Denial-of-Service Attack on Object Detection Model Using Universal Adversarial Perturbation

Avishag Shapira  
The Open University  
Ra’anana, Israel  
avishagsh@openu.ac.il

Alon Zolfi  
Ben-Gurion University of the Negev  
Be’er Sheva, Israel  
zolfi@post.bgu.ac.il

Luca Demetrio, Battista Biggio  
University of Cagliari  
Cagliari, Italy  
{lucademetrio93,battista.biggio}@unica.it

Asaf Shabtai  
Ben-Gurion University of the Negev  
Be’er Sheva, Israel  
shabtaia@bgu.ac.il

Abstract

Adversarial attacks against deep learning-based object detectors have been studied extensively in the past few years. The proposed attacks aimed solely at compromising the models’ integrity (i.e., trustworthiness of the model’s prediction), while adversarial attacks targeting the models’ availability, a critical aspect in safety-critical domains such as autonomous driving, have not been explored by the machine learning research community. In this paper, we propose NMS-Sponge, a novel approach that negatively affects the decision latency of YOLO, a state-of-the-art object detector, and compromises the model’s availability by applying a universal adversarial perturbation (UAP). In our experiments, we demonstrate that the proposed UAP is able to increase the processing time of individual frames by adding "phantom" objects while preserving the detection of the original objects.

1 Introduction

Deep learning-based computer vision models, and specifically object detection models, are becoming an essential part of intelligent systems in many domains, including autonomous driving. Two main security-related aspects are critical to the adoption of these models: (1) integrity - ensuring the accuracy and correctness of the predictions, and (2) availability - ensuring uninterrupted access to the system.

In the past few years, object detection models have been shown to be vulnerable to adversarial attacks, including attacks in which a patch containing an adversarial pattern is placed on the target object (e.g., black and white stickers [7], a cardboard plate [23], T-shirts [27, 25, 11] or a physical patch is attached to the camera’s lens [29]). These attacks share one main attribute – they are all aimed at compromising the model’s integrity. To the best of our knowledge, no studies have explored the ability to create an attack that compromises the model’s availability (i.e., prediction latency).

Recently, availability-based attacks have been shown to be effective against deep learning-based models. Shumailov et al. [20] presented sponge examples, which are perturbed inputs designed to increase the energy consumed by natural language processing (NLP) and computer vision models, when deployed on hardware accelerators, by increasing the number of active neurons during classification. Following this work, other studies have proposed sponge-like attacks, mainly targeting image classification models [4, 5, 6, 10].
In this paper, we present **NMS-Sponge**, the first availability attack against the end-to-end object detection pipeline, which is performed by applying a universal adversarial perturbation (UAP). Our initial attempt to apply the sponge attack proposed by Shumailov et al. [20] on YOLO to decelerate inference processing was unsuccessful. This is due to the fact that for most of the images, the vast majority of the model’s activation values are different than zero by default.

In the proposed attack, gradient-based optimization is used to construct a UAP that targets the non-maximum suppression (NMS) component of the object detector, creating an abundance of candidate bounding box predictions to be processed by the NMS, overloading the system and slowing it down. Our custom loss function aims to preserve the detection of the original objects present in the image. The fact that we create a universal perturbation makes our attack practical – that is, it can be applied to any stream of images in real time, thereby affecting the system’s decision latency.

We evaluated our proposed attack on the YOLOv5 object detector [12] and used the BDD100K dataset [28], containing driving images, to train the UAP and evaluate the attack. Our results show that the use of our perturbation increases the inference time by $\times 1.4$ without compromising the detection capabilities (70% of the original objects are detected by the model when the proposed attack is performed). In addition, we show that the attack is effective when the model runs both on a CPU and a GPU (i.e., not only on hardware accelerators), and that it is possible to produce a UAP that successfully attacks different object detectors.

The contributions of our work can be summarized as follows:

- We present the first availability attack against object detectors. The attack increases the processing time of a single image by up to $\times 1.4$ while allowing the detection of 70% of the original objects.
- We construct a universal perturbation that can be applied on a video stream in real time.
- We make the code of our attack available to the research community.

## 2 Object Detectors

State-of-the-art object detection models can be broadly categorized into two types: one-stage detectors (e.g., SSD [14], YOLO [18, 16, 17]) and two-stage detectors (e.g., Mask R-CNN [9], Faster R-CNN [19]). Two-stage detectors use a region proposal network to locate potential bounding boxes in the first stage; these boxes are used for classification of the bounding box content in the second stage. One-stage detectors simultaneously generate bounding boxes and predict their class label.

