electronics

MDPI

Article

Real-Time Adversarial Attack Detection with Deep Image Prior Initialized as a High-Level Representation Based Blurring Network

Richard Evan Sutanto [1] and Sukho Lee [2,*]

[1] Department of Computer Engineering, Dongseo University, 47 Jurye-ro, Sasang-gu, Busan 47011, Korea; richardwenz91@gmail.com
[2] Division of Computer Engineering, Dongseo University, 47 Jurye-ro, Sasang-gu, Busan 47011, Korea
* Correspondence: petrasuk@gmail.com; Tel.: +82-51-320-1744

Abstract: Several recent studies have shown that artificial intelligence (AI) systems can malfunction due to intentionally manipulated data coming through normal channels. Such kinds of manipulated data are called adversarial examples. Adversarial examples can pose a major threat to an AI-led society when an attacker uses them as means to attack an AI system, which is called an adversarial attack. Therefore, major IT companies such as Google are now studying ways to build AI systems which are robust against adversarial attacks by developing effective defense methods. However, one of the reasons why it is difficult to establish an effective defense system is due to the fact that it is difficult to know in advance what kind of adversarial attack method the opponent is using. Therefore, in this paper, we propose a method to detect the adversarial noise without knowledge of the kind of adversarial noise used by the attacker. For this end, we propose a blurring network that is trained only with normal images and also use it as an initial condition of the Deep Image Prior (DIP) network. This is in contrast to other neural network based detection methods, which require the use of many adversarial noisy images for the training of the neural network. Experimental results indicate the validity of the proposed method.

Keywords: adversarial attack; adversarial noise detection; deep image prior; neural network

Citation: Sutanto, R.E.; Lee, S. Real-Time Adversarial Attack Detection with Deep Image Prior Initialized as a High-Level Representation Based Blurring Network. *Electronics* **2021**, *10*, 52. https://doi.org/10.3390/electronics10010052

Received: 17 November 2020
Accepted: 24 December 2020
Published: 30 December 2020

1. Introduction

Among the technologies leading the fourth industrial revolution, the interest in artificial intelligence (AI) is increasing, and it is expanding its base by being applied to various fields such as autonomous vehicles, drones, and robots. Furthermore, the research on AI-based autonomous vehicles, which had been conducted only at the laboratory level in the past, has already reached the stage of commercialization, and IT companies such as Google, Apple, and Samsung are actively conducting commercialization research. In addition, AI technology, which contributes to various business areas such as manufacturing, finance, and health care, is creating a wide range of impacts and economic ripple effects throughout society. However, as social dependence on artificial intelligence grows, social costs due to AI malfunctions are also expected to increase. It has been shown in [1,2] that the AI no longer recognizes images correctly when a small amount of well-designed noise, called adversarial noise, is added to the images. The combination of the adversarial noise with the image is called an adversarial example. Using such adversarial examples to intentionally malfunctioning the AI system is called an AI deception attack. The seriousness of such a deception attack lies in the fact that the deception attack can be made without breaking into the system through abnormal routes. In other words, even an AI system that is 100 % protected from hacking, for example, by using generation engine to deceive the attacker [3] can be attacked as much as possible by adversarial examples coming through normal channels. Furthermore, it has been shown in [4,5] that adversarial examples could pose

a threat to the real world, for example, autonomous driving. Well-known adversarial attack methods are the Fast Gradient Sign Method (FGSM) [1], the Basic Iterative Method (BIM) [5] and the Jacobian Saliency Map Attack Method [6] which make use of the gradient of the network. Meanwhile, the Limited-Memory Broyden–Fletcher–Goldfarb–Shanno algorithm method (L-BFGS) [7], the Deepfool [8] and the Carlini–Wagner (CW) attack method [9] use optimization techniques to generate the adversarial example.

Defense methods are divided into two categories depending on whether the AI system allows the entry of the denoised adversarial example into the system or not. Defense methods that allow the entry of the denoised adversarial example assume that the denoised adversarial example can no longer harm the system. Such defense methods include the defense distillation technique [10], a kind of hostile training technique that makes the AI model less vulnerable to adversarial attacks, and methods which use obfuscated gradients to make it difficult for attackers to calculate feasible gradients to generate adversarial examples [11,12]. The method in [13] makes use of the high-level representation to guide the denoiser for better denoising, while the method in [14] leverages the expressive capability of generative models to defend deep neural networks against adversarial attacks. Unfortunately, such defense methods are not valid for all kinds of attacks, and therefore, there is a risk of allowing a valid attack on the system. Thus, the recent works defend the network by detecting adversarial examples, while not allowing their system entry once detected as an attack [15–27]. We will provide a detailed comparison of the characteristics and advantages/disadvantages of these detection methods including the proposed method in Section 2.2.

In this paper, we propose how to detect adversarial noise without knowing what kind of adversarial noise the attacker uses. To this end, we propose the use of a blurring network as an initial condition for the Deep Image Prior (DIP) network which is trained only on normal noiseless images, and therefore, does not require adversarial noisy images in the training process. This is in contrast to other neural network based detection methods, which normally require the use of many adversarial noisy images to train the neural network. Experimental results show that the proposed method can detect the adversarial noise not only in images within the CIFAR10 dataset, but also in general images where many detection methods fail to detect the noise.

We summarize the main contributions of the proposed method as follows:

- We propose a Deep Image Prior network (DIP) based detection method which detects the adversarial noise. Until now, while being used for adversarial image generation [28] or adversarial noise elimination [29], the DIP has not been used for adversarial noise detection.
- Based on the use of the parameters of the blurring network as initial conditions (initial parameters), we propose how the DIP can generate images in real-time. In general, it takes a long time for the DIP to generate an image, which is the reason why it has not been considered for real-time adversarial noise detection.
- We propose a high-level representation based loss function to train the blurring network, so that the blurry image can be trained to blur the image in the direction of correct classification result. This has a large influence on the performance of the proposed method.
- The proposed detection method surpasses other methods not only with the CIFAR10 dataset but also in general images where many detection methods fail.

2. Preliminaries for the Proposed Method

To understand the proposed method, the following preliminaries have to be understood.

2.1. Concept of AI Deception Attack

The adversarial noise is a carefully designed small perturbation that can lead the neural network to make a wrong decision when added to the original input to the network.

The combination of the original input with the adversarial noise is called an adversarial example. Using such an adversarial example to intentionally malfunctioning the AI system is called an AI deception attack. The seriousness of such an AI deception attack lies in the fact that the deception attack can be made without breaking into the system through abnormal routes. This is in contrast to hacking attacks that intrude into the system through abnormal routes. Therefore, even an AI system that is secure against hacking attacks can be attacked by adversarial examples coming through normal channels. Figure 1 shows the concept of an AI deception attack. Even though the noise added to the image is small so that in the eye of the human the original image and the adversarial example look similar, the neural network gives different decisions to the two images. This kind of adversarial example can arouse critical harms to the system which depends on the decision of the neural network.

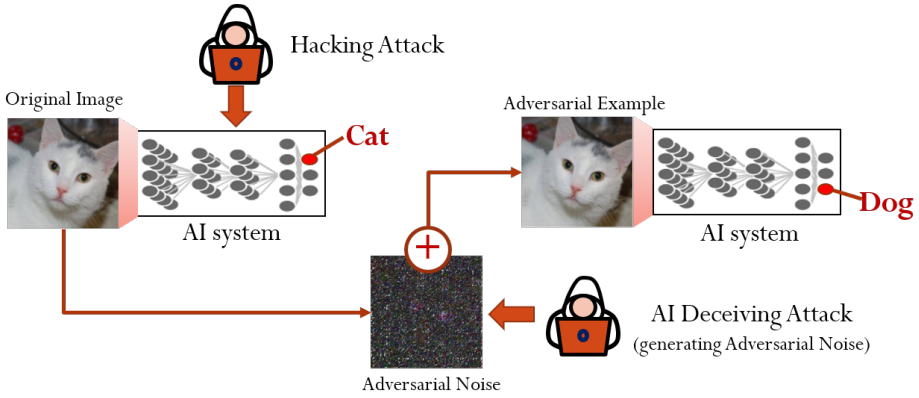


Figure 1. Concept of artificial intelligence (AI) deceiving attack. A small adversarial noise added to the original image can make the neural network to classify the image as a Guacamole instead of an Egyptian cat. This is in contrast to a hacking attack that intrudes the system through an abnormal route.

