Abstract—Object detection systems using deep learning models have become increasingly popular in robotics thanks to the rising power of central processing units (CPUs) and graphics processing units (GPUs) in embedded systems. However, these models are susceptible to adversarial attacks. While some attacks are limited by strict assumptions on access to the detection system, we propose a novel hardware attack inspired by Man-in-the-Middle attacks in cryptography. This attack generates a universal adversarial perturbations (UAPs) and injects the perturbation between the universal serial bus (USB) camera and the detection system via a hardware attack. Besides, prior research is misled by an evaluation metric that measures the model accuracy rather than the attack performance. In combination with our proposed evaluation metrics, we significantly increased the strength of adversarial perturbations. These findings raise serious concerns for applications of deep learning models in safety-critical systems, such as autonomous driving.

Impact Statement—Advancements in deep neural networks have ushered in a new era of robotics, characterized by intelligent robots with a comprehensive understanding of the environment, thanks to deep learning models. However, it is no more a secret that deep learning models are vulnerable to adversarial attacks. Besides existing digital and physical attacks, we introduce a novel “Human-in-the-Middle” hardware attack that injects digital perturbation into the physical sensor. Our research opens up new possibilities for adversarial attacks, and we hope to embrace deep learning models securely for robotic applications.

Index Terms—Adversarial attacks, deep learning, object detection.

I. INTRODUCTION

The development of deep neural networks has enabled the creation of intelligent robots that possess a more comprehensive perception of the environment than traditional robots. However, this shift toward intelligent robots has also brought with it an increasing risk of adversarial attacks, especially in safety-critical applications. It has been a decade since the existence of adversarial examples was first identified by Biggio et al. [1] and Szegedy et al. [2], in which they fooled an image classification model by adding a small perturbation to the input image. Although the perturbation was imperceptible to humans, it caused the deep learning model to produce erroneous classification results. The attack was later extended from classification models to detection models [3], [4].

Adversarial attacks against deep learning models can be divided into two categories: digital attacks and physical attacks. Digital attacks directly apply perturbations to the digital input image by modifying pixel values [5], while physical attacks involve printing the perturbation on physical objects such as posters [6] or T-shirts [7].

However, both digital and physical attacks have their limitations. Digital perturbation requires access to the detection system, making it difficult to apply in real-world scenarios such as hacking into a self-driving car. Physical attacks, on the other hand, are sensitive to position and angle variations. For instance, experiments in [4] showed that an autonomous vehicle only misclassified traffic signs placed within 0.5 m of the camera and viewed from specific angles. Moreover, these attacks lack flexibility, as once the adversarial object is printed, it can only be changed through reprinting. The trial-and-error process of finding a successful attack object can take a long time and require significant amounts of printing.

A. Contributions

This article presents a novel hardware attack that combines the flexibility of physical attacks with the efficiency of digital attacks, inspired by Man-in-the-Middle attacks in network security [refer to Fig. 1(a)]. In this attack, the adversary intercepts and manipulates the image data transmitted between a universal serial bus (USB) camera and a detection system [refer to Fig. 1(b) and 1(c)].

The key contributions of this research are summarized as follows.

1) We present a novel hardware attack, called the Human-in-the-Middle (HitM) attack, that offers both efficiency and ease of deployment for adversarial attacks¹. By utilizing learning rate decay during the generation of the perturbation, our attack is capable of generating more bounding boxes than competing attack methods.

2) We introduce three new evaluation metrics that offer a more nuanced approach to evaluating adversarial attacks.

¹The source code of the hardware attack is available on GitHub: https://github.com/wuhanstudio/adversarial-camera
Unlike existing metrics that make a binary decision for each bounding box, our metrics consider the confidence value and probability vector in a linear fashion.

3) We devise and open source the white-box adversarial toolbox\(^2\) which simplifies the process of generating adversarial perturbations. The toolbox focuses on real-time white-box attacks against object detection models.

