On feasibility of intent obfuscating attacks

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

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 incorporating intent obfuscation in generating 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 known defenses and legal repercussions.

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 was only an art. Because the sticker was on the bench but the effect was on the sign, authorities are unable to prove intent. This thought experiment highlights two serious implications of intent obfuscating attacks: it opens up new avenues for harmful exploits, and provides the culprit with "plausible deniability".

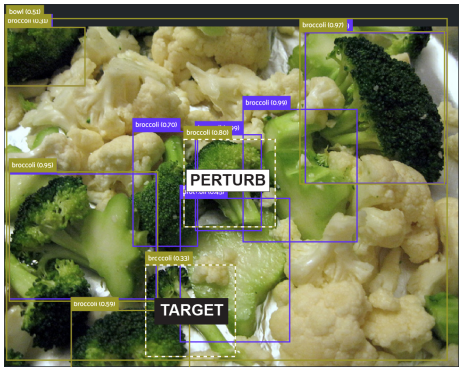
Considering the potential significance of intent obfuscating attacks, it is important for the machine learning community to understand and defend against such attacks. Intent obfuscation, though a common practice in cyberattacks for penetrating target systems (LIFARS, 2020), has rarely been raised in 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: the attacker 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 the common detection models and object classes.

Feasibility is not guaranteed because object detectors are more complex then image classifiers, and consequently adversarial attacks on object detectors are harder to implement and typically less general. Most prominent attacks like placing adversarial patches on stop-signs (Eykholt et al., 2018) or the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014) perturb the target objects themselves. But intent obfuscating attacks could nevertheless achieve success by exploiting the contextual reasoning of object detectors: Detectors are known to use contextual information

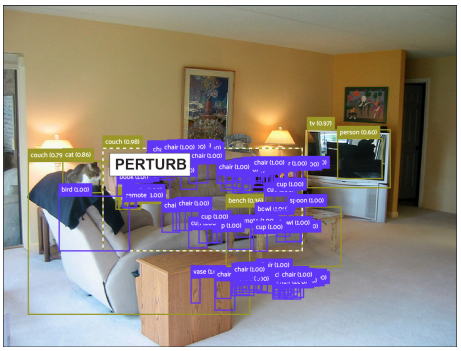
[1]Department of Mathematics and Computer Science, Rutgers University–Newark, New Jersey, USA [2]School of Mathematics, Institute for Advanced Study, New Jersey, USA. Correspondence to: Patrick Shafto <patrick.shafto@rutgers.edu>.

2^{nd} AdvML Frontiers workshop at 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.



(a) A vanishing attack perturbs a broccoli and causes YOLOv3 to no longer detect the targeted broccoli (no purple detections are seen around the target).

(b) A mislabeling attack perturbs a chair and causes SSD to mislabel the targeted stop sign to the intended class—a bear—with 1.00 confidence.



(c) An untargeted attack perturbs a couch and causes Faster R-CNN to miss correctly predicted objects (like "tv" and "person") and make up objects (like "bird" and "chair").

Figure 1. **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 yellow and those on the attacked images are in purple, 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 12.

to improve performance, either implicitly through end-to-end training (e.g. YOLO Redmon et al., 2015) or explicitly through architectural design (Tong et al., 2020, Section 2.4).

We implement intent obfuscating attacks on object detectors using the Targeted Objectness Gradient (TOG) algorithm (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—the two most common types of object detectors (Zhao et al., 2019; Zou et al., 2019)—on the large-scale Microsoft Common Objects in Context (COCO) dataset (Lin et al., 2014). In sum, we contribute to the important and understudied issue of intent obfuscation in adversarial machine learning:

1. We are the first to propose intent obfuscating attacks 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 (Section 5).
4. We implement intent obfuscating attacks by perturbing arbitrary non-overlapping regions—rather than actual objects—to disrupt the target (Section 6).
5. We demonstrate that perturbing an arbitrary region allows us to easily manipulate two validated success factors—perturb sizes and perturb-target distances—to greatly increase success on all models and attacks (Section 6.3).

2. Related Work

Intent obfuscation: Intent obfuscation is rare in the machine learning literature. One exception is a widely-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 im-

prove existing attacks or to design new attacks (Hu et al., 2021; Saha et al., 2020; Lee & Kolter, 2019; Liu et al., 2018; Zhang et al., 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 and 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 et al., 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:

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

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:

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

The optimization loss L depends on the model, which we will present in the next section. For illustration purposes, we will conduct experiments using y as the ground-truth.

3.2. Model Losses

We attack 5 prominent detection models—comprising 3 one-stage detectors (SSD, YOLOv3, and RetinaNet) and 2 two-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 widely-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. Table 1 summarizes the 5 detection models and corresponding attack losses. Full details are given in Appendix B.

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). 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

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

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

			Attack Losses[c]		
			Targeted		
Detectors	Stages[a]	COCO mAP[b]	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.

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 (refer as "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 the original range $[0, 1]$ after every iteration. Since the resulting success rates are not at ceiling, we did not try a more restrictive bound. For every model, attack and iteration combination, we resampled 5,000 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. For targeted attacks, we do not restrict success to the intended attack mode (e.g. a mislabeling attack which causes the target object to vanish is still considered as success) because it may not concern the attacker. Nevertheless, as shown in Figure 5, vanishing and mislabeling attacks do cause the target objects to vanish and mislabel respectively in most success cases. In addition, mislabeling attacks usually mislabel the target object to the intended class (Figure 6). More experimental details are included in Appendix C.1.

5. Hypotheses and Results

We conducted a thorough analysis by listing 10 hypotheses increasing success rates and systematically testing whether those hypotheses are valid. Figure 2 graphs the success rates. The hypotheses and results are summarized in Table 2 and explained in the appendices C.2 and C.3 respectively. Attacked images are illustrated in Figure 1.

6. 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 Section 5. 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 targeted attack if most self-driving cars

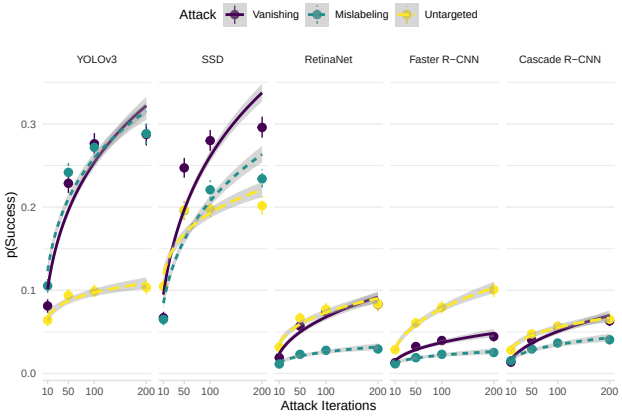


Figure 2. **Intent obfuscating attack is feasible for all models and attacks:** We conduct a randomized experiment by resampling COCO images, and within those images randomly sampling correctly predicted target and perturb objects. Then we distort the perturb objects to disrupt the target objects varying 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 5,000 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. Additionally, targeted attacks are significantly more successful than untargeted attacks on YOLOv3 and SSD, but the pattern does not exist for RetinaNet, Faster R-CNN, and Cascade R-CNN. Within targeted attacks, vanishing achieves significantly greater success than mislabeling attack on all models except YOLOv3. 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.

use a 1-stage detector (Result 2). He could also increase success by deliberately perturbing larger objects (Result 6) closer to the target (Result 7). Indeed, he can easily multiply success on a random target for any detector by perturbing a large *arbitrary* region close to the target object, as we demonstrate below.

6.1. 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 (Figure 13). We vary the length of the square perturb region and the distance between the target and perturb bounding boxes to be 10, 50, 100, or 200 pixels (in original image dimensions) and test all combinations. For every combination, we resample 200 COCO test images and run the 3 attacks all for 200 iterations (more details in Appendix D.1).

Table 2. Hypothesis testing in the randomized attack (Section 5)

Hypotheses (higher success for)	Accepted (across attacks and models)[a]
1-stage > 2-stage models (YOLOv3, SSD, RetinaNet > Faster R-CNN, Cascade R-CNN)	All except RetinaNet (YOLOv3, SSD > RetinaNet, Faster R-CNN, Cascade R-CNN)
Targeted > Untargeted attack	Only YOLOv3, SSD
Vanishing > Mislabeling attack	All except YOLOv3
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
More probable intended class (mislabeling attack only)	None except RetinaNet
Lower target IOU[b] (untargeted attack only)	All

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

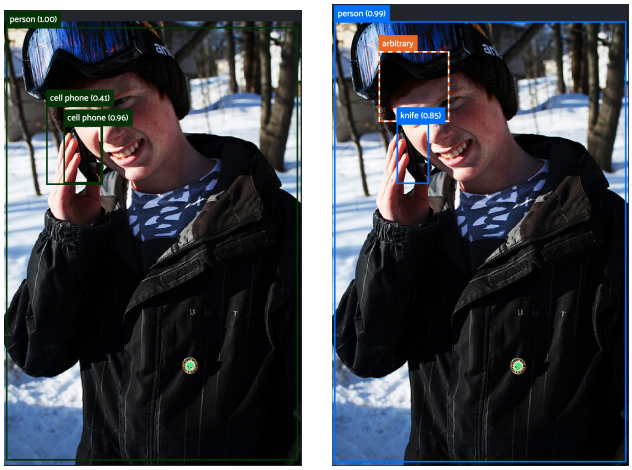
6.2. Hypotheses

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.

6.3. Results

Success rates greatly increase compared to the randomized attack (Figure 4): in the extreme at perturb lengths more than 200 pixels and perturb-target distances less than 10 pixels, the attacker obtains for the vanishing attack nearly 100% success rates on YOLOv3 and SSD and more than 50% on RetinaNet, Faster R-CNN and Cascade R-CNN. A success example is illustrated in Figure 3.

Hypothesis testing is similar to the procedure in the random-



(a) Original predictions (green)

(b) Adversarial predictions (blue) and non-overlapping arbitrary perturb region (orange dotted)

Figure 3. **We can implement intent obfuscating attack via perturbing an arbitrary region rather than an actual object:** A mislabeling attack perturbs a non-overlapping arbitrary region (b) and causes Cascade R-CNN to mislabel a cell phone (a) to the intended class—a knife (b).

