DETECTING ADVERSARIAL EXAMPLES BY ADDITIONAL EVIDENCE FROM NOISE DOMAIN

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep neural networks are widely adopted powerful tools for perceptual tasks. However, recent research indicated that they are easily fooled by adversarial examples, which are produced by adding imperceptible adversarial perturbations to clean examples. In this paper, we utilize the steganalysis rich model (SRM) to generate noise feature maps, and combine them with RGB images to discover the difference between adversarial examples and clean examples. In particular, we propose a two-stream pseudo-siamese network and train it end-to-end to detect adversarial examples. Our approach fuses the subtle difference in RGB images with the noise inconsistency in noise features. The proposed method has strong detection capability and transferability, and can be combined with any classifier without modifying its architecture or training procedure. Our extensive empirical experiments show that, compared with the state-of-the-art detection methods, the proposed method achieves excellent performance in distinguishing adversarial samples generated by popular attack methods on different real datasets. Moreover, our method has good generalization, it trained by a specific adversary can generalize to other adversaries.

1 INTRODUCTION

Deep neural networks have achieved superior performance on many perceptual tasks, such as face recognition (Jiang et al., 2020), object detection (Li et al., 2019) and image classification (Hu et al., 2018). However, there is obvious difference between the perception systems of humans and neural networks. Szegedy et al. (2014) have demonstrated that adversarial examples generated by adding tiny but elaborately designed perturbations can easily fool neural networks with high confidence. For images, the perturbations are imperceptible and do not stir any doubt about the correct classification for humans. Many different methods (Goodfellow et al., 2015; Kurakin et al., 2017b; Madry et al., 2018; Papernot et al., 2016a; Su et al., 2019; Carlini & Wagner, 2017; Dong et al., 2018; Uesato et al., 2018) have been proposed to design the worst-case perturbations. Most strikingly, adversarial examples have transferability, e.g., an adversarial example generated by a model can remain attack effective for other models. This makes adversaries can successfully attack a model without knowing its details, thereby reducing the difficulty of implementing attacks.

The undesirable property of deep neural networks has become major problem in safety-critical applications like medicine, finance and autonomous driving. Methods to increase the robustness of neural networks against adversarial examples have been proposed from augmenting the training data (Huang et al., 2016; Kurakin et al., 2017a; Xie et al., 2019; Song et al., 2020) with adversarial examples to distilling robust networks from the original networks (Papernot et al., 2016b). Unfortunately, no matter how robust a model is, there are always new attacks that can successfully fool it. When a trained model is being applied, the cost is huge of retraining it to deal with new attacks. Therefore, convenient and flexible methods to defend against adversarial examples are essential.

Detection only methods are flexible, and can provide protection to a model even if the model is being used. KD+BU (Feinman et al., 2017), LID (Ma et al., 2018) and ML-LOO (Yang et al., 2020) utilize the distribution character of different categories in hidden layers of the targeted model to detect adversarial examples. Metzen et al. (2017) grafted a detection subnetwork on the targeted model. Although these methods show compelling performance results on a number of state-of-the-art adversarial attacks, one major drawback is that these methods depend closely on the protected

network, which will restrict their application. These methods will fail when they are used to provide protection to a task that uses the online machine learning service providers such as Amazon Machine Learning and BigML. If a detector can detect adversarial examples effectively only by input samples, this problem then is no longer a problem. However, an adversarial image is very similar to its corresponding original image, hence it is extremely difficult to effectively discriminate the adversarial of an image without relying on the targeted model.

In this paper, we treat adversarial perturbations as special noise, and extract noise features to provide additional evidence for adversarial example detection. The noise features supply rich information for discriminating adversarial examples, making the proposed model can effectively detect adversarial examples without the targeted model. Our model consists of a two-stream pseudo-siamese network and a decision network. The first stream discovers clues to the subtle difference like contrast difference, unnatural pixels from the RGB features. The second stream is a noise stream which utilizes the noise features to capture the noise inconsistency between clean samples and adversarial samples. The intuition behind the second stream is that although adversarial perturbations are pretty special, they are still noise, the noise features between original images and adversarial images are unlikely to match. To utilize the noise features, we need to choose a suitable tool to convert the RGB image into the noise domain. We observe that the total variation of adversarial images is obviously larger than that of original images, which shows that the value difference between adjacent pixels is larger in adversarial images. We thus select steganalysis rich model (SRM) (Fridrich & Kodovsky, 2012; Zhou et al., 2018) to generate the noise features. SRM extracts local noise features from adjacent pixels, and amplifies local pixel difference in adversarial images. The noise feature maps are directly used as input to the second stream. We then adopt bilinear pooling (Lin et al., 2015; Fukui et al., 2016) to combine the features produced from the two streams. Bilinear pooling is often used for fine-grained classification, it can fuse two streams while preserving spatial information. Finally, the decision network uses the fused features to discriminate adversarial examples.

We summarize our contributions as follows:

1. We propose a novel two-stream adversarial example detection framework and perform end-to-end training. The proposed method can obtain rich feature information from noise features to provide additional evidence for adversarial example detection. With the rich feature information, our model gets rid of the dependence on the targeted model. Compared with the state-of-the-art detection methods, our method has good transferability while maintaining effective detection capability, and it can be reused to protect different models after once training.
2. We select the steganalysis rich model (SRM) to produce noise feature maps. We notice that the total variation of adversarial images is significantly larger than that of clean images. This means the difference in the value of neighboring pixels is larger in adversarial images. SRM amplifies the difference in noise domain, and obtains additional plentiful information to assist in detecting adversarial samples.
3. Extensive experiments show that our method achieves excellent performance in defending against both white-box attacks and black-box attacks. Moreover, our method has good generalization, it trained by an attack can defend against other attacks effectively.

