Defense against Adversarial Attacks Using High-Level Representation Guided Denoiser

Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Jun Zhu, Xiaolin Hu
Tsinghua National Lab for Information Science and Technology
State Key Lab of Intelligent Technology and Systems
Center for Bio-Inspired Computing Research
Department of Computer Science and Technology, Tsinghua University
{liaofangzhou,liangming.elgoog}@gmail.com
{dyp17@mails@mails,pty17@mails,dcszj@mail,xlhu@mail}.tsinghua.edu.cn

Abstract

Neural networks are vulnerable to adversarial examples. This phenomenon poses a threat to their applications in security-sensitive systems. It is thus important to develop effective defending methods to strengthen the robustness of neural networks to adversarial attacks. Many techniques have been proposed, but only a few of them are validated on large datasets like the ImageNet dataset. We propose high-level representation guided denoiser (HGD) as a defense for image classification. HGD uses a U-net structure to capture multi-scale information. It servers as a preprocessing step to remove the adversarial noise from the input, and feeds its output to the target model. To train the HGD, we define the loss function as the difference of the target model's outputs activated by the clean image and denoised image. Compared with the traditional denoiser that imposes loss function at the pixel-level, HGD is better at suppressing the influence of adversarial noise. Compared with ensemble adversarial training which is the state-of-the-art defending method, HGD has three advantages. First, with HGD as a defense, the target model is more robust to either white-box or black-box adversarial attacks. Second, HGD can be trained on a small subset of the images and generalizes well to other images, which makes the training much easier on large-scale datasets. Third, HGD can be transferred to defend models other than the one guiding it. We further validated the proposed method in NIPS adversarial examples dataset and achieved state-of-the-art result.

1. Introduction

As many other machine learning models [2], neural networks are known to be vulnerable to adversarial examples [22, 5]. Adversarial examples are maliciously designed inputs which have small perturbations on original inputs, but can mislead the target model. An even worse fact is that adversarial examples can be transferred across different models [22, 15]. This transferability enables black-box adversarial attacks without knowing the weights and structures of the target model. That is, adversarial examples are trained on a different model and then used to attack the target model. Black-box attacks have been shown to be feasible in real-world scenarios [16], which poses a potential threat to security-sensitive deep learning applications, such as identity authentication and autonomous driving. It is thus important to find effective defensive methods against adversarial attacks.

Since adversarial examples are constructed by adding noises to original images, a natural idea is to denoise the adversarial examples before sending them to a recognition model (Figure 1). Compared with adversarial training method, this method is more interpretable, which is beneficial for the understanding of adversarial examples as well as deep neural networks.

Two models are explored for denoising adversarial examples, and it is found that the noise level indeed can be reduced, which proves that the idea is feasible. However, none of the models can remove all perturbations introduced by the attacks, and small perturbation in the image could be amplified to large perturbation in higher-level representation (called "error amplification effect"), which leads to a wrong prediction. To solve this problem, instead of imposing the loss function at the pixel-level, we set the loss function of the denoiser as minimizing the difference of top-level outputs of the target model induced by the perturbed and clean images (Figure 1). We name the denoiser trained by this loss function "high-level representation guided denoiser" (HGD).

Compared with ensemble adversarial training [23],

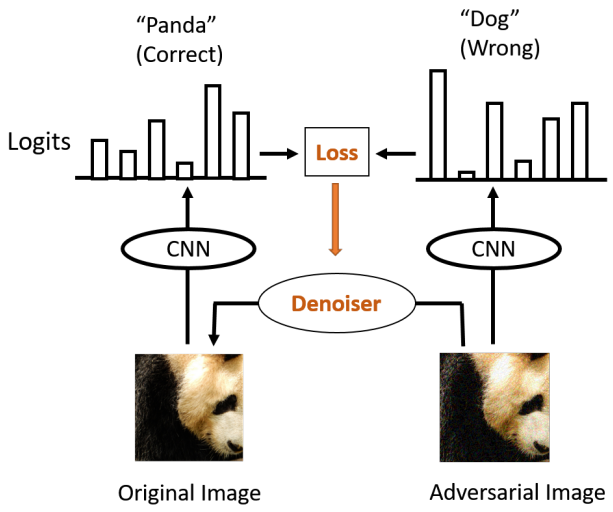


Figure 1: The idea of high-level representation guided denoising. The difference between the original image and adversarial image is tiny, but the difference is amplified in high-level representation (logits for example) of a CNN. We use the distance in high-level representation to guide the training of an image denoiser to suppress the influence of adversarial perturbation.

which is the current state-of-the-art method, the proposed method has the following advantages: first, it achieves much higher accuracy when defending both white-box and black-box attacks with only a slight decrease in accuracy on clean images. Second, it requires much less training data and training time. Third, it is more interpretable and can be transferred across different target models.

We further validate the performance of HGD on NIPS adversarial examples dataset. Our HGD method significantly outperforms other methods by a significant margin.

2. Background and Related Work

In this section, we first specify some of the notations used in this paper. Let x denote the clean image from a given dataset, and y denote the class. The ground truth label is denoted by y_{true}. A neural network $f : x \to y$ is called the target model. Given an input x, its feature vector at layer l is $f_l(x)$, and its predicted probability of class y is $p(y|x)$. $y_x = \arg\max_y p(y|x)$ is the predicted class of x. $J(x, y)$ denotes the loss function of the classifier given the input x and its target class y. For image classification, $J(x, y)$ is often chosen to be the cross-entropy loss. We use x^* to denote the adversarial example generated from x. ϵ is the magnitude of adversarial perturbation, measured by some distance metric.

2.1. Existing methods for adversarial attacks

Adversarial examples [22] are maliciously designed inputs which have a negligible difference from clean images but cause the classifier to give wrong classifications. That is, for x^* with a sufficiently small perturbation magnitude ϵ , $y_{x^*} \neq y_x$.