There are many methods to generate an adversarial example. One of the most widely used adversarial example generating methods to date is referred to as the Fast Gradient Sign Method (FGSM) [1]. This method generates th adversarial example by increasing the distance between the output of the neural network and the true label using the gradient:

$$\hat{x} = x + \epsilon \cdot \text{sign}(\nabla_x J(x, y_{\text{true}})). \tag{1}$$

Here, x is the input image, $\hat{x}$ is the generated adversarial image, $\text{sign}(a)$ is the sign operator which takes the sign of a, y_{true} is the true label of the input image, ∇_x is the gradient with respect to x, and ϵ is a small positive value. A straightforward extension of the FGSM is the Basic Iterative Method (BIM) [5] which applies the adversarial noise η iteratively with a small ϵ value:

$$\begin{aligned} x_0^* &= x \\ x_i^* &= \text{clip}_{x,\epsilon}\left(x_{i-1}^* + \epsilon\, \text{sign}\left(\nabla_{x_{i-1}^*} J(\Theta, x_{i-1}^*, y)\right)\right), \end{aligned} \tag{2}$$

where $\text{clip}_{x,\epsilon}(\cdot)$ represents a clipping of the value of the adversarial example such that it lies within an ϵ-neighborhood of the original image x. It has been shown in [5] that this recursive formula gives control on how far the adversarial example should be pushed beyond the class boundary, which makes it more effective than the FGSM attack on the ImageNet dataset.

One of the most powerful adversarial attacks is the Carlini–Wagner (CW) attack method which uses optimization techniques to generate the adversarial example [9]. The Carlini–Wagner (CW) attack method reformulates the original optimization problem in [1] by moving the given constraints into the minimization function to generate strong adversarial examples.

2.2. Works Related to Adversarial Example Detection

In this section, we describe the characteristics of works related to adversarial example detection and provide a systematic comparison between the works.

Early methods on adversarial detection use the statistical properties of images to detect attacks [15–18,30]. These methods try to discriminate adversarial examples by looking at how the statistical characteristics of them differ from normal data. However, one of the difficulty with this approach is that many adversarial examples are needed to extract the statistical characteristics. Another difficulty is to extract a good feature which can well describe the adversarial example region. To overcome the problem of requiring numerous adversarial images for the training, the authors in [26] propose a method which performs a classification based on the statistical features of adversarial and clean images extracted by a Gaussian process regression with a small number of images, while in [31], the authors propose the Local Intrinsic Dimensionality (LID) feature to well describe the adversarial example region based on the distance distribution of the adversarial example to its neighbors. However, despite these techniques using fewer data and more sophisticated features, statistical characteristics based detection methods are still limited to the use when it is somewhat easy to distinguish between the statistical characteristics between the adversarial example and normal data. Therefore, most statistical methods work only on MNIST or CIFAR10 data sets with somewhat simple statistical characteristics of images.

Another approach is to detect adversarial noise by examining the behaviour of adversarial examples under certain conditions. The work in [19] proposes the Feature Squeezing method, a method of detecting adversarial examples by measuring differences between the predictive vectors of the original and the squeezed examples. Several squeezing operations are suggested in [19], and are tried out against different adversarial attacks. The squeezing operations should satisfy the property that they reverse the effects of the adversarial noise, but do not significantly impact the classifier's predictions on legitimate examples. Therefore, in this approach it is important to find the right squeezing operation for the type of adversarial attack coming into the input. The squeezing approach shows less performance against the L_0 attack than the L_2 attack. Meanwhile, the work in [20] detects the adversarial attack by checking whether the prediction behavior is consistent with a set of neural fingerprints. In this approach, neural fingerprints are encoded into the prediction response around the data-distribution during the training process. The main drawback of this approach is that the detection works only on networks which are trained with the fingerprints, and not on general neural networks that has already been trained without fingerprints. Closely related to the work of [19], the work in [21] utilizes the fact that adversarial examples are usually sensitive to certain image transformation operations. Like [19], in this approach it is important to find a transformation operation which invalidates the adversarial noise, but has no significant impact on the prediction of the classifiers for clean images. Furthermore, it has been shown in [32] that adversarial examples which are robust against image transformation can be made, so this detection method is possible to show weaknesses against adversarial examples that are resistant to transformation. The method in [25] detects adversarial examples by observing that they produce different types of ReLU (Rectified Linear Unit) activation patterns than those produced by normal images. This method works also on real images, but mainly do the experiments with the Deep Fool attack, and it is not clear if the method works also with small global noise-like adversarial noises.

A third approach for adversarial detection is to classify images as normal or adversarial, using a second neural network [22–24,33]. In this approach, the second neural network is trained or distilled with a dataset containing both normal and adversarial data. One of the main drawbacks of this approach is that many normal and adversarial data are required to train the second network and that only the trained adversarial noise can be detected by these methods.

In this paper, we propose a method to detect the adversarial noise without knowledge of the kind of adversarial noise used by the attacker. For this end, we propose a blurring

network that is trained only with normal images and also use it as an initial condition of the Deep Image Prior (DIP) network. This is in contrast to other neural network based detection methods, which require the use of many adversarial noisy images for the training of the neural network. The DIP reconstructs a denoised version of the input image, which is then compared with original input with a proposed detection measure, to decide whether the input is an adversarial noise or not. A method close to ours is proposed in [27] which compares the results of the classification of the input with its denoised version to detect the adversarial example. However, as will be shown in the experimental section, the proposed method has better denoising properties for real-world images and obtains a higher accuracy of detection.

In summary, compared to approaches using the statistical properties described above, the proposed method neither requires a sophisticated design of statistical tools that characterize adversarial noise nor knowledge of the types of adversarial attacks made by attackers. Moreover, the proposed method does not require adversarial examples to train the network.

2.3. Deep Image Prior Network

In [34], the deep image prior (DIP) network has been proposed as a prior for image restoration methods such as image denoising, superresolution, and inpainting. The DIP network converts a random noisy vector z into a restored image $g_{\theta^*}(z)$, where $g_\theta(\cdot)$ denotes the deep image prior network with parameter θ. The parameter θ^* which results in the best restored image $g_{\theta^*}(z)$ is obtained by the way of minimizing the following loss function

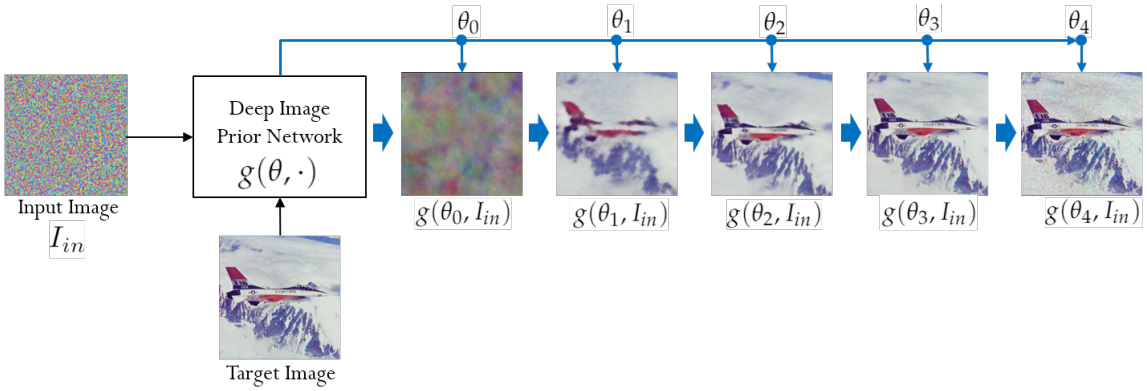
$$\min_\theta \|g_\theta(z) - x_0\|^2, \tag{3}$$

where x_0 denotes the noisy image. That is, in the course of training the DIP network to minimize (3), a good solution of the parameters θ^* which creates a well denoised image $g_{\theta^*}(z)$ can be obtained. It should be noticed that θ^* is not the minimizer of (3), but a passing parameter in the way of minimization, as the minimizer of (3) will result in the noisy image again. Figure 2a shows the progress of the reconstruction of an image with the DIP. The input image I_{in} is just a noise image, and the DIP tries to reconstruct the target image, which is normally a noisy image. As the parameter are updated from θ_0 to $\theta_1, \theta_2, ..., \theta_N$, the generated image $g(\theta_0, I_{in}), g(\theta_1, I_{in}), g(\theta_2, I_{in})...., g(\theta_N, I_{in})$ approaches the target image again. In general, the update of parameters is interrupted at a midpoint, so that only the signal, and not the noise, is reconstructed. The DIP has been shown effective for denoising, and the use of the DIP to denoise the adversarial noise has been proposed by us in [29]. However, the DIP is known to be too slow for the input to converge into the denoised image, and therefore, it cannot be used as such for real-time adversarial example detection. Therefore, in this paper, we adopt the idea of the Model-Agnostic Meta Learning (MAML) into the proposed method to start from parameters from which the fine-tuning of the DIP can be done in real-time, which enables the adversarial example detection to run in real-time.