2 Related Work

In this section, we briefly review the related work in steganalysis rich model, adversarial attack and adversarial defense.

Steganalysis Rich Model: Steganalysis rich model (SRM) is mainly used in image forensics tasks. It extracts local noise features from adjacent pixels to capture the inconsistency between authentic and tampered regions. Cozzolino et al. (2015; 2017) demonstrated the performance of SRM in distinguishing tampered regions from authentic regions, and combined SRM features with Convolutional Neural Networks to perform manipulation localization. Rao & Ni (2016) used a SRM filter kernel as the initialization of a Convolutional Neural Network to improve detection accuracy. Zhou et al. (2018) utilized SRM filter kernels to extract low-level noise used as input to a Faster R-CNN network, and captured tampering traces in the noise features.

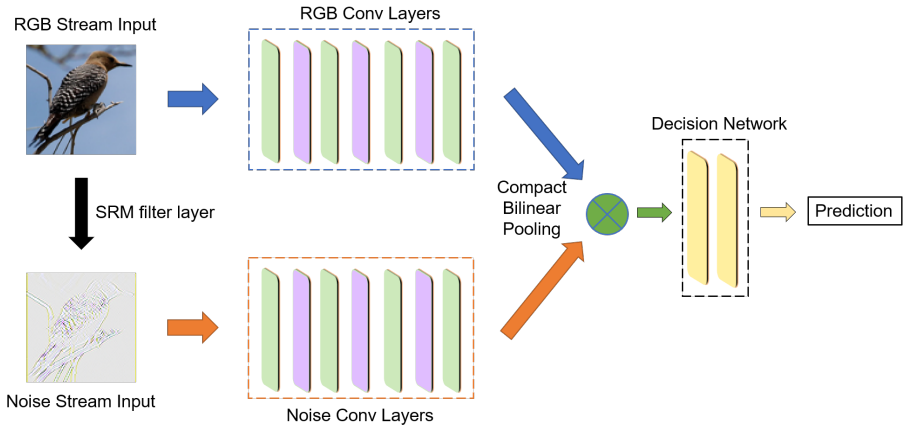

Figure 1: Illustration of our two-stream pseudo-siamese network. Color code used: light green = Conv+ReLU, purple = max pooling, green = bilinear pooling, yellow = fully connected layers. The RGB stream uses original images as input, and captures subtle difference like contrast difference, unnatural pixels from the RGB features. The noise stream first obtains noise feature maps through a SRM filter layer, and leverages the noise features to provide additional evidence for adversarial image detection. Bilinear pooling combines the spatial co-occurrence features extracted by the two streams. Finally, passing the combined features through a decision network composed of fully connected layers and a softmax layer, the network generates the predicted label and determines whether the input image is adversarial or not.

Adversarial Attack: Since Szegedy et al. (2014) first noticed the existence of adversarial examples, many attack methods have been proposed. The fast gradient sign method (FGSM) (Goodfellow et al., 2015) is a pioneering attack method, it performs a single-step gradient update along the direction of the sign of gradient at each pixel. Although FGSM is simple, it is a very effective attack. The basic iterative method (BIM) (Kurakin et al., 2017b) extends FGSM into an iterative algorithm. It replaces the single-step with multiple small steps. The projected gradient descent (PGD) (Madry et al., 2018) is similar to BIM, and chooses a randomly perturbed image of an original image as the initial image to generate the adversarial image. The Momentum Iterative Method (MIM) (Dong et al., 2018) integrates the momentum term into the iterative process for attacks to generate more transferable adversarial examples. Papernot et al. (2016a) proposed the Jacobian-based saliency map attack (JSMA) that modifies only a few pixels in an image to generate correspongding adversarial image. The one pixel attack (Su et al., 2019) only changes one pixel in an image to fool the targeted classifier. Uesato et al. (2018) used the finite difference in the random direction to estimate approximate gradients, thereby generating adversarial samples when the gradients cannot be used. Moosavi-Dezfooli et al. (2016) proposed Deepfool to find the shortest distance from the clean images to the decision boundary of the adversarial images. Carlini & Wagner (2017) converted adversarial examples into the argtanh space, making it more flexible to use the optimization solvers.

Adversarial Defense: Due to the huge potential hazard of adversarial samples, a number of defense methods have been proposed to defend against adversarial examples, including adversarial training (Huang et al., 2016; Kurakin et al., 2017a; Xie et al., 2019; Song et al., 2020), data compression (Guo et al., 2018; Bhagoji et al., 2018), defensive distillation (Papernot et al., 2016b), input reconstruction (Gu & Rigazio, 2015; Liao et al., 2018; Jia et al., 2019), feature squeezing (Xu et al., 2018), verifiable defense (Wong & Kolter, 2018) and randomized model (Lecuyer et al., 2019; Liu et al., 2018; 2019). These defense methods often involve modifications in the model training process, which usually require higher computational or example complexity, and lead to loss of accuracy (Yang et al., 2020). Complimentary to the previous defense methods, an alternative line of works focus on screening out adversarial samples in the testing phase without touching the training of the targeted model or preprocessing input samples. Metzen et al. (2017) grafted a discriminator on the targeted model, and used the output of the intermediate layers as input to the discriminator to detect adversarial examples. Feinman et al. (2017), Ma et al. (2018) and Yang et al. (2020) utilized the distribution character in hidden-layer output of different categories to identify adversarial examples. Although these defenses

$$\frac{1}{4}\begin{bmatrix}0 & 0 & 0 & 0 & 0\\ 0 & -1 & 2 & -1 & 0\\ 0 & 2 & -4 & 2 & 0\\ 0 & -1 & 2 & -1 & 0\\ 0 & 0 & 0 & 0 & 0\end{bmatrix}\quad \frac{1}{12}\begin{bmatrix}-1 & 2 & -2 & 2 & -1\\ 2 & -6 & 8 & -6 & 2\\ -2 & 8 & -12 & 8 & -2\\ 2 & -6 & 8 & -6 & 2\\ -1 & 2 & -2 & 2 & -1\end{bmatrix}\quad \frac{1}{2}\begin{bmatrix}0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 1 & -2 & 1 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

Figure 2: The SRM filter kernels used in our work to extract local noise features. In grayscale images, we only use the left kernel to generate SRM images.

