TRANSFERABLE ADVERSARIAL EXAMPLES FOR ANCHOR FREE OBJECT DETECTION

Quanyu Liao[1], *Xin Wang*[2†], *Bin Kong*[2], *Siwei Lyu*[3], *Bin Zhu*[4], *Youbing Yin*[2], *Qi Song*[2], *Xi Wu*[1†]

[1] Chengdu University of Information Technology, Chengdu, China
[2] Keya Medical, Seattle, USA
[3] University at Buffalo, State University of New York, USA
[4] Microsoft Research Asia, Beijing, China

ABSTRACT

Deep neural networks have been demonstrated to be vulnerable to adversarial attacks: subtle perturbations can completely change prediction results. The vulnerability has led to a surge of research in this direction, including adversarial attacks on object detection networks. However, previous works are dedicated to attacking anchor-based object detection models. In this work, we present the first adversarial attack on anchor-free object detection models. It conducts category-wise, instead of previously instance-wise, attacks on object detectors, and leverages high-level semantic information to efficiently generate transferable adversarial examples, which can also be transferred to attack other object detectors, even anchor-based detectors such as Faster R-CNN. Experimental results on two benchmark datasets demonstrate that our method achieves the state-of-the-art performance and transferability.

Index Terms— Category-wise attacks, adversarial attacks, object detection, anchor-free object detection

1. INTRODUCTION

The development of deep neural network has significantly improved the performance of many computer vision tasks. However, many recent works show that deep-learning-based algorithms are vulnerable to adversarial attacks [1, 2, 3, 4, 5]. The vulnerability of deep networks is observed in many different problems [6, 7], including object detection, one of the most fundamental tasks in computer vision.

Regarding the investigation of the vulnerability of deep models in object detection, previous efforts mainly focus on classical anchor-based networks such as Faster-RCNN [8]. However, the performance of these anchor-based networks is limited by the choice of anchor boxes. Fewer anchors lead to faster speed but lower accuracy. Thus, advanced anchor-free models such as CornerNet [9] and CenterNet [10] are becoming increasingly popular, achieving competitive accuracy with traditional anchor-based models yet with faster speed and stronger adaptability. However, to the best of our knowledge, there is no published work on investigating the vulnerability of anchor-free networks.

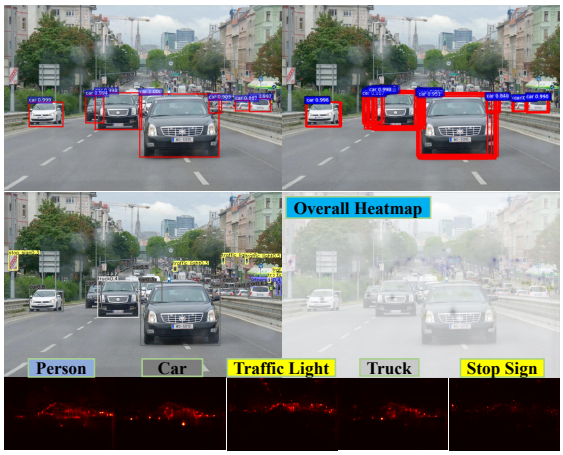

Fig. 1. **First row:** The detected results (left) and the proposals (right) of Faster R-CNN [8]. **Second row:** The detected results (left) and the overall heatmap (right) of CenterNet [10]. **Third row:** Selected target pixels (red) for each category by our method.

Previous work DAG [11] implements high level white box attack on the FasterRCNN, but DAG is hardly to complete an effective black-box attack. DAG also has the disadvantages of high time-consuming, these two shortcomings make DAG difficult to be used in real scenes. These two shortcomings of DAG principally because DAG only attacks one proposal in each attack iteration. It will make the generated adversarial perturbation only effective for one proposal, which leads to bad transferring attack performance and consumes an amount of iterations to attack all objects.

