On Feasibility of Intent Obfuscating Attacks

ZhaoBin Li[1], **Patrick Shafto**[1,2]

[1]Department of Mathematics and Computer Science, Rutgers University–Newark, New Jersey, USA
[2]School of Mathematics, Institute for Advanced Study, New Jersey, USA
zhaobin.li@rutgers.edu, patrick.shafto@rutgers.edu

Abstract

Intent obfuscation is a common tactic in adversarial situations, enabling the attacker to both manipulate the target system and avoid culpability. Surprisingly, it has rarely been implemented in adversarial attacks on machine learning systems. We are the first to propose using intent obfuscation to generate adversarial examples for object detectors: by perturbing another non-overlapping object to disrupt the target object, the attacker hides their intended target. We conduct a randomized experiment on 5 prominent detectors—YOLOv3, SSD, RetinaNet, Faster R-CNN, and Cascade R-CNN—using both targeted and untargeted attacks and achieve success on all models and attacks. We analyze the success factors characterizing intent obfuscating attacks, including target object confidence and perturb object sizes. We then demonstrate that the attacker can exploit these success factors to increase success rates for all models and attacks. Finally, we discuss main takeaways and legal repercussions. If you are reading the AAAI/ACM version, please download the technical appendix on arXiv at https://arxiv.org/abs/2408.02674

1 Introduction

A malevolent agent sticks an adversarial patch to a bench on the sidewalk, causing a self-driving car to miss the stop sign and hit a crossing pedestrian. Upon interrogation, he claims no malicious intent; the patch is only an art. Because the sticker is on the bench but the effect is on the sign, authorities are unable to prove intent, preventing them from easily securing a conviction. This thought experiment highlights two serious implications of an intent obfuscating attack: it opens up new avenues for harmful exploits, and provides the culprit with "plausible deniability".

Considering the potential significance of intent obfuscating attacks, it is important for the machine learning community to understand and defend against such attacks. Intent obfuscation, though a common practice in cyberattacks for penetrating target systems (LIFARS 2020), has rarely been raised in the adversarial machine learning literature. Most research has focused on the competition between attack and defense, which involves crafting more effective adversarial examples to deceive machine learning systems and evade detection, and conversely more robust machine learning systems and more sensitive detection algorithms to mitigate attacks (Ren et al. 2020; Xu et al. 2020). Intent obfuscation complements the attack and defense literature by adding the dimension of intent to the competition: attackers can hide their purpose of attack for plausible deniability, and defenders would have a harder time proving, or even determining, the purpose of attack from the adversarial examples.

We propose intent obfuscating attacks on object detectors through a contextual attack, in which we perturb one object to target another non-overlapping object. By attacking another object, intent is obfuscated providing plausible deniability, which conventional adversarial methods do not. As the opening example demonstrates, the attacker can manipulate an innocuous object to cause the detector to miss a critical target and simultaneously be legally shielded: they can blame the mistake on the machine learning system rather than admit to intentional deception. As a bonus, implementing intent obfuscation as a contextual attack opens up new avenues to attack the target, especially in situations where the attacker cannot manipulate the target directly. Moreover, contextual attacks are harder to detect since the defense algorithms not only need to inspect the target but also its surrounding region. The key question is whether perturbing one object to target another non-overlapping object is feasible on common detection models and object classes.

Feasibility is not guaranteed because object detectors are more complex than image classifiers. Detection involves both localization and classification, and its implementation varies widely across object detectors. The two most common types of object detectors (Zhao et al. 2019; Zou et al. 2019) are 1 and 2-stage detectors. 2-stage detectors usually perform localization and then classification, whereas 1-stage detectors typically perform both tasks simultaneously. As a result, contextual attacks on object detectors are harder to implement and typically less general, since a method that succeeds on 1-stage detectors may not apply to 2-stage detectors. But intent obfuscating attacks could nevertheless achieve success by exploiting the contextual reasoning of object detectors—detectors are known to use contextual information to improve performance, either implicitly through end-to-end training (e.g. YOLO Redmon et al. 2015) or explicitly through architectural design (Tong, Wu, and Zhou 2020, Section 2.4).

We implement intent obfuscating attacks on object detectors using the Targeted Objectness Gradient (TOG) algo-

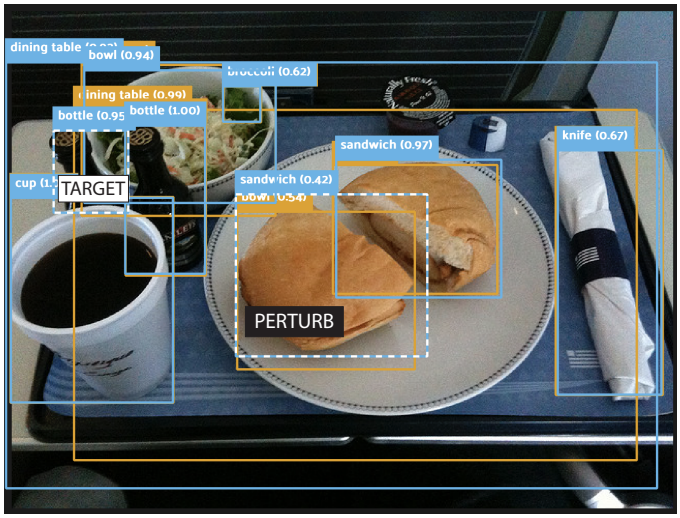

Figure 1: A vanishing attack perturbs a sandwich (dotted blue box) and causes YOLOv3 to miss the targeted bottle (no orange boxes are seen).

Figure 2: A mislabeling attack perturbs a sink and causes SSD to mislabel the targeted oven as a microwave with 0.96 confidence.

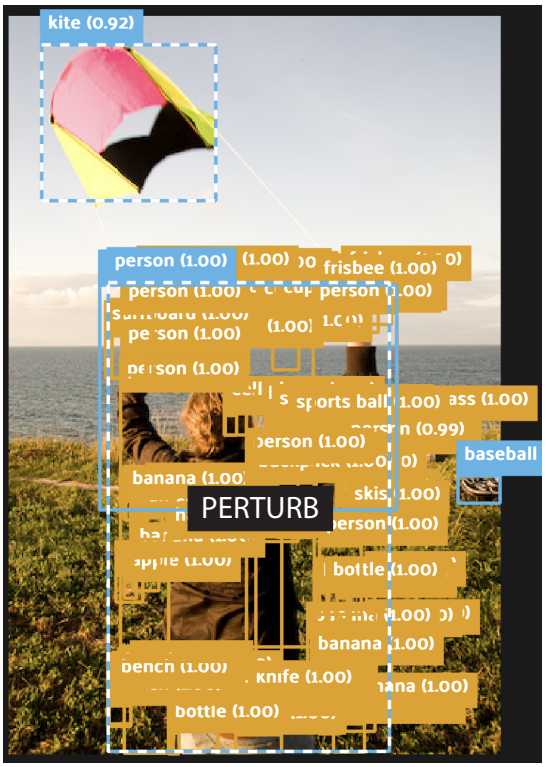

Figure 3: An untargeted attack perturbs a person and causes Faster R-CNN to miss the kite (and baseball) and hallucinate objects like bananas.