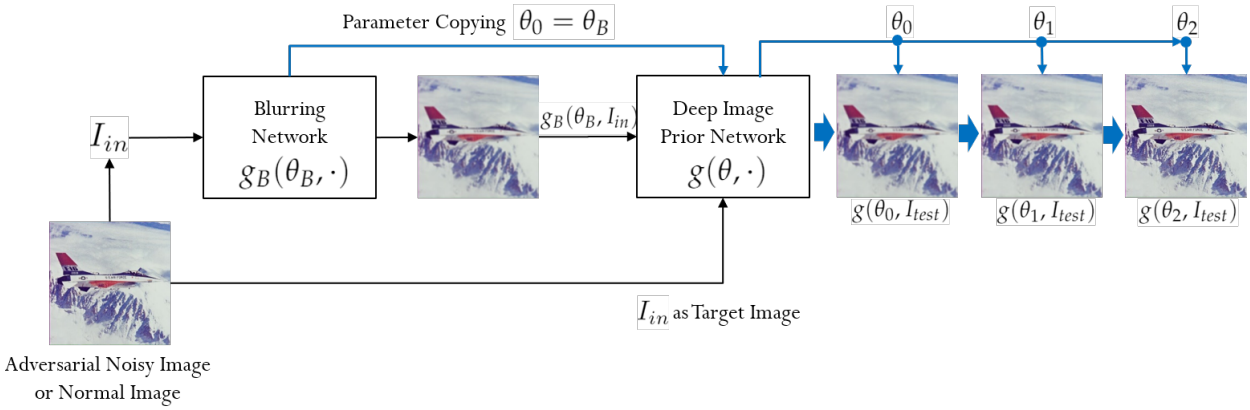
2.4. Model-Agnostic Meta Learning

Model-Agnostic Meta-Learning (MAML) is a method which finds a good initialization of the parameters of the neural network so that an optimal fast learning can be achieved in new tasks with only a small number of gradient steps [35]. Given N different tasks $\mathcal{T}_i, i = 1, 2, ..., N$, the MAML tries to find the initial parameters from which the different tasks can be learned very fast. As each task is related to a specific loss function, the MAML is trying to find the initial parameters that can quickly minimize the different loss functions for the different tasks. Applying this concept to the case of the DIP, the different tasks of the DIP are the tasks of reconstructing different images. Therefore, we obtain a good initial parameter for the DIP by iteratively taking gradient steps with respect to the tasks $\mathcal{T}_i$ for all

$i = 1, 2, ..., N$. By doing so, we optimize an objective in the expectation that the model does well on each task, i.e., reconstruct each new image fast after fine-tuning.



(a) Conventional Deep Image Prior



(b) Deep Image Prior with Blurring Network Parameter Initialization

Figure 2. Initialization of Deep Image Prior (DIP) with blurring network parameters for fast convergence. (**a**) Conventional DIP of slow convergence (**b**) DIP initialized with blurring network parameters.

3. Proposed Method

In this section, we propose an adversarial example detection method which uses the parameters of a blurring network as the initial condition for the DIP.

3.1. Main Idea of the Proposed Method

We first explain the main idea of the proposed method. Figure 3 shows a conceptual diagram of the main idea. The main idea is to compare the outputs of the target CNN (Convolutional Neural Network) with two different inputs, i.e., the test image (I_{test}) which contains the adversarial noise and the image reconstructed by the DIP. Using the test image as the target to be reconstructed by the DIP, the DIP will slowly reconstruct the noise input (I_{in}) to the DIP into the test image. During the reconstruction process, high frequency components that do not include adversarial noise are first reconstructed and the adversarial noise is reconstructed later, which is due to the noise resistance characteristics of the DIP. Therefore, if the reconstruction process is interrupted before the adversarial noise is reconstructed, the reconstructed image will show a different effect than the original test image on the target CNN. Thus, by measuring the correlation between the two outputs of the target CNN, we can determine whether the input contains adversarial noise or not.

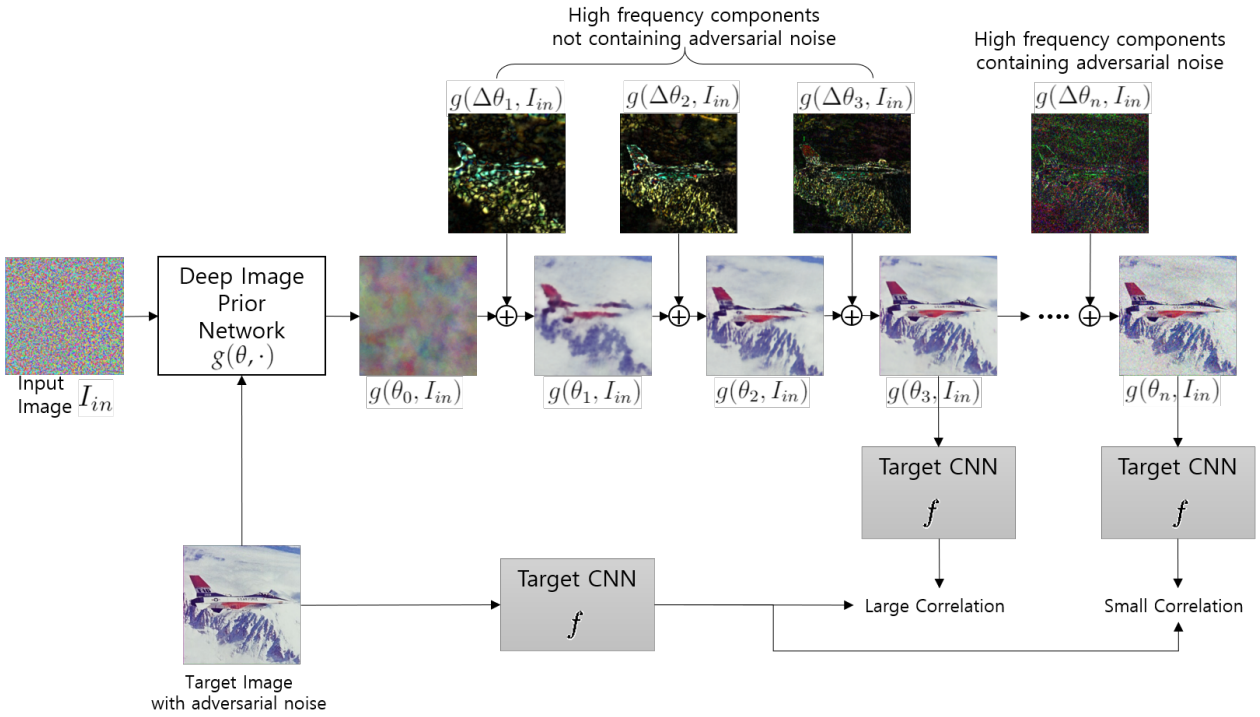


Figure 3. Concept of the proposed method.

Figure 4 shows the overall diagram of the proposed method. In the test time, the input image I_{in} can be either a clean or an adversarial image. The detection is based on a detection measure D which compares the outputs of the target model (target convolutional neural network) for different inputs. The detection measure D takes as the input $f(I_{in})$ and $f(g(\theta + \Delta\theta, I_{in}))$, where $f(\cdot)$ denotes the output of the pre-trained classifier, i.e., the target model which we want to defend by the detection method. Furthermore, $g(\theta + \Delta, I_{in}) = g_B(\theta + \Delta, I_{in})$ denotes the output of the DIP network $g(\cdot)$ after the parameters have been updated by $\Delta\theta$ from the initial parameters θ of the blurring network, i.e., $g(\theta, I_{in}) = g_B(\theta, I_{in})$, and I_{in} is the input image.

