GENERAL ADVERSARIAL DEFENSE VIA PIXEL LEVEL AND FEATURE LEVEL DISTRIBUTION ALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep neural networks (DNNs) have achieved amazing success on a wide range of high-level computer vision tasks. However, it is proved that DNNs are vulnerable to adversarial samples. The threat of adversarial samples comes from the large distribution gap between adversarial samples and clean samples in the feature spaces of the target DNNs. To this, we utilize deep generative networks with a novel training scheme to eliminate the distribution gap. Our training strategy introduces constraints in both pixel level as well as feature level, and the trained network can effectively align the distribution of adversarial samples with clean samples for target DNNs through translating their pixel values. Specifically, compared with previous methods, we propose a more efficient pixel-level training constraint to weaken the hardness of aligning adversarial samples to clean samples, which can thus obviously enhance the robustness on adversarial samples. Besides, a class-aware feature-level constraint is formulated for integrated distribution alignment. Our approach is general and suitable for multiple tasks like image classification, semantic segmentation and object detection. We conduct extensive experiments on these three tasks and different datasets, on which the superiority of our strategy over existing methods demonstrates its effectiveness and generality.

1 INTRODUCTION

Deep neural networks (DNNs) have shown impressive performance on many computer vision tasks, e.g., image classification (He et al., 2016), semantic segmentation (Zhao et al., 2017) and object detection (Liu et al., 2016a). However, recently many works have revealed that deep learning models are commonly vulnerable to adversarial examples (Szegedy et al., 2013; Goodfellow et al., 2014b; Arnab et al., 2018; Xie et al., 2017b), which are maliciously generated to fool the target model by applying adversarial perturbations on original clean inputs. Such perturbations are imperceptible to the human vision system, while cause a great threat to practical deep learning applications, such as face recognition (Dong et al., 2019; Joshi et al., 2019), autonomous driving (Jia et al., 2019; Kong et al., 2020), etc.

An efficient method to eliminate adversarial perturbations is applying input transformation to the input samples before they are processed by the target DNNs. One primitive strategy is adopting traditional image processing (e.g., image compression (Guo et al., 2017)) to implement such transformation. However, many researchers have noticed the limitations of such strategy, and they reveal the effectiveness of deep generative networks (DGNs) to accomplish such transformation: train a network with adversarial/clean samples as inputs and synthesize outputs on which the target model works well. The essence of these methods is weakening the differences between adversarial samples and the corresponding clean samples, through translating their pixel values with the network. Current approaches to train such DGNs can be divided into three categories: 1) set pixel-level constraints for the reduction of the distance between adversarial and clean samples (Shen et al., 2017; Mustafa et al., 2019); 2) adopt the constraints in the feature level of target models (Liao et al., 2018); 3) employ the constraints in pixel level as well as feature level (Naseer et al., 2020; Li et al., 2020). Nevertheless, all of them ignore the overall distribution alignment in the feature spaces of target models. Thus, there still exist spaces to improve their defense quality.

In this paper, we train DGNs that defend for target models via aligning the distribution of clean and adversarial samples in their feature spaces. Compared with existing methods, novel training con-

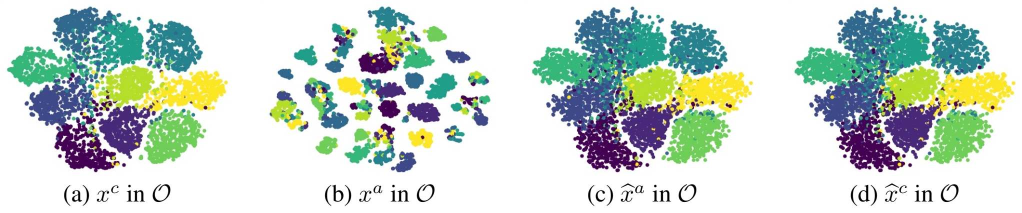


Figure 1: t-SNE visualizations in feature space of target model on CIFAR10 (Krizhevsky et al., 2009). Target model $\mathcal{O}$ has admirable distributions for clean samples x^c while disordered distributions for adversarial samples x^a. Our $\mathcal{G}$ can turn x^c/x^a into $\widehat{x}^c$/$\widehat{x}^a$ with corrected distrubutions.

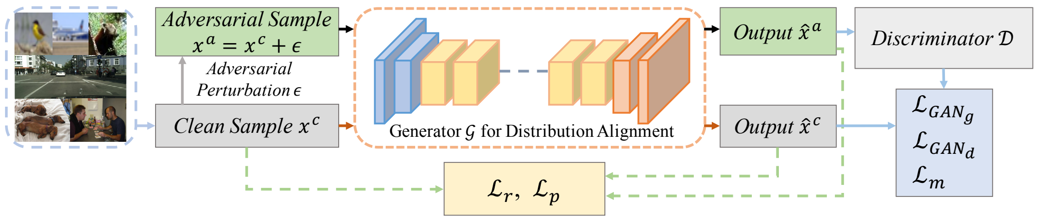
straints are set in both pixel level and feature level. In pixel level, we propose to match adversarial samples with the clean samples in the output spaces of the DGNs, and this boosts the alignment in feature level indirectly. In feature level, we design a class-aware constraint by aligning the central feature of clean and adversarial samples within each class, as well as maximizing the inter-class distance and minimizing the intra-class distance for all categories. Our strategy is suitable for numerous high-level computer vision tasks and massive experiments are conducted on classification, semantic segmentation and object detection as examples. Further, these experiments are executed with diverse datasets, models and attacks. DGNs trained with our approach can excellently align the distribution of adversarial samples to clean samples in the feature space of the target model as exhibited in Fig. 1. In summary, our contributions include:

- We are the first to train DGNs to achieve adversarial defense by viewing this task as integrated distribution alignment for clean and adversarial samples.
- We propose novel training constraints in both pixel level and feature level of target models, and the trained DGNs can defend against miscellaneous adversarial samples.
- Extensive experiments are conducted on various tasks and datasets, on which the superiority of our scheme over existing methods demonstrates its effectiveness and generality.

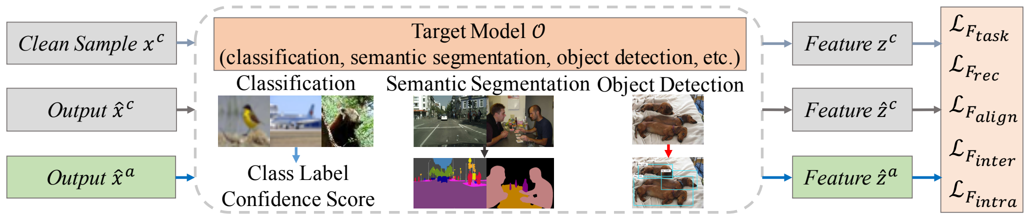
2 Related Work

Adversarial Attack Adversarial attack can be divided into two categories: white-box attack (Athalye & Carlini, 2018; Goodfellow et al., 2014b), where attackers have complete knowledge of the target model; black-box attack (Papernot et al., 2017), where attackers have almost no knowledge of the target model. Existing attacks focus on the classification task and they are normally achieved by computing or simulating gradient informations of target models (Goodfellow et al., 2014b; Kurakin et al., 2016). Meanwhile, as indicated by recent papers, semantic segmentation networks (Xie et al., 2017b; Metzen et al., 2017; Arnab et al., 2018) and object detection networks (Xie et al., 2017b; Li et al., 2018; Song et al., 2018) are also vulnerable to adversarial attacks.

Adversarial Defense To eliminate the threat of adversarial perturbations, there have been several kinds of strategies for defenses and a major class of defenses processes the input images with input transformations to achieve robustness (Guo et al., 2017; Xie et al., 2017a). Such approaches translate the pixel values of adversarial/clean samples to remove the influence of adversarial perturbations, and almost all existing methods treat such transformation as the task of denoise. Moreover, recently, some works have noticed the efficacy of input transformations which are implemented through DGNs. Current strategies that employ DGNs can be divided into three categories, according to the constraints they utilized in the training. 1) Using pixel-level constraints to reduce the differences of pixel values between clean and adversarial samples. (Shen et al., 2017; Mustafa et al., 2019; Samangouei et al., 2018; Prakash et al., 2018; Song et al., 2017); 2) applying feature-level constraints to unify representations of clean and adversarial samples in the feature space of the target model (Liao et al., 2018); 3) simultaneously setting pixel-level and feature-level constraints, which is proved to be more advantageous (Naseer et al., 2020; Li et al., 2020). Especially, in feature level, existing approaches only set the distance between clean and adversarial samples as the constraint to optimize, while have not considered the alignment of integrated distribution in the feature space.


Adversarial Sample $x^a = x^c + \epsilon$
Adversarial Perturbation ϵ
Clean Sample x^c
Generator $\mathcal{G}$ for Distribution Alignment
Output $\hat{x}^a$
Output $\hat{x}^c$
Discriminator $\mathcal{D}$
$\mathcal{L}_{GAN_g}$
$\mathcal{L}_{GAN_d}$
$\mathcal{L}_m$
$\mathcal{L}_r$, $\mathcal{L}_p$

(a) Pixel level training constraints



(b) Feature level training constraints

Figure 2: Our overall framework for the training of the deep generative network $\mathcal{G}$. To align the distribution of clean samples x^c and adversarial samples x^a for the target model, the training constraints are set in the pixel level as well as the feature level of the target model $\mathcal{O}$.

3 Method

Generally speaking, we can train a network $\mathcal{O}$ for one target task $\mathcal{S}$, e.g., image classification, semantic segmentation and object detection. The network $\mathcal{O}$ is usually trained with clean samples x^c and we suppose $x^c \sim \mathcal{C}$, where $\mathcal{C}$ is the distribution of clean samples. Normally, the trained network $\mathcal{O}$ behaves brilliantly on $x^c \sim \mathcal{C}$ for the task $\mathcal{S}$, while its performance will be remarkably decreased after adding adversarial perturbations ϵ to x^c. Specially, adversarial samples are denoted as x^a ($x^a = x^c + \epsilon$), and we represent the distribution of adversarial samples as $\mathcal{A}$. As shown in Fig. 1 (a) and (b), although the diversity of pixel values between x^c and x^a is imperceptible, there is a large gap between $\mathcal{C}$ and $\mathcal{A}$ in the feature space of the target model $\mathcal{O}$ which leads to the terrible performance of $\mathcal{O}$ on adversarial samples. Therefore, to eliminate the negative threat of adversarial samples, it is rational to align $\mathcal{C}$ and $\mathcal{A}$ for a target model $\mathcal{O}$ as exhibited in Fig. 1 (c) and (d). Based on this observation and motivation, we propose to train a deep generative network $\mathcal{G}$ which could align $\mathcal{C}$ and $\mathcal{A}$ in the feature space of the target model via modifying the pixel values of x^c and x^a. As a result, $\mathcal{G}$ can translate x^c/x^a into $\widehat{x}^c/\widehat{x}^a$, and $\mathcal{O}$ achieves great results on $\widehat{x}^c/\widehat{x}^a$. To train $\mathcal{G}$, we set the constraints for alignment in both pixel level and feature level as shown in Fig. 2, and push the distribution of $\widehat{x}^c/\widehat{x}^a$ to approach $\mathcal{C}$ since $\mathcal{O}$ performs well on $x^c \sim \mathcal{C}$.

3.1 Pixel-level Alignment

The pixel-level training constraints are employed to weaken the difference in pixel values between clean and adversarial samples (specially, the pixel values refer to the RGB space in this paper), and can help the alignment in the feature space of the target model $\mathcal{O}$ indirectly. Previous works have validated that "distance metric between clean and adversarial samples" and "adversarial learning" (Goodfellow et al., 2014a) are two effective pixel-level training constraints. And traditional pixel-level constraints (Shen et al., 2017; Mustafa et al., 2019; Samangouei et al., 2018; Prakash et al., 2018; Song et al., 2017) mainly adopt x^c to guide the formulation of $\widehat{x}^c$ and $\widehat{x}^a$ through the generator $\mathcal{G}$, as displayed in Fig. 3 (a). i.e., they compute the distance metric between $\widehat{x}^c/\widehat{x}^a$ and x^c, and set x^c as real samples, $\widehat{x}^c/\widehat{x}^a$ as fake samples to conduct adversarial learning. However, although such setting could result in admirable performance on clean samples for $\mathcal{O}$, it might lead to outcomes on adversarial samples which are not satisfactory enough. This primarily dues to the ill-posed discrepancy between $\mathcal{C}$ and $\mathcal{A}$ that causes the hardness of directly matching $\widehat{x}^a$ to x^c.

