Hindawi
Security and Communication Networks
Volume 2022, Article ID 5501035, 10 pages
https://doi.org/10.1155/2022/5501035

Research Article

From Spatial to Spectral Domain, a New Perspective for Detecting Adversarial Examples

Zhiyuan Liu,[1,2] Chunjie Cao ,[1,2] Fangjian Tao,[1,2] Yifan Li,[1,2] and Xiaoyu Lin[1,2]

[1]*School of Cyberspace Security (School of Cryptology), Hainan University, Haikou 570228, Hainan, China*
[2]*Key Laboratory of Internet Information Retrieval of Hainan Province, Haikou 570228, Hainan, China*

Correspondence should be addressed to Chunjie Cao; caochunjie@hainanu.edu.cn

Received 26 March 2022; Revised 2 June 2022; Accepted 30 June 2022; Published 5 September 2022

Academic Editor: Yuqing Zhang

Deep neural networks (DNNs) have been closely related to the Pandora's box from the moment of its birth. Although it achieves a high accuracy significantly in real-world tasks (e.g., object detecting and speech recognition), it still retains fatal vulnerabilities and flaws. Malicious attackers can manipulate DNN model misclassification just by adding tiny perturbations to the original image. These crafted samples are also called adversarial examples. One of the effective defense methods is to detect them before feeding them into the model. In this paper, we delve into the representation of adversarial examples in the original spatial and spectral domains. By qualitative and quantitative analysis, it is confirmed that the high-level representation and high-frequency components of abnormal samples contain richer discriminative information. To further explore the influence mechanism between the two factors, we perform an ablation study and the results show a win-win effect. Utilizing the finding, a detecting method (HLFD) is proposed based on extracting high-level representation and high-frequency components. Compared with other state-of-the-art detection methods, we achieve a better detection performance in most scenarios via a series of experiments conducted on MNIST, CIFAR-10, CIFAR-100, SVHN, and Tiny-ImageNet. In particular, we improve detection rates by a large margin on DeepFool and CW attacks.

1. Introduction

Exploring the inherent patterns and changing trends of data is an eternal subject for all human beings. The traditional way of processing data is driven by rules which obtain through experience or manual summary. The advent of DNNs makes it possible to process the data automatically. Relying on the characteristic, it has recently been widely applied in data-sensitive fields, such as financial payment [1], medical assistance [2], and satellite remote sensing [3]. On the contrary, the benefit of its automation also brings the property of a black-box [4–6], which means almost everything inside the network is unknown to us. Against this background, Goodfellow et al. found in [7] that adversarial examples, imperceptible for a human observer, produced by adding crafted perturbations to benign images were able to misclassify the DNNs model with high confidence. There is no doubt that how to defend against these out-of-distribution samples has become a top priority subject.

Before implementing adversarial defenses, exploring the intrinsic properties of adversarial examples is crucial. At present, there are two main explanations for the appearance of adversarial samples: low-probability regions in manifold [8] and linear explanations [7]. These studies all focused on the distribution of spatial probability (i.e., statistical probability) to make them reasonable. However, the latest studies in [9–15] indicated that adversarial examples are mainly concentrated in the high-frequency region. Moreover, Ilyas et al. further illustrated in [16] that adversarial examples are not even bugs. They are non-robust features, which means we can learn them via training a DNN model.

Inspired by these studies, we initiate to explore the distinction of the representation of adversarial examples in the original spatial and the spectral domain. In fact, the spatial and the spectral domain refer to the original samples and the original samples after Fourier transform, respectively. Intuitively, we discover that adversarial examples have many small black dots in the mid-high frequency region. By performing cluster analysis, we further discover that adversarial examples can be better classified in the spectral domain. For promoting the detection performance, we further analyze the impact of different layers of network and frequency bands on the detection performance. The experimental results surprisingly demonstrate that extracting high-level representation and high-frequency components can improve the detection performance significantly.

In this paper, we propose the detection method HLFD to detect the abnormal samples based on extracting high-level representation and high-frequency components. Using the high-level feature maps of the model as input, we transform them to the spectrum by Fourier transform and extract the high-frequency components. A detector with an ideal performance will be born after training these transformed data. Compared with other defense methods, there is no need to alter the network architecture and the lower computational cost, which are its superiority. The overview of the detection model is shown in Figure 1.

We evaluate our method on six different attacks, the fast gradient sign method (FGSM) [7], two of its variants, the basic iterative method (BIM) [17], the projected gradient descent (PGD) [18], Jacobian-based Saliency Map Attack (JSMA) [19], Carlini and Wagner (CW) [20], and DeepFool [21] methods. Using only one of the tricks of high-frequency extraction or high-level representation cannot achieve the ideal detection performance on DeepFool and CW, although we perform well in FGSM, BIM, PGD, and JSMA attacks. Considering that high-level representation and high-frequency components may affect each other, we further perform an ablation study for the two factors. Experimental result shows a win-win effect which means a better performance after applying the two tricks. DeepFool and CW attacks can be detected efficiently by employing the two factors simultaneously. For a more rigorous conclusion, the detector is evaluated on five datasets: MNIST, CIFAR-10, CIFAR-100, SVHN, and Tiny-ImageNet (aka T-ImageNet).

For a fair comparison, adversarial examples are restricted to similar L_2 norm values and employ three state-of-the-art detectors, kernel density and Bayesian uncertainly (KD + BU) [22], local intrinsic dimensionality (LID) [23], and Mahalanobis distance (M-D) [24] as a contrast. Our method outperforms the other three detection methods for each attack on CIFAR-10 and CIFAR-100. Although our detection rate is not the highest in some scenarios, the gap is not large at least. Moreover, we improve the detection rates by a large margin on DeepFool and CW attacks.

In particular, our main contributions are as follows:

(i) We intuitively and experimentally prove that the spectral samples have richer discriminative details, which can be effectively distinguished by the detector.

(ii) The ablation study shows that the detection performance can be improved effectively whether using high-level representation or high-frequency extraction.

(iii) An effective method for detecting adversarial examples is proposed, which performs better in most scenarios compared to the state-of-the-art.

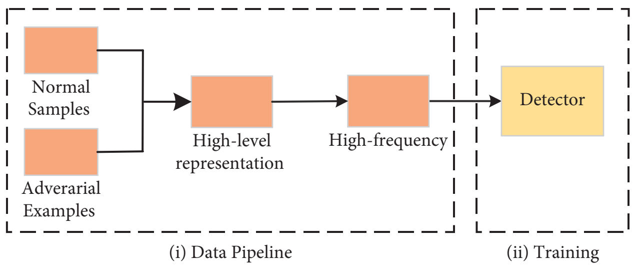

FIGURE 1: A overview framework of detection model HLFD. It consists of two modules: data pipeline and training process.

