Computer Security Symposium 2019
21 - 24 October 2019

# A Label-based System for Detecting Adversarial Examples by Using Low Pass Filters

Dang Duy Thang[1,2,a)] Taisei Kondo[1,b)] Toshihiro Matsui[1,c)]

**Abstract:** Along with significant improvements in deep neural networks, image classification tasks are solved with extremely high accuracy rates. However, deep neural networks have been recently found vulnerable to well-designed input samples that called adversarial examples. Such this issue causes deep neural networks to misclassify adversarial examples that are imperceptible to humans. Distinguishing adversarial images and legitimate images are tough challenges. To address this problem, in this paper we proposed a new automatic classification system for adversarial examples. Our proposed system can almost distinguish adversarial samples and legitimate images in an end-to-end manner without human intervention. We exploit the important role of low frequencies in adversarial samples and proposing the label-based method for detecting malicious samples based on our observation. We evaluate our method on a variety of standard benchmark datasets including MNIST and ImageNet. Our method reached out detection rates more than 96% in many settings.

**Keywords:** Deep Neural Networks, Adversarial Examples, Low Pass Filter

## 1. Introduction

Deep Neural Networks (DNNs) were developed as a machine learning approach to many complex tasks. Traditional machine learning methods are successful when the final value is a simple function of the input data. Conversely, DNNs can capture the composite relations between millions of pixels and textual descriptions, brand-related news, future stock prices, and other contextual information. DNNs attain state-of-the-art performance in practical tasks of many domains, such as natural language processing, image processing, and speech recognition [1]. Current state-of-the-art DNNs are usually designed to be robust to noisy data; that is, the estimated label of a DNN output is insensitive to small noises in the data. Noise robustness is a fundamental characteristic of DNN applications in real, uncontrolled, and possibly hostile environments. However, recent research has shown that DNNs are vulnerable to specially-crafted adversarial perturbations (also known as adversarial examples) [2], [3], well-designed fluctuating inputs that are added to clean inputs. Developers of machine learning models assume a legitimate environment in both training and testing. Intuitively, the inputs $X$ are assumed to come from the same distribution during both training and test times. That is, if the test inputs $X$ are new and previously unseen during the training process, they at least have the same properties as the inputs used for training. These assumptions ensure a powerful machine learning model, but any attacker can alter the distribution during either the training time [4] or the testing time [5]. Typical training attacks [6] try to inject adversarial training data into the original training set. If successful, these data will wrongly train the deep learning model. However, most of the existing adversarial methods attack the testing phase [7], [8], which is more reliable than attacking the training phase. Especially, training-phase attacks are more difficult to implement and should not be launched without first exploiting the machine learning system. For example, an attacker might slightly modify an image [4], causing it to be recognized incorrectly, or adjust the code of an executable file to enable its escape by a malware detector [9]. Many researchers have developed defense mechanisms against adversarial examples. Madry et al. [8] applied a natural saddle-point method that guards against adversarial examples in a principled manner. They found that the network architecture affects the adversarial robustness of a DNN, so the robust decision boundary of the saddle-point problem can be more complicated than a decision boundary that simply categorizes the legitimate data. Preprocessing-based defense strategies against adversarial examples, which are the focus of our current work, will be reviewed and discussed in Sec. 2.

#### 1.0.1 Our Contributions

This paper introduces new techniques for overcoming adversarial examples. Our proposed system can automatically detect and classify both adversarial and legitimate samples. Assuming that most of the adversarial perturbations are created in the high frequencies of the image, we seek to reduce the high-frequency adversarial noises while retaining the be-

[1] Institute of Information Security, Yokohama, Japan
[2] University of Danang, Danang, Vietnam
[a)] dgs174101@iisec.ac.jp
[b)] mgs191002@iisec.ac.jp
[c)] matsui@iisec.ac.jp

nign high-frequency features. To prove our hypothesis, we first installed a low-pass filter layer between the adversarial example and target classifier. The probability of detecting the target class by the classifier dropped significantly (to nearly zero), but the recognition results of the primary class were retained. In Sec. 3, we demonstrate the correctness of these implementations in a theoretical proof. Based on the previous observation, we propose a new end-to-end system that automatically detects adversarial examples by using a filter layer inserted between the input and DNN, which traps suspicious noises. In parallel with this process, the un-filtered input is fed to the classifier and the highest-confidence class is marked as a marked label, on the other hand, input will be passed through the filter layer before feeding into the classifier. The output labels will be observed and named as checked labels. The marked label and checked labels are then compared, and the final decision on the input (adversarial or benign) is determined by the variation of labels.