To this, we propose a novel scheme for pixel-level training constraints, where we use x^c to guide the formulation of $\widehat{x}^c$ and utilize $\widehat{x}^c$ to help the match of x^a, as shown in Fig. 3 (b). In this setting, $\widehat{x}^c$ would actually act as an intermediate to shorten the discrepancy between $\widehat{x}^a$ and x^c more efficiently. Comprehensive experiments illustrate that our novel setting can results in conspicuous improvements for the performances on adversarial samples, compared to the traditional scheme.

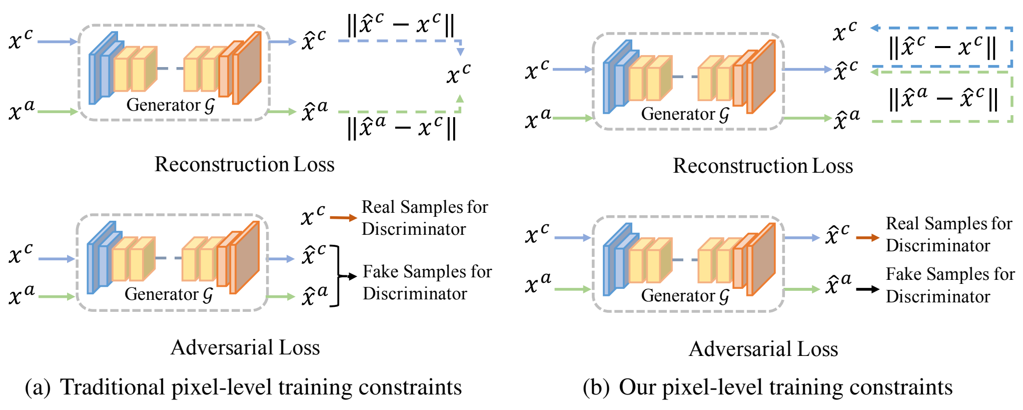

x^c
x^a
Generator $\mathcal{G}$
$\hat{x}^c$
$\hat{x}^a$
$\|\hat{x}^c - x^c\|$
$\|\hat{x}^a - x^c\|$
Reconstruction Loss
Real Samples for Discriminator
Fake Samples for Discriminator
Adversarial Loss
(a) Traditional pixel-level training constraints
$\|\hat{x}^a - \hat{x}^c\|$
(b) Our pixel-level training constraints

Figure 3: The illustration for differences between traditional and proposed pixel-level constraints.

Specifically speaking, our pixel-level training constraints also include the distance metric as well as adversarial learning. Given clean samples x^c, adversarial samples x^a are obtained by applying adversarial perturbations ϵ on x^c, and the generator $\mathcal{G}$ synthesizes the outputs $\widehat{x}^c$/$\widehat{x}^a$ with the inputs of x^c/x^a. Based on such inputs and outputs, we can define a reconstruction loss term $\mathcal{L}_r$ as

$$\widehat{x}^a = \mathcal{G}(x^a),\ \widehat{x}^c = \mathcal{G}(x^c),\ \mathcal{L}_r = \mathbb{E}(\|\widehat{x}^c - x^c\|) + \mathbb{E}(\|\widehat{x}^a - \widehat{x}^c\|), \tag{1}$$

where $\mathbb{E}$ is the operation to compute the mean value. Moreover, to help synthesize images with high resolution and complex texture, we also compute a perceptual loss term $\mathcal{L}_p$ as

$$\mathcal{L}_p = \sum_{i=1}^{5} \mathbb{E}(||\Phi_i(\widehat{x}^c) - \Phi_i(x^c)||) + \sum_{i=1}^{5} \mathbb{E}(||\Phi_i(\widehat{x}^a) - \Phi_i(\widehat{x}^c)||), \tag{2}$$

where $\Phi_1(\cdot)$ to $\Phi_5(\cdot)$ are outputs from the ReLU1_2, ReLU2_2, ReLU3_3, ReLU4_3, ReLU5_3 layers of an ImageNet-pretrained VGG-16 network (Simonyan & Zisserman, 2014). Further, we need the training constraint to match the global pattern for $\mathcal{C}$ and $\mathcal{A}$ in pixel level, which can be implemented as loss terms in adversarial learning with sampling. Therefore, we employ a discriminator $\mathcal{D}$, and the loss terms are set as the form of LSGAN (Mao et al., 2017):

$$\mathcal{L}_{GAN_d} = \mathbb{E}_{x^c\sim\mathcal{C}}((\mathcal{D}(\widehat{x}^c) - 1)^2) + \mathbb{E}_{x^a\sim\mathcal{A}}((\mathcal{D}(\widehat{x}^a) - 0)^2),\ \mathcal{L}_{GAN_g} = \mathbb{E}_{x^a\sim\mathcal{A}}((\mathcal{D}(\widehat{x}^a) - 1)^2), \tag{3}$$

where $\mathcal{L}_{GAN_d}$ is set for the discriminator, and $\mathcal{L}_{GAN_g}$ is adopted for the generator. Additionally, the feature match loss is adopted as an auxiliary part of the adversarial loss (Wang et al., 2018): we obtain intermediate features from $\mathcal{D}$ for fake and real samples, and compute their distances as

$$\mathcal{L}_m = \mathbb{E}(\|\mathcal{F}(\widehat{x}^c) - \mathcal{F}(\widehat{x}^a)\|), \tag{4}$$

where $\mathcal{F}(x)$ are the intermediate features obtained from the discriminator for real/fake samples x.

3.2 Feature-level Alignment

Besides the pixel-level training constraints, recent works have revealed the necessity of feature-level training constraints for the target model $\mathcal{O}$ (Liao et al., 2018; Naseer et al., 2020) to enhance defense effects. Existing methods formulate feature-level training constraints as the distances in the feature space of $\mathcal{O}$ between synthesized samples and clean samples to optimize. However, they ignore the constraint for the alignment of overall $\mathcal{C}$ and $\mathcal{A}$ in the feature level. To this, we propose a novel class-aware feature-level training constraint, besides the distance metric in the feature space of $\mathcal{O}$. Our class-aware constraint aims to align the integrated distribution of clean and adversarial samples within each category for the target model, and minimizing the intra-class distance as well as maximizing the inter-class distance for the distribution of adversarial samples.

Suppose $\mathcal{O}$ is trained with the paired data as (x^c, y^c) where y^c is the ground truths for x^c. And the loss term to train $\mathcal{O}$ can be represented as $\mathcal{L}_o(x^c, y^c)$. To align behaviors of clean samples and adversarial samples in the feature space of $\mathcal{O}$, we adopt a feature-level loss term as

$$\mathcal{L}_{F_{task}} = \mathcal{L}_o(\widehat{x}^c, y^c) + \mathcal{L}_o(\widehat{x}^a, y^c). \tag{5}$$

Algorithm 1 Our strategy to train the generator $\mathcal{G}$ for the target model $\mathcal{O}$ on the target task $\mathcal{S}$

Parameter: Training data (x^c, y^c), initialized generator $\mathcal{G}$ and discriminator $\mathcal{D}$

```
1: while not converged do
2:     Read a minibatch of data $D_b = \{x_1^c, ..., x_b^c\}$, $Y_b = \{y_1^c, ..., y_b^c\}$.
3:     Use the chosen attack algorithm and $\mathcal{O}$ to generate adversarial examples $A_b = \{x_1^a, ..., x_b^a\}$.
4:     Compute $\mathcal{L}_r$, $\mathcal{L}_p$, $\mathcal{L}_{GAN_g}$ and $\mathcal{L}_m$, by using $D_b$, $A_b$ and the discriminator $\mathcal{D}$.
5:     Forward $D_b$ and $A_b$ through $\mathcal{O}$ to obtain features of $K$ classes, according to $Y_b$.
6:     Compute the clustering center of each class according to the obtained features in this batch.
7:     Compute $\mathcal{L}_{F_{class}}$ by using features of $K$ classes and the corresponding clustering centers.
8:     Compute $\mathcal{L}_g$ to update the generator $\mathcal{G}$, compute $\mathcal{L}_{GAN_d}$ to update the discriminator $\mathcal{D}$.
9: end while
```

Meanwhile, similar to the reconstruction loss in pixel level, we minimize the distances between adversarial samples and clean samples in the feature space of $\mathcal{O}$ and set a loss term $\mathcal{L}_{F_{rec}}$ as:

$$\widehat{z}^a = \mathcal{O}(\widehat{x}^a),\ \widehat{z}^c = \mathcal{O}(\widehat{x}^c),\ z^c = \mathcal{O}(x^c),\ \mathcal{L}_{F_{rec}} = \mathbb{E}(\|\widehat{z}^c - z^c\|) + \mathbb{E}(\|\widehat{z}^a - z^c\|), \tag{6}$$

where $\widehat{z}^a$, $\widehat{z}^c$ and z^c are the intermediate features of $\widehat{x}^a$, $\widehat{x}^c$ and x^c in the feature space of $\mathcal{O}$.

On the other hand, previous methods have not considered aligning the overall distribution of $\mathcal{C}$ and $\mathcal{A}$ in the feature space of $\mathcal{O}$, which is however beneficial to the defense quality of $\mathcal{G}$. To this, we formulate our class-aware training constraint for the overall distribution alignment. Suppose there are K classes within the distribution of $\mathcal{C}$ and $\mathcal{A}$, and we represent $z^{c(k)}$ and $\widehat{z}^{a(k)}$ as the features for x^c and $\widehat{x}^a$ of k-th class that are extracted from $\mathcal{O}$. Moreover, the clustering centers of $z^{c(k)}$ and $\widehat{z}^{a(k)}$ can be denoted as $m^{c(k)}$ and $\widehat{m}^{a(k)}$, in the distribution of $\mathcal{C}$ and $\mathcal{A}$ respectively. The class-aware constraint consists of three terms. Firstly, we compute a loss term as the distance between $m^{c(k)}$ and $\widehat{m}^{a(k)}$ to align the distribution of adversarial samples to clean samples:

$$\mathcal{L}_{F_{align}} = \sum_{k=1:K} \mathbb{E}(\|m^{c(k)} - \widehat{m}^{a(k)}\|). \tag{7}$$

Further, favourable distributions in the feature spaces of the target models for high-level tasks should have wide distance between the features from different classes, and narrow separation among the features from the same class. Thus we propose to set intra-class and inter-class loss as

$$\mathcal{L}_{F_{intra}} = \sum_{k=1:K} \mathbb{E}(\|\widehat{z}^{a(k)} - \widehat{m}^{a(k)}\|),\ \mathcal{L}_{F_{inter}} = \sum_{k=1:K} \sum_{i=1:K, i\neq k} \mathbb{E}(M - \|\widehat{m}^{a(k)} - \widehat{m}^{a(i)}\|), \tag{8}$$

where M is a pre-defined hyper-parameter for controlling the inter-class distance. And the overall class-aware training constraint can be written as

$$\mathcal{L}_{F_{class}} = \lambda_1 \mathcal{L}_{F_{align}} + \lambda_2 \mathcal{L}_{F_{inter}} + \lambda_3 \mathcal{L}_{F_{intra}}, \tag{9}$$

where λ_1, λ_2, λ_3 are loss weights. Furthermore, the overall training constraint for the generator $\mathcal{G}$ can be formulated as

$$\mathcal{L}_g = \lambda_4 \mathcal{L}_r + \lambda_5 \mathcal{L}_p + \lambda_6 \mathcal{L}_{GAN_g} + \lambda_7 \mathcal{L}_m + \lambda_8 \mathcal{L}_{F_{task}} + \lambda_9 \mathcal{L}_{F_{rec}} + \mathcal{L}_{F_{class}}, \tag{10}$$

where λ_4 to λ_9 are weights of loss terms. Our training algorithm for $\mathcal{G}$ is summarized in Alg. 1.