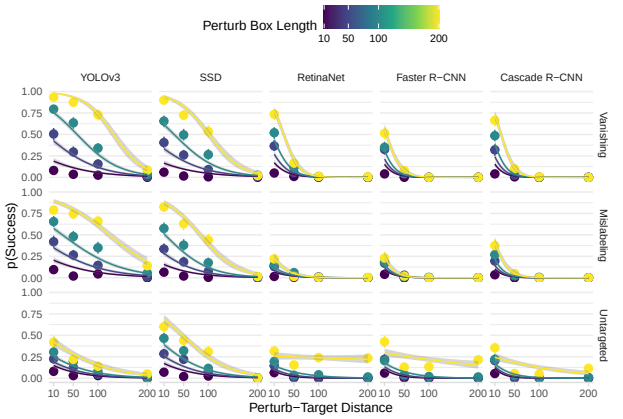


Figure 4. **A deliberate attack obfuscates intent with increased success for all models and attacks:** We implement intent obfuscating attack by perturbing an arbitrary non-overlapping square region to disrupt a randomly selected target object at various lengths and distances. The binned summaries and regression trendlines graph success proportion against perturb-target distance and perturb box length 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 2), especially at close perturb-target distance and large perturb box length. Full details are given in Section 6.

ized experiment (Section 5): A logistic regression model using both terms as predictors show that longer perturb lengths or shorter perturb-target distances cause success rates to increase significantly for all model and attack combinations, except for perturb lengths in the untargeted attack on Cascade R-CNN. The interaction terms, even when significant, are negligibly close to 0. Statistics are given in Table 11.

7. 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 (Table 12). 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 attack in both settings. Interested readers can turn to Cai et al. (2021) for black-box contextual attacks and Lee & Kolter (2019) for physical contextual attacks.

Broader impact: A malicious actor can use an intent obfuscating attack to disrupt AI systems and claim plausible deniability. There are known defenses against the attack: we implement intent obfuscation by perturbing contextual regions using gradient-based signals, making it susceptible to context-based defenses (Saha et al., 2020; Li et al., 2020; Yin et al., 2021a) and general defense techniques reviewed by Ren et al. (2020); Xu et al. (2020) like gradient obfuscation and adversarial detection. Not surprisingly, the adversarial machine learning literature is constantly evolving, with more sophisticated attacks continuously invented to overcome these defenses (Cai et al., 2022; Yin et al., 2021b). In contrast, legal protection against intent obfuscating attack is scant. 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). 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. 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. Cascade R-CNN: Delving into high quality object detection. pp. 6154–6162, December 2017.

Cai, Z., Xie, X., Li, S., Yin, M., Song, C., Krishnamurthy, S. V., Roy-Chowdhury, A. K., and Salman Asif, M. Context-Aware transfer attacks for object detection. December 2021.

Cai, Z., Rane, S., Brito, A. E., Song, C., Krishnamurthy, S. V., Roy-Chowdhury, A. K., and Asif, M. S. Zero-query transfer attacks on context-aware object detectors. pp. 15024–15034, March 2022.

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. MMDetection: Open MMLab detection toolbox and benchmark. June 2019.

Chow, K.-H., Liu, L., Gursoy, M. E., Truex, S., Wei, W., and Wu, Y. Understanding object detection through an adversarial lens. In *Computer Security – ESORICS 2020*, pp. 460–481. Springer International Publishing, 2020a.

Chow, K.-H., Liu, L., Loper, M., Bae, J., Gursoy, M. E., Truex, S., Wei, W., and Wu, Y. 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)*, pp. 263–272. ieeexplore.ieee.org, October 2020b.

COCO. COCO detection evaluation metrics. `https://cocodataset.org/#detection-eval`. Accessed: 2022-7-12.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. Robust physical-world attacks on deep learning visual classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1625–1634. IEEE, June 2018.

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

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. December 2014.

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

Hu, S., Zhang, Y., Laha, S., Sharma, A., and Foroosh, H. CCA: Exploring the possibility of contextual camouflage attack on object detection. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 7647–7654. ieeexplore.ieee.org, January 2021.

Kumar, R. S. S., O'Brien, D. R., Albert, K., and Vilojen, S. Law and adversarial machine learning. October 2018.

Kumar, R. S. S., Penney, J., Schneier, B., and Albert, K. Legal risks of adversarial machine learning research. June 2020.

Larmarange, J. and Sjoberg, D. D. *broom.helpers: Helpers for Model Coefficients Tibbles*, 2023. URL `https://larmarange.github.io/broom.helpers/`. R package version 1.11.0.

Lee, M. and Kolter, Z. On physical adversarial patches for object detection. June 2019.

Li, S., Zhu, S., Paul, S., Roy-Chowdhury, A., Song, C., Krishnamurthy, S., Swami, A., and Chan, K. S. Connecting the dots: Detecting adversarial perturbations using context inconsistency. July 2020.

LIFARS. What is obfuscation in security and what types of obfuscation are there? `https://www.lifars.com/2020/11/what-is-obfuscation-in-security/`, November 2020. Accessed: 2022-6-7.

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

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

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. August 2017b.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. SSD: Single shot MultiBox detector. December 2015.

Liu, X., Yang, H., Liu, Z., Song, L., Li, H., and Chen, Y. DPatch: An adversarial patch attack on object detectors. June 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. June 2017.

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

Papers With Code. Object detection. https://paperswithcode.com/task/object-detection. Accessed: 2023-1-24.

Papers with Code. COCO dataset. https://paperswithcode.com/dataset/coco. Accessed: 2022-7-4.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2022. URL https://www.R-project.org/.

Redmon, J. and Farhadi, A. YOLOv3: An incremental improvement. April 2018.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, Real-Time object detection. June 2015.

Ren, K., Zheng, T., Qin, Z., and Liu, X. Adversarial attacks and defenses in deep learning. *Proc. Est. Acad. Sci. Eng.*, 6(3):346–360, March 2020.

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

Robinson, D., Hayes, A., and Couch, S. *broom: Convert Statistical Objects into Tidy Tibbles*, 2022. URL https://CRAN.R-project.org/package=broom. R package version 1.0.2.

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

Sharif, M., Bhagavatula, S., Bauer, L., and Reiter, M. K. 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, pp. 1528–1540, New York, NY, USA, October 2016. Association for Computing Machinery.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for Large-Scale image recognition. September 2014.

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

Wex Definitions Team. intent. https://www.law.cornell.edu/wex/intent. Accessed: 2023-1-22.

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

Xie, Y. *knitr: A General-Purpose Package for Dynamic Report Generation in R*, 2022. URL https://yihui.org/knitr/. R package version 1.41.

Xu, H., Ma, Y., Liu, H.-C., Deb, D., Liu, H., Tang, J.-L., and Jain, A. K. Adversarial attacks and defenses in images, graphs and text: A review. *Int. J. Autom. Comput.*, 17(2): 151–178, April 2020.

Yin, M., Li, S., Cai, Z., Song, C., Asif, M. S., Roy-Chowdhury, A. K., and Krishnamurthy, S. V. Exploiting Multi-Object relationships for detecting adversarial attacks in complex scenes. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7838–7847. IEEE, October 2021a.

Yin, M., Li, S., Song, C., Salman Asif, M., Roy-Chowdhury, A. K., and Krishnamurthy, S. V. ADC: Adversarial attacks against object detection that evade context consistency checks. October 2021b.

Zhang, H., Zhou, W., and Li, H. Contextual adversarial attacks for object detection. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6. ieeexplore.ieee.org, July 2020.

Zhao, Z.-Q., Zheng, P., Xu, S.-T., and Wu, X. Object detection with deep learning: A review. *IEEE Trans Neural Netw Learn Syst*, 30(11):3212–3232, November 2019.

Zhu, H. *kableExtra: Construct Complex Table with kable and Pipe Syntax*, 2023. http://haozhu233.github.io/kableExtra/.

Zou, Z., Chen, K., Shi, Z., Guo, Y., and Ye, J. Object detection in 20 years: A survey. May 2019.

A. Anonymized Code and Data

The code is available on the anonymized github repository `https://github.com/ano938796412/icml-review`. The included README.md contains instructions to reproduce graphs and tables, download datasets and images, visualize attacked datasets or replicate experiments. The datasets and images in both experiments are stored on an anonymized Google Cloud Storage `https://console.cloud.google.com/storage/browser/icml-review` (one may still need to sign in with a google account simply to access the public bucket). More instructions are contained in the README.md.

B. Model Losses

YOLOv3: YOLOv3 (Redmon & 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 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 608×608. The model achieves 33.7 COCO mean average precision (mAP), the primary metric in the COCO challenge (COCO).

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 & Zisserman, 2014) and input size 512×512. 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 (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 & 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.

C. Randomized Attack

C.1. Setup

Since we are using a shared computing resource on an internal network, we split the attack into 50 repetitions and attacked 100 images per repetition. The images are randomly sampled without replacement within repetitions, but may repeat across repetitions. Every repetition takes 60-90 minutes on a 32GB NVIDIA Tesla V100 GPU. 750 repetitions (5 models * 3 attacks * 50 repetitions) take 30-45 V100 GPU days.

Across model, attack and iteration combinations, we sample the same images and select the same target and perturb objects per image to more accurately compare the success rates between combinations. In addition, the MMdetection models backpropagate only in training mode. Hence, we set the model to training mode in the TOG attack to backpropagate the gradients. Since the model evaluates the adversarial images in testing mode, we reset the model after every iteration to prevent updates to its weights or running statistics, to ensure the gradients are directed towards the model in testing mode. Also, we do not use data augmentation in the TOG attack, since the adversarial images are not augmented during evaluation.

C.2. Hypotheses

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. **Intended classes with higher probabilities for the mislabeling attack:** in 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 less. 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

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

latter.

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

C.3. Results

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 include model (Hypothesis 1) or attack (Hypotheses 2 and 3). Except for Hypothesis 4, we restrict the data to the maximum 200 attack iterations to analyze the strongest possible results. We computed the p-values using Wald z-test and set the significance level (α) to the usual 0.05. We will state the conclusions below and report the statistics in Appendix C.4. Attacked images are illustrated in Figure 1. Hypotheses and results are summarized in Table 2.

