* is observed into the optimal strategy of P_d. As shown in the experimental analysis, such an observation indicates that EEJE using DeepFool *Conversion* is the best strategy for defender.

In the second game, we considered $S^D = \{EEJE\ (DeepFool), FeatureSqueezing\ (2 \times 2\ mediansmoothing),\ Adversarial\ Training\}$ and $S^A = \{FGSM, BIM, DeepFool, C\&W\}$ in order to show that EEJE is more efficient than the other defense methods.

Payoffs of P_a according to different $S_j\ (\in S^D)$ in ResNet-20 are shown in Table VIII. From the solution of Equation (4) for the second game, EEJE is also selected into the optimal strategy of P_d. This observation indicates that EEJE is better suited for defending adversarial examples than the other defense methods.

V. CONCLUSION

DNN models have shown impressive accuracy in various real-world application fields. However, as adversarial examples cause the model to make a false positive or a false negative, the study on how to maintain the performance of DNN models for legitimate inputs while providing good robustness against various adversarial examples has been emerged. So far, two types of defense architectures have shown a significant effect: (1) model retraining architecture; and (2) input transformation architecture. However, previous defense methods belonging to two architectures did not produce good outputs for adversarial examples as well as legitimate inputs. In this paper, to produce good-enough outputs for every input, we proposed a new type of input transformation architecture using on two-step input transformation. Also, as a practical implementation method, we introduced a practical defense method, called EEJE. From evaluation results under various experimental conditions, we showed that the proposed EEJE method provided robustness to DNN models against various state-of-the-art adversarial perturbations while maintaining the high accuracy even for legitimate inputs. Specifically, the classification accuracy of EEJE using DeepFool *Conversion* showed better performance than *Adversarial Training* or *Feature Squeezing*. Also, the proposed EEJE method combined with *Adversarial Training* or *Feature Squeezing* showed the better classification accuracy than the stand-alone usage of EEJE. From such evaluation results, we believe that the proposed two-step input transformation architecture can support robustness of DNN models against various adversarial perturbations.

REFERENCES

[1] M. Bojarski *et al.*, "End to end learning for self-driving cars," p. 9, 2016.
[2] S. Yue, "Imbalanced malware images classification: A CNN based approach," pp. 3–7, 2017.
[3] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Representations*, 2014.
[4] G. Wang, X. Chen, and C. Xu, "Adversarial watermarking to attack deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 1962–1966.
[5] H. Guo, L. Peng, J. Zhang, F. Qi, and L. Duan, "Fooling AI with AI: An accelerator for adversarial attacks on deep learning visual classification," in *Proc. IEEE 30th Int. Conf. Appl.-Specific Syst., Architectures Processors*, 2019, vol. 2160-052X, pp. 136–136.
[6] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," pp. 1–14, 2016.
[7] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1528–1540. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978392
[8] Y. Zhao, H. Zhu, Q. Shen, R. Liang, K. Chen, and S. Zhang, "Practical adversarial attack against object detector," pp. 1–16, 2018.
[9] F. Tramèr, P. Dupré, G. Rusak, G. Pellegrino, and D. Boneh, "Ad-versarial: Defeating perceptual ad-blocking," pp. 1–17, 2018.
[10] T. Ray, "Deep learning godfathers bengio, hinton, and lecun say the field can fix its flaws," *ZDNet*, 2020. [Online]. Available: https://www.zdnet.com/article/deep-learning-godfathers-bengio-hinton-an d-lecun-say-the-field-can-fix-its-flaws/
[11] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári, "Learning with a strong adversary," pp. 1–12, 2015.
[12] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," pp. 1–17, 2016.
[13] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," pp. 1–16, 2015.
[14] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–11. [Online]. Available: http://arxiv.org/abs/1412.6572
[15] D. Meng and H. Chen, "Magnet: A two-pronged defense against adversarial examples," pp. 1–13, 2017.
[16] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," pp. 1–15, 2017.
[17] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," pp. 1–20, 2017.
[18] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," pp. 1–9, 2015.
[19] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," pp. 1–19, 2016.
[20] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," pp. 1–16, 2015.
[21] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. A. Storer, "Deflecting adversarial attacks with pixel deflection," pp. 1–17, 2018.
[22] C. Guo, M. Rana, M. Cissé, and L. van der Maaten, "Countering adversarial images using input transformations," pp. 1–12, 2017.