Meanwhile, attack an anchor-based detector is unlike to attack an anchor-free detector, which select top proposals from a set of anchors for the objects, anchor-free object detectors detect objects by finding objects' keypoints via the heatmap mechanism (see Fig. 1), using them to generate corresponding bounding boxes, and selecting the most probable keypoints to generate final detection results. This process is completely different from anchor-based detectors, making anchor-based adversarial attacks unable to directly attack

† Corresponding authors: Xin Wang (xinw@keyamedna.com), Xi Wu (xi.wu@cuit.edu.cn).

978-1-6654-3864-3/21/$31.00 ©2021 IEEE

anchor-free detectors.

To solve above two problems, we propose a novel algorithm, *Category-wise Attack (CW-Attack)*, to attack anchor-free object detectors. It attacks all instances in a category simultaneously by attacking a set of target pixels in an image, as shown in Fig. 1. The target pixel set includes not only all detected pixels, which are highly informative pixels as they contain higher-level semantic information of the objects, but also "runner-up locations pixels" that have a high probability to become rightly detected pixels under small perturbations. Our approach guarantees success of adversarial attacks. Our CW-Attack is formulated as a general framework that minimizes L_p of perturbations, where L_p can be L_0, L_1, L_2, L_∞, etc., to flexibly generate different types of perturbations, such as dense or sparse perturbations. Our experimental results on two benchmark datasets [12, 13], show that our method outperforms previous state-of-the-art methods and generates robust adversarial examples with superior transferability.

Our CW-Attack disables object detection by driving feature pixels of objects into wrong categories. This behavior is similar to but the essence is completely different from attacking semantic segmentation approaches [11]. First, they have different targets to optimize: the goal is to change the category of an object's bounding box in our attack and a detected pixel's category in attacking semantic segmentation. Second, they have different relationships to attack success: once pixels have changed their categories, the attack is successful for attacking semantic segmentation but not yet for our attack. As we will see in Fig. 3, objects can still be detected even when all heatmap pixels have been driven into wrong categories.

This paper has the following major contributions: **(i)** We propose the first adversarial attack on anchor-free object detection. It attacks all objects in a category simultaneously instead of only one object at a time, which avoids perturbation over-fitting on one object and increases transferability of generated perturbations. **(ii)** Our CW-Attack is designed as a general L_p norm optimization framework. When minimizing perturbation's L_0 norm (see Sec. 2), it generates sparse adversarial samples by only modifying less than 1% pixels. While minimizing its L_∞ norm (detail in supplement materials), it can attack all objects of *all* categories simultaneously, which further improves the attacking efficiency. **(iii)** Our method generates more transferable and robust adversarial examples than previous attacks. It achieves the state-of-the-art attack performance for both white-box and black-box attacks on two public benchmark datasets, MS-COCO and PascalVOC.

2. OUR CATEGORY-WISE ATTACK

In this section, we first define the optimization problem of attacking anchor-free detectors and then provide a detailed description of our Category-Wise Attack (CW-Attack).

Problem Formulation. Suppose there exist k object categories, $\{C_1, C_2, ..., C_k\}$, with detected object instances. We use S_{target} to denote the target pixel set of category C_{target} whose detected object instances will be attacked, leading to k target pixel sets: $\{S_1, S_2, ..., S_k\}$. The category-wise attack for anchor-free detectors is formulated as the following constrained optimization problem:

$$\begin{aligned} \underset{r}{\text{minimize}} \quad & \|r\|_p \\ s.t. \quad & \forall k, s \in S_{target} \in \{S_1, S_2, ..., S_k\} \\ & \arg\max_n\{f_n(x+r, s)\} \neq C_{target} \end{aligned} \tag{1}$$

where r is an adversarial perturbation, $\|\cdot\|_p$ is the L_p norm, $p \in \{0, 1, 2, \infty\}$, x is a clean input image, $x+r$ is an adversarial example, $f(x+r, s)$ is the classification score vector (logistic) and $f_n(x+r, s)$ is its n^{th} value, $\arg\max_n\{f_n(x+r, s)\})$ denotes the predicted object category on a target pixel $s \in S_{target}$ of adversarial example $x+r$.

