Physical Adversarial Attack on a Robotic Arm

Yifan Jia[1,3], Christopher M. Poskitt[2], Jun Sun[2],and Sudipta Chattopadhyay[1]

***Abstract*—Collaborative Robots (cobots) are regarded as highly safety-critical cyber-physical systems (CPSs) owing to their close physical interactions with humans. In settings such as smart factories, they are frequently augmented with AI. For example, in order to move materials, cobots utilize object detectors based on deep learning models. Deep learning, however, has been demonstrated as vulnerable to adversarial attacks: a minor change (noise) to benign input can fool the underlying neural networks and lead to a different result. While existing works have explored such attacks in the context of picture/object classification, less attention has been given to attacking neural networks used for identifying object locations, and demonstrating that this can actually lead to a physical attack in a real CPS. In this paper, we propose a method to generate adversarial patches for the object detectors of CPSs, in order to mis-calibrate them and cause potentially dangerous physical effects. In particular, we evaluate our method on an industrial robotic arm for card gripping, demonstrating that it can be misled into clipping the operator's hand instead of the card. To our knowledge, this is the first work to attack object locations and lead to an incident on human users by an actual system.**

***Index Terms*—Cyber-physical systems; YOLO; object detection; adversarial attack patch; physical attacks**

I. Introduction

CYBER-PHYSICAL Systems (CPSs), in which software components are deeply intertwined with physical processes, are commonly used to automate industrial processes. Collaborative Robots (Cobots) belong to a subclass of CPSs that work closely with humans for direct human-robot interaction within a shared space. Cobots have been widely deployed in the manufacturing industry, spurred by the adoption of smart factories [1], [2]. As cobots are designed to be used in close collaborations with humans, they require higher safety standards and lower fault tolerance than other kinds of systems [3].

In order to make cobots smarter and more effective at performing complex and delicate tasks, there is a trend in industry to augment them with deep learning algorithms. In particular, Convolutional Neural Networks (CNNs) can be used to add vision capabilities to the cobots in order to implement tasks that require collaboration with humans [4]. The underlying layers of CNNs, however, are challenging for humans to understand, making it difficult to ascertain that the cobot will behave correctly and *safely* in all scenarios, especially those for which humans are also involved.

Deep learning systems for computer vision are susceptible to manipulations by attackers [5], [6]. A popular type of attack is generating *adversarial examples*, in which pixel-level noise is added to images in order to cause the CNN to misclassify the input [7], [8]. Recently, researchers have begun to investigate whether such attacks are also physically viable in the real-world [9], especially when CNNs become embedded within highly safety critical systems such as autonomous vehicles. A typical real-world attack is to generate a physical (e.g. printed) *adversarial patch* that can cause the CNN to miss or incorrectly classify an object [10]–[13]. However, most existing works focus on classification, and due to the difficulty of optimizing non-differential process (e.g. process with judgement), few works analyse location detection attacks. Moreover, the possibility of causing the system to make a wrong final decision based on that location information has been less explored. Not only can a robotic arm cause damage to its surroundings, but it could endanger human users too.

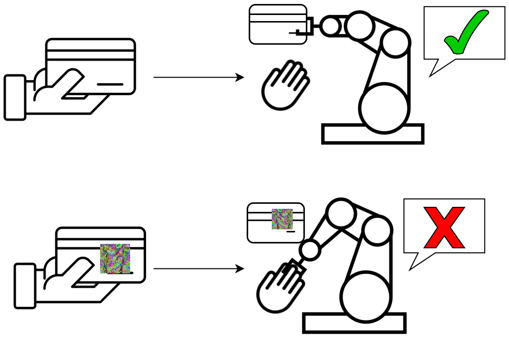

Fig. 1. A high-level overview of our attack scenario: an adversarial patch misleads the object detector, and the robotic arm clips the human's hand

Applying attacks on a real world system has several challenges. First, digital adversarial samples often do not work reliably in the physical world. This is because these adversarial samples are subject to other perturbations such as lighting and angles. Furthermore, camera images in cobots are often validated in different ways, e.g. by contrasting images taken at consecutive time points. Second, real-world CPSs are complicated systems which have many different components: compromising only one of them may not be sufficient to compromise the whole system. Though some real world adversarial attacks have been demonstrated on camera-based object detection systems (e.g. [10], [11], [14]), these typically do not go beyond misclassification, and do not explore whether the attacks can *really* lead to a potentially dangerous physical effect.

In this paper, we investigate the susceptibility of CPSs equipped with CNN-based object detectors to physical adversarial attacks on object location. In particular, we propose a method to generate adversarial patches to cause a failure of *location mispredictions* for a real world system. Our novel contributions include:

1.Singapore University of Technology and Design.
2.Singapore Management University.
3.TUV SUD Asia Pacific Pte. Ltd. Digital Service Center of Excellence

- We establish an automatic card detection and pick up system by implementing object detection mechanism on a robotic arm;
- We propose a method to generate physical adversarial patches to achieve untargeted location attacks on top of obscuring the object location;
- We improve the method to achieve targeted location attacks to make the model detect targeted locations while ensuring an identical classification;
- By optimizing the training process, we ensure that the patches are workable in the real-word in order to deceive the final decisions of a complete system.
- We demonstrate the effectiveness of the attack by implementing it on an industrial robotic arm for card gripping, and showing that it can be misled into clipping the human operator's hand.