Szegedy et al. [22] used a box-constrained L-BFGS algorithm to generate a targeted adversarial example. More specifically, they minimize the weighted sum of ϵ and $J(x^*, y_{target})$ while constraining the elements of x^* to be within the range of pixel value.

Goodfellow et al. [5] suggest that adversarial examples can be caused by the summed effects of high dimensional model weights, and they further propose a simple adversarial attack algorithm, called Fast Gradient Sign Method (FGSM):

$$x^* = x + \epsilon \cdot \text{sign}(\nabla_x J(x, y)). \tag{1}$$

FGSM only computes the gradients once, and thus is much more efficient than L-BFGS. In early practices, FGSM uses the true label $y = y_{\text{true}}$ to compute the gradients. This approach is suggested to have the label leaking [12] effect. A better alternative is to replace y_{true} with the model predicted class y_x. FGSM is untargeted and aims to increase the overall loss. The targeted FGSM can be obtained by modifying FGSM to maximize the predicted probability of a specific target y_{target}:

$$x^* = x - \epsilon \cdot \text{sign}(\nabla_x J(x, y_{\text{target}})) \tag{2}$$

y_{target} can be chosen as the least likely class predicted by the model or a random class. Kurakin et al.[12] propose an iterative FGSM (IFGSM) attack by repeating FGSM for n steps (IFGSMn). IFSGM usually results in higher classification error than FGSM.

The model used to generate adversarial attacks is called the attacking model, which can be a single model or an ensemble of models [23]. When the attacking model is the target model itself or contains the target model, the resulting attacks are white-box. An intriguing property of adversarial examples is that they can be transferred to different models and datasets [22, 5]. This property enables black-box attacks. Practical black-box attacks have been demonstrated in some real-world scenarios [16, 15]. As white-box attacks are less likely to happen in real systems, defense against black-box attacks has more practical meaning.

2.2. Existing methods for defenses

Adversarial training [5, 12, 23] is one of the most extensively investigated defenses against adversarial attacks. It aims to train a robust model from scratch on a training set augmented with adversarially perturbed data [5, 12, 23]. Adversarial training can improve the classification accuracy of the target model on adversarial examples [22, 5, 12, 23].

It even improves the accuracy of clean images on some small datasets [22, 5], although this effect is not found on the large ImageNet dataset. However, adversarial training is more time consuming than training on clean images only, because online adversarial example generation costs extra computation, and it is harder to fit both the adversarial and clean examples. These limitations hinder the usage of newer and harder attacks in adversarial training, and practical adversarial training on the ImageNet dataset only adopts FGSM.

Another family of adversarial defenses is based on the so-called gradient masking effect [16, 17, 23]. These defenses apply some regularizers or smooth labels into the training process so that the output of learned models are less sensitive to the perturbation of input. Gu and Rigazio [6] propose the deep contrastive network, whose loss function includes a layer-wise contrastive penalty term. Such penalty forces the output of each layer invariant to the perturbation of its input. Papernot et al. [18] adapts knowledge distillation [9] to adversarial defense. The output of another model is used as soft labels to train the target model. Nayebi and Surya [14] use saturating networks for adversarial robustness. A penalty term in the loss function encourages the activations to be in their saturating regime.

The basic problem with these gradient masking approaches is that they do not solve the vulnerability of the models to adversarial attacks, but makes the construction of white-box adversarial examples more difficult. These defenses suffer from black-box attacks [16, 23] generated on some other models. Besides, as far as we know, most of them have not been validated in ImageNet-level tasks.

3. Methods

3.1. Pixel guided denoiser

During the adversarial training process, the two tasks of classifying clean images and defending against adversarial noises are coupled together, which is more difficult to learn than each task alone. This coupling may be unnecessary. A more straightforward way is learning to predict the adversarial noise only. The predicted noise can then be removed from the input before it is fed to the target model. This denoising approach is conducted as a preprocessing step, and the target model is not altered.

In this section, we introduce a set of denoising networks and their motivations. These denoisers are designed in the context of image classification on the ImageNet [4]. They are used in conjunction with a pretrained classifier f (By default Inception V3 [21] in this study). The denoising function is denoted as $D : x^* \rightarrow \hat{x}$, where $\hat{x}$ denotes the denoised image. The loss function is:

$$L = |x - \hat{x}|, \tag{3}$$

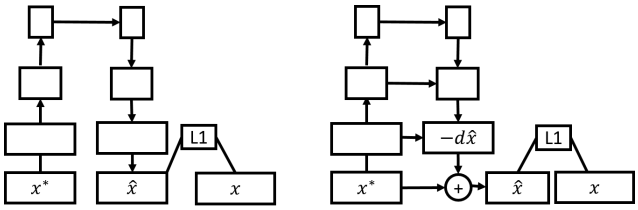


Figure 2: Diagrams of DAE (left) and DUNET (right).

where $|\cdot|$ stands for the L_1 norm. Since the loss function is defined at the pixel-level, we name this kind of denoiser pixel guided denoiser (PGD).

3.1.1 Denoising U-net

Denoising autoencoder (DAE) [24] is a potential choice of the denoising network. In previous work [6], DAE in the form of a multi-layer perceptron is used to defend images against adversarial attacks. However, their experiments were conducted over the relatively simple MNIST [13] dataset. To better represent the high-resolution images in the ImageNet, we use a convolutional version of DAE.

DAE has a bottleneck for the transmission of fine-scale information between the encoder and decoder. This bottleneck structure may not be capable of carrying the multi-scale information contained in the images. In other tasks requiring full resolution output, the network often has a multi-scale architecture, such as U-net [19] used for medical image segmentation. Here we use a denoising U-net.