In our work, we focus on the state-of-the-art one-stage object detector YOLO. YOLO was first introduced in [18], which proposed YOLO’s unique architecture consisting of two parts: a convolution-based backbone (referred to as Darknet-19) used for feature extraction, which is followed by a grid-based detection head used to predict bounding boxes and their associated labels. Following that, in YOLOv2 [16] the authors argued that predicting the offset from predefined anchor boxes [19] makes it easier for the network to learn. Later, an improved version (YOLOv3 [17]) included multiple improvements to the architecture: replacing the old backbone with a “deeper” network (referred to as Darknet-53) which contains residual connections [8] and multi-scale detection layers (which are referred to as detection heads) to predict bounding boxes at three different scales (downsampled from the original image size by 32, 16, and 8) instead of the single scale used in the first version.

The last layer of each detection head predicts a 3D tensor that encodes the bounding box, objectness score, and class predictions. More specifically, each detection head predicts $N \times N \times [3 \times (4 + 1 + N_c)]$ candidates, where $N \times N$ is the final grid size, 3 is the number of anchor boxes per cell in the grid, 4 is the number of coordinate offsets from the anchor box, 1 is the objectness score, and $N_c$ is the number of class predictions.

The **Objectness score** is the detector’s confidence that the bounding box contains an object ($Pr(\text{Object})$). The **Class score** is the detector’s confidence that the bounding box contains an object of a specific class ($Pr(\text{Class}_i|\text{Object})$). This architecture design established the foundation for many of the object detectors proposed in recent years [2, 24, 12].

**YOLO’s end-to-end detection pipeline.** As explained above, YOLO outputs a fixed amount of candidate predictions for a given image size. For example, for an image size of $640 \times 640$ pixels, the number of candidate predictions is $3 \times (80 \times 80 + 40 \times 40 + 20 \times 20) = 25,200$ (one for each
anchor in a specific cell in each final grid). These candidates are later filtered sequentially using two conditions:

- The objectness score surpasses a predefined threshold, \( Pr(\text{Object}) > T_{\text{conf}} \).
- At least one class score (unconditional) surpasses the predefined threshold, \( \exists\{Pr(Class_i) = Pr(\text{Object}) \times Pr(Class_i|\text{Object}) > T_{\text{conf}}\} \).

Finally, since many candidate predictions may overlap and predict the same object, the non-maximum suppression (NMS) algorithm is applied to remove multiple detections.

Potentially, NMS could be provided with the maximum amount of candidate predictions, significantly increasing the time it takes to process each image.

**Non-Maximum Suppression (NMS).** The NMS algorithm consists of two steps for every given class category: (a) sort all candidate predictions based on their confidence scores in descending order, and (b) select the highest ranking candidate and discard all candidates in which their IoU surpasses a pre-defined threshold (the bounding boxes overlap above a specific level). These two steps are repeated until all candidates have been chosen or discarded by the algorithm.

3 Related Works

Adversarial attacks on deep-learning based systems and particularly on object detection models have been studied extensively over the last few years [1]. Most of the previous studies focused on compromising the models’ integrity. Limited research has examined attacks in which the models’ availability is targeted. Below, we categorize prior research based on the system component targeted in the attack.

### 3.1 Integrity-Based Attacks

The increased adoption of deep-learning systems in the last few years has been accompanied by increased research aimed at challenging the systems’ robustness. First, adversarial attacks against image classification models emerged (FGSM [22], PGD [15], etc.). Later, attacks against more complex mechanisms (e.g., object detection models) were demonstrated. To evade the detection of Faster R-CNN [19], Chen et al. [5] printed stop signs that contained an adversarial pattern in the background. Sitawarin et al. [21] crafted toxic traffic signs, visually similar to the original signs. Thys et al. [23] first proposed an attack against person detectors, using a cardboard plate which is attached to the attacker’s body. In improved versions of this method the adversarial pattern, which was shown to be capable of evading the detection of various state-of-the-art object detectors, was printed on a T-shirt [25, 27]. However, as noted earlier, none of these studies proposed methods that target the system’s availability component.