A high-level overview of our attack scenario is given in Figure 1.

The paper is organized into the following sections. In section II *(Background and Related Work)*, we provide an overview of CPSs, cobots, object detection systems, and adversarial attacks. In Section III *(The Card Picker System)*, we introduce how we retrofit a traditional robotic arm to a automatic object detection and pick up system. In Section IV *(Generating Adversarial Patches)*, we describe our threat model and our method of generating adversarial patches for CPS object detectors, we explain how we overcome the difficulty of location attacks as well. In Section V *(Evaluation)*, we experimentally assess the effectiveness of our adversarial attacks both digitally and on a real-world robotic arm. We wrap up in Section VI *(Conclusion)*.

II. Background and Related Work

In this section, we state our assumptions about the structure of CPSs, and introduce the real-world cobot used to evaluate our work. Following this, we introduce object detection algorithm that the cobot implements, and discuss some background on adversarial attacks.

A. CPS and Cobots

CPSs consist of two interconnected parts. A 'physical' part includes sensors and actuators. A 'cyber' part, consisting of software components such as Programmable Logic Controllers (PLCs) [15] that implement certain control logic. We assume that the sensors and PLCs are connected over a network, with the PLCs taking the sensor readings as input and sending commands to the actuators. Furthermore, we assume the presence of a Supervisory Control and Data Acquisition (SCADA) system which are connected to the PLCs that can supervise the control process and issue commands independently.

Collaborate robots (cobots) have the characteristics of CPSs. We evaluate our method on a robotic arm cobot, the *Universal Robot UR10e* (*UR10e*). The *UR10e* is a versatile collaborative industrial robot and can be implemented with different handlers and programs for different tasks such as assembly, quality inspection, material handling, and dispensing. Such robot arms are used in many industries and some of which are safety critical. In order to support object gripping, a 'handling gripper' has been mounted to the robotic arm and a tool camera (*Intel RealSense D435*) was attached to the top of the gripper to detect objects.

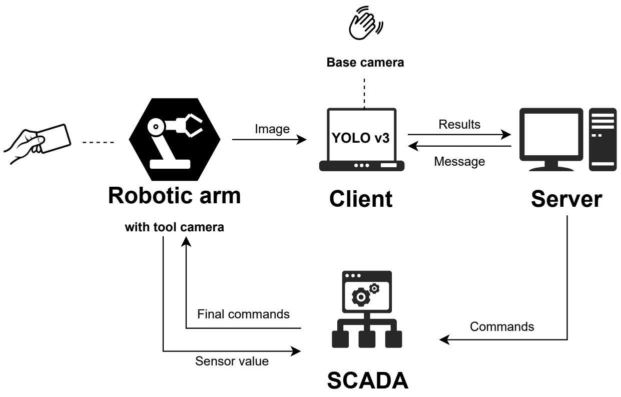

Fig. 2. Networks of the cobot system

In our scenario, the system is used as a card picker. A working cycle is realized as follows: 1) The robot is initially set at a default position, and the system initiates once it detects a human waving a hand. 2) Whenever a card is detected by the tool camera, images are retrieved by the client for processing and analysis. 3) Afterwards, the results are sent to the server to generate commands for the SCADA of the robotic arm for further processing. 4) According to some pre-defined logic, the SCADA will process the commands with sensor values from the robotic arm and send the final commands to execute. Figure 2 shows the networks and the working cycles of the system.

B. Object Detection

The pictures are analyzed by an object detection algorithm. In this case, the server implements YOLO v3 [16], one of the fastest object detection algorithms, and one that is widely used for real time detection. In particular, the latest version—YOLO v3—conducts detection detection at three different scales, and thus is more robust.

The network directly predicts bounding boxes with object location, object score, and class score in a single pass. YOLO v3 includes a fully connected CNN which is our attack target. Input to the network is a batch of images, for example, with shape (m, 416, 416, 3), where m is the number of images. The images are divided into multiple cells with strides of 32, 16 and 8 to detect small, medium and large objects. The output is a list of bounding boxes along with the recognized classes for each cell, and each bounding box is composed of 6 numbers ($b_x, b_y, b_h, b_w, obj, cls$), where cls is class score, representing a multidimensional vector that is related to classification. In our case, as we are only interested in the class of "card", cls is a number rather than a vector. A higher cls indicates a higher possibility that the object is a card. b_x, b_y, b_h, b_w are the x, y coordinates and height (b_h), weight (b_w) respectively for each cell. Finally, obj is an object score indicating the possibility that an object exists. A higher score indicates a higher possibility that an object exists. Since there are many bounding boxes predicted for an image with size 416*416, the bounding boxes are then filtered by thresholds of object score and class score and those with maximum intersection over union (IoU) are chosen as the final object box.

Given the features of YOLO v3 (i.e. 3 scale predictions), an attack on a single image is hard to apply, although not impossible [11]. Even more challenging, in our case, the algorithm is implemented using a video: the output updates fast and continuously, which requires a stable and robust attack to hack the system. (We discuss the detailed requirements in Section IV-B.)

