Applied Intelligence (2025) 55:462
https://doi.org/10.1007/s10489-025-06246-2

IPAttack: imperceptible adversarial patch to attack object detectors

Yongming Wen[1,2] · Peiyuan Si[3] · Wei Zhou[1] · Zongheng Zhao[1] · Chao Yi[1] · Renyang Liu[4]

Accepted: 30 December 2024

Abstract
With the widespread application of deep learning, general object detectors have become increasingly popular in our daily lives. Extensive research, however, has shown that existing detectors are vulnerable to patch-based adversarial attacks, which fool such detectors by crafting adversarial patches. Although existing methods have made significant progress in terms of attack success rate, they still suffer from a highly perceptible problem, making it easy for humans to distinguish these evil examples. To address this issue, in this paper, we propose a novel spatial transform-based end-to-end patch attack method, called IPAttack, to synthesize imperceptible adversarial patches. Our approach estimates a flow field $\boldsymbol{f}$ to formulate adversarial examples rather than introduce small L_p-norm constrained external perturbations. Besides, to improve the imperceptibility and maintain a high attack performance, we propose the Object Detector Class Activation Map (OD-CAM) for objectors to extract the most interesting region, which will be applied to spatial transform to generate the final adversarial examples. Extensive experiments demonstrate that the proposed IPAttack can generate patch-wised adversarial examples with high imperceptibility while achieving the best attack performance compared to existing methods.

Keywords Adversarial attack · Adversarial patch · Object detectors · Spatial transform · OD-CAM

1 Introduction

Deep learning techniques have been applied to a huge number of real applications and achieved great success, such as face recognition [1–4], image classification [5, 6], object detection [7–10], semantic segmentation [11–14] et. al.. The deep neural network (DNN), however, can easily be fooled to make the wrong decisions and encounter adversarial examples, which are formulated by some well-designed small perturbations [15–17] or patches [18, 19] to the original clean data. Such a fooling means is dubbed an adversarial attack [20–22], which reveals the vulnerability of existing neural networks. Therefore, a deep study of adversarial attacks needs to be conducted to understand the characteristics of neural networks [23] and further improve the robustness of neural networks [24–26].

As one of the important tasks of computer vision, object detection is widely used in unmanned driving systems [27] et al. The common object detectors can be divided into the one-stage-based and the two-stage-based. The one-stage object detector, like YOLOv4 [7] and et al., is a regression-based model, and the two-stage object detector is a proposal-based model, such as Faster R-CNN [8]. Compared to image classifiers, object detectors are more complicated and challenging to attack. Generally, adversarial attacks against object detectors can be divided into the following two types. The first type disturbs the whole image with external perturbations and uses L_p-norm to constrain the magnitude of the perturbations, such as DAG [28], UEA [29] and MI [30]. The other type is formulating the adversarial examples with a lit-

✉ Chao Yi
yichao@ynu.edu.cn

✉ Renyang Liu
ryliu@nus.edu.sg

Yongming Wen
wenyongming@csnwd.com.cn

Peiyuan Si
peiyuan001@ntu.edu.sg

Wei Zhou
zwei@ynu.edu.cn

Zongheng Zhao
zhaozongheng@mail.ynu.edu.cn

1 Engineering Research Center of Cyberspace, Yunnan University, Kunming 650500, Yunnan, China

2 China South-to-North Water Diversion Group Water Networks Intelligent Technology Co., Ltd., Beijing, China

3 Nanyang Technological University, Singapore 639798, Singapore

4 National University of Singapore, Singapore 117602, Singapore

Published online: 19 February 2025

tle sticker to avoid distorting the whole image, such as Dpatch [31], IPatch [32], DPAttack [33], RPAttack [34].

Although image-size-based perturbation attack is an effective way to fool object detectors in the digital world, they are impractical for attacking object detectors deployed in the physical world. On the other hand, the L_p-norm beyond adversarial patches is too conspicuous and can be easily detected by the naked human eye. Besides, for patch-based attacks, determining the optimal area to paste the well-calculated patch and the patch size is another big challenge. However, the current patch-based adversarial attacks for object detectors can not adequately deal with these issues well.

To bridge this gap, we propose an end-to-end attack for object detectors called IPAttack, which can generate imperceptible adversarial patches and guarantee attack performance at the same time. This type of attack can further reveal the vulnerability of neural networks. For imperceptibility, we use spatially transformed [35–37] techniques to optimize our patch, which is formulated by changing each pixel's position. To obtain such an adversarial patch, IPAttack first optimizes a flow field matrix $\boldsymbol{f}$, which has the same size as the patch, and its value represents the direction and magnitude of each pixel's transformation. Once the flow field is well-calculated, we can apply it to the pre-set area to obtain the optimal adversarial patch. On the other hand, as aforementioned, the patch area selection is an essential aspect to guarantee the attack performance; thus, in this paper, we introduce a new CAM-based [38–40] scheme, namely OD-CAM, for object detector to extract the interested area as the optimal area.

Usually, the object detector will perform post-processing after the model output. In such a process, the bounding boxes with a confidence score, i.e., greater than the preset threshold, will be retained, while the others will be discarded, and then an NMS operation [41] will be performed to obtain the final detection results. To hide the objects in the image from the object detector, our goal is to lower the confidence score of the bounding boxes higher than the threshold through optimization, resulting in the object detector not recognizing any objects.

As shown in Fig. 1, we illustrate the adversarial examples generated by DPAttack, Object Hider, RPAttack and IPAttack, and find that our proposed method alters the attack area slightly, thus illustrating the invisibility of the adversarial patches. Furthermore, the extensive statistical results on two distinct object detection benchmark datasets demonstrate that the IPAttack significantly improves the quality of the generated adversarial examples while guaranteeing a high attack success rate compared to existing patch-based methods on target models while outperforming them concerning attack success rate. The main contributions can be summarized as follows:

- We propose a novel and imperceptible adversarial patch attack against object detectors, called IPAttack, by introducing spatially transform techniques. As far as we know, IPAttack is the first work to apply spatial transformation to build adversarial patches and achieve good results.
- We innovatively designed a method named OD-CAM to obtain the regions of interest of the object detector in the image; with the help of OD-CAM, we can accurately determine the patch size and further find the optimal position to paste the calculated patches to achieve a satisfactory attack performance.

Fig. 1 Illustration of adversarial examples. The first one is the original image, and the following are the prediction box and confidence score and the category predicted by YOLOv4, and the adversarial images generated by DPAttack, Object Hider, RPAttack, and IPAttack(ours), respectively. As we can see, the adversarial example generated by our method can successfully escape from detecting YOLOv4; besides, its perturbation is more imperceptive compared to other methods

- Extensive empirical results on two object detection datasets and two different types of object detectors conclusively demonstrate that the proposed IPAttack can remarkably improve the generated adversarial examples' imperceptibility and image quality and efficiently reduce the mAP of the object detectors.

The rest of this paper is organized as follows: we first briefly review the methods associated with Object Detection, Adversarial Patch, and Class Activate Map in Sec. 2. Then, Sec. 3 introduces the details of the proposed IPAttack. Finally, the experimental results are presented in Sec. 4, with the conclusion drawn in Sec. 5.