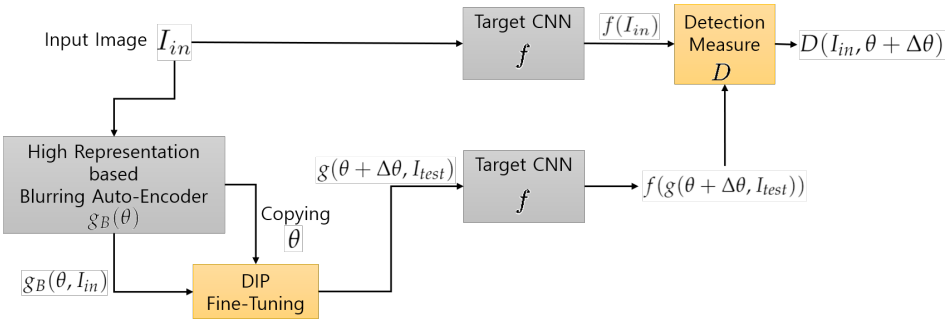


Figure 4. Diagram of the proposed AI deception attack detection method.

The role of the blurring network g_B is two-fold: first, g_B blurs the input image to eliminate the adversarial noise. Second, the parameters in g_B serve as an initial condition for the deep image prior (DIP) network. In the following sections, we explain first the role of the g_B as the blurring network. Then, we explain the role of g_B as an initial condition for the DIP and how the detection measure D is defined.

3.2. Role of g_B as the Blurring Network

The network g_B is trained so that it blurs the input image. The blurring will eliminate the high frequency components, and therefore, removes the adversarial noise to some

extent. However, when eliminating the high frequency components, the components which are helpful for correct classification will also be eliminated. Therefore, to prevent this undesired side-effect, we add a high-level representation guiding term in the loss function when training g_B, so that g_B will blur the image in the direction that its high-level responses are similar to those of non-blurred noise-free images.

We describe the main idea of the proposed method in Figure 5, which shows the space of the high-level features of images. The horizontal axis is the axis of the degree of blur, i.e., as the feature vector is placed more to the right it corresponds to a feature vector of an image which is more blurred. The domain Ω_{true} in Figure 5a,b is the domain of images which are rightly classified. As the blurriness intensifies, the feature vector corresponding to the blurred image will cross the boundary of Ω_{true} and will be misclassified. This is because the high frequency components that contribute to the correct classification are eliminated by the blurring process. However, we want to train the blurring network so that images blurred by this network are still classified in the right class. To be more specific, with regard to Figure 5a, we want the network g_B to be trained so that g_B has parameters θ_2 instead of θ_1, i.e., that it achieves a mapping from $f(I_{in})$ to $f(g_B(\theta_2, I_{in}))$ which still gives the same classification result as $f(I_{in})$, rather than to $f(g_B(\theta_1, I_{in}))$ which classification result differs from $f(I_{in})$ in the case that I_{in} is a noise-free clean image. This can be achieved by training g_B with the following loss function:

$$\mathcal{L}_{blur} = \|g_B(I_{in}) - G_\sigma * I_{in}\|_2^2 + \lambda \|f(g_B(I_{in})) - f(I_{in})\|_2^2, \tag{4}$$

where G_σ refers to the Gaussian kernel with standard deviation σ and $*$ denotes the convolution operator. The minimization of the first term ($\|g_B(I_{in}) - G_\sigma * I_{in}\|_2^2$) aims to achieve a network which results in an output that is a blurred version of the input image, while the minimization of the second term ($\|f(g_B(I_{in})) - f(I_{in})\|_2^2$) aims to result in an output which has similar classification result as I_{in}, and λ is a positive value which controls the balance between the two terms. Here, it should be noticed that in the training we use only noise-free clean images for I_{in}, while in the test time, I_{in} can be either a noisy or a noise-free image.

Meanwhile, for the case that I_{in} contains adversarial noise, we want the blurring network g_B to effectively remove the noise. That is, with regard to Figure 5c, we want the network to achieve a mapping from $f(I_{in})$ to $f(g_B(\theta_2, I_{in}))$ which classification result is different from I_{in} rather than a mapping to $f(g_B(\theta_1, I_{in}))$ which gives the same wrong classification result as I_{in}. In contrast to the case that I_{in} is a clean image, the reason that g_B tries to map the noisy images to the space with different classification results than the input images is that g_B is trained only on clean images. Therefore, a large difference between $f(I_{in})$ and $f(g_B(\theta, I_{in}))$ indicates that the input image is an adversarial image, while a small difference indicates that it is a normal image, and we already can design an adversarial noise detection measure by measuring the similarity between $f(I_{in})$ and $f(g_B(\theta, I_{in}))$. A simple measure would be

$$S(I_{in}, \theta) = \frac{1}{|f(g_B(\theta, I_{in})) - f(I_{in})| + \alpha} \tag{5}$$

which has a large value if $f(I_{in})$ and $f(g_B(\theta, I_{in}))$ are similar, and a small value if they are different, where α is a small positive value to avoid dividing by zero. So, if $S(I_{in}, \theta)$ is small, we can conclude that there is adversarial noise in I_{in}. Thus, we detect the adversarial noise by comparing it with a pre-defined threshold value (Th) as follows:

$$\begin{cases} \text{Detect Adversarial Attack} & \text{if } S(I_{in}, \theta) \geq Th \\ \text{No Adversarial Attack} & \text{if } S(I_{in}, \theta) < Th. \end{cases} \tag{6}$$

The network g_B already has the ability to denoise and detect the adversarial noise.

However, to increase the ability to denoise and detect, we take one further step as in the following section.

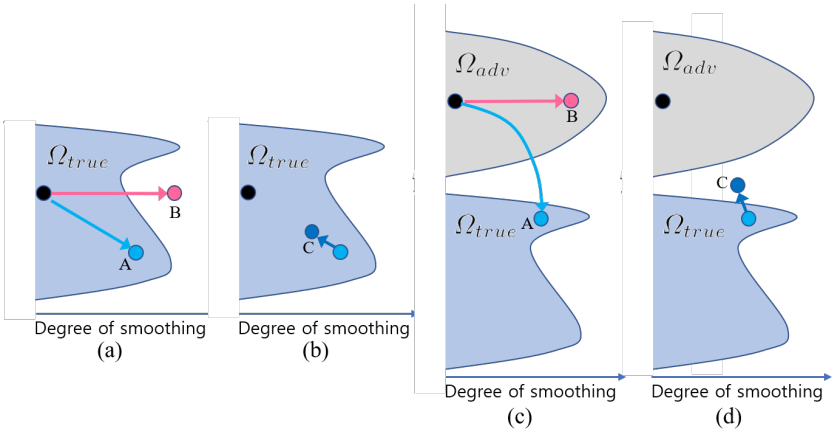


Figure 5. Explanation of the parameter domain. (**a**) effect of putting a normal image through the blurring network (**b**) effect of putting a normal image through the DIP (**c**) effect of putting an adversarial image through the blurring network (**d**) effect of putting an adversarial image through the DIP.

3.3. Role of g_B as the Initial Condition for the DIP

According to the concept of the MAML, the parameters of the blurring network can be regarded as good initial parameters for the task of regenerating the input image, as these parameters have been obtained by learning how to produce blurry versions of different input images. Using the parameters of the blurring network as the initial parameters, we will fine-tune the blurring network to the input image. That is, we use the parameters of the blurring network as the initial condition of the DIP which reproduces the input image. By doing so, we expect that the DIP will converge very fast, so that the reconstruction of the input image can be performed in real-time. Figure 2b shows this concept. Using the parameters of the blurring network, the DIP will first reproduce a blurry version of the input image. As the parameters are getting updated, the DIP will generate a sharper and sharper version of the blurred image in real-time. Therefore, the blurring network becomes overfitted to the test image $I_{test} = g_B(\theta, I_{in})$. This can be achieved by minimizing the following loss function:

$$\mathcal{L}_{fine} = \|g(\theta, I_{test}) - I_{in}\|_2^2. \tag{7}$$

whereas I_{in} in (4) refers to all the images in the training dataset, I_{test} in (7) refers only to the test image.