So we designed a denoising U-net (DUNET). Compared with DAE, the DUNET adds some lateral connections from encoder layers to their corresponding decoder layers of the same resolution (Figure 2). These lateral connections serve as shortcuts to transmit fine-scale information. Thus, along with the top-down processing of the decoder, the fine-scale (i.e., low-level) information is gradually integrated with the coarse-scale (i.e., high-level) information. Note that there is also a shortcut from input to output to additively combine them. In this way, the network is learning to predict adversarial noise only ($d\hat{x}$ in Figure 2), which is more relevant to denosing and easier than reconstructing the whole image [26]. The clean image can be readily obtained by substracting the noise (adding -$d\hat{x}$)from the corrupted input.

3.1.2 Network structure

We use DUNET as an example to illustrate the architecture (Figure 3). DAE can be obtained simply by removing the lateral connections from DUNET. C is defined as a stack of a 3×3 convolutional layer, a batch normalization layer [10] and a rectified linear unit, and Ck is defined as k consecutive C. The network is composed of a feedforward path and a feedback path. The feedforward path is composed of five

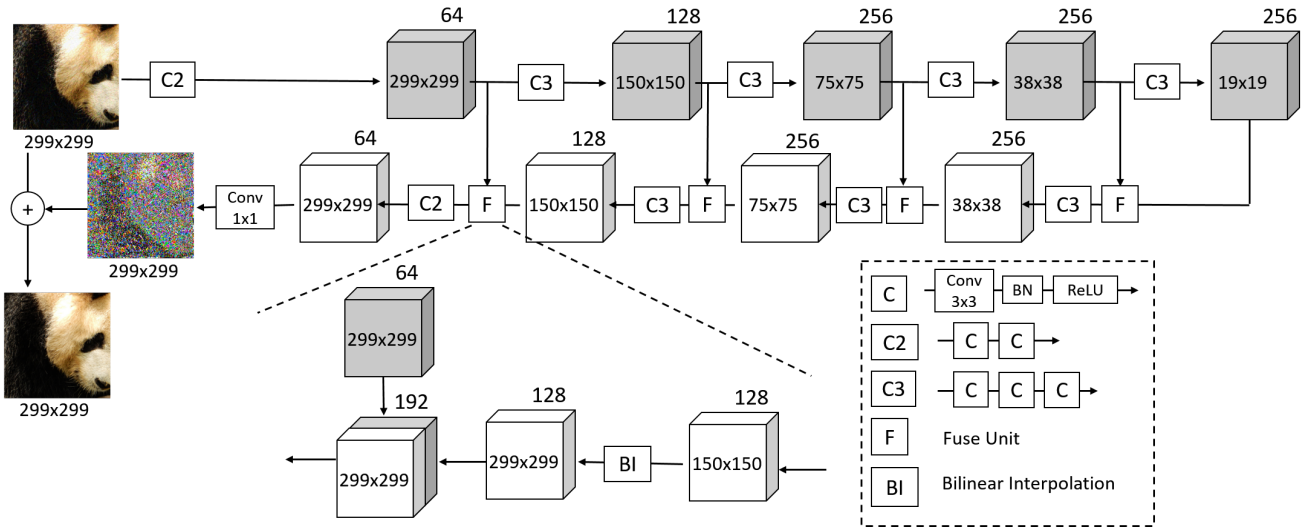


Figure 3: The detail of DUNET. The numbers inside each cube stand for width $\times$ height, and the number outside the cube stands for the number of channels. In all the C3 of the feedforward path, the stride of the first C is 2×2.

blocks, corresponding to one C2 and four C3, respectively. The stride of the first convolutional layer of each C3 is set to 2×2 for downsampling. The feedforward path receives the image as input, and generates a set of feature maps of increasingly lower resolutions (See the top pathway of Figure 3).

The feedback path is composed of four blocks and a 1×1 convolution. Each block receives a feedback input from the feedback path and a lateral input from the feedforward path. It first upsamples the feedback input to the same size as the lateral input using bilinear interpolation, and then processes the concatenated feedback and lateral inputs with a Ck. From top to bottom, three C3 and one C2 are used by the four blocks, respectively. Along the feedback path, the resolution of feature maps is progressively larger. The output of the last block is transformed to a predicted negative noise map $-d\hat{x}$ by the 1×1 convolution. (See the bottom pathway of Figure 3)The final output is the sum of the negative noise map and the input image:

$$\hat{x} = x^* - d\hat{x}. \tag{4}$$

3.2. High-level representation guided denoiser

A potential problem with PGD is the amplifying effect of adversarial noise. Adversarial examples have negligible differences from the clean images. However, this small perturbation is progressively amplified by deep neural networks and yields a wrong prediction. Even if the denoiser can significantly suppress the pixel-level noise, the remaining noise may still distort the high-level responses of the target model. Refer to Section 5.1 for details.

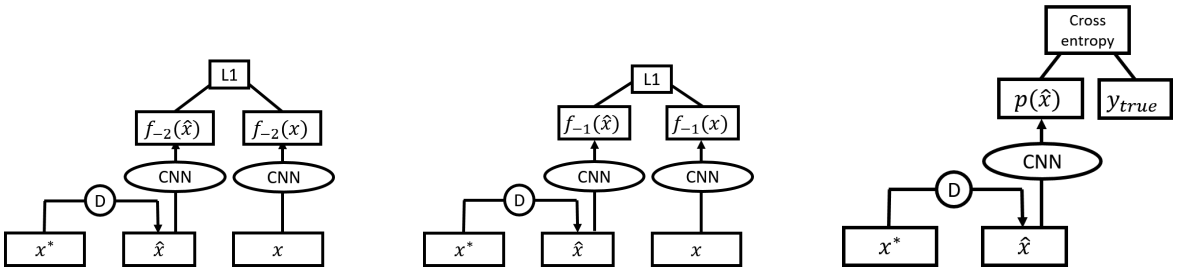
To overcome this problem, we replace the pixel-level loss function with the reconstruction loss of the target model's outputs. More specifically, given a target neural network, we extract its representation at l-th layer for x and $\hat{x}$, and calculate the loss function as:

$$L = |f_l(\hat{x}) - f_l(x)|. \tag{5}$$

The corresponding models are called HGD, in that the supervised signal comes from certain high-level layers of the classifier and carries guidance information related to image classification. HGD uses the same U-net structure as DUNET. They only differ in their loss functions.