II. PRELIMINARIES

A. Object Detection Models

The task of object detection aims to locate the position and classify the category of each object in an image. Therefore, the task consists of two distinct problems: localization and classification. Existing object detection models can be categorized into two types, one-stage and two-stage methods, based on whether these two problems are solved together or separately [8]. Two of the most widely deployed one-stage models are you only look once (YOLO) [9], [10], and [11], and single shot detection (SSD) [12], which can achieve real-time performance on central processing units (CPUs) without graphics processing units (GPUs). Faster region-based convolutional neural network (RCNN) [13] and Mask RCNN [14] are two well-known two-stage models.

In robotic applications, one-stage models are generally preferred due to their speed and acceptable accuracy in most situations. In this study, we investigate how these attacks affect real-time robotic applications and focus on energy-efficient one-stage models.

\(^2\)The source code of the toolbox is available on GitHub: https://github.com/wuhanstudio/whitebox-adversarial-toolbox

B. Adversarial Attacks

The fast gradient sign method (FGSM) [15] was the first adversarial attack against classification models that uses gradients of deep neural networks to generate image-specific perturbations, which is more efficient than optimization-based methods proposed by Biggio et al. [1] and Szegedy et al. [2].

However, for real-world robotic applications, it is more practical to use universal adversarial perturbations (UAPs) [16], [17], [18], which are image-agnostic. UAPs demonstrated the ability to fool classification models on most images in a dataset using a single perturbation. Adversarial attacks have since been extended from image classification to detection models [3], [19].

In addition to image-specific and image-agnostic methods, it is also possible to classify adversarial attacks into data-driven and data-independent approaches. Data-driven approaches require access to the input image, while data-independent methods do not need access to the input data. Both approaches may need access to the parameters and the architecture of the target model, depending on whether they are white-box attacks or black-box attacks. Generally, data-driven approaches achieve a higher fooling rate as they have more information at their disposal. Data-driven methods include gradient-based methods [17], [20], [21], [22], methods using generative adversarial networks (GANs) [23], [24], and optimization-based methods [25], [26].

To deploy the UAPs generated using the aforementioned methods, Wang et al. categorized existing physical attacks into invasive attacks and noninvasive attacks [27]. Invasive attacks deploy the perturbation by attaching a patch [6], [7], [28], [29], or changing the texture [30], [31] of the target object. For example, by adding extra constraints, such as the subsampled nonprintability score (SNPS), to the loss function of data-driven methods, we can generate physical perturbations that preserve the adversarial effect if printed out on a poster [6]. On the other hand, noninvasive attacks perform physical attacks by changing the environment rather than the target object. For example, Gnanasambandam et al. fool classification models by changing the illumination of the environment using a low-cost projector [32], and Zhong et al. exploit the shadow to fool deep learning models [33].

We propose a novel HitM hardware attack that neither modifies the target object (invasive attacks) nor changes the illumination or shadow of the environment (noninvasive attacks). Instead, our attack injects the perturbation into the physical communication channel, combining the advantages of digital and physical attacks. Wang et al. propose a Main-in-the-Middle attack against machine-learning-as-a-service applications that exploit vulnerabilities in networking to stealthily manipulate the submitted data, which is a traditional network attack [34]. In a parallel research endeavor, Liu et al. focus on local-search-based black-box attacks against image classification models and inject adversarial strips to a mobile industry processor interface (MIPI) camera using an FPGA [35]. On the other hand, we focus on gradient-based white-box attacks against object detection models and inject UAPs to a USB camera via an ARM Linux board, as depicted in
Despite the differences in their structures, all these models and two-stage models (e.g., Faster-RCNN and Mask-RCNN) can be categorized into one-stage models (e.g., YOLO and SSD) and two-stage models. This distinction is important because it affects the way we approach adversarial perturbations.

III. THE HitM ATTACK

This section introduces the PCB attack, a novel gradient-based method designed to generate image-agnostic UAPs. The name “PCB” comes from the fact that the output of the object detection model is separated into three components: probability vector (P), confidence value (C), and bounding boxes (B). The perturbation is then applied using a hardware attack. The acronym PCB is fitting for a hardware attack, as it is also used to refer to printed circuit boards (PCBs).

A. Problem Formulation

In Section II, we discussed that object detection models can be categorized into one-stage models (e.g., YOLO and SSD) and two-stage models (e.g., Faster-RCNN and Mask-RCNN). Despite the differences in their structures, all these models share common inputs and outputs. To describe these inputs and outputs, we introduce the following mathematical notation.