Figure 4: Disrupting a target object by perturbing another non-overlapping object enables intent obfuscating attacks to hide the attacker's intended target: The attacker can implement intent obfuscation using targeted (a) vanishing and (b) mislabeling attacks and (c) untargeted attacks, depending on their desired end result. Predictions on the original images are in blue and those on the adversarial images are in orange, with predictive confidence stated beside the class labels. The target and perturb objects are both dotted and labeled with "target" and "perturb" respectively. These examples are generated in the randomized experiment on the COCO dataset (Section 4). For clarity, the annotations are shown over the original images. Corresponding perturbed images are shown in Figure 16 in the appendix.

rithm (Chow et al. 2020b) because TOG achieves greater success than previous attacks like DAG (Xie et al. 2017), according to Chow et al. (2020a). In addition, as an iterative gradient-based algorithm, TOG can not only attack any modern state-of-the-art detector trained using backpropagation, but also enable the attacker to specify a precise target object for intent obfuscation. We apply TOG to both 1 and 2-stage detectors on the large-scale Microsoft Common Objects in Context (COCO) dataset (Lin et al. 2014). We contribute to the important and understudied issue of intent obfuscation in adversarial machine learning:

1. We are the first to propose an intent obfuscating attack on object detectors (Section 3).
2. We determine the feasibility of intent obfuscating attacks on 5 prominent detectors—YOLOv3, SSD, RetinaNet, Faster R-CNN, and Cascade R-CNN—for both targeted

and untargeted attacks (Section 4).
3. We analyze the success factors for intent obfuscating attacks, including detection models, attack modes, target object confidence and perturb object sizes (Sections 4.2 and 4.3).
4. We then exploit positive factors to increase success on all models and attacks by deliberately selecting perturb and target objects, as well as perturbing arbitrary regions, as shown in Figures 5 and 6 respectively (Section 5).

2 Related Work

Intent obfuscation: Intent obfuscation is rare in the machine learning literature. One exception is a paper by Zhang et al. (2019), which investigates intent obfuscation in inverse reinforcement learning and applies the modeling results to an intrusion detection system. Another is a highly cited article on intent obfuscation by Sharif et al. (2016). The article uses adversarially patterned spectacles to conduct intent obfuscating attacks on face recognition systems and enable "plausible deniability" (Sharif et al. 2016, introduction). In comparison, we execute intent obfuscating attacks on object detectors, which is a more general and challenging problem. Moreover, as opposed to wearing conspicuously printed spectacles (Sharif et al. 2016, Figure 4 and 5), we use contextual attacks to obfuscate intent, which not only arouse less suspicion but also open up new avenues for manipulating the target.

Contextual attacks: Previous research has attempted to exploit the contextual reasoning of object detectors to improve existing attacks or to design new attacks (Hu et al. 2021; Saha et al. 2020; Lee and Kolter 2019; Liu et al. 2018; Zhang, Zhou, and Li 2020; Cai et al. 2021). The first 4 citations illustrate purely contextual attacks by perturbing non-overlapping regions, most notably through an adversarial patch. We extend those papers to cover greater breadth with 5 models, 3 attack modes and 80 COCO classes, as well as depth by systematically testing 10 success factors. More importantly, intent obfuscating attacks and contextual attacks diverge in 3 important aspects:

1. **Aim:** Intent obfuscating attack aims to disrupt the target *and* hide intent. Contextual attack is a means to obfuscate intent. Alternative means could include showing the detection system a manipulated image while recording the original image in the system logs.
2. **Method:** Perturbing actual objects intuitively obfuscates intent more than perturbing a background region. A contextual attack does not distinguish the two.
3. **Results:** We analyze success factors which preserve intent obfuscation through non-overlapping perturbations. For contextual attacks, an overriding factor for ensuring success is to perturb the target object together with its surrounding context, as shown in (Zhang, Zhou, and Li 2020).

3 Intent Obfuscation

3.1 Attack Methods

We execute intent obfuscating attacks using the Targeted Objectness Gradient (TOG) algorithm (Chow et al. 2020b). TOG is an iterative gradient-based method similar to the Projected Gradient Descent (PGD) (Madry et al. 2017) attack and can be implemented both as untargeted and targeted attacks. We are most interested in the targeted attack because it gives the attacker precise control over the desired end result. A targeted attack achieves its purpose by manipulating the ground-truth for training the object detector. [1] The attacker can aim for the detector to mislabel the target object by changing its class label and retaining its original bounding box ("mislabeling" attack), or for the target object to vanish entirely by removing both its bounding box and class label from the ground-truth ("vanishing" attack). Their technical details are elaborated below:

Let θ be the model parameters, x the input image, y' the desired target, and $L(\theta, x, y')$ the optimization loss. The desired target y' could be derived by manipulating either the ground-truth or the model predictions. At iteration $t+1$, we add the signed gradients $\nabla_x L(\theta, x, y')$ times the learning rate α to the perturbed image in the previous iteration x^t. Then we limit the change in x to within the bounds S and iterate the process for a total of T iterations:

$$x^{t+1} = \Pi_{x+S}\left[x^t - \alpha \cdot \operatorname{sgn}(\nabla_x L(\theta, x, y'))\right] \quad (1)$$

Whereas a targeted attack minimizes the training loss towards the desired target, an untargeted attack maximizes (note the change in sign) the training loss $L(\theta, x, y)$ towards the original target y, which could either be the ground-truth or the model predictions:

$$x^{t+1} = \Pi_{x+S}\left[x^t + \alpha \cdot \operatorname{sgn}(\nabla_x L(\theta, x, y))\right] \quad (2)$$

The optimization loss L depends on the model, which we will present in the next section. Since the attacker will not have access to the ground-truth in most scenarios, we will conduct experiments by using the model predictions as y.

3.2 Model Losses

We attack 5 prominent detection models—comprising 3 1-stage detectors (SSD, YOLOv3, and RetinaNet) and 2 2-stage detectors (Faster R-CNN and Cascade R-CNN)—implemented in the versatile MMDetection toolbox (Chen et al. 2019) and pretrained on the COCO dataset (Lin et al. 2014). All models, besides the more recent and highly cited Cascade R-CNN, are spotlighted in reviews by Zhao et al. (2019) and Zou et al. (2019) and stated as the most widely implemented according to Papers With Code (2024). Table 1 summarizes the 5 detection models and corresponding attack losses. Full details are given below:

YOLOv3: YOLOv3 (Redmon and Farhadi 2018) prioritizes speed and uses a single convolutional network to predict bounding boxes and class labels. The class label is described by the objectness score, defined as the probability that the bounding box contains an object, and the class probability conditioned on the objectness score. Consequently, YOLOv3 has 3 training losses: the objectness loss, the class loss and the box regression loss (Redmon et al. 2015, equation 3). We attack the objectness loss for the vanishing attack and the

[1] For object detection, the ground-truth for a labeled object comprise 4 bounding box coordinates and 1 class label.