The main contributions of this paper are as follows:

- We summarized the different attack strategies and we provided an intuitive overview of these current attack methods.
- We assumed that most of the adversarial perturbations are created at high frequencies. After implementing many experiments based on our theoretical framework, we confidently affirm our hypothesis.
- After thoroughly analyzing our experimental and theoretical observations, we created a new automated detection method for adversarial examples. Our approach differs from previous researches, in which the experimental steps are typically based only on the original hypothesis. Our approach was successfully applied to two types of common datasets: a small-scale dataset (MNIST) and a large-scale dataset (ImageNet). Our defense method accurately classified adversarial examples and legitimate samples. Moreover, in some cases, it recovered the high accuracy rates of the DNN classification.

## 2. Related Works and Background

### 2.1 Related Works

Removing the adversarial noises and regaining the recognition integrity of classifiers have been attempted in several works. Liao et al. [10] developed High-level representation Guided Denoiser (HGD) as a defense for image classification systems. They argued that many defense models cannot remove all adversarial perturbations, and that the non-removed adversarial noises are greatly amplified in the top layers of the target model. Consequently, the model will output a wrong prediction. To overcome this problem, they trained a denoiser by an HGD loss function. However, their proposed system was implemented only on ImageNet, which contains color images, and was not trialed on grayscale datasets such as MNIST. Although this omission is not highly important, the performance of a method based on high-level representation in a very deep neural network may degrade on grayscale images, whereas a simple neural network performs accurately on MNIST data. The strategy of Xu et al. [11], which they called "feature squeezing", reduces the number of degrees of freedom available to an adversary by squeezing out the unnecessary input features. The squeezing is performed by two denoisers performing different denoising methods: (1) squeezing the color bit depth, and (2) spatial smoothing. The prediction results are then compared with those of the target model, and the input is inferred as adversarial or legitimate. Although Xu et al. [11] evaluated their proposed method on various adversarial attacks, how they specified their thresholds on different benchmark datasets is unclear. Deciding appropriate thresholds for their system will overburden operators, and the method cannot easily adapt to new and unknown datasets.

### 2.2 Background

#### 2.2.1 Deep Neural Networks

In this subsection, we review neural networks in detail and introduce the required notation and definitions. Neural networks consist of elementary computing units named neurons organized in interconnected layers. Each neuron applies an activation function to its input, and produces an output. Starting with the input to the machine learning model, the output produced by each layer of the network provides the input to the next layer. Networks with a single intermediate hidden layer are called shallow neural networks, whereas those with multiple hidden layers are DNNs. The multiple hidden layers hierarchically extract representations from the model input, eventually producing a representation for solving the machine learning task and outputting a prediction. A neural network model $F$ can be formalized as multidimensional and parametrized functions $f_i$, each corresponding to one layer of the network architecture and one representation of the input. Specifically, each vector $\theta_i$ parametrizes layer $i$ of the network $F$ and includes weights for the links connecting layer $i$ to layer $i_1$. The set of model parameters $\theta = \{\theta_i\}$ is learned during training. For instance, in supervised learning, the parameter values are learned by computing the prediction errors $f(x) - y$ on a collection of known input–output pairs $(x, y)$.

#### 2.2.2 Adversarial Attacks

The adversarial examples and their counterparts are defined as indistinguishable from humans.

**C&W method.** Carnili et al. [7] proposed a new and powerful adversarial attack with several optimal perturbation settings. In this work, Carnili et al. used three distance metrics to approximate human's perception based on the $L_p$ norm:

$$||x||_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}} \tag{1}$$

Carnili et al. used $L_0, L_2, L_\infty$ metrics for expressing the different aspects of visual significance. $L_0$ counts the number of pixels with different values at corresponding positions in two images. It describes how many pixels are changed between the two images. $L_2$ is used for measuring the Euclidean distance between two images. And $L_\infty$ will help to measure the maximum difference for all pixels at corresponding positions in two images. There is no agreement on which distance metric is the best so it depends on the proposed algorithms.

**EAD:Elastic-Net Attacks.** EAD adversarial attack was invented by Pin-Yu Chen et al. [12] inspired from [7]. This paper use elastic-net regularization technique [13] that is widely used in solving high-dimensional feature selection problems to invent new attack method by extending from C&W method.

**L-BFGS.** Szegedy et al. [14] used a method name L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) to create targeted adversarial examples. This method minimize the weighted sum of perturbation size $\varepsilon$ and loss function $L(x^*, y_{target})$ while constraining the elements of $x^*$ to be normal pixel value.