C. Adversarial Attacks

Neural networks have been demonstrated to be vulnerable to adversarial samples. In computer vision, adversarial samples are small perturbations designed to deceive the neural networks while remaining undetectable by humans [17]. Adversarial attacks can be categorized as untargeted or targeted, i.e. attempting to cause some misclassification or a particular misclassification respectively.

Adversarial attacks were first mentioned by Biggio et al. [18] in 2013. Given the broadening applications of deep learning, much more successful real-world attacks [6], [19], [20] continue to arise and has been widely discussed. However, to achieve practicality in physical attacks, adversarial patches must be more practical than adversarial noises. Adversarial noises require the attackers to access the intranet and modify the original images. In contrast, adversarial patches can be easily printed out and pasted into the practical environment without accessing the intranet and the target system.

In our work, we consider adversarial patches, which are continuous blocks of image pixels. An adversarial patch could replace a part of an image in order to given harass to the model. For such attacks, the perturbation is additionally constrained by the size of the patch itself.

Adversarial patch attacks can target both classification and object detection. Brown et al. [21] proposed to use adversarial patches to deceive classification models. The loss of the target class is calculated first before using the reverse values as loss to update the adversarial patch by back propagation. The work of Karmon et al. [22] is similar: they added more parameters to the loss function in order to attack the model to give results not only near the target class but also far from the original class. Liu et al. [12] proposed a *DPatch* that can perform targeted and untargeted attacks on object detectors (Fast R-CNN and YOLO). They demonstrated transferability among different detectors as well as datasets. Zhou et al. [23] proposed an attack on automobile vehicles to make the object detector predict wrong angles of direction by adding an adversarial billboard, but only in a simulated environment.

In the above examples, the loss function is designed to consider only the maximum value of object/class scores, or alternatively, create a target output to generate loss, which is complicated and inaccurate for some images. In our work, we take *all* related values of objects to calculate the loss function, making the attack more robust and effective. Additionally, we print the patches out and test them physically on an actual system to prove the feasibility of attacks in the real world.

To achieve the same results in real-world scenarios, an attack must be stable and robust. Tepan, Komkov, and Petiushko [14] proposed the idea of attacking a face detector to decrease the confidence of the detection algorithm by minimizing the similarity of the original image and attacked image. They improved the training process by considering the real-world environment, such as angles and contract. Lee and Kolter [11] did some similar work with *DPatch*. They increased the confidence of patches and demonstrate the attacks in the real-world, deceiving the model that detects the patch. Thys et al. [13] proposed a method to generate patches to hide a person from a person detector. They optimized the patches by using rotation, scaling, and adjusting brightness, proving that the method works well in a real-world scenario.

Although these works were tested with physical attacks for object detectors, none of the them have been tested on a complete system, which comes with additional challenges. In contrast, our work is able to overcome the difficulties of dealing with a complete cyber-physical system, demonstrating effective attacks both in the digital and physical domains. In particular, we are able to overcome the difficulties posed by multi-stage processing pipelines for object detectors, which are usually successful at filtering out or detecting other kinds of attacks.

We remark that adversarial attacks against object detectors are more complicated than those against classifiers. For image classifiers, the output is a vector of class probability, and thus loss can be easily calculated as the difference between the output vector and desired class vector. For object detection, however, the output has a higher dimension and includes more information, such as raw data of object location and object confidence for each cell, which means that we cannot calculate the loss directly.

III. Our Scenario: The Card Picker System

This section introduces our industrial case study: a card picker system implemented using a Universal Robot UR10e robotic arm. We motivate this choice of case study, before discussing the hardware, networking, and algorithmic aspects of the system.

A. Motivation

Universal Robots is currently the market leader in cobots. The UR10e is a versatile robotic arm with a 3-Position teach pendant and which can be programmed for different tasks such as assembly, quality inspection, material handling, and dispensing. Cobots with object detection can greatly expand these working scenarios in manufacturing. However, existing model-based object detection methods are not intelligent, and the cobot cannot adjust quickly according to the environment.

In contrast, AI-based object detection methods can make the system more adaptive by playing the role of 'eyes'. For example, the robotic arm can follow a detected object as that object is moving. Such AI-based object detection is typically implemented using deep neural networks (DNN), which can make the system quickly and accurately react to complicated tasks. There are several existing industrial cobots with such visual functions [24]–[26], and there is a trend in academic and industrial research to explore how to integrate intelligent DNN-based object detection algorithms in cobots [27]–[29]. This is not a straightforward task in general: DNNs are powerful but vulnerable, and may be easier to attack than the traditional rule-based algorithms.

In this work, we will develop and evaluate an attack on a UR10e robotic arm equipped with an AI-based object detector that was developed as a case study by an industrial company [30]. The algorithm was presented at the Industrial Transformation Asia Pacific (ITAP) 2019 conference to show the future direction of AI in industry. We demonstrate that our attack is effective for such systems and thus could potentially be performed in the wild.

B. System Design

The robotic arm is retrofitted into an automatic object detection and pick-up system. The system is designed to replace humans to pick up and classify objects intelligently. The system can detect an object, follow its movement, and place the object in a designated place according to the object classification. In this work, the scenario involves the detection of a *card*, followed by the robotic arm picking it up.