After a few updates, we get $g(\theta_B + \Delta\theta, I_{test})$ that is close to I_{in} in the L_2 norm sense, but does not include the adversarial noise. When $g(\theta + \Delta\theta, I_{test})$ is put into the classifier f, this will give an exact or at least similar classification result to I_{in}, if I_{in} is a normal image. Figure 5b illustrates the case that the classification results of the normal images get more similar to $f(I_{test})$ by the update of the parameters. On the contrary, if I_{in} is an adversarial image, the overfitting of the DIP will also restore the high frequency components. However, the few iterations are not enough to restore fully the adversarial noise, as we start from a blurred image, and $f(g(\theta + \Delta\theta, I_{test}))$ will be still different from $f(I_{in})$ as illustrated in Figure 5d. Therefore, if we use the similarity measure on $f(I_{in})$ and $f(g(\theta + \Delta\theta, I_{test}))$, we can discriminate between normal and adversarial images. In the next section, we explain the proposed detection measure based on the similarity in details.

3.4. Proposed Detection Measure for Detecting an AI Deception Attack

Let S_5 denote the set of nodes which give the top five activation values in the vector $f(I_{in})$, and denote by $f_n(I_{in})$ the activated value of the n-th node in S_5. The similarity measure we use as the detection measure in the proposed method is

$$S(I_{in}, \theta + \Delta\theta) = \sum_{n \in S_5} \frac{f_n(g(\theta + \Delta\theta, I_{test})) + f_n(I_{in})}{|f_n(g(\theta + \Delta\theta, I_{test})) - f_n(I_{in})| + \alpha}. \quad (8)$$

here, compared with (5), we multiplied by $f_n(g(\theta + \Delta\theta, I_{test})) + f_n(I_{in})$ to give priority to the node value which has larger values than others, whether in $f_n(g(\theta + \Delta\theta, I_{test}))$ or in $f_n(I_{in})$. Actually, instead of using the very last layer of the network, we use the second to the last layer, which we still denote by f to avoid confusion. Furthermore, instead of using all the node values in f, we use only a set of the nodes, i.e., the set S_5. Figure 6 shows the accuracy values for the top five probable classes of the adversarial image, and the corresponding accuracy values for the same classes in the normal and the reconstructed images. It can be observed from Figure 6a,b that the accuracy values for the top five classes of the adversarial image are different in the reconstructed image as the blurring network and the DIP decrease the accuracies of the wrong classes. Therefore, $S(I_{in}, \theta + \Delta\theta)$ becomes small in this case. On the other hand, if the input is the normal image, then the reconstructed image shows somewhat similar accuracy values for the top five classes of the input image, as the blurring network and the DIP will reconstruct the image in the direction that favors the same classification result as the normal image. Therefore, if we use a pre-defined threshold value, we can determine whether an AI deception attack has occurred by comparing the value of $S(I_{in}, \theta + \Delta\theta)$ with the threshold value.

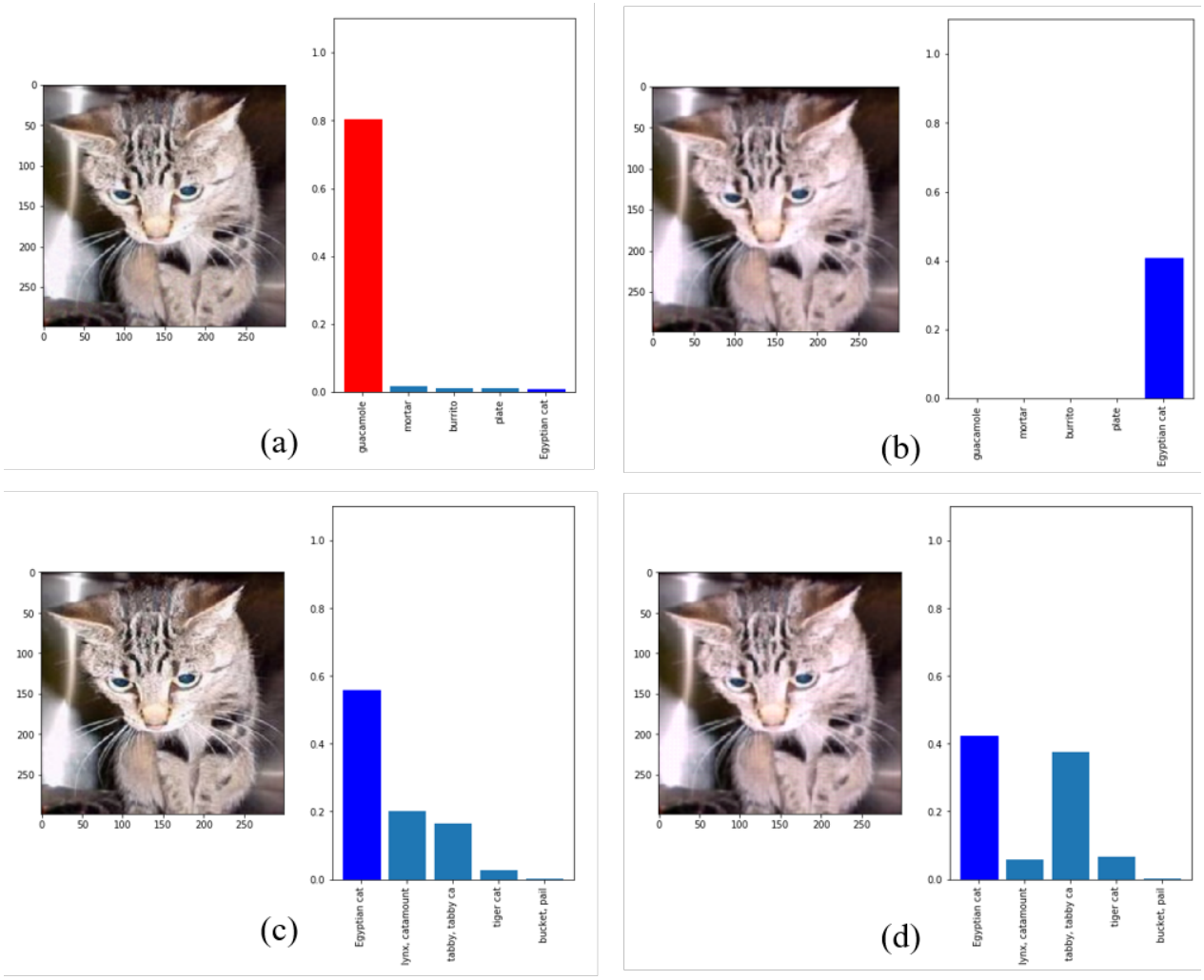


Figure 6. Showing the accuracy values for (**a**) the top five classes in the adversarial image. (**b**) the same five classes as in (**a**) for the output of the DIP with (**a**) as the input. (**c**) the top five classes in the normal image. (**d**) the same five classes as in (**c**) for the output of the DIP with (**c**) as the input.

Finally, a simple analysis on the detection measure will help to understand why it shows a good detection performance, even if the classification results are somewhat inaccurate. Let $f_n(g(\theta + \Delta\theta, I_{test})) = f_n(I_{in}) + \epsilon_n$, where ϵ_n denotes the difference between $f_n(g(\theta + \Delta\theta, I_{test}))$ and $f_n(I_{in})$. Then, we can rewrite (8) as

$$\begin{aligned} S(I_{in}, \theta) &= \sum_{n \in S_5} \frac{|f_n(I_{in}) + \epsilon_n + f_n(I_{in})|}{|f_n(I_{in}) + \epsilon_n - f_n(I_{in})| + \alpha}. \\ &= \sum_{n \in S_5} \frac{|2f_n(I_{in}) + \epsilon_n|}{|\epsilon_n| + \alpha}. \end{aligned} \tag{9}$$

The value ϵ_n is negative due to the following facts: if I_{in} is an adversarial image, $f_n(g(\theta + \Delta\theta, I_{test})) < f_n(I_{in})$, since the accuracy values for the top five classes in the adversarial image decrease with the reconstructed image, while if I_{in} is a normal image, again $f_n(g(\theta + \Delta\theta, I_{test})) < f_n(I_{in})$, since the accuracy values of the reconstructed image cannot be as large as those of the normal image for the top five classes of the normal image. If I_{in} is an adversarial image, $|\epsilon_n|$ is large, while if I_{in} is a normal image, $|\epsilon_n|$ is small. This is due to the fact that $g(\theta, I_{in})$ and I_{in} are significantly different if I_{in} contains adversarial noise, since $g(\theta + \Delta\theta, \cdot)$ acts as a denoiser of the adversarial noise. Therefore, ignoring α, for $\frac{|2f_n(I_{in}) + \epsilon_n|}{|\epsilon_n| + \alpha}$ to have a value less than 1, $f_n(I_{in})$ only needs to be less than $|\epsilon_n|$, which would easily be so if I_{in} were an adversarial image. Therefore, the detection performance will be good even if the accuracy value $f_n(I_{in})$ is not very accurate. That is, even if the DIP does not reconstruct images that provide accurate classification results, the detection performance will be good. This is one of the reasons that the proposed method shows a good detection performance as will be shown in the experimental section. Figure 7 shows the flowchart of the proposed method.