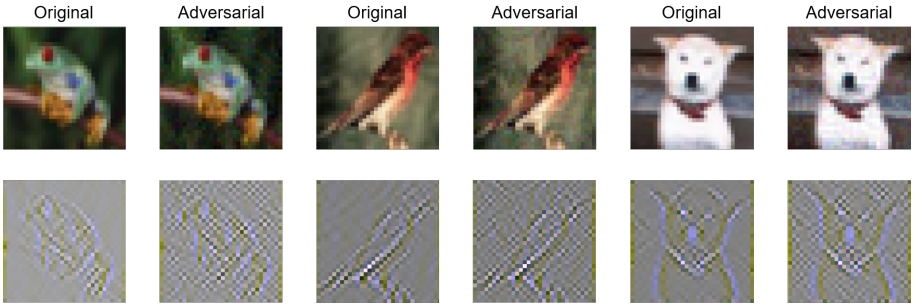

Figure 3: Examples of original, adversarial images and corresponding SRM images in CIFAR-10. The adversarial images are produced using PGD with maximum perturbation $\epsilon = 0.05$ (out of 1.0). Each column shows a RGB image and its corresponding SRM image. As shown in second row, the SRM images generated from adversarial images are significantly noisier than the SRM images generated from original images. A clean image usually consists of different smooth regions and complex texture regions. The adversarial perturbations will destroy the smooth regions and make the complex texture area more cluttered. The main focus of a RGB channel is on semantic image content, thus ignoring these difference.

do not modify the model and input samples, they rely closely on the targeted model. This will restrict the transferability of these methods. Our work focuses on presenting an effective defense that does not modify the model and input samples, while has good transferability and generalization.

3 Proposed Method

We adopt a two-stream pseudo-siamese network to detect adversarial images. As shown in Figure 1, the RGB stream uses RGB images as input and the noise stream uses SRM images as input. We utilize compact bilinear pooling to fuse the features produced by the two steams before a decision network. The SRM filter layer is used to produce SRM images. Without relying on the targeted model, our method can still effectively identify adversarial examples. And, our approach can be combined with different models repeatedly after once training.

3.1 Steganalysis Rich Model

When generating adversarial images, L_p norm, including L_0, L_2 and L_∞ norm, are usually used to restrict the change of pixel values. Hence an adversarial image and its corresponding original image are very similar. RGB channels are not sufficient to tackle all different information to detect adversarial examples. We notice that the total variation of adversarial images is significantly larger than that of original images. This is easy to understand, due to the addition of adversarial perturbations, the value difference of adjacent pixels in an image becomes larger. We use SRM filter kernels to extract local noise features from RGB images to amplify the difference, and provide additional evidence for adversarial example detection. This is novel–while a number of detection only methods have been proposed to improve the robustness of models, no prior work in defending against adversarial attacks has investigated learning from noise distribution in detection.

Steganalysis rich model generates the noise features of one image through the residual between a pixel value and the estimation of the pixel value generated by only interpolating the adjacent pixel value. SRM uses 30 basic filters to gather the basic noise features by performing nonlinear operations like maximum and minimum of the nearby output after filtering. By quantifying and truncating the output of these filters, SRM extracts the nearby co-occurrence information as final features. Zhou et al. (2018) demonstrated that using only 3 kernels can achieve similar performance as using 30 kernels. Therefore, we choose 3 kernels and their weights are shown in Figure 2. We copy each kernel twice to form three 5×5×3 convolution kernels as the parameters of SRM filter layer with 3-channel input and 3-channel output. For grayscale images, we only use the left kernel as the parameters of SRM filter layer with 1-channel input and 1-channel output.

Figure 3 shows the resulting noise feature maps after the SRM layer. We can clearly see that there is significant difference between original images and adversarial images in the SRM images although they are similar in the RGB images. Especially in smooth regions, SRM amplifies the insignificant difference between neighboring pixels.

3.2 Two-Stream Network

The SRM images are directly used as input to the noise stream, which has same architecture with the RGB stream (structure details as shown in Lable 2 in Appendix). Then, we adopt bilinear pooling (Lin et al., 2015) to fuse the noise stream and the RGB stream, and pass the fused results into the decision network. Bilinear pooling is proposed for fine-grained classification, it combines two CNN network streams while preserving spatial information. However, the bilinear features combined by bilinear pooling are high dimensional, typically on the order of hundreds of thousands to millions. To speed up training, we use compact bilinear pooling (Fukui et al., 2016) to fuse the two streams. The compact bilinear pooling has the same performance as the full bilinear representation but with only thousands dimensions.

After the fused features pass through a decision network consisting of two fully connected layers and a softmax layer, the final predicted result is obtained (see Figure 1). We use cross entropy loss and squared L_2-norm regularization that leads to the following objective function:

$$minimize\ L = \lambda||\omega||_2 + L_{cross}(f_D(CBP(f_{RGB}(x), f_N(f_{SRM}(x)))), y), \tag{1}$$

where λ is a hyperparameter for balancing the regularization and loss. x is the input image and y is x's label. $||\cdot||_2$ denotes the L_2 norm. ω are the weights of networks except for the SRM network. f_{SRM} denotes the SRM network with fixed weights. f_{RGB} and f_N are the RGB stream network and the noise stream network. CBP denotes the compact bilinear pooling. L_{cross} denotes the cross entropy loss.