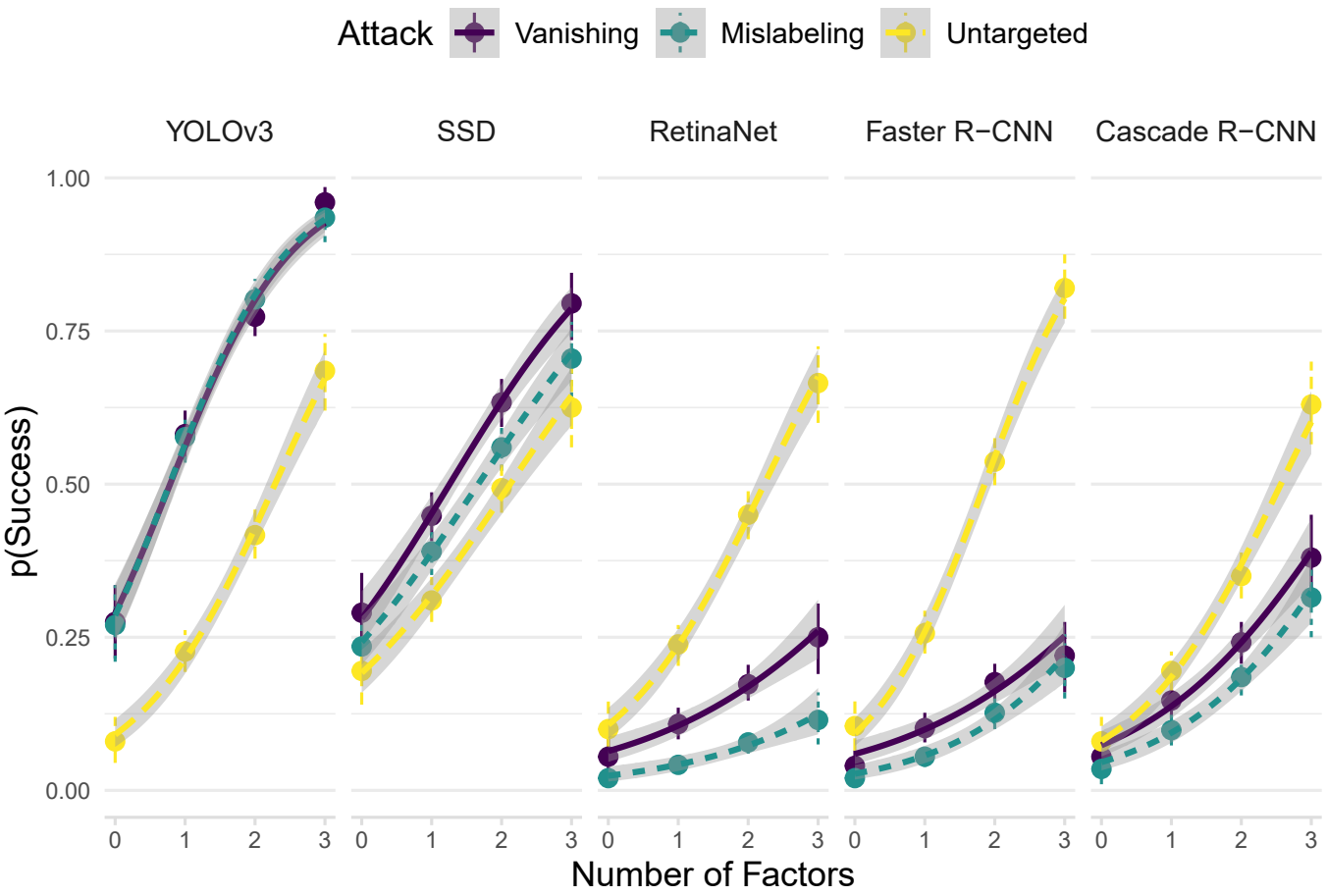

Figure 5: Success factors can be exploited in combination to significantly increase success rates: We sampled target and perturb objects based on three validated success factors in Table 2 by targeting objects with low predicted confidence, perturbing large objects and selecting target and perturb objects close to one another. The binned summaries and regression trendlines graph success proportion against number of factors in the deliberate attack experiment. Errors are 95% confidence intervals and every point aggregates success over 200 images. Success rates significantly increase as the number of factors combined increases. Significance is determined at $\alpha < 0.05$ using a Wald z-test on the logistic estimates. Full details are given in Section 5.1.

class loss for the mislabeling attack. For untargeted attack, we attack all training losses. Additionally, YOLOv3 is optimized through end-to-end training and "implicitly encodes contextual information" (Redmon et al. 2015, introduction). Therefore, it should be more vulnerable to contextual attacks. In the experiment, we use a pretrained YOLOv3 with a DarkNet-53 backbone and input size 608 $\times$ 608. The model achieves 33.7 COCO mean average precision (mAP), the primary metric in the COCO challenge (COCO 2024).

SSD: Like YOLOv3, SSD (Liu et al. 2015) also uses a single convolutional network and is optimized through end-to-end training, improving both speed and accuracy. Uniquely, SSD adds several convolutional layers which successively decrease in sizes after the base network. These layers predict bounding boxes at multiple sizes and aspect ratios. The training losses in SSD include box regression loss and class loss. Since the class loss includes the background class in addition to the 80 COCO class labels, we target the class loss for both vanishing and mislabeling attacks. For untargeted attack, we attack all training losses. In the experiment, we use a pretrained SSD with a VGG-16 backbone (Simonyan and Zisserman 2014) and input size 512 $\times$ 512. The model achieves 29.5 COCO mAP.

RetinaNet: RetinaNet (Lin et al. 2017b) uses a novel Focal Loss to address class imbalance in training 1-stage detectors: most training examples belong to the easily categorized background class and thereby overwhelm the training signal. Focal Loss mitigates the issue by down-weighting easily categorized background examples during training to emphasize the harder object examples and thereby increases training accuracy. RetinaNet also incorporates convolutional layers structured as a Feature Pyramidal Network (FPN) (Lin et al. 2017a) for multi-scale detection. Like SSD, RetinaNet's training losses comprise both the class loss (which includes the background class) and bounding box loss. We target the class loss for both vanishing and mislabeling attacks. For untargeted attack, we attack all training losses. In the experiment, we use a pretrained RetinaNet with a ResNet-50 backbone