4 EXPERIMENTS

Our framework is applicative for the defense of numerous tasks. In the following, we conduct adequate experimental evaluations on image classification, semantic segmentation and object detection, acrossing different datasets.

4.1 DATASETS

To show the performance of our method on our chosen three tasks, we select representative datasets in each task. For image classification, we choose CIFAR10 (Krizhevsky et al., 2009), CIFAR100 (Krizhevsky et al., 2009) and ImageNet (Deng et al., 2009); in the semantic segmentation task, we employ Cityscapes (Cordts et al., 2016) and VOC2012 (Everingham et al., 2012); for the experiments of object detection, VOC07+12 (Everingham et al., 2012) setting is adopted.

Table 1: Results of ablation study on the classification task.

	CIFAR10 (Accuracy %)				CIFAR100 (Accuracy %)				ImageNet (Accuracy %)			
	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W
No Defense	**95.1**	0.0	0.0	0.0	**78.1**	0.1	0.1	0.1	**64.5**	0.2	0.1	0.2
$\mathcal{L}_I$	90.4	62.9	85.6	86.9	67.9	36.2	63.7	65.0	63.4	6.4	56.1	59.8
$\mathcal{L}_{F(w/o\ c)}$	87.2	84.0	86.8	87.2	61.8	50.5	61.3	61.6	62.7	31.1	57.7	61.5
$\mathcal{L}_{I+F(w/o\ c)}$	90.7	81.9	89.0	89.4	68.1	50.1	65.3	66.1	63.2	32.4	57.3	60.4
$\mathcal{L}_T$	92.9	29.5	84.0	85.9	71.2	15.8	61.2	63.7	63.4	0.7	54.0	58.8
$\mathcal{L}_{T+F(w/o\ c)}$	92.5	76.2	86.6	87.1	70.3	45.8	63.7	64.7	63.4	30.8	54.4	58.6
Full	90.7	**85.1**	**90.3**	**90.6**	68.7	**51.6**	**66.9**	**67.7**	62.9	**34.6**	**59.1**	**62.1**

Table 2: Results of ablation study on the segmentation and detection tasks.

	Cityscapes (mIoU %)				VOC2012 (mIoU %)				VOC07+12 (mAP %)			
	clean	BIM	DeepFool	C&W	clean	BIM	DeepFool	C&W	clean	cls	loc	cls+loc
No Defense	**73.5**	3.7	37.4	6.6	**76.4**	8.2	41.5	2.6	**72.5**	5.4	2.8	1.1
$\mathcal{L}_I$	61.5	50.3	51.6	55.9	73.1	48.8	60.5	67.7	63.4	37.4	48.4	38.3
$\mathcal{L}_{F(w/o\ c)}$	60.4	57.5	59.6	62.6	47.4	48.2	53.3	57.3	34.5	33.1	33.4	33.0
$\mathcal{L}_{I+F(w/o\ c)}$	64.9	56.5	59.7	63.3	70.3	60.1	63.4	67.2	58.7	41.6	50.4	42.3
$\mathcal{L}_T$	63.8	43.6	50.5	53.4	74.1	34.8	51.9	55.6	65.1	26.9	38.9	27.8
$\mathcal{L}_{T+F(w/o\ c)}$	65.0	53.2	57.5	61.5	71.2	56.8	62.9	66.6	60.0	32.4	44.7	33.7
Full	67.6	**58.5**	**60.9**	**64.4**	71.2	**61.3**	**64.5**	**68.7**	60.5	**44.0**	**52.0**	**44.7**

4.2 Training and Evaluation

In the classification task, $\mathcal{O}$ is adopted as the structure of WideResNet (Zhang et al., 2019) and ResNet50 (He et al., 2016); as for semantic segmentation, we employ $\mathcal{O}$ with the architecture of PSPNet (Zhao et al., 2017) and DeepLabv3 (Chen et al., 2017); in object detection, $\mathcal{O}$ is set as the framework of SSD (Liu et al., 2016a) and RFBNet (Liu et al., 2018) (due to page limitations, we only provide the results with WideResNet, PSPNet and SSD here, and the outcomes with ResNet50, DeepLabv3 and RFBNet are given in the appendix). We use $\mathcal{O}$ (which is trained with no defense) to train $\mathcal{G}$ in input transformation strategies (including ours), and the processed clean/adversarial samples $\widehat{x}^c$/$\widehat{x}^a$ are finally evaluated by $\mathcal{O}$. Moreover, to implement the experiments within classification, the adversarial perturbations ϵ during training are generated by PGD (Madry et al., 2017) with KL criterion, and obtained by PGD with cross-entropy criterion, DeepFool attack (Moosavi-Dezfooli et al., 2016) and C&W attack (Carlini & Wagner, 2017) during testing; for semantic segmentation, we employ BIM (Kurakin et al., 2016) for training, and adopt BIM, DeepFool and C&W for evaluation; in object detection, classification attack ("cls") and localization attack ("loc") (Zhang & Wang, 2019) are utilized during training and "cls", "loc" and "cls+loc" (simultaneously conduct classification and localization attacks) are adopted for testing. The parameters of attacks in three tasks are detailedly described in the appendix. In addition, we execute the experiments with white-box attack for $\mathcal{O}$ here, and the consequences with black-box attack is reported in the appendix.

4.3 Ablation study

In experiments of each task, we conduct ablation studies to analyze the impact of each loss term and verify the superiority of our novelty on the pixel-level and feature-level constraints. Especially, we denote $\mathcal{L}_I$ as the results with only pixel-level constraints in our approach, $\mathcal{L}_{F(w/o\ c)}$ as the outcomes with sole feature-level constraints (except $\mathcal{L}_{F_{class}}$), and $\mathcal{L}_{I+F(w/o\ c)}$ as the effects with our full constraints apart from $\mathcal{L}_{F_{class}}$. And $\mathcal{L}_T$ represents the consequences with traditional pixel-level constraints merely and $\mathcal{L}_{T+F(w/o\ c)}$ refers to the results with traditional pixel-level constraints as well as feature-level constraints other than $\mathcal{L}_{F_{class}}$.

The results in Table 1 and 2 illustrate that, the scheme with only pixel-level constraint ($\mathcal{L}_I$) performs poorly on adversarial samples, while the setting with sole feature-level constraint ($\mathcal{L}_{F(w/o\ c)}$) leads to unsatisfactory consequences on the clean samples (especially, in experiments of detection, such setting will also largely reduce the defense quality on adversarial samples). Moreover, compared with traditional pixel-level constraints, our novel pixel-level alignment strategy achieves much stronger robustness on adversarial samples with a negligible degradation on clean samples as cost. Such superiority is prominent within the comparison between $\mathcal{L}_I$ and $\mathcal{L}_T$, as well as $\mathcal{L}_{I+F(w/o\ c)}$ and $\mathcal{L}_{T+F(w/o\ c)}$. Further, as compared to $\mathcal{L}_{I+F(w/o\ c)}$, our full setting exhibits stable improvements for the performance on clean/adversarial samples and this indicates the effect of $\mathcal{L}_{F_{class}}$.

Table 3: Comparison between our approach and other methods on the classification task.

Method	CIFAR10 (Accuracy %)				CIFAR100 (Accuracy %)				ImageNet (Accuracy %)			
	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W
No Defense	95.1	0.0	0.0	0.1	78.1	0.1	0.1	0.1	**64.5**	0.2	0.1	0.2
No Defense (finetune)	**95.6**	0.3	0.5	0.4	**79.5**	0.3	0.3	0.3	63.9	0.1	0.3	0.4
TRADES (Zhang et al., 2019)	87.3	51.3	52.3	86.8	62.8	27.3	17.5	59.5	58.5	19.8	13.8	55.9
TRADES (finetune) (Zhang et al., 2019)	85.4	52.7	56.5	84.8	78.1	0.1	26.5	55.1	43.3	16.8	15.0	42.5
Free-adv (Shafahi et al., 2019)	77.1	29.4	31.8	75.2	49.2	15.0	14.4	47.6	55.5	20.2	19.1	54.6
Free-adv (finetune) (Shafahi et al., 2019)	88.5	45.9	52.4	87.4	63.7	22.5	24.6	61.7	42.2	8.3	12.5	48.1
Quilting (Guo et al., 2017)	87.2	83.4	83.2	82.8	64.9	42.1	43.2	42.6	58.4	24.5	25.1	25.9
Quilting + TVM (Guo et al., 2017)	89.2	83.7	84.2	83.7	66.2	43.5	44.1	44.5	56.1	25.2	25.6	26.1
JPEG (Liu et al., 2019)	89.1	38.6	65.4	84.5	67.1	21.5	35.3	42.1	56.3	12.1	20.4	25.2
Randomization (Xie et al., 2017a)	88.2	56.8	76.2	77.4	66.4	35.8	41.7	42.7	57.4	20.3	26.5	25.3
NRP (Naseer et al., 2020)	91.8	82.7	88.6	88.8	70.3	47.2	65.0	65.9	59.4	33.4	57.3	59.7
Denoise (Liao et al., 2018)	89.8	80.3	87.8	87.8	67.2	46.3	64.0	64.3	59.8	33.5	56.1	60.3
APE (Shen et al., 2017)	90.2	19.5	83.2	82.5	73.2	10.7	60.8	62.6	62.4	0.5	57.9	60.4
FPD (Li et al., 2020)	48.5	47.8	48.4	48.5	52.5	41.8	53.5	52.8	39.7	32.1	39.2	39.7
Defense (Samangouei et al., 2018)	39.9	40.0	43.1	44.4	31.1	31.0	32.0	32.1	20.4	22.6	21.5	22.3
SR (Mustafa et al., 2019)	48.0	47.4	47.9	48.0	33.5	33.1	34.5	34.5	31.1	29.7	32.5	33.2
Ours	90.7	**85.1**	**90.3**	**90.6**	68.7	**51.6**	**66.9**	**67.7**	62.9	**34.6**	**59.1**	**62.1**

Table 4: Comparison between our approach and other methods on segmentation and detection tasks.