### 3.2 Availability-Based Attacks

Availability-based attacks have only recently gained the attention of researchers, despite the fact that a system’s availability is a security-critical aspect of many applications. Shumailov et al. [20] were the first to present an attack (called sponge examples) targeting the availability of computer vision and NLP models. In their paper, they demonstrated that adversarial examples are capable of doubling the inference time of NLP transformer-based models, with inference times \( \times 6000 \) greater than regular input. Boutros et al. [4] extended the sponge examples attack so it could be applied on FPGA devices. In [3], the authors presented a method for creating sponge examples that preserve the original input’s visual appearance. Cina et al. [6] proposed sponge poisoning, a technique that performs sponge attacks during training time, resulting in a poisoned model with decreased performance. Hong et al. [10] showed that crafting adversarial examples (including a universal perturbation) could slow down multi-exit networks.

While the previous studies presented above target classification models in the computer vision domain, in this paper we focus on object detection models, a target that was not addressed in prior research. We propose a universal perturbation that is able to fool all images simultaneously. It should be noted, that due to the diverse nature of images in the object detection domain (i.e., objects appear in different locations and scales), aiming at creating a universal perturbation increases the difficulty of executing a successful attack.
In this research, we aim to produce a digital UAP capable of causing a delay in the end-to-end processing time of an image processed by the YOLO object detector. Our attack is designed to achieve the following objectives:

- Increase the amount of time it takes for the object detector to process a single image.
- Create a universal perturbation that be applied to any image (frame) within a stream of images, for example, in the autonomous driving scenario.
- Preserve the detection of original objects in the image.

As explained in Section 2, YOLO outputs a fixed amount of candidate predictions, which are then filtered based on their confidence scores. The remaining candidates are then passed to the NMS algorithm to reduce redundancy.

Therefore, increasing the number of candidates passed to the NMS algorithm will increase the postprocessing time of the end-to-end detection pipeline, resulting in a decision delay.

### 4.1 Optimization Process

To optimize our perturbation’s parameters, we use projected gradient descent (PGD) with the $L_2$ norm. We compose a novel loss function that aims to achieve the objectives presented above; our loss function consists of two components: (a) the max-objects loss, and (b) the intersection over union (IoU) loss.

### 4.2 Max-Objects loss

Let $C$ be the group of all candidates produced by the object detector, and let $C^{clean}$ (resp. $C^{pert}$) be the group of candidates for a clean image (resp. perturbed image). In order to increase the number of candidate predictions that are passed to the NMS stage ($C^{pert}_{out}$), we need to increase the number of predictions that are not filtered by the confidence threshold $T_{conf}$ (see Section 2). Therefore, we aim to increase the confidence scores of all candidates that do not surpass $T_{conf}$. We also limit the increase of the candidate’s confidence to $T_{conf}$, so that the loss favors increased prediction of far-from-threshold candidates. More formally, the loss for a single candidate is as follows:

$$\ell_{single \, conf}(c) = -(c_{objectness \, score} \cdot c_{class \, score}) + T_{conf}$$  \hspace{1cm} (1)
While this component focuses on the confidence of the predictions, we also want to consider the amount of candidate predictions that were filtered prior to the NMS stage. Therefore, the loss over all candidates is as follows:

$$\ell_{\text{max objects}} = \frac{1}{|C_{\text{out}}|} \sum_{c \in C_{\text{out}}} \ell_{\text{single conf}}(c)$$  \hspace{1cm} (2)

4.3 IoU loss

To achieve our third objective of preserving the detection of the original objects in the image, we maximize the IoU score between the final predictions’ bounding boxes (predictions that are returned from the NMS stage) in a clean image (not attacked) $C_{\text{clean, final}}$ and the perturbed image $C_{\text{per, final}}$.

Therefore, for a single bounding box prediction $\text{bbox}_{\text{clean}} \in C_{\text{clean, final}}$, we extract the maximum IoU value:

$$\text{Max IoU} = \max_{\text{bbox}_{\text{per}} \in C_{\text{per, final}}} \text{IoU}_{\text{bbox}_{\text{clean}}, \text{bbox}_{\text{per}}}$$  \hspace{1cm} (3)

To be more precise, since we aim to minimize a loss function, and IoU’s value is in the range $[0, 1]$, the loss value is defined as:

$$\ell_{\text{single bbox}} = 1 - \text{Max IoU}$$  \hspace{1cm} (4)

Finally, the loss of this component is defined as follows:

$$\ell_{\text{max IoU}} = \frac{1}{|C_{\text{clean, final}}|} \sum_{\text{bbox}_{\text{clean}} \in C_{\text{clean, final}}} \ell_{\text{single bbox}}$$  \hspace{1cm} (5)

**Final loss function.** The final loss function of the attack consists of the two components presented above and is defined as follows:

$$\min_p [\alpha \ast \ell_{\text{max objects}} + (1 - \alpha) \ast \ell_{\text{max IoU}}]$$  \hspace{1cm} (6)

where $p$ is the perturbation and $\alpha$ is a weighting factor. The computed gradients are backpropagated to update our perturbation’s pixels, as shown in Figure 1.