**FGSM.** Goodfellow et al. [2] consumed that adversarial examples can be caused by cumulative effects of high dimensional model weights. They proposed a simple attack method, called Fast Gradient Sign Method (FGSM):

$$x^* = x + \varepsilon \cdot sign(\nabla_x L(x, y)) \tag{2}$$

where $\varepsilon$ denotes the perturbation size for crafting adversarial example $x^*$ from original input $x$. Given a clean image $x$, this method tries to create a similar image $x^*$ in $L_\infty$ neighborhood of $x$ that fools the target classifier. This leads to maximize loss function $L(x, y)$ which is the cost of classifying image $x$ as the target label $y$. The fast gradient sign method solves this problem by performing one step gradient update from $x$ in the input space with a small size of perturbation $\varepsilon$. Increasing $\varepsilon$ will lead to higher and faster attack success rate however it may also make your adversarial sample to be more different from the original input. FGSM computes the gradients for once, so it is much more efficient than L-BFGS. FGSM is very simple however it is fast and powerful for creating the adversarial examples.

**PGD.** Madry el al. [8] proposed an attack method named Projected Gradient Descent (PGD):

$$x^* = x + \delta \cdot (\bigtriangledown L(x, y)) \text{ respect to } \text{project}_{(x,\epsilon)}(x^*) \tag{3}$$

Where $\text{project}_{(x,\epsilon)}(x^*)$ defines a projection operator with parameter $x^*$ on the circle area around $x$ with radius $\epsilon$, $\delta$ is a clip value that is searched in a box $(x, \epsilon)$. FGSM is created based on Gradient Descent (GD) to maximize the loss function $L(x, y)$. And GD is a standard method to solve an unconstrained optimization problem. In another hand, PGD is a way to solve a constrained problem. Madry el al. [8] used PGD to propose a new adversarial attack method.

In this paper, we implement FGSM [2], PGD [8], CW_$L_2$ [7] means C&W method with $L_2$ norm optimization for searching adversarial perturbation, and EAD [12] methods in attack phase. The model is used to create adversarial attacks is called the attacking model. When the attacking model is the target model itself or contains the target model, the resulting attacks are white-box. In this work, we also implement our method in a white-box manner.

#### 2.2.3 Adversarial Defenses

Adversarial training of machine learning systems has been extensively researched [15], [16]. This strategy trains the models on adversarial examples to improve their attack robustness. Some researchers have combined data augmentation with adversarial perturbed data for training [14], [15], [16]. However, this training is more time consuming than traditional training on clean images alone, because it adds extra training dataset to the training set, which clearly extends the training time. In other defense strategies based on pre-processing, the perturbation noise is removed before feeding the data into a machine learning model. Meng et al. [17] proposed a two-phase defense model that first detects the adversarial input, and then reforms the original input based on the difference between the manifolds of the original and adversarial examples. Another adversarial defense direction is based on the gradient masking method [16]. By virtue of the gradient masking, this defense strategy typically ensures high smoothness in specific directions and neighborhoods of the training data, inhibiting attackers from finding the gradients of the good candidate directions. Accordingly, they cannot perturb the input in a damaging way. Papernot et al. [18] adapted distillation to adversarial defense and trained the target model on soft labels output by another machine learning model. Nguyen and Sinha [19] developed a gradient masking method to defend against C&W attacks [7], in which the noise is appended to the network logit layer. Gu et al. [3] proposed the deep contrastive network, which imposes a layer-wise contrastive penalty to achieve output invariance under input perturbations. However, methods based on gradient masking can be replaced by a substitute model (a copy that imitates the defended model), which attackers can train by observing the labels assigned by the defended model to inputs that are chosen carefully by the adversary.

## 3. Proposed End-to-End System

### 3.1 Attack phase

We consider the white-box targeted attack settings, where the attacker can fully access the model type, model architecture, all trainable parameters, and the adversary aims to change the classifier's prediction to some specific target class. The attackers use available information to identify the feature space where the model is vulnerable or try to find the victim decision boundaries. Then the victim model is exploited by altering a clean input by using adversarial example methods. To create adversarial samples that are misclassified by the machine learning model, an adversary with knowledge of the model's classifier $f$ and its trainable parameters. In this work, we use FGSM [2], PGD [8], CW [7], and EAD [12] methods for crafting adversarial examples.