Method	Cityscapes (mIoU %)				VOC2012 (mIoU %)				VOC07+12 (mAP %)			
	clean	BIM	DeepFool	C&W	clean	BIM	DeepFool	C&W	clean	cls	loc	cls+loc
No Defense	**73.5**	3.7	37.4	6.6	**76.4**	8.2	41.5	2.6	72.5	5.4	2.8	1.1
No Defense (finetune)	73.3	7.0	36.2	7.0	76.3	8.8	40.9	2.6	**73.6**	7.3	5.2	2.2
SAT (Xu et al., 2020)	65.7	28.2	52.7	44.3	73.9	45.7	60.9	66.5	–	–	–	–
SAT (finetune) (Xu et al., 2020)	66.3	21.4	47.5	37.4	74.2	20.0	56.1	42.3	–	–	–	–
DDCAT (Xu et al., 2020)	67.7	30.2	54.1	45.7	75.1	47.9	62.8	68.4	–	–	–	–
DDCAT (finetune) (Xu et al., 2020)	68.3	23.6	49.2	38.7	76.0	22.6	57.3	44.4	–	–	–	–
CLS (Zhang & Wang, 2019)	–	–	–	–	–	–	–	–	47.8	22.3	31.5	30.4
LOC (Zhang & Wang, 2019)	–	–	–	–	–	–	–	–	52.9	24.2	27.2	26.0
CON (Zhang & Wang, 2019)	–	–	–	–	–	–	–	–	40.7	18.8	27.8	26.6
MTD (Zhang & Wang, 2019)	–	–	–	–	–	–	–	–	49.1	29.5	32.6	31.8
NRP (Naseer et al., 2020)	65.0	52.2	48.4	63.4	70.5	54.9	51.6	66.5	60.4	40.7	53.2	42.1
Denoise (Liao et al., 2018)	64.4	51.5	52.8	63.3	70.4	52.8	54.5	65.5	61.6	38.9	50.4	39.1
APE (Shen et al., 2017)	54.5	31.7	33.3	44.8	74.3	27.2	49.4	48.6	62.3	24.4	36.0	25.4
FPD (Li et al., 2020)	55.9	52.6	53.2	54.8	61.5	56.5	58.2	60.0	57.2	40.6	43.7	40.9
Defense (Samangouei et al., 2018)	22.2	21.2	25.2	26.1	23.5	25.6	28.1	30.9	34.6	32.5	34.0	33.8
SR (Mustafa et al., 2019)	60.7	49.0	50.1	52.0	71.2	53.0	58.3	66.1	54.6	34.6	36.6	35.7
Ours	67.6	**58.5**	**60.9**	**64.4**	71.2	**61.3**	**64.5**	**68.7**	60.5	**44.0**	**52.0**	**44.7**

4.4 Comparison with existing methods

We choose current approaches of input transformation and adversarial training for the comparison. Most existing adversarial training works focus on image classification, and we adopt two recent methods, Zhang et al. (2019) and Shafahi et al. (2019), to compare. For the semantic segmentation task, Xu et al. (2020) firstly conducted a comprehensive exploration on the impact of adversarial training for semantic segmentation, and we employ two strategies proposed by Xu et al. (2020) that are SAT and DDC-AT. As for object detection, Zhang & Wang (2019) proposed four variants of adversarial training, which can be utilized. All these adversarial training approaches are trained with their original configurations and "finetune" means finetuning pre-trained models (with no defense) via adversarial training. As for the input transformation strategies, they can be divided into two categories: learning-based and non-learning-based. For non-learning-based methods, their effects are only validated on classification networks and we select Guo et al. (2017), Liu et al. (2019) and Xie et al. (2017a) to compare in the classification task. For learning-based schemes, we use six representative works including Naseer et al. (2020); Liao et al. (2018); Shen et al. (2017); Li et al. (2020); Samangouei et al. (2018); Mustafa et al. (2019). Moreover, we adopt their original generator structures, and re-train them with the same epochs, batch size and target models as us.

The results in classification task are summarized in Table 3. Although a few approaches have comparable effects on clean samples with us, our full setting results in the strongest robustness on adversarial samples which illustrates the superiority of our method. Further, as exhibited in Table 4, in semantic segmentation and object detection, our results also outperforms all the competing methods on adversarial samples. At the same time, only a fraction of approaches have greater outcomes on clean samples than ours while they exhibit weak robustness on adversarial samples (e.g., APE). In

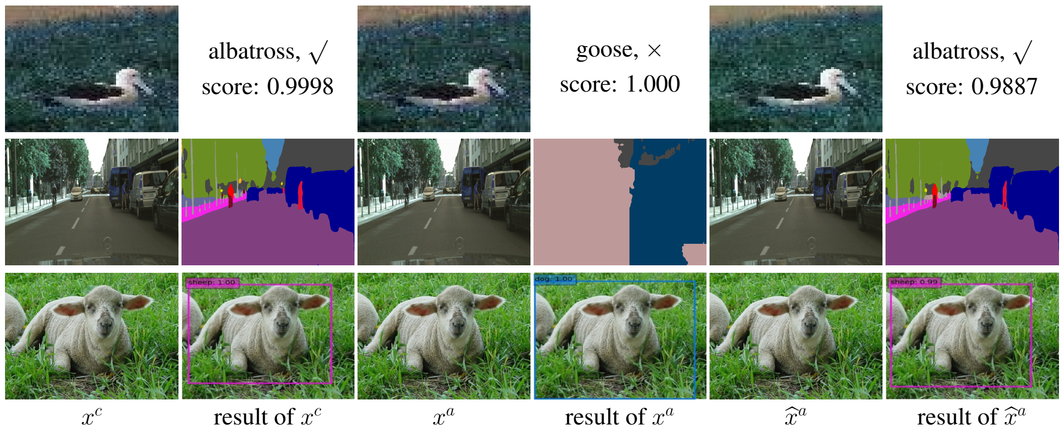


Figure 4: The visual illustrations for the results of clean samples x^c, adversarial samples x^a and processed adversarial samples $\widehat{x}^a$ with our trained generator $\mathcal{G}$, in image classification on ImageNet, semantic segmentation on Cityscapes, object detection on VOC07+12.

Table 5: Results with no-differentiable operation in our framework on three tasks.

	CIFAR10 (Accuracy %)				CIFAR100 (Accuracy %)				ImageNet (Accuracy %)			
	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W
No Defense	95.1	0.0	0.0	0.1	78.1	0.1	0.1	0.1	64.5	0.2	0.1	0.2
Full	90.7	85.1	90.3	90.6	68.7	51.6	66.9	67.7	62.9	34.6	59.19	62.1
Full with jpeg	90.4	84.3	89.9	90.3	66.6	51.4	65.4	66.1	62.8	32.0	57.4	60.1
	Cityscapes (mIoU %)				VOC2012 (mIoU %)				VOC07+12 (mAP %)			
	clean	BIM	DeepFool	C&W	clean	BIM	DeepFool	C&W	clean	cls	loc	cls+loc
No Defense	73.5	3.7	37.4	6.6	76.4	8.2	41.5	2.6	72.5	5.4	2.8	1.1
Full	67.6	58.5	60.9	64.4	71.2	61.3	64.5	68.7	60.5	44.0	52.0	44.7
Full with jpeg	66.5	57.1	61.1	63.0	70.9	60.7	63.3	68.1	60.3	43.9	51.9	44.6

addition, we provide visual illustrations to show the effectiveness of our method. As shown in Fig. 4, our trained $\mathcal{G}$ leads to satisfactory defense effects on adversarial samples in all three tasks.

4.5 NO-DIFFERENTIABLE OPERATIONS

In the worst situation, attackers have access to the weights and structure of $\mathcal{G}$, and they then utilize the gradients of $\mathcal{G}$ for generating more threatening adversarial examples. To this, recent works have illustrated the function of no-differentiable operations to avoid such attack (Gupta & Rahtu, 2019; Guo et al., 2017): no-differentiable operations (e.g., JPEG compression) can be applied on the synthesized images from the generator, and then the processed images would be sent to the target model. In this way, the gradients of the generator are truncated by no-differentiable operations and can not be obtained by attackers. Though desired effects of no-differentiable operations, we need to consider whether such operations will harm the performance of our method.

To this, we choose JPEG compression as such no-differentiable operation and explore the effect of our framework when the synthesized images from $\mathcal{G}$ will be processed by JPEG compression before they are sent to the target model. The results are displayed in Table 5 with name of "Full with jpeg", and it is obvious that even we add such no-differentiable operation into our method, the performance is reduced less than 3% on the classification task, 2% in the segmentation and detection experiments.

5 CONCLUSION

In this paper, we propose a novel training scheme for DGNs which can help to align the distribution of adversarial samples to clean samples for a given target model. The effectiveness of our strategy is derived from the novelty on the pixel-level as well as feature-level constraints. As a general approach, our framework is suitable for various computer vision tasks, including image classification, semantic segmentation and object detection. Extensive experiments reveal the effect of our novel constraints and illustrate the advantage of our method compared with existing defense strategies.

REFERENCES

Anurag Arnab, Ondrej Miksik, and Philip HS Torr. On the robustness of semantic segmentation models to adversarial attacks. In *CVPR*, 2018.

Anish Athalye and Nicholas Carlini. On the robustness of the cvpr 2018 white-box adversarial example defenses. *arXiv:1804.03286*, 2018.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE symposium on security and privacy*, 2017.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587*, 2017.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Yinpeng Dong, Hang Su, Baoyuan Wu, Zhifeng Li, Wei Liu, Tong Zhang, and Jun Zhu. Efficient decision-based black-box adversarial attacks on face recognition. In *CVPR*, 2019.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, 2012.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

Ian Goodfellow, Jean Pouget-Abadie, and Mehdi Mirza. Generative adversarial nets. In *NIPS*, 2014a.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2014b.

Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv:1711.00117*, 2017.

Puneet Gupta and Esa Rahtu. Ciidefence: Defeating adversarial attacks by fusing class-specific image inpainting and image denoising. In *ICCV*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Yunhan Jia, Yantao Lu, Junjie Shen, Qi Alfred Chen, Hao Chen, Zhenyu Zhong, and Tao Wei. Fooling detection alone is not enough: Adversarial attack against multiple object tracking. In *ICLR*, 2019.

Ameya Joshi, Amitangshu Mukherjee, Soumik Sarkar, and Chinmay Hegde. Semantic adversarial attacks: Parametric transformations that fool deep classifiers. In *CVPR*, 2019.

Zelun Kong, Junfeng Guo, Ang Li, and Cong Liu. Physgan: Generating physical-world-resilient adversarial examples for autonomous driving. In *CVPR*, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *ICLR*, 2016.

Guanlin Li, Shuya Ding, Jun Luo, and Chang Liu. Enhancing intrinsic adversarial robustness via feature pyramid decoder. In *CVPR*, 2020.

Yuezun Li, Daniel Tian, Xiao Bian, Siwei Lyu, et al. Robust adversarial perturbation on deep proposal-based models. *arXiv:1809.05962*, 2018.

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.

Songtao Liu, Di Huang, et al. Receptive field block net for accurate and fast object detection. In *ECCV*, 2018.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016a.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv:1611.02770*, 2016b.

Zihao Liu, Qi Liu, Tao Liu, Nuo Xu, Xue Lin, Yanzhi Wang, and Wujie Wen. Feature distillation: Dnn-oriented jpeg compression against adversarial examples. In *CVPR*, 2019.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083*, 2017.

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *ICCV*, 2017.

Jan Hendrik Metzen, Mummadi Chaithanya Kumar, Thomas Brox, and Volker Fischer. Universal adversarial perturbations against semantic image segmentation. In *ICCV*, 2017.

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

Aamir Mustafa, Salman H Khan, Munawar Hayat, Jianbing Shen, and Ling Shao. Image super-resolution as a defense against adversarial attacks. *IEEE TIP*, 2019.

Muzammal Naseer, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Fatih Porikli. A self-supervised approach for adversarial robustness. In *CVPR*, 2020.

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

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *ACM on Asia conference on computer and communications security*, 2017.

Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. Deflecting adversarial attacks with pixel deflection. In *CVPR*, 2018.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv:1805.06605*, 2018.

Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *NIPS*, 2019.

Shiwei Shen, Guoqing Jin, Ke Gao, and Yongdong Zhang. Ape-gan: Adversarial perturbation elimination with gan. *arXiv:1707.05474*, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramer, Atul Prakash, and Tadayoshi Kohno. Physical adversarial examples for object detectors. In *12th {USENIX} Workshop on Offensive Technologies*, 2018.

Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv:1710.10766*, 2017.

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

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *ICLR*, 2017a.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *CVPR*, 2017b.

Xiaogang Xu, Hengshuang Zhao, and Jiaya Jia. Dynamic divide-and-conquer adversarial training for robust semantic segmentation. *arXiv:2003.06555*, 2020.

Haichao Zhang and Jianyu Wang. Towards adversarially robust object detection. In *CVPR*, 2019.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. *arXiv:1901.08573*, 2019.

Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

A Appendix

A.1 Details

Details of Training Constraints There are obvious distinctions among the networks of image classification, semantic segmentation and object detection. Thus, for these three tasks, there are also differences in extracting features to implement feature-level training constraints.

For the classification task, we adopt WideResNet and ResNet50 for experiments, and implement feature-level training constraints by extracting the feature of the fully connected layer before the final layer. Especially, each image $x \in \mathbb{R}^{H \times W \times 3}$ (H and W are the height and width of the image) has one class label $y \in \mathbb{R}^{K}$ and one feature vector $z \in \mathbb{R}^{L}$ (L is the length of the vector). And we group features into K classes according to y.