2 Related work

2.1 Object detection

In recent years, remarkable advancements have been made in the field of general object detection. Among them, the deep learning-based methods [7, 8], which greatly improve the detection performance, have attracted more and more attention. Existing deep learning-based object detectors comprise the following two parts: the backbone part, which is usually pre-trained on ImageNet [42] and used to extract the image's feature; and the other part is the head, which is used to predict classes and bounding boxes of objects. Moreover, the object detectors can be divided into one-stage and two-stage object detectors depending on the head part. The most representative one-stage object detector is YOLOv4 [7] and SSD [43]. As for the two-stage object detector, the most representative ones are RCNN [44] and Faster R-CNN [8]. Specifically, the one-stage object detector directly regresses the bounding box and predicts the class probability regarding the input image, while the two-stage object detector first produces region proposals and then uses the detector head to classify each region proposal. In the post-processing stage, these two types of detectors will first remove the bounding box whose confidence score is lower than the pre-defined threshold and then perform non-maximum suppression (NMS) to get the final detection result. In this paper, to evaluate the proposed attack method, we selected two kinds of object detectors as our victim models, i.e., YOLOv4 and Faster R-CNN.

2.2 Adversarial patch

Brown et al. [18] first proposed the patch-based adversarial attack to generate adversarial examples for the image classifier. Subsequent works [18, 45] also centered their focus on deceiving Deep Neural Networks (DNNs) in the context of image classification. DPatch [31] first proposed adversarial patches for object detection tasks and attained better attack effects by increasing the confidence of regression boxes that contain patches. DPAttack [33] optimizes the asteroid-like diffusion patches with an I-FGSM [46] style to obtain the optimal position according to the returned detection results to paste patches to hide objects of the input image. Object Hider [47] first computes image-size perturbations and then uses a consensus-based algorithm and takes a vote to obtain the grid-like patch mask to formulate the final adversarial example. Besides, RPAttack [34] proposes a sparse patch-based attack method to search for the minimal patch mask by reducing the modified area iteratively. Although the final pasted patch only influences a few pixels, it can still be recognized by the human eyes.

Compared with the previous works, our proposed IPAttack can produce more imperceptibly adversarial patches, which can easily deceive the human eyes and can achieve better attack performance.

2.3 Class Activation Map

In recent years, researchers have introduced various techniques to enhance the transparency of convolutional neural networks (CNNs) by visualizing the regions of interest in input images that are crucial for making predictions or providing visual explanations. Among these techniques, Class Activation Mapping (CAM) [38–40] is one of the most widely recognized and extensively used methods in computer vision. CAM provides visual explanations of CNNs by highlighting the most critical regions in an input image for the network's predictions. It achieves this by generating a weighted combination of activation maps from convolutional layers. These visualization techniques have proven to be valuable tools for understanding CNN's decision-making processes, thereby improving their transparency and interoperability.

However, existing CAM-based methods are specifically designed for classification models and are not directly applicable to object detectors due to fundamental differences in their architecture and outputs. Classification models typically involve a single classification loss, with the final prediction being a single class confidence score. In contrast, object detectors generate multiple class scores along with the corresponding object positions. Furthermore, object detection models, such as Faster R-CNN and YOLOv4, employ diverse modeling approaches and produce varied outputs, making it challenging to develop a unified CAM-based method for object detectors. Additionally, in object detection tasks, overlapping regions often occur among different objects within an image, meaning that the final heatmap cannot simply be derived by summing the individual object heatmaps.

To address these challenges, we propose a novel approach called OD-CAM. This scheme is designed to identify the

Table 1 Important symbols in this paper

$\boldsymbol{x}$	image	L	the coordinate
$\boldsymbol{x}_{adv}$	adversarial example	H	heatmap group
b	object in bounding box	y^c	the highest confidence score of the object in the bounding box
$P_{initial}$	initialization patch	P_{adv}	adversarial patch
P^i_{adv}	the pixel value of the i-th pixel on the patch	P_i	the pixel value of the i-th pixel of the initialization patch
$\boldsymbol{f}$	flow field	t	pre-defined threshold
D^C_i	the confidence score of the object in the i-th bounding box	lr	learning rate
$\mathcal{L}_{adv}$	adversarial loss	$\mathcal{L}_{flow}$	flow loss
α	adversarial loss hyper-parameters	β	flow loss hyper-parameters

regions of interest for object detectors by first generating a group of heatmaps and then refining them through a process of judgment and redundancy elimination. This results in a final, comprehensive heatmap that accurately represents the regions of interest for the object detector.

3 Methodology

In this chapter, we introduce the IPAttack in detail. First, we define the problem in Sec. 3.1. Then, in Sec. 3.2, we introduced an overview of IPAttack. Then, in Sec. 3.3, we propose the OD-CAM method for selecting patch positions. Finally, we introduce the generation of the adversarial patch by leveraging spatial transform and the design of the object function in Secs. 3.4 and 3.5.

3.1 Problem formulation

In this paper, we propose an attack on object detectors, specifically targeting YOLOv4 and Faster R-CNN. Given a well-trained detector $\mathcal{D}$ and an image $\boldsymbol{x}$ containing at least one object detectable by $\mathcal{D}$, our objective is to cause the detector to fail in identifying the object(s) present in $\boldsymbol{x}$.

One-stage detectors such as YOLOv4 directly predict the bounding box coordinates, which indicate the object's location in the image, and simultaneously estimate the class probabilities of objects in the input image. In contrast, two-stage detectors like Faster R-CNN first generate region proposals where objects might exist and subsequently refine these proposals using a detection head. This process produces the final confidence scores, which quantify the certainty of detection, along with the bounding box coordinates. Both types of detectors employ non-maximum suppression (NMS) to obtain the final detection results, including the regression box and class confidence scores.

We define $R = \{R_1, R_2, R_3, \ldots, R_m\}$ as the set of detection results from detector $\mathcal{D}$ applied to image $\boldsymbol{x}$, such that $\mathcal{D}(\boldsymbol{x}) = R$. Here, $R_i = \{L, C\}$, where L represents the coordinates of the detection result (i.e., the bounding box specifying the object's location in the image, such as the corners of the box), and C is the confidence score of the detection, ranging from 0 to 1, with higher values indicating greater certainty. As m decreases, the number of detected objects by $\mathcal{D}$ diminishes.

Our goal is to create an almost imperceptible adversarial patch that reduces the number of detected objects in the image as much as possible. To achieve this, we formulate our attack as the following optimization problem:

$$R_{\min(m)} + \min(\boldsymbol{x}_{adv} - \boldsymbol{x}) \tag{1}$$

where $R_{\min(m)}$ represents the smallest number of detection results in the adversarially modified image $\boldsymbol{x}_{adv}$, as opposed to parameters like L and C, which pertain to individual detection details. Minimizing $R_{\min(m)}$ reduces the number of detected objects, thereby lowering the detector's overall confidence in identifying objects. Simultaneously, minimizing $\boldsymbol{x}_{adv} - \boldsymbol{x}$ ensures that the adversarial patch remains visually inconspicuous, satisfying the essential requirements of our attack strategy.

