Adversarial Objectness Gradient Attacks in Real-time Object Detection Systems

Ka-Ho Chow[1], Ling Liu[1], Margaret Loper[2], Juhyun Bae[1], Mehmet Emre Gursoy[1],
Stacey Truex[1], Wenqi Wei[1], Yanzhao Wu[1]
[1]School of Computer Science, Georgia Institute of Technology
[2]Georgia Tech Research Institute
Atlanta, GA, USA

***Abstract*—Real-time object detection is one of the key applications of deep neural networks (DNNs) for real-world mission-critical systems. While DNN-powered object detection systems celebrate many life-enriching opportunities, they also open doors for misuse and abuse. This paper presents a suite of adversarial objectness gradient attacks, coined as TOG, which can cause the state-of-the-art deep object detection networks to suffer from untargeted random attacks or even targeted attacks with three types of specificity: (1) object-vanishing, (2) object-fabrication, and (3) object-mislabeling. Apart from tailoring an adversarial perturbation for each input image, we further demonstrate TOG as a universal attack, which trains a single adversarial perturbation that can be generalized to effectively craft an unseen input with a negligible attack time cost. Also, we apply TOG as an adversarial patch attack, a form of physical attacks, showing its ability to optimize a visually confined patch filled with malicious patterns, deceiving well-trained object detectors to misbehave purposefully. We report our experimental measurements using three benchmark datasets (PASCAL VOC, MS COCO, and INRIA) on models from three dominant detection algorithms (YOLOv3, SSD, and Faster R-CNN). The results demonstrate serious adversarial vulnerabilities and the compelling need for developing robust object detection systems. The source code is available at https://github.com/git-disl/TOG.**

***Index Terms*—object detection, adversarial attacks, deep neural networks**

I. Introduction

Deep neural networks (DNNs) have revolutionized object detection [1]–[3] and fueled the development of a wide range of applications, such as traffic sign/pedestrian detection on autonomous vehicles [4] and region of interest identification in radiology [5]. Given an input video frame or image, an object detector detects multiple instances of semantic objects using the known categories [6]. Existing deep object detection algorithms can be classified into two paradigms: proposal-based two-phase learning and regression-based one-phase learning. Proposal-based approaches, dominated by the R-CNN family [3], [7], [8], exploit a two-phase procedure by first using a region proposal network (RPN) to recommend a set of proposal regions, which are subsequently refined by bounding box prediction and class label prediction. Regression-based techniques, represented by YOLO [2], [9], [10] and SSD [1], take a different approach by jointly detecting multiple objects with their bounding box and class label in one-shot.

While deep object detectors [1]–[3] offer real-time performance with unparalleled accuracy over traditional techniques [6], recent studies have revealed their vulnerabilities to adversarial perturbations [11]–[17]. For instance, DAG [11] and RAP [12] disable the functionality of the region proposal network in two-phase models (e.g., Faster R-CNN [3]) with an iterative gradient-based approach. UEA [13] proposes to train a generative adversarial network to produce adversarial examples. While the above techniques reveal adversarial vulnerabilities of deep object detectors, they focus on two-phase models where RPNs are employed. Furthermore, their adverse effects are random and can only deceive the victim to misbehave with uncontrollable behavior.

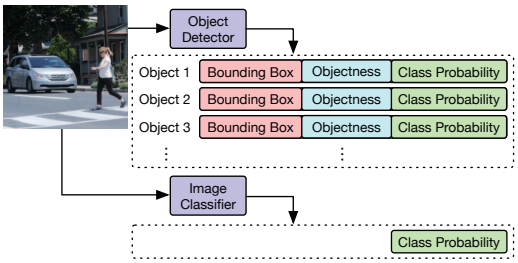


Fig. 1: The output structure of an object detector v.s. the output structure of an image classifier.

In this paper, we demonstrate that adversarial attacks on deep object detection networks can be more sophisticated beyond simply deceiving the victim to misbehave randomly [11]–[13]. We develop a suite of attacks, coined as TOG, that enable adversaries to fool the victim with three types of targeted specificity: object-vanishing, object-fabrication, and object-mislabeling. This is accomplished by the observation on the unique output structure of object detectors, as shown in Figure 1. A deep image classifier takes a query image as input and outputs its prediction class with the confidence probability vector, producing only one class label for the entire image (e.g., road). In comparison, a deep object detector takes a query image as input and outputs three types of prediction results: (1) a set of bounding boxes, one for each object detected in the input image, (2) a set of objectness measures, indicating the probability of the associated bounding box being a real object, and (3) a set of class probability vectors. Each output prediction type represents one type of attack surfaces, and their combinations formulate a wide array of attack possibilities. Figure 2 provides an illustration by examples. Compared with the perfect detection made by

978-1-7281-8543-9/20/$31.00 ©2020 IEEE
DOI 10.1109/TPS-ISA50397.2020.00042

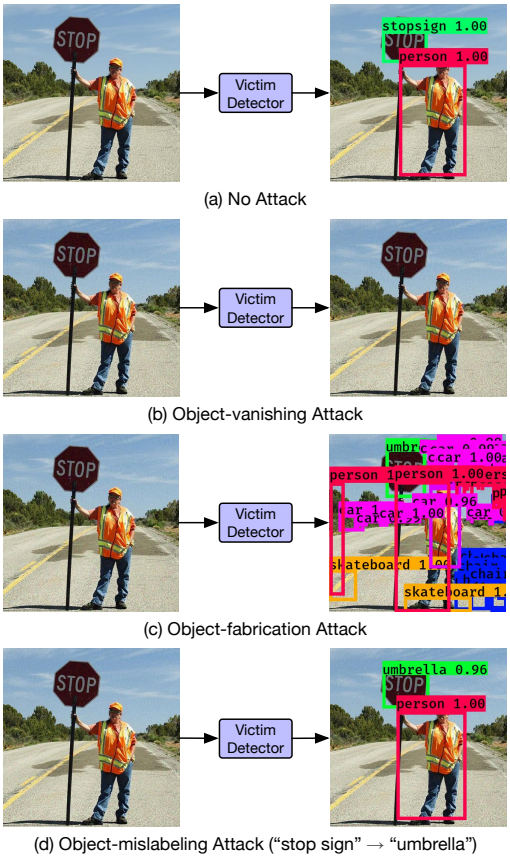


Fig. 2: Attacks with three types of targeted specificity (2nd-4th rows) in deep object detection networks. *Left*: Benign input in row 1 and adversarial inputs in row 2 to row 4 to the object detector YOLOv3 [2]. *Right*: Detection results.

the detector under no attack (Figure 2(a)), (1) the object-vanishing attack in Figure 2(b) removes the victim's ability in recognizing any object, (2) the object-fabrication attack in Figure 2(c) causes the detector to output additional false objects with high confidence, and (3) the object-mislabeling attack in Figure 2(d) fools the detector to mislabel detected objects (e.g., stop sign as an umbrella), which can result in disastrous consequences.

To understand the adversarial vulnerability of deep object detection networks and underscore the urgent need to develop robust systems, this paper makes three unique contributions: (1) We develop TOG, a suite of adversarial attacks supporting not only untargeted random attacks but also targeted specificity attacks to purposefully deceive victims with object-vanishing, object-fabrication, and object-mislabeling effects. Unlike existing approaches, TOG is applicable to both one-phase and two-phase detectors, as no specific structure is manipulated. (2) Apart from tailoring an adversarial perturbation for each query input image, we further describe a routine for TOG to train a universal perturbation [18] offline over a training set. It can be used to maliciously perturb an unseen input with a negligible time cost, which is particularly detrimental to time-sensitive applications. (3) We also show that TOG can be launched as a form of adversarial patches, a visually confined patch filled with malicious patterns that can fool the victim to misdetect purposefully. Finally, to show the applicability of TOG on different object detectors and datasets with different degrees of difficulties, we conduct extensive experiments on three dominant object detection algorithms (YOLOv3 [2], SSD [1], and Faster R-CNN [3]), covering both one-phase and two-phase techniques, and three benchmark datasets (PASCAL VOC [19], MS COCO [20], and INRIA [21]). Our experimental results indicate high attack success rates and serious vulnerabilities of real-time object detectors.

