Supplementary Material: Phantom Sponges: Exploiting Non-Maximum Suppression to Attack Deep Object Detectors

1. You Only Look Once (YOLO) Object Detector

In this paper, we focus on the state-of-the-art one-stage object detector YOLO, first introduced by [4]. YOLO's architecture consists 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.

In YOLOv2 [5] the authors argued that predicting the offset from predefined anchor boxes [7] makes it easier for the network to learn.

Later, YOLOv3 [6] included multiple improvements to the architecture: (a) replacing the old backbone with a "deeper" network (referred to as *Darknet-53*) which contains residual connections [2], and (b) using multi-scale detection layers (which are referred to as *detection heads*) to predict bounding boxes at three different scales, instead of the single scale used in the first version. This architecture design established the foundation for many of the object detectors proposed in recent years [1, 8, 3].

YOLO's detection layer. The last layer of each detection head predicts a 3D tensor that encodes three parts:

- The bounding box - coordinate offsets from the anchor box.
- The objectness score - the detector's confidence that the bounding box contains an object ($Pr(\text{Object})$).
- The class scores - the detector's confidence that the bounding box contains an object of a specific class category($Pr(\text{Class}_i|\text{Object})$).

More specifically, each detection head predicts $N \times N \times [3 \times (4 + 1 + N_c)]$ candidates, where $N \times N$ is the final feature map size, 3 is the number of anchor boxes per cell in the feature map, 4 for the bounding box, 1 is the objectness score, and N_c is the number of class scores. Therefore, YOLO produces a fixed number of candidate predictions (denoted by $\mathcal{C}$) for a given image size. For example, for an image size of 640×640 pixels, the number of candidates is $|\mathcal{C}| = 3 \cdot (80 \cdot 80 + 40 \cdot 40 + 20 \cdot 20) = 25,200$ (one for each anchor in a specific cell in each final feature map).

YOLO's end-to-end detection pipeline. As explained above, YOLO outputs a fixed number of candidate predictions for a given image size. The candidates $\mathcal{C}$ are later filtered sequentially using two conditions:

- Objectness score filtering -

$$F_1 = \{c_{\text{obj score}} > T_{\text{conf}} | c \in \mathcal{C}\}. \tag{1}$$

- Unconditional class score filtering ($Pr(Class_i) = Pr(Object) \cdot Pr(Class_i|Object)$) -

$$F_2 = \{c_{\text{obj score}} \cdot \max\{c_{\text{class score}\,i}\}_{i=0}^{N_c} > T_{\text{conf}} | c \in \mathcal{C}\}. \tag{2}$$

Finally, since many candidate predictions may overlap and predict the same object, the NMS algorithm is applied to remove multiple detections.

2. UAP Example

Figure 1 presents two UAPs trained with two different λ_1 values (for YOLOv5 model). By visually examining the UAPs, it is possible to see that when setting $\lambda_1 = 0.6$, the attack detects areas in the natural images where objects commonly appear in, forcing the perturbation to add candidates on the image's sides while the center of the perturbed image remains unattacked. As opposed to this case, when setting $\lambda_1 = 1.0$, candidates are added all over the image. This is an outcome we expected, since there are naturally fewer objects in these areas (where we would usually find the sky, a road, or a sidewalk in the autonomous driving domain).

In Figure 2, we provide examples of perturbations trained using different ϵ values.