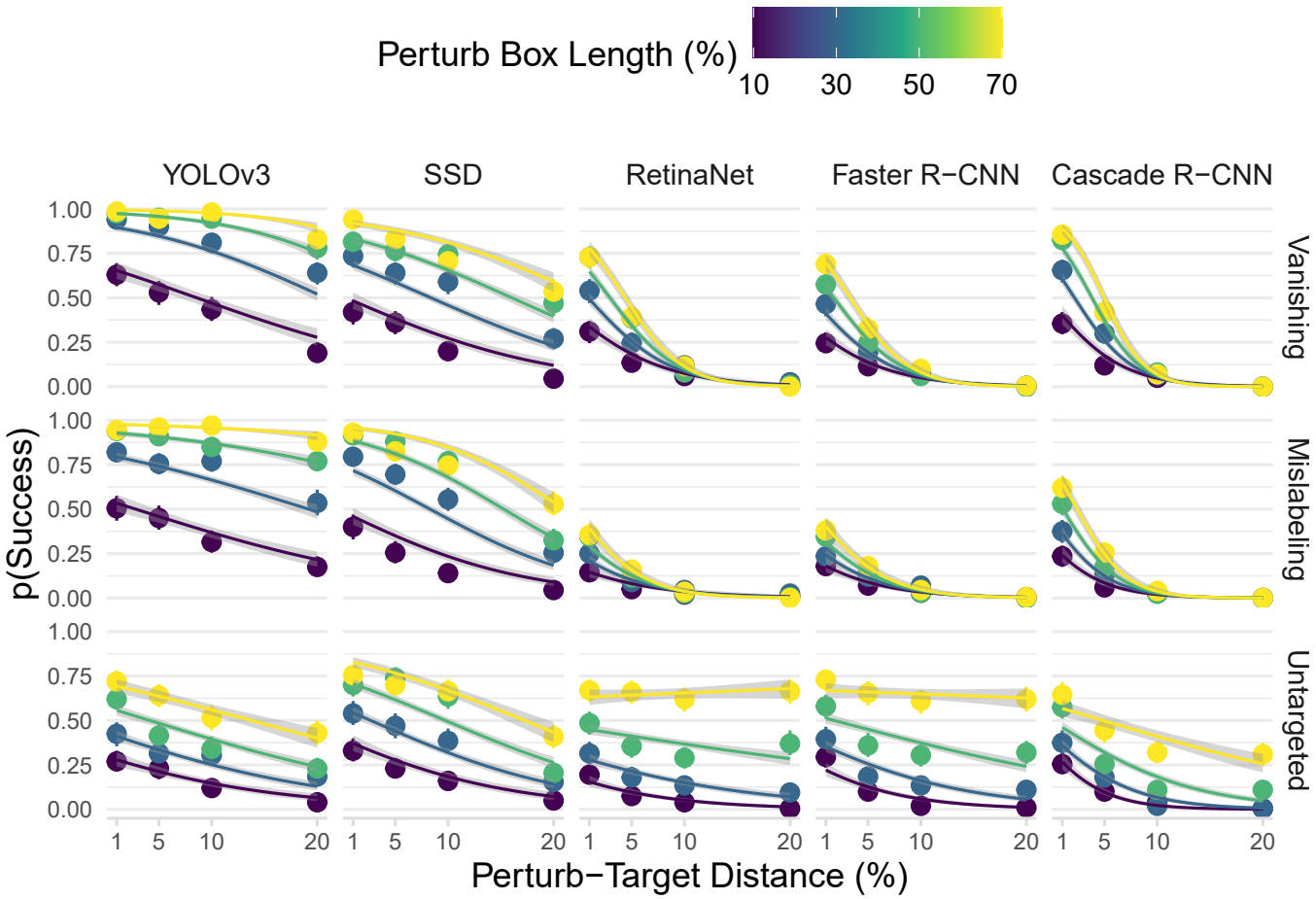

Figure 6: Perturbing an arbitrary region obfuscates intent with increased success for all models and attacks: We implement intent obfuscating attack by perturbing an arbitrary non-overlapping square region to disrupt a randomly selected target object at various lengths and distances. The binned summaries and regression trendlines graph success proportion against perturb-target distance and perturb box length, both relative to image width or height, in the deliberate attack experiment. Errors are 95% confidence intervals and every point aggregates success over 200 images. The deliberate attack multiplies success as compared to the randomized attack (Figure 7), especially at close perturb-target distance and large perturb box length. Full details are given in Section 5.2.

(He et al. 2015). The model achieves 36.5 COCO mAP.

Faster R-CNN: Faster R-CNN (Ren et al. 2015) adds a region proposal network (RPN) to the detection network in Fast R-CNN (Girshick 2015) to improve both speed and accuracy. Faster R-CNN begins detection with a base network to extract convolutional features. Then using these convolutional features, the RPN proposes object regions with associated objectness scores. The detection network then uses both the convolutional features and region proposals to predict bounding boxes and class labels. Hence, Faster R-CNN has 4 training losses: the box regression loss and objectness loss in the RPN and the box regression loss and class loss in the detection network. Since the class loss for the detection network also includes the background class in addition to the 80 COCO class labels (Girshick 2015, equation 1), we attack both the class loss and objectness loss for the vanishing attack and attack only the class loss for the mislabeling attack. For untargeted attack, we attack all training losses. In the experiment, we use the pretrained Faster R-CNN with a ResNet-50 backbone and FPN. The model achieves 37.4 COCO mAP.

Cascade R-CNN: Cascade R-CNN (Cai and Vasconcelos 2017) extends the Faster R-CNN architecture with a cascade structure to generate more accurate detections. Cascade R-CNN repeats the RPN stage in Faster R-CNN thrice to increase proposals quality. The 2nd and 3rd RPNs in Cascade R-CNN also propose class labels (which include the background class) rather than only the objectness score in the 1st RPN. All 3 RPNs also predict bounding box coordinates. Hence, the training losses for Cascade R-CNN comprise 4 box regression losses, 3 class losses and 1 objectness loss. We attack the objectness loss and class losses for the vanishing attack and attack all class losses for the mislabeling attack. For untargeted attack, we attack all training losses. In the experiment, we use a pretrained Cascade R-CNN with a ResNet-50 backbone and FPN. The model achieves 40.3

COCO mAP.

4 Randomized Attack

4.1 Setup

We evaluate the 3 intent obfuscating attacks—vanishing, mislabeling and untargeted—on the 5 models using the 2017 COCO dataset (Lin et al. 2014). The COCO dataset has 80 categories of common objects in everyday scenes for object detection and the 2017 split has 118,000 train images and 5,000 test images (Papers with Code 2024). We use the test images to attack the 5 models with pretrained weights obtained through MMDetection (Chen et al. 2019) and visualized the results using the FiftyOne visualization app (Moore, B. E. and Corso, J. J. 2020).

Target and perturb objects selection: First, we evaluate the models on the original images and count a detection as correct when both the bounding box and the class label match the ground-truth with at least 0.3 intersection-over-union (IOU) and 0.3 confidence respectively. Note that we do not use the standard COCO mean average precision (mAP) metric since mAP measures detection precision over the whole dataset, but we are interested in evaluating success for single objects. After getting the initial predictions, we restrict only to the correctly predicted objects. Then we randomly sample a target object and another *non-overlapping* perturb object per image. Images with less than 2 correctly predicted non-overlapping objects are ignored.

Ground-truth manipulation for targeted attack: Then we create the desired target y' from the ground-truth y for the 2 targeted attacks (vanishing and mislabeling equation 1). For the vanishing attack, we remove the target object entirely—both the class label and bounding box—from the ground-truth y to get y'. For the mislabeling attack, we change the class label of the target object in y to a random class ("intended class" from now on) to get the desired target y'. For the untargeted attack, we evaluate the randomly selected target object only to compare success rates with the 2 targeted attacks.

Attack parameters: Next, we run the 3 attacks using iterations 10, 50, 100, and 200, but not more than 200 since success rates plateau after. For every iteration, we set a learning rate α which could maximally change a pixel from 0 (black) to 1 (white). For instance, we use a 0.1 learning rate for 10 iterations. In addition, we set a perturbation bound S such that the image remains in its original range of $[0, 1]$ after every iteration. We also repeated the simulations with an l_∞-norm of 0.05 applied after every iteration. Since the norm constraint is not central to intent obfuscating attacks, we put its results in the appendix. For every model, attack and iteration combination, we resampled 4,000 test images.

Results evaluation: We distort the bounding box of the perturb object and then re-evaluate the generated adversarial image: as in the initial evaluation step, we use IOU and confidence thresholds of 0.3 to determine whether the attack succeeds in disrupting the target object. The attack speed mainly depended on model complexity. More experimental details are included in Appendix B.1.