2. Related Work

In this section, we will briefly introduce several state-of-the-art methods for adversarial attacks and adversarial defenses.

2.1. Adversarial Attack

2.1.1. Fast Gradient Sign Method (FGSM) [7]. Goodfellow et al. purposed fast gradient sign method (FGSM), one of the simplest ways based on gradient, to generate adversarial examples. One can obtain them only by computing the direction of the gradient of the loss function. The method can be expressed as

$$X^{\mathrm{adv}} = X + \varepsilon \cdot \operatorname{sign}\left(\nabla_X J(\theta, X, Y)\right). \tag{1}$$

2.1.2. Basic Iterative Method (BIM) [17]. In most cases, adversarial samples generated solely by the FGSM method are ineffective. As a variant of FGSM, it performs multiple gradient computations on the direction of loss function and can be represented as follows:

$$\begin{aligned} X_0^{\mathrm{adv}} &= X, \\ X_{N+1}^{\mathrm{adv}} &= \mathrm{Clip}_{X,\varepsilon}\left\{X_{N+1}^{\mathrm{adv}} + \alpha \cdot \operatorname{sign}\left(\nabla_X J\left(X_N^{\mathrm{adv}}, Y_{\mathrm{true}}\right)\right)\right\}, \end{aligned} \tag{2}$$

where N, α refer to the number of iterations and the step size of each iteration, respectively, and $J(X, Y_{\mathrm{true}})$ means the cross-entropy loss function on a given image X and label Y. The norm of $(X_{N+1}^{\mathrm{adv}} - X)$ is limited to ε by the clipping function.

2.1.3. Projected Gradient Descent (PGD) [18]. PGD is an advancement of BIM, which alters an initialized with uniform random noise.

2.1.4. Jacobian-Based Saliency Map Attack (JSMA) [19]. Utilizing the Jacobian matrix, Papernot et al. put forward Jacobian-based Saliency Map Attack (JSMA). It mainly adopts the priori probability of the last layer to backpropagate, thereby obtaining the corresponding gradient information.

$$S(X,t)[i] = \begin{cases} 0 \text{ if } \dfrac{\partial F_t(X)}{\partial X_i} < 0 \text{ or } \displaystyle\sum_{j \neq t} \dfrac{\partial F_j(X)}{\partial X_i} > 0, \\ \left(\dfrac{\partial F_t(X)}{\partial X_i}\right) \left| \displaystyle\sum_{j \neq t} \dfrac{\partial F_j(X)}{\partial X_i} \right| \text{otherwise,} \end{cases} \quad (3)$$

where t is the class to be attacked. By constructing a saliency map S, the pixels which contribute the most to the result can be found. Adversarial examples can be generated via exploiting the information.

2.1.5. Carlini and Wagner (CW) [20]. CW abstracts the adversarial attack task into an optimization problem. On the basis of guaranteeing the model misclassify, constantly seeking for the smallest adversarial perturbation is its key idea. We formulate it as follows:

$$\min \left\| \frac{1}{2}\left(\tanh\left(X^{\text{adv}}\right) + 1\right) - X \right\|_2^2 + c \cdot f\left(\tanh\left(X^{\text{adv}}\right) + 1\right),$$
$$f(x) = \max\left(Z(x)_{\text{true}} - \max_{i \neq \text{true}}\{Z(x)_i\}, 0\right), \quad (4)$$

where $Z(x)$ is the output of the pre-softmax layer. Utilizing the $\tanh(\cdot)$ function, map the X^{adv} to $[-1, 1]$, thereby avoiding the loss caused by truncation.

2.1.6. DeepFool [21]. Compared with other attack methods, DeepFool is known for its minimal disturbances, which makes adversarial examples harder to detect. It stops seeking when the samples just cross the decision boundary as described by the formula below:

$$\Delta\left(X, X^{\text{adv}}\right) = \arg\min_Z \|Z\|_2, \text{subject to} : g(X + Z) \neq g(X), \quad (5)$$

where Z is actually the smallest perturbation.

We apply a python tool Foolbox [25] to generate adversarial examples for all attack methods. An intuitive rendering of the comparison of various attack methods is represented in Figure 2.

2.2. Adversarial Defense and Detection. In general, adversarial defenses can be mainly divided into two categories. One strategy is to modify the architecture or parameters of the network, and the other is to defend by preprocessing benign images before feeding them into the model. Adversarial training belonging to the first strategy [26, 27] has achieved great success in the defensive areas. The model will be retrained on normal and abnormal samples to learn the decision boundary details, thereby avoiding misclassification and having stronger robustness. Nonetheless, massive data support and ineffectiveness against specialized attacks have made it less attractive. It did not take long for

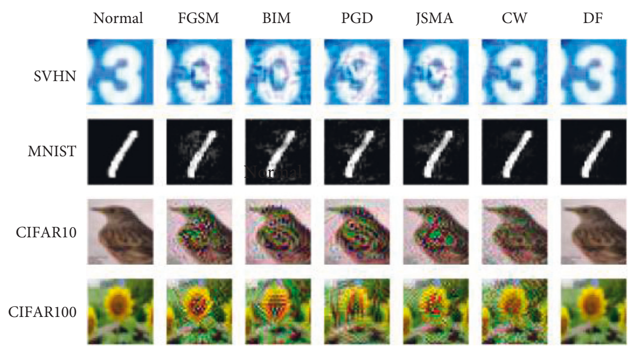

Figure 2: Comparison of adversarial examples under various attacks. The rows and columns represent different datasets and various attack methods, respectively.

adversarial distillation [28] to be raised. Although the experiment conducted on small datasets showed that it can defend against adversarial examples effectively, it was limited to be used in DNN models with probability distribution vectors. Methods belonging to the latter, like JPG image compression [29], rejecting classification [30], and detecting [22–24, 31–33] are trying to eliminate the abnormal statistical characteristics before poisoning the model. The defense methods without touching the training process are undoubtedly more exciting.