5 Evaluation

5.1 Evaluation setup

**Model.** In our evaluation, we conduct experiments on the latest version of the state-of-the-art YOLO object detector, YOLOv5 [12]. YOLOv5 offers several model networks in different sizes [12] (small, medium, etc.), where all the models are pre-trained on the MS-COCO dataset [13].

**Datasets.** Since we focus on the autonomous driving domain, we opted to use the Berkeley DeepDrive dataset [26], also referred to as BDD-100k. The BDD100k dataset contains 100K images with various attributes such as: weather (clear, rainy, etc.), scene (city street, residential, etc.), and time of day (daytime, night, etc.), resulting in a diverse dataset.

**Evaluation metrics.** Our attack aims two achieve two goals: (1) increase the model’s end-to-end inference time, and (2) preserve the detection of the objects in the original image. To quantify the effectiveness of our attack in achieving the first goal we propose the following two metrics:

- **# of objects** - the number of objects processed during the NMS stage.
- **time** - the total processing time of the end-to-end detection process in milliseconds. We also denote the NMS stage processing time.
Figure 2: The number of candidate predictions processed by the NMS algorithm for different \( \epsilon \) values \((\alpha=1.0)\). Red line indicates the maximum number of candidates output by YOLO for image of size 640 \( \times \) 640 pixels.

| \( \alpha \) | Image Type | Total time (NMS time) | \# of Predictions | Recall |
|-------------|------------|-----------------------|-------------------|--------|
|             | Clean      | 24 (2.2)              | 80                | 100%   |
| 0.7         | Adv        | 33.8 (12.0)           | 9,100             | 69%    |
| 0.8         | Adv        | 36.5 (14.7)           | 11,700            | 58%    |
| 1.0         | Adv        | 54.2 (32.4)           | 18,600            | 0.5%   |
|             |            |                       |                   |        |
| 0.7         | Adv        | 33.5 (11.7)           | 9,000             | 69%    |
| 0.8         | Adv        | 36.9 (15.1)           | 12,200            | 52%    |
| 1.0         | Adv        | 65.1 (43.3)           | 22,500            | 0.3$   |

Table 1: Average results when using various \( \alpha \) values (different weight balancing of the loss function components).

To evaluate our attack’s effectiveness in achieving the second goal, we evaluate the:

- **recall** - the number of original objects that were detected in the perturbed image (compared to the recall in the clean image).

**Implementation details.** To train and test our adversarial perturbation, we randomly chose 2,000 images from the BDD100K validation set. We used 1,500 images to train the UAP, and then examined its effectiveness on the remaining 500 images. \( T_{\text{conf}} \) is set at 0.25, since this is the default value used for these models in the inference phase (i.e., throughout this section the presented recall values are based on this threshold). the small sized YOLOv5 model (also referred to as YOLOv5s) is used as our target model. The model’s inference running time may vary between different runs. Therefore, for each image, we calculated the average inference time over 40 iterations to obtain non-biased measurements.

The experiments were conducted on a GPU (NVIDIA Quadro T1000) and a CPU (Intel Core i7-9750H).

### 5.2 Results

In this section we present results for different experiments conducted using our UAP.

**Effectiveness of the UAP for different epsilon (\( \epsilon \)) values.** The \( \epsilon \) parameter in the PGD attack denotes the radius of the hyper-sphere, i.e., the maximum amount of noise that will be added to image. Larger \( \epsilon \) values will result in a more substantial perturbation that while being more perceptible to the human eye, will result in a more successful attack. In Figure 2, we present the number of
candidate predictions passed to the NMS algorithm for different $\epsilon$ values (when the $\alpha$ parameter is set at 1.0, i.e., the attack ignores the IoU loss component in the loss function and thus does not attempt to preserve the original objects). As expected, we can see that the larger the $\epsilon$, the more candidate predictions surpass the confidence conditions and processed by the NMS component, almost reaching the maximum amount of possible candidates. However, the magnitude at which the candidate predictions are added to the image, decreases as the value of the $\epsilon$ increases. For example, increasing $\epsilon=5$ to 25, added an average of 10K predictions to an image, however, increasing $\epsilon=50$ to 70 barely influenced the amount of candidates (see Figure 2). The recall values remained relatively constant for the various $\epsilon$ values, although it was slightly higher for lower $\epsilon$ values. In Figure 5 we provide examples of different perturbations trained using different $\epsilon$ values.