4.2 Hypotheses

We conducted a thorough analysis by listing 10 hypotheses increasing success rates and systematically testing whether those hypotheses are valid in the next section. For all attacks, we expect to achieve higher success rates for:

1. **1-stage (YOLOv3, SSD, and RetinaNet) than 2-stage (Faster R-CNN and Cascade R-CNN) detectors:** intuitively, perturbing an input pixel to change one loss component in an intended direction is easier than for multiple loss components. As the number of loss components increases, the chances that the same perturbation will change all losses in the same direction decreases, making the overall attack harder. Because we attack more loss components for 2-stage than 1-stage detectors, we expect to achieve correspondingly lower success rates for 2-stage detectors, beyond what could be explained by their higher COCO mAPs listed in Table 1.
2. **Targeted than untargeted attack:** the gradient signal in a targeted attack is precisely aimed at the target object, whereas for an untargeted attack the gradient signal is broadly aimed at all objects in the image. Therefore, the chances that an untargeted attack disrupts the target object is lower.
3. **Vanishing than mislabeling attack:** converting the original class label to the background class should be easier than to non-background classes, since the background class contains everything not labeled in the COCO dataset and thereby makes up a large portion of the input space.
4. **Larger attack iterations:** we expect larger attack iterations to achieve better local minima and maxima respectively for targeted and untargeted attacks since more iterations allow more possible routes to navigate across the loss landscape.
5. **Target objects with lower predicted confidence:** the higher the predicted confidence, the larger the decrease in class probability needed to achieve success and the more the attack has to perturb the class loss.
6. **Perturb objects with larger bounding boxes:** larger bounding boxes enable the attack to perturb more pixels, after controlling for Hypothesis 7.
7. **Shorter distance between perturb and target objects:** since object detectors likely utilize nearby context to make predictions, perturbing nearby pixels should change the predictions more. Because larger perturb objects (Hypothesis 6) are more likely to be closer to the target object, we will control for both with a regression model.
8. **Target object classes with lower COCO mean accuracy:** when an object detector achieves lower mean accuracy for particular classes on the COCO dataset, attacking target objects belonging to those classes should be easier. When the target object class has lower mean accuracy, the target object will likely be predicted with lower confidence. Considering Hypothesis 5, we will also control for the latter.

For specific attacks, we expect to achieve higher success rates for

9. **Target objects with lower intersection-over-union (IOU) for the untargeted attack:** the lower the IOU of predicted and ground-truth bounding boxes, the less the

Detectors	Stages[a]	COCO mAP[b]	Attack Losses[c]		
			Targeted		
			Vanishing	Mislabeling	Untargeted[d]
YOLOv3	1	33.7	Object	Class	Class, Box, Object
SSD	1	29.5	Class	Class	Class, Box
RetinaNet	1	36.5	Class	Class	Class, Box
Faster R-CNN	2	37.4	RPN: Object; Det: Class	Det: Class	RPN: Object, Box; Det: Class, Box
Cascade R-CNN	2	40.3	RPN 1: Object; RPNs 2, 3 + Det: Class	RPNs 2, 3: Class; Det: Class	RPN 1: Object, Box; RPNs 2, 3 + Det: Class, Box

[a] In general, 1-stage detectors are quicker whereas 2-stage detectors are more accurate, though the 1-stage RetinaNet aims to be both quick and accurate. In a 2-stage detector, the input image passes through a Region Proposal Network (RPN) stage and a detection (Det) stage.
[b] COCO mean Average Precision (mAP) is the primary metric on the COCO challenge.
[c] The training losses in detectors typically include the box regression loss (Box), the class loss on the 80 COCO labels and/or the background class (Class), and the objectness loss on categorizing an image region as background or object (Object).
[d] Untargeted attack targets all training losses in a model, i.e. the backpropagation loss.

Table 1: Detection models and attack losses. Full details are given in Appendix 3.2.

untargeted attack has to perturb the box loss to misalign the detection to less than the IOU threshold.

10. **Intended classes with higher probabilities for the mislabeling attack:** in a mislabeling attack we aim to change the target prediction to the intended class. When the intended class has higher probability on the original image, the increase in probability of the intended class required for the detector to mislabel the target is smaller, and the attack would have to change the class loss by a lesser degree. The reasoning is similar to the one in Hypothesis 5. In addition, since higher probability of the intended class likely entails lower confidence of the predicted class [2], we will also control for the latter.

4.3 Results

The success rates without norm constraint are shown in Figure 7. Imposing a 0.05 l_∞-norm constraint slightly decreases success, as shown in Figure 15 in the appendix, but the trends remain the same. Hence, we will only conduct hypothesis testing on the results without norm constraint.

For all hypotheses, we use logistic regression to determine if the stated variables significantly predict success rates. We transform the predictors as appropriate and run separate regressions for every model and attack combination, unless the predictor variable includes model (Hypothesis 1) or attack (Hypotheses 2 and 3). Except for testing the effect of iterations (Hypothesis 4), we restrict the data to the maximum 200 attack iterations to analyze the strongest possible results. We computed the p-values using a Wald z-test and set the significance level (α) to the usual 0.05. Attacked images are illustrated in Figure 4 and hypotheses and results are summarized in Table 2. We will state the conclusions below. Graphs and tabulated statistics are in Appendix B.2.

1. **1-stage (YOLOv3, SSD, and RetinaNet) than 2-stage (Faster R-CNN and Cascade R-CNN) detectors:** As shown in Figure 7, both vanishing and mislabeling attacks achieve significantly higher success rates for 1-stage than 2-stage detectors. The higher success on 1-stage detectors could not be explained by their lower COCO mAPs. Surprisingly, the 1-stage RetinaNet is as robust as 2-stage detectors—training RetinaNet using Focal Loss not only boosts COCO accuracy but also increases resilience against intent obfuscating attacks (Table 3).
2. **Targeted than untargeted attack:** The results are mixed: targeted attack is significantly more successful than untargeted attack for YOLOv3 and slightly more successful for SSD, but the increase is non-existent or reversed for RetinaNet, Faster R-CNN and Cascade R-CNN (Table 4 and Figure 7). As stated in Result 1, RetinaNet, Faster R-CNN and Cascade R-CNN are more robust than YOLOv3 and SSD against intent obfuscating attack, and perhaps more robust models require a coordinated attack against all loss components to achieve success.
3. **Vanishing than mislabeling attack:** Vanishing attack achieves significantly more success than mislabeling attack for all models (Table 4 and Figure 7).
4. **Larger attack iterations:** Larger attack iterations (log-transformed) significantly increase success for all models and attacks (Table 5).
5. **Target objects with lower predicted confidence:** Lower target confidence significantly increases success rates for all models and attacks (Table 6 and Figure 10).
6. **Perturb objects with larger bounding boxes:** Larger perturb objects significantly increase success rates for all models and attacks, except for mislabeling attacks on Faster R-CNN, after controlling for perturb-target dis-

[2] To be clear, class probability and confidence are the same. In alignment with the object detection literature, I will use confidence to mean probability only for the predicted class.

tances (Table 7 and Figure 11).

7. **Shorter distance between perturb and target objects:** Shorter perturb-target distances significantly increase success rates for all models and attacks, after controlling for perturb object sizes (Table 7 and Figure 11).
8. **Target classes with lower COCO mean accuracy:** The results are mixed: of the 15 model and attack combinations, higher COCO class accuracy significantly decreases success rates for 5 combinations but increases success rates for 4, after controlling for target class confidence. The relatively large interaction terms make interpretation challenging (Table 8 and Figure 12).
9. **Target objects with lower intersection-over-union (IOU) for the untargeted attack:** Lower IOU increases success rates for untargeted attack on all models (Table 10 and Figure 14).
10. **Intended classes with higher probabilities for the mislabeling attack:** The results are mixed: higher intended class probability (log-transformed) does *not* predict success rates for mislabeling attack after controlling for target class confidence for SSD, Faster R-CNN, and Cascade R-CNN. However, it is significantly negative for YOLOv3 and positive for RetinaNet. (Table 9 and Figure 13).