Table 1 summarizes the key symbols and their meanings used in this article.

3.2 Overview of IPAttack

The overview of IPAttack is illustrated in Fig. 2, and the attack framework is primarily divided into two steps: 1) In the first step, OD-CAM is utilized to identify the region of interest for the target detector within the input image to determine the position of the patch, i.e., the mask. 2) An initial adversarial patch is generated from the input image with such a mask. By optimizing the loss function, the candidate adversarial patch is refined by spatial transformation until it can attack the detector successfully.

3.3 Patch location

Recall that the patch's position is essential to the attack performance. In this section, we explain how to find the optimal patch location with the proposed OD-CAM to paste the patch.

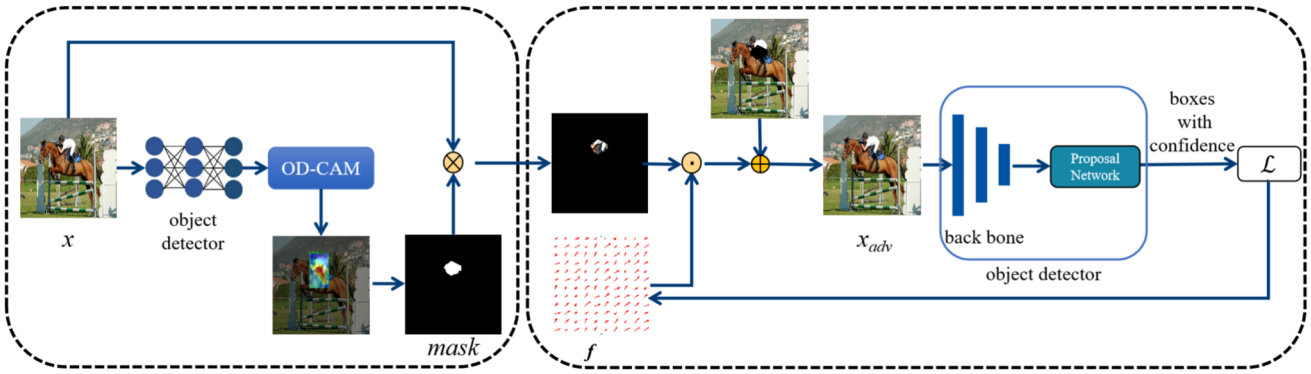

Fig. 2 Overview of IPAttack, where ⊗ represents the product of elements, and ⊙ represents the spatially transformed operation. ⊕ represents the sum of elements

OD-CAM is a visual output of intermediate convolution layers of detectors, which involves the idea of a heatmap that could help to indicate the important area of the input image more obviously. Add the patch to the area obtained by OD-CAM, which has the greatest impact on the model, in order to improve the success rate of the attack. A heatmap is the output of the gradient function with respect to the intermediate convolution layers. Next, we detail the process of obtaining such a heatmap for one-stage and two-stage detectors, YOLOv4 and Faster R-CNN, respectively.

3.3.1 Obtaining the heatmap

Mathematically, we use the following formula to calculate the heatmap:

$$H = \left\{ H_{n_{head}} \right\}, \tag{2}$$

where n_head represent the number of detector header, i.e., $n_{head} = \{1, 2, 3\}$ in YOLOv4 and $n_{head} = \{1\}$ in Faster-RCNN.

$$H_{n_{head}} = \sum_{b \in B} h_b, \quad w.r.t\ b \in e\ (e \in E), \tag{3}$$

where e represents a detection head, E represents the set of detection heads, b are the objects in bounding box, B represents the set of object in bounding boxes detected by detection head e in the image, and h_b is the heatmap of single bounding box b, $b \in e$ means object in bounding box b is detected by the detection head e. Specifically, we use the following formula to calculate h_b:

$$h_b = \max\left(0, w_k^c \otimes A_{ij}^k\right), \tag{4}$$

$$w_k^c = \sum_i \sum_j \alpha_{ij}^{kc} \cdot \text{relu}\left(\frac{\partial y^c}{\partial A_{ij}^k}\right), \tag{5}$$

$$\alpha_{ij}^{kc} = \frac{\frac{\partial^2 y^c}{\left(\partial A_{ij}^k\right)^2}}{2\frac{\partial^2 y^c}{\left(\partial A_{ij}^k\right)^2} + \sum_a \sum_b A_{ab}^k \left\{ \frac{\partial^3 y^c}{\left(\partial A_{ij}^k\right)^3} \right\}}, \tag{6}$$

where A_{ij}^k represents the k-th active layer and position (i, j) of the convolution layer on the channel, w_k^c represents the weight corresponding to the k-th active layer, ⊗ represents the element product, y^c is the highest confidence score of the object in the bounding box. Here, it should be noted that (i, j) and (a, b) are regarded as the same concept.

3.3.2 Heatmap for YOLOv4

The process of OD-CAM for the YOLOv4 is as follows:

Firstly, YOLOv4 generates detection outcomes, consisting of bounding boxes around objects in the image along with the highest confidence scores for each object. This can be thought of as identifying objects in the image, marking their locations with bounding boxes, and attaching a confidence level to each detection–higher scores indicate greater certainty of the object's presence.

Next, the confidence scores of the detected objects are backpropagated to the three detection heads of the model, resulting in the creation of heatmap groups, by (4), (5) and (6). A heatmap acts as an "importance map", where brighter or hotter areas represent regions where the model concentrated its focus for object detection. Each heatmap group denoted as H, contains m heatmaps, where m represents the number of objects detected in the image. Thus, this step produces a total of $3 * m$ heatmaps.

Subsequently, the bounding box coordinates of each detected object are used to refine the heatmaps. For the j-th object (where j can be any number ranging from 1 to m), the region outside the corresponding bounding box on the

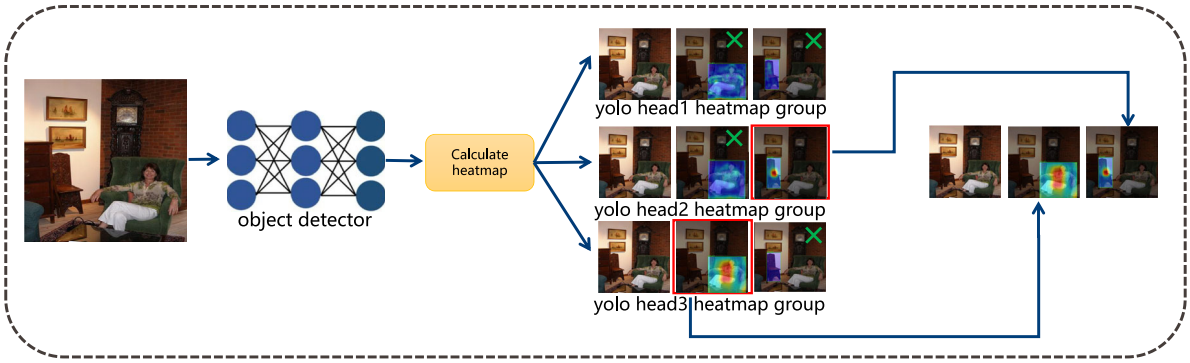

Fig. 3 Obtaining the heatmap group through the OD-CAM for YOLOv4. The red box represents the retained heatmap, and the × represents the discarded heat map

heatmap is zeroed out, leaving only non-zero values within the bounding box. This process is repeated for all objects, resulting in $3 * m$ heatmaps with active regions confined to their respective bounding boxes.