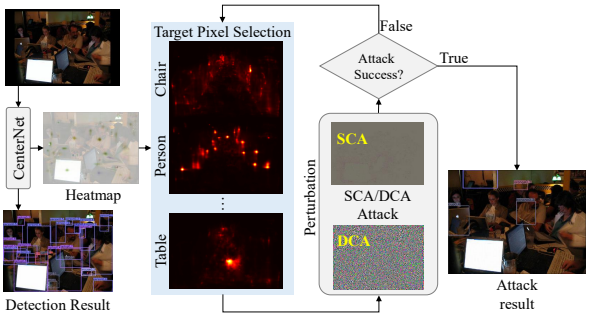

Fig. 2. CW-Attack overview. Target pixel sets $\{S_1, S_2, ..., S_k\}$ are first extracted from the heatmap for all object categories. SCA or DCA is then used to generates perturbations, depending on minimizing perturbation's L_0 or L_∞ norm. Finally, we check whether the attack is successful. If not, a new perturbation is generated from the current adversarial example in the next iteration.

The overview of the proposed CA-Attack is shown in Fig. 2. In the following description of our method, we assume the task is a non-target multi-class attack. If the task is a target attack, our method can be described in a similar manner.

Category-wise Target Pixel Set Selection. In solving our optimization problem (1), it is natural to use all *detected pixels* of category C_{target} as target pixel set S_{target}. The detected pixels are selected from the heatmap of category C_{target} generated by an anchor-free detector such as CenterNet [10] with their probability scores higher than the detector's preset visual threshold and being detected as right objects. Unfortunately, it does not work. After attacking all detected pixels into wrong categories, we expect that the detector should not detect any correct object, yet our experiments with CenterNet turn out that it still can.

Further investigation reveals two explanations: **(1)** Neighboring background pixels of the heatmap not attacked can become detected pixels with the correct category. Since their detected box is close to the old detected object, CenterNet can still detect the object even though all the previously detected pixels are detected into wrong categories. An example

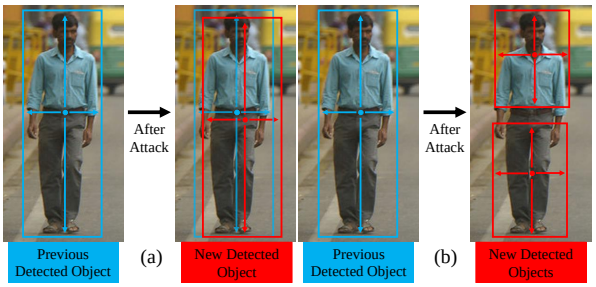

Fig. 3. **Blue** points denote originally detected keypoints before the attack. **Red** points denote newly detected keypoint after the attack. **(a)-Left & (b)-Left:** A detected object and a detected keypoint at the center of the person. **(a)-Right:** Result of attacking only detected pixels. After attacking all detected pixels, a neighboring pixel of the previously detected keypoint is detected as the correct object during the attack. **(b)-Right:** Result of attacking only detected pixels. After attacking all detected pixels, the centers of the top half and the bottom half of the person appear as newly detected keypoints still detected as a person. mAP is barely reduced in both cases.

is shown in Fig. 3-(a). **(2)** CenterNet regards center pixels of an object as keypoints. After attacking detected pixels located around the center of an object, newly detected pixels may appear in other positions of the object, making the detector still be able to detect multiple local parts of the correct object with barely reduced mAP. An example is shown in Fig. 3-(b).

Pixels that can produce one of the above two changes are referred to as *runner-up locations pixels*. We find that almost all *runner-up locations pixels* have a common characteristic: their probability scores are only a little below the visual threshold. Based on this characteristic, our category-wise attack sets an attacking threshold, t_{attack}, lower than the visual threshold, and then selects all the pixels from the heatmap whose probability score is above t_{attack} into S_k. This makes S_k include all detected pixels and *runner-up locations pixels*. Perturbations generated in this way can also improve robustness and transferable attacking performance.