5 Deliberate Attack

Rather than randomly selecting target and perturb objects in the randomized experiment, the attacker can—and will—select objects to exploit the success factors listed in the previous section. For instance, to maximize havoc on a congested street, he may target the stop sign with the lowest predicted confidence (Result 5) and use a vanishing attack if most self-driving cars use a detector based on YOLO (Result 1). He could also increase success by deliberately perturbing larger objects (Result 6) closer to the target (Result 7). Moreover, he can easily multiply success on a random target for any detector by perturbing a large *arbitrary* region close to the target object. We run experiments for the two common scenarios of deliberately selecting target and perturb objects and perturbing an arbitrary region in Sections 5.1 and 5.2 respectively.

5.1 Selecting Easier Targets

Building on our randomized attacks described in Section 4, we deliberately exploit 3 validated success factors in Table 2 to select:

1. Target objects with less than 0.5 predicted confidence.
2. Perturb objects with bounding boxes more than 25% of the image size.
3. Perturb and target objects with distances less than 25% across the image. [3]

We test all combinations. For every combination, we resample 200 COCO test images and run the 3 attacks for 200 iterations.

[3]We use an algorithm in game development (congusbongus 2018) to compute the minimum distances between the perturb and target bounding boxes. We set the image width and height to 1 and select perturb and target objects with distances less than 0.25.

Hypotheses (higher success for)	Accepted (across attacks and models)[a]
1-stage $>$ 2-stage models (YOLOv3, SSD, RetinaNet $>$ Faster R-CNN, Cascade R-CNN)	YOLOv3, SSD $>$ RetinaNet, Faster R-CNN, Cascade R-CNN in vanishing and mislabeling attacks (1-stage RetinaNet is as resilient as 2-stage models)
Targeted $>$ Untargeted attack	YOLOv3 only
Vanishing $>$ Mislabeling attack	All
Larger attack iterations	All
Less confident targets	All
Larger perturb boxes	All except mislabeling attack on Faster R-CNN
Shorter perturb-target distance	All
Less accurate target COCO class	Mixed
Lower target IOU[b] (untargeted attack only)	All
More probable intended class (mislabeling attack only)	Mixed

[a] $p < .05$ for Wald z-test on logistic estimate

[b] intersection-over-union

Table 2: Hypothesis testing in the randomized attack (Sections 4.2 and 4.3)

Hypotheses We tested the 3 success factors in Section 4.3 and all are shown to individually increase success rates. Now we hypothesize that these success factors *independently* increase success rates (i.e., success rates increase as the number of factors combined increases).

Results As shown in Figure 5, success rates increase as the number of factors used in combination increases. The attacker who includes all 3 factors obtains for the vanishing and mislabeling attacks more than 90% success on YOLOv3 and more than 70% success on SSD, and for the untargeted attack more than 60% success on RetinaNet, Faster R-CNN and Cascade R-CNN. A success example is illustrated in Figure 8. Imposing a 0.05 l_∞-norm constraint slightly decreases success, as shown in Figure 18 in the appendix. Since the trends remain the same, we will only conduct hypothesis testing based on the results without norm constraint. Hypothesis testing is similar to the procedure in the randomized experiment (Sections 4.2 and 4.3). A logistic regression model shows that success rates significantly increase as more factors are combined to select target and perturb objects for all models and attacks. Statistics are given in Table 11 in the appendix.

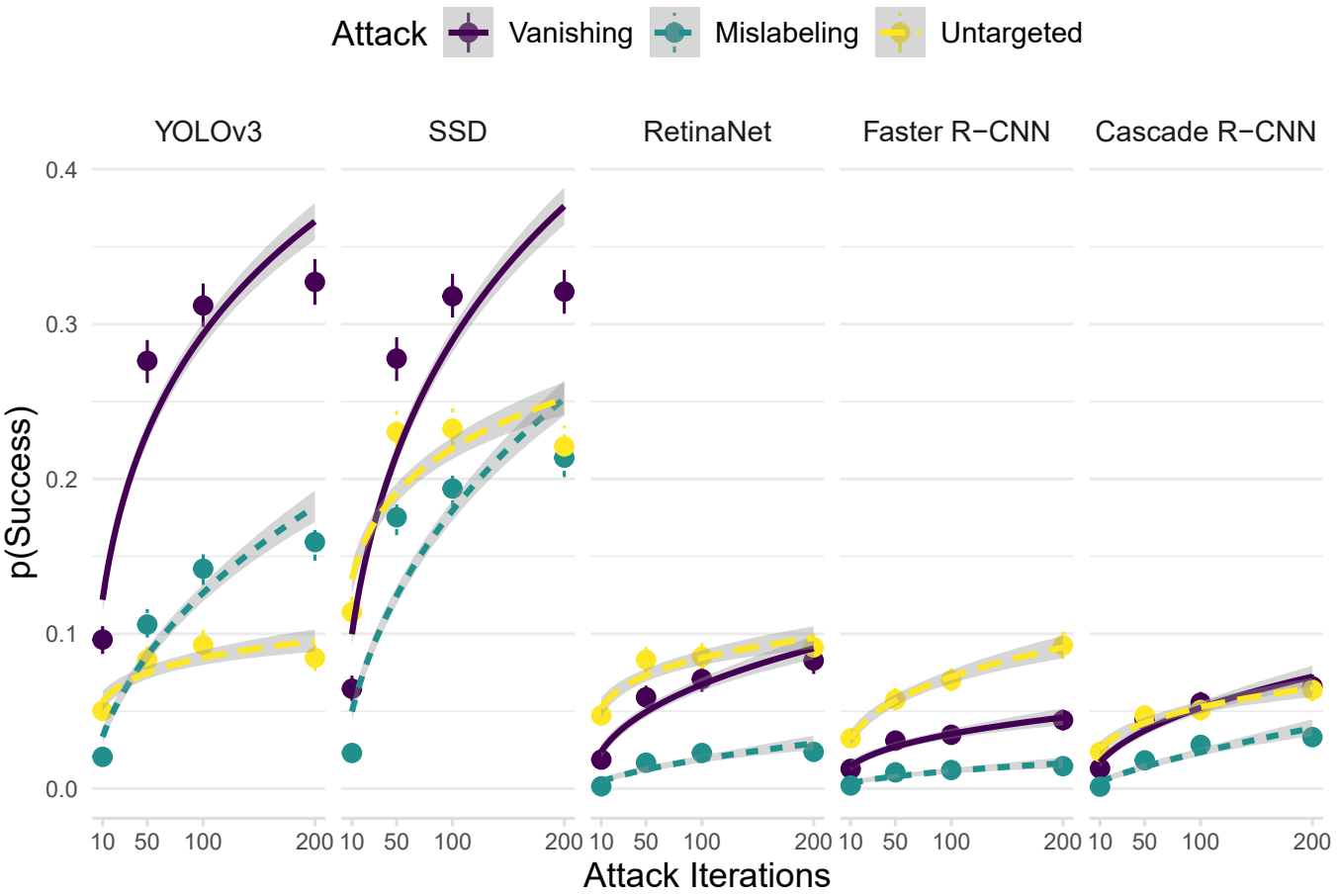

Figure 7: Intent obfuscating attack is feasible for all models and attacks: We conduct a randomized experiment by resampling COCO images, and within those images randomly sampling correctly predicted target and perturb objects. Then we distort the perturb objects to disrupt the target objects varying the attack iterations. The binned summaries and regression trendlines graph success proportion against attack iterations in the randomized attack experiment. Errors are 95% confidence intervals and every point aggregates success over 4,000 images. Targeted vanishing and mislabeling attacks obtain significantly greater success on the 1-stage YOLOv3 and SSD than the 2-stage Faster R-CNN and Cascade R-CNN detectors. However, the 1-stage RetinaNet is as resilient as the 2-stage detectors. Moreover, success rates significantly increase with larger attack iterations. Significance is determined at $\alpha < 0.05$ using a Wald z-test on the logistic estimates. Full details are given in Section 4.