1) \( x \): The original clean input image.
2) \( \delta \): The adversarial perturbation.
3) \( x' \): The adversarial input image \( x' = x + \delta \).
4) \( K \): The total number of candidate classes.
5) \( N \): The total number of candidate bounding boxes.
6) \( O(x) \): The output of \( N \) candidate bounding boxes from the model given the input image.
7) \( o_i(x) \): The \( i \)th output in \( O(x) = \{o_1, o_2, o_3, \ldots, o_N\} \), where \( o_i = (b_i, c_i, p_i) \), \( 1 \leq i \leq N \).
8) \( b_i \): The location and dimension of the \( i \)th candidate box. \( b_i = (b_{ix}^i, b_{iy}^i, b_{iw}^i, b_{ih}^i) \) represents a bounding box at position \( (b_{ix}^i, b_{iy}^i) \) with width \( b_{iw}^i \) and height \( b_{ih}^i \).
9) \( c_i \): The confidence value (objectness) of the \( i \)th candidate box that represents how probable it is that the candidate box represents an object.
10) \( p_i \): The softmax probability vector of the \( i \)th candidate box. \( p_i = (p^1_i, p^2_i, \ldots, p^K_i) \) for \( K \) classes and \( \sum p_i = 1 \).

Given an input image \( x \), the object detection model outputs \( N \) candidate bounding boxes \( O(x) = \{o_1, o_2, o_3, \ldots, o_N\} \). Each candidate box \( o_i = (b_i, c_i, p_i) \) contains \( b_i = (b_{ix}^i, b_{iy}^i, b_{iw}^i, b_{ih}^i) \) that represents the location and dimension of the box, the confidence value \( c_i \in [0, 1] \) that represents how probable it is that the candidate box represents an object, and the softmax probability vector, \( p_i = (p^1_i, p^2_i, \ldots, p^K_i) \) for \( K \) classes. The raw outputs from the detection model \( O(x) \) may contain several thousand candidate bounding boxes. We then use the nonmaximum suppression (NMS) method [36] to filter out bounding boxes with low confidence values, and high intersection over union (IoU) to generate final detection results.

An adversarial example \( x' = x + \delta \) aims to fool the detection model by making it output candidate boxes \( O(x') \) that are different from the candidate boxes \( O(x) \) output by the model for the original input image \( x \). For example, the adversarial output \( O(x') \) may detect more false positive objects after the NMS process. To achieve this, we need to generate a perturbation \( \delta \) that can be added to the original image \( x \) to produce the adversarial image \( x' \). In the following sections, we will describe how to generate the perturbation \( \delta \) using the proposed PCB attack.

B. Generating the Perturbation (The PCB Attack)

Gradient-based methods use a similar approach to generate both image-specific and image-agnostic perturbations. For image-specific perturbations, given an input image, the method iterates over a single image to produce the perturbation. For image-agnostic perturbations, given the entire dataset, the method iterates over multiple images to generate the UAP. We will first describe how we generate the image-specific perturbations and then extend the attack to generate the image-agnostic perturbations.

1) Image-Specific PCB Attack: The intuition behind gradient-based methods is straightforward. During the training process, we minimize the training loss

\[
\min_{W} L_{\text{train}} = f(W; x, O) \tag{1}
\]

by updating the model weights. Note that the training loss \( f \) is a function of the input \( x \), the model weights \( W \), and the ground truth \( O \).

However, our objective is to fool the detection model to make inaccurate predictions. Therefore, during the attack, we maximize the adversarial loss

\[
\max_{x} L_{\text{adv}} = f(x; O^*, W) \tag{2}
\]

by updating the input \( x \) and using the desired adversarial outputs \( O^* \). Different gradient-based methods use different adversarial loss functions \( L_{\text{adv}} \) and construct desired adversarial outputs \( O^* \) differently. In our attack, we separate the probability vector and confidence value (PC) with bounding boxes (B) and investigate the two adversarial loss functions

\[
L_{\text{PC}}(x) = \sum \sigma(c_i) * \sigma(p_i) \tag{3}
\]

and

\[
L_{\text{PCB}}(x) = \frac{\sum (\sigma(c_i) * \sigma(p_i))}{\sum [\sigma(w_i) * \sigma(h_i)]^2} \tag{4}
\]

where \( \sigma(\cdot) \) is the sigmoid function.
Algorithm 1: Image-specific PCB Attack