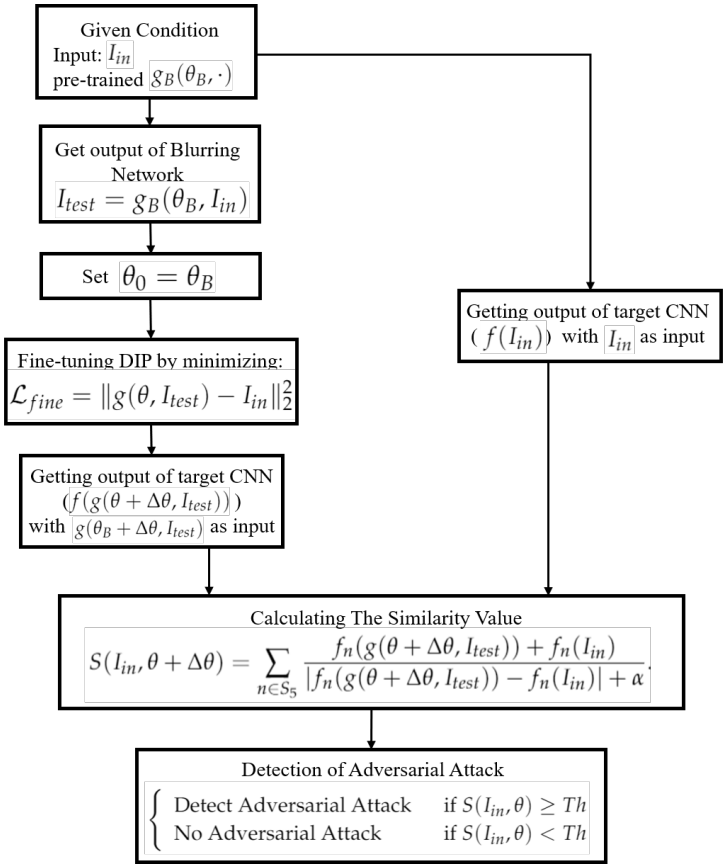


Figure 7. Flowchart of the proposed method.

4. Results and Discussion

We compared the performance of the proposed detection method with those of the Artifact based detector (A-detector) [18], the Gaussian Process Regression based detector (GPR-detector) [26], the Adaptive Noise Reduction (ANR) method [27], and the Local Intrinsic Dimensionality (LID) method [31]. For the test images, we used the test data in the MNIST [36] and the CIFAR10 [37] datasets, and partial subsets of the ImageNet [38] and the 'Dog and Cat' datasets [39]. We used the FGSM [1], the BIM [5], and the Carlini–Wagner (CW) [9] attacks for the experiments. For the BIM attack, we iterated it for a fixed number of iterations so that the adversarial example is created well beyond the decision boundary. As a measure of the detection performance, we used the accuracy of detection, which is calculated as

$$Detection\,Accuracy = \frac{Number\,of\,correct\,detections}{Number\,of\,total\,detections} \times 100\%, \tag{10}$$

where a correct detection means that the detector has classified the input image rightly as an adversarial example or a normal image. The number of adversarial examples or normal images used in the testing are different for different datasets.

4.1. Experimental Settings with the Proposed Method

We used an autoencoder architecture with skip connections for both the blurring network and the DIP network. The encoder part of the network is constructed by three convolutional layers, where the spatial size of the output decreases according to a stride-2 downsampling convolution operation. The numbers of filters in each layer in the encoder are 32, 64, and 128, respectively. The decoder also consists of three convolutional layers, where the spatial size of the output increases according to a stride-2 deconvolution operation. The numbers of filters in each layer of the decoder are 128, 64, and 32, respectively. Both networks are trained with the AdamOptimizer, where used a uniform random initialization between -1 and 1, for all the filters in the networks. We fixed the learning rate to 0.0001 and don't use any learning rate decay. We did an early stopping for the blurring network if the validation loss did not improve for 10 continues epochs to avoid over-training. With the DIP, we performed an early stopping to recover only the image and not the adversarial noise. As we copied the parameters of the blurring network into the DIP, the DIP with only a few iterations already reconstructed the image as explained above. We used 10 iterations for the experiments with the MNIST and the CIFAR10 datasets, and two iterations for the experiments with the ImageNet and 'Dog and Cat' datasets. The number of iterations is set to the value that provide the highest accuracy and has been manually found by many experiments. This is also true for the setting of the threshold value (Th) in (6). The manually found threshold values are 98 for the MNIST and the CIFAR10 datasets, and 74 for the ImageNet and the 'Dog and Cat' datasets. For the Gaussian kernel G_σ in (4), we used a Gaussian Filter of size 5×5.

4.2. Results on the MNIST and the CIFAR10 Datasets

While the MNIST and the CIFAR10 dataset are not composed of real-life images, they are often used as reference datasets to evaluate the performance of the detection methods. The MNIST [36] and the CIFAR-10 [37] datasets consists both of 60,000 images in 10 different classes, where the images in the CIFAR-10 dataset are colour images of size 32×32, while those in the MNIST dataset are gray images of size 28×28. We used four different settings for the ϵ values in the FGSM and the BIM attacks, respectively, and two different settings of confidence values for the CW attack which are shown in Table 1. We tested on 10,000 images for both the MNIST and the CIFAR10 datasets, where half of the images (5000 images) were normal images and the other half images (5000 images) were added with the generated adversarial noise. The target classifier was a simple CNN (Convolutional Neural Network), which consists of five layers. The proposed method uses the same DIP structure as for the experiments with the 'Dog and Cat' dataset, but the sizes

of the input and output images matched the sizes of the images in the CIFAR10 dataset with resizing operations after the input and before the output of the DIP. The adversarial noise was generated with respect to the input image, so the resizing operation can be seen as a part of the proposed detector.

Table 1. Comparison of detection accuracy between different detection methods. Here 'n/w' stands for 'Not working', while '-' stands for 'Unknown'.

Dataset	Attack Method	Strength	[18]	[26]	[27]	[31]	Proposed
MNIST	FGSM	$\epsilon = 0.1$	0.7768	0.7952	0.9514	0.8030	0.8566
		$\epsilon = 0.2$	0.8672	0.8977	0.9826	0.7767	0.6131
		$\epsilon = 0.3$	0.8925	0.9380	0.9887	0.8681	0.5613
		$\epsilon = 0.4$	0.9122	0.9424	0.9866	0.9246	0.5144
	BIM	$\epsilon = 0.1$	0.9419	0.8096	0.9716	0.8092	0.7894
		$\epsilon = 0.2$	0.9768	0.8330	0.9890	0.9027	0.5294
		$\epsilon = 0.3$	0.9801	0.7088	0.9896	0.9574	0.5039
		$\epsilon = 0.4$	0.9779	0.7002	0.9798	0.9797	0.4905
	CW	confidence = 0	n/w	0.9067	0.9419	0.6667	0.7058
		confidence = 10	n/w	0.9652	0.8893	0.6411	0.7926
CIFAR10	FGSM	$\epsilon = 1/255$	0.5521	0.5876	-	0.7108	0.7610
		$\epsilon = 3/255$	0.5892	0.5839	-	0.7097	0.8045
		$\epsilon = 6/255$	0.5994	0.6088	-	0.6667	0.8016
		$\epsilon = 9/255$	0.5931	0.6452	-	0.6671	0.8001
	BIM	$\epsilon = 1/255$	0.6588	0.5502	-	0.7807	0.7968
		$\epsilon = 3/255$	0.6172	0.6342	-	0.7755	0.8014
		$\epsilon = 6/255$	0.5962	0.8215	-	0.8117	0.8005
		$\epsilon = 9/255$	0.5802	0.8335	-	0.8654	0.7941
	CW	confidence = 0	n/w	0.8812	-	0.8757	0.7930
		confidence = 10	n/w	0.9489	-	0.8239	0.8705
Dog and Cat Dataset	FGSM	$\epsilon = 1$	n/w	n/w	0.9132	n/w	0.8516
		$\epsilon = 2$	n/w	n/w	0.6757	n/w	0.8847
		$\epsilon = 4$	n/w	n/w	0.8255	n/w	0.8908
		$\epsilon = 6$	n/w	n/w	0.5729	n/w	0.9096
	BIM	$\epsilon = 10$	n/w	n/w	0.8265	n/w	0.9155
		$\epsilon = 20$	n/w	n/w	0.7943	n/w	0.9010
	CW	confidence = 0	n/w	n/w	0.9198	n/w	0.9167
		confidence = 10	n/w	n/w	0.8365	n/w	0.9587
ImageNet	FGSM	$\epsilon = 1$	n/w	n/w	0.8643	n/w	0.8010
		$\epsilon = 2$	n/w	n/w	0.7763	n/w	0.8375
		$\epsilon = 4$	n/w	n/w	0.6116	n/w	0.8400
		$\epsilon = 6$	n/w	n/w	0.5026	n/w	0.8615
	BIM	$\epsilon = 10$	n/w	n/w	0.7789	n/w	0.8833
		$\epsilon = 20$	n/w	n/w	0.7431	n/w	0.8808
	CW	confidence = 0	n/w	n/w	0.9215	n/w	0.9243
		confidence = 10	n/w	n/w	0.8441	n/w	0.9650