The remainder of this paper is organized as follows. We overview DNN-based object detection and adversarial attacks in Section II. Then, the methodology of TOG is presented in Section III, followed by experimental evaluation in Section IV. We review related works in Section V and finally conclude the paper in Section VI.

II. DNN-based Object Detection and Adversarial Attacks

Deep object detection networks, in general, share the same input-output structure with a similar formulation. Given an input image $\boldsymbol{x}$, an object detector first detects a large number of S candidate bounding boxes $\{\hat{\boldsymbol{o}}_1, \hat{\boldsymbol{o}}_2, \ldots, \hat{\boldsymbol{o}}_S\}$ where $\hat{\boldsymbol{o}}_i = (\hat{b}_i^x, \hat{b}_i^y, \hat{b}_i^W, \hat{b}_i^H, \hat{C}_i, \hat{\boldsymbol{p}}_i)$ is a candidate centered at $(\hat{b}_i^x, \hat{b}_i^y)$ having a dimension $(\hat{b}_i^W, \hat{b}_i^H)$ with an objectness probability of $\hat{C}_i \in [0, 1]$ being a real object, and a K-class probability vector $\hat{\boldsymbol{p}}_i = (\hat{p}_i^1, \hat{p}_i^2, \ldots, \hat{p}_i^K)$. This is done by dividing the input $\boldsymbol{x}$ into mesh grids in different scales (resolutions). Each grid cell produces multiple candidate bounding boxes based on the anchors and is responsible for locating objects centered at the cell. The final detection result $\hat{\boldsymbol{\mathcal{O}}}(\boldsymbol{x})$ is obtained by applying confidence thresholding to remove candidates with low prediction confidence and non-maximum suppression (NMS) to exclude those with high overlapping. Taking YOLOv3 [2] as an example. It divides an input image of size (416×416) into (13×13), (26×26), and (52×52) mesh grids where each grid cell is associated with three anchors. Hence, $S = 3 \times (13 \times 13 + 26 \times 26 + 52 \times 52) = 10,647$ candidate bounding boxes are predicted on a given input. Those candidates with prediction confidence lower than $t_{\text{conf}} = 0.20$ will be immediately discarded. The remaining ones will be further filtered by NMS with an intersection over union threshold $t_{\text{IOU}} = 0.45$.

An adversarial example $\boldsymbol{x}'$ is generated by perturbing a benign input $\boldsymbol{x}$ sent to the victim detector, aiming to fool the victim to misdetect randomly (untargeted) or purposefully (targeted). The generation process of the adversarial example can be conceptually formulated as

$$\min ||\boldsymbol{x}' - \boldsymbol{x}||_p \quad s.t.\ \hat{\boldsymbol{\mathcal{O}}}(\boldsymbol{x}') \neq \hat{\boldsymbol{\mathcal{O}}}(\boldsymbol{x}), \tag{1}$$

where p is the distance metric, such as the L_0 norm measuring the percentage of the pixels that are changed, the L_2 norm computing the Euclidean distance, or the L_∞ norm denoting the maximum change to any pixel.

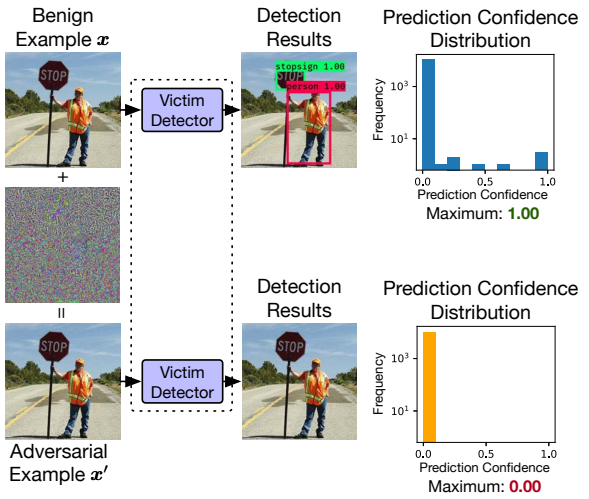


Fig. 3: An example of adversarial attacks on a YOLOv3 detector. The perturbation is magnified for visibility.

Figure 3 gives an example of adversarial attacks on the YOLOv3 detector. Given a benign image with a person holding a stop sign, the top row shows that the victim detector can accurately locate the two objects with high confidence (both 1.00). Upon adding a small and malicious perturbation to the benign input image, we generate an adversarial example $\boldsymbol{x}'$ for this benign input $\boldsymbol{x}$. While $\boldsymbol{x}'$ is visually indistinguishable from the benign input, when taking $\boldsymbol{x}'$ as the query input, the *same* object detector returns no object (i.e., $\hat{\boldsymbol{\mathcal{O}}}(\boldsymbol{x}') = \emptyset$), because this attack is to perturb the input image such that the prediction confidence at every candidate bounding box is very low (close to zero). As a result, none of the candidate objects survive the confidence thresholding ($t_{\text{conf}} = 0.20$).

III. Methodology

Training a deep neural network starts with random initialization of model weights, which will be updated slowly by taking the derivative of the loss function $\mathcal{L}$ with respect to the learnable model weights $\boldsymbol{\mathcal{W}}$ over a mini-batch of input-output pairs $\{(\tilde{\boldsymbol{x}}, \boldsymbol{\mathcal{O}})\}$ with the following equation until convergence:

$$\boldsymbol{\mathcal{W}}_{t+1} = \boldsymbol{\mathcal{W}}_t - \alpha \frac{\partial\, \mathbb{E}_{(\tilde{\boldsymbol{x}}, \boldsymbol{\mathcal{O}})}[\mathcal{L}(\tilde{\boldsymbol{x}}; \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{W}}_t)]}{\partial \boldsymbol{\mathcal{W}}_t} \tag{2}$$

where α is the learning rate controlling the step size of the update. In deep object detection networks, every ground-truth object in a training sample $\tilde{\boldsymbol{x}}$ will be assigned to one of the S bounding boxes (recall Section II) according to the center coordinates and the amount of overlapping with the anchors. Let $\mathbb{1}_i = 1$ if the i-th bounding box is responsible for an object and 0 otherwise. Then, $\boldsymbol{\mathcal{O}} = \{\boldsymbol{o}_i | \mathbb{1}_i = 1, 1 \le i \le S\}$ is a set of ground-truth objects where $\boldsymbol{o}_i = (b_i^x, b_i^y, b_i^W, b_i^H, \boldsymbol{p}_i)$ with $\boldsymbol{p}_i = (p_i^1, p_i^2, \ldots, p_i^K)$ and $p_i^c = 1$ if $\boldsymbol{o}_i$ is a class c object. Deep object detection is an instance of multi-task learning with three learning components, each of them corresponds to one of the three output structures (recall Figure 1) describing a detected object: (1) objectness, (2) bounding box, and (3) class label. The objectness score $\hat{C}_i \in [0, 1]$ determines the existence of an object, which can be learned by minimizing the binary cross-entropy ℓ_{BCE}:

$$\mathcal{L}_{\text{obj}}(\tilde{\boldsymbol{x}}; \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{W}}) = \sum_{i=1}^{S} \Big[\mathbb{1}_i \ell_{\text{BCE}}(1, \hat{C}_i) + (1 - \mathbb{1}_i) \ell_{\text{BCE}}(0, \hat{C}_i) \Big]. \tag{3}$$