Two cameras (Intel Realsense) are physically mounted on the robotic arm. The robotic arm has six rotating joints degree of freedom with a gripper at the end to hold objects. To realize the scenario, one base camera to detect human hand movement, and the camera is placed near the system. A tool camera is mounted on the top of the gripper. The tool camera has a fixed distance and angle with respect to the gripper, and follows the grippers as it moves.

The network of the system is shown in Figure 2: the robotic arm and its SCADA (i.e. the robotic arm system) is considered a complete cyber-physical system. The base camera and the tool camera are connected with the client. The robotic arm system, the client, and the server are connected with an Ethernet switch. The base camera is monitoring human hand movement, and sending an initiation message to the server once the movement is captured. The server keeps listening to the client and runs the tool camera and object detection algorithm after receiving the initiation message. The client then analyses the image and sends the card center coordinates to the server. The server monitors the coordinates and sends either the command to: 1) follow the card with the robotic arm system if the coordinates are changing; or 2) close the gripper if the coordinates are stable for 2 seconds.

The object detector (i.e. object detection algorithm with YOLO v3) is implemented in the client to analyse images. The algorithm is trained for card detection. Whenever the algorithm is activated, the video of card images is analysed by the object detector and the results of card coordinates are repeatedly sent out to the server. The details of the object detector is in Section II-B.

When the system starts, the server first checks if the base camera, tool camera, and the robot are connected. Then, the server waits for the initiation message from the base camera. As we are only interested in the card detection part, in the rest of the work, we only analyse the tool camera and card images without needing to focus too much on the base camera.

Our system is retrofitted on a widely used industrial robotic arm model. Thus there are already some protection mechanisms implemented. For example, there are 14 safety functions, including emergency stop, safeguard stop, joint position, joint speed limit, power and force limit. Such mechanisms can protect humans from severe harm and allow us to test the attack experiments with more practical scenarios (e.g., with humans involved). However, while such safety functions help avoid collisions or accidents, they are insufficient for deliberate attacks.

IV. Generating Adversarial Patches

In this section, we introduce the threat model and our methodology to generate adversarial patches. In addition, we provide an optimization method to ensure an effective physical attack.

A. Threat Model

In a cobot system, the client and server parts (Figure 2) are typically separated from the well-protected robot itself (e.g. the robotic arm and the SCADA in our case) for compositionality, thus these parts are easier to access from the cyber perspective. According to the safety requirements for industrial robots (EN ISO 10218), human operators are constrained in how much they can modify the physical part and integrated control system of robots (SCADA), thus our threat model assumes the attacker can only access the client and server parts of the whole system with the following constraints:

1) The attacker is able to access information about the object detector structure and parameters from the client;
2) The attacker is able to access information about the data processing and commands generation from the server;
3) The attacker has access to the detected object (i.e. cards) to make modifications;
4) The attacker cannot change or modify any program of the system. That is, we assume that code modification attacks are dealt with through integrity checks [31], [32].

To evaluate the safety and security of cobots under the threat model, we set up the previously described scenario in which our industrial robotic arm helps humans by picking up cards presented by visitors and placing them in a given location. Our attack objective is to deceive the object detector and make the robotic arm physically hit the human's hand (Figure 1). Note that this is set up as a demonstration of physical damages that could potentially be caused by such cobots in manufacturing facilities.

B. Methodology

We propose an attack method to generate a sustained adversarial patch for attacking the system, i.e. causing the object detector to detect our targeted object (a human hand) as a card inside the video feed and sent targeted object location as final commands to the robotic arm.

1) Adversarial Patch for Object Detector: First, we need to address the challenge that the adversarial patch should be possible to apply the attack in the real world (not just a digital simulation). It is impractical to apply noises at the pixel level on the full image in the real world, and thus we design a method to generate adversarial patches which can be physically attached to the card to attack the system.

Second, we must consider that the attack needs to deceive the object detector into detecting the targeted object instead of cards. The difficulty for this point is that determining the location of an object is not a continuous function and thus not differentiable. To address this challenge, we use a *rectified linear activation function (ReLU) function* to filter scores to calculate the gradients and optimize the attack.

YOLO v3 has a CNN structure with bounding boxes as the output. The bounding boxes provide the information of each cell (i.e., b_x, b_y, b_h, b_w), as well as the object confidence (if an object exists) and object class (the probability of the object class). The goal of the targeted attacks is to change the object classification to a targeted classification. Even though we can achieve targeted classification attacks by manipulating the classification score [12], the location of the targeted class object is uncontrollable. Therefore, targeted location attacks are much more difficult because we need the object detector to detect the targeted location object with the same classification regardless of the actual object.

To achieve the targeted location attacks, we design the training process as follows. We choose the sum of object scores and class scores with classification as "card" to be our loss function L_{card}, and the sum of object score with classification as "non-card" to be L_{hand}. The untargeted and targeted attacks with adversarial patch p can be defined as follows:

Untargeted attack:

$$f(\mathbf{x}+p) \neq f(\mathbf{x}) \quad \forall \; loc(p) \tag{1}$$

Targeted attack:

$$f(\mathbf{x}+p) = y_t \quad \forall \; loc(p) \tag{2}$$

where y_t is the targeted output and $loc(p)$ is the location of p. As the classification is a binary problem, we can also either focus on the object score or class score separately. If we focus on object score, we attack the model to detect the "card" part as no object exists and "non-card" part as an object exists. On the contrary, if we focus on class score, we aim to attack the model to classify "card" as "non-card", and "non-card" object as "card".