**Effectiveness of the UAP for different alpha ($\alpha$) values.** As mentioned in Section 4, the loss function consists of two components. The $\alpha$ value enables control of the balance between the components. As expected, the higher the $\alpha$ value is, the more candidate predictions the UAP adds to an image; at the same time, however, the recall value decreases, and the perturbation manages to preserve less objects that were detected in the original image (see Table 1).

By visually examining the UAPs (e.g., Figure 4), it is possible to see that the attack learns the most common areas in the natural images in which objects appear, i.e., when setting $\alpha = 1$ (preservation
Figure 5: Examples of three images with the UAP applied for different $\epsilon$ values, where $\alpha = 1.0$.

component deactivated) objects are added all over the image, and when setting $\alpha < 1$ the center of the perturbed image remains un-attacked. This is an outcome we expected, since there are naturally fewer objects in these areas (where we would usually find a sky, road, or sidewalk in the autonomous driving domain).

Furthermore, we also observed that from a certain $\alpha$ value (\sim 0.65), the loss function strongly favors the preservation of the objects in the original image, i.e., the balance between the loss’s two components in such a way that any small change influencing the final predicted objects in the original image increases the overall loss value, resulting an unsuccessful attack.

**Executing the attack on a GPU vs. CPU.** We tested the attack for different values of $\alpha$ and $\epsilon$ on a GPU and a CPU. The attack works efficiently on both of the devices, increasing the inference running time proportionally for different UAPs. For example, Figure 3 presents the GPU and CPU inference time results on two different UAPs. It is possible to see that the inference running time results for the UAP that was created with $\epsilon=70$ and $\alpha=1.0$ values are four times longer than results of the UAP that was created with $\epsilon=30$ and $\alpha=0.7$ values, for both GPU and CPU. In addition, we can observe that our UAP substantially increases the NMS processing time portion out of the total end-to-end detection process time compared to it’s portion in the clean image.

**Generic (common) UAP.** As note earlier, YOLOv5 has several version varying in their size. During the experiments, we proved that all these networks are individually vulnerable to our attack. However, our UAP does not seem to be transferable to the other YOLO networks and to other object detection models. Nonetheless, in our experiments we show that it is possible to build a common UAP that efficiently attacks different models by using the ensemble technique. Specifically, in our experiments, we created UAPs that were randomly trained on two different sized YOLO’s networks. We found that the attack manages to produce successful common UAPs for the two models the UAPs were trained on. For example, a UAP that was created with $\epsilon=70$ and $\alpha=1.0$ values and was trained on YOLOv5s (small) and YOLOv5m (medium), adds an average of 21K predictions for these two models (while, the UAP that was trained only on the YOLOv5s network with the same parameter values, adds an average of 22.5K predictions to an image). Another UAP that was created with $\epsilon=30$ and $\alpha=0.8$ values and was trained on YOLOv5n (nano) and YOLOv5m, adds an average of 10.9K predictions to the YOLOv5m and an average of 8.5K predictions to the YOLOv5n network (while,
the UAP that was trained only on the YOLOv5m model using the same values, adds an average of 12K predictions to an image).

These results indicate that an attacker aimed at performing the attack does not need to know the type/version of the attacked model. In order to generate a successful attack, one UAP trained on an ensemble of models can be generated and still be effective.

5.3 Discussion

The "phantom" predictions. The UAP mainly adds person and chair class predictions. We assume that this has to do with the fact that the model is pretrained on the COCO-MS dataset [13] in which ‘person’ is the most common class, and ‘chair’ is the third most common target class in the train set.

It is also interesting to see that the UAP adds a different target class in different areas in the image. In Figure 6 we can see that the perturbation mainly adds person predictions in the bottom area of the image (where there usually would be a street) and mainly adds chair predictions in the top area of the image (where the sky usually is).

Mitigation. One possible mitigation against the NMS-Sponge attack is to limit the image’s processing time. If the processing time is longer than a predefined threshold, the system interrupts the detection process. This mitigation can bound the system’s latency time, but it actually serves the attacker’s purpose, harming the availability of the system. Another approach would be to bound the number of candidates passed to the NMS stage; however, this also serves a different purpose of the attacker, compromising the integrity of the model. Therefore, it might not be an appropriate solution for real-time systems such as an autonomous vehicle’s object detection system, since interrupting the detection process every several frames might have severe consequences, endangering the car’s drivers and passengers, pedestrians, and other drivers on the road.