4 Experiments

In this section, we present an experimental evaluation of our method, and compare it with several state-of-the-art detection methods.

4.1 Experimental Setting

Datasets: We extensively evaluate our method on MNIST, CIFAR-10 and CIFAR-100. MNIST is a gray scale image dataset of handwritten digits with the image shape 28×28, including 60000 training samples and 10000 testing samples. CIFAR-10 consists of 60000 32×32 color images in 10 classes, with 5000 training samples and 1000 testing samples per class. CIFAR-100 is just like CIFAR-10, except it has 100 classes, with 500 training images and 100 test images per class.

The adversarial datasets corresponding to MNIST are generated with maximum perturbation $\epsilon = 0.3$ (out of 1.0), and the adversarial datasets corresponding to CIFAR-10 and CIFAR-100 are generated with maximum perturbation $\epsilon = 0.05$ (out of 1.0).

Classifiers: For MNIST dataset, we design two classifiers, as shown in Label 3 in Appendix, a local model and a black model. All classifiers are used to generate adversarial images with different attack

methods, but only the local model is used to assist detection methods. That is, if the local model as the protected model, then defending against the adversarial examples generated by the local model is defending against white-box attacks, and defending against the adversarial examples generated by the black model is defending against black-box attacks. The classifiers for CIFAR-10 and CIFAR-100 are shown in Lable 3 and 4 in Appendix. We use three classifiers as black models for CIFAR-10 and CIFAR-100, respectively. All classifiers are trained by Adam optimizer (Kingma & Ba, 2015) ($\beta_1 = 0.9, \beta_2 = 0.999$) with the batch size of 128, learning rate of 0.001, and epochs of 50.

Baseline models: We compare our method with state-of-the-art detection methods including graft network (G-RGB) (the authors did not give their method a proper name, for convenience we named it graft network) (Metzen et al., 2017), two-stream graft network (G-RGB-N), KD+BU (Feinman et al., 2017) and single-stream network (RGB). G-RGB grafts a detection network on the targeted model, and uses the features in the hidden layer of the targeted model as input to the detection network. Due to the authors shown that choosing the output of the upper layer of the fully connected layers as input has good generalization. Hence, we choose the input of G-RGB and G-RGB-N by this way. G-RGB-N is G-RGB with an additional noise stream. Similar to our method, we use SRM to generate SRM images and pass them to the targeted model to obtain hidden-layer features. KD+BU detects adversarial samples by looking at Bayesian uncertainty estimates and performing density estimation in the subspace of the deep features learned by the targeted model. RGB is a single stream network with RGB images as input, it has the same architecture as the RGB stream of our method. For convenience, we use RGB-N to represent our method.

Attack methods: We consider five attack methods, FGSM (Goodfellow et al., 2015), MIM (Dong et al., 2018), PGD (Madry et al., 2018), SPSA (Uesato et al., 2018) and BIM (Kurakin et al., 2017b), to evaluate the discrimination power of different detection methods. For G-RGB and G-RGB-N, we select the MIM to assist their training. And for RGB and RGB-N, the BIM is selected to assist their training. This is not randomly selected. New attack methods are emerging one after another. The generalization of a detector is very important, so we only choose one attack method to assist the training of the detector each time. We find that G-RGB and G-RGB-N with MIM-assisted training and RGB and RGB-N with BIM-assisted training have the best generalization.

Evaluation metric: We use true positive rate (TPR) and Area Under the receiver operating characteristic Curve (AUC) as the evaluation metrics for performance comparison. TPR is the proportion of original images classified as original. There are four situations: TPR and AUC scores are both high, TPR and AUC scores are both low, TPR score is high and AUC score is low, TPR score is low and AUC score is high. In the first case, it indicates that the detection method can classify original samples and adversarial samples effectively. The second case shows the detection method is rather chaotic. The third case means that the detector cannot effectively identify adversarial samples. And the forth case means that the detector cannot effectively identify original samples.

Parameter setting: The four detectors that need training are trained by Adam ($\beta_1 = 0.5, \beta_2 = 0.999$) with the batch size of 128, learning rate of 0.0001, and epochs of 100. We set the value of $\lambda = 0.0005$ in Equation 1. Each batch consists of 64 clean images and their corresponding adversarial images.

4.2 Experimental Results

Table 1 shows the TPR and AUC scores of different detection methods on the three datasets with the local model as the targeted model. On the MNIST dataset, we can see that RGB and our method (RGB-N) have excellent effects in defending against both white-box attacks and black-box attacks. As we mention above, RGB and RGB-N are trained by using the adversarial examples generated by BIM based on the local model. Thus, we can see that they have good generalization, they can well defend against adversarial samples generated by other attack methods. Although G-RGB and G-RGB-N are not as good as RGB and RGB-N, their performance is also good. In addition, G-RGB-N is obviously better than G-RGB. Due to space limitation, a more intuitive display is shown in Figure 5 in Appendix.

On the CIFAR-10 dataset. In the face of white-box attacks, our method (RGB-N) is significantly superior to other methods. Although the AUC scores of RGB are close to that of RGB-N, the scores of TPR are significantly behind RGB-N. This means that RGB misjudges a part of original images as adversarial. Combining TPR and AUC scores, G-RGB-N performs second to our method and

Table 1: Performance of detection methods in defending against adversarial examples generated by different models and attack methods on different datasets.