2) Adversarial Patch for the Robot: To apply the attack in the real-world, the attack should be able to change the robotic arm's behavior as desired. Thus, we need to consider the behavior of the system. One problem is the system will filter out one-shot attacks as bias; another is that the client will send the results to the server only if the results (i.e. location coordinates) remained the same for more than 2 seconds. For the first problem, from the information of the server, the system will ignore the results if the detection confidence is lower than a threshold. We thus add the threshold to the training process to address the former problem. For the second problem, we designed the attacks by fixing the targeted location in order to make sure the results are always the same. The loss functions will thus be improved as follows.

Loss function (card). We use L_{card} to represent the loss function attributed to the detection of a hand object. L_{card} consists of two parts. Firstly, we need to find where the card is. For each image output, the card part should have both object score and class score II-B higher than a certain threshold. We keep the card part only. Existing approaches [13], [33] select the maximum object/class score as the loss function. Nevertheless, to address the challenge of that determining object's location is not differentiable, we consider all scores above the threshold that make the loss function more effective. We use a rectified linear activation function (ReLU) function to filter the scores in order to calculate the gradients and back propagation. Secondly, we sum up the object scores and class score within the card part for all three scales to handle various object size problems. Thus the equation is designed as follows, where L_{obj} represents the object scores of the card in the image, and L_{cls} represents the classification score of a card. The goal of our training is to minimize the value of loss function for card detection, thus:

$$\begin{aligned} &Loc_{card} = ReLU(\sum_{i=1}^{n}(obj_i) - \alpha) * ReLU(\sum_{i=1}^{n}(cls_i) - \beta) \\ &L_{card} = sum(Loc_{card} * (L_{obj} + L_{cls})) \end{aligned} \tag{3}$$

where Loc_{card} is the function to determine the location of the card and n represents the number of cells. To calculate the location of a card, both obj and cls need to be *larger* than the thresholds. We use the ReLU function to help us filter the area with obj and cls lower than the object threshold α and class threshold β. The loss of card is the sum of $L_{obj} = sum(obj_1, obj_2, ...obj_n)$ and $L_{cls} = sum(cls_1, cls_2, ...cls_n)$.

Loss function (hand). We use L_{hand} to represent the loss function attributed to the detection of hand object. L_{hand} consists of two parts. Firstly the hand position and then the sum of object and class scores. L_{obj} represents the sum of object scores which needs to be larger than the object threshold α, and L_{cls} represents the classification score which needs to be *smaller* than the class threshold β. The goal of our training is to maximize the scores of hand object, thus:

$$\begin{aligned} &Loc_{hand} = ReLU(\sum_{i=1}^{n} obj_i - \alpha) * ReLU(\beta - \sum_{i=1}^{n} cls_i) \\ &L_{hand} = \frac{1}{sum(Loc_{hand} * (L_{obj} + L_{cls})}) \end{aligned} \tag{4}$$

The two parts compose the total loss function of our optimization process of $\min \sum_{i=1}^{k}(L_{card_i} + L_{hand_i})$, where k is the number of images for each training batch.

3) Physical Adversarial Patch: Finally, we need solve the challenge that the physical attack should be able to achieve the same results in terms of deceiving the object detector as the digital attack (e.g. applying the digital patch to the image through the intranet). There are several sub-problems to consider. First, the attack should be effective in different scenarios with various card colors, contents, and contexts. Second, the patch should be effective regardless of the card position since the card keeps moving in practice. Finally, the patch need to be printable, and attacks should have the same effects as the digital ones, that is, the adversarial attack must be robust. This is challenging given that adversarial samples are often close to the decision boundary.

To solve the above problems, we adopt the following optimization process. To address the first problem, we prepared 50 images with different cards, angles, distances, contexts, and held by different people as training data, and another 30 images with the same conditions as testing data. We add the initial patch to all images and feed them into the object detector at the same time to make sure the patch is trained with features of all images. The output determines where the card is (Loc_{card}) and where is the hand is (Loc_{hand}). We measure the scores and calculate the loss function. Afterwards, we apply the back propagation algorithm to update the patch along the calculated gradients. Subsequently, the updated patch is adopted for the second round. The training stops when the loss is smaller than a specific threshold or after running a certain number of iterations. With regards to the second problem, the patch is pasted onto images with random locations but with a higher probability to be pasted in the $1/3$ central area to imitate the actual scenarios. To solve the last problem, we scan the printed patches and compare them with digital patches to calculate the bias δ. The bias is considered before the well-trained patch is printed. In addition, to simulate the physical environment, we scale the patch randomly up and down within 20% of the original size.

To avoid being regarded as noise, the attack must keep a high success rate, we thus employs the context of the real-world system. We prepare more training data with similar backgrounds to the actual one to increase the success rate. Thus we can focus on collect training images with different color and contents of cards and varieties of hand positions.