As one of the approaches in the defensive field, adversarial detection has attracted the attention of scholars due to its higher flexibility and lower computation. Sample statistics and training a detector are two main routes. Exploiting the statistical properties, Feinman et al. dived into the kernel density (KD) and Bayesian uncertainly (BU) in the hidden layers of the model and purposed an effective detection method in [22]. Ma et al. further applied local intrinsic dimension (LID) in [23] to describe the intrinsic characteristics of adversarial subspaces. Considering that the information of the last layer may not be enough to judge the out-of-distribution data, Lee et al. in [24] made full use of each layer of DNN and obtained a detector via calculating Mahalanobis distance (M-D). Hendrycks and Gimpel indicated in [34] that samples with a large principal component had higher weights to attack successfully. Still, the latest research showed that DNNs were sensitive to the direction of the Fourier basis function. In [9, 10], it was found that the high-frequency components of adversarial examples affected seriously on the robustness of the model. On the assumption that each layer of DNN obeyed the generalized Gaussian distribution, Ma et al. in [35] calculated the Benford-Fourier coefficients of each layer, thereby obtaining a support vector machine with an ideal detection performance.

3. Methodology

In this section, we will introduce the detection mechanism in detail to identify adversarial examples.

3.1. Threat Model. Existing studies are constantly exploring how to generate adversarial examples. Fortunately, we can summarize into two main points: make the model misclassify and minimize the disturbance as much as possible. Suppose $X \in R^n$ as a n-dimensional input image, $Z \in R^n$ as a perturbation of X, and $M(\cdot)$ as a model trained by X. If $M(X) \neq M(X+Z)$, we can define that $X^{\mathrm{adv}} = X + Z$ is the adversarial example specified to model $M(\cdot)$ and normal image X. However, if the perturbation Z is so large that neither the model nor the human eye can correctly identify, the adversarial example X^{adv} has become no practical significance. Hence, our objective shall be expressed like

$$\begin{aligned} &\arg\min_{Z} \|Z\|_p \text{ subject to } M(X) \neq M(X+Z), \\ &\|Z\|_p = \left(|Z_1|^p + |Z_2|^p + \cdots + |Z_n|^p\right)^{1/p}, \end{aligned} \tag{6}$$

where Z_i means the ith dimensional value of Z and $\|Z\|_p$ is the L_p norm of Z. In general, the norm is frequently utilized to limit the increase in perturbation. In this paper, we apply the L_2 norm for all adversarial attacks.

For a detection task, suppose $D(\cdot)$ as a detector which is a binary classifier essentially. An ideal detector can classify normal images as label 0 (i.e., $D(X) = 0$) and abnormal images as label 1 (i.e., $D(X^{\mathrm{adv}}) = 1$). We formulate this objective as follows:

$$\arg\max_{D(\cdot)} \frac{1}{m} \cdot \sum_{i=0}^{m-1} \left(D(X_i) \neq D\left(X_i^{\mathrm{adv}}\right)\right), \tag{7}$$

where m refers to the number of samples and $D(X)$ represents the result of feeding X into detector D. If $D(X_i) \neq D(X_i^{\mathrm{adv}})$, return true, otherwise returns false. $D(\cdot)$ is exactly the detector we seek and the maximizing result will be employed as one of our evaluation metrics, which is also called detection accuracy.

3.2. High-Level Representation.

3.2. High-Level Representation. As shown in (7), making the detector D identify as many samples as possible is our objective. In general, there are two strategies to achieve the purpose: transform the input data and alter the internal structure of the detector. Through the subsequent experiments shown in Figure 8, it was found that altering the detector model would not improve the detection performance significantly. Hence, feature engineering on the input data turns into our principal subject. Surprisingly, a tiny attempt to extract high-level representation from the raw data breaks the technical difficulty. As Harder et al. illustrated in [11], high-level representation has more stable and robust discriminative details for adversarial detection.

Suppose $M(\cdot)$ as a DNN model trained by X, we can obtain the mth feature map via calculating the $M_m(X)$ simply. Using $M_m(X)$ instead of X as the input data, (7) will have a higher value, which means better detection accuracy under the condition of the same training time.

3.3. Fourier Transform and High-Frequency Extraction

3.3.1. Fourier Transform. In general, Fourier transform is resorted to transform signals between the time domain (or spatial domain) and the frequency domain. After converting data into the spectrum, multiply characteristics hidden in the spatial domain are revealed. The low-frequency components correspond to slowly changing regions (i.e., the flat regions), while the high-frequency components do the opposite (i.e., the edges or noise). Exploiting these properties, we can obtain blurred or edge sharpened images, respectively, by suppressing high or low-frequency components. Still, unlike continuous mathematical signals, images are discrete data consisting of pixels, which means we shall convert it using discrete Fourier transform (DFT). For a low computational cost, we employ the fast Fourier transform (FFT) [36] which has a time complexity of $O(N \cdot \log(N))$. Suppose an image $X \in [0,255]^{M\times N}$, where M, N represent the width and height of the image, respectively. We can acquire the Fourier coefficient by the following formula:

$$F(X)(l,k) = \sum_{n,m=0}^{N} e^{-2\pi i l m + k n/N} X(m,n), \tag{8}$$

where $l,k = 0,1,\ldots,N-1$ and $X(m,n)$ refers to the pixel value of the coordinate (m,n). $F(X)$ is actually a complex matrix with the same size as image X. The magnitude matrix $|F(X)|$ will be acquired via calculating the following formula:

$$|F(X)(l,k)| = \sqrt{\mathrm{Real}(F(X)(l,k))^2 + \mathrm{Image}(F(X)(l,k))^2}, \tag{9}$$

where $\mathrm{Real}(\cdot)$ and $\mathrm{Image}(\cdot)$ refer to the real and imaginary parts, respectively. In subsequent experiments, the magnitude of the spectrum $|F(X)|$ will be applied to represented the spectral domain.

