Computational Visual Media
https://doi.org/10.1007/s41095-021-0202-3 Vol. 7, No. 2, June 2021, 217–227

Research Article

An end-to-end convolutional network for joint detecting and denoising adversarial perturbations in vehicle classification

Peng Liu[1], Huiyuan Fu[1] (✉), and Huadong Ma[1]

Abstract Deep convolutional neural networks (DCNNs) have been widely deployed in real-world scenarios. However, DCNNs are easily tricked by adversarial examples, which present challenges for critical applications, such as vehicle classification. To address this problem, we propose a novel end-to-end convolutional network for joint detection and removal of adversarial perturbations by denoising (DDAP). It gets rid of adversarial perturbations using the DDAP denoiser based on adversarial examples discovered by the DDAP detector. The proposed method can be regarded as a pre-processing step—it does not require modifying the structure of the vehicle classification model and hardly affects the classification results on clean images. We consider four kinds of adversarial attack (FGSM, BIM, DeepFool, PGD) to verify DDAP's capabilities when trained on BIT-Vehicle and other public datasets. It provides better defense than other state-of-the-art defensive methods.

Keywords adversarial defense; adversarial detection; vehicle classification; deep learning

1 Introduction

In recent years, deep convolutional neural networks (DCNNs) have been widely used in many different tasks, such as image recognition [1–3], self-driving vehicles [4], semantic segmentation [5], and vehicle re-identification [6]. As an essential requirement of an intelligent transport system, remarkable performance has been achieved in vehicle classification [7, 8]. However, recent studies [9–11] have shown that DCNNs are vulnerable to adversarial examples, specially-crafted by making minute perturbations to natural images. Such perturbations can cause a classifier to misclassify an image with high confidence in the wrong result. Figure 1 shows that an adversarial example causes misclassification of an SUV as a bus. Clearly, it is important to make deep convolutional neural networks that are robust in the face of adversarial attacks.

Previous defenses to adversarial attacks are mainly of two kinds. The first kind trains a detector network [12–14], which acts as a filter rejecting malicious input to the target model. The other kind uses a defensive model to decrease the effects of adversarial perturbations and improve the adversarial robustness of the target model [15–17]. However, Xie et al. [18] have shown that denoising may affect the performance of the target model on clean images, as valid information for classification may be lost in the denoising process.

In this paper, we propose a new method based on joint detection and removal of adversarial perturbations by denoising (DDAP). Unlike previous work, our defensive method combines an adversarial perturbation detector and a denoiser, using joint learning for end-to-end training. Adversarial examples detected by the detector are passed to the denoiser to remove perturbations. The detector and denoiser share the same parameters in the feature extraction stage to reduce the amount of calculation.

The main contributions of this paper are:

- an end-to-end defensive method combining detection and removal of adversarial perturbations by denoising, for vehicle classification. It can be applied as a pre-processing method to improve the robustness of vehicle classification models;

1 Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing 100876, China. E-mail: P. Liu, lp864172708@bupt.edu.cn; H. Fu, fhy@bupt.edu.cn (✉); H. Ma, mhd@bupt.edu.cn.

Manuscript received: 2020-11-03; accepted: 2021-01-02

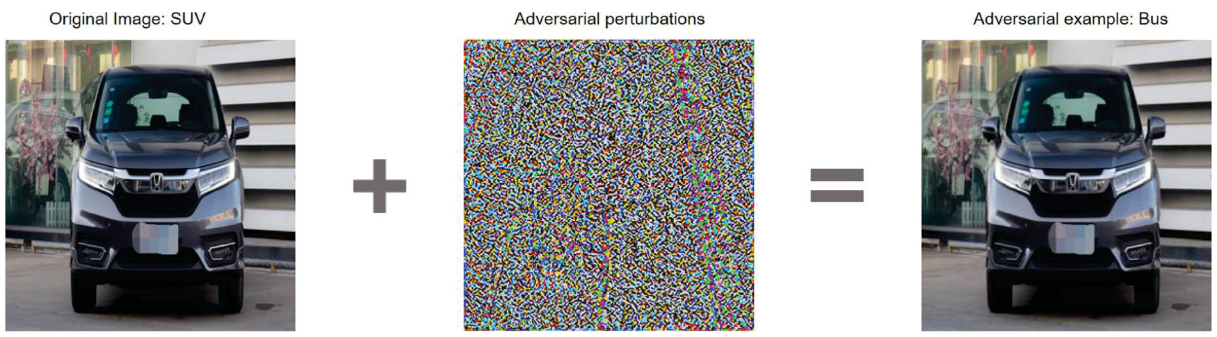

Fig. 1 Generation of an adversarial example.

- a new loss function for joint supervised training of the adversarial perturbation detector and denoiser, which is beneficial to optimization of the model;
- an evaluation on two vehicle datasets which shows that our method provides state-of-the-art performance against both white-box and black-box attacks, with negligible reduction in classifier performance for clean images.

The rest of this paper is organized as follows. Section 2 reviews popular adversarial attacks and defense methods. Our proposed approach is described in detail in Section 3. Section 4 provides experimental results in defending against adversarial attacks. Finally, Section 5 concludes our work.

2 Related work

In this section, the literature on adversarial attack methods is reviewed, and then mechanisms for detection and defense against adversarial attacks are introduced.

2.1 Adversarial attack methods

The concept of adversarial examples was proposed by Szegedy et al. [19]; subsequent work [9–11] showed that DNNs are vulnerable to adversarial examples. Given a classification model f, deliberately adding some subtle perturbations p to the correctly classified image x, will cause f to give a wrong output yet with high confidence that $f(x+p) \neq f(x)$. We now describe some well-known adversarial attack algorithms. It should be emphasized that we are concerned with untargeted adversarial attacks.