The location $(\hat{b}_i^x, \hat{b}_i^y)$ and dimension $(\hat{b}_i^W, \hat{b}_i^H)$ constitute a bounding box, learned by minimizing the squared error ℓ_{SE}:

$$\begin{aligned} \mathcal{L}_{\text{bbox}}(\tilde{\boldsymbol{x}}; \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{W}}) = \sum_{i=1}^{S} \mathbb{1}_i \big[& \ell_{\text{SE}}\big(b_i^x, \hat{b}_i^x\big) + \ell_{\text{SE}}\big(b_i^y, \hat{b}_i^y\big) \\ & + \ell_{\text{SE}}\big(\sqrt{b_i^W}, \sqrt{\hat{b}_i^W}\big) + \ell_{\text{SE}}\big(\sqrt{b_i^H}, \sqrt{\hat{b}_i^H}\big) \big] \end{aligned} \tag{4}$$

The last part is the K-class probability vector $\hat{\boldsymbol{p}}_i = (\hat{p}_i^1, \hat{p}_i^2, \ldots, \hat{p}_i^K)$ that estimates the class label of the candidate, optimized by minimizing the binary cross-entropy:

$$\mathcal{L}_{\text{class}}(\tilde{\boldsymbol{x}}; \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{W}}) = \sum_{i=1}^{S} \mathbb{1}_i \sum_{c=1}^{K} \ell_{\text{BCE}}\big(p_i^c, \hat{p}_i^c\big) \tag{5}$$

As a result, a deep object detector can be optimized using Equation 2 with the combination of the above loss functions:

$$\begin{aligned} \mathcal{L}(\tilde{\boldsymbol{x}}; \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{W}}) = & \mathcal{L}_{\text{obj}}(\tilde{\boldsymbol{x}}; \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{W}}) + \\ & \mathcal{L}_{\text{bbox}}(\tilde{\boldsymbol{x}}; \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{W}}) + \mathcal{L}_{\text{class}}(\tilde{\boldsymbol{x}}; \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{W}}). \end{aligned} \tag{6}$$

While training deep object detection networks is done by *fixing* the input image $\tilde{\boldsymbol{x}}$ and progressively *updating* the model weights $\boldsymbol{\mathcal{W}}$ towards the goal defined by the loss function (Equation 2), TOG conducts adversarial attacks by reversing the training process. We *fix* the model weights of the victim detector and iteratively *update* the input image $\boldsymbol{x}$ towards the goal defined by the type of the attack to be launched with the following general equation:

$$\boldsymbol{x}'_t = \prod_{\boldsymbol{x}, \epsilon} \left[\boldsymbol{x}'_{t-1} - \alpha_{\text{TOG}} \Gamma \left(\frac{\partial \mathcal{L}^*(\boldsymbol{x}'_{t-1}; \boldsymbol{\mathcal{O}}^*, \boldsymbol{\mathcal{W}})}{\partial \boldsymbol{x}'_{t-1}} \right) \right] \tag{7}$$

where $\prod_{\boldsymbol{x}, \epsilon}[\cdot]$ is the projection onto a hypersphere with a radius ϵ centered at $\boldsymbol{x}$ in L_p norm, followed by clipping to ensure the validity of pixel values, α_{TOG} is the attack learning rate, and Γ is a sign function. By strategically manipulating the adversarial loss $\mathcal{L}^*$ and the auxiliary target detection $\boldsymbol{\mathcal{O}}^*$, TOG supports not only untargeted random attacks, fooling the victim to misbehave with uncontrollable effects [11]–[13], but also three types of targeted specificity attacks which deceive the victim to misdetect purposefully.

Figure 4 overviews the adversarial attacks using TOG. Given an input source $\boldsymbol{x}$ (e.g., an image or video frame), the TOG attack module takes the configuration specified by the adversary to prepare for the corresponding adversarial perturbation, which will be added to the input to cause the victim to misdetect. The first four attacks: TOG-untargeted, TOG-vanishing, TOG-fabrication, and TOG-mislabeling tailor an adversarial perturbation for each input. Figure 5 illustrates the iterative approach adopted by TOG. Initialized with the benign example (i.e., $\boldsymbol{x}'_0 = \boldsymbol{x}$), the adversarial example $\boldsymbol{x}'_t$ at the t-th iteration is sent to the victim detector to observe the detection result $\hat{\boldsymbol{\mathcal{O}}}(\boldsymbol{x}'_t)$. If the termination condition defined by the attack goal is achieved or the maximum number of

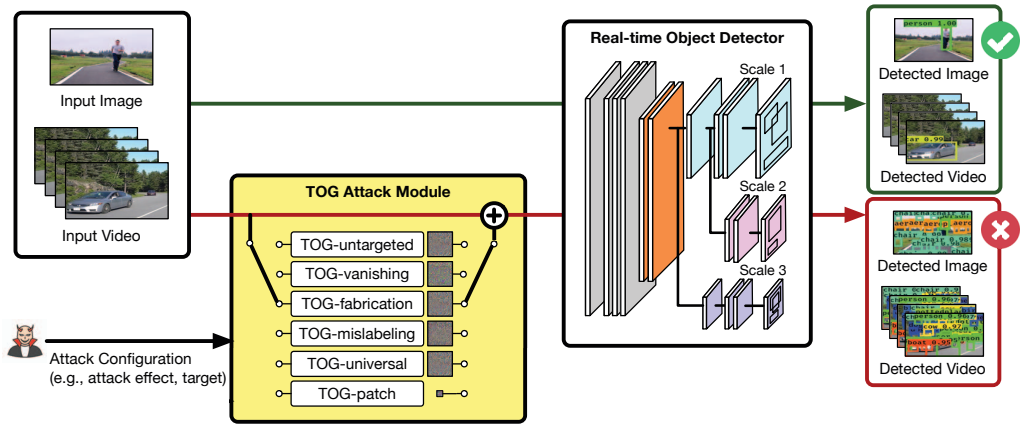


Fig. 4: An illustration of the adversarial attacks using TOG.

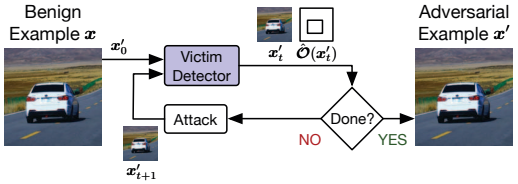


Fig. 5: The iterative attack strategy adopted by TOG to tailor an adversarial perturbation for each given input.

iterations T is reached, $\boldsymbol{x}'_t$ will be returned. Otherwise, it will be further perturbed to become $\boldsymbol{x}'_{t+1}$ for the next iteration.

(1) TOG-untargeted. Random untargeted attacks fool the victim detector to *randomly* misdetect without targeting at any specific object. This class of attacks succeeds if the adversarial example fools the victim detector to give incorrect results of any form, such as having objects vanished, fabricated, or mislabeled randomly. TOG-untargeted exploits gradients from both $\mathcal{L}_{\text{obj}}$, $\mathcal{L}_{\text{bbox}}$, and $\mathcal{L}_{\text{class}}$ and formulates the attack to be

$$\boldsymbol{x}'_t = \prod_{\boldsymbol{x},\epsilon} \left[\boldsymbol{x}'_{t-1} + \alpha_{\text{TOG}} \Gamma \left(\frac{\partial \mathcal{L}(\boldsymbol{x}'_{t-1}; \hat{\boldsymbol{\mathcal{O}}}(\boldsymbol{x}), \boldsymbol{\mathcal{W}})}{\partial \boldsymbol{x}'_{t-1}} \right) \right], \quad (8)$$

where $\mathcal{L}$ is the detection loss from Equation 6. This is equivalently to deceiving the victim to make wrong decisions on object existences, bounding boxes, and class labels.