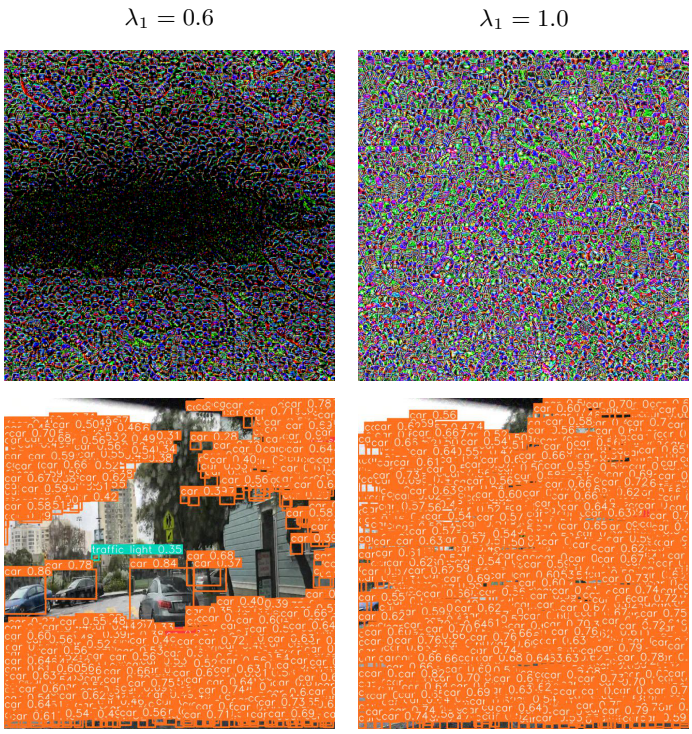


Figure 1: Top: UAPs trained with different λ_1. Bottom: perturbed images with the corresponding UAP predicted using YOLOv5.

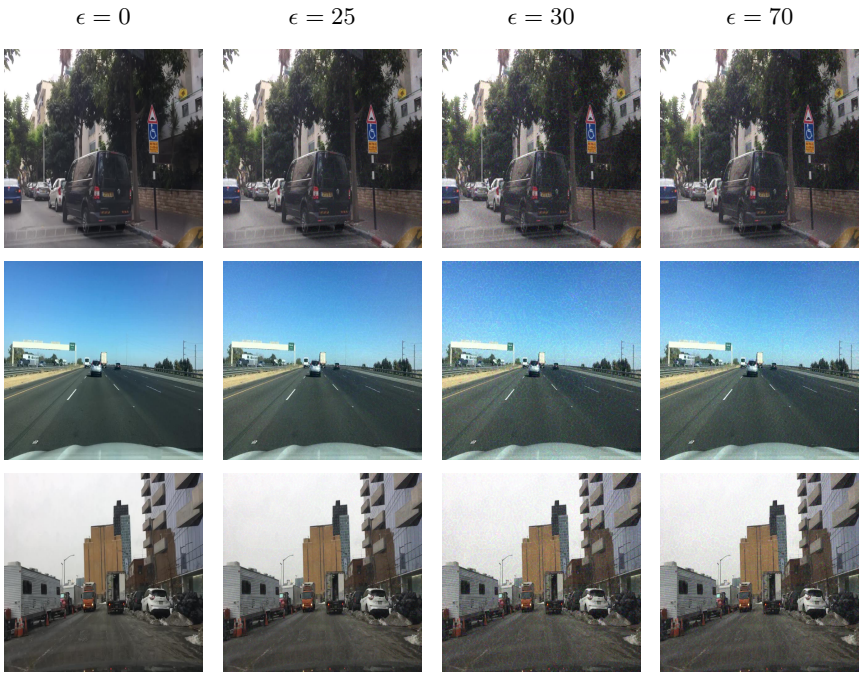


Figure 2: Examples of three images with the different UAPs applied with varying in ϵ values they were trained on, where $\lambda_1 = 1.0$.

3. Extended Results: Ensemble Experiment

In addition to the experiments presented in section 4.2 under the 'Ensemble learning' subsection, we also evaluate the effectiveness of our attack using the ensemble technique on two additional UAP's configurations. The results are reported in Table 1 and Table 2. Table 1, presents the evaluation results of UAPs with the (λ_1=0.6, λ_2=10, λ_3=0.4, ϵ=30) values. Table 2 presents the evaluation results of UAPs created with the (λ_1=0.8, λ_2=10, λ_3=0.2, ϵ=70) values. In each table, the first UAP was trained only on the YOLOv5s model, the second UAP was trained on YOLOv5s and YOLOv4 and the third UAP was trained on the three different versions of YOLO.

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

	Victim Models		
	YOLOv3	YOLOv4	YOLOv5s
	NMS time (ms) ↑ /$\lvert F(\mathcal{C}')\rvert$ ↑ / Recall ↑		
YOLOv5s	2.2 / 60 / 69%	2.2 / 40 / 55%	**13 / 9000 / 77%**
Ens_1	2.2 / 80 / 70%	**8.3 / 6300 / 72%**	**7.2 / 5400 / 80.5%**
Ens_2	**8.5 / 6700 / 75.1%**	**8 / 6000 / 74.3%**	**6.9 / 5200 / 81.9%**

Table 1: Average results for a UAP trained on different model combinations and evaluated on YOLOv3, YOLOv4, and YOLOv5. Ens_1=YOLOv4+YOLOv3, Ens_2=YOLOv5+YOLOv4+YOLOv3,
configuration: : $(\epsilon, \lambda_1, \lambda_2) = (30, 0.6, 10)$.