We define classifier function $f : \mathbb{R}^n \rightarrow \Big[1...k\Big]$ that maps image pixel value vectors to a particular label. Then we assume that function $f$ has a loss function $L : \mathbb{R}^n \times \Big[1...k\Big] \rightarrow \mathbb{R}$. For an input image $x \in \mathbb{R}^n$ and target label $y \in \Big[1...k\Big]$, our system aims to solve the following optimization problem: $\delta + L(x + \delta, y)$ subject to $x + \delta \in \Big[0, 1\Big]^n$ , where $\delta$ is a perturbation noise that we add to the original image $x$. We have to note that this function method would yield the solution for $f(x)$ in the case of convex losses, however, the neural networks are non-convex so we end up with an approximation in this case. In this case, we use the output of the second-to-last layer logits for calculating the gradient instead of using the output of the Softmax. So our attack phase is denoted as algorithm 1 by using PGD. For FGSM and CW attacks, we use $||\delta_x||_2$ instead of $||\delta_x||_\infty$ and EAD attack still uses $||\delta_x||_\infty$ like PGD method. For FGSM, CW_L$_2$ and EAD attacks, the algorithm 1 will be executed without using project$_{(x,\epsilon)}(x^*)$. In the attacking phase, we set the learning rate for crafting adversarial examples is 0.01 that keeps adversarial noises are as small as possible and the iterative process is 500 times. From the clean images, we will create the targeted output images.

**Algorithm 1:** Crafting Adversarial Examples Algorithm

```
input      : $x$, $y_{true}$, $y^*$, $f$, $\epsilon$, $\alpha$
output     : $x^*$
parameter  : learning rate = 0.01, epochs = 500
1  $x \leftarrow x^*$ // initial adversarial sample
2  $\delta_x \leftarrow \vec{0}$ // initial perturbation factor
3  $iter \leftarrow 1$ // initial iteration counter
4  while $||\delta_x||_\infty < \epsilon$ and $f(x^*) \neq y^*$ and $iter <= epochs$ do
5      $x^* \leftarrow x + \delta \cdot sign(\bigtriangledown L(y^*|x^*))$
6      $x^* \leftarrow \text{project}_{(x,\epsilon)}(x^*)$
7      maximize $L(y^*|x^*)$ respect to $||\delta_x||_\infty$
8      $\delta \leftarrow clip(x^*, x - \epsilon, x + \epsilon)$
9      $iter \leftarrow iter + 1$
10 end
11 return $x^*$
```

## 3.2 Detection phase

To create a new benchmark dataset for our detection system, we combined benign images with the adversarial images created in the attack phase. Assuming that the adversarial noises are high- frequency features on the images, we targeted the high-frequency domains on the images while retaining all features in the low-frequency areas. Various common algorithms are available for reducing image noises before further processing such as classification. In this work, we investigate the two most well-known filters in image denoising studies: linear and non-linear filters. For example, consider a new array with the same dimensions as the specified image. Fill each location of this new array with the weighted sum of the pixel values from the locations surrounding the corresponding location in the image, using a constant weight set. The result of this procedure is shift-invariant meaning that the output value depends on the pattern (not the positions) of the image neighborhood. It is also linear, meaning that summing the two images yields the same output as summing the separate outputs of both images. This procedure, known as linear filtering, smooths the noises in the images. One famous linear filter is the Gaussian filter, defined as

$$G_\sigma(i, j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}. \tag{4}$$

Here, $i, j$ denotes the coordinate signal of the input and $\sigma$ is the standard deviation of the Gaussian distribution. Alternatively, noise removal can be considered as filtering by a statistical estimator. In particular, the goal is to estimate the actual image value of a pixel in a noisy measurement scenario. The class of noise-removal filters is difficult to analyze but is extremely useful. Smoothing an image by a symmetric Gaussian kernel replaces a pixel value with some weighted average of its neighbors. If an image has been corrupted by stationary additive zero-mean Gaussian noise, then this weighted average can reasonably estimate the original value of the pixel. The expected noise response is zero. Weighting the spatial frequencies provides a better estimate than simply averaging the pixel values. However, when the image noise is not stationary additive Gaussian noise, the situation becomes more complicated. In particular, consider that a region of the image has a constant dark value with a single bright pixel composed of noise. After smoothing with a Gaussian, a smooth, Gaussian-like bright bump will be centered on the noise pixel. In this way, the weighted average can be arbitrarily and severely affected by very large noise values. The bump can be rendered arbitrarily bright by introducing an arbitrarily bright pixel, possibly by a transient error in reading a memory element. When this undesirable property does not develop, the estimator outputs robust estimates. The most well-known robust estimator computes the median of a set of values from its neighborhood. A median filter assigns a neighborhood shape (which can significantly affect the behavior of the filter). As in convolution, this neighborhood shape is passed over the image, but the median filter replaces the current value of the element by a median of the neighborhood values. For the neighborhood surrounding $(i,j)$, the filter is described by:

$$x_{ij} = median(X_{uv}|X_{uv} \in \mathbb{N}_{ij}), \tag{5}$$

where $X_{uv}$ denotes the neighborhood points of $x_{ij}$. Any adversarial noises can be attenuated by smoothing the pixels in the image. When adversarial noises are absent, smoothing the pixels does not severely affect the input-image quality, so the target classifier still recognizes the correct label. We name this process the sieve process.