Dataset	Model	Method	FGSM		MIM		PGD		SPSA		BIM	
			TPR	AUC	TPR	AUC	TPR	AUC	TPR	AUC	TPR	AUC
MNIST	Local	G-RGB	0.946	0.933	0.950	0.986	0.950	0.989	0.947	0.809	0.950	0.980
		G-RGB-N	0.954	0.955	0.958	0.991	0.958	0.993	0.955	0.852	0.958	0.985
		KD+BU	0.480	0.771	0.489	0.795	0.498	0.809	0.464	0.749	0.500	0.810
		RGB	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
		RGB-N	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Black	G-RGB	0.947	0.979	0.946	0.953	0.950	0.919	0.926	0.755	0.948	0.865
		G-RGB-N	0.956	0.985	0.955	0.962	0.957	0.929	0.929	0.803	0.956	0.866
		KD+BU	0.513	0.785	0.482	0.765	0.490	0.772	0.397	0.645	0.463	0.752
		RGB	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
		RGB-N	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
CIFAR-10	Local	G-RGB	0.940	0.670	0.904	0.961	0.903	0.956	0.979	0.621	0.902	0.962
		G-RGB-N	0.934	0.882	0.908	0.967	0.908	0.952	0.966	0.784	0.907	0.944
		KD+BU	0.301	0.599	0.562	0.721	0.617	0.761	0.660	0.521	0.656	0.798
		RGB	0.830	0.996	0.843	0.996	0.843	0.995	0.839	0.995	0.843	0.965
		RGB-N	**0.948**	**0.999**	**0.950**	**0.998**	**0.950**	**0.996**	**0.955**	**0.998**	**0.950**	**0.984**
	VGG16	G-RGB	**0.967**	0.477	**0.968**	0.500	**0.988**	0.527	**0.987**	0.493	**0.987**	0.527
		G-RGB-N	0.960	0.833	0.962	0.742	0.980	0.716	0.986	0.763	0.976	0.619
		KD+BU	0.700	0.493	0.308	0.513	0.320	0.524	0.708	0.500	0.323	0.524
		RGB	0.849	0.994	0.846	0.991	0.848	0.993	0.867	0.996	0.839	0.913
		RGB-N	0.960	**0.998**	0.956	**0.997**	0.960	**0.996**	0.962	**0.998**	0.953	**0.959**
	MobileNet	G-RGB	**0.977**	0.475	**0.981**	0.493	**0.989**	0.495	**0.989**	0.500	**0.988**	0.488
		G-RGB-N	0.965	0.850	0.969	0.766	0.983	0.710	0.983	0.771	0.977	0.613
		KD+BU	0.694	0.486	0.707	0.494	0.289	0.522	0.307	0.528	0.301	0.537
		RGB	0.853	0.995	0.849	0.993	0.849	0.993	0.859	0.995	0.852	0.935
		RGB-N	0.962	**0.997**	0.958	**0.992**	0.959	**0.994**	0.960	**0.999**	0.956	**0.972**
	ResNet50	G-RGB	**0.973**	0.450	**0.972**	0.500	**0.985**	0.529	**0.988**	0.509	**0.981**	0.516
		G-RGB-N	0.964	0.838	0.964	0.783	0.975	0.731	0.978	0.774	0.967	0.665
		KD+BU	0.703	0.504	0.690	0.497	0.325	0.520	0.721	0.491	0.328	0.531
		RGB	0.851	0.993	0.850	0.991	0.853	0.993	0.853	0.995	0.846	0.956
		RGB-N	0.959	**0.994**	0.960	**0.991**	0.959	**0.993**	0.961	**0.998**	0.959	**0.975**
CIFAR-100	ResNet101V2	G-RGB	0.879	0.777	0.873	0.919	0.873	0.960	0.906	0.637	0.873	0.954
		G-RGB-N	0.903	0.966	0.900	0.960	0.900	0.957	0.913	0.865	0.900	0.933
		KD+BU	0.586	0.597	0.717	0.862	0.870	0.958	0.600	0.569	0.881	0.965
		RGB	0.869	0.973	0.871	0.981	0.871	0.983	0.871	0.974	0.871	0.959
		RGB-N	**0.924**	**0.987**	**0.926**	**0.990**	**0.926**	**0.990**	**0.921**	**0.994**	**0.926**	**0.986**
	ResNet152	G-RGB	0.905	0.759	0.914	0.713	0.912	0.624	0.926	0.622	0.912	0.586
		G-RGB-N	**0.918**	0.973	**0.927**	0.915	**0.930**	0.801	0.933	0.881	0.923	0.670
		KD+BU	0.619	0.590	0.645	0.613	0.616	0.593	0.598	0.567	0.604	0.594
		RGB	0.879	0.975	0.884	0.974	0.877	0.975	0.889	0.973	0.873	0.922
		RGB-N	**0.918**	**0.987**	0.925	**0.986**	0.925	**0.988**	**0.934**	**0.996**	**0.924**	**0.984**
	DenseNet169	G-RGB	0.913	0.743	0.913	0.701	0.924	0.600	0.935	0.613	0.925	0.567
		G-RGB-N	0.924	0.967	0.924	0.917	0.928	0.801	**0.936**	0.874	0.925	0.676
		KD+BU	0.620	0.582	0.631	0.595	0.600	0.566	0.609	0.567	0.591	0.559
		RGB	0.883	0.975	0.885	0.974	0.904	0.977	0.885	0.976	0.895	0.928
		RGB-N	**0.926**	**0.992**	**0.929**	**0.985**	**0.937**	**0.989**	0.934	**0.996**	**0.935**	**0.983**
	DenseNet201	G-RGB	0.904	0.741	0.909	0.697	0.927	0.623	0.931	0.613	0.920	0.584
		G-RGB-N	0.921	0.966	0.921	0.920	0.929	0.819	**0.932**	0.870	0.922	0.682
		KD+BU	0.623	0.582	0.619	0.582	0.601	0.576	0.597	0.555	0.607	0.579
		RGB	0.875	0.974	0.878	0.973	0.886	0.972	0.891	0.976	0.870	0.932
		RGB-N	**0.924**	**0.993**	**0.923**	**0.988**	**0.933**	**0.989**	0.924	**0.995**	**0.925**	**0.981**