	Victim Models		
	YOLOv3	YOLOv4	YOLOv5s
	NMS time (ms) ↑ /$\lvert F(\mathcal{C}')\rvert$ ↑ / Recall ↑		
YOLOv5s	2.1 / 60 / 53.4%	2.2 / 40 / 31.6%	**25.8 / 16000 / 45.3%**
Ens_1	2.1 / 70 / 56.9%	**14.7 / 10100 / 50.3%**	**19.8 / 12300 / 56.4%**
Ens_2	**18.4 / 12000 / 56.7%**	**13.9 / 9200 / 52.1%**	**15.1 / 10200 / 61%**

Table 2: Average results for a UAP trained on different model combinations and evaluated on YOLOv3, YOLOv4, and YOLOv5. Ens_1=YOLOv4+YOLOv3, Ens_2=YOLOv5+YOLOv4+YOLOv3,
configuration: : $(\epsilon, \lambda_1, \lambda_2) = (70, 0.8, 10)$.

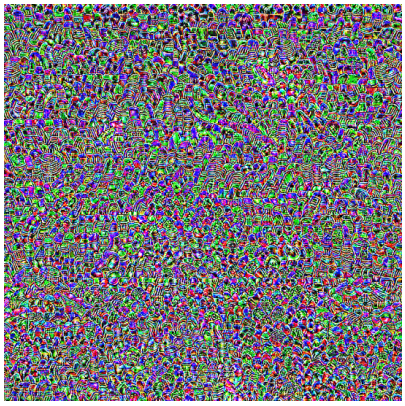
4. Extended Results: Target Class

In Table 3 we present evaluation results for UAPs (with ϵ=70, λ_1=1, λ_2=10, λ_3=0 values) created for different target classes. It can be seen that the attack manages to create efficient UAPs for different target classes (*i.e.*, $\sim 20K$ objects are passed to the NMS step).

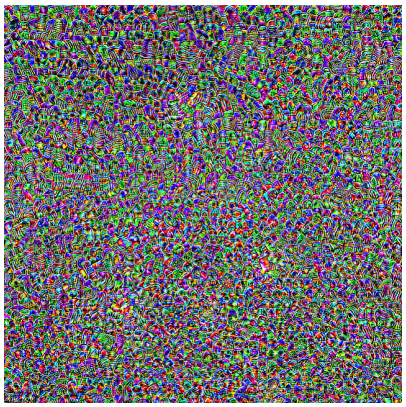
	car	person	bicycles
	NMS time (ms) $\uparrow$ / $\lvert F(\mathcal{C}')\rvert \uparrow$ / Recall $\uparrow$		
clean	2.2 / 80 / 100%		
config	37.1 / 19200 / 18%	37.9 / 19500 / 10%	35.8 / 18800 / 14%

Table 3: Average results of UAPs created for three different target classes: 'car', 'person' and 'bicycles'. config: : $(\epsilon, \lambda_1, \lambda_2) = (70, 1, 10)$.

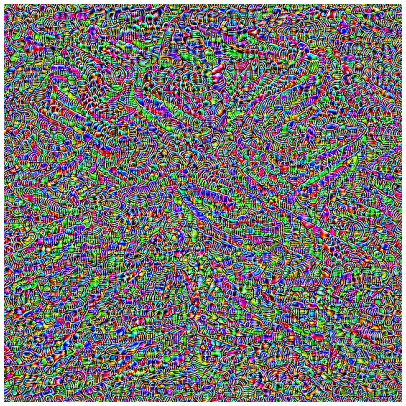
As mentioned in Section 4.2, the UAPs' patterns for different target classes seem to consist of tiny objects that resemble the UAPs' target class. UAPs for the different target classes are presented in Figure 3.



(a) UAP created for the 'car' target class.



(b) UAP created for the 'person' target class.



(c) UAP created for the 'bicycle' target class.

Figure 3: UAPs created for different target classes, with the $(\epsilon, \lambda_1, \lambda_2) = (70, 1, 10)$ values.

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] Glenn Jocher. ultralytics/yolov5: v6.0 - yolov5n 'nano' models, roboflow integration, tensorflow export, opencv dnn support, oct 2021.

[4] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali 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.

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

[6] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[7] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[8] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark 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.