1: Input: The target model, the input image \( x \).
2: Parameters: The learning rate \( \alpha \), learning rate decay \( k \), number of iterations \( n \), and strength of the attack \( \epsilon \).
3: Output: Image-specific perturbation \( \delta \)
4: Initialize \( \delta \leftarrow 0 \)
5: for \( i = 1 : n \) do
6: \( x' = x + \delta \)
7: \( \nabla = \frac{\partial L_{\text{adv}}(x';O^*)}{\partial x} \)
8: \( \delta \leftarrow \delta + \alpha \times \text{sign}(\nabla) \)
9: \( \delta \leftarrow \text{clip}(-1,1) \)
10: \( \delta \leftarrow \text{proj}_\infty(\delta,\epsilon) \)
11: // Learning Rate Decay*
12: \( \alpha = \alpha \times k \)
13: end for

By maximizing the adversarial loss [\( L_{\text{PCB}}(x) \) and \( L_{\text{PC}}(x) \)], we generate large amounts of incorrect bounding boxes (fabrication attack). By minimizing the loss, we remove bounding boxes (vanishing attack). Using \( L_{\text{PCB}}(x) \) gives smaller bounding boxes, while \( L_{\text{PC}}(x) \) gives larger ones.

The optimization of (2) is performed by first zero-initializing the perturbation \( \delta \), and then using projected gradient descent (PGD) [37] with learning rate decay in every iteration \( t \), so that

\[
\delta_{t+1} = \text{proj}_\infty \left( \delta_t + \alpha \times \text{sign} \left( \frac{\partial L_{\text{adv}}(x_t';O^*)}{\partial x_t} \right) \right). \tag{5}
\]

The image-specific PCB attack is summarized in Algorithm 1, where \( \text{proj}_\infty \) is the projection function \( \min(\delta,\epsilon) \) using \( l_\infty \) norm, and \( \text{clip}(-1,1) \) is the unit clip function.

2) Image-Agnostic PCB Attack: We can extend the method to an image-agnostic attack by iterating over a collection of images \( X_s = x_1, x_2, \ldots, x_n \), where \( n \) is the number of available images to the attacker. \( X_s \) can be thought of as the training set or a video clip from the target scene. Initially, we generate a random or zero-initialized perturbation \( \delta \) that is of the same dimension as the input of the detection model.

In each iteration, we update \( \delta \) using the gradient of input with respect to \( L_{\text{adv}} \). The learning rate \( \alpha \) is relatively small compared to the image-specific PCB attack to ensure that the perturbation is universal across images. We summarize the image-agnostic PCB attack in Algorithm 2.

C. Applying the Perturbation (The Hardware Attack)

In Section I, we mentioned that conducting digital attacks can be challenging due to the lack of access to the internal system. Input images are often resized and processed by intermediate components before being fed into the detection system. Therefore, an attacker needs to penetrate the operating system and inject malicious code into the embedded system.

To address this problem, the HitM hardware attack was developed. By eavesdropping and manipulating the image data before it reaches the detection system, the perturbation can be applied without access to the operating system. Unlike physical attacks, the hardware attack is robust to position and angle variations because the perturbation is injected by directly modifying pixel values. Thus, the perturbation won’t be sheared when viewing it from different angles.

To implement the hardware attack, specialized hardware such as Raspberry Pi ZeroV2 or LMX6UL is required, which can read raw images from the USB camera and then inject the perturbation (see Fig. 3). To conceal the attack from the operating system, a virtual camera needs to be simulated for the detection system. This requires a Linux kernel that supports the V4L2 driver, the USB gadget framework, and configfs.

IV. EXPERIMENTAL EVALUATION

This section aims to provide insight into why the mAP not suitable for evaluating adversarial attacks. For adversarial attacks, the choice of the adversarial loss function determines the type of attack to be conducted (e.g., fabrication or vanishing), whereas the strength of the attack is determined by the iterative optimization process. In this study, we employ our novel evaluation metrics to investigate the iterative optimization process and achieve more efficient attacks against a one-stage detection model, namely YOLO, on the VOC2012 [38] and CARLA [39] datasets.