1. **1-stage (YOLOv3, SSD, and RetinaNet) than 2-stage (Faster R-CNN and Cascade R-CNN) detectors:** As shown in Figure 2, the success rates are significantly higher for 1-stage than 2-stage detectors, especially for vanishing and mislabeling attacks. 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 SSD, but the increase is non-existent or reversed for RetinaNet, Faster R-CNN and Cascade R-CNN (Table 4 and Figure 2). 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, except for YOLOv3 in which the two are similar (Table 4 and Figure 2).
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 7).
6. **Perturb objects with larger bounding boxes:** Larger perturb objects significantly increase success rates for all models and attacks, except for mislabeling attack on Faster R-CNN, after controlling for perturb-target distances (Table 7 and Figure 8).
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 8).
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 9).
9. **Intended classes with higher probabilities for the mislabeling attack:** Higher intended class probability (log-transformed) does *not* predict success rates for mislabeling attack after controlling for target class confidence, except for RetinaNet. (Table 9 and Figure 10).
10. **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 11).

C.4. Statistics

C.4.1. Table Headers

We generate the graphs and tables in R (R Core Team, 2022). The upper table headers are generated using R knitr (Xie, 2022) and kableExtra (Zhu, 2023): We run one regression per group (model and/or attack combination). The terms with a blank row and 0.000 estimate are reference variables in the regression model, e.g. YOLOv3 in Table 3. The lower regression headers are generated using R broom (Robinson et al., 2022) and broom.helpers (Larmarange & Sjoberg, 2023). To adapt the broom documentation:

mean probability only for the predicted class.

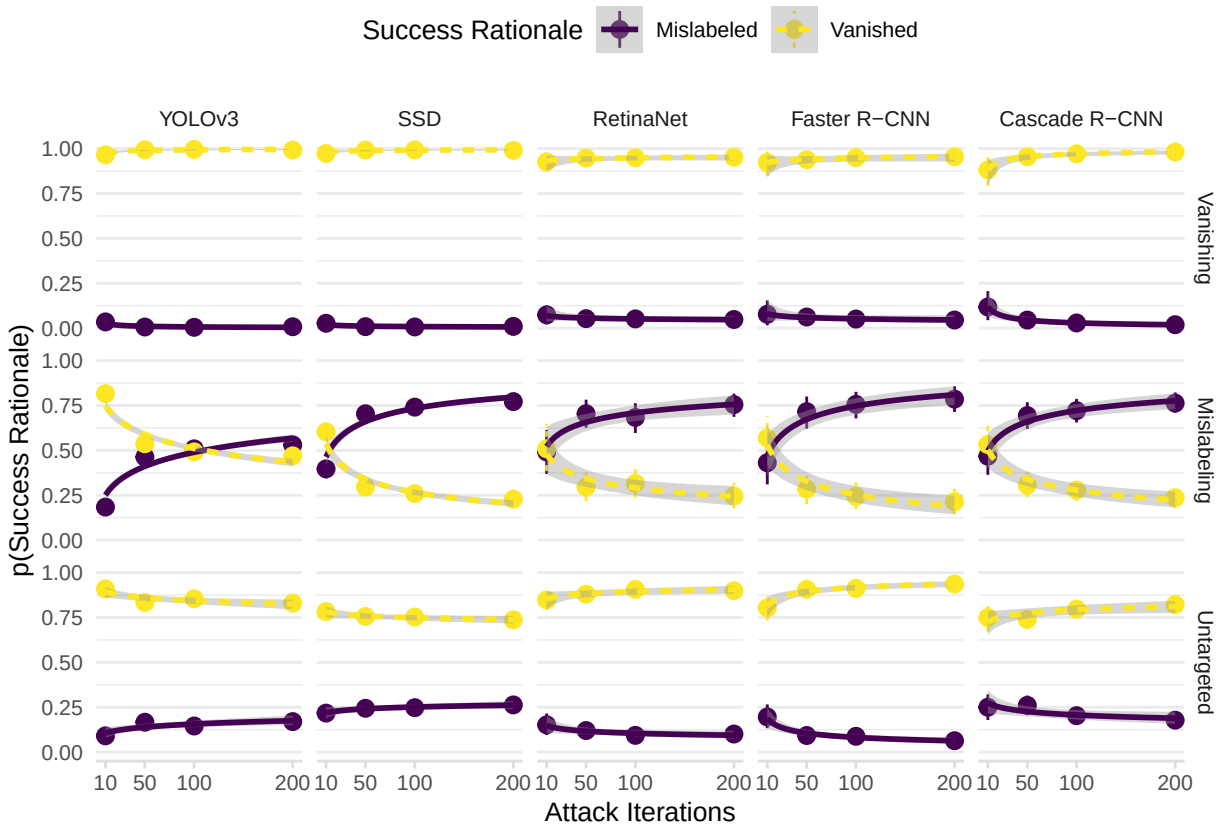


Figure 5. **Vanishing and mislabeling attacks mostly cause target objects to vanish and get mislabeled:** The graph breaks down the success rationale within the success cases (Figure 2). Though we did not restrict success to the intended attack mode (e.g. a vanishing attack which mislabels the target object still count as a success case), the target objects do vanish and get mislabeled in most success cases respectively in the vanishing and mislabeling attacks. The binned summaries and regression trendlines break down the success cases into proportion vanished and mislabeled—separated by attack—against attack iterations in the randomized attack experiment. Errors are 95% confidence intervals.

term The name of the regression term.

sig Terms which are significant ($p < .05$) are denoted by "*".

estimate The estimated value of the regression term.

std.error The standard error of the regression term.

statistic The value of a Wald z-statistic to use in a hypothesis that the regression term is non-zero.

p.value The two-sided p-value associated with the observed statistic.

conf.low Lower bound on the 95% confidence interval for the estimate.

conf.high Upper bound on the 95% confidence interval for the estimate.

Table 3: We run a logistic model regressing success against detection models, split by attack, in the randomized attack experiment. All attacks, especially vanishing and mislabeling, obtain higher success on 1-stage (YOLOv3, SSD) than 2-stage (Faster R-CNN, Cascade R-CNN) detectors. However, the 1-stage RetinaNet is as resilient as 2-stage detectors. Table headers are explained in Appendix C.4.1.

Group	Regression

Attack	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
Vanishing	YOLOv3		0.000					
	SSD		0.041	0.044	0.924	0.355	-0.046	0.127
	RetinaNet	*	-1.492	0.060	-24.865	0.000	-1.610	-1.375
	Faster R-CNN	*	-2.161	0.075	-28.651	0.000	-2.311	-2.015
	Cascade R-CNN	*	-1.788	0.066	-27.097	0.000	-1.919	-1.660
Mislabeling	YOLOv3		0.000					
	SSD	*	-0.283	0.046	-6.183	0.000	-0.372	-0.193
	RetinaNet	*	-2.594	0.089	-29.029	0.000	-2.773	-2.422
	Faster R-CNN	*	-2.752	0.095	-28.826	0.000	-2.944	-2.569
	Cascade R-CNN	*	-2.259	0.078	-28.907	0.000	-2.415	-2.109
Untargeted	YOLOv3		0.000					
	SSD	*	0.782	0.058	13.411	0.000	0.668	0.896
	RetinaNet	*	-0.239	0.069	-3.463	0.001	-0.375	-0.104
	Faster R-CNN		-0.031	0.066	-0.462	0.644	-0.160	0.099
	Cascade R-CNN	*	-0.505	0.074	-6.850	0.000	-0.650	-0.361

Table 4: We run a logistic model regressing success against attacks, split by detection models in the randomized attack experiment. Targeted attacks achieve higher success than untargeted attack on YOLOv3 and SSD. Within targeted attacks, vanishing attacks achieve higher success than mislabeling attack, except on YOLOv3. Table headers are explained in Appendix C.4.1.

Group	Regression							
Model	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
YOLOv3	Vanishing		0.000					
	Mislabeling		0.005	0.044	0.110	0.912	-0.082	0.091
	Untargeted	*	-1.250	0.056	-22.340	0.000	-1.360	-1.141
SSD	Vanishing		0.000					
	Mislabeling	*	-0.318	0.046	-6.990	0.000	-0.408	-0.229
	Untargeted	*	-0.509	0.047	-10.844	0.000	-0.601	-0.417
RetinaNet	Vanishing		0.000					
	Mislabeling	*	-1.097	0.098	-11.181	0.000	-1.292	-0.907
	Untargeted		0.003	0.072	0.036	0.971	-0.139	0.145
Faster R-CNN	Vanishing		0.000					
	Mislabeling	*	-0.586	0.113	-5.171	0.000	-0.811	-0.366
	Untargeted	*	0.881	0.083	10.587	0.000	0.719	1.045
Cascade R-CNN	Vanishing		0.000					
	Mislabeling	*	-0.466	0.092	-5.056	0.000	-0.648	-0.287
	Untargeted		0.033	0.082	0.408	0.683	-0.127	0.193