RGB ranks third. When defending against the adversarial examples generated by the black models, the performance of RGB-N is still outstanding. But we can see that the TPR scores of RGB-N are slightly lower than the TPR scores of G-RGB. This does not mean that G-RGB is better than RGB-N. The reason is obviously that G-RGB categorizes most of the testing images as original. RGB and G-RGB-N have their pros and cons. In general, RGB ranks second and G-RGB-N ranks third. In Figure 4, we select some ROC curves for display. For a complete version, please see Figure 6 in Appendix. In Figure 4, we can clearly see that, the performance of RGB-N is better than other detection methods in all cases. G-RGB and G-RGB-N perform well when defending against white-box attacks, but they are much less effective when defending against black-box attacks.

The evaluation results of detection methods on CIFAR-100 dataset are similar to those on CIFAR-10 dataset, and our method is still significantly superior to other methods. The difference is that G-RGB-N performs better than RGB. The ROC curves of different detection methods on CIFAR-100 dataset are shown in Figure 7 in Appendix. To sum up, our method shows excellent performance on all datasets, followed by G-RGB-N and RGB. It is worth noting that G-RGB-N is obviously better than G-RGB. This shows that the high-level features generated by SRM images through the targeted model can still provide useful information for adversarial example detection. For the three methods other than RGB and our method (RGB-N), they still require the knowledge of the targeted model during the testing phase. They cannot be used separately from the targeted model, which

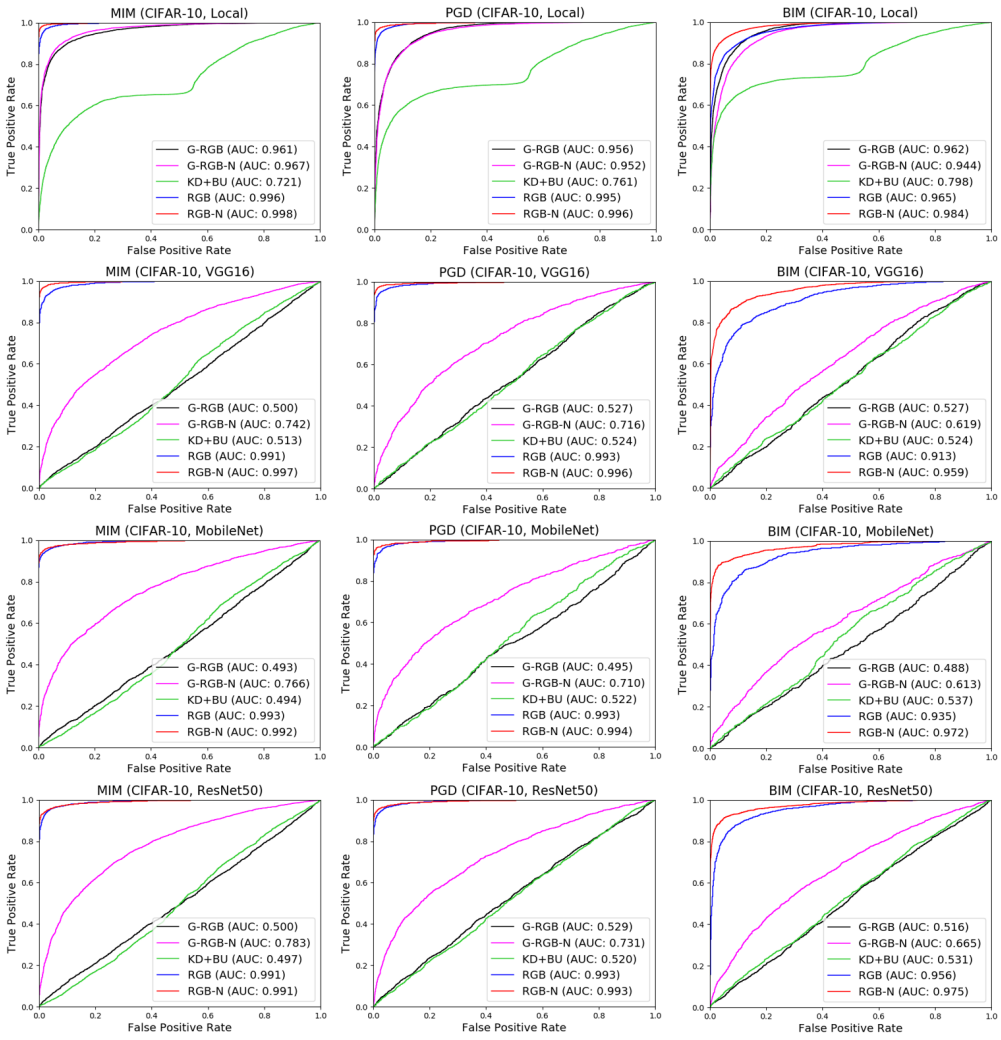

Figure 4: Some ROC curves of detection methods on CIFAR-10. We choose to display the ROC curves of detection methods on three different attacks. We can intuitively see that our method (RGB-N) is better than other methods in all cases.

limits their scope of application. Our model is independent of the targeted model in the testing phase. Therefore, there is no difference for our model whether to protect the local model or the black model in the testing phase.

5 Conclusion

In this paper, we propose a novel method using both a RGB stream and a noise stream to learn rich features for adversarial example detection. We extract the local noise features by SRM, which amplifies the inconsistency between original images and adversarial images. Rely on the additional evidence extracted by SRM, our method can be independent of the protected model. That is, our method has strong transferability, it can be reused to protect different model after once training. Before the decision network, we use bilinear pooling to fuse the two streams. The bilinear pooling can combine the two streams while preserving the spatial information. Experiments on standard datasets show that our method has excellent performance on both white-box and black-box attacks. Moreover, our method has good generalization, it trained by an attack method is fully capable of defending against other attack methods.