Goodfellow et al. [20] introduced the attack known as the single step fast gradient sign method (FGSM). It keeps the amount of change consistent with the direction of the gradient, thus inverting the output of the classifier. An adversarial example can be expressed in the following form:

$$\hat{x} = x + \varepsilon\,\mathrm{sign}(\nabla_x J(x, y)) \tag{1}$$

The acquisition of adversarial examples aims to maximize the loss function $J(x, y)$ which measures the classification error, usually cross-entropy. Maximizing J causes the example no longer belong to the correct class y after adding noise. In the optimization process, the difference between the original example and adversarial example needs to be within a certain range $\| \hat{x} - x \| \leqslant \epsilon$. sign() is the sign function, acting on the partial derivative of the loss function with respect to x.

Extending FGSM, BIM [9] generates adversarial examples by using FGSM multiple times with a smaller step α. In each iteration, clip() is used to ensure that generated perturbations stay within the $\epsilon-$neighborhood of the image x. clip is defined as

$$\mathrm{clip}_{x,\epsilon} = \min\left\{255, x+\epsilon, \max\left\{0, x-\epsilon, \hat{x}\right\}\right\} \tag{2}$$

DeepFool [21] perturbs the image by a small vector, and gradually pushes the image within the classification boundary until incorrect classification occurs. The adversarial example for iteration $k+1$ is

$$x_{k+1} = x_k + \frac{\left|f_i(x_k) - f_{\hat{i}}(x_k)\right|}{\| \nabla f_i(x_k) - \nabla f_{\hat{i}}(x_k) \|_2^2} \times \left(\nabla f_i(x_k) - \nabla f_{\hat{i}}(x_k)\right) \tag{3}$$

where f denotes the classifier model, $f_i(x)$ is the ith dimension of the output, and $f_{\hat{i}}(x)$ represents the dimension with the largest output. DeepFool proves that the generated perturbations are smaller than those of FGSM, while achieving similar deception rates.

Projected gradient descent (PGD) [22] can be regarded as a similar iterative attack method to FGSM. FGSM uses only one iteration, while PGD performs multiple iterations, taking a small step each time, and each iteration's change is clipped to a specified range. The difference between PGD and BIM is that the former applies random perturbations. PGD calculates adversarial samples for iteration $k+1$ using:

$$x_{k+1} = \prod_{x+S} \left(x_k + \alpha\,\mathrm{sign}\left(\nabla_x J\left(x_k, y\right)\right)\right) \tag{4}$$

2.2 Detection and defense for adversarial examples

Different methods have been suggested for discovering adversarial attacks. Carrara et al. [12] utilized KNN classification for hidden layer activation to distinguish correctly classified images and adversarial images. Rakin and Fan [14] built a simple convolutional neural network with two convolutional layers and fully connected layers to detect adversarial examples instead of adversarial training. Similarly, Metzen et al. [23] attached a subnetwork to the classifier model to deal with adversarial examples. Feinman et al. [24] verified the combination of kernel density estimates in the subspace of the last hidden layer; Bayesian neural network uncertainty estimates can effectively discover adversarial perturbations. Adaptive noise reduction with scalar quantization and smoothing spatial filter are used to detect adversarial noise in Ref. [25]. A transferability prediction difference method [13] detects adversarial examples by measuring the transferability difference in various DNN models.

Papernot et al. [26] considered a defensive distillation method to resist adversarial attacks. Results generated by the example using the original neural network are regarded as new labels to train a distillation network with the same architecture and distillation temperature T, which is used for classification. However, Carlini and Wagner [27] showed that defensive distillation is unable to increase the robustness of neural networks. Samangouei et al. [28] borrowed a generative adversarial network to suppress adversarial attacks on the MNIST [29] digits dataset, but the results are hard to transfer to other datasets. Liao et al. [15] proposed to eliminate adversarial perturbations using high-level features, with the output difference before softmax as the loss function. Prakash et al. [17] developed a technology that combines computationally efficient image transformation, redistribution of pixel values, and soft wavelet noise reduction to overcome perturbations. Mustafa et al. [16] considered a novel defensive mechanism: image super-resolution enhances the quality, and projects adversarial examples onto the manifold of natural images.

3 Method

The framework of our proposed DDAP method (see Fig. 2) consists of two models. The first is an adversarial perturbation detector and the second is an adversarial perturbation denoiser. Both share the same parameters for feature extraction. We now explain the proposed defensive method in detail.

3.1 Adversarial perturbations detector

Adversarial attackers generate small perturbations that are often imperceptible to humans, yet fool the classifier. We emphasize that adversarial images with added perturbations change the pixel distribution and fall outside the data manifold for real examples [14, 30]. Therefore, we can train a detection network to determine whether an example is adversarial through the feature representation of the input data [14, 23].

As illustrated in Fig. 3, given an input image $x \in \mathbb{R}^{3\times w\times h}$, we define E to be a mapping function from x to the features generated by feature extraction;

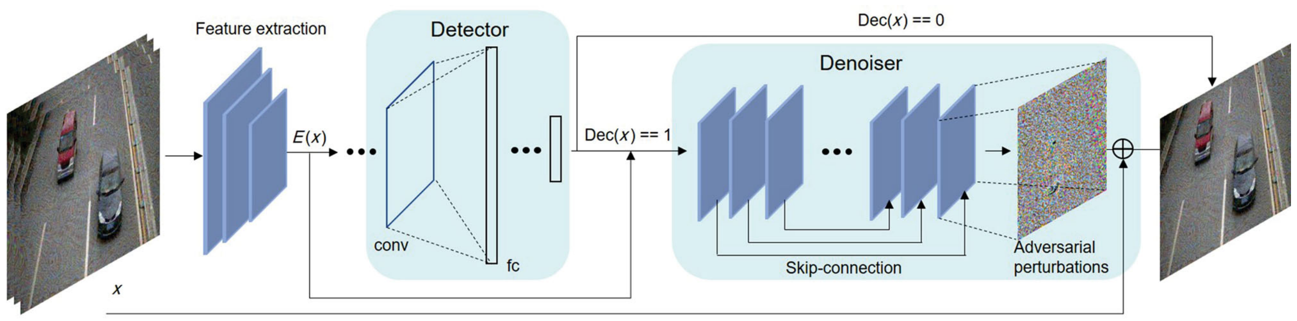

Fig. 2 Overview of our end-to-end framework. We jointly detect and remove adversarial perturbations by denoising. When the input x is recognized as an adversarial example by the detector, Dec(x) == 1, adversarial perturbations are excluded by the denoiser model before x is classified. Any clean image x is directly passed to the classification model.