(2) TOG-vanishing. Having a high recall to retrieve the objects in the input image is crucial in many applications. For instance, object detection has been applied in radiology to conduct automated breast cancer diagnosis [5], and positive detection alarms the patient to receive a further medical examination by a human expert. TOG-vanishing aims at removing the victim's ability to identify objects (i.e., adding false negatives). We iteratively push the benign example $\boldsymbol{x}$ along the direction of producing an empty detection set by exploiting $\mathcal{L}_{\text{obj}}$ which dominates the decision on object existences. The adversarial example $\boldsymbol{x}'_t$ at the t-th iteration is formulated as:

$$\boldsymbol{x}'_t = \prod_{\boldsymbol{x},\epsilon} \left[\boldsymbol{x}'_{t-1} - \alpha_{\text{TOG}} \Gamma \left(\frac{\partial \mathcal{L}_{\text{obj}}(\boldsymbol{x}'_{t-1}; \emptyset, \boldsymbol{\mathcal{W}})}{\partial \boldsymbol{x}'_{t-1}} \right) \right]. \quad (9)$$

The desired effect is achieved by making no candidate survives the confidence thresholding (see Figure 2(b) and Figure 3).

(3) TOG-fabrication. Different from the above, the TOG-fabrication attack fabricates additional detections (i.e., adding false positives). It maximizes the object existences using gradients from $\mathcal{L}_{\text{obj}}$ with the following formulation:

$$\boldsymbol{x}'_t = \prod_{\boldsymbol{x},\epsilon} \left[\boldsymbol{x}'_{t-1} + \alpha_{\text{TOG}} \Gamma \left(\frac{\partial \mathcal{L}_{\text{obj}}(\boldsymbol{x}'_{t-1}; \emptyset, \boldsymbol{\mathcal{W}})}{\partial \boldsymbol{x}'_{t-1}} \right) \right]. \quad (10)$$

As shown in Figure 2(c), this attack results in numerous false detections.

(4) TOG-mislabeling. This attack consistently causes the victim detector to misclassify the objects detected on the input image by replacing their source class label with the maliciously chosen target class label, while maintaining the same set of correct bounding boxes. Such an attack can cause fatal collisions in many scenarios, e.g., misclassifying the stop sign as an umbrella in Figure 2(d). By focusing on the classification loss (i.e., $\mathcal{L}_{\text{class}}$) and keeping the gradients of the other two parts unchanged, the adversarial example can then be formulated as follows:

$$\boldsymbol{x}'_t = \prod_{\boldsymbol{x},\epsilon} \left[\boldsymbol{x}'_{t-1} - \alpha_{\text{TOG}} \Gamma \left(\frac{\partial \mathcal{L}(\boldsymbol{x}'_{t-1}; \boldsymbol{\mathcal{O}}^*(\boldsymbol{x}), \boldsymbol{\mathcal{W}})}{\partial \boldsymbol{x}'_{t-1}} \right) \right]. \quad (11)$$

The auxiliary target detection $\boldsymbol{\mathcal{O}}^*(\boldsymbol{x})$ is constructed by setting each object $\hat{\boldsymbol{o}}$ in $\hat{\boldsymbol{\mathcal{O}}}(\boldsymbol{x})$ to its maliciously chosen class label. Alternatively, we can conduct the most-likely (ML) class attack by setting it to the incorrect class label with the highest probability predicted by the victim (i.e., $\arg\max_{c,\hat{p}^c \neq \max_c \hat{p}^c} \hat{p}^c$) or the least-likely (LL) class attack with the lowest probability (i.e., $\arg\min_c \hat{p}^c$).

Instead of tailoring an adversarial perturbation upon receiving an input image from users, the last two attacks in Figure 4: TOG-universal and TOG-patch train an adversarial perturbation offline on a training set and apply the same perturbation to any unseen input. Once an input is received, the only operation is to add the trained perturbation to the image without any per-input optimization, which is necessary in the first four attacks. The attack time cost during the online detection phase is hence significantly lowered, and the attacks are particularly detrimental to time-sensitive applications such as autonomous driving.

TABLE I: An example of TOG-patch with object-vanishing or object-mislabeling effect.

(5) TOG-universal. Inspired by [18], TOG-universal generates a universal perturbation allowed to maliciously alter the pixel values of the entire image. By exploiting the partial derivative in Equation 9-10, one can designate the attack specificity caused by the universal perturbation, rather than some arbitrary effects. Let $\boldsymbol{\mathcal{D}}$ denote the set of N training images, we want to gradually build the perturbation $\boldsymbol{\eta}$. At each iteration t, we obtain a training sample $(\tilde{\boldsymbol{x}}, \boldsymbol{\mathcal{O}}) \in \boldsymbol{\mathcal{D}}$ and find the additional perturbation $\Delta\boldsymbol{\eta}_t$ that causes the victim detector to make errors towards object vanishing (Equation 9) or fabrication (Equation 10) attack specificity in the current perturbed image $\tilde{\boldsymbol{x}} + \boldsymbol{\eta}_t$. We then add this additional perturbation $\Delta\boldsymbol{\eta}_t$ to the current universal adversarial perturbation $\boldsymbol{\eta}_t$ and clip the new perturbation to ensure the distortion is constrained within $[-\epsilon, \epsilon]$. The training repeats for Q epochs. Early termination can be incorporated by, say, monitoring the number of detected objects vanished from the training set.

(6) TOG-patch. Different from TOG-universal, TOG-patch trains an adversarial patch with a fixed size (e.g., 64×64). At the online detection phase, the trained patch is place either digitally [15] or physically [22] on the objects with the class of interest (e.g., person). The image with the adversarial patch can then deceive the victim to misdetect with purposeful effects. Table I shows an image of a person detected accurately by a YOLOv3 detector (1st column). By placing a TOG-patch on the person, he can either vanish from the victim (2nd column) or be mislabeled purposefully as a chair (3rd column), depending on the adversarial loss and auxiliary target detection employed during the training phase of the TOG-patch. The training procedure of TOG-patch is conducted as follows. We first randomly initialize the adversarial patch of a given size. Then, we get a mini-batch of training samples from the training set and place the current patch on the person detected by the victim. The gradient towards the attack specificity (the partial derivative in Equation 9 for object-vanishing or Equation 11 for object-mislabeling) is computed w.r.t. the patch instead of the entire image and is used to update the TOG-patch for the next iteration of training until a predefined number of epochs Q is reached. Existing proposals on enhancing the robustness of adversarial patches under physical environments can be incorporated, such as minimizing the non-printability score to mitigate printer color reproduction errors [23] and conducting data augmentation to cater different viewing angles [22].