REFERENCES

Arjun Nitin Bhagoji, Daniel Cullina, Chawin Sitawarin, and Prateek Mittal. Enhancing robustness of machine learning systems via data transformations. In *CISS*, 2018.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *S&P*, 2017.

Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Splicebuster: A new blind image splicing detector. In *WIFS*, 2015.

Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Recasting residual-based local descriptors as convolutional neural networks: An application to image forgery detection. In *IH&MMSec*, 2017.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *CVPR*, 2018.

Reuben Feinman, Ryan Curtin, Saurabh Shintre, and Andrew Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv: 1703.00410*, 2017.

Jessica Fridrich and Jan Kodovsky. Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 2012.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *EMNLP*, 2016.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. In *ICLR*, 2015.

Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. In *ICLR*, 2018.

Jie Hu, Shen Li, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvari. Learning with a strong adversary. In *ICLR*, 2016.

Xiaojun Jia, Xingxing Wei, Xiaochun Cao, and Hassan Foroosh. Comdefend: An efficient image compression model to defend adversarial examples. In *CVPR*, 2019.

Luo Jiang, Juyong Zhang, and Bailin Deng. Robust rgb-d face recognition using attribute-aware loss. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:2552–2566, 2020.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *ICLR*, 2017a.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *ICLR*, 2017b.

Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *S&P*, 2019.

Linhao Li, Qinghua Hu, and Xin Li. Moving object detection in video via hierarchical modeling and alternating optimization. *IEEE Transactions on Image Processing*, 28:2021–2036, 2019.

Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *CVPR*, 2018.

Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *ICCV*, 2015.

Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In *ECCV*, 2018.

Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. Adv-bnn: Improved adversarial defense through robust bayesian neural network. In *ICLR*, 2019.

Xingjun Ma, Bo Li, Yisen Wang, Sarah Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. In *ICLR*, 2018.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *ICLR*, 2017.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *CVPR*, 2016.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *EuroS&P*, 2016a.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *S&P*, 2016b.

Yuan Rao and Jiangqun Ni. A deep learning approach to detection of splicing and copy-move forgeries in images. In *WIFS*, 2016.

Chuanbiao Song, Kun He, Jiadong Lin, Liwei Wang, and John Hopcroft. Robust local features for improving the generalization of adversarial training. In *ICLR*, 2020.

Jiawei Su, Danilo Vargas, Vasconcellos, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23:828–841, 2019.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

Jonathan Uesato, Brendan O'Donoghue, Aaron Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. In *ICML*, 2018.

Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*, 2018.

Cihang Xie, Yuxin Wu, Laurens Maaten, Alan Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *CVPR*, 2019.

Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *NDSS*, 2018.

Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael Jordan. Ml-loo: Detecting adversarial examples with feature attribution. In *AAAI*, 2020.

Peng Zhou, Xintong Han, Vlad Morariu, and Larry Davis. Learning rich features for image manipulation detection. In *CVPR*, 2018.

A APPENDIX

Table 2: The detailed two-stream network architectures for MNIST, CIFAR-10 and CIFAR-100. Conv(d,k,s) denotes the convolutional layer with d as dimension, k as kernel size and s as stride.

MNIST		CIFAR-10 and CIFAR-100	
RGB Stream	Noise Stream	RGB Stream	Noise Stream
Conv(32,3,1), ReLU	Conv(32,3,1), ReLU	Conv(32,3,1), ReLU	Conv(32,3,1), ReLU
Max Pooling	Max Pooling	Max Pooling	Max Pooling
Conv(64,3,1), ReLU	Conv(64,3,1), ReLU	Conv(64,3,1), ReLU	Conv(64,3,1), ReLU
Max Pooling	Max Pooling	Max Pooling	Max Pooling
Conv(128,3,1), ReLU	Conv(128,3,1), ReLU	Conv(128,3,1), ReLU	Conv(128,3,1), ReLU
Compact Bilinear Pooling		Max Pooling	Max Pooling
Full Connected 256, ReLU		Conv(256,3,1), ReLU	Conv(256,3,1), ReLU
Full Connected 256, ReLU		Max Pooling	Max Pooling
Softmax 2		Conv(256,3,1), ReLU	Conv(256,3,1), ReLU
		Full Connected 256, ReLU	
		Full Connected 256, ReLU	
		Softmax 2	

Table 3: Classifier architectures for MNIST and CIFAR-10. These three classifiers are designed by us, so it is specifically explained here in the form of a table. All models will be used to generate adversarial examples with different attack methods, but only local models can be used to assist detection methods.

MNIST Local Model	MNIST Black Model	CIFAR-10 Local Model
Conv(32,3,1), ReLU	Conv(32,3,1), ReLU	Conv(64,3,1), ReLU
Max Pooling 2×2	Conv(32,3,1), ReLU	Conv(64,3,1), ReLU
Conv(64,3,1), ReLU	Max Pooling 2×2	Max Pooling 2×2
Max Pooling 2×2	Conv(64,3,1), ReLU	Conv(128,3,1), ReLU
Full Connected 200	Conv(64,3,1), ReLU	Conv(128,3,1), ReLU
Softmax 10	Max Pooling 2×2	Max Pooling 2×2
	Full Connected 200	Full Connected 256
	Full Connected 200	Full Connected 256
	Softmax 10	Softmax 10

Table 4: Classifiers for CIFAR-10 and CIFAR-100

CIFAR-10 Black Model	CIFAR-100 Local Model	CIFAR-100 Black Model
VGG-16	ResNet101V2	ResNet152
MobileNet		DenseNet169
ResNet50		DenseNet201