Our proposed detection system runs the filter and mark processes in parallel. The filter process arrests the high frequencies in the input processing while the marking process transfers the input directly to the machine learning model.

The highest-confidence class from the classifier is assigned as the marked label. The filter process then tracks the labels similar to the marked label. If the marked label is equal to all filtered labels, our system confidently determines the input as benign, but in the case, there are more than two different marked labels and they are different with the marked label, out the detection system will point out it as adversarial example. Our system proceeds by Algorithm 2, where x defines the input image, $L$ is filtered labels, $l_{marked}$ is marked label, $\kappa$ denotes the kernel sizes, $f$ is a machine learning function that computes the predicted label with the highest probability, and $s$ is the filter function. The filter function based on the Gaussian filter is called the label-based Detection System based on Gaussian (LDG); the other filter function is Detection System based on Median (LDM).

**Algorithm 2:** The Label-based System for Detecting Adversarial Examples by Using Low Pass Filters

```
input      : $x, s, f, L$
output     : $0, 1$
// 0: benign; 1: adversarial
parameter  : $\kappa = [(3 \times 3); (5 \times 5)]$
1  $l_{marked} \leftarrow f(x)$
2  for $i$ in $\kappa$ do
3      $x_{filtered} \leftarrow s(x, i)$
4      $l_{filtered} \leftarrow f(x_{filtered})$
5      $L_i \leftarrow l_{filtered}$
6  end
7  if $l_{marked} == \forall l_i \in L$ then
8      return 0
9  else
10     return 1
11 end
```

## 4. Implementation and Results

### 4.1 Datasets

The classification task was evaluated on two common benchmark datasets, namely, MNIST and ImageNet.

#### 4.1.1 Setup of MNIST

The MNIST dataset [20] includes 70,000 gray images of hand-written digits ranging from 0 to 9. It is separated into 60,000 training images and 10,000 testing images. A single MNIST image is composed of 28 × 28 pixels, each encoded by an 8-bit grayscale. We randomly extracted 200 images of the digit "0" from the 10,000 testing images. From each of these 200 images, we created nine adversarial images targeting the remaining digits (1-9). Finally we created a new benchmark dataset of 2,000 images (200 benign images and 1,800 adversarial images).

#### 4.1.2 Setup of ImageNet

We consider the ImageNet dataset [21] that is a very large database designed for use in visual object recognition research. The original ImageNet includes more than 14 million images in 20,000 categories with a typical category, such as "koala" or "spindle", consisting of several hundred images. The machine learning model that we use is Google Inception V3 [22] that was trained with 1,000 common categories ImageNet. We randomly select 360 testing images. By applying FGSM, PGD, CW_L$_2$ and EAD methods to craft adversarial images target to randomly targeted labels, we generate new 1,440 adversarial images. We combine them together to form a new benchmark repository including 1,800 images for our experiment.

### 4.2 Implementation

Although adversarial examples have recently attracted much interest from researchers, a public benchmark dataset for evaluating the robustness of defense systems remains lacking. In the attack phase of our system, we thus created a new benchmark dataset for evaluating the detection capabilities of our detection system.

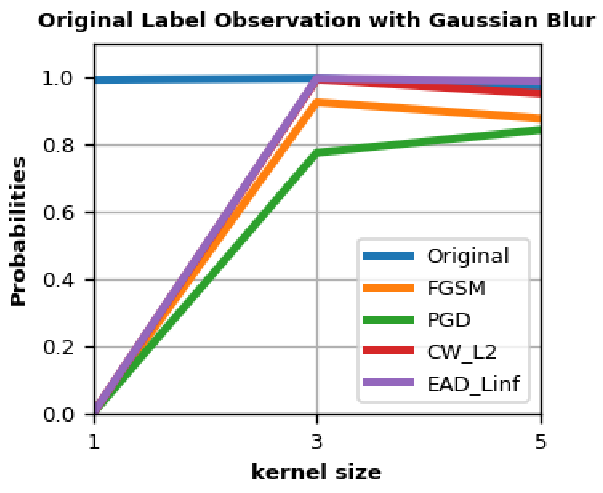
The 200 random images of digit "0" extracted from the MNIST dataset were converted to adversarial images of digits 1-9 by the FGSM method. The FGSM was run through 1,000 iterations (epochs). The adversarial images were combined with original images into the new benchmark dataset for evaluating our detection system. The proposed detection system knows the true labels of the input. When presented with the unknown input, our system automatically processes the input and returns a decision (adversarial or benign).