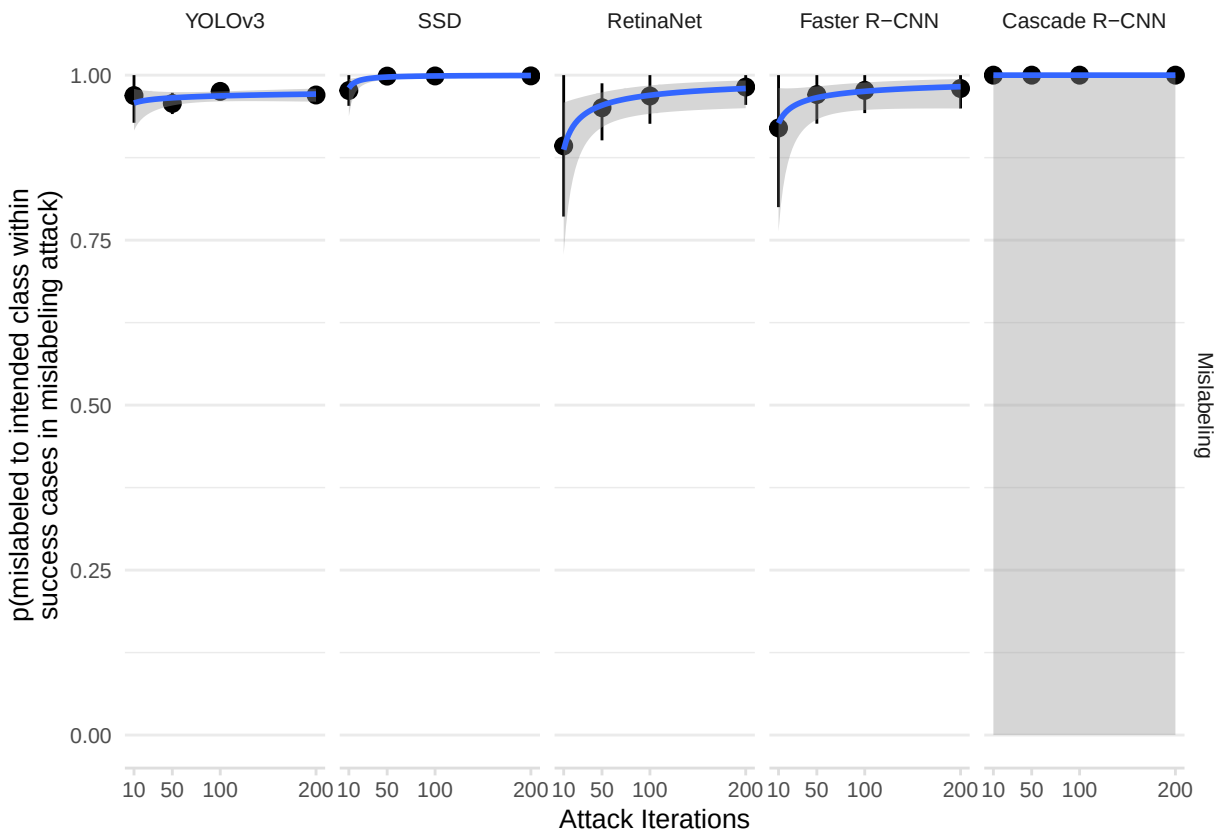


Figure 6. **Mislabeling attacks usually mislabel the target objects to the intended class:** The binned summaries and regression trendlines give us the proportion mislabeled to the intended class within the success cases in the mislabeling attack. The proportion is plotted against attack iterations in the randomized attack experiment. Errors are 95% confidence intervals. For Cascade R-CNN, the logistic model did not converge because the mislabel intended proportion is constant at 100%.

Table 5: We run a logistic model regressing success against log(attack iterations) in the randomized attack experiment. Success rates increase with attack iterations for all models and attacks. Table headers are explained in Appendix C.4.1.

Group	Regression							
Attack	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
YOLOv3								
Vanishing	log(iterations)	*	0.480	0.018	26.729	0	0.445	0.515
Mislabeling	log(iterations)	*	0.397	0.017	23.267	0	0.363	0.430
Untargeted	log(iterations)	*	0.174	0.023	7.404	0	0.128	0.220
SSD								
Vanishing	log(iterations)	*	0.528	0.018	29.009	0	0.493	0.564
Mislabeling	log(iterations)	*	0.452	0.019	23.386	0	0.414	0.490
Untargeted	log(iterations)	*	0.246	0.018	13.735	0	0.211	0.281
RetinaNet								
Vanishing	log(iterations)	*	0.475	0.033	14.339	0	0.411	0.541
Mislabeling	log(iterations)	*	0.312	0.048	6.489	0	0.219	0.408
Untargeted	log(iterations)	*	0.328	0.029	11.206	0	0.271	0.385
Faster R-CNN								

Vanishing	log(iterations)	*	0.394	0.042	9.316	0	0.313	0.479
Mislabeling	log(iterations)	*	0.264	0.051	5.204	0	0.166	0.364
Untargeted	log(iterations)	*	0.440	0.030	14.511	0	0.381	0.500
Cascade R-CNN								
Vanishing	log(iterations)	*	0.495	0.039	12.772	0	0.420	0.572
Mislabeling	log(iterations)	*	0.327	0.042	7.758	0	0.245	0.410
Untargeted	log(iterations)	*	0.291	0.033	8.886	0	0.228	0.356

Table 6: We run a logistic model regressing success against target confidence in the randomized attack experiment. Lower target confidence significantly increases success rates for all models and attacks. Table headers are explained in Appendix C.4.1.

Group	Regression							
Attack	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
YOLOv3								
Vanishing	confidence	*	-0.618	0.141	-4.375	0.000	-0.895	-0.341
Mislabeling	confidence	*	-1.924	0.142	-13.520	0.000	-2.203	-1.645
Untargeted	confidence	*	-3.417	0.215	-15.919	0.000	-3.841	-2.999
SSD								
Vanishing	confidence	*	-0.428	0.130	-3.288	0.001	-0.684	-0.173
Mislabeling	confidence	*	-1.144	0.140	-8.199	0.000	-1.418	-0.871
Untargeted	confidence	*	-2.024	0.149	-13.602	0.000	-2.317	-1.733
RetinaNet								
Vanishing	confidence	*	-2.762	0.278	-9.925	0.000	-3.314	-2.222
Mislabeling	confidence	*	-5.951	0.595	-10.002	0.000	-7.162	-4.826
Untargeted	confidence	*	-5.002	0.328	-15.238	0.000	-5.657	-4.370
Faster R-CNN								
Vanishing	confidence	*	-2.814	0.290	-9.706	0.000	-3.382	-2.244
Mislabeling	confidence	*	-3.927	0.382	-10.290	0.000	-4.683	-3.184
Untargeted	confidence	*	-3.578	0.207	-17.308	0.000	-3.985	-3.174
Cascade R-CNN								
Vanishing	confidence	*	-1.690	0.259	-6.533	0.000	-2.194	-1.179
Mislabeling	confidence	*	-3.329	0.305	-10.928	0.000	-3.928	-2.732
Untargeted	confidence	*	-3.927	0.250	-15.679	0.000	-4.421	-3.438

Table 7: We run a logistic model regressing success against perturb-target distance (100 pixels) and perturb box size (100,000 squared pixels) in the randomized attack experiment. Larger perturb objects significantly increase success rates for all models and attacks, except for mislabeling attack on Faster R-CNN, after controlling for perturb-target distances; shorter perturb-target distances significantly increase success rates for all models and attacks, after controlling for perturb object sizes. Table headers are explained in Appendix C.4.1.

Group	Regression							
Attack	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
YOLOv3								

	Vanishing	distance	*	-1.903	0.102	-18.601	0.000	-2.107	-1.706
		size	*	6.436	0.407	15.804	0.000	5.656	7.252
		distance * size	*	-2.167	0.344	-6.297	0.000	-2.853	-1.502
	Mislabeling	distance	*	-1.706	0.087	-19.719	0.000	-1.879	-1.540
		size	*	3.182	0.270	11.791	0.000	2.667	3.724
		distance * size		-0.384	0.252	-1.523	0.128	-0.886	0.102
	Untargeted	distance	*	-2.191	0.160	-13.656	0.000	-2.515	-1.886
		size	*	1.357	0.182	7.470	0.000	1.007	1.720
		distance * size		0.444	0.287	1.547	0.122	-0.138	0.992
SSD									
	Vanishing	distance	*	-2.264	0.112	-20.125	0.000	-2.488	-2.047
		size	*	3.896	0.313	12.467	0.000	3.299	4.524
		distance * size	*	-0.978	0.306	-3.194	0.001	-1.594	-0.389
	Mislabeling	distance	*	-2.203	0.121	-18.194	0.000	-2.445	-1.970
		size	*	2.787	0.252	11.061	0.000	2.306	3.295
		distance * size	*	-0.640	0.295	-2.172	0.030	-1.238	-0.079
	Untargeted	distance	*	-2.299	0.124	-18.514	0.000	-2.547	-2.060
		size	*	1.283	0.189	6.805	0.000	0.922	1.662
		distance * size		0.164	0.263	0.623	0.533	-0.368	0.666
RetinaNet									
	Vanishing	distance	*	-5.130	0.374	-13.709	0.000	-5.886	-4.420
		size	*	1.912	0.249	7.695	0.000	1.440	2.415
		distance * size		0.152	0.588	0.259	0.795	-1.040	1.266
	Mislabeling	distance	*	-4.411	0.525	-8.405	0.000	-5.494	-3.437
		size	*	0.920	0.248	3.703	0.000	0.435	1.410
		distance * size		0.693	0.759	0.913	0.361	-0.872	2.103
	Untargeted	distance	*	-1.555	0.151	-10.285	0.000	-1.862	-1.270
		size	*	1.377	0.174	7.927	0.000	1.039	1.720
		distance * size	*	1.683	0.230	7.315	0.000	1.240	2.143
Faster R-CNN									
	Vanishing	distance	*	-6.113	0.604	-10.114	0.000	-7.351	-4.982
		size	*	1.638	0.275	5.964	0.000	1.114	2.192
		distance * size		-0.610	0.997	-0.611	0.541	-2.674	1.236
	Mislabeling	distance	*	-5.569	0.630	-8.839	0.000	-6.870	-4.398
		size		0.238	0.275	0.867	0.386	-0.301	0.780
		distance * size	*	2.237	0.767	2.918	0.004	0.578	3.597
	Untargeted	distance	*	-1.925	0.168	-11.451	0.000	-2.267	-1.607
		size	*	1.880	0.194	9.677	0.000	1.504	2.265
		distance * size	*	2.143	0.259	8.262	0.000	1.648	2.665
Cascade R-CNN									
	Vanishing	distance	*	-6.971	0.573	-12.163	0.000	-8.137	-5.890
		size	*	2.167	0.282	7.693	0.000	1.635	2.741

	distance * size		-0.329	0.883	-0.372	0.710	-2.161	1.309
Mislabeling	distance	*	-6.440	0.585	-10.999	0.000	-7.639	-5.343
	size	*	0.486	0.237	2.049	0.040	0.023	0.955
	distance * size	*	1.829	0.798	2.292	0.022	0.144	3.280
Untargeted	distance	*	-2.677	0.224	-11.971	0.000	-3.132	-2.255
	size	*	0.693	0.174	3.991	0.000	0.352	1.034
	distance * size	*	2.181	0.258	8.442	0.000	1.678	2.693

Table 8: We run a logistic model regressing success against mean COCO accuracy for the target class, with target confidence as covariate, in the randomized attack experiment. The results are mixed after controlling for target class confidence and the relatively large interaction terms make interpretation challenging. Table headers are explained in Appendix C.4.1.