A. Evaluation Metrics

The mAP [40] is typically used to both to measure the accuracy of object detection models and to evaluate the strength of adversarial attacks. However, it can be noticed that the mAP cannot distinguish between different attacks.

For example, both the fabrication and vanishing attacks result in an mAP \( \approx 0 \), even though they serve different attacking purposes [see Fig. 4(a)]. Similarly, while an attacker will prefer a stronger attack (Dog 99%) over a weaker attack (Dog 60%),
Fig. 3. Architecture of the HitM hardware attack and its differences from physical and digital attacks. (a) Design of the embedded system. (b) Physical attack, hardware attack, and digital attack.

Fig. 4. Limitations of mean average precision (mAP). (a) mAP cannot distinguish between fabrication and vanishing attacks (both mAP = 0). (b) mAP does not consider confidence values (both mAP = 0). mAP does not reflect the strength of an attack [see Fig. 4(b)].

\[
\frac{1}{N} \sum_{i=1}^{N} (c_{t,i} - c_{t-1,i})
\]

where \( c_{t,i} \) is the confidence value of the \( i \)-th bounding box at iteration \( t \), and \( c_{t-1,i} \) is the confidence value of the same bounding box at iteration \( t-1 \). This metric reflects the strength of the attack on the confidence value and is expressed as follows:

3) **Relative Box Variation:** After each iteration, the position of false positive bounding boxes fluctuates. This metric measures the percentage of consistent bounding boxes (bounding boxes that have the same position as in the previous step) at the current step and can be expressed as follows:

\[
\left| \frac{\text{NMS}(\mathcal{O}(x'_t)) + \text{NMS}(\mathcal{O}(x'_{t-1})) - \text{NMS}(\mathcal{O}(x'_t), \mathcal{O}(x'_{t-1}))}{\text{NMS}(\mathcal{O}(x'_t))} \right|
\]

Further, we compare our attack with the targeted objectness gradient (TOG) attack [20], which is also a gradient-based attack that updates the adversarial perturbation using the gradient of input with respect to the adversarial loss function. The main difference is that the TOG attack uses uniform initialization and updates the perturbation with a constant learning rate (no learning rate decay). Besides, the TOG attack uses the YOLO training loss [10], [11] as the adversarial loss function. We study the effect of these factors in the following sections, respectively.

**B. Initialization Methods**

The TOG attack employs uniform initialization, as noted in [20]. In contrast, other attacks use zero initialization, including [17], [37], [41]. Gradient-based attacks rely on gradients to iterate from the original image to an adversarial input. The uniform initialization may impede the initial gradient at the first iteration, potentially limiting the attack’s effectiveness. To investigate this hypothesis, we conducted 10 runs of the attack with uniform initialization and compared the results with zero initialization using the mean confidence variation, number of boxes, and relative box variation.

Our results showed that, for the first two evaluation metrics, only two out of ten runs with uniform initialization resulted in a more effective attack than zero initialization. This supports our hypothesis that uniform initialization can impede the initial gradient and hinder the attack’s effectiveness. Regarding the third evaluation metric, relative box variation, we observed convergence to 1 for both initialization methods, indicating that all false positive bounding boxes were stable, and the convergence speed was similar for both initialization methods.

**C. Learning Rate Decay**

To achieve a stable and efficient adversarial attack, it is necessary to avoid gradient counteraction, which can cause the attack
to vary significantly over iterations. The PCB attack addresses this issue by introducing the learning rate decay factor $k$ to stabilize the attack over different iterations.

Neither the original PGD attack nor the TOG attack uses the learning rate decay and thus has an unstable iteration process (as shown by the black line in Fig. 6). This issue has not been thoroughly studied in prior research that relies on mAP as the evaluation metric. For example, at each step, we generate false positive bounding boxes at various positions, but none of them match the ground truth (mAP = 0). As a result, even though the location of bounding boxes varies a lot (unstable) from one iteration to the next, the mAP stables at 0, which does not indicate the instability of the iteration process.