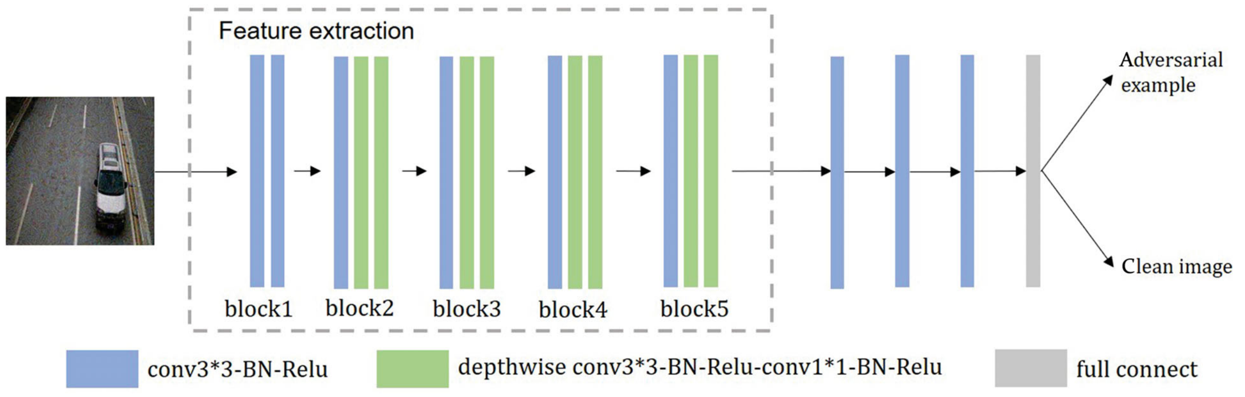

Fig. 3 Architecture of the adversarial perturbation detector. Each conv unit contains a 3×3 convolution layer, a batch normalization layer, and a RELU function. Each light conv unit contains a 3×3 depthwise convolution layer, a batch normalization layer, a RELU function and a 1×1 pointwise convolution layer, a batch normalization layer, and a RELU function.

Dec is a mapping function from features to the prediction category of the detector. More specifically, the input data is forwarded through multiple blocks to obtain $E(x)$. For calculation speed and to reduce the number of parameters, normal convolution in blocks 2 to 5 is replaced by depthwise convolution and pointwise convolution except for the first convolution layer of each block [31]. Furthermore, the first convolutional layer in blocks 2 to 5 adapts a 2×2 stride for feature downsampling, making $E(x)$ 4 times smaller than x. $E(x)$ is then fed into three conv units and a fully-connected layer to learn the discriminative difference between the features of clean images and adversarial examples. Note that the last conv unit also utilizes 2×2 stride operations; the fully-connected layer following softmax produces a two-dimensional vector. In the inference phase, we select the index with maximum value as the detector output. Zero represents a clean image and one represents an adversarial example. If the output of $\mathrm{Dec}(E(x))$ is a clean image, the original image x is delivered directly to the target classification model. Otherwise, the feature representation $E(x)$ is passed to the denoiser to eliminate the adversarial perturbations.

3.2 Adversarial perturbation denoiser

If the detector classifies the input x as an adversarial sample, the denoiser employs the features $E(x)$ to reconstruct the sample, to deal with the perturbations. The reconstruction function Den has limitations; the goal is for the reconstructed sample to be as similar as possible to the original data: $\mathrm{Den}(E(x)) \approx x$. In fact, the combination of feature extraction and the denoiser can be considered as a variant of an autoencoder, where the feature extractor is responsible for encoding the feature and the denoiser reconstructs the clean features. Previous literature [30, 32] show that adversarial examples usually lie outside the data manifold, and the autoencoder can place them on the manifold by learning the manifold structure. Thus, the feature extractor and denoiser can defend against adversarial examples attacks by removing perturbations.

Figure 4 details the denoising process. The feature extraction parameters are reused; the denoiser comprises four blocks and a 1×1 convolutional layer. To reduce loss of spatial information caused by downsampling, skip-connections [5] are introduced, and the feature maps recovered by upsampling contain additional low-level feature information provided by a fusion unit. This performs upsampling and a concatenation operation. The output of each denoiser block is upsampled using bilinear interpolation, and then feature concatenation is performed with the output of the corresponding block from feature extraction. Unlike the blocks in feature extraction, the conv units of all blocks in the denoiser use a stride of 1×1. In addition, we follow Refs. [15, 33] to implement residual learning instead of directly reconstructing a whole image, thus benefiting deep neural network training. The residual generated by the last 1×1 convolutional layer is added to the input x, which is converted into a clean image. Although the structure of our denoiser is similar to that in Ref. [15], some obvious differences exist. Our denoiser shares feature extraction with the detector, and the light convolution is used. We next consider the denoiser loss function.