Finally, By (3), for each detected object, we determine which detection head (denoted as the i-th detection head, where $i \in \{1, 2, 3\}$) was responsible for detecting the j-th object. Once identified, only the heatmaps corresponding to that object within the specific heatmap group of the detection head are retained, while the others are discarded. This results in a refined heatmap group containing m heatmaps, which provides a clearer understanding of how the model focused on detecting the objects in the image.

The process of obtaining the YOLOv4 heatmap group is shown in Fig. 3.

3.3.3 Heatmap for faster R-CNN

The process of OD-CAM for the Faster R-CNN is as follows:

First, similar to YOLOv4, the detection results of the Faster R-CNN, that is, the corresponding bounding boxes and the highest confidence score of each object on the image, are obtained.

Secondly, the confidence scores of each object are back-propagated to the ROIs of the Faster R-CNN, and the heatmap group contains the heatmap of m objects detected by the detector in the image. We calculate each heatmap by (4), (5) and (6);

Finally, obtain the coordinates of the bounding box of each object in image, and intersect the bounding box of the j-th object with the bounding box of the object, and obtain m heatmaps with non-zero values in the bounding box and zero values in other parts. The process of obtaining a Faster R-CNN heatmap group is shown in Fig. 4

We select the part in each heatmap group that is greater than the preset threshold value as the $mask$ (we set the threshold value to 0.5 in our experiments), and then add these $masks$ together to form the final $mask$.

3.4 Imperceptible patch

In this subsection, we introduce the details of generating imperceptible patches by leveraging spatial transform techniques. We initialize the adversarial patch as $P_{initial} = M(\boldsymbol{x})$, which is part of the benign image. We use P_i to represent the pixel value of the i-th pixel of the initialization patch and (u^i, v^i) to represent the coordinates of the pixel on the image. We initialize a flow field $\boldsymbol{f}$ with the same size as the original image. The vector $\boldsymbol{f_i} = (\triangle u^{(i)}, \triangle v^{(i)})$ represents the i-th vector in the flow field $\boldsymbol{f}$, where $\triangle u^{(i)}$ represents the offset along the positive x-axis, $\triangle v^{(i)}$ represents the offset along the negative y-axis. We apply the flow field $\boldsymbol{f}$ to the initial patch to formulate the adversarial patch, that is, $P_{adv} = \boldsymbol{f} \odot M(\boldsymbol{x})$. Similarly, we define P^i_{adv} as the pixel value of the i-th pixel on the patch and use (u^i_{adv}, v^i_{adv}) to represent the coordinates of the pixel on the image. So we can get the value P^i_{adv} by transforming the value of the j-th pixel

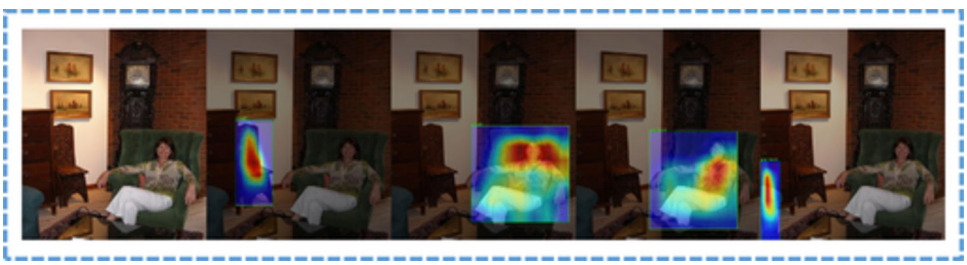

Fig. 4 Obtaining the heatmap group through the OD-CAM for Faster R-CNN

of the initialization patch to the i-th pixel of the adversarial patch, where the coordinate transfer between the j-th pixel of the original sample and the i-th pixel of the adversarial example is $(u^i_{adv}, v^i_{adv})=(u^i + \triangle u^{(i)}, v^i + \triangle v^{(i)})$.

Since (u^i, v^i) can be decimals and may not be on the integer image grid, we refer to [35] to apply differentiable bilinear interpolation [48] to the transformed input image $\boldsymbol{x}$. We calculate P^i_{adv} following the method described in [35]:

$$P^{(i)}_{adv} = \sum_{q \in \mathcal{Z}(u^{(i)}, v^{(i)})} p^{(q)} \left(1 - \left|u^{(i)} - u^{(q)}\right|\right) \cdot \left(1 - \left|v^{(i)} - v^{(q)}\right|\right), \tag{7}$$

where $\mathcal{Z}(u^{(i)}, v^{(i)})$ is the index of four pixels which are around $(u^{(i)}, v^{(i)})$. We calculate the value of each pixel within the adversarial patches to obtain the final adversarial patch through (7). Then, our final adversarial example can be defined as:

$$\boldsymbol{x}_{adv} = P_{adv} + (1 - mask) * \boldsymbol{x}. \tag{8}$$

3.5 Object function

The objective of our imperceptible patch is to make the object detector unable to detect objects in the image. We are well aware that in the detector's post-processing stage, the bounding box whose confidence score is less than the thresholds will be eliminated first, and then the NMS will be performed. Therefore, the purpose of our adversarial loss is to lower the corresponding confidence score before performing the NMS. We express the adversarial loss as:

$$\mathcal{L}_{adv} = \frac{\sum_i^N \max\left(0, D_i^C\left(x_{adv}\right) - t\right)}{S}, \tag{9}$$

where N is the bounding box before the post-processing detected by the detector on the adversarial example, i is the i-th bounding box, D_i^C is the confidence score of the object in the i-th bounding box, and t is the pre-defined threshold for detector post-processing stage, which is usually the same as the confidence score threshold of object detectors. S is the number of bounding boxes whose confidence score is higher than the threshold, that is, only the bounding box of positive examples will be used to cause the loss value. In order to make our adversarial patch more imperceptible, we also introduce the flow loss $\mathcal{L}_{flow}$ [35] to constrain the calculated flow field $\boldsymbol{f}$.

$$\mathcal{L}_{\text{flow}}(\boldsymbol{f}) = \sum_{p}^{all\ pixels} \sum_{q \in \mathcal{Z}(p)} \sqrt{\left\|\Delta u^{(p)} - \Delta u^{(q)}\right\|_2^2 + \left\|\Delta v^{(p)} - \Delta v^{(q)}\right\|_2^2}, \tag{10}$$

where p is a given arbitrary pixel point, and $q \in \mathcal{Z}(p)$ is the adjacent point of pixel point p. As we implement a local smooth spatial transformation perturbation $\mathcal{L}_{flow}$ based on the total variation, we use $\mathcal{L}_{\text{flow}}$ to calculate the sum of the spatial moving distance of any two adjacent pixels. Therefore, the final loss function is defined as:

$$\mathcal{L} = \alpha * \mathcal{L}_{adv} + \beta * \mathcal{L}_{flow}, \tag{11}$$

where α and β are hyper-parameters of $\mathcal{L}_{adv}$ and $\mathcal{L}_{flow}$. We use the *Adam* optimizer [49] with learning rate $lr = 0.1$ to optimize our calculate flow field to obtain adversarial patches and finally obtain adversarial examples.