Sparse Category-wise Attack. The goal of the sparse attack is to fool the detector while perturbing a minimum number of pixels in the input image. It is equivalent to setting $p = 0$ in our optimization problem (1) i.e. minimizing $\|r\|_0$ according to S_{target}. Unfortunately, this is an NP-hard problem. To solve this problem, SparseFool [14] relaxes this NP-hard problem by iteratively approximating the classifier as a local linear function in generating sparse adversarial perturbations for image classification.

Motivated by the success of SparseFool on image classification, we propose Sparse Category-wise Attack (SCA) to generate sparse perturbations for anchor-free object detectors. It is an iterative process. In each iteration, one target pixel set is selected from category-wise target pixel sets to attack.

More specifically, given an input image x and current category-wise target pixel sets $\{S_1, S_2, ..., S_k\}$, SCA selects the pixel set that has the highest probability score from $\{S_1, S_2, ..., S_k\}$ as target pixel set S_{target} and use *Category-Wise DeepFool (CW-DF)*[1] to generate dense adversarial example x^B by computing perturbations on S_{target}. The CW-DF adapts Deepfool [15] to become a category-wise attack algorithm for anchor-free object detection.

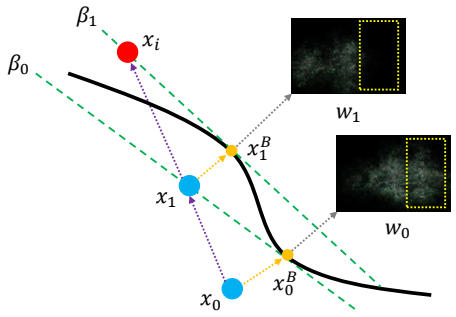

Fig. 4. Illustration of SCA with the 'Car' category of Fig. 1. The black solid line denotes the real decision boundary of the object detector. Blue points denote adversarial examples that have not attacked all objects successfully. Red points denote adversarial examples that have already attacked all objects successfully. This figure illustrates two iterations of the attack, $x_0 \rightarrow x_1$ and $x_1 \rightarrow x_i$. Take $x_0 \rightarrow x_1$ for example, SCA first generates dense adversarial example x_0^B (yellow points) by *CW-DF* and approximated linear decision boundary β_0 (green dash lines). Then it uses *LinearSolver* (purple dash lines) to add a sparse perturbation to support x_0 to approximate decision boundary β by satisfying $\beta = \{x : w^T(x - x^B) = 0\}$ until a validate sparse adversarial example x_1 is obtained. The two images are the visualization of the normal vector w, the yellow boxes indicates the weights for the 'Car' object are reduced.

Then, SCA uses the *ApproxBoundary* to approximate the decision boundary, which is locally approximated with a hyperplane β passing through x^B:

$$\beta \triangleq \{x : w^T(x - x^B) = 0\}, \tag{2}$$

where w is the normal vector of hyperplane β and approximated with the following equation [14]:

$$w := \nabla \sum_{i=1}^{n} f_{\text{argmax}_n f_n(x^B,s)}(x^B, s) - \nabla \sum_{i=1}^{n} f_{\text{argmax}_n f_n(x,s)}(x^B, s). \tag{3}$$

A sparser adversarial perturbation can then be computed via the *LinearSolver* process [14]. The process of generating perturbations through the *ApproxBoundary* and the *LinearSolver* of SCA is illustrated in Fig. 4.

After attacking S_{target}, SCA uses *RemovePixels* to update S_{target} by removing the pixels that are no longer detected. Specifically, it takes $x_{i,j}$, $x_{i,j+1}$, and S_{target} as input. *RemovePixels* first generates a new heatmap for perturbed image $x_{i,j+1}$ with the detector. Then, it checks whether the probability score of each pixel in S_{target} is still higher than t_{attack}

[1] See supplement material for the detail of the *CW-DF*, *ApproxBoundary*, *LinearSolver* and *RemovePixels*.