IV. Experimental Evaluation

We conduct experiments on three benchmark datasets: PASCAL VOC [19], MS COCO [20], and INRIA [21] , and report the results based on the entire test set ($4,952$ for VOC, $5,000$ for COCO, and 288 for INRIA). To evaluate the performance of TOG attacking different object detectors, three state-of-the-art algorithms: YOLOv3 [9], SSD [1], and Faster R-CNN [3], covering both one-phase and two-phase detection techniques, are studied. We also consider detection algorithms with different backbone architectures. In the rest of this paper, YOLOv3-D and YOLOv3-M denote YOLOv3 detectors with a Darknet53 and a MobileNetV1 backbone respectively. Both of them have an input size of (416×416). SSD300 and SSD512 are both SSD models with a VGG16 backbone but require input resolution of (300×300) and (512×512) respectively. FRCNN is a Faster R-CNN model with a VGG16 backbone and input resolution of (600×600). The default confidence thresholds (i.e., t_{conf}) for YOLOv3, SSD, and FRCNN models are 0.20, 0.50, and 0.70 respectively. All experiments were run on NVIDIA RTX 2080 SUPER (8 GB) GPU.

A. Untargeted Random Attacks

We begin with comparing TOG with three state-of-the-art untargeted random attacks: DAG [11], RAP [12], and UEA [13]. All comparison schemes are set with their default hyperparameters. For TOG, we generate adversarial examples in L_∞ norm with a maximum distortion $\epsilon = 0.031$, a step size $\alpha_{\text{TOG}} = 0.008$, and the number of iterations $T = 10$. Table II gives two visual examples of the adverse effect brought by four different untargeted random attacks on FRCNN. Given that FRCNN can accurately identify the objects in benign (no attack) inputs (1st column), it fails in detecting correctly given adversarial examples (2nd-5th columns). Due to the untargeted random nature, different attacks cause the victim to misbehave differently. For instance, TOG causes FRCNN to recognize false objects in both images while UEA has a vanishing effect on the 1st image but misdetect the two people as one sofa in the 2nd image. Indeed, this class of attacks has no control in the adverse effect imposed on the victim, and the incorrect detection made by the victim is unpredictable (e.g., vanishing, fabricating, mislabeling, or any combination).

Table III provides the quantitative measurements on all victim detectors under the four attack algorithms. Note that DAG, RAP, and UEA can only attack FRCNN, and hence we do not evaluate them on YOLOv3 and SSD models. The comparisons are done in three perspectives: (1) mean Average Precision (mAP) [19], (2) time cost, and (3) distortion cost.

The mean Average Precision (mAP) [19] is a standard evaluation metric to quantify the detection quality of an object detector. All attacks successfully bring down the mAP of the victim. Considering the TOG attack, the mAP (benign) of any victim detector is drastically reduced to less than 2.64% with four victims having a close to zero mAP given adversarial examples. This indicates that the victims fail miserably with no detection capability. Focusing on FRCNN which has an mAP (benign) of 67.37%, TOG is the most powerful attack

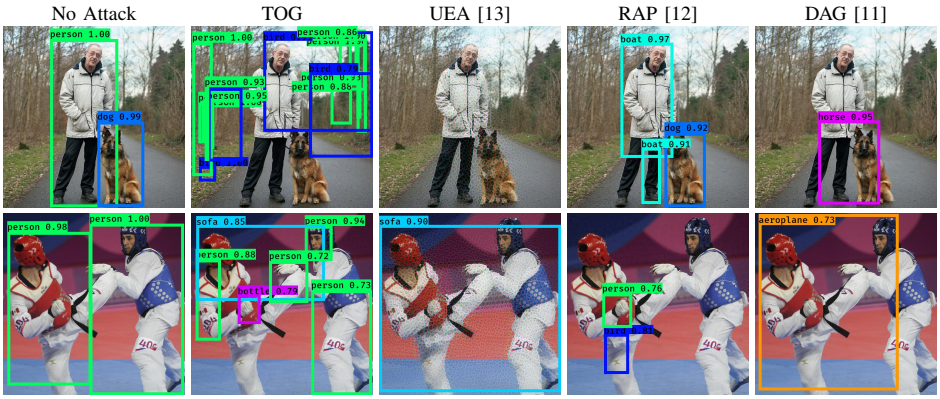


TABLE II: Visualization of untargeted random attacks by four different attack algorithms.

Attack	Victim Detector	mAP (%)		Time Cost (s)		Distortion Cost			
		Benign	Adv.	Benign	Adv.	L_∞	L_2	L_0	SSIM
TOG	YOLOv3-D	83.43	0.56	0.03	0.98	0.031	0.083	0.984	0.875
TOG	YOLOv3-M	71.84	0.43	0.02	0.59	0.031	0.083	0.978	0.876
TOG	SSD300	76.11	0.86	0.02	0.39	0.031	0.120	0.975	0.879
TOG	SSD512	79.83	0.74	0.03	0.69	0.031	0.070	0.974	0.869
TOG	FRCNN	67.37	**2.64**	0.14	1.68	0.031	0.058	0.976	0.862
UEA	FRCNN	67.37	18.07	0.14	**0.17**	0.343	0.191	0.959	0.652
RAP	FRCNN	67.37	4.78	0.14	4.04	0.082	0.010	0.531	0.994
DAG	FRCNN	67.37	3.56	0.14	7.99	0.024	0.002	0.493	**0.999**

TABLE III: Random untargeted attacks on five different victim detectors (VOC dataset).

causing the victim to achieve the lowest mAP of 2.64%, followed by DAG (3.56%), RAP (4.78%), and UEA (18.07%). By default, UEA generates adversarial examples with a fixed resolution of (300×300). When attacking FRCNN taking inputs with resolution of (600×600), resizing and interpolation are required. Hence, the effectiveness of UEA is hindered. In comparison, TOG, RAP, and DAG are much more adaptive and capable of generating adversarial examples that fit the input resolution, as they do not rely on additional networks.

Apart from attack effectiveness, time costs are equally important. For the benign case, we measure the detection time of the victim, while for the adversarial scenario, the time cost includes both adversarial example generation and detection time. UEA [13] has the lowest time cost of 0.17s, slightly higher than the detection time on the benign example (no attack), because UEA uses a generative adversarial network, of much lower complexity compared to the victim detector, to generate adversarial perturbation against the gradient of the input example under attack. TOG can be launched with the second lowest time cost (1.68s) but RAP and DAG have much higher time cost (4.04s and 7.99s) compared to the average detection time for benign inputs. This is primarily due to two factors: (1) the number of iterations required to succeed the attack in RAP and DAG is high, because RAP and DAG need to run more than 30 rounds while TOG needs 10 iterations; and (2) TOG is a more general attack method capable of attacking the objectness loss, the bounding box loss and the classification loss. For the distortion cost, we consider L_0, L_2, L_∞ distances and the structural similarity (SSIM) [24]. Note that L_2 and L_0 costs reported here are normalized by the number of pixels and the L_2 cost has a magnitude of 10^3. Interestingly, spending more iterations allows RAP and DAG to have a much lower distortion cost and exceptionally high SSIM measures of 0.994 and 0.999 respectively. TOG also has a high imperceptibility with SSIM higher than 0.862. In contrast, adversarial perturbations generated by UEA are highly perceptible with a low SSIM of 0.652. Further, RAP and DAG have a low L_0 cost, indicating that their perturbations are more localized. In comparison, both TOG and UEA have the L_0 cost close to 1.000, indicating that most pixels are modified under attack.