Algorithm 1 Imperceptible adversarial patch for object detectors.

Input: D: the victim model to be attacked; α: the learning rate; T: the maximal optimization iterations; $\boldsymbol{x}$: the clean image with at least one object detected D; $\boldsymbol{x}_{adv}$:adversarial examples;
Output: The adversarial example with imperceptible patches $\boldsymbol{x}_{adv}$ is used for attack.
Parameter: The flow field $\boldsymbol{f}$.

```
Initialize the parameters of the flow f with zeros;
Obtain the Masked Image M(x) by OD-CAM;
for i = 1 to T do
    Compute x_adv by (8);
    Obtain the results of the detector before NMS i.e. D(x_adv);
    Compute L_adv by (9);
    Optimize f according to (11);
    if M = 0 then
        break
    end if
end for
```

The whole algorithm of IPAttack is listed in Alg. 1, which could help readers to re-implement our method step-by-step.

4 Experiments

In this section, we evaluate the proposed IPAttack on two object detection benchmark datasets. We first compare our proposed method with several baseline techniques concerned with Attack Success Rate (ASR) and the image quality of the adversarial examples on one-stage and two-stage object detectors.

4.1 Settings

Dataset We verify the performance of our method on two benchmark datasets for computer vision tasks, i.e., the VOC0712[1] dataset and the COCO2017[2] dataset [50]. In

[1] http://host.robots.ox.ac.uk/pascal/VOC/

[2] https://cocodataset.org

Table 2 Attack performance comparison based on YOLOv4 model trained on the VOC0712 dataset

Method	ASR	PSNR	SSIM
DPAttack	0.8200	33.4658	0.9712
RPAttack	0.9600	41.0980	0.9840
Object Hider	0.8750	29.4531	0.9434
Ours	**0.9850**	**41.6613**	**0.9928**

The best results are highlighted in bold

detail, the VOC0712 dataset is mainly composed of two parts: VOC2007 dataset [51] and VOC2012 dataset [52], spanning 20 classes. The COCO2017 dataset consists of 123,287 real-world images with different sizes, spanning 80 classes. We randomly select 1000 images from each dataset as benign images and ensure all these images can be correctly recognized by the victim model, i.e., at least one object can be detected.

Models We attack two types of detectors involving one-stage and two-stage, namely YOLOv4 and Faster R-CNN. We use the VOC0712 dataset and the COCO2017 dataset to train YOLOv4 and Faster R-CNN models, respectively. The mAP values of the YOLOv4 and Faster R-CNN on the VOC0712 dataset are 0.8904 and 0.7471, while on the COCO2017 dataset are 0.7024 and 0.5900, respectively. The input sizes are 416*416 for the YOLOv4 and 600*600 for the Faster R-CNN, respectively; in terms of the backbone network, CSPDarknet53 [7] is used in the YOLOv4 and ResNet-50 is used for the Faster R-CNN.

Baselines We choose the most comparable methods for attacking object detectors as our baselines, such as DPAttack [33], RPAttack [34], and Object Hider [47].

Metrics The evaluation metrics used in this part are as follows: to measure the attack performance, 1) the Attack Success Rate (ASR), which is calculated as $Q_{success}/Q$, where $Q_{success}$ is the number of samples successfully attacked and Q is the number of total samples). 2) The mAP is an all-important evaluation metric for object detectors, and we also use the mAP of object detectors to evaluate the performance of IPAttack and baselines.

Table 3 Attack performance comparison based on YOLOv4 model trained on the COCO2017 dataset

Method	ASR	PSNR	SSIM
DPAttack	**0.9040**	31.1714	0.9562
RPAttack	0.9000	37.5537	0.9670
Object Hider	0.6340	30.6835	0.9449
Ours	0.8650	**43.6336**	**0.9939**

The best results are highlighted in bold

Table 4 Attack performance comparison based on Faster R-CNN model trained on the VOC0712 dataset

Method	ASR	PSNR	SSIM
DPAttack	0.8150	34.0797	0.9731
RPAttack	0.9600	**41.0980**	0.9840
Object Hider	0.7310	31.1162	0.9468
Ours	**0.9810**	37.8593	**0.9856**

The best results are highlighted in bold

The other metrics used to evaluate the image quality of generated adversarial examples are included: 3) the Peak Signal-to-Noise Ratio (PSNR) is used to evaluate the image distortion and 4) the Structural Similarity Index (SSIM) is used to test the structural similarity between the generated adversarial examples and their clean counterparts. A higher ASR means better attack performance and a larger PSNR and SSIM indicate better image quality.

Implementation details In experiments, we set the maximum iteration number T = 1k, the optimizer uses $Adam$ [49] and the learning rate lr=0.1, we set t=0.5 in (9). In terms of the loss function's hyperparameters in our method, the specific settings are $\alpha = 1$ and $\beta = 0.0001$. We will discuss the specific settings for α and β in Sec. 4.5.1. All the indicators in terms of image quality and mAP are measured on saved JPEG images.

For the baselines, we perform attacks by using the code with default settings provided by the papers. All the experiments are conducted on a GPU server with 4 * Tesla A100 40GB GPU, 2 * Xeon Glod 6112 CPU, and RAM 512GB.

4.2 Quantitative comparison with the previous works

In this subsection, we will evaluate the proposed IPAttack and the baselines DPAttack, RPAttack, and Object Hider in ASR, PSNR, and SSIM on the VOC0712 dataset and COCO2017 dataset.

Tables 2 and 4 show the results of ASR, as well as the PSNR and SSIM, which are obtained by attacking YOLOv4 and Faster R-CNN on the VOC0712 dataset, respectively.

Table 5 Attack performance comparison based on Faster R-CNN model trained on the COCO2017 dataset

Method	ASR	PSNR	SSIM
DPAttack	0.4580	30.9781	0.9543
RPAttack	0.9000	37.5537	0.9670
Object Hider	0.6160	36.7802	0.9647
Ours	**0.9240**	**40.0242**	**0.9821**

The best results are highlighted in bold

Simultaneously, Tables 3 and 5 show the emprical results obtained by attacking YOLOv4 and Faster R-CNN on the COCO2017 dataset, respectively. As can be seen, IPAttack can improve the performance of baseline methods in all of the target models in terms of ASR, PSNR, and SSIM. It is worth noting that we maintain a higher ASR than the baseline method while its image quality far exceeds that of the baseline method. From Tables 2 and 4 we can see that in the case of the same dataset, the adversarial examples' PSNR and SSIM obtained by the proposed method to attack Faster R-CNN are much lower than those obtained by attacking YOLOv4, that is, the image quality is much worse. This is because Faster R-CNN is harder to attack than YOLOv4 in our method and requires more iterations. Note that the ASR and image quality, including PSNR and SSIM of YOLOv4, are the same as Faster R-CNN on the same dataset.