3.3.2. High-Frequency Extraction. Due to the conjugate property of FFT, its effective spectrum only accounts a quarter of $|F(X)|$ for a two-dimensional image. We divide the effective spectrum into four parts (a), (b), (c), and (d), as shown in Figure 3, according to low, medium-low, medium-high, and high-frequency, respectively. For a fair division, it is necessary to ensure that each part occupies 25% pixels of the image. Hence, we will introduce a threshold function $\varphi(;R)$ that separates the frequency components according to the radius R. Suppose the effective spectrum $X_e \in R^{N\times N}$, the formal definition of equation $\varphi(X_e;R)$ is as follows:

$$\phi(X_e; r_L, r_R) = \begin{cases} X_e(i,j), & \text{if } r_L < d((i,j),(N-1,0)) \leq r_R, \\ 0, & \text{otherwise,} \end{cases} \tag{10}$$

where $X_e(i,j)$ represents the effective spectrum X_e at position (i,j) and $(N-1,0)$ is exactly the lower left of the effective spectrum. $d(\cdot,\cdot)$ refers to the Euclidean distance. By calculating r_L, r_R as follows, we can obtain each frequency band simply.

$$\frac{i\cdot N^2}{4} = \frac{\pi \cdot r_i^2}{4} \longrightarrow r_i = \sqrt{\frac{i\cdot N^2}{\pi}}, \quad i = 0,1,2,3, \tag{11}$$

$$r_4 = \sqrt{2}\cdot N, \tag{12}$$

```
    Input:
        $X$ Original samples,
        $X_{\text{adv}}$ Adversarial samples
        $N$ The number of samples
        $m$ Selected high-level representation layer
        $M(\cdot)$ Model trained by $X$
    Output:
        $D(\cdot)$ Detector
 1  For $i$ in $0, 1 \ldots N-1$ do
 2  # the $m$th feature map
 3  XL $\longleftarrow M_m(X^i)$
 4  XadvL $\longleftarrow M_m(X^i_{\text{adv}})$
 5  XLH $\longleftarrow$ getHighFrequencyComponent(XL)
 6  XadvLH $\longleftarrow$ getHighFrequencyComponent(XL)
 7  XLH_list.append(XLH.flatten())
 8  XadvLH_list.append(XadvLH.flatten())
 9  End
10  Trained $D(\cdot)$ with XLH_list, XadvLH_list
11  Return $D(\cdot)$
```

Algorithm 1: HLFD adversarial detection algorithm

where r_i is the boundary value for quartering the matrix X_e. Hence, low-frequency component $X_e^{\text{Low}}(i, j)$ can be obtained via calculating $\varphi(X_e; 0, \sqrt{N^2/\pi})$ where $r_L = r_0 = 0$, $r_R = r_1 = \sqrt{N^2/\pi}$. Learning by analogy, medium-low, medium-high, and high-frequency can be obtained by computing $\varphi(X_e; r_1, r_2)$, $\varphi(X_e; r_2, r_3)$, and $\varphi(X_e; r_3, r_4)$, respectively.

3.4. HLFD Detection Method. As shown in Figure 4, we divide the HLFD detection method into three parts: extracting high-level representation, extracting high-frequency components, and training process. Inputting the normal and abnormal samples X, X_{adv}, we can obtain the mth feature map of model M via calculating $M_m(X)$, $M_m(X_{\text{adv}})$. According to the experimental result in section 4.2, further converting *Mm(X),Mm(Xadv)* to spectral domain makes the model have a greater improvement in detection tasks. Thus, we employ the Fourier transform to acquire the spectral characteristics $F(M_m(X))$, $F(M_m(X_{\text{adv}}))$ and further obtain high-frequency components $F_H(M_m(X))$, $F_H(M_m(X_{\text{adv}}))$ by equation (11). As emphasized in the previous paragraphs, feature engineering is the key to our HLFD method. Whether the detector is logistic regression [37], support vector machine [38], or neural network model, we can obtain a better detection performance as long as using $F_H(M_m(X))$, $F_H(M_m(X_{\text{adv}}))$ as input. Both Figures 5 and 6 illustrated this conclusion well by intuition and experiment, respectively. More specific procedures can be acquired in Algorithm 1.

4. Experiment

In this section, we will rigorously conduct experiments to demonstrate the effectiveness of our detection method. We initiate with a basic experimental setup and explore the discrepancy between the spatial and spectral domains. To improve the detection performance, it is indispensable to further explore the impact of the representations of different layers, different frequency bands, and different detectors on a detection task. At last, we will conduct an ablation study and compare our method with the existing state-of-the-art methods.

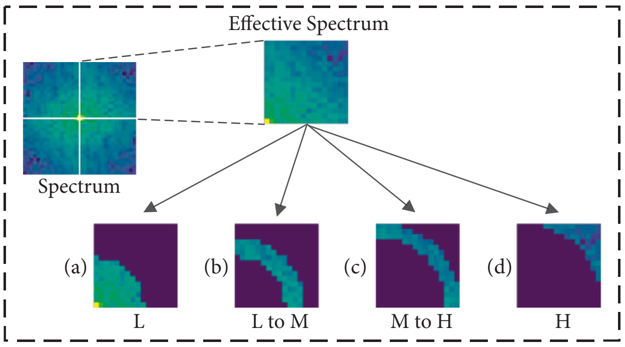

Figure 3: A schematic diagram of frequency band division. (a), (b), (c), and (d) represent the low, medium-low, medium-high, and high-frequency bands, respectively.

4.1. Experimental Setup. For a more generalized conclusion, we conduct a series of experiments on five datasets (MNIST, CIFAR-10, CIFAR-100, SVHN, and T-ImageNet) and six attack methods (FGSM, BIM, PGD, JSMA, CW, and DeepFool). We employ open source pretrained models for convenience, which achieve 98.4%, 93.7%, 74.2%, 96.0%, and 51.2% accuracy on MNIST, CIFAR-10, CIFAR-100, SVHN, and T-ImageNet, respectively. To obtain corresponding adversarial examples, we only select samples that can attack the pretrained models successfully. An average of 10,000 adversarial examples are generated for each dataset, which means we can obtain 20,000 samples after adding the

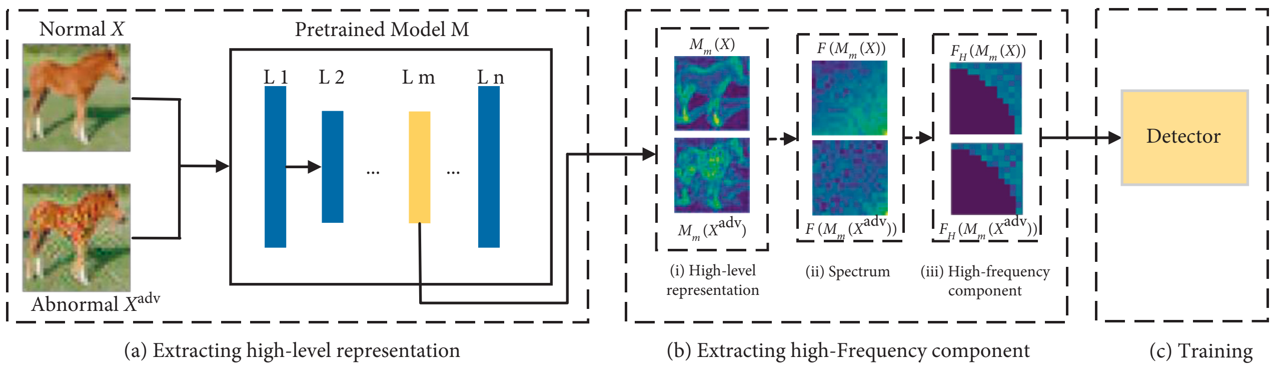

FIGURE 4: The whole framework of our HLFD method. It is divided into three parts: extracting high-level representation (a), extracting high-frequency component (b), and training process (c). (a) and (b) can be considered as data preprocessing. The detector will be trained on these transformed data.