Group	Regression							
Attack	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
YOLOv3								
Vanishing	accuracy		-0.436	1.053	-0.414	0.679	-2.497	1.634
	confidence		0.475	1.145	0.415	0.678	-1.771	2.722
	accuracy * confidence		-1.233	1.414	-0.873	0.383	-4.007	1.538
Mislabeling	accuracy		0.159	1.034	0.154	0.877	-1.871	2.186
	confidence		0.538	1.147	0.469	0.639	-1.716	2.782
	accuracy * confidence	*	-2.902	1.418	-2.047	0.041	-5.679	-0.118
Untargeted	accuracy	*	-3.282	1.330	-2.468	0.014	-5.906	-0.689
	confidence	*	-4.205	1.659	-2.535	0.011	-7.517	-1.009
	accuracy * confidence		1.199	2.069	0.580	0.562	-2.799	5.314
SSD								
Vanishing	accuracy		-0.743	0.817	-0.909	0.363	-2.342	0.864
	confidence		-0.042	0.869	-0.048	0.962	-1.746	1.664
	accuracy * confidence		-0.396	1.089	-0.363	0.716	-2.532	1.740
Mislabeling	accuracy		-0.797	0.842	-0.946	0.344	-2.444	0.857
	confidence		-0.277	0.910	-0.305	0.761	-2.064	1.504
	accuracy * confidence		-0.977	1.145	-0.853	0.394	-3.221	1.271
Untargeted	accuracy	*	-2.087	0.867	-2.408	0.016	-3.789	-0.389
	confidence	*	-3.125	0.990	-3.157	0.002	-5.081	-1.198
	accuracy * confidence		1.486	1.241	1.198	0.231	-0.933	3.932
RetinaNet								
Vanishing	accuracy	*	-3.520	1.630	-2.160	0.031	-6.700	-0.309
	confidence	*	-6.644	2.646	-2.511	0.012	-11.879	-1.504
	accuracy * confidence		4.770	3.090	1.544	0.123	-1.259	10.858
Mislabeling	accuracy	*	-7.902	2.929	-2.697	0.007	-13.606	-2.128
	confidence	*	-20.491	5.908	-3.469	0.001	-32.313	-9.179
	accuracy * confidence	*	17.153	6.788	2.527	0.012	4.011	30.592

	Untargeted	accuracy	*	-4.352	1.707	-2.549	0.011	-7.701	-1.007
		confidence	*	-9.046	2.985	-3.030	0.002	-14.984	-3.279
		accuracy * confidence		5.142	3.520	1.461	0.144	-1.699	12.099
Faster R-CNN									
	Vanishing	accuracy	*	4.935	2.132	2.315	0.021	0.839	9.204
		confidence	*	4.666	2.288	2.040	0.041	0.200	9.180
		accuracy * confidence	*	-9.142	2.824	-3.238	0.001	-14.707	-3.627
	Mislabeling	accuracy	*	8.515	2.767	3.078	0.002	3.222	14.084
		confidence	*	6.412	3.183	2.015	0.044	0.114	12.625
		accuracy * confidence	*	-12.713	3.888	-3.270	0.001	-20.304	-5.038
	Untargeted	accuracy		1.447	1.442	1.003	0.316	-1.373	4.289
		confidence		0.733	1.588	0.462	0.644	-2.395	3.836
		accuracy * confidence	*	-5.146	1.969	-2.614	0.009	-8.999	-1.273
Cascade R-CNN									
	Vanishing	accuracy		1.644	2.104	0.781	0.435	-2.419	5.840
		confidence		0.766	2.173	0.353	0.724	-3.466	5.064
		accuracy * confidence		-2.987	2.679	-1.115	0.265	-8.273	2.237
	Mislabeling	accuracy	*	4.762	2.347	2.029	0.042	0.236	9.446
		confidence		1.915	2.651	0.722	0.470	-3.301	7.107
		accuracy * confidence	*	-6.491	3.243	-2.002	0.045	-12.843	-0.119
	Untargeted	accuracy	*	3.752	1.805	2.079	0.038	0.241	7.324
		confidence		1.669	2.033	0.821	0.412	-2.339	5.637
		accuracy * confidence	*	-6.887	2.519	-2.734	0.006	-11.812	-1.930

Table 9: We run a logistic model regressing success against log(intended class probability) for the mislabeling attack, with predicted class's confidence as covariate, in the randomized attack experiment. Intended class probability does not predict success rates after controlling for target class confidence, except for RetinaNet. Table headers are explained in Appendix C.4.1.

Group	Regression							
Model	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
Mislabeling								
YOLOv3	log(probability)		0.052	0.036	1.425	0.154	-0.019	0.123
	confidence	*	-1.777	0.431	-4.125	0.000	-2.624	-0.935
	log(probability) * confidence		0.010	0.050	0.193	0.847	-0.089	0.108
SSD	log(probability)		0.034	0.042	0.791	0.429	-0.049	0.117
	confidence	*	-0.760	0.375	-2.026	0.043	-1.494	-0.023
	log(probability) * confidence		0.025	0.054	0.473	0.636	-0.080	0.131
RetinaNet	log(probability)	*	0.626	0.310	2.018	0.044	0.003	1.218
	confidence	*	-8.462	1.804	-4.691	0.000	-12.016	-4.950
	log(probability) * confidence		-1.016	0.648	-1.568	0.117	-2.242	0.295
Faster R-CNN	log(probability)		0.060	0.100	0.593	0.553	-0.137	0.257

	confidence	*	-3.022	0.925	-3.265	0.001	-4.835	-1.201
	log(probability) * confidence		0.076	0.147	0.521	0.603	-0.207	0.368
Cascade R-CNN	log(probability)		0.117	0.080	1.474	0.140	-0.039	0.274
	confidence	*	-2.872	0.722	-3.979	0.000	-4.283	-1.450
	log(probability) * confidence		-0.014	0.108	-0.130	0.897	-0.224	0.200

Table 10: We run a logistic model regressing success against target IOU for the untargeted attack in the randomized attack experiment. Target IOU for the untargeted attack increases success rates on all models. Table headers are explained in Appendix C.4.1.

Group	Regression							
Model	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
Untargeted								
YOLOv3	iou	*	-2.199	0.278	-7.904	0	-2.741	-1.650
SSD	iou	*	-1.704	0.226	-7.549	0	-2.146	-1.260
RetinaNet	iou	*	-2.344	0.275	-8.533	0	-2.880	-1.802
Faster R-CNN	iou	*	-1.961	0.267	-7.340	0	-2.481	-1.433
Cascade R-CNN	iou	*	-2.122	0.307	-6.902	0	-2.718	-1.512

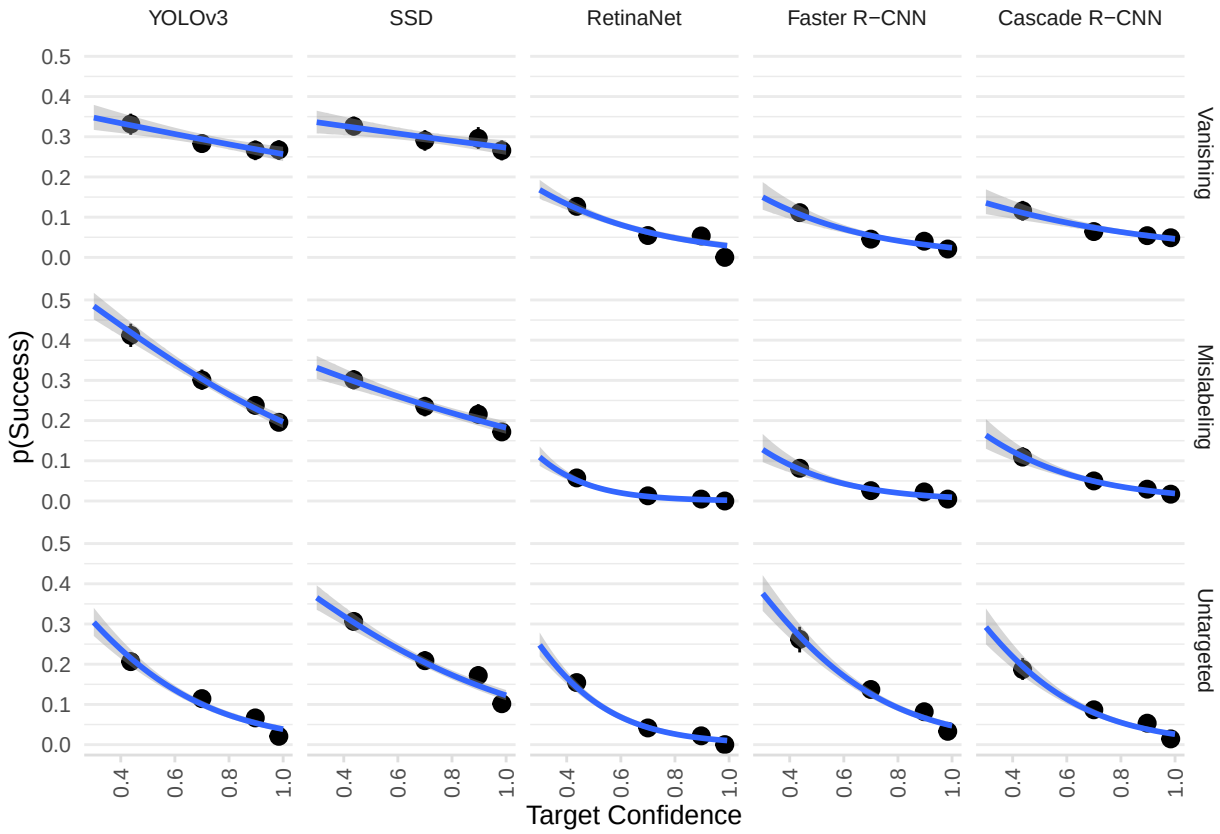


Figure 7. **Lower target confidence significantly increases success rates for all models and attacks:** The binned summaries and regression trendlines graph success proportion against target confidence in the randomized attack experiment. Bins are split into quantiles. Errors are 95% confidence intervals.