As for semantic segmentation, we use PSPNet and DeepLabv3 with ResNet50 as the backbone and complete feature-level constraints by using the feature of the convolution layer before the final layer. Different with the classification task, each image in semantic segmentation task has multiple feature vectors with multiple class labels: for an image $x \in \mathbb{R}^{H \times W \times 3}$, it has the corresponding segmentation map $y \in \mathbb{R}^{H \times W \times K}$. And the feature of this image can be denoted as $z' \in \mathbb{R}^{h \times w \times L}$ (h and w are the height and width of the feature) with a resized segmentation map $y' \in \mathbb{R}^{h \times w \times K}$. Then we can obtain a set of feature vectors $\{z_1, ..., z_{h \times w}\}(z_i \in \mathbb{R}^{L}, i \in \{1, ..., h \times w\})$ with their labels as $\{y_1, ..., y_{h \times w}\}(y_i \in \mathbb{R}^{K}, i \in \{1, ..., h \times w\})$, which are reshaped from z' as well as y'. Such set of feature vectors can be divided into K classes and utilized in feature-level constraints.

Furthermore, SSD and RFBNet with VGG16 as the backbone are employed for experiments within the object detection task. And the features for feature-level constraints are obtained as the outputs of the backbone. In this task, an image has one class label for each bounding box and thus has multiple feature vectors with multiple bounding boxes: an image $x \in \mathbb{R}^{H \times W \times 3}$ can have B bounding boxes and the feature of this image can be represented as $z' \in \mathbb{R}^{h \times w \times L}$. The feature of each bounding box can be cropped from this feature map with its coordinate as well as the corresponding class label, as $\{z_1, ..., z_B\}(z_i \in \mathbb{R}^{L}, i \in \{1, ..., B\})$ and $\{y_1, ..., y_B\}(y_i \in \mathbb{R}^{K}, i \in \{1, ..., B\})$. We separate such features into K classes to implement feature-level constraints.

Details of Generator For the classification task, images from CIFAR10/CIFAR100 have size of 32×32 and images from ImageNet have size of 64×64 for the input of the target models as well as the generators, and the generators have encoder-decoder structures with two down-sample/up-sample layers; in the semantic segmentation task, the images from Cityscapes and VOC2012 are shaped as 512×512 for the generators with four down-sample/up-sample layers, and are shaped as 425×425 and 417×417 for the target models respectively; as for object detection, the input size of the generators, which have four down-sample/up-sample layers, is 256×256 for VOC07+12, and the input size of the target models is 300×300. In addition, we employ the structure for the generator as the "Global Generator" in pix2pixHD (`https://github.com/NVIDIA/pix2pixHD`) for our experiments. Besides, our training constraints are indeed suitable for various generative structures.

Details of Attacks In this paper, all attacks are conducted with L_∞ constraint and untargeted form. For the classification task, we utilize PGD attack with KL criterion during training, and the perturbation range $\varepsilon = 0.031 \times 255$, the step size $\alpha = 0.0175 \times 255$, the number of attack iteration $n = 4$. During testing, we adopt PGD with cross-entropy criterion ($\varepsilon = 0.031 \times 255$, $\alpha = 0.0075 \times 255$, $n = 8$), DeepFool attack ($\varepsilon = 0.031 \times 255$), C&W attack ($\varepsilon = 0.031 \times 255$, the step size is 0.0075×255). In the semantic segmentation task, we use BIM ($\varepsilon = 0.03 \times 255$, $\alpha = 0.01 \times 255$, $n = 3$) during training, and BIM ($\varepsilon = 0.03 \times 255$, $\alpha = 0.01 \times 255$, $n = 4$), DeepFool attack ($\varepsilon = 0.03 \times 255$), C&W attack ($\varepsilon = 0.03 \times 255$, the step size is 0.01×255) for evaluation. Moreover, the attacks for classification loss and location loss are employed within the object detection task, and the perturbation range is 8 for pixel values within $[0, 255]$.

Table 6: Results of ablation study on the classification task with the structure of target model $\mathcal{O}$ as ResNet50 (He et al., 2016).

	CIFAR10 (Accuracy %)				CIFAR100 (Accuracy %)				ImageNet (Accuracy %)			
	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W
No Defense	**94.3**	0.0	0.0	0.0	**76.1**	0.1	0.1	0.2	**62.1**	0.2	0.1	0.2
$\mathcal{L}_I$	90.6	53.2	85.1	85.9	65.9	41.4	64.0	65.1	61.0	7.2	56.7	59.0
$\mathcal{L}_{F(w/o\ c)}$	88.9	81.0	88.6	88.8	58.8	54.4	58.4	58.7	60.4	31.9	56.5	59.5
$\mathcal{L}_{I+F(w/o\ c)}$	90.3	80.4	87.7	88.1	68.3	53.9	65.3	66.0	60.7	32.4	57.1	59.5
$\mathcal{L}_T$	91.0	23.8	83.4	84.9	69.9	24.5	60.7	62.5	61.5	1.2	54.1	57.4
$\mathcal{L}_{T+F(w/o\ c)}$	91.7	75.2	85.6	86.2	69.9	48.2	63.5	64.5	61.3	30.5	55.8	57.1
Full	90.0	**82.7**	**89.1**	**89.5**	67.3	**55.5**	**66.6**	**67.3**	61.0	**33.7**	**57.6**	**60.2**

Table 7: Results of ablation study on the semantic segmentation and object detection tasks, with the structure of target model $\mathcal{O}$ as DeepLabv3 (Chen et al., 2017) and RFBNet (Liu et al., 2018) respectively.

	Cityscapes (mIoU %)				VOC2012 (mIoU %)				VOC07+12 (mAP %)			
	clean	BIM	DeepFool	C&W	clean	BIM	DeepFool	C&W	clean	cls	loc	cls+loc
No Defense	**73.2**	3.1	32.4	8.0	**76.9**	8.2	38.0	2.9	**80.6**	4.5	4.0	2.0
$\mathcal{L}_I$	64.0	50.9	51.2	56.9	72.6	54.8	59.2	66.5	73.2	53.2	62.3	54.2
$\mathcal{L}_{F(w/o\ c)}$	61.5	**59.9**	61.0	63.9	42.7	42.7	52.5	54.5	36.5	35.7	36.4	35.9
$\mathcal{L}_{I+F(w/o\ c)}$	65.1	57.9	60.1	63.4	68.1	59.8	62.3	67.2	68.4	57.1	63.2	58.3
$\mathcal{L}_T$	64.5	44.1	50.4	54.2	74.8	33.4	50.4	53.8	75.1	44.3	56.2	45.6
$\mathcal{L}_{T+F(w/o\ c)}$	64.9	55.1	57.6	60.5	70.5	55.9	61.5	66.1	62.3	51.0	56.1	53.1
Full	67.3	59.0	**61.7**	**64.8**	69.1	**61.3**	**63.9**	**68.3**	70.2	**59.2**	**65.0**	**60.3**

A.2 Experiments

Experiments under White-box Attack Evaluation We execute experiments of evaluation under white-box attack with different model structures for classification, semantic segmentation and object detection tasks, and the results for the model structure of WideResNet (Zhang et al., 2019), PSPNet (Zhao et al., 2017) and SSD (Liu et al., 2016a) have been reported in the manuscript. Here, we summarize the outcomes under white-box attack for ResNet50 (He et al., 2016), DeepLabv3 (Chen et al., 2017) and RFBNet (Liu et al., 2018). The consequences of the ablation study are listed in Table 6 and 7 and it is obvious that the models trained with only pixel-level constraints exhibit weak robustness on adversarial samples, while the models trained with sole feature-level constraints have poor performance on clean samples. Moreover, it is illustrated again that the models trained with our pixel-level constraints achieve better generalization on adversarial samples than traditional pixel-level constraints, and our class-aware feature-level constraint can stably enhance the effects on clean/adversarial samples. The results of the comparison with existing methods are summarized in Table 8 and 9, which support the superiority of our approach over them.

The visual illustration for injecting the no-differentiable operation into our framework is shown in Fig. 5. We have reported the results of our framework with no-differentiable operations in the manuscript, with target model structures as WideResNet (Zhang et al., 2019), PSPNet (Zhao et al., 2017) and SSD (Liu et al., 2016a). Here, we show the outcomes with target model architectures as ResNet50 (He et al., 2016), DeepLabv3 (Chen et al., 2017) and RFBNet (Liu et al., 2018). The outcomes are summarized in Table 10 as well as Table 11, and it is validated again that no-differentiable operations in our framework will only lead to an insignificant reduction of performance.

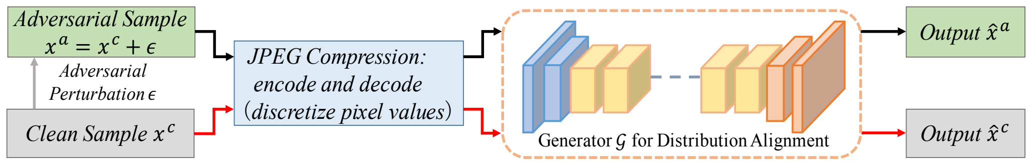

Adversarial Sample
$x^a = x^c + \epsilon$
Adversarial Perturbation ϵ
Clean Sample x^c
JPEG Compression: encode and decode (discretize pixel values)
Generator $\mathcal{G}$ for Distribution Alignment
Output $\hat{x}^a$
Output $\hat{x}^c$

Figure 5: The visual illustration for injecting the no-differentiable operation into our framework.

Table 8: Comparison between our approach and existing methods on the classification task, with the structure of target model $\mathcal{O}$ as ResNet50 (He et al., 2016).

Method	CIFAR10 (Accuracy %)				CIFAR100 (Accuracy %)				ImageNet (Accuracy %)			
	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W
No Defense	94.3	0.0	0.0	0.0	76.1	0.1	0.1	0.2	**62.1**	0.2	0.1	0.2
No Defense (finetune)	**94.7**	0.0	0.0	0.0	**77.8**	0.0	0.2	0.3	61.3	0.1	0.3	0.4
TRADES (Zhang et al., 2019)	85.3	51.7	47.1	83.2	59.8	26.7	21.2	56.6	44.3	19.3	16.1	43.6
TRADES (finetune) (Zhang et al., 2019)	83.2	52.6	53.9	82.4	56.3	28.5	26.7	55.4	43.3	16.5	15.0	42.3
Free-adv (Shafahi et al., 2019)	86.3	46.2	48.1	85.5	64.0	24.7	24.2	47.6	53.4	18.8	17.3	52.6
Free-adv (finetune) (Shafahi et al., 2019)	88.3	43.5	49.2	87.1	63.4	20.6	30.1	61.8	39.9	6.5	7.1	39.5
Quilting (Guo et al., 2017)	86.3	81.9	82.5	82.1	63.3	41.3	42.6	40.9	56.2	23.8	24.1	24.2
Quilting + TVM (Guo et al., 2017)	88.4	80.1	83.3	82.6	65.5	42.5	44.2	42.1	54.1	25.1	25.8	26.1
JPEG (Liu et al., 2019)	88.0	36.6	64.3	82.7	66.3	20.6	36.5	40.8	55.2	11.6	20.0	24.9
Randomization (Xie et al., 2017a)	87.5	55.2	74.3	75.5	65.0	34.3	40.5	41.5	56.3	20.0	25.8	26.1
NRP (Naseer et al., 2020)	90.3	80.5	87.0	87.3	66.7	47.2	65.5	66.5	56.5	31.7	56.2	56.8
Denoise (Liao et al., 2018)	88.3	79.0	87.1	87.2	66.7	46.8	64.7	64.9	56.6	32.8	**57.8**	57.3
APE (Shen et al., 2017)	89.1	12.2	80.1	81.3	71.7	17.4	60.2	61.4	60.2	0.96	56.5	59.3
FPD (Li et al., 2020)	53.0	52.4	53.0	53.0	50.0	43.6	49.2	49.8	34.5	27.8	33.7	34.4
Defense (Samangouei et al., 2018)	40.1	40.0	40.2	40.1	31.1	31.1	33.3	32.6	21.5	20.7	23.1	22.9
SR (Mustafa et al., 2019)	45.3	44.8	45.2	45.3	35.6	34.8	35.5	34.5	30.6	29.4	34.6	34.2
Ours	90.0	**82.7**	**89.1**	**89.5**	67.3	**55.5**	**66.6**	**67.3**	61.0	**33.7**	**57.6**	**60.2**

Table 9: Comparison between our approach and existing methods on the semantic segmentation and object detection tasks, with the structure of target model $\mathcal{O}$ as DeepLabv3 (Chen et al., 2017) and RFBNet (Liu et al., 2018) respectively.