The above allows us to significantly reduce the training data size so as to increase the efficiency of training a patch. For example, to the attacker, a 10 second video could provide enough training data to generate an adversarial patch. After testing, we found that 50 well-prepared images are enough to train an effective adversarial patch. We tested the patch on 30 images, and the results have proved the patch is effective for the scenario of a card pick-up robotic arm. The experiment details and results are discussed in our evaluation (Section V).

V. Evaluation

We evaluate our methodology by answering the following two research questions (RQ):

1) RQ1: Are we able to design a digital adversarial patch to attack an object detector?
2) RQ2: Are we able to make the adversarial patch physically attack a robotic arm?

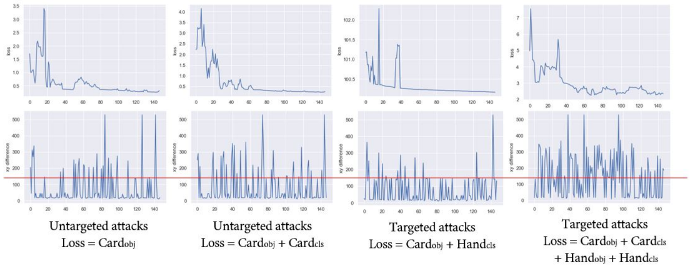

Fig. 3. An overview of loss (above) vs location difference (below) i.e. xy difference with different loss functions.

A. Results with Digital Images

We answer RQ1 by showing the results for digital images. Unlike attacks on classifiers, we do not focus on whether the object is detected correctly and accurately. Instead, we are more interested in the object location. Thus, for a single image, we use loss and the coordinates (x, y) of detected card center as criteria to determine if the attack is successful. For a video or batch of images, we define a *success rate* (percentage of successful attacks over all attacks) to measure the the number of attacks that *achieve our goals*, and we define a *fooling rate* (percentage of attacks fooled the detector over all attacks) to measure if the attacks fooled the detector to *predict a wrong location*.

We train the patch with two kinds of loss. For the first scenario, we use object score of card and class score of hand to calculate loss, as we want the card part has a lower object score and hand area has a higher class score. In this case, the loss is $L_{card} = sum(Loc_{card} * L_{obj})$ and $L_{hand} = 1/sum(Loc_{hand} * L_{cls})$. For the second scenario, the loss is the same as equation 3 and 4, where we optimize both object and class score for card and hand.

Figure 3 shows an overview of loss and card center difference trends when train the patch on a single image. The graphs on the top row are four kinds of loss, whereas the graphs on the bottom row are the l_2 distance between initial center coordinates (x,y) and coordinates (x',y') with patch. The red line indicates the value between hand and card: we manually checked the images, and if the values of "xy difference" is near the red line, we regard the center as at the hand.

From the figure, we can see the loss is decreasing with the increase of iterations. Moreover, the center distance may be different even if the loss is low. Thus the patch position plays an important role during the attack. While we want the patch to achieve an attack effects regardless the patch position, the results suggest that the position always influence the results. This could due to the fact that YOLO v3 will check the IoU (see details in Section II-B), which is related to cell positions, and patch position may affect the check results as it is part of the image. For example, the algorithm may detect the patch as a small card.

Another discovery is that for untargeted attacks, the performance is better if we include the card class. As for targeted attacks, the patch can achieve our attacks most of the time when we only consider card object score and hand class score. However, if we consider both object and class score for hand

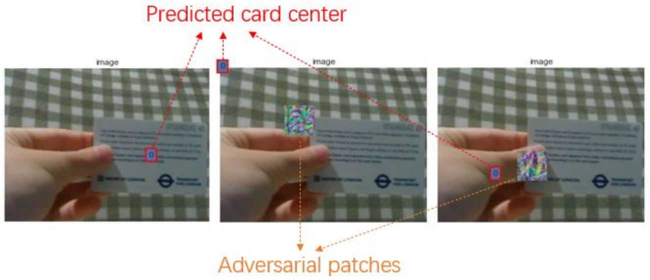

Fig. 4. An example of digital patch results. (*Left:* no attacks; *Middle:* untargeted adversarial patch; *Right:* targeted adversarial patch

and card, the attack is the strongest (i.e. does not detect the card any more) but will not always attack the target.

Figure 4 shows an example when we train the digital patch. The red and blue square indicate the predicted center of card. The left image shows the original output, the middle one shows the untargeted attack, and the right image shows the targeted attack output.

We evaluate the adversarial patch on 30 test images, and calculate the coordinates' l_2 distance of before after the attack to measure if the attack is a success. If the distance is positive, which means the patch affect the model output, we take it as a successful attack.

Input images are with size 2000*1500 pixels and the unit is number of pixels.

- For untargeted attacks, the patch has a success rate of 100% and fooling rate 100%. The minimum change is 0.5 and average distance is 445.21 pixels. Values larger than 1000 pixels usually indicate the detector cannot determine any object and thus returns a location at (0,0).
- For targeted attacks, the patch has a success rate of 23.33% and fooling rate of 100% to predict the targeted object as a card. However, for unsuccessful attacks with the patch, the detector still mis-predicts the center far away from card with a minimum value of 0.5 and average value of 415.02 pixels.