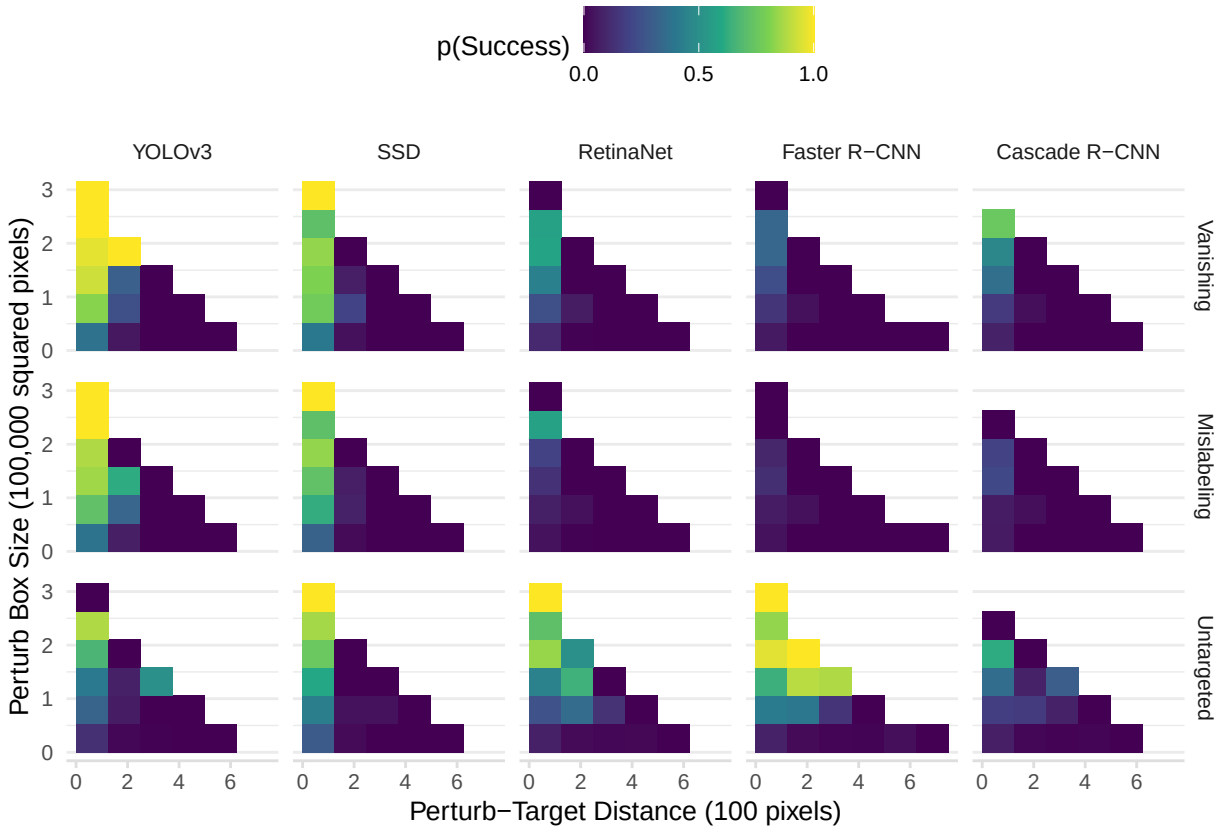


Figure 8. **Larger perturb objects significantly increase success rates for all models and attacks, except for mislabeling attack on Faster R-CNN, after controlling for perturb-target distances; Shorter perturb-target distances significantly increase success rates for all models and attacks, after controlling for perturb object sizes:** The binned summaries graph success proportion against perturb-target distance (100 pixels) and perturb box size (100,000 squared pixels) in the randomized attack experiment.

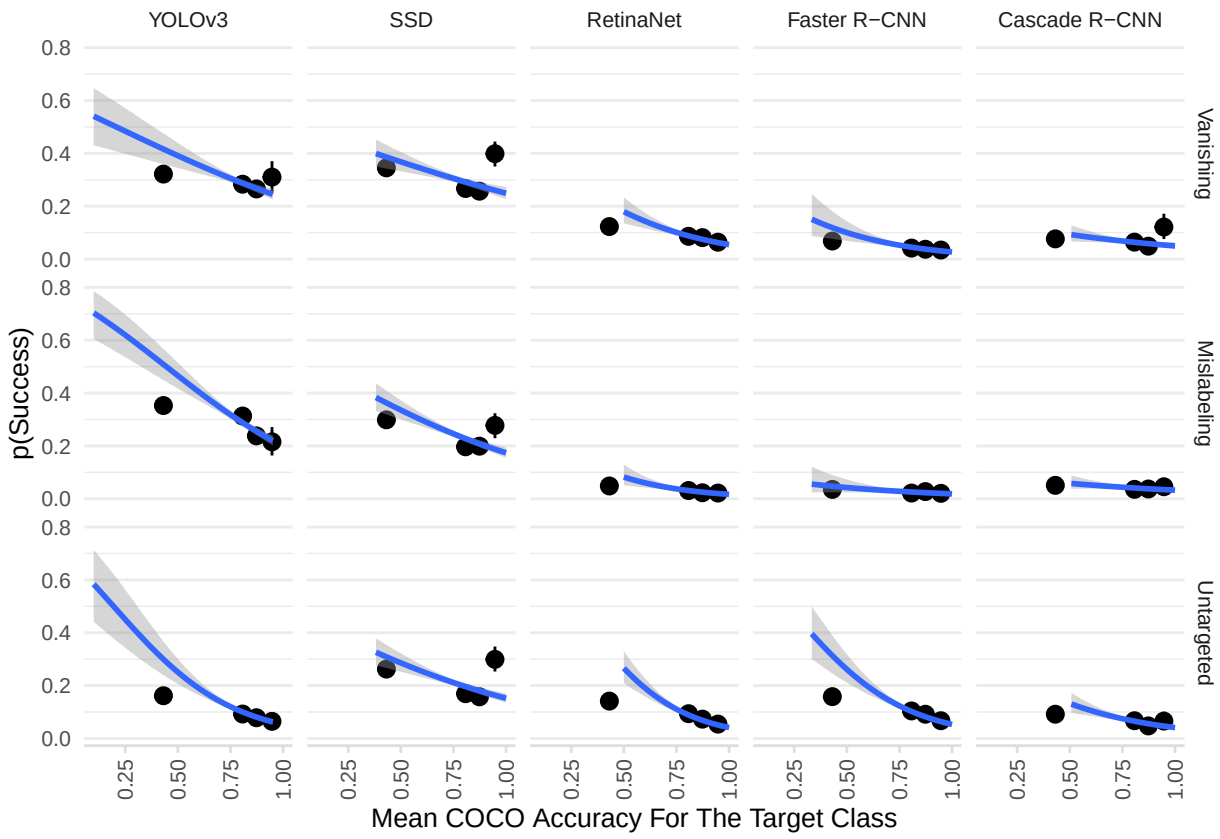


Figure 9. **Although higher mean COCO accuracy for the target class seem to decrease success rates, the results are mixed after controlling for target class confidence (Table 8):** The binned summaries and regression trendlines graph success proportion against mean COCO accuracy for the target class in the randomized attack experiment. Bins are split into quantiles. Errors are 95% confidence intervals.

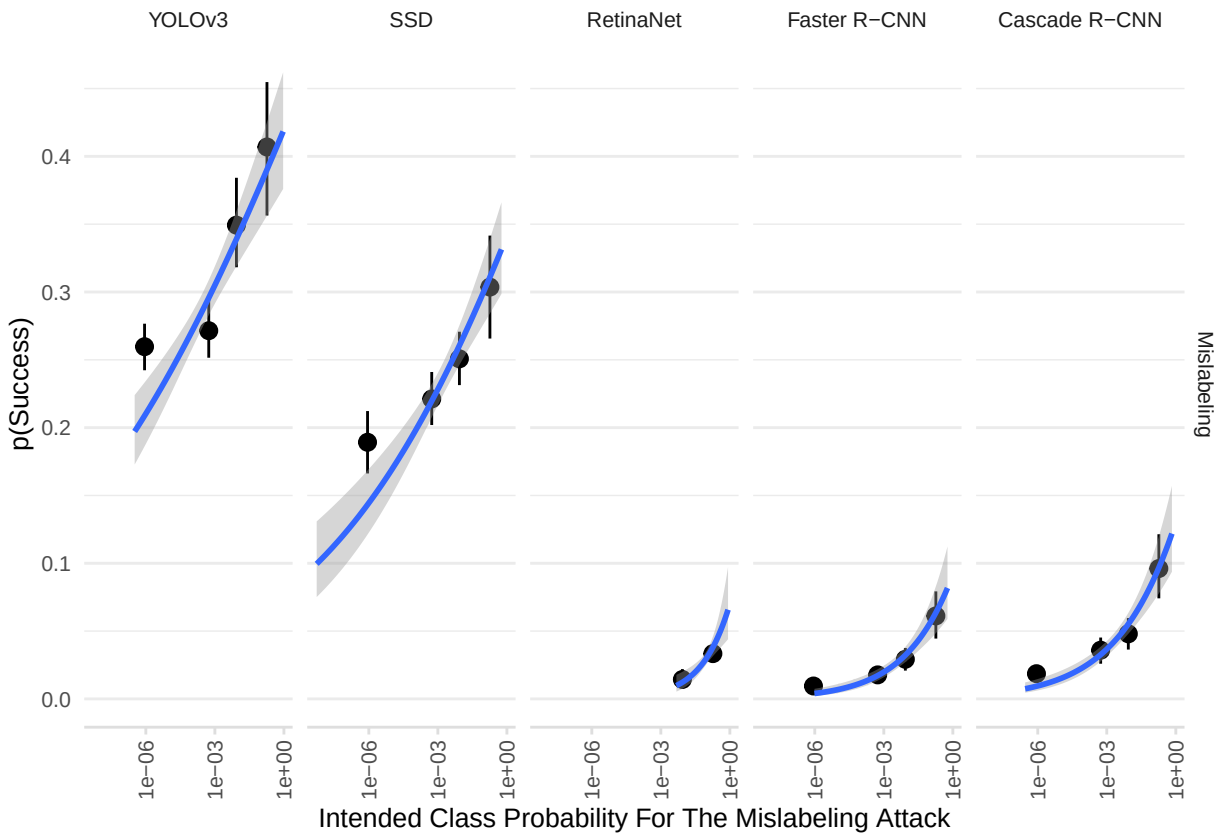


Figure 10. **Although intended class probability seem to increase success rates for the mislabeling attack, it does not predict success rates after controlling for target class confidence, except for RetinaNet (Table 9):** The binned summaries and regression trendlines graph success proportion against intended class probability in the randomized attack experiment. Bins are split into quantiles. Errors are 95% confidence intervals.