To better observe the difference between the adversarial examples generated by the proposed method and the baselines from the visual aspect, we draw the adversarial examples generated on VOC0712 by baselines and the proposed method in Fig. 5. The target model is YOLOv4. The first column is clean images, The second column is the results detected by YOLOv4, and the following are the adversarial images of Object Hider, DPAttack, RPAttack, and IPAttack, respectively. From Fig. 5, we can clearly see the patches on the adversarial examples generated by Object Hider and DPAttack and the scattered patches on the adversarial examples generated by RPAttack. However, for our method, we cannot even find the location of the patch, which means our generated adversarial patches are less perceptible and able to deceive the human eye.

Table 6 The mAP of YOLOv4 and Faster R-CNN

Method	YOLOv4		Faster R-CNN	
	mAP	decline	mAP	decline
Original	0.9051	–	0.8982	–
DPAttack	0.7778	0.1273	0.7833	0.1149
RPAttack	0.8997	0.0054	0.8830	0.0152
Object Hider	0.8130	0.0921	0.7539	0.1443
Ours	0.7471	**0.1959**	0.3389	**0.5593**

The best results are highlighted in bold

4.3 The comprising results of mAP

In Table 6, we list the decline of the mAP on the target detectors with adversarial examples generated by IPAttack and baselines. First, we obtain that the mAPs of YOLOv4 and Faster R-CNN on the 1000 clean images selected from the VOC0712 dataset reach 0.9051 and 0.8982, respectively. Then, we compute the mAPs of these two detectors on the adversarial examples generated by different methods.

From Table. 6, we can find that 1) our method significantly reduces the mAP of the target model compared to previous work. 2) Most existing attack methods show better performance when the object detector cannot detect objects in the image, but they cannot significantly reduce the mAP of the object detector.

4.4 Attack on YOLOv5 and YOLOv7

To further validate the effectiveness of the proposed IPAttack, we also evaluated its performance against two other widely

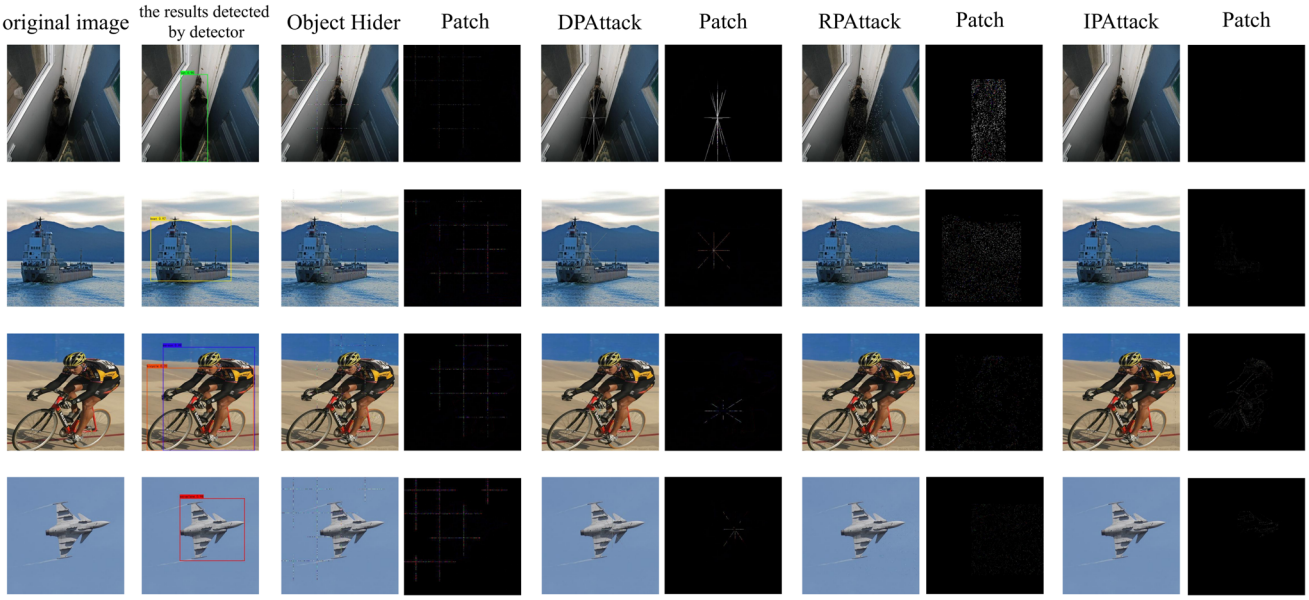

Fig. 5 The perceptibility comparison of generated adversarial images. The first column is clean images, the second column is the results detected by YOLOv4, and the following are the adversarial images and adversarial patches of Object Hider, DPAttack, RPAttack, and IPAttack, respectively

Table 7 IPAttack attack performance based on YOLOv5 model and YOLOv7 model

Model	YOLOv5		YOLOv7	
	VOC0712	COCO2017	VOC0712	COCO2017
ASR	0.9830	0.8439	0.9738	0.8347
PNSR	41.1464	43.8376	42.3283	41.5477
SSIM	0.9920	0.9856	0.9960	0.9896

used object detectors, namely YOLOv5 and YOLOv7, as shown in Table 7. By comparing the results presented in Table 7 with those in Tables 2 and 3, it becomes evident that, when using the same dataset, the success rate of IPAttack against the YOLOv4 model surpasses that against YOLOv5 and YOLOv7. This discrepancy can be attributed to the successive enhancements and optimizations incorporated into YOLOv5 and YOLOv7, which have significantly improved their robustness. These advancements have provided these models with stronger resistance mechanisms, enabling them to defend against attacks such as IPAttack more effectively compared to their predecessor, the YOLOv4 model.

4.5 Ablation study

In order to investigate the effects of hyper-parameters α and β, as well as OD-CAM, on the attack success rate, we conducted ablation experiments separately in this subsection.

4.5.1 Loss and hyper-parameters

The proposed method concerns the settings of the hyper-parameters, such as α and β, which will deeply affect the attack performance. We examine the influence of these factors on YOLOv4 and Faster R-CNN with the VOC0712 dataset and the COCO2017 dataset. During the attack, the maximal number of iterations is limited to 1000. It can be seen from Fig. 6 that ASR gradually increases along with β reduction. And the best ASR is achieved when the α=1 and β=0.0001.