We propose two HGDs with different choices of l. For the first HGD, we define $l = -2$ as the index of the topmost convolutional layer. The activations of this layer are fed to the linear classification layer after global average pooling, so it is more related to the classification objective than lower convolutional layers. This denoiser is called feature guided denoiser (FGD) (see Figure 4a). For the second HGD, we define $l = -1$ as the index of the layer before the final softmax function, which is called the logits. So it is called logits guided denoiser (LGD). In this case, the loss function is the difference between the two logits activated by $\hat{x}$ and x (see Figure 4b). We consider both FGD and LGD for the following reason. The convolutional feature maps provide richer supervising information, while the logits directly represent the classification results.

All PGD and these HGDs are unsupervised models, in that the ground truth labels are not needed in their training process. The denoiser just learns to remove the perturbation in the input or the feature maps of the target model. An alternative is to use the classification loss of the target model



(a) FGD (b) LGD (c) CGD

Figure 4: Three different training methods for HGD. The square boxes stand for data blobs, the circles and ovals stand for networks. D stands for denoiser. CNN is the model to be defended. The parameters of the CNN are shared and fixed.

as the denoising loss function, which is supervised learning as ground truth labels are needed. This model is called class label guided denoiser (CGD) (see Figure4c).

4. Experimental settings

Throughout experiments, the pre-trained Inception v3 (IncV3) [21] is assumed to be the target model that attacking models attempt to fool and our denoisers attempt to defend. Therefore this model is used for training the three HGDs illustrated in Figure 4. However, it will be seen that the HGDs trained with this target model can also defend other models (see Section 5.3).

4.1. Dataset

For both training and testing of the proposed method, adversarial images are needed. To prepare the training set, we first extract 30K images from the ImageNet training set (30 images per class). Then we use a bunch of adversarial attacking models to distort these images and form a training set by collecting the outputs of all models. Different attacking methods including FGSM and IFGSM are applied to the following models: Pre-trained IncV3, InceptionResnet v2 (IncResV2) [20], ResNet50 v2 (Res) [8] individually or in combinations (the same model ensemble as the work of Tramer et al. [23]). For each training sample, the perturbation level ϵ is uniformly sampled from integers in $[1, 16]$. See Table 1 for details. As a consequence, we gather 210K images in the training set (TrainSet).

To prepare the validation set, we first extract 10K images from the ImageNet training set (10 images per class), then apply the same method as described above. Therefore the size of the validation set (ValSet) is 70K.

Two different test sets are needed, one for white-box attack (WhiteTestSet)[1] and the other for black-box attack (BlackTestSet). They are obtained from the same clean 10K images from the ImageNet validation set (10 images per class) but using different attacking methods. The WhiteTestSet uses two attacks targeting at IncV3, which are also used for generating training images, and the BlackTestSet uses two attacks based on a holdout model Pre-trained Inception V4 (IncV4) [20], which is not used for generating training images. Every attacking method is conducted on two perturbation levels $\epsilon \in \{4, 16\}$. So both WhiteTestSet and BlackTestSet have 40k images (see Table 1 for details).

[1]The white-box attacks defined in this paper should be called oblivious attacks according to Carlini and Wagner's definition [3]

Table 1: Adversarial images generated by different models for training and testing.

	Attacking method	Attacked model	ϵ
TrainSet and ValSet	FGSM	IncV3	[1,16]
	FGSM	IncResV2	
	FGSM	Res	
	FGSM	IncV3/IncResV2/Res	
	IFGSM2	IncV3/IncResV2/Res	
	IFGSM4	IncV3/IncResV2/Res	
	IFGSM8	IncV3/IncResV2/Res	
WhiteTestSet	FGSM	IncV3	{4,16}
	IFGSM4	IncV3/IncResV2/Res	
BlackTestSet	FGSM	IncV4	{4,16}
	IFGSM4	IncV4	

4.2. Implementation details

The denoisers are optimized using Adam [11]. The learning rate is initially set to 0.001, and decayed to 0.0001 when the training loss converges. The model is trained on six GPUs and the batch size is 60. The number of training epochs ranges from 20 to 30, depends on the convergence speed of the model. The model with the lowest validation loss is used for testing.

5. Results

5.1. PGD and the error amplication effect

The results of DAE and DUNET on the test sets are shown in Table 2. The original IncV3 without any defense

Table 2: Denosing loss and classification accuracy of different PGD methods on the test sets. Denosing loss is the L_1 distance between the input image and the denoised image. NA means no defense. Clean stands for original images.