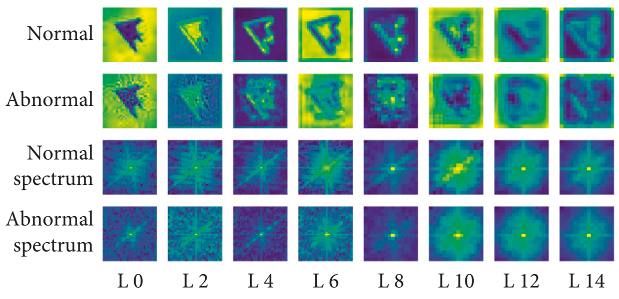

FIGURE 5: Visualizing the representations of different layers and its corresponding spectrum. The horizontal and vertical axes represent the feature maps of different layers and the corresponding spectrum, respectively. The pretrained model is trained on CIFAR-10 and adversarial examples are generated by FGSM.

number of normal samples. We split them into training set (64%), validation set (16%), and test set (20%), and apply the detection rate ACC (accuracy) and AUC (area under curve) as the evaluation metrics. All adversarial examples are generated by a python tool Foolbox [25]. For a fair comparison, each pixel is changed by an average of 10%. For MNIST, $L_2 = (28 * 28 * 1 * 0.1^2)^{1/2} = 2.8$. For SVHN, CIFAR-10, CIFAR-100, $L_2 = (32 * 32 * 3 * 0.1^2)^{1/2} = 5.5$. Similarly, $L_2 = 22$ for T-ImageNet. The L_2 norm is used to limit the size of the perturbation in subsequent experiments.

4.2. Spatial Vs. Spectral Domain. The spatial and the spectral domain refer to the original samples and the original samples after Fourier Transform, respectively. To intuitively observe the discrepancy between the spatial and spectral domains, we make a visual diagram shown in Figure 5. It seems that the pixel distribution of adversarial examples is discontinuous in the spatial domain, which is caused by random perturbation. Although humans can recognize the difference between normal and abnormal samples, the machine is hard to learn the pattern since the distribution is not generalized and stable. On the contrary, adversarial examples in the spectral domain have many small black dots in the mid-high frequency region, which express fixed and generalized patterns. The pattern may be effective for training detectors. To obtain a more rigorous conclusion, we further perform cluster analysis in the spatial and spectral domains as shown in Figure 6. In the first column, it seems that normal and abnormal samples cannot be separated by clustering whether in the spatial or the spectral domain. Still, it is not hard to discover that normal and abnormal samples are gradually classified as deepening of the network layer. Despite illustrating the effectiveness of high-level representation, we find that data in the spectral domain can be linearly separated, which cannot implement in the spatial domain. The phenomenon demonstrates the effectiveness of the spectral domain in a sense. To further explore the performance of spatial domain data on detection tasks, we conduct a series of experiments on CIFAR-10. As shown in Table 1, although spatial data are effective for FGSM, PGD, BIM, and JSMA attacks, they are powerless in CW and DeepFool attacks. It is possible that the perturbations generated by these two attack methods are little and just cross the decision boundary which are harsh to detect in the spatial domain.

4.3. Influence of High-Level Representation. The above experiments reveal the effectiveness of high-level representation and high-frequency extraction qualitatively. Yet, it is not clear how high-level representation affects the detection task. For this, we further conduct experiments to explore the impact of representations of different layers. As shown in Figure 7, the detection rate gradually increases as the deepening of the network layer, although decreases occasionally. Concentrating on CW and DeepFool, which are harsh to detect in the spatial domain, we can also detect them effectively after high-level representation. Nonetheless, we are still not sure which layer works best for the detector. For insurance, we support extracting the last two or three layers for aggregation. An intuitive understanding of why high-level representations work is that these features are incomprehensible (i.e., nonrobust) to humans and are extracted as the deepening of network layers. However, both robust and nonrobust features are crucial for model training, as Ilyas et al. illustrated in [16]. This is exactly the reason why clustering analysis can separate them farther and farther as deepening of network layers, as shown in Figure 6.

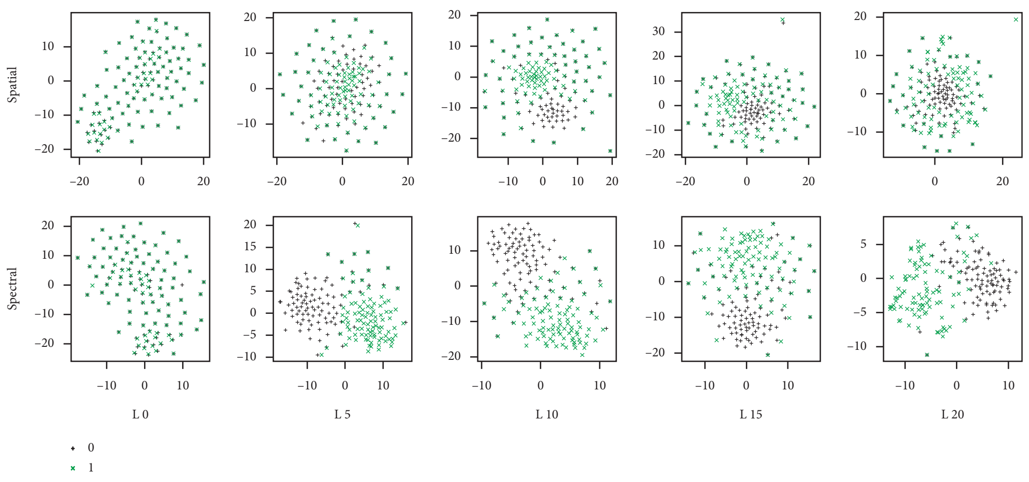

FIGURE 6: Performing cluster analysis on normal and abnormal samples using t-SNE [39]. The horizontal axis denotes the feature maps of different layers for clustering and the vertical axis represents clustering in different number fields: the spatial and the spectral domain. The pretrained model is trained on CIFAR-10 and adversarial examples are generated by CW.