Method	Cityscapes (mIoU %)				VOC2012 (mIoU %)				VOC07+12 (mAP %)			
	clean	BIM	DeepFool	C&W	clean	BIM	DeepFool	C&W	clean	cls	loc	cls+loc
No Defense	73.2	3.1	32.4	8.0	**76.9**	8.2	38.0	2.9	80.6	4.5	4.0	2.0
No Defense (finetune)	**73.5**	3.6	32.3	8.3	75.7	26.4	40.4	2.7	**80.8**	8.1	7.6	5.5
SAT (Xu et al., 2020)	64.3	28.9	50.2	43.8	72.8	44.5	57.8	64.9	–	–	–	–
SAT (finetune) (Xu et al., 2020)	65.4	23.4	47.5	37.1	74.8	9.3	61.8	46.7	–	–	–	–
DDCAT (Xu et al., 2020)	67.7	30.4	52.9	45.3	74.2	46.8	61.1	66.8	–	–	–	–
DDCAT (finetune) (Xu et al., 2020)	68.2	25.6	49.7	39.3	76.2	12.5	63.8	48.5	–	–	–	–
CLS (Zhang & Wang, 2019)	–	–	–	–	–	–	–	–	52.0	27.2	37.1	35.3
LOC (Zhang & Wang, 2019)	–	–	–	–	–	–	–	–	57.6	29.2	31.2	30.5
CON (Zhang & Wang, 2019)	–	–	–	–	–	–	–	–	43.5	22.8	32.1	30.4
MTD (Zhang & Wang, 2019)	–	–	–	–	–	–	–	–	53.7	34.5	36.5	35.1
NRP (Naseer et al., 2020)	65.6	55.1	47.9	61.8	68.1	54.8	52.6	66.9	68.5	53.7	63.9	55.0
Denoise (Liao et al., 2018)	64.6	56.5	52.3	62.0	67.2	58.6	53.6	67.6	68.5	51.3	54.4	51.7
APE (Shen et al., 2017)	54.2	28.5	21.2	33.4	75.5	25.1	47.0	41.6	72.2	42.2	53.9	43.8
FPD (Li et al., 2020)	49.5	49.0	49.7	50.8	62.1	58.4	60.0	61.3	58.7	49.5	52.2	50.2
Defense (Samangouei et al., 2018)	23.3	20.6	25.3	26.1	22.5	24.7	26.5	27.4	42.2	40.6	41.4	40.7
SR (Mustafa et al., 2019)	62.8	48.2	50.2	51.6	70.1	52.2	57.0	67.0	56.2	45.8	47.6	46.8
Ours	67.3	**59.0**	**61.7**	**64.8**	69.1	**61.3**	**63.9**	**68.3**	70.2	**59.2**	**65.0**	**60.3**

Table 10: Results with the no-differentiable operation in our framework on the classification task, with the structure of target model $\mathcal{O}$ as ResNet50 (He et al., 2016).

	CIFAR10 (Accuracy %)				CIFAR100 (Accuracy %)				ImageNet (Accuracy %)			
	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W
No Defense	94.3	0.0	0.0	0.0	76.1	0.1	0.1	0.2	62.1	0.2	0.1	0.2
Full	90.0	82.7	89.1	89.5	67.3	55.5	66.6	67.3	61.0	33.7	57.6	60.2
Full with jpeg	89.3	81.1	88.6	89.1	66.4	53.6	65.4	66.2	60.9	32.0	57.4	60.1

Table 11: Results with the no-differentiable operation in our framework on the semantic segmentation and object detection tasks, with the structure of target model $\mathcal{O}$ as DeepLabv3 (Chen et al., 2017) and RFBNet (Liu et al., 2018) respectively.

	Cityscapes (mIoU %)				VOC2012 (mIoU %)				VOC07+12 (mAP %)			
	clean	BIM	DeepFool	C&W	clean	BIM	DeepFool	C&W	clean	cls	loc	cls+loc
No Defense	73.2	3.1	32.4	8.0	76.9	8.2	38.0	2.9	80.6	4.5	4.0	2.0
Full	67.3	59.0	61.7	64.8	69.1	61.3	63.9	68.3	70.2	59.2	65.0	60.3
Full with jpeg	65.7	57.3	59.6	62.3	68.2	60.8	64.1	67.1	70.2	59.1	58.6	64.2

Table 12: Results of ablation study on the classification, semantic segmentation and object detection tasks under the evaluation of black-box attack. "WideResNet → ResNet50" means attacking WideResNet while generating adversarial perturbations from ResNet50; "PSPNet → DeepLabv3" means attacking PSPNet while generating adversarial perturbations from DeepLabv3; "SSD → RFBNet" means attacking SSD while generating adversarial perturbations from RFBNet.

WideResNet → ResNet50	CIFAR10 (Accuracy %)				CIFAR100 (Accuracy %)				ImageNet (Accuracy %)			
	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W
No Defense	**95.1**	2.1	5.3	6.4	**78.1**	7.5	9.3	10.4	**64.5**	19.2	20.4	21.2
$\mathcal{L}_I$	90.4	66.4	85.3	86.1	67.9	46.4	66.6	67.5	63.4	34.3	60.7	61.7
$\mathcal{L}_{F(w/o\ c)}$	87.2	51.8	86.7	87.1	61.8	51.9	60.9	61.6	62.7	45.6	61.8	62.6
$\mathcal{L}_{I+F(w/o\ c)}$	90.7	78.2	89.0	89.5	68.1	53.4	67.0	67.8	63.2	46.1	61.5	62.4
$\mathcal{L}_T$	92.9	49.5	84.4	85.5	71.2	39.3	65.1	66.6	63.4	32.0	58.3	59.2
$\mathcal{L}_{T+F(w/o\ c)}$	92.5	74.1	87.6	88.3	70.3	50.4	65.9	66.8	63.4	44.4	60.2	61.1
Full	90.7	**81.4**	**90.4**	**90.6**	68.7	**54.0**	**67.5**	**68.4**	62.9	**46.4**	**62.2**	**63.2**
PSPNet → DeepLabv3 / SSD → RFBNet	Cityscapes (mIoU %)				VOC2012 (mIoU %)				VOC07+12 (mAP %)			
	clean	BIM	DeepFool	C&W	clean	BIM	DeepFool	C&W	clean	cls	loc	cls+loc
No Defense	**73.5**	3.6	38.6	12.5	**76.4**	11.1	46.1	16.6	**72.5**	17.9	14.5	15.0
$\mathcal{L}_I$	61.5	50.8	52.4	56.1	73.1	59.8	63.0	68.3	63.4	59.4	54.4	58.2
$\mathcal{L}_{F(w/o\ c)}$	60.4	58.1	59.5	62.6	47.4	47.3	47.3	47.4	34.5	34.6	33.8	33.9
$\mathcal{L}_{I+F(w/o\ c)}$	64.9	58.7	60.7	63.4	70.3	62.9	64.7	68.6	58.7	60.7	57.4	60.1
$\mathcal{L}_T$	63.8	46.3	50.9	54.6	74.1	45.0	57.2	63.1	65.1	58.3	53.7	56.3
$\mathcal{L}_{T+F(w/o\ c)}$	65.0	55.6	59.1	61.3	71.2	61.3	62.4	66.1	60.0	59.6	55.0	58.2
Full	67.6	**59.5**	**62.0**	**64.7**	71.2	**63.6**	**66.0**	**69.5**	60.5	**61.2**	**58.3**	**60.9**

Table 13: Comparison between our approach and existing methods on the classification task under the evaluation of black-box attack. "WideResNet → ResNet50" means attacking WideResNet while generating adversarial perturbations from ResNet50.

WideResNet → ResNet50	CIFAR10 (Accuracy %)				CIFAR100 (Accuracy %)				ImageNet (Accuracy %)			
	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W
No Defense	95.1	2.1	5.3	6.4	78.1	7.5	9.3	10.4	**64.5**	19.2	20.4	21.2
No Defense (finetune)	**95.6**	3.8	7.5	8.7	**79.5**	8.0	9.7	10.9	63.9	22.1	23.2	23.6
TRADES (Zhang et al., 2019)	87.3	79.7	87.1	87.2	62.8	53.9	62.6	62.7	58.5	45.3	58.3	58.5
TRADES (finetune) (Zhang et al., 2019)	85.4	78.5	85.3	85.4	78.1	53.9	55.8	44.2	43.3	42.2	43.2	43.3
Free-adv (Shafahi et al., 2019)	77.1	73.1	76.7	77.0	49.2	46.6	49.0	49.2	55.5	45.8	55.4	55.5
Free-adv (finetune) (Shafahi et al., 2019)	88.5	79.8	88.4	88.5	63.7	**54.1**	63.5	63.7	42.2	40.9	48.6	48.7
NRP (Naseer et al., 2020)	91.8	79.0	89.2	89.8	70.3	52.8	66.6	67.2	59.4	45.5	59.0	59.4
Denoise (Liao et al., 2018)	89.8	80.0	90.1	90.0	67.2	53.1	66.7	66.1	59.8	45.5	59.4	59.9
APE (Shen et al., 2017)	90.2	41.9	89.1	89.8	73.2	37.2	65.7	67.7	62.4	30.5	61.6	62.3
FPD (Li et al., 2020)	48.5	47.6	48.4	48.5	52.5	41.2	42.4	42.5	39.7	33.7	39.5	39.8
Defense (Samangouei et al., 2018)	39.9	38.7	38.1	39.3	31.1	30.5	30.9	31.0	20.4	18.5	19.7	19.9
SR (Mustafa et al., 2019)	48.0	47.2	48.0	48.1	33.6	33.1	33.5	33.8	31.1	30.5	30.9	31.1
Ours	90.7	**81.4**	**90.4**	**90.6**	68.7	54.0	**67.5**	**68.4**	62.9	**46.4**	**62.2**	**63.2**

Experiments under Black-box Attack Evaluation In the evaluation of black-box attack, attackers cannot utilize the exact gradient information of the target model. Instead, they normally obtain gradient information from a substitute network, which is trained on the same dataset (Papernot et al., 2017; 2016; Liu et al., 2016b) with different model structures. Thus, in the black-box evaluation of the classification task, we can use the perturbations computed from ResNet50 to attack the defense framework trained with WideResNet; for the semantic segmentation task, the adversarial samples obtained from DeepLabv3 can be adopted to achieve black-box attacks for PSPNet; as for object detection, the black-box attacks for SSD can be implemented by employing the adversarial perturbations generated from RFBNet. The results of these black-box experiments are reported in Table 12, 13, 14, and we can obtain the same conclusions as that in the experiments of white-box attack.

Experiments under Model Transfer Evaluation For defenses via input transformation with deep generative models, we shall use a target model $\mathcal{O}$ for the feature-level training and it has been verified that the trained generator $\mathcal{G}$ has great defense quality for $\mathcal{O}$. On the other hand, we indeed require $\mathcal{G}$ to defend for different target models that have not been employed during the training. Thus, the evaluations with model transfer are conducted: in the classification task, we can defend for ResNet50 under white-box attack while the generator $\mathcal{G}$ is trained with the target model of WideResNet; as for semantic segmentation, the generator $\mathcal{G}$ is trained with PSPNet while utilized for the defense of DeepLabv3. The results are summarized in Table 15 and 16. These outcomes provide empirical

Table 14: Comparison between our approach and existing methods on the semantic segmentation and object detection tasks under the evaluation of black-box attack. “PSPNet $\rightarrow$ DeepLabv3” means attacking PSPNet while generating adversarial perturbations from DeepLabv3; “SSD $\rightarrow$ RFBNet” means attacking SSD while generating adversarial perturbations from RFBNet.