B. *Targeted Specificity Attacks*

We next analyze the targeted attacks offered by TOG to fool the victim with three types of specificity: object vanishing, fabrication, and mislabeling. Given that DAG, UEA and RAP attack algorithms do not support targeted specificity attacks, we evaluate TOG algorithms in this set of experiments. Table IV presents a visualization using the same example images from the study of untargeted random attacks in Table II. This provides an interesting comparison on the adverse effect of targeted attacks by TOG-vanishing, TOG-fabrication, TOG-mislabeling (ML) (Most-Likely target) and TOG-mislabeling (LL) (Least-Likely target). Comparing to untargeted random attacks that lead to arbitrary detection faults, the victim detector under TOG targeted specificity attacks is deceived with a persistent and adversarial goal and thus more detrimental. For TOG-vanishing, no objects (person and dog) are detected at all (2nd column). For TOG-fabrication, a large number of false bounding boxes and thus fake objects are detected with high confidence (3rd column). The two TOG-mislabeling attacks (ML and LL) deceive the victim purposefully, which

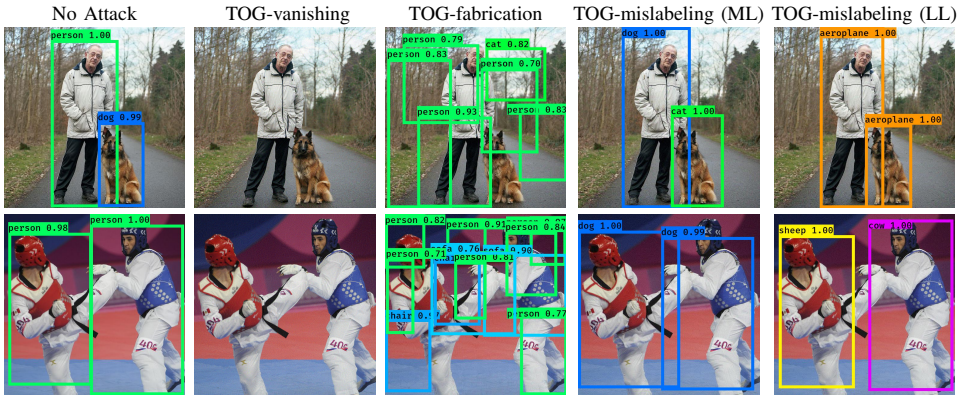


TABLE IV: Visualization of four targeted specificity attacks by TOG.

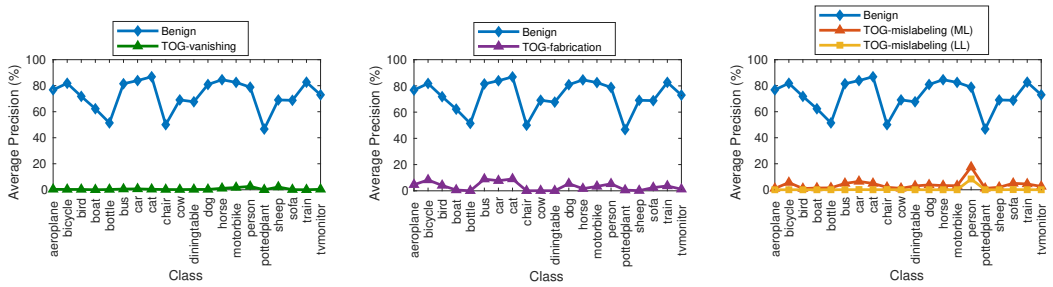
Victim Detector	Targeted Attack	mAP (%)		Time Cost (s)		Distortion Cost			
		Benign	Adv.	Benign	Adv.	L_∞	L_2	L_0	SSIM
YOLOv3-D (Darknet53)	TOG-vanishing	83.43	0.32	0.03	**0.77**	0.031	0.082	0.983	0.877
	TOG-fabrication	83.43	**0.25**	0.03	0.93	0.031	0.084	0.984	0.873
	TOG-mislabeling (ML)	83.43	3.15	0.03	0.95	0.031	0.080	0.972	**0.879**
	TOG-mislabeling (LL)	83.43	2.80	0.03	0.96	0.031	0.081	0.972	**0.879**
YOLOv3-M (MobileNetV1)	TOG-vanishing	71.84	0.36	0.02	**0.37**	0.031	0.082	0.978	0.878
	TOG-fabrication	71.84	**0.17**	0.02	0.57	0.031	0.084	0.976	0.873
	TOG-mislabeling (ML)	71.84	2.67	0.02	0.56	0.031	0.079	0.953	**0.882**
	TOG-mislabeling (LL)	71.84	1.60	0.02	0.56	0.031	0.079	0.953	0.881
SSD300 (VGG16)	TOG-vanishing	76.11	5.54	0.02	**0.36**	0.031	0.120	0.978	0.880
	TOG-fabrication	76.11	**0.57**	0.02	0.37	0.031	0.122	0.978	0.877
	TOG-mislabeling (ML)	76.11	2.53	0.02	0.37	0.030	0.110	0.945	**0.891**
	TOG-mislabeling (LL)	76.11	1.44	0.02	0.37	0.030	0.111	0.945	0.889
SSD512 (VGG16)	TOG-vanishing	79.83	6.23	0.03	**0.62**	0.031	0.071	0.975	0.868
	TOG-fabrication	79.83	**0.50**	0.03	0.69	0.031	0.071	0.976	0.866
	TOG-mislabeling (ML)	79.83	2.53	0.03	0.65	0.031	0.065	0.957	**0.878**
	TOG-mislabeling (LL)	79.83	1.20	0.03	0.65	0.031	0.066	0.956	0.877
FRCNN (VGG16)	TOG-vanishing	67.37	**0.14**	0.14	1.66	0.031	0.058	0.975	0.862
	TOG-fabrication	67.37	1.24	0.14	1.68	0.031	0.057	0.977	0.866
	TOG-mislabeling (ML)	67.37	2.14	0.14	1.64	0.030	0.054	0.935	**0.873**
	TOG-mislabeling (LL)	67.37	1.44	0.14	**1.60**	0.030	0.054	0.935	0.872

TABLE V: Targeted specificity attacks by TOG on five different victim detectors (VOC).

allows the same set of bounding boxes to be detected by the victim detector but each bounding box is associated with an obstinately incorrect class (4th-5th column).

Table V summarizes the quantitative results of 20 attacks from 4 different TOG targeted attack configurations on 5 different victim detectors. All 20 attacks significantly reduce the mAP of the victim detectors at a reasonable cost of time and distortion, similar to that of the untargeted scenario (recall Table III). YOLOv3-M achieves a high mAP of 71.84% given benign examples. However, under any of the TOG targeted attacks, its performance is severely harmed with a low mAP from 0.17% to 2.67%. Figure 6 shows that the victim detector suffers significant utility loss under attack and such adverse effect is equally distributed across all VOC classes. Similar effects can also be observed on COCO dataset for the victim detector YOLOv3-D with a high mAP of 54.16% on benign inputs as shown in Table VI.

Figure 7 shows the total number of objects detected by YOLOv3-D on the entire VOC test set ($4,952$ images) given benign and adversarial examples with different settings of confidence thresholds (recall in Section II that all candidate objects with prediction confidence lower than a threshold will be discarded). First, TOG-vanishing fools the victim to detect much less objects than the benign scenario (the blue curve). Differently, given adversarial examples generated by TOG-fabrication, the victim returns much more objects than usual (benign case). These observations validate that TOG-vanishing and TOG-fabrication deceive the victim with our expected effects: for object-vanishing attacks, the victim should detect almost no object, while for object-fabrication attacks, additional false objects should be mistakenly recognized. Interestingly, the number of detected objects on adversarial examples generated by TOG-mislabeling is almost identical to the benign scenario. This is because TOG-mislabeling has the ability to keep the same set of bounding boxes (objects) and fool the victim to mislabel them. We can conduct quantitative measurements using two metrics: (1) the attack success rate (ASR) measures the percentage of objects that are detected on benign inputs but purposefully mislabeled as their maliciously chosen incorrect class and (2) the mislabeling rate (MR) measures the percentage of objects that are detected on benign inputs but mislabeled as any incorrect class. Note that MR$\geq$ASR. Figure 8 reports the ASR and MR for each source class. We observe that objects of some classes are easier to be purposefully mislabeled. For instance, the ASR of aeroplane



(a) TOG-vanishing (b) TOG-fabrication (c) TOG-mislabeling

Fig. 6: Average precision of each class given benign (blue) and adversarial examples generated by TOG attacks (VOC dataset).