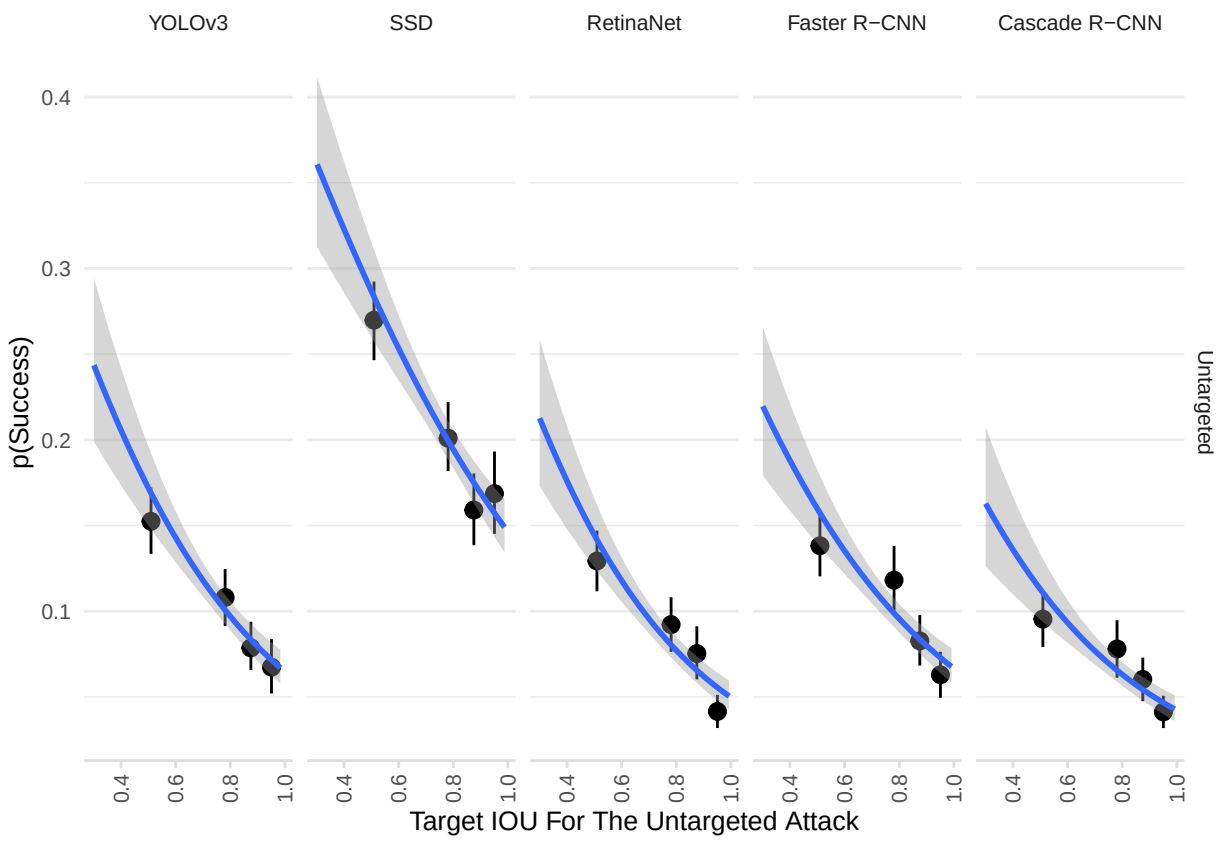


Figure 11. **Target IOU for the untargeted attack increases success rates on all models:** The binned summaries and regression trendlines graph success proportion against target IOU for the untargeted attack in the randomized attack experiment. Bins are split into quantiles. Errors are 95% confidence intervals.

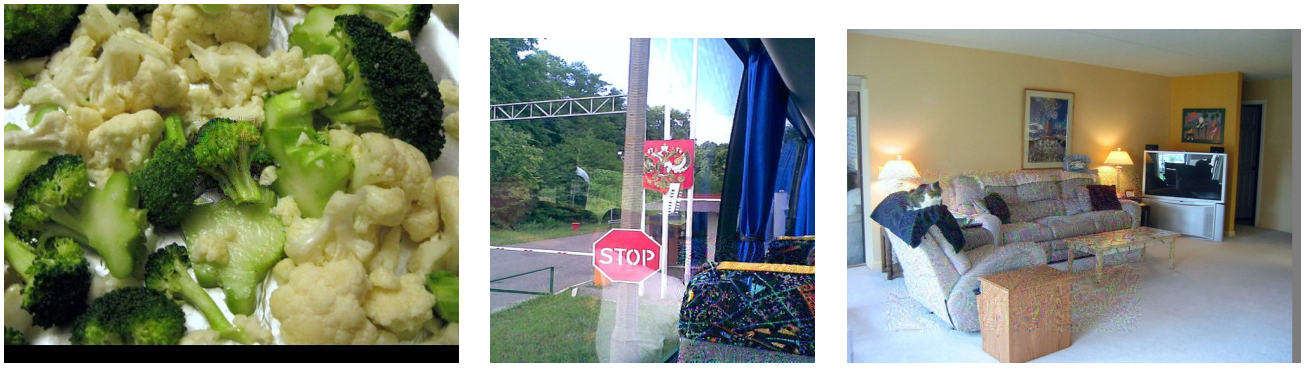


Figure 12. These are the perturbed images corresponding to attacked examples illustrated in Figure 1.

D. Deliberate Attack

D.1. Setup

Since we are using a shared computing resource on an internal network, we split the attack into 2 repetitions and attacked 100 images per repetition. The images are randomly sampled without replacement within repetitions, but may repeat across repetitions. Every repetition takes 45-60 minutes on a 32GB NVIDIA Tesla V100 GPU. 480 repetitions (5 models * 3 attacks * 4 perturb box lengths * 4 perturb-target distances * 2 repetitions) take 15-20 V100 GPU days. We also re-ran 3 SSD repetitions which arbitrarily crashed.

D.2. Results

Table 11: We run a logistic model regressing success against perturb-target distance and perturb box length in the deliberate attack experiment. Longer perturb box length or shorter perturb-target distance cause success rates to significantly increase for all model and attack combinations, except for perturb box length in untargeted attack on Cascade R-CNN. The interaction terms, even when significant, are negligibly close to 0. Table headers are explained in Appendix C.4.1.

Group		Regression						
Attack	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
YOLOv3								
Vanishing	distance	*	-0.015	0.002	-7.681	0.000	-0.018	-0.011
	length	*	0.030	0.002	19.637	0.000	0.027	0.033
	distance * length	*	0.000	0.000	-6.081	0.000	0.000	0.000
Mislabeling	distance	*	-0.015	0.002	-8.540	0.000	-0.018	-0.011
	length	*	0.019	0.001	16.603	0.000	0.016	0.021
	distance * length		0.000	0.000	-1.733	0.083	0.000	0.000
Untargeted	distance	*	-0.021	0.003	-7.440	0.000	-0.026	-0.015
	length	*	0.007	0.001	6.528	0.000	0.005	0.009
	distance * length		0.000	0.000	1.467	0.142	0.000	0.000
SSD								
Vanishing	distance	*	-0.015	0.002	-7.055	0.000	-0.019	-0.011
	length	*	0.024	0.001	17.747	0.000	0.021	0.027
	distance * length	*	0.000	0.000	-4.823	0.000	0.000	0.000
Mislabeling	distance	*	-0.018	0.002	-7.553	0.000	-0.023	-0.014
	length	*	0.020	0.001	15.991	0.000	0.017	0.022
	distance * length	*	0.000	0.000	-2.458	0.014	0.000	0.000
Untargeted	distance	*	-0.018	0.002	-7.742	0.000	-0.023	-0.014
	length	*	0.013	0.001	11.665	0.000	0.011	0.015
	distance * length		0.000	0.000	-0.873	0.383	0.000	0.000
RetinaNet								
Vanishing	distance	*	-0.045	0.006	-7.187	0.000	-0.058	-0.033
	length	*	0.016	0.001	10.614	0.000	0.013	0.019
	distance * length	*	0.000	0.000	-2.147	0.032	0.000	0.000
Mislabeling	distance	*	-0.031	0.007	-4.240	0.000	-0.047	-0.018
	length	*	0.008	0.002	4.541	0.000	0.005	0.012
	distance * length		0.000	0.000	-1.021	0.307	0.000	0.000

Untargeted	distance	*	-0.038	0.005	-7.446	0.000	-0.049	-0.029
	length	*	0.005	0.001	3.969	0.000	0.003	0.008
	distance * length	*	0.000	0.000	6.925	0.000	0.000	0.000
Faster R-CNN								
Vanishing	distance	*	-0.061	0.010	-6.407	0.000	-0.081	-0.044
	length	*	0.011	0.001	7.127	0.000	0.008	0.014
	distance * length		0.000	0.000	-0.490	0.624	0.000	0.000
Mislabeling	distance	*	-0.054	0.012	-4.664	0.000	-0.080	-0.034
	length	*	0.007	0.002	3.706	0.000	0.003	0.010
	distance * length		0.000	0.000	-0.717	0.473	0.000	0.000
Untargeted	distance	*	-0.044	0.006	-8.012	0.000	-0.056	-0.034
	length	*	0.005	0.001	3.676	0.000	0.002	0.007
	distance * length	*	0.000	0.000	6.889	0.000	0.000	0.000
Cascade R-CNN								
Vanishing	distance	*	-0.063	0.010	-6.579	0.000	-0.083	-0.046
	length	*	0.015	0.002	9.395	0.000	0.012	0.018
	distance * length		0.000	0.000	-1.003	0.316	0.000	0.000
Mislabeling	distance	*	-0.062	0.012	-5.240	0.000	-0.088	-0.041
	length	*	0.010	0.002	5.795	0.000	0.006	0.013
	distance * length		0.000	0.000	-0.122	0.903	0.000	0.000
Untargeted	distance	*	-0.061	0.008	-7.544	0.000	-0.079	-0.047
	length		0.002	0.001	1.498	0.134	-0.001	0.005
	distance * length	*	0.000	0.000	6.198	0.000	0.000	0.000

Table 12: We combined the data in the randomized and deliberate attack experiments to run a logistic model regressing success against object (versus non-object), with perturb-target distance (100 pixels) and perturb box size (100,000 squared pixels) as covariates. The "object" term codes object as 1 and non-object as 0. 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. Table headers are explained in Appendix C.4.1.