As shown in the accuracy values in Table 1, the proposed method was no better than other detection methods against the FGSM and BIM attacks on the images in the MNIST dataset. This is due to the fact that the images in the MNIST dataset small-sized grayscaled images had no complex background and only two principal colors (black and white), which made it easy to extract the statistical characteristics from the images that could discriminate the adversarial examples from normal data. Therefore, the A-detector [18] and the LID method [31] could easily extract valid statistical features, and the GPR-detector [26] could easily draw the Gaussian distribution of the features from the image, which made it easy for them to detect the adversarial noise. The ANR method [27] could also eliminate

the noise well by using strong spatial smoothing and scalar quantization on the adversarial noise. With the proposed method, the large black-and-white adversarial noises were also easily reconstructed by the DIP, which made it difficult to detect the adversarial cases, so the accuracy with the proposed method was not high. However, since such black-and-white images are not common in real life, we believe that the experiments on the MNIST dataset are less important than those on the ImageNet or the 'Dog and Cat' datasets.

With the more complex CIFAR10 dataset, the accuracy values of the A-detector [18] and the GPR-detector [26] dropped a lot as it became more difficult to discriminate between the statistical properties of the adversarial and the clean cases. The LID method [31] had lower accuracy values than the proposed method for the FGSM attack because it was more difficult to extract LID features from images in the CIFAR10 dataset than the MNIST dataset. However, for the BIM and the CW attacks, which yielded larger perturbations in the LID features in the inner layers than the FGSM attack, the LID method was still comparable with the proposed method. The proposed method also showed a higher detection accuracy, about 80%, than with the MNIST dataset case, which is due to the fact that the adversarial noise was not so extreme as in the MNIST case, and therefore, could be effectivly removed by the DIP. For the ANR method [27], there was no code with the CIFAR10 dataset, so the result is unknown. However, according to the results of the MNIST dataset and the real-world datasets (ImageNet and 'Dog and Cat' datasets), we can derive that the ANR method was still the most effective adversarial denoiser with the CIFAR10 dataset as the images in the CIFAR10 dataset are smoothed, small-sized versions of real-world images. However, we think that the comparison with the ANR method on real-world images is more important, which we describe in the next section.

4.3. Results on the 'Dog and Cat' and the ImageNet Datasets

The 'Dog and Cat' [39] and the ImageNet [38] datasets consist of real-life images, and therefore, are more suitable for the experiments of AI deception attack detection. The 'Dog and Cat' consisted of 25,000 labeled color photos of size 224×224, i.e., 12,500 images of dogs and the same number of images of cats, and the ImageNet consists of about 1 million color images of different sizes divided into 1000 classes. For the experiments with the 'Dog and Cat' datasets, we tested on a total of 1000 images randomly selected from each dataset. We added the adversarial noise to half of this subsampled dataset, so that 500 images are normal images while the other 500 images are added with the generated adversarial noise. For the experiments with the ImageNet dataset, we prepared the dataset in the same way as in [27]. We randomly chose a class in the ImageNet dataset, added the adversarial noise to all the images in this class and experimented whether the detection methods can detect the adversarial noise for this class. We used four different settings for the ϵ values in the FGSM attack and two different settings for the BIM attack. For the CW attack, we used two different settings for the confidence value which are shown in Table 1.

Like many other detection methods, the A-detector [18], the GPR-detector [26], and the LID method [31] did not work with the real-life images as can be seen in Table 1, and could not be evaluated with the 'Dog and Cat' and the ImageNet datasets. This is because the statistical characteristics of normal data and adversarial examples are not clearly distinguished in real-world images because the noise intensity in real-world images is smaller than in the artificial MNIST and CIFAR10 datasets.

The proposed method and the ANR method [27] both worked well on real-world images, but the tendencies in the change of accuracy values according to the strength of the noise were different. With the ANR method, the accuracy tended to decrease as the noise became stronger. This is because the spatial smoothing and scalar quantization of the ANR method could not eliminate the noise well when the noise intensity was increased. However, since the proposed method did not reconstruct noise when the noise is below a certain level, the accuracy increased as the noise increased because the detection measure could better distinguish adversarial examples from denoised data. However, with a very large noise, as with the case of the BIM attack with $\epsilon = 20$, the noise was somewhat

reconstructed with the proposed method, which decreased the accuracy value. However, even in this extreme case, the proposed method still outperformed other detection methods.

4.4. Results on the Computation Time

Table 2 compares the computation time to detect the adversarial example of a single image between the different detection methods. The operation was tested on a PC with Intel i9 CPU, 16GB RAM, and GPU of RTX 2080Ti using the Windows10 OS. All the detection methods ran on Tensorflow v1.12.0 with Python 3.6, and CUDA 9.0 settings. It can be seen from Table 2, that the running time of the proposed method was very fast. The detection times for a single image of the 'Dog and Cat' dataset and the ImageNet dataset were about 0.114 and 0.121 seconds, respectively. To show how the use of the blurring network as an initial condition has accelerated the convergent speed of the DIP in the proposed method, we further measured the reconstruction time of a single image by the normal DIP. Starting from the same input image as the proposed method, the image reconstruction time for the normal DIP was about 4.35 seconds for the a single image in the 'Dog and Cat' dataset. This illustrates that the use of the blurring network as an initial condition is essential for real-time implementation, as proposed.

Table 2. Comparison of detection running time for one image between different detection methods.

	Detection Method					
Dataset	**[18]**	**[26]**	**[27]**	**[31]**	**DIP**	**Proposed**
MNIST	0.022 s	0.434 s	0.050 s	0.002 s	1.85 s	0.039 s
CIFAR10	0.028 s	0.456 s	0.053 s	0.002 s	2.01 s	0.048 s
Dog and Cat	-	-	0.123 s	0.004 s	4.35 s	0.114 s
ImageNet	-	-	0.131 s	0.005 s	4.43 s	0.121 s

5. Conclusions

In this paper, we proposed a real-time adversarial detection method which utilizes the use of a high-level representation based blurring network as the initial condition of the Deep Image Prior(DIP) network. The high-level representation based blurring network is trained with only normal noiseless images, which is in contrast to other neural network based detection methods which require the use of many adversarial noisy images to train the neural network. Due to the nature of not needing adversarial noisy images for training the network, the proposed method can detect AI deception attacks regardless of the type of attack. Furthermore, it has been shown in the experiments that the proposed method outperforms other detection methods in all datasets. The proposed detection method works also for real images while showing a detection speed of less than 0.05 s per image. This performance shows that the proposed method is applicable for real-time AI systems. Another big advantage of the proposed method is that the detection method is not based on deterministic neural networks, but on the deep image prior where the parameters changes for every incoming image. This makes it difficult for an attacker who is aware of the proposed detection system to generate an adversarial example which can avoid the detection. The subject of further research could be an analysis of how to make an attack method that can avoid the detection of the proposed detection method, and again, how to re-detect such an attack method. Research for better measures for the proposed framework to better determine whether an AI deception attack has occurred may be another additional topic of further study.