Algorithm 1 Sparse Category-wise Attack (SCA)

Input: image x, target pixel set $\{S_1, S_2, ..., S_k\}$,
available categories $\{C_1, C_2, ..., C_k\}$

Output: perturbation r

Initialize: $x_1 \leftarrow x, i \leftarrow 1, j \leftarrow 1, S_0 \leftarrow S$

while $\{S_1, S_2, ..., S_k\} \notin \varnothing$ **do**

$target = \text{argmax}_k \sum_{s \in S_k} \text{softmax}_{C_k}\ f(x_i, s)$

$S_{target,1} \leftarrow S_{target}$

$x_{i,j} \leftarrow x_i$

while $j \leq M_s$ or $S_{target,j} \in \varnothing$ **do**

$x_j^B = \text{CW-DF}\ (x_{i,j})$

$w_j = \text{ApproxBoundary}\ (x_j^B, S_{target,j})$

$x_{i,j+1} = \text{LinearSolver}\ (x_{i,j}, w_j, x_j^B)$

$S_{target} = \text{RemovePixels}\ (x_{i,j}, x_{i,j+1}, S_{target})$

$j = j + 1$

end while

$x_{i+1} \leftarrow x_{i,j}$

$i = i + 1$

end while

return $r = x_i - x_1$

on the new heatmap. The pixels whose probability score is lower than t_{attack} are removed from S_{target}, while the remaining pixels are retained in S_{target}. Target pixel set S_{target} is thus updated. If $\{S_1, S_2, ..., S_k\} \in \varnothing$, which indicates that no original object can be detected after the attack, the attack for all objects of x is successful and we output the generated adversarial example.

The SCA algorithm is summarized in Alg. 1. Note that SCA will not fall into an endless loop. In an iteration, if SCA fails to attack any pixels of S_{target} in the inner loop, SCA will attack the same S_{target} in the next iteration. During this process, SCA keeps accumulating perturbations on these pixels, with the probability score of each pixel in S keeping reducing, until the probability score of every pixel in S_{target} is lower than t_{attack}. By then, S_{target} is attacked successfully.

Dense Category-wise Attack. It is interesting to investigating our optimization problem (1) for $p = \infty$. FGSM and PGD are two most widely used attacking approaches by minimizing L_∞.

As we mentioned before, PGD iteratively takes smaller steps in the direction of the gradient. It achieves a higher attack performance and generates smaller L_∞ perturbations than FGSM. Our adversarial perturbation generation procedure is base on PGD and is named as *Dense Category-wise Attack* (DCA) since it generates dense perturbations compared to SCA.

	Method	Network	Clean	Attack	ASR (%)	Time (s)
PascalVOC	DAG	FR	0.70	0.050	0.92	9.8
	UEA	FR	0.70	0.050	0.93	–
	SCA	R18	0.67	0.060	0.91	20.1
	SCA	DLA34	0.77	0.110	0.86	91.5
	DCA	R18	0.67	0.070	0.90	0.3
	DCA	DLA34	0.77	0.050	**0.94**	0.7
MS-COCO	SCA	R18	0.29	0.027	0.91	50.4
	SCA	DLA34	0.37	0.030	0.92	216.0
	DCA	R18	0.29	0.002	0.99	**1.5**
	DCA	DLA34	0.37	0.002	**0.99**	2.4

Table 1. White-box performance comparison. The top row denotes the metrics. Clean and Attack denote the mAP of clean input and adversarial examples, respectively. Time is the average time to generate an adversarial example.

Given an input image x and category-wise target pixel sets $\{S_1, S_2, ..., S_k\}$, DCA[2] applies two iterative loops to generate adversariabl perturbations: each inner loop iteration j computes the local gradient for each category S_j and generates a total gradient for all existing categories; while each outer loop iteration i uses the total gradient generated in the inner loop iteration to generate a perturbation for all the objects of all existing categories.