Victim Detector	Targeted Attack	mAP (%)		Time Cost (s)		Distortion Cost			
		Benign	Adv.	Benign	Adv.	L_∞	L_2	L_0	SSIM
YOLOv3-D (Darknet53)	TOG-untargeted	54.16	3.52	0.03	1.02	0.031	0.083	0.986	0.872
	TOG-vanishing	54.16	**0.41**	0.03	**0.78**	0.031	0.082	0.986	0.874
	TOG-fabrication	54.16	1.46	0.03	**0.78**	0.031	0.083	0.986	**0.871**
	TOG-mislabeling (ML)	54.16	5.43	0.03	1.00	0.031	0.080	0.968	0.878
	TOG-mislabeling (LL)	54.16	0.76	0.03	1.00	0.031	0.080	0.968	0.877

TABLE VI: TOG attacks on YOLOv3-D (COCO dataset).

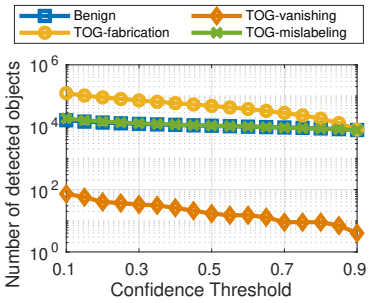


Fig. 7: The number of detected objects on VOC given different input images with varying t_{conf}.

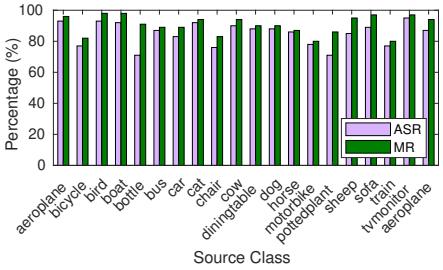


Fig. 8: The attack success rate (ASR) and mislabeling rate (MR) of TOG-mislabeling (VOC dataset).

is 91%, which is much higher than that of bottle (70%). On average, 80% of the objects of any class can be purposefully mislabeled, showing the effectiveness of TOG-mislabeling.

C. Evaluation of TOG-universal

We evaluate the effectiveness of TOG-universal on VOC and COCO with the same set of hyperparameters: a maximum distortion of 0.031, an attack learning rate of 0.0001, and 50 training epochs. The first 12,800 images from the training set are extracted to form $\mathcal{D}$. We train three universal adversarial perturbations (YOLOv3-M and SSD512 for VOC and YOLOv3-D for COCO) using TOG-universal with an object-vanishing configuration.

Dataset	Detector	mAP (%)		Time Cost	
		Benign	Adv.	Offline (hr)	Online (s)
VOC	YOLOv3-M	71.84	20.44	8.05	0.00136
	SSD512	79.83	52.29	9.06	0.0001
COCO	YOLOv3-D	54.16	12.73	14.15	0.00183

TABLE VII: Evaluation of TOG-universal.

To demonstrate the effectiveness under different real-world situations, such as light traffic v.s. heavy traffic, simple scene v.s. complex scene, we use the YOLOv3-D detector on COCO to detect objects in a video clip of a driving scene[1]. Our attack successfully reduces the number of identified objects from 398,048 to 54, showing the high success rate and serious threat to systems using deep object detectors.

Table VII summarizes the quantitative evaluation of TOG-universal in mAP and time cost. We apply the same adversarial perturbation on all images of the respective test set, and observe a significant reduction of mAP from 71.84% to 20.44% for YOLOv3-M on VOC and from 54.16% to 12.73% for YOLOv3-D on COCO. It also reports that TOG-universal can reduce the mAP of SSD512 on VOC from 79.83% to 52.29%. While it is not as drastic as YOLOv3-M, our attack can reduce mAP by 27%, seriously affecting the detection performance. TOG-universal takes time to iteratively train an universal adversarial perturbation. Upon the completion of training, its attack during the online detection phase takes only a millisecond (Table VII) to simply add the pre-computed universal perturbation to the input image. Such a lightweight attack can be easily handled at the edge as no optimization is involved and can be done in real-time as shown in our demo video of the driving scene where the frame-per-second (FPS) of detection on the adversarial video (about 39 FPS) is very close to that on the benign video (about 42 FPS).

[1]https://youtu.be/_IZfQofiL9Q

Adversarial Patch	% of Objects Vanished
Random Patch	1.49
TOG-patch (vanishing)	97.35

TABLE VIII: Percentage of objects vanished by TOG-patch on INRIA dataset compared with a random patch.

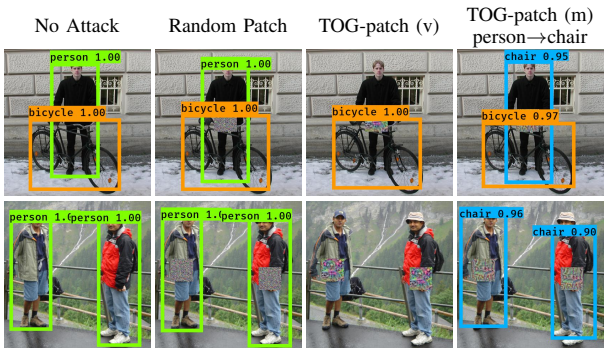


TABLE IX: Two images attached with a random patch (2nd column) and two adversarial patches trained by the TOG-patch algorithm with object-vanishing (3rd column) and object-mislabeling (4th column) effects (INRIA dataset).

D. Evaluation of TOG-patch

We evaluate TOG-patch with vanishing and mislabeling effects. Without fine-tuning hyperparameters for each case, we generate adversarial patches of size (64×64) with 30 epochs, a batch size of 8, and a learning rate of 0.1. Following [22], the INRIA dataset for human detection [21] is exploited with YOLOv3-D as the victim detector. The trained TOG-patch is placed at the center of the person detected on a benign test image sent to the victim to generate its adversarial example. Table VIII reports the percentage of vanished objects (person) by using a vanishing patch or a random patch filled with random noise. The victim detector can still correctly detect objects (person) attached with the random patch (see the 2nd column in Table IX). Only 1.49% of objects (person) vanish from the detection by the victim. TOG-patch (vanishing) is remarkably effective (see the 3rd column in Table IX), achieving 97.35% of objects (person) vanished from the victim's detection. For TOG-patch with a mislabeling effect, we train 19 adversarial patches, aiming at deceiving the victim to mislabel the person with the patch as one of the 19 different target classes on VOC (see the 4th column in Table IX with "chair" as the target class). Figure 9 shows that such highly aggressive adversarial patches fool the victim to purposefully mislabel 80% (on average) of detected objects.

E. Transferability Studies

We compare the adversarial transferability of TOG with other three untargeted random attacks: UEA, RAP and DAG. Table X reports the results measured in mAP. Using the same model to craft adversarial examples always achieves the highest transferability, as indicated in boldface. We first focus on TOG and consider its transferability by sending adversarial examples generated on different source models to different

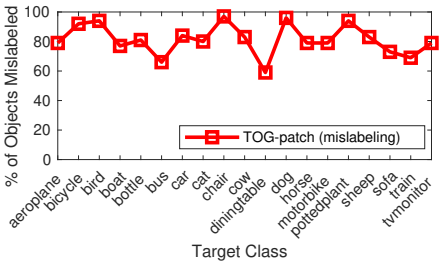


Fig. 9: Percentage of objects mislabeled by TOG-patch with different target classes (INRIA dataset).