In the ImageNet dataset, from 360 random testing images, we created 1,440 adversarial images with a randomly targeted label by using FGSM, PGD, CW_L$_2$ and EAD methods. The number of iteration (epochs) is 500 times. We end up with a new synthesize evaluation dataset that includes 1,800 images.

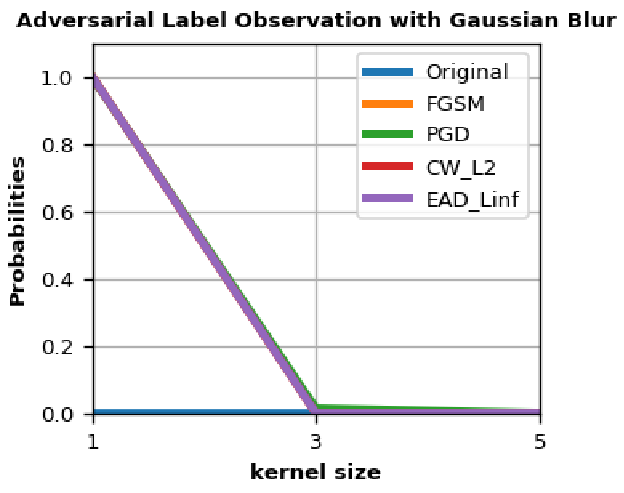
### 4.3 Results

Our results were compared with those of Xu et al. [11]. Our system is more convenient than Xu's system, owing to its high detection accuracy and easy setup. Specifically, our system adopts a fixed threshold whereas Xu et al.'s system must adopt the threshold value to individual cases. The performance of our system was evaluated by the F1-score.

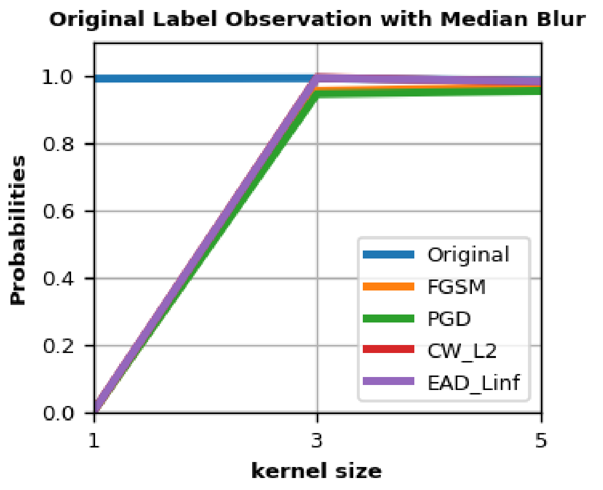
In the observation phase, we observed and analyzed a typical tench image. From a benign tench image with a detection probability of 99.08%, we created four adversarial images with the targeted label is a walking stick that is randomly selected. Afterward, the adversarial noises were stuck by the filter function, and the "tench" features were regained. As shown in Fig. A·1, the probabilities of the targeted "walking stick" and legitimate "tench" dramatically differed when processed by the filter functions. When an input is Original "tench", classification probability is 99.08% for "tench" label and the probabilities for the true label are still remained more than 90% after using a gaussian and median filter. Conversely, with an adversarial image with targeted class "walking stick", filters not only remove adversarial noises but only regain the probabilities of true label nearly equal to when using original input. This observation confirms our assumption that adversarial noises are high-frequency noises, and that adversarial samples are



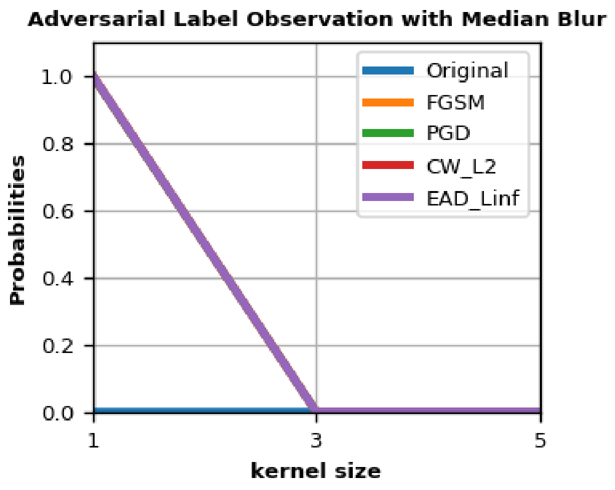
(a) Probabilities of original Tench label with Gaussian Filters



(b) Probabilities of targeted walking stick label with Gaussian Filters



(c) Probabilities of original Tench label with Median Filters



(d) Probabilities of targeted walking stick label with Median Filter

Fig. 1: Adversarial examples (true class: Tina Tench) suffers to our filter layer

powerfully detected by adopting the low-pass filter in our model.