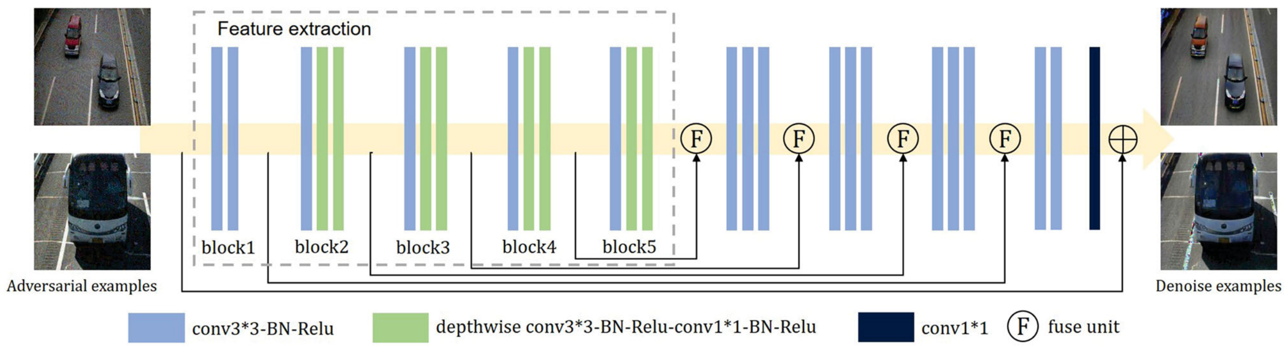

Fig. 4 Architecture of the adversarial perturbation denoiser. If the input x is an adversarial image, the denoiser aims to transform it into clean data. The conv unit and light conv unit are the same as in Fig. 3. The fusion unit performs bilinear interpolation and feature concatenation.

3.3 Network loss function

To detect and eliminate adversarial perturbations, our end-to-end defensive method includes a detector and a denoiser. When training the detector, each image belongs to only one of two categories: clean image or adversarial image. Therefore, we use a cross-entropy cost function to measure the difference between prediction and expectation.

$$L_{\text{dec}}\left(\{x_i\},\{y_i\}\right)=\frac{1}{N}\sum_{i}^{N} y_i \ln\left(\text{Dec}\left(E\left(x_i\right)\right)\right) + \left(1-y_i\right)\ln\left(1-\text{Dec}\left(E\left(x_i\right)\right)\right) \tag{5}$$

where i and N respectively represent the index and the number of examples in a mini-batch. x_i is the ith image, and y_i classifies x_i: if x_i is an adversarial example, $y_i = 1$; otherwise $y_i = 0$. $\text{Dec}(E(x_i))$ represents the predicted probability that x_i is an adversarial example.

The goal of training the denoiser is to make the difference between the recovered image and the clean image as small as possible. However, the remaining perturbations may influence the response of the target classification model. To overcome this problem, we combine pixel-level loss and high-level feature loss in the cost function used to supervise the training stage of the denoiser, unlike Ref. [15]. The cost function of the denoiser is

$$L_{\text{den}}\left(\{x_i\},\{\hat{x_i}\}\right)=\frac{1}{N}\sum_{i}^{N}\parallel \text{Den}\left(E\left(\hat{x_i}\right)\right)-x_i\parallel + \parallel f_l\left(\text{Den}\left(E\left(\hat{x_i}\right)\right)\right)-f_l\left(x_i\right)\parallel \tag{6}$$

where $\hat{x_i}$ denotes the adversarial sample of x_i, and $\text{Den}\left(E\left(\hat{x_i}\right)\right)$ is the recovered output of the denoiser. f_l denotes the response of the lth convolutional layer from the bottom of the target classification model and l is set to 1. The L_1 norm is used to calculate pixel-level loss and high-level feature loss.

Based on the above analysis, the cost function of DDAP can be formulated as

$$L\left(\{x_i\},\{\hat{x_i}\},\{y_i\}\right)=\alpha L_{\text{dec}}\left(\{x_i\},\{y_i\}\right) + \beta L_{\text{den}}\left(\{x_i\},\{\hat{x_i}\}\right) \tag{7}$$

where α, β are hyperparameters. For training stability, alternate training is adopted. First, the parameters for feature extraction and the denoiser are trained using clean images and adversarial examples until the network converges. Next, the feature extraction parameters are frozen, and the detector is trained until convergence. Finally, we adopt a small learning rate to fine-tune the detector and denoiser.

4 Evaluation

In this section, we validate the capabilities of the proposed DDAP method in the presence of various adversarial attacks, including FGSM [20], BIM [9], DeepFool [21], and PGD [22]. Then DDAP is compared with other advanced defense methods: LGD [15], SR [16], PD [17], and TPD [13] using the BIT-Vehicle dataset [34] and an online Public dataset `https://github.com/CNHNLP/public-dataset`. In addition, we also consider the performance on clean images and visualization of feature maps.

4.1 Setup

We use pre-trained Inception-v3 [3] as the target classifier. From the BIT-Vehicle dataset, we randomly selected 7880 images as the training set and 1970 images as the test set. For the Public dataset, we picked 1400 images as the training set and 200 images as the test set. The learning rate was set to 0.001 for the BIT-Vehicle classifier and 0.01 for the

Public classifier. Both were trained for 20 epochs with a batch size of 32, using the SGD optimization algorithm. The accuracy of the target classifier on the BIT-Vehicle and Public datasets was 97.1% and 98.0%, respectively.

To assess the proposed model, adversarial images are required for training and testing. We used several attack methods to construct these adversarial samples. The adversarial training set was generated using FGSM, BIM, and DeepFool, while the adversarial test set was generated using FGSM, BIM, DeepFool, and PGD. Adding PGD during testing aims to verify the robustness of our method against unmet adversarial attacks. The perturbation level of these attack methods was set to 0.15.

4.2 Implementation details

Table 1 gives the parameters used for training our method. Different experimental settings are used in different training phases to keep the optimization stable. While training the denoiser, hyperparameter α is set to 0. Conversely, when training the detector, the hyperparameter β is set to 0. SGD optimization was applied in the experiment, and an early stop strategy was also used in the training process. The proposed DDAP method was implemented with the Pytorch deep learning framework, and deployed on an NVIDIA TESLA P100 GPU.