TABLE 1: Detection performance of AUC (in %) on CIFAR-10 using original samples. Detector is trained and evaluated on the specified adversarial examples. For example, 92.6 below represents that the detector is trained on BIM, whereas evaluated on adversarial examples generated by FGSM.

Dataset	Detector	FGSM	BIM	PGD	JSMA	CW	DF
CIFAR-10	FGSM	99.1	65.5	63.7	55.6	48.9	50.4
	BIM	92.6	91.6	91.7	73.5	52.2	54.7
	PGD	97.5	98.3	98.5	82.3	50.2	50.6
	JSMA	81.5	75.6	80.4	88.5	48.9	51.3
	CW	51.8	49.8	50.9	50.6	49.5	52.6
	DF	48.9	51.7	50.0	49.3	51.7	53.4

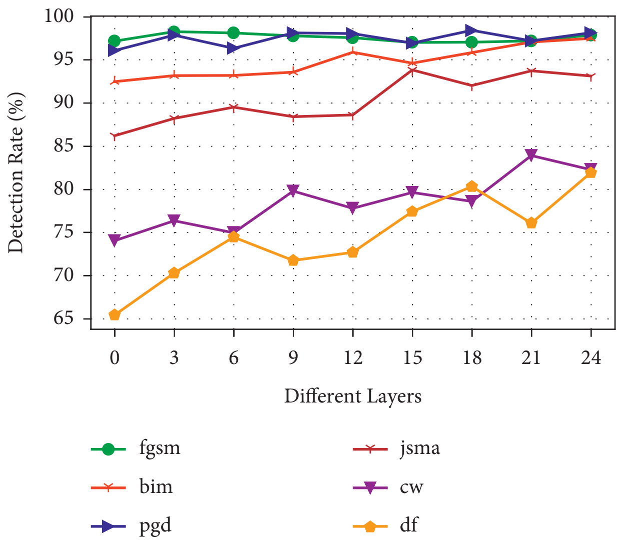

FIGURE 7: Impact of the representations of different layers on the detection rate AUC (in %). The detector is trained on normal samples and adversarial samples generated by FGSM.

4.4. Influence of High-Frequency. Wang et al. illustrated in [9] that high-frequency components can affect model perception and further proposed that high-frequency regions are correlated with semantic components of images. Inspired by this, we initiate to explore the impact of different frequency bands on the detection performance. As the experimental results shown in Tables 2 and 3, high-frequency regions can indeed promote the performance of the detector to a certain extent. However, which frequency bands are considered high is an issue. Although we obtain the highest detection rate from 3/4 to 4/4 frequency bands (high-frequency component) as shown in Table 2, 2/4 to 4/4 frequency bands (mid-high and high-frequency components) acquire highest detection rate as shown in Table 3. For insurance, we suggest frequency bands from 2/4 to 4/4 as the output of extracting high-frequency. The high-frequency components actually correspond to the part of the image that changes drastically and the perturbation is the same. The commonality makes the high-frequency components contain more perturbation detail, which is effective for detecting. The method for frequency band division can be found in equations (10) and (11).

4.5. Influence of Different Detectors. Although the input data is crucial, the choice of the detector model also has an impact on the detection results. We apply three classifier models: LR [37], SVM [38], and simple neural network for comparison. As shown in Figure 8, it is not hard to comprehend that since the detector is trained on CW, it works well for detecting the CW attack. However, the three classifiers are less regular in the detection performance. Therefore, we believe that it is inefficient to promote the detection performance significantly via altering the model structure. The result further confirms the rationality of concentrating on feature engineering.

4.6. Ablation Study. To explore the influence mechanism between the representation of different layers and the frequency bands, we perform an ablation study on them. As

Table 2: The detecting results of the different frequency band ACC/AUC (%) in FGSM, using CIFAR-10 and L_2 norm = 5.5.

From below to right	1/4	2/4	3/4	4/4
0/4	76.5/83.9	97.2/99.2	97.4/99.4	98.2/99.3
1/4	—	97.6/99.1	98.6/99.1	98.6/99.3
2/4	—	—	97.6/99.6	99.0/99.9
3/4	—	—	—	99.4/99.9

Table 3: The detecting results of the different frequency band ACC/AUC (%) in DeepFool, using CIFAR-10 and L_2 norm = 5.5.

From below to right	1/4	2/4	3/4	4/4
0/4	48.9/50.4	49.6/50.5	53.1/53.0	61.1/64.2
1/4	—	50.2/50.47	53.1/53.3	60.5/64.6
2/4	—	—	53.9/53.5	63.1/64.5
3/4	—	—	—	62.9/63.8

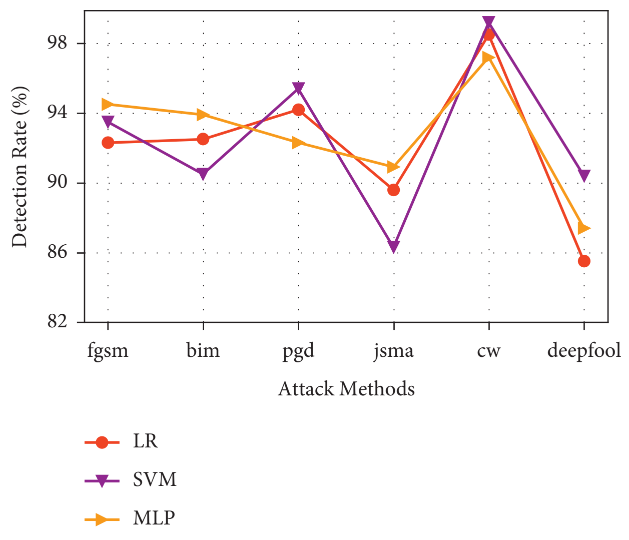

Figure 8: The impact of different detectors on the detection results AUC (in %). The detector is trained on CW and evaluated on six attacks using SVHN. The perturbation sets to L_2 = 5.5.

shown in Figure 9, the low-frequency of the original images (i.e., the left brown red pillar) are viewed as the benchmark of the detection rate. Two dimensions, the layer of the feature map and the interval of the frequency band, are taken into account in the experiment. For verifying the effectiveness of high-level representation and high-frequency extraction, we compared the benchmark results with the two dimensions. For a fair comparison, the L_2 norm of each image is controlled around 5.5 and the SVHN dataset is used here. From Figure 9, we can observe that the detection rates show an upward trend whether only considering different layers of the network or frequency bands. The result shows that there is a win-win effect between the representations of different layers and frequency bands, which further confirms the effectiveness of our HLFD method. It is also found that an 83% detection rate can be achieved even on DeepFool, which is harsh to detect in the spatial domain.