Using our new evaluation metrics, we can observe the complete iteration process (as depicted in Fig. 6). As the learning rate decay factor $k$ decreases from 0.99 to 0.90, more iterations are required before the two evaluation metrics, the mean confidence variation and the number of boxes, converge. However, both metrics converge at higher values when $k$ decreases, indicating a more efficient attack.

D. Adversarial Loss Function

In this section, we aim to compare the effectiveness of three different adversarial loss functions for the fabrication attack (PC, PCB, and TOG adversarial loss). Rather than determining the best loss function, our goal is to use our evaluation metrics to highlight the advantages and disadvantages of each method.

As shown in Fig. 7, we observe that the relative box variation of all three methods converges to 1, indicating that the locations of most bounding boxes are stable in the final iterations. Though the PC attack generates the most bounding boxes and achieves the highest mean confidence variation when $k = 0.99$, it requires a larger number of iterations to reach the plateau, which can be computationally expensive when the number of sample images $X_s$ is large.

In contrast, both the TOG and PCB attacks converge faster and perform better than the PC attack when $k = 0.90$. Therefore, we cannot definitively state that one method is superior to the others. Instead, the newly proposed evaluation metrics provide useful references for decision-making.

E. The Attack Performance and Transferability

We used an autonomous driving dataset collected from the CARLA simulator [39]. The dataset includes driving records from seven maps, including different areas (city, rural, urban, and highway), daytime (noon, sunset, and night), and weather (clear, cloudy, soft rain, and hard rain) (see Fig. 8). A 30-s driving video was collected from each map, sampling at 10 FPS, which is the same as the KITTI autonomous vehicle.
For the image-specific PCB attack, we measured the performance of the attack on an NVIDIA RTX 2080Ti GPU. The image-specific attack achieved 12.44 FPS, faster than the 10 FPS sampling rate of the KITTI autonomous vehicle, thus achieving a real-time attack.

For online attacks against video streams, it is unnecessary to regenerate the adversarial perturbation for each frame because there is a high correlation between consecutive video frames. Thus, we can reuse the perturbation computed from the previous frame and iterate only one step for each frame to achieve real-time performance.

For the image-agnostic attack, we trained the UAPs using Map 01. The same attack strength ($\epsilon = 8$) is used for both PCB and TOG attacks. Since the original TOG attack does not use learning rate decay, we set $k = 1.00$. After each iteration, we tested the attack performance on all seven maps using the three evaluation metrics (see Table 1).

1) The mean confidence variation reflects the changes in the model outputs (before softmax) after applying the UAP. The experimental results show that our image-agnostic PCB attack produces higher variations than TOG, which indicates a stronger UAP. Besides, our method has a more stable iteration process, while the TOG generates the UAP in a more stochastic way (similar to Fig. 6).

2) The average number of boxes measures the total number of bounding boxes after applying the UAP. The image-agnostic attack generates the most number of boxes on the training map but also generalizes to other maps. With learning rate decay, the PCB attack generates significantly more bounding boxes than the TOG attack.

3) The relative box variation computes the percentage of consistent bounding boxes. During the UAP training process, as the UAP becomes stable, this metric converges to 100%, meaning all bounding boxes are consistent. However, to measure the final attack performance, a high value (100%) indicates the model generates the same bounding boxes before and after applying the UAP. Thus, a successful attack has a low value (0%), indicating none of the bounding boxes are the same as the ones predicted without applying the UAP.

We also measured the transferability of the UAPs across maps (100 iterations, $k = 0.98, \epsilon = 8$). The image-agnostic PCB attack achieves the best performance on the training map but also generalizes to other maps (see Fig. 9). In summary, experimental results using our evaluation metrics demonstrate the importance of learning rate decay for UAP training.