5.2 Perturbing Arbitrary Regions

When a perturbed object could not be selected easily, the attacker can also perturb an arbitrary region in the image to obfuscate intent.

Setup We adopt the setup in the randomized attack (Section 4.1). However, rather than randomly selecting target and perturb objects, we randomly select a target object and then enclose a non-overlapping square perturb region beside it. We vary the length of the square perturb region to be 10, 30, 50, and 70% of the image width or height, and vary the distance between the target and perturb bounding boxes to be 1, 5, 10, and 20% of the image width or height. More details are given in Figure 21 in the appendix. We test all combinations. For every combination, we resample 200 COCO test images and run the 3 attacks for 200 iterations.

Hypotheses Actively manipulating only the perturb sizes and target-perturb distances makes the deliberate attack more controlled than the randomized attack. Hence, although we are proposing similar hypotheses to those in the randomized attack (Hypotheses 6 and 7), we can more strongly claim that larger perturb sizes or shorter distances *cause* success rates to increase.

Results Success rates greatly increase compared to the randomized attack (Figure 6): when perturb lengths are more than 50% of the image length and perturb-target distances are less than 5% of the image length, the attacker obtains for the vanishing attack nearly 100% success rates on YOLOv3 and SSD, and for the untargeted attack more than 25% on RetinaNet, Faster R-CNN and Cascade R-CNN. Imposing a 0.05 l_∞-norm constraint slightly decreases success, as shown in Figure 20 in the appendix, but it is still greater than the randomized attack. A success example is illustrated in Figure 9. Since the trends remain the same, we will only conduct hypothesis testing based on the results without norm constraint,

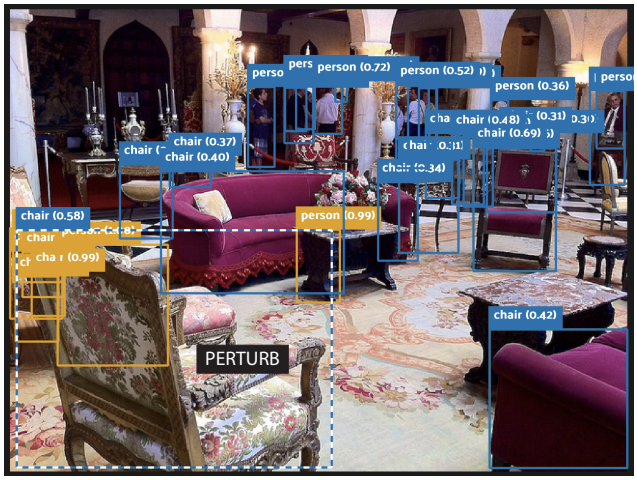

Figure 8: We can increase success rates by intentionally selecting target and perturb objects: An untargeted attack perturbs a large chair (dotted blue box) and causes RetinaNet to misplace the locations of people and chairs. Predictions on the original image are shown in blue and those on the adversarial image are shown in orange. For simplicity, only detections of people and chairs are shown. The corresponding perturbed image is shown in Figure 17 in the appendix.

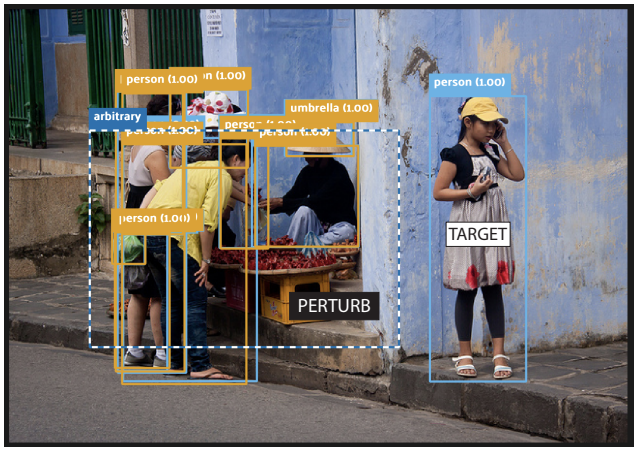

Figure 9: We can implement an intent obfuscating attack via perturbing an arbitrary region rather than an actual object: A mislabeling attack on Cascade R-CNN perturbs a non-overlapping arbitrary region (dotted blue) and causes the targeted person to vanish. Predictions on the original image are shown in blue and those on the adversarial image are shown in orange. The corresponding perturbed image is shown in Figure 19 in the appendix.

similar to the previous two experiments.

Hypothesis testing is similar to the procedure in the randomized experiment (Sections 4.2 and 4.3): A logistic regression model using both terms as predictors show that longer perturb lengths and shorter perturb-target distances cause success rates to increase significantly for all model and attack combinations. Statistics are given in Table 12 in the appendix.

6 Discussion and Conclusion

Perturbing objects versus non-objects: For intent obfuscating attacks, perturbing actual objects is intuitively more misleading than perturbing non-objects, and there is no a priori reason to believe that either will change success rates. Should the attacker then always perturb objects rather than non-objects? Surprisingly no: hypothesis testing showed that perturbing an object (in the randomized attack) rather than a non-object (in the deliberate attack) significantly *decreases* success rates for most model and attack combinations, after controlling for perturb sizes and perturb-target distances, as shown in Table 13 in the appendix. Interestingly, while intent obfuscation is possible, it is more difficult to achieve than a mere contextual attack.

Limitations: We have shown that intent obfuscating attacks are feasible for the 5 prominent object detectors and analyzed 10 success factors. Although we did not conduct experiments in which the attacker has no access to the victim detector, we believe that the breadth and depth of the paper will illuminate the success characteristics of intent obfuscating attacks in both settings. Interested readers can turn to Cai et al. (2021) for black-box contextual attacks and Lee and Kolter (2019) for physical contextual attacks.

7 Broader Impact

We have demonstrated that a malicious actor can use an intent obfuscating attack to disrupt AI systems while maintaining plausible deniability. An intent obfuscating attack goes beyond a mere contextual attack. By carefully selecting non-overlapping target and perturb regions, the malicious actor can deceive a human detective into believing their actions were innocuous.

A key defense against the attack is to use 2-stage detectors like Faster R-CNN and Cascade R-CNN. These models are shown to be more robust than 1-stage detectors like YOLOv3 and SSD against all three attacks. Indeed, whether to use 1-stage or 2-stage detectors is not only a matter of speed or accuracy; machine learning engineers also have to consider whether the increased resilience against intent obfuscating attacks makes 2-stage detectors more suitable, particularly in security-critical applications.