Defense	Clean	WhiteTestSet		BlackTestSet	
		$\epsilon = 4$	$\epsilon = 16$	$\epsilon = 4$	$\epsilon = 16$
NA	**0.0000**	0.0177	0.0437	0.0176	0.0451
DAE	0.0360	0.0359	0.0360	0.0360	0.0369
DUNET	0.0150	**0.0140**	**0.0164**	**0.0140**	**0.0181**
NA	**76.7%**	14.5%	14.4%	61.2%	41.0%
DAE	58.3%	**51.4%**	**36.7%**	55.9%	48.8%
DUNET	75.3%	20.0%	13.8%	**67.5%**	**55.7%**

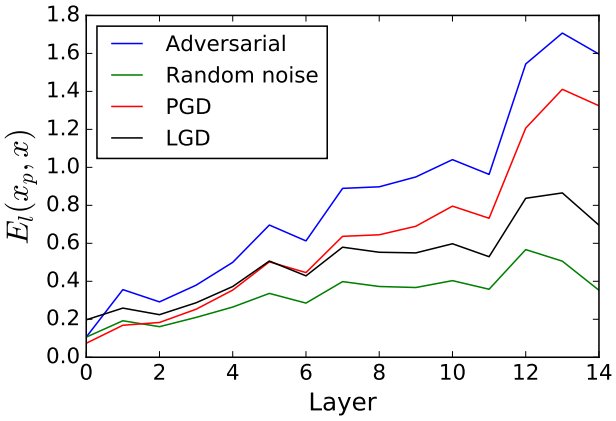


Figure 5: Layerwise perturbation levels of the target model. Adversarial, Random noise, PGD and LGD correspond to the E_l for adverarial images, Gaussian noise perturbed images, PGD denoised images, and LGD denoised images, respectively.

is used as a baseline, denoted as NA. For all types of attacks, DUNET has much lower denoising loss than DAE and NA, which demonstrates the structural advantage of DUNET. DAE does not perform well on encoding the high-resolution images, as its accuracy on clean images significantly drops. DUNET slightly decreases the accuracy of clean images, but significantly improves the robustness of the target model to black-box attacks. In the following content, DUNET is used as the default PGD method.

A notable result is that the denoising loss and classification accuracy of PGD are not so consistent. For white-box attacks, DUNET has much lower denoising loss than DAE, but its classification accuracy is significantly worse. To investigate this inconsistency, we analyze the layer-wise perturbations of the target model activated by PGD denoised images. Let x_p denote a perturbed image. The perturbation level at layer l is computed as:

$$E_l(x_p, x) = |f_l(x_p) - f_l(x)|/|f_l(x)|. \quad (6)$$

Table 3: The classification accuracy on test sets obtained by different defenses. NA means no defense.

Defense	Clean	WhiteTestSet		BlackTestSet	
		$\epsilon = 4$	$\epsilon = 16$	$\epsilon = 4$	$\epsilon = 16$
NA	76.7%	14.5%	14.4%	61.2%	41.0%
PGD	75.3%	20.0%	13.8%	67.5%	55.7%
ensV3 [23]	**76.9%**	69.8%	58.0%	72.4%	62.0%
FGD	76.1%	73.7%	67.4%	74.3%	71.8%
LGD	76.2%	75.2%	69.2%	**75.1%**	**72.2%**
CGD	74.9%	**75.8%**	**73.2%**	74.5%	71.1%

The E_l for PGD denoised images, adversarial images, and Gaussian noise perturbed images are shown in Figure 5. The latter two are used as baselines. For convenience, they are abbreviated as PGD perturbation, adversarial perturbation, and random perturbation. 30 images generated by the attack "IFGSM4 x IncV3/IncResV2/Res ($\epsilon = 16$)", which shows a high inconsistency, are used in this experiment. Although the pixel-level PGD perturbation significantly suppressed, the remaining perturbation is progressively amplified along the layer hierarchy. At the top layer, PGD perturbation is much higher than random perturbation and close to adversarial perturbation. Because the classification result is closely related to the top-level features, this large perturbation well explains the inconsistency between the denoising performance and classification accuracy of PGD.

5.2. Evaluation results of HGD

Compared to PGD, LGD strongly suppressed the error amplification effect (the black curve in Figure 5). LGD perturbation at the final layer is much lower than PGD and adversarial perturbations and close to random perturbation.

HGD is more robust to white-box and black-box adversarial attacks than PGD and ensV3 (Table 8). All three HGD methods significantly outperform PGD and ensV3 for all types of attacks. The accuracy of clean images only slightly decreases (by 0.5% for LGD). The difference between these HGD methods is insignificant. In later sections, LGD is chosen as our default HGD method for it achieves a good balance between accuracy on clean and adversarial images.

Compared to adversarial training, HGD only uses a small fraction of training images and is efficient to train. Only 30K clean images are used to construct our training set, while all 120K clean images of the ImageNet dataset are used for training ensV3. HGD is trained for less than 30 epochs on 21K images, while ensV3 is trained for about 200 epochs on 120K images [23].

To summary, with less training data and time, HGD significantly outperforms adversarial training on defense against adversarial attacks. These results suggest that learning to denoise only is much easier than learning the coupled

Table 4: The transferability of HGD to different classes. The 1000 ImageNet classes are separated in training and test test.

Defense	Clean	WhiteTestSet		BlackTestSet	
		$\epsilon = 4$	$\epsilon = 16$	$\epsilon = 4$	$\epsilon = 16$
NA	**76.6%**	14.5%	14.4%	61.2%	41.0%
LGD	76.3%	**73.9%**	**65.7%**	**74.8%**	**72.2%**

Table 5: The transferability of HGD to different model. Resnet is used as the target model in this table.

Denoiser for Resnet	Clean	WhiteTestSet		BlackTestSet	
		$\epsilon = 4$	$\epsilon = 16$	$\epsilon = 4$	$\epsilon = 16$
NA	**78.5%**	63.3%	38.4%	67.8%	48.6%
IncV3 guided LGD	77.4%	75.8%	71.7%	76.1%	72.7%
Resnet guided LGD	78.4%	**76.1%**	**72.9%**	**76.5%**	**74.6%**

task of classification and defense.

5.3. Transferability of HGD

Unlike adversarial training, HGD uncouples defense from classification, and its learning objective is the adversarial noise itself. As adversarial attacks can be transferred on different models and datasets [22], HGD is expected to have similar transferabilities.