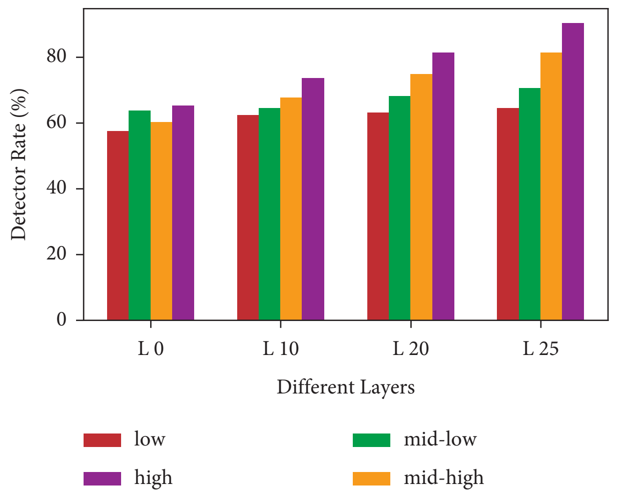

Figure 9: Ablation studies on the representations of different layers and frequency bands. The colous represent different frequency bands. The detector is trained on normal samples and adversarial samples generated by CW. The detection rate AUC (in %) is evaluated by adversarial samples generated by DeepFool.

Table 4: Comparison of AUC (%) under various evaluation setups. Our method HLFD takes the last two layers of representation and the mid-high and high-frequency regions as input. The L_2 norm of perturbation of MNIST is 2.8, the L_2 norm of T-ImageNet is 22, and the other three datasets are all L_2 = 5.5.

Dataset	Detector	Detection of six attack methods					
		FGSM	BIM	PGD	JSMA	CW	DF
MNIST	KD + PU	90.4	92.5	88.9	85.3	89.8	83.1
	LID	92.8	87.9	85.2	89.5	85.3	90.4
	M-D	97.6	99.1	98.5	92.7	88.6	84.3
	HLFD (ours)	99.8	98.5	99.1	89.4	95.4	92.8
SVHN	KD + PU	85.4	80.5	85.4	75.6	76.5	86.3
	LID	95.8	75.6	86.4	85.2	84.7	88.9
	M-D	99.4	96.3	94.3	87.6	82.5	92.5
	HLFD (ours)	99.5	99.4	97.4	95.6	93.4	89.5
CIFAR-10	KD + PU	83.4	95.2	94.5	82.4	65.4	72.5
	LID	94.2	93.5	93.8	85.4	80.5	84.3
	M-D	97.5	98.1	98.6	90.7	83.2	87.5
	HLFD (ours)	99.5	98.6	98.8	94.5	95.6	92.6
CIFAR-100	KD + PU	92.3	89.5	90.1	84.5	65.4	68.4
	LID	98.5	95.4	96.7	82.6	70.5	75.6
	M-D	99.2	97.1	96.4	89.3	78.9	82.9
	HLFD (ours)	99.7	99.4	97.5	96.4	86.4	85.3
T-ImageNet	KD + PU	85.4	90.5	88.2	73.6	62.5	69.8
	LID	86.2	88.4	89.3	77.2	65.6	64.2
	M-D	92.5	87.7	85.4	72.2	75.4	79.5
	HLFD (ours)	94.3	91.5	86.4	88.6	80.4	78.3

4.7. Comparison with Existing Methods. We compare our method with three state-of-the-art detection methods (KD + PU, LID, and M-D). To obtain a more objective conclusion, we conduct a series of experiments on five datasets and evaluate on six attacks. For a fair comparison, we set the same perturbation for each attack. As shown in Table 4, our method outperforms the other three detection methods for each attack on CIFAR-10 and CIFAR-100. To make the results more convincing, we use more realistic datasets (T-ImageNet) for testing. Although our detection rate is not the highest in some scenarios, the gap is not large at least. Moreover, we improve the detection rates by a large margin on DeepFool and CW attacks. Overall, our HLFD method is more robust and stable in various real-world environments compared to the existing state-of-the-art methods.

5. Conclusion

In this paper, we propose a simple yet effective HLFD method for detecting adversarial examples. By exploring from the spatial to the spectral domain, it is found that adversarial examples after transforming to the spectrum have richer characteristics which are beneficial for training the detector. Moreover, we further discover that extracting high-level representations and high-frequency components can promote the detection performance and the two factors show a win-win relationship via the ablation study. We intuitively and experimentally explain why these two factors work. Exploiting these findings, HLFD detection method is proposed. Although our method outperforms other state-of-the-art adversarial detection methods in most scenarios, the detectors are still faced with a more complex and unknown attacks in a real-world environment. Extending our method to more realistic settings (e.g., ImageNet dataset) is crucial. Exploring how to detect more aggressive attacks effectively are also a worthwhile research subject.

Data Availability

The public datasets can be downloaded from https://paperswithcode.com/dataset/mnist, https://www.cs.toronto.edu/̃kriz/cifar.html and https://paperswithcode.com/dataset/svhn. The pretrained model can be downloaded from https://github.com/aaron-xichen/pytorch-playground.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No. 2021YFB2700600) and the National Natural Science Foundation of China Enterprise Innovation and Development Joint Fund (No. U19B2044).

References

[1] I. Kaastra and M. Boyd, "Designing a neural network for forecasting financial and economic time series," *Neurocomputing*, vol. 10, no. 3, pp. 215–236, 1996.

[2] M. Chen, X. Shi, Y. Zhang, Di Wu, and M. Guizani, "Deep feature learning for medical image analysis with convolutional autoencoder neural network," *IEEE Transactions on Big Data*, vol. 7, no. 4, pp. 750–758, 2021.

[3] P. D. Heermann and N. Khazenie, "Classification of multispectral remote sensing data using a back-propagation neural network," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 30, no. 1, pp. 81–88, 1992.