V. DISCUSSION

In real-world applications, the operating system that deploys the deep learning models is highly secure, while sensors are exposed to the environment to collect the data. Besides
TABLE I
EVALUATION RESULTS OF THE IMAGE-AGNOSTIC ATTACK TESTED ON THE CARLA DATASET ($\epsilon = 8$)

| Iteration | PCB Attack ($k = 0.98$) | TOG Attack ($k = 1.00$) |
|-----------|--------------------------|--------------------------|
| 1 it      | 0.19% 6.68% 6.99% 7.10% | 3.04% 3.12% 3.71% 2.66% |
| 10 it     | 1.89% 21.72% 49.71%       | 1.61% 1.47% 1.61% 1.63% |
| 50 it     | 60.65% 6.40% 2.27%        | 80.37% 81.92% 86.92% 83.50% |
| 100 it    | 4.68% 10.47% 18.45%       | 4.30% 5.37% 2.37% 2.18%  |
| 1 it      | 32.29% 74.44% 87.96%      | 32.29% 74.44% 87.96% 52.57% |
| 10 it     | 1.66% 38.44% 59.73%       | 1.66% 38.44% 59.73% 52.57% |
| 50 it     | 3.00% 41.05% 52.57%       | 3.00% 41.05% 52.57% 52.57% |
| 100 it    | 6.25% 27.53% 30.61%       | 6.25% 27.53% 30.61% 30.61% |

(a) The mean confidence variation.

| Iteration | PCB Attack ($k = 0.98$) | TOG Attack ($k = 1.00$) |
|-----------|--------------------------|--------------------------|
| 1 it      | 0.19% 0.11% 0.09%        | 0.08% 36.43% 43.23% 32.32% |
| 10 it     | 60.65% 6.40% 2.27%       | 80.37% 81.92% 86.92% 83.50% |
| 50 it     | 63.88% 24.64% 14.34%     | 87.34% 86.02% 85.99% 87.41% |
| 100 it    | 2.60% 0.54% 0.44%       | 68.17% 66.50% 54.42% 64.89% |
| 1 it      | 51.92% 1.30% 0.47%      | 81.33% 76.89% 85.33% 76.00% |
| 10 it     | 24.90% 4.30% 7.37%      | 95.29% 96.59% 96.79% 96.61% |
| 50 it     | 32.55% 1.87% 1.38%      | 75.14% 73.78% 74.89% 74.89% |
| 100 it    | 8.84% 0.76% 1.04%       | 95.29% 96.59% 96.79% 96.79% |

(b) The average number of boxes.

(c) The relative box variation (the percentage of consistent bounding boxes).

Note: The evaluation results on the training set are in bold.

VI. CONCLUSION

This article presents a novel hardware attack that reveals a previously unknown vulnerability of deep learning object detection systems, posing serious threats to safety-critical applications. Unlike existing attack frameworks, our approach does not rely on any assumptions about access to the object detection system, but rather leverages perturbations injected at the hardware level. Our experiments on the VOC2012 dataset, the CARLA dataset, and the YOLO detection model demonstrate the high efficiency of our attack. Further research may explore the extension of the attack to other tasks beyond object detection, or to other sensors, such as Lidar.

REFERENCES

[1] B. Biggio et al., “Evasion attacks against machine learning at test time,” in Proc. Mach. Learn. Knowl. Discovery Databases (ECML, PKDD), Prague, Czech Republic, Berlin, Heidelberg: Springer, 2013, pp. 387–402.
[2] C. Szegedy et al., “Intriguing properties of neural networks,” 2013, arXiv:1312.6199.
[3] H. Wu, S. Yunas, S. Rowlands, W. Ruan, and J. Wahlström, “Adversarial detection: Attacking object detection in real time,” in Proc. IEEE Intell. Veh. Symp. (IV), 2023, pp. 1–7.
[4] J. Lu, H. Sibai, E. Fabry, and D. A. Forsyth, “No need to worry about adversarial examples in object detection in autonomous vehicles,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2017.
[5] H. Wu, S. Yunas, S. Rowlands, W. Ruan, and J. Wahlström, “Adversarial driving: Attacking end-to-end autonomous driving,” in Proc. IEEE Intell. Veh. Symp. (IV), 2023, pp. 1–7.
[6] M. Lee and Z. Koller, “On physical adversarial patches for object detection,” 2019, arXiv:1906.11897.
[7] K. Xu et al., “Adversarial t-shirt! evading person detectors in a physical world,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2020, pp. 665–681.
[8] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, “Object detection with deep learning: A review,” IEEE Trans. Neural Netw. Learn. Syst., vol. 30, no. 11, pp. 3212–3232, Nov. 2019.
[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 779–788.