4.3 Experimental results

4.3.1 BIT-Vehicle dataset

The BIT-Vehicle dataset targets vehicle classification. It contains 9850 vehicle images with differences in lighting, scale, vehicle color, and viewpoint. Vehicles are divided into 6 categories: Bus, Microbus, Minivan, Sedan, SUV, and Truck. Since generating adversarial examples for the entire dataset is expensive, we randomly selected 2000 correctly classified images from the training set of the target classifier as the training set, and use each adversarial attack method to generate adversarial examples, giving an adversarial training set with 6000 examples. Similarly, the adversarial test set provides 7880 samples based on the test set of the target classifier. We used the adversarial training set to supervise training of the denoiser. In addition, we also selected 1000 images from both the training set and adversarial training set to supervise the detector. The optimized model is compared with other advanced defense models on the adversarial test set.

Table 1 Training parameters for the proposed method.

	Detector	Denoiser	Joint training
α/β	1/0	0/1	1/1
Learning rate	10^{-4}	10^{-2}	10^{-8}
Batch size	8	8	4
Epochs	20	100	10

Table 2 reports the defensive performance of DDAP on the BIT-Vehicle dataset. The classification accuracy of the target classifier without a defensive model is significantly lowered, and for BIM and PGD attacks, it drops to zero. DDAP provides better accuracy for all kinds of attack than other defensive methods. Although PGD is not used for training, we still achieve 95.4% accuracy, reflecting the robustness of our defense method.

We also investigated the capabilities of these defense methods under black-box attack. In a black-box attack setting, the attacker has no knowledge of the target classifier. Therefore, another classifier ResNet-18 [2] was optimized on the BIT-Vehicle training set, and the adversarial test set was constructed using the ResNet-18 classifier and the same attack methods. As illustrated in Table 3, even though the classification accuracy of DDAP decreases slightly compared with a white-box attack, it still outperforms other methods. Obviously, LGD is sensitive to black-box attack, and its classification accuracy rate drops by nearly 30%. This may be caused by insufficient ability to learn manifold structure.

4.3.2 Public dataset

The Public dataset contains 10 vehicle categories: Bus, Family Sedan, Fire Engine, Heavy Truck, Jeep, Minibus, Racing Car, SUV, Taxi, and Truck. We extracted 1400 images as the training set and 200 images as the test set. An adversarial training set and adversarial test set were constructed as before.

Table 4 shows defensive performance of DDAP on Public dataset. SR and PD may fail to defend against FGSM attack because the classification accuracy of the target classifier is only 25% and 16%. On the contrary, both LGD and DDAP achieve acceptable defensive effect, and DDAP gets higher accuracy compared with LGD. DDAP supports the target classifier to obtain 96.0% accuracy under PGD attack, confirming the robustness of the proposed method under different attackers. It should be

Table 2 Classification accuracy on the BIT-Vehicle dataset obtained by different defensive methods

	No defense	LGD [15]	SR [16]	PD [17]	DDAP
FGSM [20]	58.7%	88.8%	59.3%	58.8%	**96.4%**
BIM [9]	0.0%	92.2%	65.2%	74.4%	**95.5%**
DeepFool [21]	23.0%	95.1%	95.4%	94.1%	**96.6%**
PGD [22]	0.0%	92.2%	74.8%	76.0%	**95.4%**

Table 3 Classification accuracy on the BIT-Vehicle dataset for different defensive methods, under black-box attack

	No defense	LGD [15]	SR [16]	PD [17]	DDAP
FGSM [20]	58.8%	59.0%	58.9%	58.8%	**82.1%**
BIM [9]	70.8%	60.3%	71.7%	73.9%	**88.0%**
DeepFool [21]	95.4%	64.3%	95.1%	93.4%	**97.1%**
PGD [22]	68.7%	60.1%	69.9%	71.3%	**88.1%**

Table 4 Classification accuracy using different defensive methods and the Public dataset

	No defense	LGD [15]	SR [16]	PD [17]	DDAP
FGSM [20]	4.0%	92.0%	25.0%	16.0%	**97.5%**
BIM [9]	0.0%	94.5%	66.0%	73.5%	**98.0%**
DeepFool [21]	8.0%	97.5%	97.5%	96.5%	**98.0%**
PGD [22]	0.0%	94.5%	70.5%	68.0%	**96.0%**

pointed out that compared with other attackers, DeepFool's effect on defensive models is limited, and the accuracy of the target classifier slightly decreases. Table 5 compares average forward time of these defensive methods, and DDAP spends less inference time due to lightweight convolution and parameter sharing of the feature extraction. Figure 5 exhibits some examples of defensive. The target classification is mispredicted by adversarial images, which is added