[23] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–23. [Online]. Available: https://openreview.net/forum?id=rJzIBfZAb
[24] N. Papernot, P. D. McDaniel, A. Sinha, and M. P. Wellman, "Towards the science of security and privacy in machine learning," pp. 1–19, 2016.
[25] N. Carlini and D. A. Wagner, "Defensive distillation is not robust to adversarial examples," pp. 1–3, 2016.
[26] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," pp. 1–9, 2014.
[27] M. Shirvanian and N. Saxena, "Stethoscope: Crypto phones with transparent & robust fingerprint comparisons using inter text-speech transformations," in *Proc. IEEE 17th Int. Conf. Privacy, Secur. Trust*, 2019, pp. 1–10.
[28] Z. Whittaker, "Dozens of popular iphone apps vulnerable to man-in-the-middle attacks," *ZDNet*, 2017. [Online]. Available: https://www.zdnet.com/article/dozens-of-popular-iphone-apps-vulnerable-to-man-in-the-middle-attacks/
[29] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)," 2009. [Online]. Available: http://www.cs.toronto.edu/ kriz/cifar.html
[30] N. Carlini *et al.*, "On evaluating adversarial robustness," pp. 1–24, 2019.
[31] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[32] N. Papernot, I. Goodfellow, R. Sheatsley, R. Fein-man, and P. McDaniel, "cleverhans v1.0.0: An adversarial machine learning library," 2016, *arXiv:1610.00768*.
[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," pp. 1–12, 2015.
[34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015.
[35] S. Song, Y. Chen, N. Cheung, and C. J. Kuo, "Defense against adversarial attacks with saak transform," pp. 1–7, 2018.

Seok-Hwan Choi was born in Busan, Korea, in 1992. He received the B.E. degree from Pusan National University, Busan, Korea, in 2016. He is currently working toward M.E. degree in computer science and engineering at Pusan National University, Busan, Republic of Korea. His research interests include intrusion detection, network security, and adversarial deep learning.

Jin-Myeong Shin was born in Busan, South Korea, in 1994. He received the B.E. degree in computer science and engineering from Pusan National University, Busan, South Korea, in 2017. In 2017, he joined the Department of Electrical Engineering, Pusan National University, as a master's student.

Peng Liu (Member, IEEE) received the B.S. and M.S. degrees from the University of Science and Technology of China and the Ph.D. degree from George Mason University, in 1999. He is a Professor of information sciences and technology, Founding Director of the Center for Cyber-Security, Information Privacy, and Trust, and Founding Director of the Cyber Security Lab, Penn State University. His research interests include all areas of computer and network security. He has published a monograph and more than 260 refereed technical papers. His research has been sponsored by US National Science Foundation, ARO, AFOSR, DARPA, DHS, DOE, AFRL, NSA, TTC, CISCO, and HP. He has served on more than 100 program committees and reviewed papers for numerous journals. He received the DOE Early Career Principle Investigator Award. He has co-led the effort to make Penn State an NSA-certified National Center of Excellence in Information Assurance Education and Research. He has advised or co- advised more than 30 Ph.D. dissertations to completion.

Yoon-Ho Choi received the M.S. and Ph.D. degrees from Seoul National University, Seoul, Korea, in 2004 and 2008, respectively. From September 2008 to December 2008, he was a Postdoctoral Scholar with Seoul National University. From January 2009 to December 2009, he was a Postdoctoral Scholar with Pennsylvania State University, University Park, PA, USA. While working as a Senior Engineer with Samsung Electronics from May 2010 to February 2012, he had been deeply involved in the development of a commercial Long-Term Evolution cloud communication system. Also, he was an Assistant Professor at Kyonggi University, Suwon, Korea from May 2012 to August 2014. He is currently an Associate Professor with the School of Computer Science and Engineering, Pusan National University, Busan, Korea. His research interests include deep packet inspection, anomaly detection algorithms, adversarial deep learning, privacy preserving machine learning and so on. Dr. Choi has served as a member of several technical program committees in various international conferences and journals.