PSPNet $\rightarrow$ DeepLabv3 / SSD $\rightarrow$ RFBNet	Cityscapes (mIoU %)				VOC2012 (mIoU %)				VOC07+12 (mAP %)			
	clean	BIM	DeepFool	C&W	clean	BIM	DeepFool	C&W	clean	cls	loc	cls+loc
No Defense	**73.5**	3.6	38.6	12.5	**76.4**	11.1	46.1	16.6	72.5	17.9	14.5	15.0
No Defense (finetune)	73.3	3.6	37.1	12.5	76.3	11.2	45.2	15.4	**73.6**	19.6	16.8	16.1
SAT (Xu et al., 2020)	65.7	50.5	52.5	51.0	73.9	57.1	64.3	69.3	–	–	–	–
SAT (finetune) (Xu et al., 2020)	66.3	42.1	47.7	42.4	74.2	60.5	60.1	63.4	–	–	–	–
DDCAT (Xu et al., 2020)	67.7	51.4	54.4	51.8	75.1	58.8	65.1	69.3	–	–	–	–
DDCAT (finetune) (Xu et al., 2020)	68.3	43.3	49.4	43.1	76.0	62.4	62.4	64.8	–	–	–	–
CLS (Zhang & Wang, 2019)	–	–	–	–	–	–	–	–	47.8	34.5	43.3	44.1
LOC (Zhang & Wang, 2019)	–	–	–	–	–	–	–	–	52.9	36.7	38.8	39.9
CON (Zhang & Wang, 2019)	–	–	–	–	–	–	–	–	40.7	31.3	39.5	40.3
MTD (Zhang & Wang, 2019)	–	–	–	–	–	–	–	–	49.1	42.0	44.3	44.1
NRP (Naseer et al., 2020)	65.0	55.2	49.3	64.1	70.5	62.6	59.3	68.5	60.4	59.9	55.4	58.9
Denoise (Liao et al., 2018)	64.4	55.1	53.8	64.0	70.4	61.9	61.5	67.8	61.6	52.2	50.9	51.8
APE (Shen et al., 2017)	54.5	34.7	33.3	45.4	74.3	37.8	54.3	59.9	62.3	57.9	58.0	55.8
FPD (Li et al., 2020)	55.9	53.0	53.2	55.0	61.5	57.5	58.6	59.9	57.2	58.1	55.8	57.8
Defense (Samangouei et al., 2018)	22.2	20.2	19.9	21.2	23.5	21.9	22.6	23.3	34.6	30.2	30.8	29.7
SR (Mustafa et al., 2019)	60.7	52.0	50.6	51.5	71.2	58.1	60.3	67.2	54.6	36.2	37.5	34.5
Ours	67.6	**59.5**	**62.0**	**64.7**	71.2	**63.6**	**66.0**	**69.5**	60.5	**61.2**	**58.3**	**60.9**

evidence that even the trained generator $\mathcal{G}$ is applied for the defense of the target model which is not adopted during training, our trained $\mathcal{G}$ can still have great defense effects and outperforms most of the existing methods as well as ablation settings.

A.3 Analysis

Theoretical Analysis–Superiority of Our Pixel-level Constraint Using a network $\mathcal{G}$ to weaken the discrepancy between adversarial samples with the corresponding clean samples, $\|\widehat{x}^a - x^c\|$ can not be zero unless over-fitting, since the output space of $\mathcal{G}$ is smaller than the value space of real images. In the traditional setting, given x^c, $\|\widehat{x}^a - x^c\|$ is set as the objective to optimize and there exist multiple solutions $\widehat{x}^a$ that have the same value for $\|\widehat{x}^a - x^c\|$. Thus, the traditional setting is highly ill-posed and is very likely to result in local optimum (i.e., $\widehat{x}^a$ generated from $\mathcal{G}$ could not approach x^c enough). On the other hand, in our pixel-level constraint, we set $\|\widehat{x}^a - \widehat{x}^c\|$ as the objective and the distance between $\widehat{x}^a$ and $\widehat{x}^c$ is relatively narrower than the distance between $\widehat{x}^a$ and x^c since $\widehat{x}^a$ as well as $\widehat{x}^c$ locate in the same output space of $\mathcal{G}$. Thus, our setting has a smaller solution space and is relatively less ill-posed to avoid local optimum (i.e., it is simpler to train $\mathcal{G}$ so that $\widehat{x}^a$ is very close to $\widehat{x}^c$). Moreover, the target model $\mathcal{O}$ can perform well on both x^c and $\widehat{x}^c$ which has been validated by experiments, and $\widehat{x}^a$ obtained from our setting is close to $\widehat{x}^c$ (in fact, $\widehat{x}^c$ is very close to x^c, and $\widehat{x}^a$ is very close to $\widehat{x}^c$, thus $\widehat{x}^a$ is also close to x^c). Therefore, $\mathcal{O}$ can behave better on $\widehat{x}^a$ with our setting compared with the traditional setting.

Visual Analysis–Visualizations for Distribution Alignment Besides the t-SNE visualizations for the classification task in the manuscript, we provide the t-SNE visualizations for the semantic segmentation and object detection tasks as shown in Fig. 6. It is obvious that the distributions of $\widehat{x}^c$ and $\widehat{x}^a$ are close to the distribution of x^c, which remedies the distribution gap between clean samples and adversarial samples. The adversarial samples in the semantic segmentation task are obtained by BIM with hyper-parameters as $\varepsilon = 0.03 \times 255$, $\alpha = 0.01 \times 255$ and $n = 3$ and the adversarial samples in the object detection task are obtained by “cls+loc” attack with perturbation as 8.

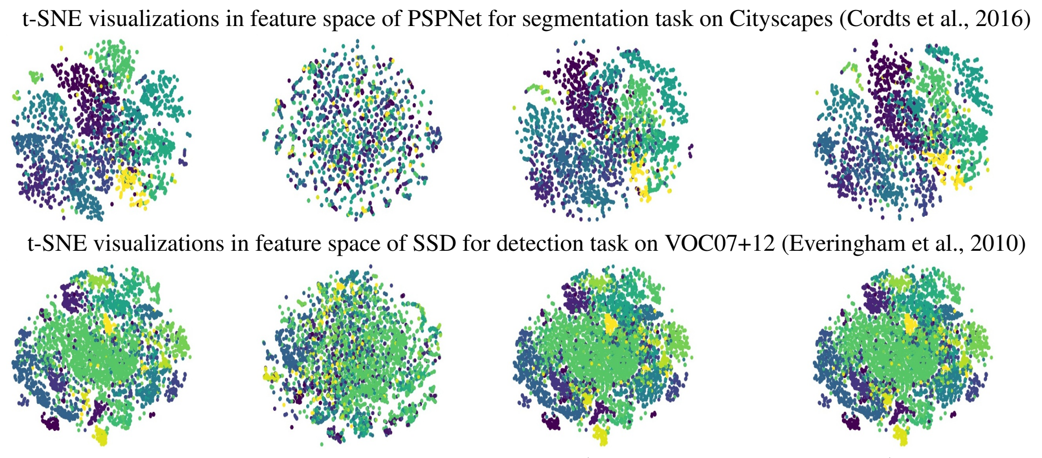
Visual Analysis–Illustrations for Three Tasks As exhibited in Fig. 7, 8 and 9, we provide visual illustrations which support the conclusion that our trained generator $\mathcal{G}$ can lead to satisfactory results on adversarial samples in different tasks.

Table 15: Comparison among our approach, existing methods and ablation settings on the classification task, under the evaluation of model transfer setting. "WideResNet ⇒ ResNet50" means the generator is trained with the target model of WideResNet, while we evaluate its defense effect for ResNet50.

WideResNet ⇒ ResNet50	CIFAR10 (Accuracy %)				CIFAR100 (Accuracy %)			
	clean	PGD	DeepFool	C&W	clean	PGD	DeepFool	C&W
$\mathcal{L}_I$	90.4	44.5	87.6	87.7	67.9	29.6	63.4	64.2
$\mathcal{L}_{F(w/o\ c)}$	87.2	17.2	17.7	17.9	61.8	23.8	28.2	28.7
$\mathcal{L}_{I+F(w/o\ c)}$	90.6	67.1	88.0	88.6	68.1	40.1	64.1	65.5
$\mathcal{L}_T$	**92.8**	18.1	87.0	86.7	71.2	14.9	62.6	63.4
$\mathcal{L}_{T+F(w/o\ c)}$	92.5	59.9	87.0	87.0	70.3	36.6	63.4	65.2
NRP (Naseer et al., 2020)	91.9	60.4	87.3	88.0	70.3	36.2	63.5	64.6
Denoise (Liao et al., 2018)	89.8	73.0	87.7	87.1	67.3	35.7	63.7	64.2
APE (Shen et al., 2017)	90.2	10.1	86.2	86.4	**73.3**	11.4	64.8	65.4
FPD (Li et al., 2020)	48.6	44.3	44.4	44.4	52.6	30.5	31.1	31.2
Defense (Samangouei et al., 2018)	39.9	38.5	39.8	39.9	31.2	21.5	22.8	22.5
SR (Mustafa et al., 2019)	48.0	44.8	45.2	45.3	33.6	24.8	25.5	25.5
Ours	90.7	**73.3**	**88.4**	**89.1**	68.7	**42.0**	**64.9**	**65.8**

Table 16: Comparison among our approach, existing methods and ablation settings on the semantic segmentation task, under the evaluation of model transfer setting. "PSPNet ⇒ DeepLabv3" means the generator is trained with the target model of PSPNet, while we evaluate its defense effect for DeepLabv3.

PSPNet ⇒ DeepLabv3	Cityscapes (mIoU %)				VOC2012 (mIoU %)			
	clean	BIM	DeepFool	C&W	clean	BIM	DeepFool	C&W
$\mathcal{L}_I$	61.5	48.5	49.9	55.0	73.1	54.9	59.8	68.1
$\mathcal{L}_{F(w/o\ c)}$	60.4	57.1	58.5	62.9	47.4	45.0	45.2	45.4
$\mathcal{L}_{I+F(w/o\ c)}$	64.9	57.8	61.3	64.2	70.3	59.9	63.2	68.2
$\mathcal{L}_T$	63.8	43.0	48.9	54.7	74.1	33.2	49.9	53.7
$\mathcal{L}_{T+F(w/o\ c)}$	65.0	53.8	59.9	63.0	71.2	55.1	60.3	66.8
NRP (Naseer et al., 2020)	65.0	53.4	47.8	64.1	70.6	53.1	51.2	67.4
Denoise (Liao et al., 2018)	64.4	52.7	51.1	64.4	70.4	60.1	53.7	67.8
APE (Shen et al., 2017)	54.5	28.5	26.8	40.0	**74.3**	25.7	46.9	44.5
FPD (Li et al., 2020)	55.9	51.5	51.7	53.6	61.5	57.7	59.3	60.6
Defense (Samangouei et al., 2018)	22.2	20.1	19.7	21.7	23.5	21.6	21.2	23.1
SR (Mustafa et al., 2019)	60.7	41.6	40.5	42.5	71.1	52.1	56.9	66.8
Ours	**67.6**	**59.0**	**61.7**	**65.3**	71.3	**60.5**	**64.6**	**68.9**



(a) x^c in $\mathcal{O}$ (b) x^a in $\mathcal{O}$ (c) $\widehat{x}^a$ in $\mathcal{O}$ (d) $\widehat{x}^c$ in $\mathcal{O}$

Figure 6: Target model $\mathcal{O}$ has admirable distributions for clean samples x^c while disordered distributions for adversarial samples x^a. Our $\mathcal{G}$ can turn x^c/x^a into $\widehat{x}^c$/$\widehat{x}^a$ with corrected distrubutions.