Specifically, in each inner loop iteration j, DCA computes the gradient for every pixel in S_j to attack all instances in C_j as follows: DCA first computes the total loss of all pixels in target pixel set S_j corresponding to each category C_j:

$$loss_{sum} = \sum_{s_n \in S_j} \text{CrossEntropy}\ (f(x_i, s_n),\ C_j), \tag{4}$$

and then computes local adversarial gradient r_j of S_j on $loss_{sum}$ and normalizes it with L_∞, yielding r'_j:

$$r_j = \bigtriangledown_{x_i} loss_{sum}, \quad r'_j = \frac{r_j}{\|r_j\|_\infty}. \tag{5}$$

After that, DCA adds up all r'_j to generate total adversarial gradient G. Finally, in the outer loop iteration i, DCA computes perturbation $pert_i$ by applying $sign$ operation to the total adversarial gradient G [16]:

$$pert_i = \frac{\epsilon_D}{M_D} \cdot sign(G), \tag{6}$$

where M_D denotes the maximum number of cycles of the outer loop, term $\frac{\epsilon_D}{M_D}$ is optimal max-norm constrained weight to constraint the amplitude of $pert_i$ [17]. At the end of the outer loop, DCA uses $RemovePixels$ to remove the target pixels that have already been attacked successfully on x_{i+1} from of $\{S_1, S_2, ..., S_k\}$.

Since an adversarial perturbation in DCA is generated from normalized adversarial gradients of all categories' objects, DCA attacks all object instances of all the categories simultaneously. It is more efficient than SCA.

[2]DCA is summarized in Alg. 5 in the supplement materials. A figure (Fig. 1 in the supplement materials) that shows the perturbation generation process of DCA is also included in the supplement materials

Fig. 5. Qualitative comparison between DAG and our methods. Each row is an example. **Column 1:** Detection results of clean inputs on CenterNet. **Column 2&3:** DAG perturbations and DAG attacked results on Faster-RCNN. **Column 4&5:** DCA perturbations and DCA attacked results on CenterNet. **Column 6&7:** SCA perturbations and SCA attacked results on CenterNet. Note that in **Column 6**, the percentage of the changed pixels for the SCA perturbation is 3.4% and 3.51% from top to bottom. We can see that the perturbations of DCA and SCA are smaller than the DAG's. Notably, the proposed SCA changes only a few percentage of pixels. To better show perturbations, we have multiplied the intensity of all perturbation images by 10.

From \ To	Resdcn18		DLA34		Resdcn101		Faster-RCNN		SSD300	
	mAP	ATR	mAP	ATR	mAP	ATR	mAP	ATR	mAP	ATR
Clean	0.67	–	0.77	–	0.76	–	0.71	–	0.77	–
DAG [11]	0.65	0.19	0.75	0.16	0.74	0.16	0.60	1.00	0.76	0.08
R18-DCA	0.10	1.00	0.62	0.23	0.65	0.17	0.61	0.17	0.72	0.08
DLA34-DCA	0.50	0.28	0.07	1.00	0.62	0.2	0.53	0.28	0.67	0.14
R18-SCA	0.31	1.00	0.62	0.36	0.61	0.37	0.55	0.42	0.70	0.17
DLA34-SCA	0.42	0.90	0.41	1.00	**0.53**	**0.65**	**0.44**	**0.82**	**0.62**	**0.42**

Table 2. Black-box attack results on the PascalVOC dataset. **From** in the leftmost column denotes the models where adversarial examples are generated from. **To** in the top row means the attacked models that adversarial examples transfer to.