Table 5 Average inference time on test images for different defense methods

LGD [15]	SR [16]	PD [17]	DDAP
0.066 s	0.50 s	0.515 s	**0.056 s**

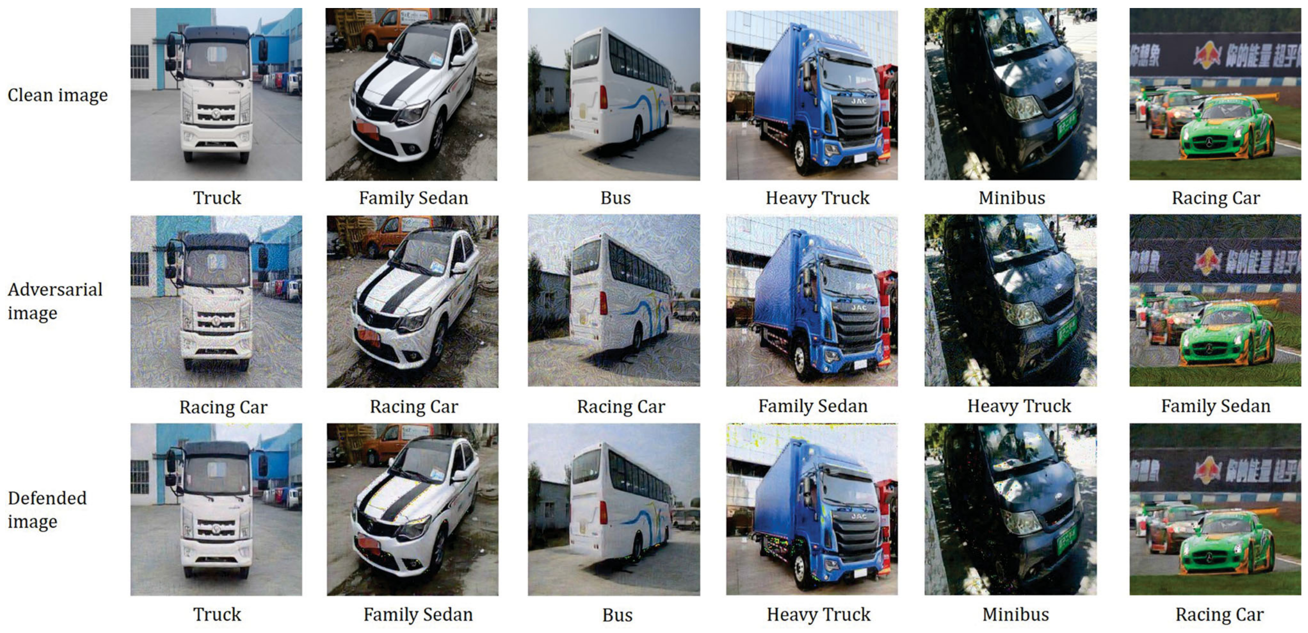

Fig. 5 Selected examples produced by DDAP. Labels below each image indicate the output classification.

with imperceptible perturbations. After removing perturbations, there is almost no difference between defended images and clean images, which enhances the correct prediction of the target classifier.

As the first step of the network, it is critical that the detector correctly detects adversarial examples. We compare our method with TPD for recognition accuracy on the Public adversarial test set. In TPD, we assume trained Inception-v3, and Resnet-18 with softmax normalization, as defensive model and target model. Table 6 summarizes the recognition performance of TPD and DDAP. In order to balance classification accuracy of TPD for both clean and adversarial examples, thresholds of 0.05 and 0.1 were considered. The recognition accuracy achieved by TDP is 73.9% and 71.2% for these different thresholds, which is far short of the 99.1% achieved by DDAP.

4.3.3 Performance on clean images

As mentioned in the introduction, using defensive models may reduce the accuracy of the target classifier when presented with clean images. Table 7 provides target classifier results for various defensive methods on the BIT-Vehicle and Public test sets. Using LGD greatly reduces the classification accuracy from 79.5% to 65.4%. DDAP hardly affects the performance of the classifier on clean images, reducing it to 97.1% from 97.5%. This is also better than the results obtained by SR and PD.

4.3.4 Ablation study for DDAP

In order to verify the effectiveness of the design, we compared the performance of DDAP with or without the detector on two vehicle datasets. The test set includes clean images and the adversarial test set. Table 8 shows that DDAP with the detector achieves better classification accuracy than without, indicating the benefit of adding the detector.

4.3.5 DUNET

If we remove the detector from DDAP, it basically degenerates to DUNET [15]. Ideally, if we combine clean examples and adversarial examples to train DUNET, it may learn the ability to recognizing clean images and adversarial examples, and denoise adversarial perturbations at the same time. When a clean example is entered, it outputs the original clean example. When an adversarial example is input, a repaired example is output. Therefore, we combined the adversarial training set and the training set to form a joint training set, using BIT-Vehicle. The test set and adversarial samples generated by attack methods were employed to construct the joint test set. The joint training set was used to train both DDAP and DUNET. Table 9 gives the classification accuracy on the joint test set. DDAP achieves better results than DUNET for all kinds of attack, which may be due to it being difficult for DUNET to learn to recognize and reduce perturbations at the same time.

4.3.6 Further validation on the CompCars dataset

In order to further validate the effectiveness of the design, various defense methods were also tested in the web-nature scenario of the CompCars dataset [35], which includes MPV, SUV, Sedan, Hatchback, Minibus, Fastback, Estate, Pickup, Hardtop Convertible, Sports, Crossover, Convertible. Since it is very expensive to generate adversarial samples for the entire dataset, we randomly constructed 8640 adversarial training set examples and 2880 adversarial test set examples. Table 10 gives the results. As before, these methods effectively defend against DeepFool. DDAP achieves the best classification accuracy for all attacks.

Table 6 Accuracy of recognition of adversarial examples from the Public dataset

TPD [13] (t=0.05)	TPD (t=0.1)	DDAP
73.9%	71.2%	**99.1%**

Table 7 Classification accuracy of clean images using different defense methods

	No defense	LGD [15]	SR [16]	PD [17]	DDAP
BIT-Vehicle [34]	97.1%	65.4%	95.5%	94.4%	**97.1%**
Public dataset	98.0%	79.5%	96.0%	95.5%	**97.5%**

Table 8 Classification performance of DDAP with and without the detector, for two datasets

	DDAP without detector	DDAP
BIT-Vehicle [34]	92.3%	**96.2%**
Public dataset	88.1%	**97.1%**

Table 9 Classification accuracy comparison with DUNET on BIT-Vehicle

	FGSM [20]	BIM [9]	DeepFool [21]
DUNET [15]	87.1%	86.6%	87.2%
DDAP	**96.7%**	**96.3%**	**96.6%**

Table 10 Classification accuracy for different defensive methods using the CompCars dataset

	No defensive	LGD [15]	SR [16]	PD [17]	DDAP
FGSM [20]	75.4%	97.5%	66.0%	68.1%	**99.7%**
BIM [9]	21.4%	98.3%	21.9%	22.2%	**99.4%**
DeepFool [21]	2.4%	99.9%	99.4%	99.4%	**99.9%**
PGD [22]	13.9%	98.2%	14.6%	14.7%	**99.7%**

4.3.7 CAMs visualization

CAMs [36] is a method to help interpret convolutional neural networks: it can visualize the discriminative features they learn. Usually, redder regions are more sensitive to the target classifier. Figure 6 shows the class activation mapping for the top-1 prediction of the Inception-v3 classifier on the BIT-Vehicle data. The target classifier focuses on the vehicle region in clean images. The perturbations generated by FGSM take attention from these discriminative features, resulting in incorrect classification. We can see that DDAP can refocus on the discriminative region by removing the perturbations.