To evaluate the transferability of HGD over different models, we use the IncV3 guided LGD to defend Resnet [7]. As expected, this LGD significantly improve the robustness of Resnet to all attacks. Furthermore, it achieves very close defending performance as the Resnet guided LGD. (Table 5)

To evaluate the transferability of HGD over different classes, we build another dataset. Its key difference from the original dataset is that there are only 750 classes in the TrainSet, and the other 250 classes are put in ValSet and TestSets. The number of original images in each class in all datasets are changed to 40 to keep the size of dataset unchanged. It is found that although the 250 classes in the test set are never trained, the LGD still learns to defend against the attacks targeting at them (Table 4).

5.4. Qualitative analysis on denoised images

HGD is much more robust to adversarial attacks than PGD. However, HGD denoised images have higher pixel-level noise than adversarial images (see Figure 5), indicating HGD even increases the "noise" on images. In this section, we qualitatively analyze this phenomenon. The images used in the experiments are the same as those in Section 5.1.

An example image, its adversarial image, and its denoised outputs by PGD and LGD are shown in Figure 6.

Table 6: The results on the NIPS dataset. Time stands for average evaluating time.

Team/Method	Normalized Score	Time(s)
iyswim	0.9235	121.8
Anil Thomas	0.9148	95.3
erko	0.9120	86.4
FGD(ours)	**0.9532**	**50.2**

It can be observed that LGD does not suppress the overall pixel-level perturbation like what PGD does. Instead, it even increases the perturbation level. To investigate this issue further, We plot the 2D histogram of the adversarial perturbation ($dx^* = x^* - x$) and the predicted perturbation ($d\hat{x} = x^* - \hat{x}$) given by the PGD and LGD (Figure 7). The ideal result should be $d\hat{x} = dx^*$, which means the adversarial perturbations are completely removed.

Two lines $d\hat{x} = kdx^*$ are fit for PGD and LGD, respectively (the red lines in Figure 7). The slope of PGD's line is lower than 1, indicating that PGD only removes a portion of the adversarial noises. On contrary, the slope of LGD's line is even larger than 1. Moreover, the estimation is very noisy, which leads to high noise in the pixel level. These results suggest that LGD may denoise the images in a relatively aggressive way.

5.5. The solution in NIPS dataset

The NIPS adversarial examples dataset provided 5000 ImageNet-compatible original images, on which 91 non-targeted attacks and 65 targeted attacks are conducted. All these images and attacks are unknown to users, and the evaluation is conducted on the cloud by organizers. Each defending method is evaluated on all the attacks, and a normalized score is calculated based on the accuracy of all attacks.

To achieve higher performance, we gathered 14 powerful attacks, all with $\epsilon = 16$. Most of them are iterative attacks on an ensemble of many models (for details, please refer to supplementary file). We choose four pre-trained models (ensV3[23], ensIncResV2[23], Resnet152[7], ResNext101[25]) and trained a FGD for each one. The logit output of the four defended models are averaged, and the class with the highest score is chosen as the prediction.

Without elaborately parameter tuning, our solution achieved the best result with a significantly higher score and shorter evaluating time than other top methods (Table 6).

6. Conclusion

In this study, we discovered the error amplification effect of adversarial examples in neural networks and proposed to

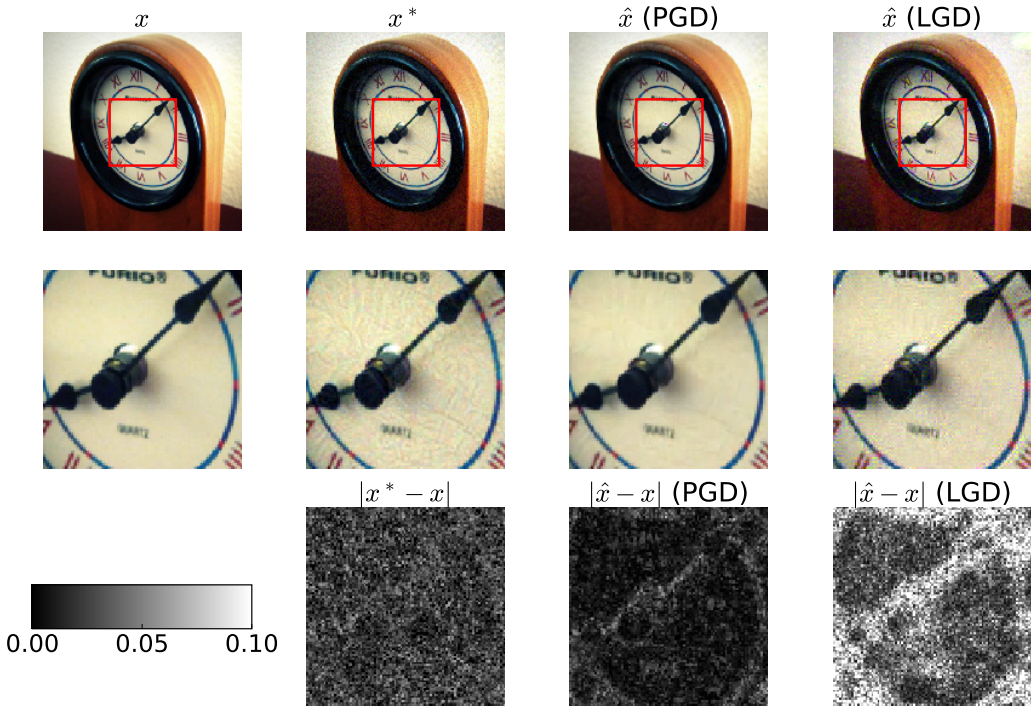


Figure 6: The first row is an original image (1st column), adversarial image (2nd column), denoised adversarial image generated by PGD (3rd column) and LGD (4th column). The second row shows zoomed in images. The third row visualizes the L_1 norm of differences between the original image and the last three images in the second row, relatively.