Author Contributions: conceptualization, S.L. and R.E.S.; methodology, S.L. and R.E.S.; software, R.E.S.; validation, S.L.; formal analysis, S.L. and R.E.S.; investigation, S.L.; resources, R.E.S.; writing—original draft preparation, S.L.; writing—review and editing, S.L. and R.E.S.; visualization, R.E.S.; supervision, S.L.; project administration, S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Institute for Information and Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2018-0-00245, Development of prevention technology against AI dysfunction induced by deception attack) and by the Basic Science Research Program through the National Research Foundation of Korea under Grant NRF-2019R1I1A3A01060150.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. In Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
2. Akhtar, N.; Mian, A. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *IEEE Access* **2018**, *6*, 14410–14430. [CrossRef]
3. Chakraborty, T.; Jajodia, S.; Katz, J.; Picariello, A.; Sperli, G.; Subrahmanian, V.S. FORGE: A Fake Online Repository Generation Engine for Cyber Deception. *IEEE Trans. Depend. Secure Comput.* **2019**, 1–16. [CrossRef]
4. Boloor, A.; He, X.; Gill, C.D.; Vorobeychik, Y.; Zhang, X. Simple Physical Adversarial Examples against End-to-End Autonomous Driving Models. In Proceedings of the 15th IEEE International Conference on Embedded Software and Systems (ICESS), Las Vegas, NV, USA, 2–3 June 2019; pp. 1–7. [CrossRef]
5. Kurakin, A.; Goodfellow, I.J.; Bengio, S. Adversarial examples in the physical world. In Proceedings of the 5th International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
6. Papernot, N.; McDaniel, P.D.; Jha, S.; Fredrikson, M.; Celik, Z.B.; Swami, A. The Limitations of Deep Learning in Adversarial Settings. In Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P), Saarbrücken, Germany, 21–24 March 2016; pp. 372–387. [CrossRef]
7. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.J.; Fergus, R. Intriguing properties of neural networks. In Proceedings of the 2nd International Conference on Learning Representations (ICLR), Banff, AB, Canada, 14–16 April 2014.
8. Moosavi-Dezfooli, S.; Fawzi, A.; Frossard, P. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2574–2582. [CrossRef]
9. Carlini, N.; Wagner, D.A. Towards Evaluating the Robustness of Neural Networks. In Proceedings of the IEEE Symposium on Security and Privacy, San Jose, CA, USA, 22–26 May 2017; pp. 39–57. [CrossRef]
10. Papernot, N.; McDaniel, P.D.; Wu, X.; Jha, S.; Swami, A. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In Proceedings of the IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2016; pp. 582–597. [CrossRef]
11. Papernot, N.; McDaniel, P.D.; Goodfellow, I.J.; Jha, S.; Celik, Z.B.; Swami, A. Practical Black-Box Attacks against Machine Learning. In Proceedings of the Asia Conference on Computer and Communications Security, Abu Dhabi, UAE, 2–6 April 2017; pp. 506–519. [CrossRef]
12. Athalye, A.; Carlini, N.; Wagner, D.A. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In Proceedings of the 35th International Conference on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018.
13. Liao, F.; Liang, M.; Dong, Y.; Pang, T.; Hu, X.; Zhu, J. Defense Against Adversarial Attacks Using High-Level Representation Guided Denoiser. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 1778–1787. [CrossRef]
14. Samangouei, P.; Kabkab, M.; Chellappa, R. Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models. In Proceedings of the 6th International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
15. Bhagoji, A.N.; Cullina, D.; Mittal, P. Dimensionality Reduction as a Defense against Evasion Attacks on Machine Learning Classifiers. *arXiv* **2017**, arXiv:1704.02654.

16. Li, X.; Li, F. Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 5775–5783. [CrossRef]
17. Hendrycks, D.; Gimpel, K. Early Methods for Detecting Adversarial Images. In Proceedings of the 5th International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
18. Feinman, R.; Curtin, R.R.; Shintre, S.; Gardner, A.B. Detecting Adversarial Samples from Artifacts. *arXiv* **2017**, arXiv:1703.00410.
19. Xu, W.; Evans, D.; Qi, Y. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 18–21 February 2018.
20. Dathathri, S.; Zheng, S.; Murray, R.M.; Yue, Y. Detecting Adversarial Examples via Neural Fingerprinting. *arXiv* **2018**, arXiv:1803.03870.
21. Tian, S.; Yang, G.; Cai, Y. Detecting Adversarial Examples Through Image Transformation. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), New Orleans, LO, USA, 2–7 February 2018; pp. 4139–4146.
22. Metzen, J.H.; Genewein, T.; Fischer, V.; Bischoff, B. On Detecting Adversarial Perturbations. In Proceedings of the 5th International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
23. Grosse, K.; Manoharan, P.; Papernot, N.; Backes, M.; McDaniel, P.D. On the (Statistical) Detection of Adversarial Examples. *arXiv* **2017**, arXiv:1702.06280.
24. Gong, Z.; Wang, W.; Ku, W. Adversarial and Clean Data Are Not Twins. *arXiv* **2017**, arXiv:1704.04960.
25. Lu, J.; Issaranon, T.; Forsyth, D.A. SafetyNet: Detecting and Rejecting Adversarial Examples Robustly. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 446–454. [CrossRef]
26. Lee, S.; Kim, N.; Cho, Y.; Choi, J.; Kim, S.; Kim, J.; Lee, J. Adversarial Detection with Gaussian Process Regression-based Detector. *KSII Trans. Internet Inf. Syst.* **2019**, *13*, 4285–4299. [CrossRef]
27. Liang, B.; Li, H.; Su, M.; Li, X.; Shi, W.; Wang, X. Detecting Adversarial Examples in Deep Networks with Adaptive Noise Reduction. *arXiv* **2017**, arXiv:1705.08378.
28. Gittings, T.; Schneider, S.A.; Collomosse, J.P. Robust Synthesis of Adversarial Visual Examples Using a Deep Image Prior. In Proceedings of the 30th British Machine Vision Conference (BMVC), Cardiff, UK, 9–12 September 2019; p. 283.
29. Sutanto, R.E.; Lee, S. Adversarial Attack Defense Based on the Deep Image Prior Network. *Inf. Sci. Appl.* **2020**, *621*, 519–526.
30. Zhao, C.; Fletcher, P.T.; Yu, M.; Peng, Y.; Zhang, G.; Shen, C. The Adversarial Attack and Detection under the Fisher Information Metric. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, The Thirty-First Innovative Applications of Artificial Intelligence Conference, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI), Honolulu, HI, USA, 27 January–1 February 2019; pp. 5869–5876. [CrossRef]
31. Ma, X.; Li, B.; Wang, Y.; Erfani, S.M.; Wijewickrema, S.N.R.; Schoenebeck, G.; Song, D.; Houle, M.E.; Bailey, J. Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality. In Proceedings of the 6th International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
32. Athalye, A.; Engstrom, L.; Ilyas, A.; Kwok, K. Synthesizing Robust Adversarial Examples. In Proceedings of the 35th International Conference on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018; Volume 80, pp. 284–293.
33. Yin, X.; Kolouri, S.; Rohde, G.K. GAT: Generative Adversarial Training for Adversarial Example Detection and Robust Classification. In Proceedings of the 8th International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia, 26–30 April 2020.
34. Ulyanov, D.; Vedaldi, A.; Lempitsky, V.S. Deep Image Prior. *Int. J. Comput. Vis.* **2020**, *128*, 1867–1888. [CrossRef]
35. Finn, C.; Abbeel, P.; Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Proceedings of the 34th International Conference on Machine Learning, (ICML), Sydney, NSW, Australia, 6–11 August 2017; Volume 70, pp. 1126–1135.
36. The MNIST Database of Handwritten Digits. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 18 December 2020).
37. The CIFAR-10 Dataset. Available online: https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 18 December 2020).
38. IMAGENET. Available online: http://www.image-net.org/ (accessed on 18 December 2020).
39. Dogs vs. Cats. Available online: https://www.kaggle.com/c/dogs-vs-cats/data (accessed on 18 December 2020).