5 Conclusions

In this paper, we have presented an end-to-end approach, DDAP, to defend against adversarial attacks in vehicle classification. It detects adversarial examples and eliminates adversarial perturbations without changing the structure of the vehicle classifier. As our experiments demonstrate, DDAP resists a variety of powerful adversarial attacks and is robust in both white-box and black-box attacks. The DDAP model only slightly decreases the performance of the vehicle classifier on clean images, and outperforms other state-of-the-art defensive methods on available vehicle datasets.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (61872047, 61720106007), the National Key R&D Program of China (2017YFB1003000), the Beijing Nova Program (Z201100006820124), the Beijing Natural Science Foundation (L191004), and the 111 Project (B18008).

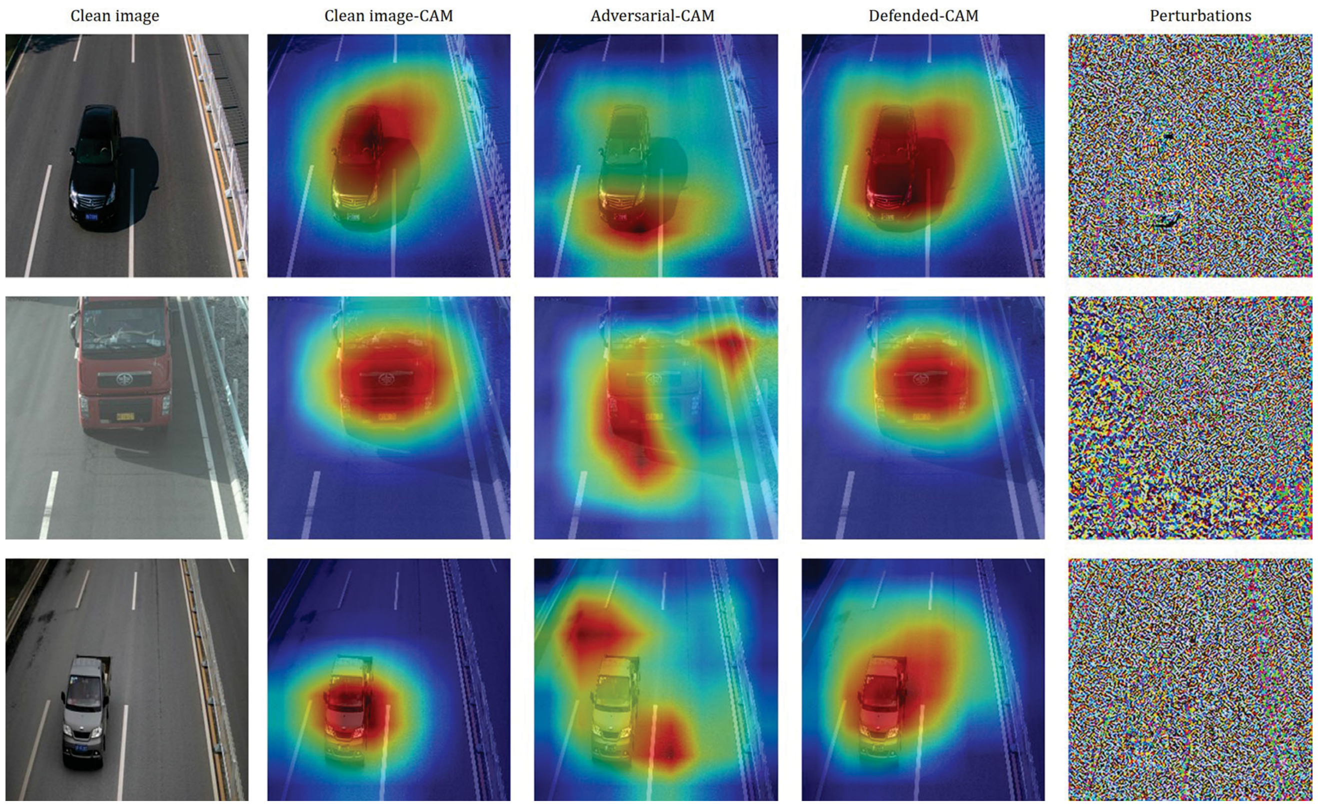

Fig. 6 Class activation map visualization of FGSM attacks on BIT-Vehicle data. CAMs before and after adversarial and defensive perturbations are displayed.

References

[1] Fu, H. Y.; Ma, H. D.; Wang, G. Y.; Zhang, X. M.; Zhang, Y. F. MCFF-CNN: Multiscale comprehensive feature fusion convolutional neural network for vehicle color recognition based on residual learning. *Neurocomputing* Vol. 395, 178–187, 2020.

[2] He, K. M.; Zhang, X. Y.; Ren, S. Q.; Sun, J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778, 2016.

[3] Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2818–2826, 2016.

[4] Oh, M.; Cha, B.; Bae, I.; Choi, G.; Lim, Y. An urban autodriving algorithm based on a sensor-weighted integration field with deep learning. *Electronics* Vol. 9, No. 1, 158, 2020.

[5] Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional networks for biomedical image segmentation. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. Lecture Notes in Computer Science, Vol. 9351.* Navab, N.; Hornegger, J.; Wells, W.; Frangi, A. Eds. Springer Cham, 234–241, 2015.

[6] Liu, X. C.; Liu, W.; Ma, H. D.; Fu, H. Y. Large-scale vehicle re-identification in urban surveillance videos. In: Proceedings of the IEEE International Conference on Multimedia and Expo, 1–6, 2016.

[7] Zhuo, L.; Jiang, L. Y.; Zhu, Z. Q.; Li, J. F.; Zhang, J.; Long, H. X. Vehicle classification for large-scale traffic surveillance videos using Convolutional Neural Networks. *Machine Vision and Applications* Vol. 28, No. 7, 793–802, 2017.