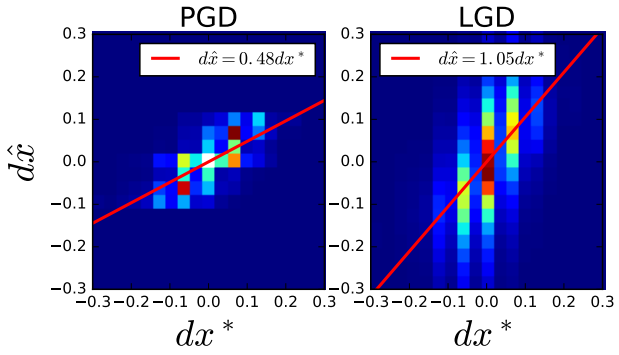


Figure 7: The relationship between dx^* and $d\hat{x}$ in PGD and HGD.

use the error in the top layers of the neural network as loss functions to guide the training of an image denoiser. This method turned to be very robust against both white-box and black-box attacks. The proposed HGD has simple training procedure, good generalization, and high flexibility.

In future work, we aim to build an optimal set of training attacks. The denoising ability of HGD depends on the representability of the training set. In current experiments, we used FGSM and iterative attacks. Incorporating other different attacks, such as the attacks generated by adversarial transformation networks [1], probably improves the performance of HGD. It is also possible to explore an end-to-end training approach, in which the attacks are generated online by another neural network.

Table 7: The influence of adding a LGD before ensV3. For fair comparison, the white-box attacks (WhiteTestSet) used in the two blocks are targeting at IncV3 and ensV3 respectively.

Defense	Clean	WhiteTestSet		BlackTestSet	
		$\epsilon = 4$	$\epsilon = 16$	$\epsilon = 4$	$\epsilon = 16$
IncV3	76.7%	14.5%	14.4%	61.2%	41.0%
LGD + IncV3	76.2%	**75.2%**	69.2%	**75.1%**	**72.2%**
ensV3	**76.9%**	71.4%	61.7%	72.4%	62.0%
LGD + ensV3	**76.9%**	75.0%	**72.5%**	74.7%	72.1%

Supplementary materials

A. Combination of HGD and adversarial trained model

As two different approaches, denoising networks and adversarial training may have complementary effects, use a HGD to process the distorted images before they are inputted to an adversarially trained model may further improve the performance. To test this idea, we train an LGD with ensemble adversarially trained Inception V3 (ensV3) [23] as the target model. For fair comparison, we replace the attack methods targeting at IncV3 (and other models) with attack methods targeting at ensV3 (and other models) and make a new dataset correspondingly. The LGD+ensV3 model is trained and tested on this new dataset.

Because ensV3 is more robust than IncV3, we expect to see higher robustness in LGD+ensV3. However, the results show that although LGD helps the ensv3 to improve the robustness, their combination is not significantly better than the LGD+IncV3 (Table 7) (The results of LGD+IncV3 are copied from Table 3.).

B. Remedy for the low slope of PGD

In Section 5.4 of the paper, we show the different denoising properties of PGD and LGD ($d\hat{x} = kdx^*$), and it may be inferred that the low k value of PGD is an important factor for PGD's poor performance. To validate this assumption, we replace the output of PGD with $\hat{x} = x^* - 2d\hat{x}$ (i.e. replace $-d\hat{x}$ in Fig. 2 by $-2d\hat{x}$), so that its k is close to 1. The results (denoted by PGD x2 in Table 8) are significantly higher than those of the original PGD but still not as high as those of LGD. Besides, this change also significantly decreases the accuracy on clean images. Therefore the low k value of PGD is indeed a reason for its poor robustness, but it cannot explain all difference between PGD and LGD.

C. Ensembles of HGD protected models

Ensemble is an effective method to boost the performance of classifiers. We explore the ensemble of several HGDs and target models. Specifically, we train two LGDs. The first one is denoted by LGD1, which is trained with IncV3 as the target model and the second one is denoted by LGD2, which is trained with ensV3 as the target model[2]. Different combinations of the two denoisers and various target models are tested:

Table 8: The classification accuracy on test sets obtained by different defenses. NA means no defense.

Defense	Clean	WhiteTestSet		BlackTestSet	
		$\epsilon = 4$	$\epsilon = 16$	$\epsilon = 4$	$\epsilon = 16$
NA	**76.7%**	14.5%	14.4%	61.2%	41.0%
PGD	75.3%	20.0%	13.8%	67.5%	55.7%
PGD x2	73.7%	40.7%	50.8%	70.9%	67.4%
LGD	76.2%	**75.2%**	**69.2%**	**75.1%**	**72.2%**

- **IncV3&ensV3**. The adversarial images (x^*) are fed directly to the two models without protection, and the output logits of IncV3 and ensV3 are averaged as the result. The symbol & indicate an ensemble.

  Motivation: This method serves as the baseline ensemble.

- **LGD1→ IncV3&ensV3**. x^* is firstly fed to the LGD1, resulting in a denoised image $\hat{x}$, which is then fed to the ensemble of IncV3&ensV3. The $\rightarrow$ indicates the flow of data.

  Motivation: LGD shows certain transferability across different models (Section 5.3 in main paper). So it is possible to use an LGD to protect multiple models.

- **(LGD1→IncV3&ensV3)&(LGD2→IncV3&ensV3)**. LGD1 and LGD2 give two denoised images $\hat{x}_1$ and $\hat{x}_2$, which are then fed to IncV3&ensV3 independently. The four output logits are averaged as the result.

  Motivation: This method is similar with the last one, but make use of LGD2.

- **LGD1&LGD2→IncV3&ensV3**. The output of LGD1 and LGD2 are averaged ($\hat{x} = (\hat{x}_1 + \hat{x}_2)/2$). The averaged denoised image is then fed to IncV3&ensV3.