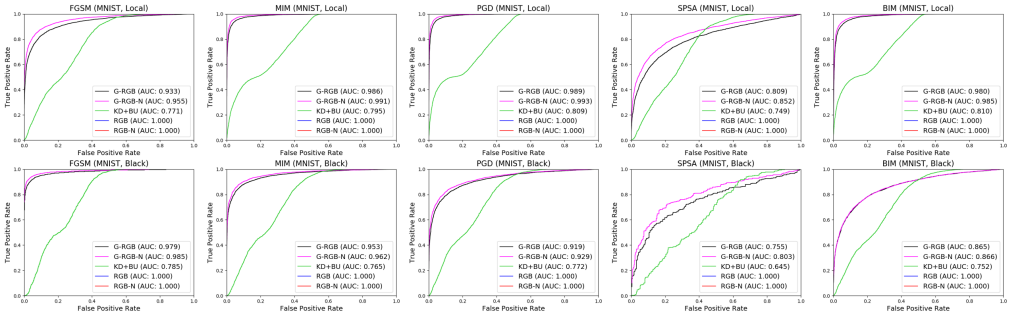

Figure 5: ROC curves of detection methods on MNIST. Due to the curves of RGB and RGB-N coincide with the coordinate axes, they are not displayed.

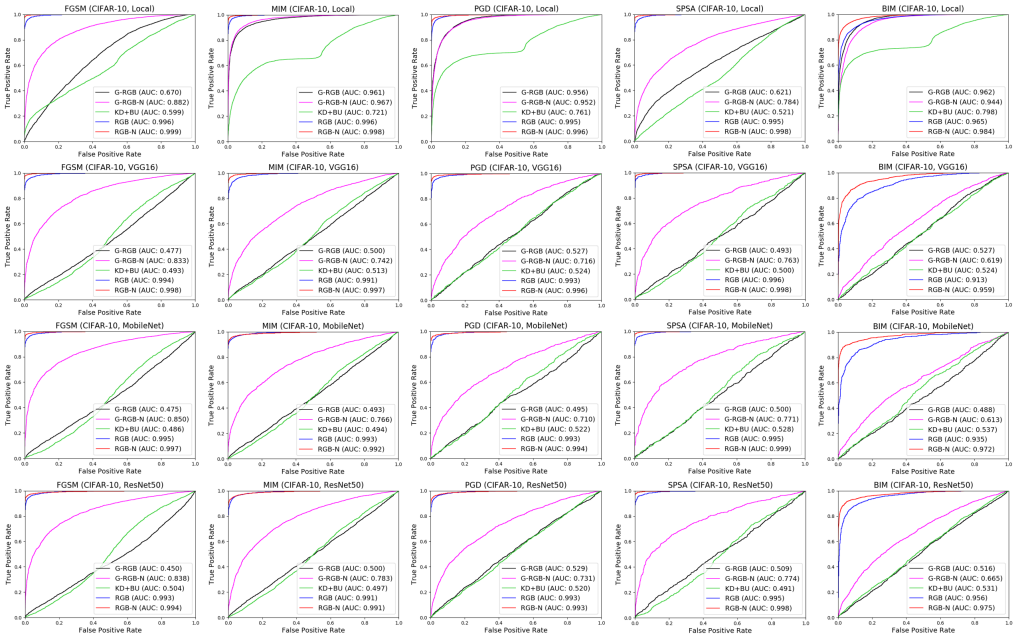

Figure 6: ROC curves of detection methods on CIFAR-10.

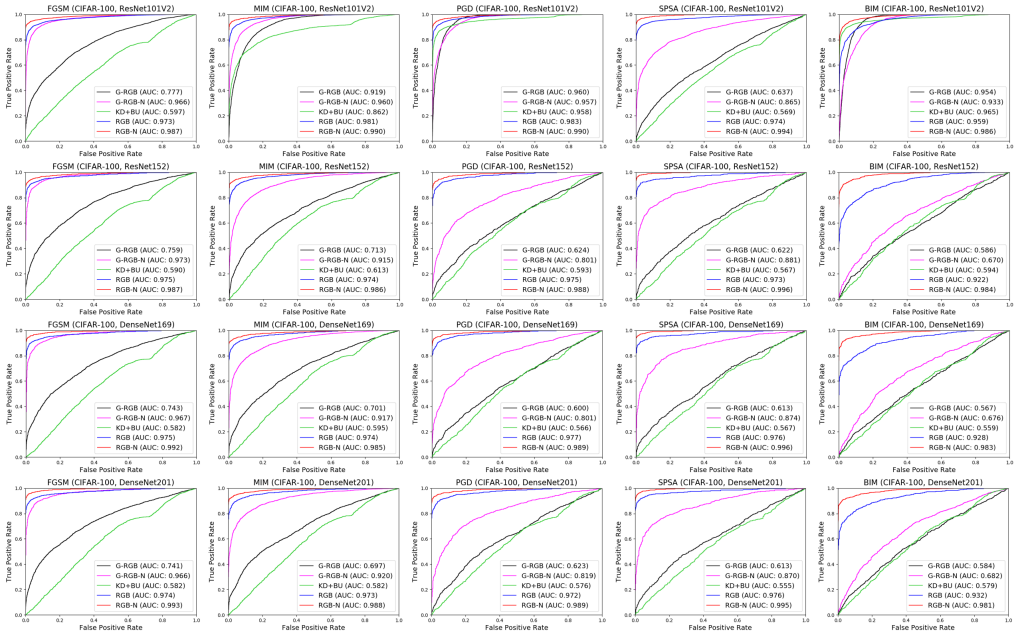

Figure 7: ROC curves of detection methods on CIFAR-100.