The detection results on the MNIST dataset are reported in Table 1. The dashes in this table signify a lack of information from earlier research. Although the same number of images was compared in ours and Xu et al.'s methods, we created a more challenging test set than Xu et al. [11]. Whereas Xu et al. created a balanced dataset of 1,000 legitimate images and 1,000 adversarial examples, we created 1,800 adversarial images from 200 legitimate inputs, thus imposing an imbalanced [23] dataset in our experimental test. Nevertheless, our detection rates are highly competitive with those of Xu et al. and slightly surpass the earlier detection rates. Moreover, our system applies a fixed threshold for all settings, whereas in Xu et al.'s work, the threshold must be adjusted in different settings.

On the ImageNet dataset, our detection rates exceeded those of Xu et al. As highlighted in Table 1, we analyzed more files in this implementation than Xu et al., while maintaining the imbalance in our benchmark dataset. Our detection rate was 96.9% and 99.6% with LDG and LDM, respectively, greatly outperforming Xu et al.'s system.

## 5. Conclusion

We investigated the high-frequency noises in adversarial image examples. Based on the high-frequency noise assumption and a theoretical framework, we demonstrated the effectiveness of a low-pass filter in removing these noises. This observation guided the development of our automated detection system for adversarial examples. On the MNIST and ImageNet datasets, our system achieved maximum accuracy rates of 99.2% and 99.6%, respectively. For evaluating our system, we constructed new benchmark datasets posing more challenges than previously constructed datasets [10], [11]. Whereas the earlier studies evaluated their systems on images from the training set, our evaluation employed the testing images. As another important contribution to the existing datasets, our system not only defeated adversarial noises but also regained the legitimate class from adversarial examples.

Table 1: Detection Results

(a) MNIST

| | Our Method | | Xu et al. [11] | | |
|---|---|---|---|---|---|
| | LDG | LDM | Bit-D | Smooth | Best-J |
| No. Files | 2,000 | 2,000 | 2,000 | 2,000 | 2,000 |
| Threshold | NA | NA | 0.0005 | 0.0029 | 0.0029 |
| TP | 1,786 | 1,776 | - | - | - |
| TN | 199 | 182 | - | - | - |
| FP | 1 | 18 | - | - | - |
| FN | 14 | 24 | - | - | - |
| Accuracy | 0.993 | 0.979 | - | - | - |
| Precision | 0.999 | 0.999 | - | - | - |
| Recall | **0.992** | 0.987 | 0.903 | 0.868 | 0.982 |
| F1 score | 0.996 | 0.988 | - | - | - |

(b) ImageNet

| | Our Method | | Xu et al. [11] | | |
|---|---|---|---|---|---|
| | LDG | LDM | Bit-D | Smooth | Best-J |
| No. Files | 1,800 | 1,800 | 1,800 | 1,800 | 1,800 |
| Threshold | NA | NA | 1.4417 | 1.1472 | 1.2128 |
| TP | 1,395 | 1,434 | - | - | - |
| TN | 319 | 294 | - | - | - |
| FP | 41 | 66 | - | - | - |
| FN | 45 | 6 | - | - | - |
| Accuracy | 0.952 | 0.960 | - | - | - |
| Precision | 0.971 | 0.956 | - | - | - |
| Recall | 0.969 | **0.996** | 0.751 | 0.816 | 0.859 |
| F1 score | 0.970 | 0.976 | - | - | - |

**Acknowledgments** We would like to thank Professor Akira Otsuka for his helpful and valuable comments. This work is supported by the Iwasaki Tomomi Scholarship.

## References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[2] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations ICLR*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6572

[3] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *CoRR*, vol. abs/1412.5068, 2014.

[4] C. Xiao, B. Li, J. yan Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 7 2018, pp. 3905–3911.

[5] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013, pp. 387–402.

[6] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. ACM, 2011, pp. 43–58.

[7] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (S&P 2017)*. IEEE, 2017, pp. 39–57.

[8] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[9] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 62–79.

[10] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR*, 2018, pp. 1778–1787.

[11] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21*, 2018.

[12] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "Ead: elastic-net attacks to deep neural networks via adversarial examples," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[13] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the royal statistical society: series B (statistical methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[14] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations ICLR*, 2014.

[15] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *International Conference on Learning Representations ICLR*, 2017.

[16] A. Kurakin, D. Boneh, F. Tramr, I. Goodfellow, N. Papernot, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *International Conference on Learning Representations ICLR*, 2018.

[17] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 135–147.