  Motivation: Each LGD give an independent estimation of the adversarial perturbation, so averaging the outputs of two LGD may result in a better estimation of perturbation.

- **(LGD1→IncV3)&(LGD2→ensV3)**. $\hat{x}_1$ and $\hat{x}_2$ are fed to IncV3 and ensV3 respectively. The logits of the two models are averaged as result.

  Motivation: The most straightforward way of ensemble.

[2]Different from Section A, the LGD2 used in this section is trained and evaluated on the default dataset.

Except for these ensembles, two single model baselines **LGD1→IncV3** and **LGD2→ensV3** are also tested.

The results of these methods are shown in Table 9. (LGD1→IncV3)&(LGD2→ensV3) performs the best and shows consistent improvement comparing to baselines. Other ensemble methods achieve little improvements comparing with the single models. Some of them even have degraded performance.

D. The details of NIPS 2017 solution

In this section, we list the attacks we used to generate the training and validation sets for training the denoiser. ϵ is the L_∞ constraint of the adversarial perturbation. In our experiment, $\epsilon = 16$ is fixed.

- **No operation.** The clean images are directly adopted.
- **Images withs random noise.** Each pixel of an original image is randomly perturbed with ϵ or $-\epsilon$ with equal probability.
- **FGSM x IncV3.** The word before "x" indicates the attacking method, and the word after it indicates the target model. FGSM meas fast gradient sign method (defined in the main paper).
- **FGSM x ensV3&advV3.** advV3 is an adversarially trained Inception V3 model [12].
- **FGSM x IncV3&IncV4&ensIncResV2.** IncV4 is Inception V4 [20]. EnsIncResV2 is an ensemble adversarially trained InceptionResnet V2 [20, 23].
- **IFGSM^2 x 7 models.** IFGSM^k means k step iterative FGSM. The step size of each step is ϵ/k. 7 models means the ensemble of IncV3, advV3, ensV3, ens4V3[23], IncV4, IncResV2 and ensIncResV2. ens4V3 is another ensemble adversarially trained Inception V3 [23]. IncResV2 is the Inception-Resnet V2 [20].
- **IFGSM^4 x 7 models.**
- **IFGSM^8 x 7 models.**
- **IFGSM^2 x 8 models.** 8 models means the ensemble of the 7 models mentioned above and Resnet101[7].
- **IFGSM^8 x 8 models.**
- **dIFGSM^4 x 8 models.** dIFGSM means IFGSM with decayed step size, which is set as 0.4ϵ, 0.3ϵ, 0.2ϵ, 0.1ϵ for dIFGSM^4.
- **adaIFGSM^1 x 8 models.** adaIFGSM means adaptive IFGSM. In adaIFGSM, there is a list of IFGSM attacks with increasing power and running time (i.e. more iterations). An original image is firstly perturbed by the simplest attack. If this adversarial image successfully fools all the target models, it is used as output and the procedure ends, else the original image would be perturbed by the attack in the next level. This procedure continues until the adversarial image fools all the target models or the last attack in the list is used.

  In adaIFGSM^1, the attack list is (dIFGSM^4 x 8 models, IFGSM^{20} x 8 models).
- **adaIFGSM^2 x 8 models.** In adaIFGSM^2, the attack list is (dIFGSM^3 x 8 models, dIFGSM^4 x 8 models, IFGSM^{20} x 8 models). The step size is set as $\frac{1}{2}\epsilon$, $\frac{1}{3}\epsilon$, $\frac{1}{6}\epsilon$ for dIFGSM^3.
- **adaIFGSM^3 x 8 models.** In adaIFGSM^3, the attack list is (FGSM x 8 models, IFGSM^2 x 8 models, IFGSM^4 x 8 models, IFGSM^8 x 8 modelS, IFGSM^{20} x 8 models).

These attacks are applied to 15,000 original images, resulting in a training set of 210,000 adversarial images. And they are applied to other 5,000 original images, resulting in a validation set of 70,000 adversarial images.

References

[1] Shumeet Baluja and Ian Fischer. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387*, 2017.

[2] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402, 2013.

[3] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *arXiv preprint arXiv:1705.07263*, 2017.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[5] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[6] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

Table 9: The classification accuracy on the test set obtained by different methods.

Defense	Clean	WhiteTestSet		BlackTestSet	
		$\epsilon = 4$	$\epsilon = 16$	$\epsilon = 4$	$\epsilon = 16$
LGD1→IncV3	76.2%	75.2%	69.2%	75.1%	72.2%
LGD2→ensV3	76.9%	74.4%	71.8%	75.2%	73.8%
IncV3&ensV3	**78.8%**	35.6%	30.0%	70.2%	56.6%
LGD1→IncV3&ensV3	77.6%	75.6%	71.3%	75.5%	72.7%
(LGD1→IncV3&ensV3)&(LGD2→IncV3&ensV3)	78.5%	69.9%	67.8%	77.0%	74.7%
LGD1&LGD2→IncV3&ensV3	78.2%	70.6%	67.6%	76.5%	74.5%
(LGD1→IncV3)&(LGD2→ensV3)	78.6%	**77.4%**	**73.4%**	**77.7%**	**75.7%**

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645, 2016.

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[11] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[14] Aran Nayebi and Surya Ganguli. Biologically inspired protection of deep networks from adversarial attacks. *arXiv preprint arXiv:1703.09202*, 2017.

[15] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[16] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *ACM Asia Conference on Computer and Communications Security*, pages 506–519, 2017.

[17] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814*, 2016.

[18] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 582–597, 2016.

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, 2015.

[20] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017.

[21] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[22] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[23] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[24] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing

robust features with denoising autoencoders. In *International Conference on Machine learning*, pages 1096–1103, 2008.

[25] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.

[26] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 2017.