Group	Regression							
Attack	term	sig	estimate	std.error	statistic	p.value	conf.low	conf.high
YOLOv3								
Vanishing	object	*	-0.317	0.063	-5.031	0.000	-0.440	-0.193
	distance	*	-1.711	0.076	-22.383	0.000	-1.863	-1.563
	size	*	8.585	0.367	23.423	0.000	7.878	9.315
	distance * size	*	-3.258	0.310	-10.498	0.000	-3.872	-2.655
Mislabeling	object		-0.026	0.059	-0.440	0.660	-0.141	0.089
	distance	*	-1.515	0.066	-22.796	0.000	-1.647	-1.386
	size	*	4.538	0.253	17.940	0.000	4.050	5.041
	distance * size	*	-0.908	0.231	-3.938	0.000	-1.365	-0.462
Untargeted	object	*	-0.201	0.079	-2.544	0.011	-0.357	-0.046
	distance	*	-1.970	0.114	-17.351	0.000	-2.197	-1.752

	size	*	1.593	0.170	9.364	0.000	1.265	1.932
	distance * size		0.356	0.250	1.423	0.155	-0.149	0.837
SSD								
Vanishing	object		0.096	0.063	1.532	0.125	-0.027	0.219
	distance	*	-1.924	0.084	-22.955	0.000	-2.090	-1.762
	size	*	5.883	0.296	19.896	0.000	5.313	6.472
	distance * size	*	-2.263	0.289	-7.826	0.000	-2.844	-1.707
Mislabeling	object		-0.039	0.064	-0.609	0.542	-0.166	0.087
	distance	*	-1.953	0.091	-21.407	0.000	-2.134	-1.776
	size	*	4.228	0.249	16.958	0.000	3.749	4.726
	distance * size	*	-1.509	0.282	-5.356	0.000	-2.077	-0.971
Untargeted	object	*	0.176	0.066	2.661	0.008	0.047	0.306
	distance	*	-2.060	0.093	-22.041	0.000	-2.246	-1.879
	size	*	1.958	0.187	10.482	0.000	1.599	2.331
	distance * size		-0.227	0.244	-0.929	0.353	-0.719	0.240
RetinaNet								
Vanishing	object	*	-0.551	0.093	-5.947	0.000	-0.734	-0.370
	distance	*	-4.949	0.261	-18.960	0.000	-5.472	-4.448
	size	*	2.686	0.251	10.722	0.000	2.208	3.190
	distance * size		-0.881	0.569	-1.548	0.122	-2.035	0.197
Mislabeling	object		-0.245	0.136	-1.799	0.072	-0.513	0.022
	distance	*	-3.968	0.357	-11.109	0.000	-4.697	-3.297
	size	*	1.163	0.231	5.032	0.000	0.712	1.621
	distance * size		0.117	0.696	0.168	0.867	-1.323	1.403
Untargeted	object	*	-0.448	0.091	-4.902	0.000	-0.628	-0.269
	distance	*	-1.333	0.106	-12.560	0.000	-1.546	-1.130
	size	*	1.675	0.165	10.157	0.000	1.355	2.002
	distance * size	*	1.766	0.203	8.701	0.000	1.373	2.170
Faster R-CNN								
Vanishing	object	*	-0.768	0.113	-6.813	0.000	-0.991	-0.549
	distance	*	-6.002	0.408	-14.728	0.000	-6.829	-5.230
	size	*	2.062	0.256	8.052	0.000	1.572	2.577
	distance * size		-1.190	0.905	-1.315	0.188	-3.059	0.485
Mislabeling	object	*	-0.384	0.139	-2.770	0.006	-0.657	-0.113
	distance	*	-5.868	0.483	-12.144	0.000	-6.858	-4.961
	size		0.461	0.252	1.832	0.067	-0.029	0.958
	distance * size	*	2.055	0.747	2.752	0.006	0.440	3.362
Untargeted	object	*	-0.275	0.089	-3.096	0.002	-0.449	-0.101
	distance	*	-1.804	0.124	-14.599	0.000	-2.053	-1.568
	size	*	2.104	0.182	11.585	0.000	1.752	2.464
	distance * size	*	2.226	0.233	9.570	0.000	1.778	2.690

Cascade R-CNN								
Vanishing	object	*	-0.665	0.104	-6.395	0.000	-0.870	-0.462
	distance	*	-6.496	0.388	-16.731	0.000	-7.279	-5.757
	size	*	2.905	0.277	10.474	0.000	2.378	3.465
	distance * size		-1.579	0.840	-1.881	0.060	-3.310	-0.020
Mislabeling	object	*	-0.282	0.117	-2.402	0.016	-0.513	-0.052
	distance	*	-6.317	0.438	-14.410	0.000	-7.210	-5.489
	size	*	0.886	0.220	4.018	0.000	0.459	1.325
	distance * size		1.310	0.746	1.757	0.079	-0.265	2.666
Untargeted	object		-0.175	0.100	-1.739	0.082	-0.371	0.022
	distance	*	-2.464	0.159	-15.457	0.000	-2.786	-2.160
	size	*	0.913	0.161	5.677	0.000	0.598	1.229
	distance * size	*	2.093	0.216	9.686	0.000	1.670	2.519

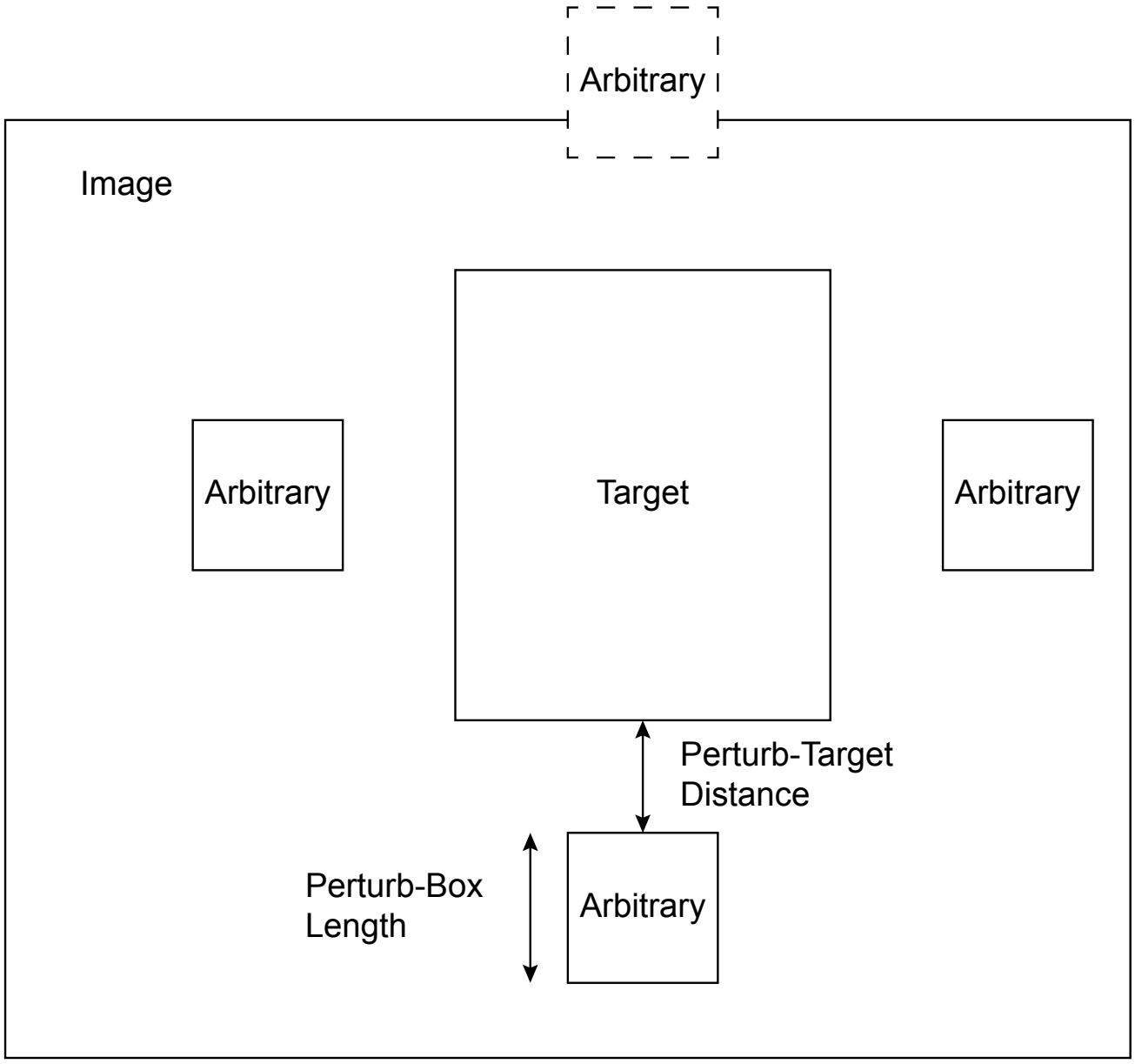


Figure 13. We enclose a non-overlapping square perturb region in 1 over 4 possible directions around the target object, on the top, left, right or bottom, as illustrated. The square perturb region is axes and center-aligned to the target bounding box, and the perturb-target distance is the shortest distance between the perturb and target boundaries. We randomly sample among the eligible directions in which the perturb region is within image bounds. In the illustrated example, the top dashed region is not eligible. When all directions are not eligible, we discard the image and resample. Across model and attack combinations, we sample the same images and select the same target object and perturb direction per image to more accurately compare the success rates between combinations.