[18] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (S&P 2016)*. IEEE, 2016, pp. 582–597.

[19] L. Nguyen, S. Wang, and A. Sinha, "A learning and masking approach to secure learning," in *International Conference on Decision and Game Theory for Security*. Springer, 2018, pp. 453–464.

[20] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, 2010.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[22] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[23] Y. Sun, A. K. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 04, pp. 687–719, 2009.

## Appendix

### A.1 Our Observation on Tench Image

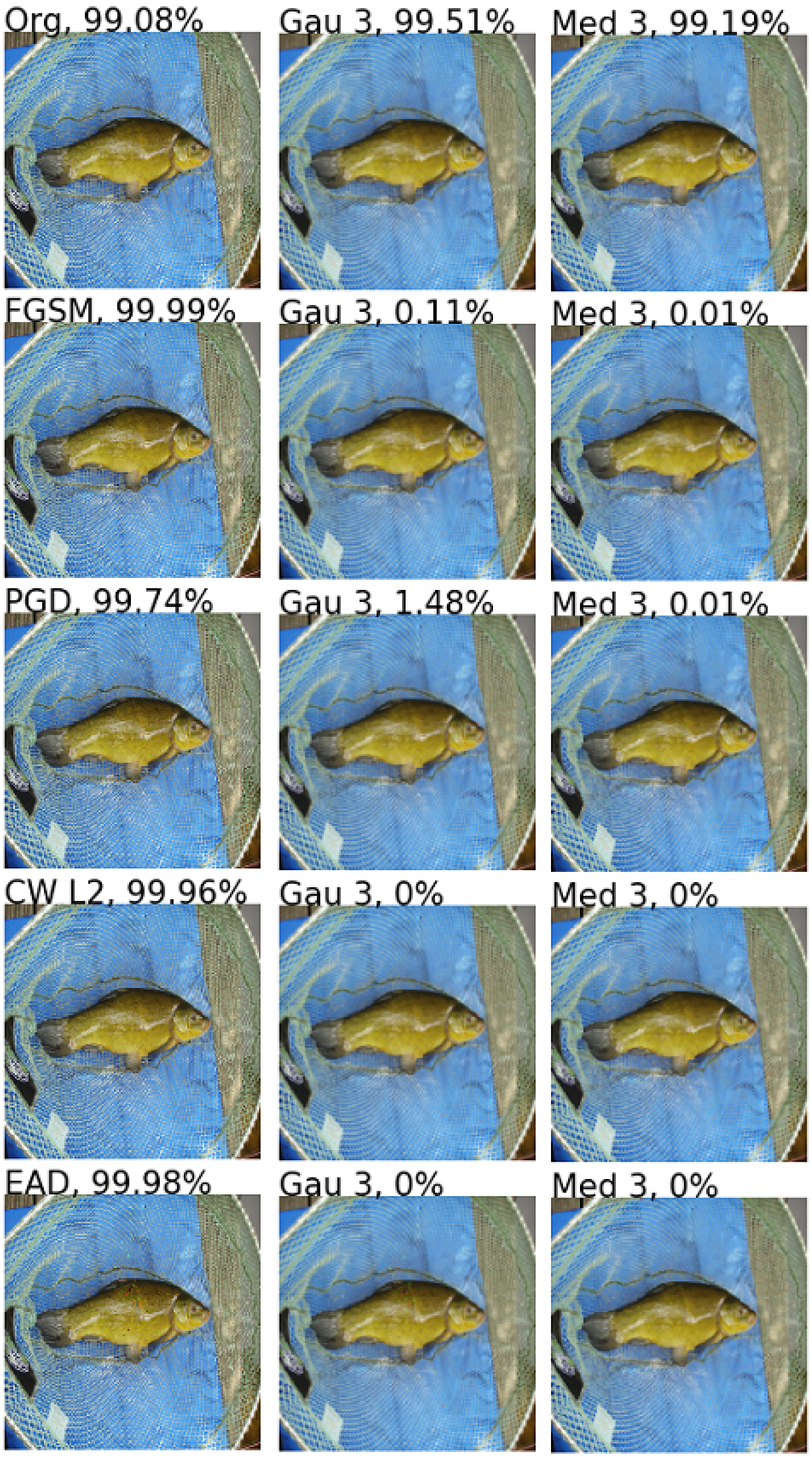


Fig. A·1: Our observation on original tench image and adversarial walking stick images that created by FGSM [2], PGD [8], CW_$L_2$ [7], and EAD [12] methods and when we use Gaussian and Median filters with kernel size is 3x3. First row, percentage values illustrate classification rates for tench label. Other rows, percentage values are classification rates for walking stick label.