[4] J. E. Dayhoff and J. M. DeLeo, "Artificial neural networks," *Cancer*, vol. 91, no. S8, pp. 1615–1635, 2001.

[5] V. Buhrmester, D. Münch, and M. Arens, "Analysis of explainers of black box deep neural networks for computer vision: a survey," *Machine Learning and Knowledge Extraction*, vol. 3, no. 4, pp. 966–989, 2021.

[6] T. Byun, S. Rayadurgam, and M. Heimdahl, "Black-box testing of deep neural networks," in *Proceedings of the 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, pp. 309–320, Wuhan, China, October 2021.

[7] I. J. Goodfellow, J. Shlens, and S. Christian, "Explaining and Harnessing Adversarial Examples," 2014, https://arxiv.org/abs/1412.6572.

[8] C. Szegedy, W. Zaremba, I. Sutskever et al., "Intriguing properties of neural networks," 2013, https://arxiv.org/abs/1312.6199?context=cs.

[9] H. Wang, X. Wu, Z. Huang, and P. Eric, "High-frequency component helps explain the generalization of convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8684–8694, Seattle, WA, USA, June 2020.

[10] A. A. Abello, R. Hirata, and Z. Wang, "Dissecting the high-frequency bias in convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 863–871, Nashville, TN, USA, June 2021.

[11] P. Harder, F.-J. Pfreundt, M. Keuper, and J. Keuper, "Spectraldefense: detecting adversarial attacks on cnns in the fourier domain," in *Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, Shenzhen, China, July 2021.

[12] D. Yin, R. Gontijo Lopes, J. Shlens, E. Dogus Cubuk, and J. Gilmer, "A fourier perspective on model robustness in computer vision," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[13] M. Mozes, P. Stenetorp, K. Bennett, and L. D. Griffin, "Frequency-guided word substitutions for detecting textual adversarial examples," 2020, https://arxiv.org/abs/2004.05887.

[14] S. Baluja and I. Fischer, "Adversarial transformation networks: learning to generate adversarial examples," 2017, https://arxiv.org/abs/1703.09387.

[15] T. Brunner, F. Diehl, M. T. Le, and A. Knoll, "Guessing smart: biased sampling for efficient black-box adversarial attacks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4958–4966, Seoul, Republic of Korea, November 2019.

[16] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[17] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial Intelligence Safety and Security*, pp. 99–112, Chapman and Hall/CRC, London, UK, 2018.

[18] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” 2017, https://arxiv.org/abs/1706.06083.

[19] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. Berkay Celik, and A. Swami, “The limitations of deep learning in adversarial settings,” in *Proceedings of the 2016 IEEE European symposium on security and privacy (EuroS&P)*, pp. 372–387, IEEE, Saarbruecken, Germany, March 2016.

[20] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in *Proceedings of the 2017 ieee symposium on security and privacy (sp)*, pp. 39–57, IEEE, San Jose, CA, USA, May 2017.

[21] S. M. Moosavi-Dezfooli, A. Fawzi, and F. Pascal, “Deepfool: a simple and accurate method to fool deep neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, Las Vegas, NV, USA, June 2016.

[22] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, “Detecting adversarial samples from artifacts,” 2017, https://arxiv.org/abs/1703.00410.

[23] X. Ma, B. Li, Y. Wang et al., “Characterizing adversarial subspaces using local intrinsic dimensionality,” in *Proceedings of the International Conference on Learning Representations*, Vancouver, Canada, May 2018.

[24] K. Lee, K. Lee, H. Lee, and J. Shin, “A simple unified framework for detecting out-of-distribution samples and adversarial attacks,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[25] J. Rauber, W. Brendel, and en Matthias Bethge, “Foolbox v0.8.0: a python toolbox to benchmark the robustness of machine learning models,” 2017, https://arxiv.org/abs/1707.04131.

[26] T. Miyato, A. M. Dai, and I. Goodfellow, “Adversarial training methods for semi-supervised text classification,” *Stat*, vol. 1050, p. 7, 2016.

[27] M. Andriushchenko and N. Flammarion, “Understanding and improving fast adversarial training,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 16048–16059, 2020.

[28] N. Papernot, P. McDaniel, Xi Wu, S. Jha, and A. Swami, “Distillation as a defense to adversarial perturbations against deep neural networks,” in *Proceedings of the 2016 IEEE symposium on security and privacy (SP)*, pp. 582–597, IEEE, San Jose, CA, USA, May 2016.

[29] N. Das, M. Shanbhogue, S.-T. Chen et al., “Keeping the bad guys out: protecting and vaccinating deep learning with jpeg compression,” 2017, https://arxiv.org/abs/1705.02900.

[30] J. Lu, T. Issaranon, and D. Forsyth, “Safetynet: detecting and rejecting adversarial examples robustly,” in *Proceedings of the IEEE international conference on computer vision*, pp. 446–454, Venice, Italy, October 2017.

[31] W. Xu, D. Evans, and Y. Qi, “Feature squeezing: detecting adversarial examples in deep neural networks,” 2017, https://arxiv.org/abs/1704.01155?s_tact=C3970CMW.

[32] K. Roth, Y. Kilcher, and T. Hofmann, “The odds are odd: a statistical test for detecting adversarial examples,” in *Proceedings of theInternational Conference on Machine Learning*, pp. 5498–5507, PMLR, Long Beach, CA, USA, June 2019.

[33] S. Tian, G. Yang, and Y. Cai, “Detecting adversarial examples through image transformation,” in *Proceedings of the Thirty-second AAAI conference on artificial intelligence*, New Orleans, LA, USA, February 2018.

[34] D. Hendrycks and K. Gimpel, “A baseline for detecting misclassified and out-of-distribution examples in neural networks,” in *Proceedings of the 5th International Conference on Learning Representations, ICLR 2017*, Toulon, France, April 2017.

[35] C. Ma, B. Wu, S. Xu et al., “Effective and robust detection of adversarial examples via benford-fourier coefficients,” 2020, https://arxiv.org/abs/2005.05552.

[36] J. W. Cooley and J. W. Tukey, “An algorithm for the machine calculation of complex Fourier series,” *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[37] R. J. CarrollCarroll and S. Pederson, “On robustness in the logistic regression model,” *Journal of the Royal Statistical Society: Series B*, vol. 55, no. 3, pp. 693–706, 1993.

[38] W. S. Noble, “What is a support vector machine?” *Nature Biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.

[39] L. Van der Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of Machine Learning Research*, vol. 9, p. 11, 2008.