We found for some images, the hand cannot be detected as an object, and the object score of hand cannot be increased in that case. This could because the lighting or background color is similar to hands. Thus targeted attacks are harder than untargeted attack in the real-world.

B. *Results with Real-World Robotic Arm*

We present the results of attacking the robotic arm to answer RQ2. For simplicity, we only consider the part of the system that we are interested in, so we don't discuss much on the base camera and how the system is initiated. Therefore, after the tool camera is initiated, it keeps capturing images and sends them back to the server. Inside the server there are two main programs to analyze the images. The images go through the first program—the YOLO v3 object detector. Afterwards, the object detector will send the detected card center 2D[1] coordinates to the second program which includes a state machine to analyze and make decisions. The second program will audit the state of the system and compare current coordinates and the previous ones to deduce if the card is moving. If the card is still moving, the robotic arm will receive the commands to follow the card, and if the card is detected to be stationary for a period of 2 seconds, the arm will stop moving and the gripper is asked to clip. Moreover, once the robot finds the card is ready to be clipped, the object detector will stop working.

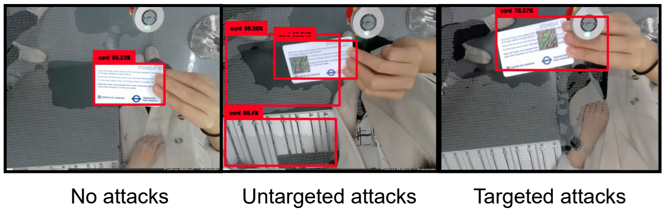

Fig. 5. Screen shot of no attacks, untargeted attack and targeted attack.

There are other processes after clipping the card such as placing the card in a designated location to scan. As we do not attempt to attack these processes, they are ignored in our study. We assume that once the robot hits a human's hand during the card clipping stage, it is considered a successful attack.

We evaluate our patches on the *UR10e* cobot. For each patch, we recorded videos and monitored the final results to deduce if the gripper had hit a human's hand. At the same time, we also considered the object detector performance by checking each frame. The attacking videos can be found in the supplementary material[2]

Figure 5 shows a screenshot of robot arm camera records: we can see that if there is no patch, the card is detected with a confidence of 99.23%, whereas when we put an untargeted patch on the card, the camera will give many inaccurate predictions. For targeted attacks, the screenshot shows the camera detection when the hand is clipped. More images of untargeted attacks can be found from the link [3]. The setup and the moment when the gripper clip the card can be found at the link [4].

In our experiments, we performed a total of 40 attacks: 20 times untargeted, and 20 times targeted. For untargeted attacks, the success rate is 90%, while targeted attacks have a success rate of 25%. For the targeted attacks, though it cannot make the gripper hit the human's hand every time, it still can make the gripper clip other location rather than the card with a fooling rate of 100%. In total, we have designed the patch with a success rate of 57.5% and a fooling rate of 95% to attack the real world robotic arm. Though the success rate might seem 'low', from a safety-critical perspective it's actually very high. Especially if it translates to other cobots in factory settings.

One discovery is that when there is a patch, even if the the detector can draw the correct box for an object, the gripper still cannot clip the card. This could be because the attack

[1] The camera is 3D camera, but we only interested in card 2D position here, thus we ignore the depth information

[2] Click to get the videos

[3] Untargeted attacks: https://s2.loli.net/2022/05/18/t8QnjWyhqDa7Jbm.png

[4] Robotic arm: https://s2.loli.net/2022/05/18/y2INEWge4TqJhXY.jpg

patch makes the model prediction very unstable, and there is a delay in the communication between the camera and the gripper. Thus the predicted center of the card is keep changing, and the information sent to the gripper is not stable enough to generate commands.

VI. Conclusion and Future Work

In this work, we presented a method to generate adversarial patches that can be printed and used in the real-world to modify a cobot behavior as the attack designed. Unlike previous adversarial patches that attack on object detectors, our approach is able to deceive a complete system to make the system do some desired physical action. We trained the patch by considering the logic of a real-world system and increased the detection confidence of a human hand. We then optimized the images to make them printable and workable in the real-world. Finally, we achieved both untargeted attacks and targeted location attacks on an industrial robotic arm which to our knowledge is the first adversarial patch to successfully attack a real-world industrial system.

There are some challenges to overcome in future work. One challenge is that the attack may not perform as well in the presence of other defense mechanisms, such as a hand detection filter. However, such a defense is only helpful in scenarios specifically working with human hands. In general, deliberate adversarial attacks can be mitigated with a more robust model. However, unlike accuracy, model robustness is seldom considered when AI techniques are adopted for industrial systems. Finally, we train the adversarial patch as an attack without any camouflage, which means the patch looks unusual to human eyes and is easy to notice. In such situations, hiding the adversarial patch from the human is also a challenge. In the future, our adversarial patch training process can be optimized with specific shapes and colors as constraints.

References

[1] F. Sherwani, M. M. Asad, and B. Ibrahim, "Collaborative robots and industrial revolution 4.0 (ir 4.0)," in *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*. IEEE, 2020, pp. 1–5.