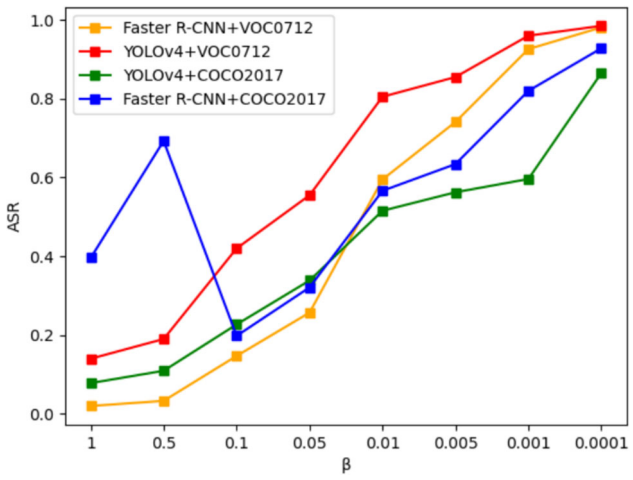

Fig. 6 The performance of IPAttack when using different hyper-parameters

Table 8 The attack performance comparison of IPAttack optimized with and without the OD-CAM in the Faster R-CNN model, *w.* means with OD-CAM, *w.o.* means without OD-CAM

Method	IPAttack *w.*		IPAttack *w.o.*	
	VOC0712	COCO2017	VOC0712	COCO2017
ASR	0.9810	0.9240	0.5620	0.2660
PSNR	37.8593	40.0242	38.2539	37.0001
SSIM	0.9856	0.9821	0.9870	0.9838

4.5.2 OD-CAM

To verify the effectiveness of OD-CAM, we conducted ablation experiments on the Faster R-CNN model with the VOC0712 and COCO2017 datasets. The attack with OD-CAM can output the position and the size of patches simultaneously, for the attack without OD-CAM, we set the patch size reading of the average of the patch sizes obtained by OD-CAM and set the patch position as the image center. And the number of iterations is the same as Sec. 4.2. The experimental results are listed in Table 8.

From the results in Table 8, it can be observed that the attack success rate is significantly decreased without OD-CAM, followed by a slight drop in image quality. This proves the effectiveness of OD-CAM in improving the attack success rate and image quality of the generated adversarial examples.

4.5.3 Patch in the suboptimal area

To simulate real-world challenges, we place adversarial patches obtained using OD-CAM into constrained or suboptimal regions of the image. In the specific experiment, we use the partitions of each heatmap group that are greater than the threshold of 0.3 and less than the threshold of 0.8 as masks. And the YOLOv4 model and Faster R-CNN model were used to complete this experiment on the VOC0712 dataset and COCO2017 dataset, respectively. The experimental results are listed in Table 9.

Table 9 The attack performance of IPAttack when the patch is in the suboptimal area

Model	YOLOv4		Faster R-CNN	
	VOC0712	COCO2017	VOC0712	COCO2017
ASR	0.6375	0.5463	0.7246	0.2660
PSNR	37.8456	39.1341	38.2539	36.7464
SSIM	0.8813	0.8122	0.8351	0.7587

As shown in Table 9, it can be observed that when the patch is placed in suboptimal regions, the ASR, PSNR, and SSIM generated by the IPAttack method are significantly reduced compared to patches placed in optimal regions.

4.6 Potential defense mechanisms

Similar to image classification tasks, adversarial training is one of the most effective approaches to improving the robustness of object detectors. Specifically, the adversarial examples generated using the proposed method can be used to perform adversarial training on the target detector, thereby enhancing its robustness against unseen patches. Additionally, existing universal attack methods can be employed to generate universal adversarial examples, which can then be used to conduct adversarial training, further improving the detector's resilience against unknown attacks.

5 Conclusions

In this paper, we propose IPAttack to generate imperceptible adversarial patches for object detectors. Our experiment's results in terms of ASR and image quality on the VOC0712 dataset and COCO2017 dataset demonstrate the effectiveness and superiority of our attack against general one-stage object detectors and two-stage object detectors. To determine the location of the patch, we propose OD-CAM to search for regions of interest for the object detector. Besides, to make adversarial patches more imperceptible, we use spatial transform to generate adversarial patches and achieve good results. To our knowledge, we are the first to propose a method that applies spatial transform to generate adversarial examples for object detectors. The experimental results show that the DNN-based object detectors are vulnerable to the imperceptible adversarial patch attack, which provides inspiration for later workers to study the vulnerability and further improve the robustness of the object detectors.

Author Contributions Yongming Wen: Methodology; Writing-Original draft. Peiyuan Si: Investigation; Software; Revise, Investigation. Wei Zhou: Validation; Funding acquisition. Zongheng Zhao: Data curation; Resources. Chao Yi: Supervision; Project administration. Renyang Liu: Formal analysis, Writing, Review, Revision.

Funding This work is supported in part by the National Natural Science Foundation of China under Grant 62162067 and 62101480, Research and Application of Object detection based on Artificial Intelligence, in part by the Yunnan Province expert workstations under Grant202305AF150078, in part by Yunnan Fundamental Research Projects under Grant Nos. 202401AT070474.

Data Availability The datasets used or analyzed during the current study are available from the link provided in this paper.

The datasets used in this paper is available online publicly.

Declarations

Competing Interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical and informed consent for data used This article does not contain any studies with human participants performed by any of the authors.

References

1. Farfade SS, Saberian MJ, Li L-J (2015) Multi-view face detection using deep convolutional neural networks. In: ICMR pp 643–650. https://doi.org/10.1145/2671188.2749408
2. Sun Y, Liang D, Wang X, Tang X (2015) Deepid3: face recognition with very deep neural networks. arXiv:1502.00873
3. Bajpai S, Mishra G, Jain R, Jain DK, Saini D, Hussain A (2025) Ri-l1a pprox: a novel resnet-inception-based fast l1-approximation method for face recognition. Neurocomputing. https://doi.org/10.1016/J.NEUCOM.2024.128708
4. Gim T, Sohn K (2024) Regularization using noise samples identified by the feature norm for face recognition. IEEE Access. https://doi.org/10.1109/ACCESS.2024.3453030
5. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
6. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: CVPR pp 770–778
7. Bochkovskiy A, Wang C-Y, Liao H-YM (2020) Yolov4: optimal speed and accuracy of object detection. arXiv:2004.10934
8. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: towards real-time object detection with region proposal networks. NeurIPS,
9. Zhang G, Chen J, Gao G, Li J, Liu S, Hu X (2024) Safdnet: a simple and effective network for fully sparse 3d object detection. In: CVPR pp 14477–14486. https://doi.org/10.1109/CVPR52733.2024.01372
10. Kim J, Cho H, Kim J, Tiruneh YY, Baek S (2024) SDDGR: stable diffusion-based deep generative replay for class incremental object detection. In: CVPR. https://doi.org/10.1109/CVPR52733.2024.02718
11. Zhou Z, Rahman Siddiquee MM, Tajbakhsh N, Liang J (2018) Unet++: a nested u-net architecture for medical image segmentation. In: MICCAI pp 3–11. https://doi.org/10.1007/978-3-030-00889-5_1
12. Nirkin,Y, Wolf L, Hassner T (2021) Hyperseg: patch-wise hypernetwork for real-time semantic segmentation. In: CVPR pp 4061–4070. https://doi.org/10.1109/CVPR46437.2021.00405
13. Fan X, Wang X, Gao J, Wang J, Luo Z, Liu R (2024) Bi-level learning of task-specific decoders for joint registration and one-shot medical image segmentation. In: CVPR. https://doi.org/10.1109/CVPR52733.2024.01114
14. Jiang S, Wu H, Chen J, Zhang Q, Qin J (2024) Ph-net: semi-supervised breast lesion segmentation via patch-wise hardness. In: IEEE/CVF conference on computer vision and pattern recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024. https://doi.org/10.1109/CVPR52733.2024.01085