Besides the technical recommendation, we would like to raise an important legal concern: there is hardly any legal protection against intent obfuscating attacks. Established cybersecurity laws (like the United States CFAA) do not address adversarial machine learning explicitly (Kumar et al. 2018, 2020). Intent obfuscation attacks only compound the problem, since proving malicious intent is required for criminal prosecution (Wex Definitions Team 2024). To conclude, we believe that establishing the feasibility of intent obfuscating attacks will galvanize the machine learning community to develop more robust technical and legal solutions.

8 Code and Data

The code is available on the github repository https://github.com/zhaobin-li/intent-obfusc. The included README.md

contains instructions to reproduce graphs and tables, download datasets and images, visualize attacked datasets, and replicate experiments. The datasets and perturbed images in both experiments are stored on a Google Cloud Storage bucket https://console.cloud.google.com/storage/browser/intent-obfusc (you will still need to sign in with a google account simply to access the public bucket).

Acknowledgements

We thank Scott Cheng-Hsin Yang and Wei-Ting Chiu for editing the paper. This work was supported in part by a grant from the DARPA RED program (20-430 Rev00-NJ-112) to PS.

References

Cai, Z.; and Vasconcelos, N. 2017. Cascade R-CNN: Delving into high quality object detection. 6154–6162.

Cai, Z.; Xie, X.; Li, S.; Yin, M.; Song, C.; Krishnamurthy, S. V.; Roy-Chowdhury, A. K.; and Salman Asif, M. 2021. Context-Aware Transfer Attacks for Object Detection.

Chen, K.; Wang, J.; Pang, J.; Cao, Y.; Xiong, Y.; Li, X.; Sun, S.; Feng, W.; Liu, Z.; Xu, J.; Zhang, Z.; Cheng, D.; Zhu, C.; Cheng, T.; Zhao, Q.; Li, B.; Lu, X.; Zhu, R.; Wu, Y.; Dai, J.; Wang, J.; Shi, J.; Ouyang, W.; Loy, C. C.; and Lin, D. 2019. MMDetection: Open MMLab Detection Toolbox and Benchmark.

Chow, K.-H.; Liu, L.; Gursoy, M. E.; Truex, S.; Wei, W.; and Wu, Y. 2020a. Understanding Object Detection Through an Adversarial Lens. In *Computer Security – ESORICS 2020*, 460–481. Springer International Publishing.

Chow, K.-H.; Liu, L.; Loper, M.; Bae, J.; Gursoy, M. E.; Truex, S.; Wei, W.; and Wu, Y. 2020b. Adversarial Objectness Gradient Attacks in Real-time Object Detection Systems. In *2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, 263–272. IEEE.

COCO. 2024. COCO. https://cocodataset.org/. Accessed: 2024-5-2.

congusbongus. 2018. Efficient minimum distance between two axis aligned squares? https://gamedev.stackexchange.com/questions/154036/efficient-minimum-distance-between-two-axis-aligned-squares. Accessed: 2024-3-6.

Girshick, R. 2015. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 1440–1448. IEEE.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep residual learning for image recognition. 770–778.

Hu, S.; Zhang, Y.; Laha, S.; Sharma, A.; and Foroosh, H. 2021. CCA: Exploring the Possibility of Contextual Camouflage Attack on Object Detection. In *2020 25th International Conference on Pattern Recognition (ICPR)*, 7647–7654. IEEE.

Kumar, R. S. S.; O'Brien, D. R.; Albert, K.; and Vilojen, S. 2018. Law and Adversarial Machine Learning.

Kumar, R. S. S.; Penney, J.; Schneier, B.; and Albert, K. 2020. Legal Risks of Adversarial Machine Learning Research.

Larmarange, J.; and Sjoberg, D. D. 2024. *broom.helpers: Helpers for Model Coefficients Tibbles*. R package version 1.15.0.

Lee, M.; and Kolter, Z. 2019. On Physical Adversarial Patches for Object Detection.

LIFARS. 2020. What Is Obfuscation In Security And What Types of Obfuscation Are There? https://www.lifars.com/2020/11/what-is-obfuscation-in-security/. Accessed: 2023-1-26.

Lin, T.-Y.; Dollar, P.; Girshick, R.; He, K.; Hariharan, B.; and Belongie, S. 2017a. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017b. Focal Loss for Dense Object Detection.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014*, 740–755. Springer International Publishing.

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C. 2015. SSD: Single Shot MultiBox Detector.

Liu, X.; Yang, H.; Liu, Z.; Song, L.; Li, H.; and Chen, Y. 2018. DPatch: An Adversarial Patch Attack on Object Detectors.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards Deep Learning Models Resistant to Adversarial Attacks.

Moore, B. E. and Corso, J. J. 2020. FiftyOne. *GitHub. Note: https://github.com/voxel51/fiftyone*.

Papers with Code. 2024. COCO Dataset. https://paperswithcode.com/dataset/coco. Accessed: 2024-5-2.

Papers With Code. 2024. Object Detection. https://paperswithcode.com/task/object-detection. Accessed: 2024-5-2.

R Core Team. 2024. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2015. You Only Look Once: Unified, Real-Time Object Detection.

Redmon, J.; and Farhadi, A. 2018. YOLOv3: An Incremental Improvement.

Ren, K.; Zheng, T.; Qin, Z.; and Liu, X. 2020. Adversarial Attacks and Defenses in Deep Learning. *Engineering*, 6(3): 346–360.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks.

Robinson, D.; Hayes, A.; and Couch, S. 2024. *broom: Convert Statistical Objects into Tidy Tibbles*. R package version 1.0.6.

Saha, A.; Subramanya, A.; Patil, K.; and Pirsiavash, H. 2020. Role of spatial context in adversarial robustness for object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 784–785. IEEE.

Sharif, M.; Bhagavatula, S.; Bauer, L.; and Reiter, M. K. 2016. 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*, CCS '16, 1528–1540. New York, NY, USA: Association for Computing Machinery.

Simonyan, K.; and Zisserman, A. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition.

Tong, K.; Wu, Y.; and Zhou, F. 2020. Recent advances in small object detection based on deep learning: A review. *Image and Vision Computing*, 97: 103910.

Wex Definitions Team. 2024. intent. https://www.law.cornell.edu/wex/intent. Accessed: 2024-5-2.

Xie, C.; Wang, J.; Zhang, Z.; Zhou, Y.; Xie, L.; and Yuille, A. 2017. Adversarial examples for semantic segmentation and object detection. 1369–1378.

Xie, Y. 2024. *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.47.

Xu, H.; Ma, Y.; Liu, H.-C.; Deb, D.; Liu, H.; Tang, J.-L.; and Jain, A. K. 2020. Adversarial Attacks and Defenses in Images, Graphs and Text: A Review. *International Journal of Automation and Computing*, 17(2): 151–178.

Zhang, H.; Zhou, W.; and Li, H. 2020. Contextual Adversarial Attacks For Object Detection. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, 1–6. IEEE.

Zhang, X.; Zhang, K.; Miehling, E.; and Başar, T. 2019. Non-cooperative inverse reinforcement learning. 9482–9493.

Zhao, Z.-Q.; Zheng, P.; Xu, S.-T.; and Wu, X. 2019. Object Detection With Deep Learning: A Review. *IEEE Trans Neural Netw Learn Syst*, 30(11): 3212–3232.

Zhu, H. 2024. *kableExtra: Construct Complex Table with kable and Pipe Syntax*. R package version 1.4.0.

Zou, Z.; Chen, K.; Shi, Z.; Guo, Y.; and Ye, J. 2019. Object Detection in 20 Years: A Survey.