[2] L. Tamas and M. Murar, "Smart cps: vertical integration overview and user story with a cobot," *International Journal of Computer Integrated Manufacturing*, vol. 32, no. 4-5, pp. 504–521, 2019.

[3] M. Gleirscher, N. Johnson, P. Karachristou, R. Calinescu, J. Law, and J. Clark, "Challenges in the safety-security co-assurance of collaborative industrial robots," *CoRR*, vol. abs/2007.11099, 2020. [Online]. Available: https://arxiv.org/abs/2007.11099

[4] O. D. M. Lázaro, W. M. Mohammed, B. R. Ferrer, R. Bejarano, and J. L. M. Lastra, "An approach for adapting a cobot workstation to human operator within a deep learning camera," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1. IEEE, 2019, pp. 789–794.

[5] S. Qiu, Q. Liu, S. Zhou, and C. Wu, "Review of artificial intelligence adversarial attack and defense technologies," *Applied Sciences*, vol. 9, no. 5, p. 909, 2019.

[6] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *Ieee Access*, vol. 6, pp. 14 410–14 430, 2018.

[7] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A python toolbox to benchmark the robustness of machine learning models," *arXiv preprint arXiv:1707.04131*, 2017.

[8] D. Goodman, H. Xin, W. Yang, W. Yuesheng, X. Junfeng, and Z. Huan, "Advbox: a toolbox to generate adversarial examples that fool neural networks," *arXiv preprint arXiv:2001.05574*, 2020.

[9] S. T. Jan, J. Messou, Y.-C. Lin, J.-B. Huang, and G. Wang, "Connecting the digital and physical world: Improving the robustness of adversarial attacks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 962–969.

[10] Z. Wu, S.-N. Lim, L. S. Davis, and T. Goldstein, "Making an invisibility cloak: Real world adversarial attacks on object detectors," in *European Conference on Computer Vision*. Springer, 2020, pp. 1–17.

[11] M. Lee and Z. Kolter, "On physical adversarial patches for object detection," *arXiv preprint arXiv:1906.11897*, 2019.

[12] X. Liu, H. Yang, Z. Liu, L. Song, H. Li, and Y. Chen, "Dpatch: An adversarial patch attack on object detectors," *arXiv preprint arXiv:1806.02299*, 2018.

[13] S. Thys, W. Van Ranst, and T. Goedemé, "Fooling automated surveillance cameras: adversarial patches to attack person detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[14] S. Komkov and A. Petiushko, "Advhat: Real-world adversarial attack on arcface face id system," *arXiv preprint arXiv:1908.08705*, 2019.

[15] R. Alur, *Principles of cyber-physical systems*. MIT Press, 2015.

[16] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

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

[18] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013, pp. 387–402.

[19] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," *arXiv preprint arXiv:1812.05271*, 2018.

[20] P. Saadatpanah, A. Shafahi, and T. Goldstein, "Adversarial attacks on copyright detection systems," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8307–8315.

[21] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," *arXiv preprint arXiv:1712.09665*, 2017.

[22] D. Karmon, D. Zoran, and Y. Goldberg, "Lavan: Localized and visible adversarial noise," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2507–2515.

[23] H. Zhou, W. Li, Z. Kong, J. Guo, Y. Zhang, B. Yu, L. Zhang, and C. Liu, "Deepbillboard: Systematic physical-world testing of autonomous driving systems," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 2020, pp. 347–358.

[24] "Home - SCAPE Bin-Picking for increased productivity & improved working environment," https://www.scapetechnologies.com/node/5.

[25] "speedbot, classification robots," https://www.speedbot.net/en, accessed: 2022-5-10.

[26] "Machine vision system, visual inspection systems, machine vision detection, size detection, contour," http://en.kmischina.com/, accessed: 2022-5-10.

[27] S. Kulik and A. Shtanko, "Experiments with neural net object detection system yolo on small training datasets for intelligent robotics," in *Advanced Technologies in Robotics and Intelligent Systems*. Springer, 2020, pp. 157–162.

[28] P. Sharma and D. Valles, "Deep convolutional neural network design approach for 3d object detection for robotic grasping," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2020, pp. 0311–0316.

[29] S. K. Paul, M. T. Chowdhury, M. Nicolescu, M. Nicolescu, and D. Feil-Seifer, "Object detection and pose estimation from rgb and depth data for real-time, adaptive robotic grasping," in *Advances in Computer Vision and Computational Biology*. Springer, 2021, pp. 121–142.

[30] L. L, M. S, and T. P. K, "Business card demo," https://github.com/martinatgit/BusinessCardRobot, 2020.

[31] R. Taylor, "An integrity check value algorithm for stream ciphers," in *Annual International Cryptology Conference*. Springer, 1993, pp. 40–48.

[32] Y. Chen, C. M. Poskitt, and J. Sun, "Code integrity attestation for PLCs using black box neural network predictions," in *Proc. ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2021)*. ACM, 2021, pp. 32–44.

[33] S. Hoory, T. Shapira, A. Shabtai, and Y. Elovici, "Dynamic adversarial patch for evading object detection models," *CoRR*, vol. abs/2010.13070, 2020. [Online]. Available: https://arxiv.org/abs/2010.13070