Transfer Attack	Source Model	Target Model				
		YOLOv3-D	YOLOv3-M	SSD300	SSD512	FRCNN
Benign (No Attack)		83.43	71.84	76.11	79.83	67.37
TOG	YOLOv3-D	**0.56**	60.13	72.70	73.86	55.57
TOG	YOLOv3-M	74.62	**0.43**	73.27	75.27	59.1
TOG	SSD300	56.87	42.85	**0.86**	38.79	50.36
TOG	SSD512	56.21	46.00	58.00	**0.74**	35.98
TOG	FRCNN	79.47	68.60	75.80	78.09	**2.64**
UEA	FRCNN	51.92	31.88	47.08	47.66	**18.07**
RAP	FRCNN	81.80	69.45	75.77	76.84	**4.78**
DAG	FRCNN	81.21	70.37	75.15	78.38	**3.56**

TABLE X: Transferability of adversarial examples generated by untargeted attacks (VOC dataset).

target models (the 2nd-6th rows). We observe that adversarial examples generated by TOG on SSD models have high transferability than others. For instance, adversarial examples from SSD300 and SSD512 can reduce the mAP of YOLOv3-D from 83.43% to 56.87% and 56.21%, much better than adversarial examples generated on YOLOv3-M and FRCNN that only reduction to 74.62% and 79.47% are recorded. Note that with adversarial transferability, the attacks are black-box, generated and launched without any prior knowledge of the target (victim) models [25]. Considering the transferability of different attack algorithms with the same source model FRCNN (the last four rows), we find that adversarial examples by UEA exhibit a higher transferability consistently. This can be attributed to its high distortion cost incurred to perturb each adversarial example (recall Table III).

V. Related Work

Existing adversarial attacks on object detection systems often attack a specific structure in object detection networks. In particular, DAG [11] manipulates the region proposal network (RPN) in two-phase detection algorithms (e.g., Faster R-CNN [3]) to generate a large number of proposal regions. Then, an iterative gradient-based attack is launched with the goal of fooling the victim to mislabel those regions as randomly chosen incorrect class labels. RAP [12] targets at disabling the functionality of RPN in two-phase algorithms. It fools the RPN to not returning foreground objects and to produce wrong estimation on the bounding box even if foreground objects are proposed. UEA [13] extends DAG with a feature loss to train a generative adversarial network for generating adversarial examples. Both of the above can cause the victim to randomly misdetect (untargeted) without maintaining a specified specificity (e.g., vanishing, fabricating,

or mislabeling objects). They can only directly attack two-phase approaches but not one-phase techniques where no RPN is used. Differently, TOG is a suite of attacks that can not only support untargeted random attacks but also targeted ones with three types of specificity as demonstrated in our experiments. As no specific structure is manipulated and it works by strategically configuring the adversarial loss and auxiliary target detection in Equation 7, TOG is applicable to both two-phase and one-phase algorithms.

Apart from adversarial perturbation to the input image, adversarial patches refer to those attacks that can be launched in either a digital or physical form. DPATCH [15] puts a small patch on a benign example, fooling the victim to fabricate objects at random position or the location where the patch is placed. Extended-RP_2 [14] and Thys's Patch [22] propose printable adversarial patches. By physically presenting the patch in the scene captured by the camera, the captured image will become adversarial and fool the victim to misdetect. In particular, Extended-RP_2 supports "disappearance" and "creation", corresponding to the object-vanishing and object-fabrication effects, while Thys's Patch aims to make the object vanishing from the detector.

VI. Conclusion

We have presented TOG, a family of adversarial objectness gradient attacks on deep object detection networks. TOG attacks generate adversarial perturbations in three alternative forms: (i) by employing adversarial perturbation optimized for each input, (ii) by offline training a universal perturbation that is effective on any inputs to the victim detector, and (iii) by optimizing a confined adversarial patch. In addition to compare TOG with other white-box attack algorithms, we also studied black-box attacks via the adversarial transferability from one detector to others. Extensive experiments have been conducted on three benchmark datasets (VOC, COCO, INRIA) to show and analyze adversarial vulnerabilities of the three state-of-the-art families of deep object detection networks (YOLOv3, SSD, and Faster R-CNN). Based on our formal and experimental studies on the vulnerabilities of different victim detectors and the adversarial transferability of representative attack algorithms, we observe the divergence of attack effects on different detectors, e.g., the weak spot of attack transferability tends to vary from one detector to another. The more diverse from the source victim detector an object detector is, the more robust it is against the black-box attack, and the less transferable from the attack generated from the source victim detector [26], [27]. We are working on developing an attack-resilient object detection system by exploiting diversity-enhanced ensemble learning for strong robustness and high defensibility.

Acknowledgment

This research is partially sponsored by National Science Foundation under NSF 1564097, NSF 2038029, a Cisco grant, and an IBM faculty award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other funding agencies and companies mentioned above.

References

[1] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *ECCV*, 2016.
[2] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.
[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NIPS*, 2015.
[4] M. Simon, K. Amende, A. Kraus, J. Honer, T. Samann, H. Kaulbersch, S. Milz, and H. Michael Gross, "Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds," in *CVPR Workshops*, 2019.
[5] R. Platania, S. Shams, S. Yang, J. Zhang, K. Lee, and S.-J. Park, "Automated breast cancer diagnosis using deep learning and region of interest detection (bc-droid)," in *ACM BCB*, 2017.
[6] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *ICCV*, IEEE, 1998.
[7] R. Girshick, "Fast r-cnn," in *CVPR*, 2015.
[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.
[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016.
[10] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *CVPR*, 2017.
[11] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic segmentation and object detection," in *ICCV*, 2017.
[12] Y. Li, D. Tian, X. Bian, S. Lyu, *et al.*, "Robust adversarial perturbation on deep proposal-based models," *BMVC*, 2018.
[13] X. Wei, S. Liang, N. Chen, and X. Cao, "Transferable adversarial attacks for image and video object detection," in *IJCAI*, 2019.
[14] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, T. Kohno, and D. Song, "Physical adversarial examples for object detectors," *arXiv preprint arXiv:1807.07769*, 2018.
[15] X. Liu, H. Yang, Z. Liu, L. Song, H. Li, and Y. Chen, "Dpatch: An adversarial patch attack on object detectors," in *SafeAI*, 2019.
[16] J. Lu, H. Sibai, and E. Fabry, "Adversarial examples that fool detectors," *arXiv preprint arXiv:1712.02494*, 2017.
[17] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR*, 2014.
[18] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *CVPR*, 2017.
[19] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *IJCV*, vol. 111, no. 1, pp. 98–136, 2015.
[20] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014.
[21] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, IEEE, 2005.
[22] S. Thys, W. Van Ranst, and T. Goedemé, "Fooling automated surveillance cameras: adversarial patches to attack person detection," in *CVPR*, 2019.
[23] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 acm sigsac conference on computer and communications security*, pp. 1528–1540, 2016.
[24] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 2, pp. 1398–1402, Ieee, 2003.
[25] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.
[26] K.-H. Chow, W. Wei, Y. Wu, and L. Liu, "Denoising and verification cross-layer ensemble against black-box adversarial attacks," in *IEEE International Conference on Big Data*, 2019.
[27] L. Liu, W. Wei, K.-H. Chow, M. Loper, E. Gursoy, S. Truex, and Y. Wu, "Deep neural network ensembles against deception: Ensemble diversity, accuracy and robustness," in *MASS*, IEEE, 2019.