15. Goodfellow IJ, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples. In: ICLR
16. Liu R, Zhang J, Li H, Zhang J, Wang Y, Zhou W (2023) AFLOW: developing adversarial examples under extremely noise-limited settings. In: ICICS, vol 14252, pp 502–518. https://doi.org/10.1007/978-981-99-7356-9_30
17. Li L, Guan H, Qiu J, Spratling MW (2024) One prompt word is enough to boost adversarial robustness for pre-trained vision-language models. In: IEEE/CVF conference on computer vision and pattern recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024. https://doi.org/10.1109/CVPR52733.2024.02304
18. Brown TB, Mané D, Roy A, Abadi M, Gilmer J (2017) Adversarial patch. arXiv:1712.09665
19. Liu T, Yang C, Liu X, Han R, Ma J (2024) RPAU: fooling the eyes of uavs via physical adversarial patches. IEEE Trans Intell Transp Syst 2586–2598. https://doi.org/10.1109/TITS.2023.3317054
20. Guo F, Sun Z, Chen Y, Ju L (2023) Towards the universal defense for query-based audio adversarial attacks on speech recognition system. Cybersecurity 40. https://doi.org/10.1186/S42400-023-00177-6
21. Guo F, Sun Z, Chen Y, Ju L (2023) Towards the transferable audio adversarial attack via ensemble methods. Cybersecurity 44. https://doi.org/10.1186/S42400-023-00175-8
22. Chen T, Liu J, Xiang Y, Niu W, Tong E, Han Z (2019) Adversarial attack and defense in reinforcement learning-from AI security view. Cybersecurity 11. https://doi.org/10.1186/S42400-019-0027-X
23. Dong Y, Su H, Zhu J, Bao F (2017) Towards interpretable deep neural networks by leveraging adversarial examples. arXiv:1708.05493
24. Pang T, Du C, Dong Y, Zhu J (2018) Towards robust detection of adversarial examples. In: NeurIPS, pp 4584–4594
25. Wang J, Chang X, Wang Y, Rodríguez RJ, Zhang J (2021) LSGAN-AT: enhancing malware detector robustness against adversarial examples. Cybersecurity 38. https://doi.org/10.1186/S42400-021-00102-9
26. Garaev R, Rasheed B, Khan AM (2024) Not so robust after all: evaluating the robustness of deep neural networks to unseen adversarial attacks. Algorithms. https://doi.org/10.3390/A17040162
27. Mao J, Shi S, Wang X, Li H (2022) 3d object detection for autonomous driving: a review and new outlooks. arXiv:2206.09474
28. Xie C, Wang J, Zhang Z, Zhou Y, Xie L, Yuille AL (2017) Adversarial examples for semantic segmentation and object detection. In: ICCV, pp 1378–1387. https://doi.org/10.1109/ICCV.2017.153
29. Wei X, Liang S, Chen N, Cao X (2018) Transferable adversarial attacks for image and video object detection. arXiv:1811.12641
30. Liang S, Wei X, Cao X (2021) Generate more imperceptible adversarial examples for object detection. In: ICML
31. Liu X, Yang H, Liu Z, Song L, Chen Y, Li H (2019) DPATCH: an adversarial patch attack on object detectors. In: AAAI
32. Mirsky Y (2023) Ipatch: a remote adversarial patch. Cybersecurity 18
33. Wu S, Dai T, Xia S-T (2020) Dpattack: diffused patch attacks against universal object detection. arXiv:2010.11679
34. Huang H, Wang Y, Chen Z, Tang Z, Zhang W, Ma K (2021) Rpattack: refined patch attack on general object detectors. In: ICME, pp 1–6. https://doi.org/10.1109/ICME51207.2021.9428443
35. Xiao C, Zhu J, Li B, He W, Liu M, Song D (2018) Spatially transformed adversarial examples. In: ICLR
36. Liu R, Jin X, Hu D, Zhang J, Wang Y, Zhang J, Zhou W (2023) Dualflow: generating imperceptible adversarial examples by flow field and normalize flow-based model. Front Neurorobot. https://doi.org/10.3389/FNBOT.2023.1129720
37. Liu R, Zhou W, Wu S, Zhao J, Lam K-Y (2023) Ssta: salient spatially transformed attack. arXiv:2312.07258
38. Chattopadhay A, Sarkar A, Howlader P, Balasubramanian VN (2018) Grad-cam++: generalized gradient-based visual explanations for deep convolutional networks. In: WACV, pp 839–847. https://doi.org/10.1109/WACV.2018.00097
39. Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D (2017) Grad-cam: visual explanations from deep networks via gradient-based localization. In: ICCV, pp 618–626. https://doi.org/10.1007/S11263-019-01228-7
40. Zhou B, Khosla A, Lapedriza A, Oliva A, Torralba A (2016) Learning deep features for discriminative localization. In: CVPR. https://doi.org/10.1109/CVPR.2016.319
41. Neubeck A, Van Gool L (2006) Efficient non-maximum suppression. In: ICPR, pp 850–855
42. Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: CVPR, pp 248–255
43. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC (2016) Ssd: single shot multibox detector. In: ECCV, pp 21–37. https://doi.org/10.1007/978-3-319-46448-0_2
44. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR, pp 580–587. https://doi.org/10.1109/CVPR.2014.81
45. Chindaudom A, Siritanawan P, Sumongkayothin K, Kotani K (2020) Adversarialqr: an adversarial patch in qr code format. In: ICIEV & icIVPR, pp 1–6
46. Kurakin A, Goodfellow IJ, Bengio S (2017) Adversarial examples in the physical world. In: ICLR
47. Zhao Y, Yan H, Wei X (2020) Object hider: adversarial patch attack against object detectors. arXiv:2010.14974
48. Jaderberg M, Simonyan K, Zisserman A, et al (2015) Spatial transformer networks. NeurIPS
49. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv:1412.6980
50. Lin T, Maire M, Belongie SJ, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft COCO: common objects in context. In: ECCV, pp 740–755. https://doi.org/10.1007/978-3-319-10602-1_48
51. Everingham M (2007) The pascal visual object classes challenge 2007 (voc2007) results. http://www.Pascal-network.org/challenges/VOC/voc2008/year=Workshop/index.Html
52. Everingham M, Van Gool L, Williams C, Winn J, Zisserman A (2012) The Pascal visual object classes challenge 2012 results, vol 5