A complementary approach for defending against the NMS-Sponge attack should focus on real-time detection of the attack and eliminating the "phantom" objects.

6 Conclusion

In this paper, we present the NMS-Sponge attack, which significantly increases the inference time of the state-of-art YOLO object detector by applying a UAP that increases the amount of candidate predictions processed during the NMS postprocessing stage. This UAP adds "phantom" predictions to the image while preserving the predictions made by the object detector in the original (unattacked) image.

The NMS algorithm is widely used in object detection models. Therefore, by demonstrating that YOLO is vulnerable to the NMS-Sponge attack, it can be assumed that it is possible to attack the NMS stage in other object detections models by applying the NMS-Sponge attack principles.
In future work, we plan to improve the attack by reducing the number of final predictions that the UAP adds to an image. This can be done by considering the IoU between the "phantom" predictions in the loss function. In addition, it seems that the model’s decision latency time can be made even longer by adding predictions (objects) from the same class. This can be done by adding a component that maximizes the number of predictions from the same type to the loss function. Another direction for future research could focus on making the attack physical by using a translucent patch on the lens of a camera (similar to \[29\]). Finally, we believe that future research should aim to develop a countermeasure that will identify and eliminate the “phantom” predictions in real time.
References

[1] N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018.

[2] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[3] N. Boucher, I. Shumailov, R. Anderson, and N. Papernot. Bad characters: Imperceptible nlp attacks. *arXiv preprint arXiv:2106.09898*, 2021.

[4] A. Boutros, M. Hall, N. Papernot, and V. Betz. Neighbors from hell: Voltage attacks against deep learning accelerators on multi-tenant fpgas. In *2020 International Conference on Field-Programmable Technology (ICFPT)*, pages 103–111. IEEE, 2020.

[5] S.-T. Chen, C. Cornelius, J. Martin, and D. H. P. Chau. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 52–68. Springer, 2018.

[6] A. E. Cinà, A. Demontis, B. Biggio, F. Roli, and M. Pelillo. Energy-latency attacks via sponge poisoning. *arXiv preprint arXiv:2203.08147*, 2022.

[7] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[10] S. Hong, Y. Kaya, I.-V. Modoranu, and T. Dumitraş. A panda? no, it’s a sloth: Slowdown attacks on adaptive multi-exit neural network inference. *arXiv preprint arXiv:2010.02432*, 2020.

[11] L. Huang, C. Gao, Y. Zhou, C. Xie, A. L. Yuille, C. Zou, and N. Liu. Universal physical camouflage attacks on object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 720–729, 2020.

[12] G. Jocher. ultralytics/yolov5: v6.0 - yolov5n ‘nano’ models, roboflow integration, tensorflow export, opencv dnn support, oct 2021. URL https://doi.org/10.5281/zenodo.5563715.

[13] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[15] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[16] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

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

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[19] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[20] I. Shumailov, Y. Zhao, D. Bates, N. Papernot, R. Mullins, and R. Anderson. Sponge examples: Energy-latency attacks on neural networks. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 212–231. IEEE, 2021.

[21] C. Sitawarin, A. N. Bhagoji, A. Mosenia, M. Chiang, and P. Mittal. Darts: Deceiving autonomous cars with toxic signs. *arXiv preprint arXiv:1802.06430*, 2018.
[22] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.

[23] S. Thys, W. Van Ranst, and T. Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, pages 0–0, 2019.

[24] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Scaled-yolov4: Scaling cross stage partial network. In Proceedings of the IEEE/cvf conference on computer vision and pattern recognition, pages 13029–13038, 2021.

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

[26] Y. Xia, D. Zhang, J. Kim, K. Nakayama, K. Zipser, and D. Whitney. Predicting driver attention in critical situations. In Asian conference on computer vision, pages 658–674. Springer, 2018.

[27] K. Xu, G. Zhang, S. Liu, Q. Fan, M. Sun, H. Chen, P.-Y. Chen, Y. Wang, and X. Lin. Adversarial t-shirt! evading person detectors in a physical world. In European conference on computer vision, pages 665–681. Springer, 2020.

[28] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 2636–2645, 2020.

[29] A. Zolfi, M. Kravchik, Y. Elovici, and A. Shabtai. The translucent patch: A physical and universal attack on object detectors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 15232–15241, 2021.