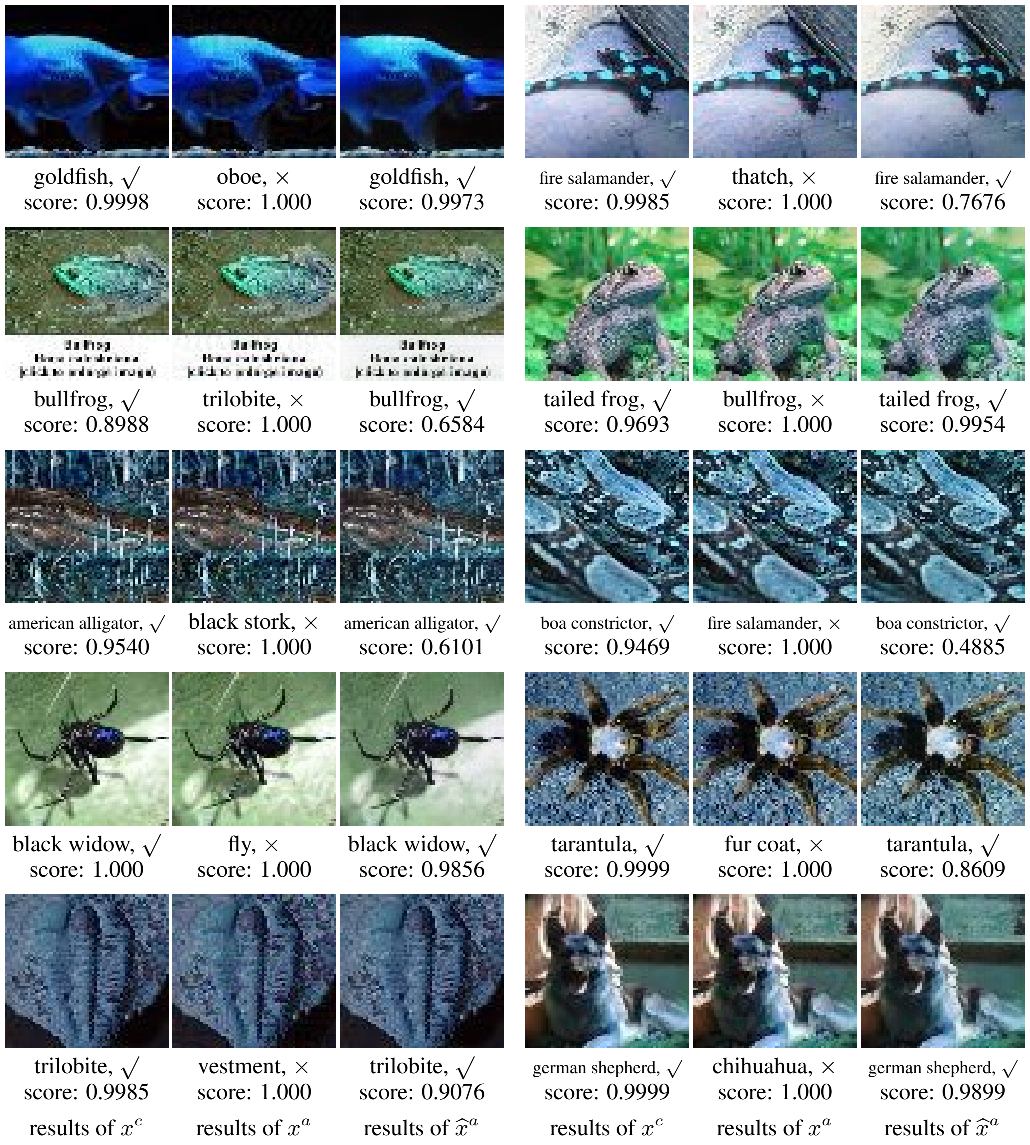

goldfish, √ score: 0.9998
oboe, × score: 1.000
goldfish, √ score: 0.9973
fire salamander, √ score: 0.9985
thatch, × score: 1.000
fire salamander, √ score: 0.7676
bullfrog, √ score: 0.8988
trilobite, × score: 1.000
bullfrog, √ score: 0.6584
tailed frog, √ score: 0.9693
bullfrog, × score: 1.000
tailed frog, √ score: 0.9954
american alligator, √ score: 0.9540
black stork, × score: 1.000
american alligator, √ score: 0.6101
boa constrictor, √ score: 0.9469
fire salamander, × score: 1.000
boa constrictor, √ score: 0.4885
black widow, √ score: 1.000
fly, × score: 1.000
black widow, √ score: 0.9856
tarantula, √ score: 0.9999
fur coat, × score: 1.000
tarantula, √ score: 0.8609
trilobite, √ score: 0.9985
vestment, × score: 1.000
trilobite, √ score: 0.9076
german shepherd, √ score: 0.9999
chihuahua, × score: 1.000
german shepherd, √ score: 0.9899
results of x^c
results of x^a
results of $\widehat{x}^a$
results of x^c
results of x^a
results of $\widehat{x}^a$

Figure 7: The visual illustration for the results of x^c, x^a and $\widehat{x}^a$ with our trained generator $\mathcal{G}$, in image classification on ImageNet. The adversarial samples are obtained by PGD with hyper-parameters as $\epsilon = 0.031 \times 255$, $\alpha = 0.0075 \times 255$ and $n = 8$.

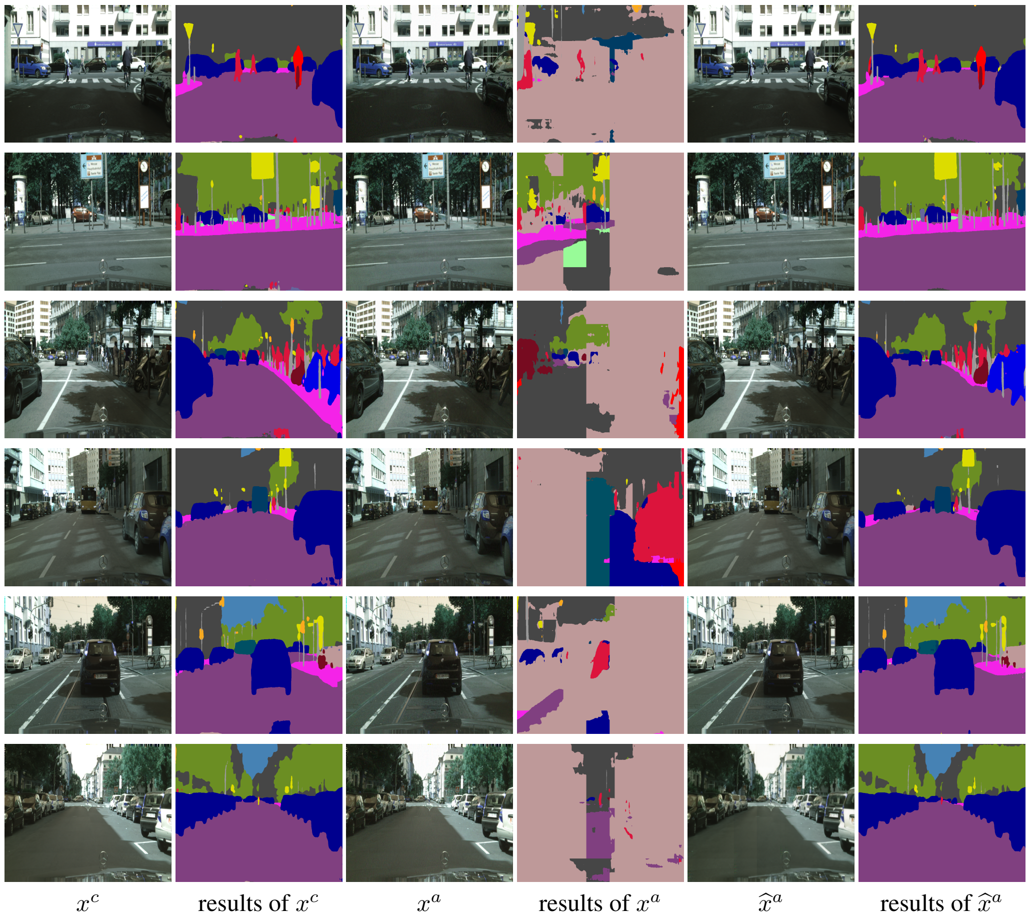


Figure 8: The visual illustration for the results of x^c, x^a and $\widehat{x}^a$ with our trained generator $\mathcal{G}$, in semantic segmentation on Cityscapes. The adversarial samples are obtained by BIM with hyper-parameters as $\epsilon = 0.03 \times 255$, $\alpha = 0.01 \times 255$ and $n = 3$.

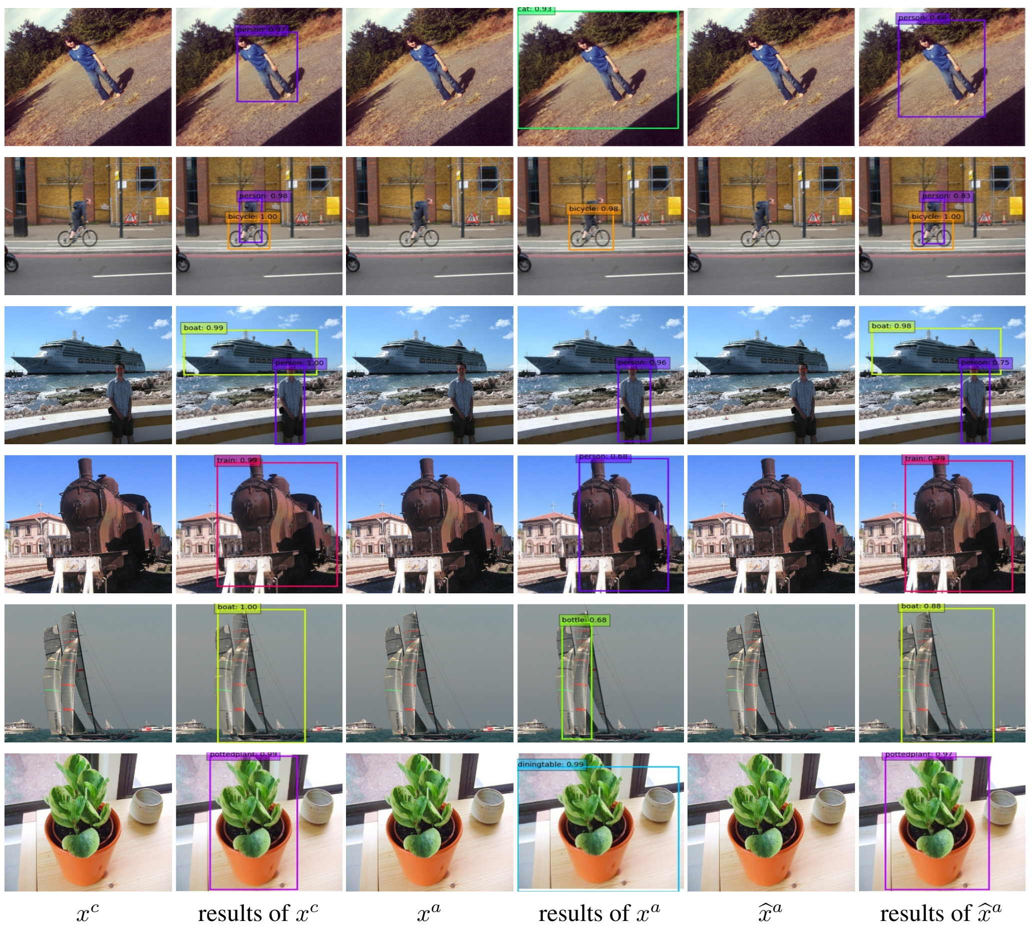


Figure 9: The visual illustration for the results of x^c, x^a and $\widehat{x}^a$ with our trained generator $\mathcal{G}$, in object detection on VOC07+12. The adversarial samples are obtained by "cls+loc" attack with perturbation as 8.