[8] Won, M. Intelligent traffic monitoring systems for vehicle classification: A survey. *IEEE Access* Vol. 8, 73340–73358, 2020.

[9] Kurakin, A.; Goodfellow, I.; Bengio, S. Adversarial examples in the physical world. *arXiv preprint* arXiv:1607.02533, 2016.

[10] Liu, Y.; Chen, X.; Liu, C.; Song, D. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint* arXiv:1611.02770, 2016.

[11] Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z. B.; Swami, A. The limitations of deep learning in adversarial settings. In: Proceedings of the IEEE European Symposiumon Security and Privacy, 372–387, 2016.

[12] Carrara, F.; Falchi, F.; Caldelli, R.; Amato, G.; Fumarola, R.; Becarelli, R. Detecting adversarial example attacks to deep neural networks. In: Proceedings of the 15th International Workshop on Content-based Multimedia Indexing, Article No. 38, 2017.

[13] Guo, F.; Zhao, Q. J.; Li, X.; Kuang, X. H.; Zhang, J. W.; Han, Y. H.; Tan, Y.-a. Detecting adversarial examples via prediction difference for deep neural networks. *Information Sciences* Vol. 501, 182–192, 2019.

[14] Rakin, A. S.; Fan, D. L. Defense-net: Defend against a wide range of adversarial attacks through adversarial detector. In: Proceedings of the IEEE Computer Society Annual Symposium on VLSI, 332–337, 2019.

[15] Liao, F. Z.; Liang, M.; Dong, Y. P.; Pang, T.; Hu, X. L.; Zhu, J. Defense against adversarial attacks using high-level representation guided denoiser. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1778–1787, 2018.

[16] Mustafa, A.; Khan, S. H.; Hayat, M.; Shen, J. B.; Shao, L. Image super-resolution as a defense against adversarial attacks. *IEEE Transactions on Image Processing* Vol. 29, 1711–1724, 2020.

[17] Prakash, A.; Moran, N.; Garber, S.; DiLillo, A.; Storer, J. Detecting adversarial attacks with pixel detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 8571–8580, 2018.

[18] Xie, C. H.; Wu, Y. X.; van der Maaten, L.; Yuille, A. L.; He, K. M. Feature denoising for improving adversarial robustness. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 501–509, 2019.

[19] Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; Fergus, R. Intriguing properties of neural networks. *arXiv preprint* arXiv:1312.6199, 2013.

[20] Goodfellow, I. J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint* arXiv:1412.6572, 2014.

[21] Moosavi-Dezfooli, S. M.; Fawzi, A.; Frossard, P. DeepFool: A simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2574–2582, 2016.

[22] Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint* arXiv:1706.06083, 2017.

[23] Metzen, J. H.; Genewein, T.; Fischer, V.; Bischofi, B. On detecting adversarial perturbations. *arXiv preprint* arXiv:1702.04267, 2017.

[24] Feinman, R.; Curtin, R. R.; Shintre, S.; Gardner, A. B. Detecting adversarial samples from artifacts. *arXiv preprint* arXiv:1703.00410, 2017.

[25] Liang, B.; Li, H. C.; Su, M. Q.; Li, X. R.; Shi, W. C.; Wang, X. F. Detecting adversarial image examples in deep neural networks with adaptive noise

reduction. *IEEE Transactions on Dependable and Secure Computing* Vol. 18, No. 1, 72–85, 2019.

[26] Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In: Proceedings of the IEEE Symposium on Security and Privacy, 582–597, 2016.

[27] Carlini, N.; Wagner, D. Towards evaluating the robustness of neural networks. In: Proceedings of the IEEE Symposium on Security and Privacy, 39–57, 2017.

[28] Samangouei, P.; Kabkab, M.; Chellappa, R. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint* arXiv:1805.06605, 2018.

[29] LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* Vol. 86, No. 11, 2278–2324, 1998.

[30] Santhanam G. K.; Grnarova, P. Defending against adversarial attacks by leveraging an entire GAN. *arXiv preprint* arXiv:1805.10652, 2018.

[31] Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint* arXiv:1704.04861, 2017.

[32] Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P. A. Extracting and composing robust features with denoising auto encoders. In: Proceedings of the 25th International Conference on Machine Learning, 1096–1103, 2008.

[33] Zhang, K.; Zuo, W. M.; Chen, Y. J.; Meng, D. Y.; Zhang, L. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing* Vol. 26, No. 7, 3142–3155, 2017.

[34] Dong, Z.; Pei, M. T.; He, Y.; Liu, T.; Dong, Y. M.; Jia, Y. D. Vehicle type classification using unsupervised convolutional neural network. In: Proceedings of the 22nd International Conference on Pattern Recognition, 172–177, 2014.

[35] Yang, L. J.; Luo, P.; Loy, C. C.; Tang, X. O. A large-scale car dataset for fine-grained categorization and verification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3973–3981, 2015.

[36] Zhou, B. L.; Khosla, A.; Lapedriza, A.; Oliva, A.; Torralba, A. Learning deep features for discriminative localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2921–2929, 2016.

Peng Liu is a master degree student at the School of Computer Science, Beijing University of Posts and Telecommunications. His research interests include adversarial defense and semantic segmentation.

Huiyuan Fu received his Ph.D. degree in computer science from Beijing University of Posts and Telecommunications in 2014. He is an associate professor at the School of Computer Science, Beijing University of Posts and Telecommunications. His research area includes visual big data, machine learning and pattern recognition, multimedia systems, etc. He received the Best Student Paper Award at ICME in 2016.

Huadong Ma received his Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Science (CAS), in 1995, M.S. degree in computer science from Shenyang Institute of Computing Technology, CAS, in 1990, and B.S. degree in mathematics from Henan Normal University, China, in 1984. He is a professor at the School of Computer Science, Beijing University of Posts and Telecommunications. His research interests include multimedia networks and systems, the internet of things, and sensor networks. He has published over 300 papers in these fields.