From \ To	Resdcn18		DLA34		Resdcn101		CornerNet	
	mAP	ATR	mAP	ATR	mAP	ATR	mAP	ATR
Clean	0.29	–	0.37	–	0.37	–	0.43	–
R18-DCA	0.01	1.00	0.29	0.21	0.28	0.25	0.38	0.12
DLA34-DCA	0.10	0.67	0.01	1.00	0.12	0.69	0.13	0.72
R18-SCA	0.11	1.00	0.27	0.41	0.24	0.57	0.35	0.30
DLA34-SCA	0.07	0.92	0.06	1.00	**0.09**	**0.92**	**0.12**	**0.88**

Table 3. Black-box attack results on the MS-COCO dataset. **From** in the leftmost column denotes the models where adversarial examples are generated from. **To** in the top row means the attacked models that adversarial examples transfer to.

3. EXPERIMENTAL EVALUATION

Dataset. Our method is evaluated on two object detection benchmarks: PascalVOC [12] and MS-COCO [13].

Evaluation Metrics. i) *Attack Success Rate (ASR)*: $ASR = 1 - mAP_{attack}/mAP_{clean}$, where mAP_{attack} and mAP_{clean} are the mAP of the adversarial example and the clean input, respectively. **ii)** *Attack Transfer Ratio (ATR)*: It is evaluated as follows: $ATR = ASR_{target}/ASR_{origin}$, where ASR_{target} is the ASR of the target detector to be black-box attacked, and ASR_{origin} is the ASR of the detector that generates the adversarial example. **iii)** *Perceptibility*: The perceptibility of an adversarial perturbation is quantified by its P_{L_2} and P_{L_0} norm. **i)** P_{L_2}**:** Formally, $P_{L_2} = \sqrt{1/k \sum r_k^2}$, where the k is the number of the pixels. We normalize the P_{L_2} from $[0, 255]$ to $[0, 1]$. **ii)** P_{L_0}**:** P_{L_0} computed by measuring the proportion of changed pixels after attack.

White-Box Attack[3]**.** We conducted white-box attacks on two popular object detection methods. Both use CenterNet but with different backbones: one, denoted as R18, with Resdcn18 [18] and the other, DLA34 [19], with Hourglass [20].

Network	P_{L_2}	P_{L_0}
DAG	2.8×10^{-3}	$\geq 99.0\%$
R18-Pascal	5.1×10^{-3} (DCA)	0.22% (SCA)
DLA34-Pascal	5.1×10^{-3} (DCA)	0.27% (SCA)
R18-COCO	4.8×10^{-3} (DCA)	0.39% (SCA)
DLA34-COCO	5.2×10^{-3} (DCA)	0.65% (SCA)

Table 4. Perceptibility of the perturbation.

Table 1 shows the white-box attack results on both PascalVOC and MS-COCO. For comparison, it also contains the reported attack results of DAG and UEA attacking Faster-RCNN with VGG16 [21] backbone, denoted as FR, on PascalVOC. There is no reported attack performance on MS-COCO for DAG and UEA. UEA's average attack time in Table 1 is marked as "–" (unavailable) because, as a GAN-based apporach, UEA's average attack time should include GAN's training time, which is unavailable. Compare with optimization-based attack methods [11], a GAN-based attack method consumes a lot of time for training and needs to re-train a new weight to attack another task. Thus a GAN-based attack method sacrifices attack flexibility, and it can not be used in some scenarios with high flexibility requirements.

The top half of Table 1 shows the attack performance on PascalVOC. We can find that: **(1)** DCA achieves higher ASR than DAG and UEA, and SCA achieves the best ASR performance. **(2)** DCA is 14 times faster than DAG. We cannot compare with UEA since its attack time is unavailable. Qualitative comparison between DAG and our methods in shown in Fig. 5. The bottom half of Table 1 shows the attack performance of our methods on MS-COCO. SCA's ASR on both R18 and DLA34 is in the same ballpark as the ASR of DAG and UEA on PascalVOC, while DCA achieves the highest ASR, 99.0%. We conclude that both DCA and SCA achieve the state-of-the-art attack performance.

Black-Box Attack and Transferability. Black-box attacks can be classified into two categories: cross-backbone and cross-network. For cross-backbone attacks, we evaluate the transferability with Resdcn101 [18] on PascalVOC and MS-

[3]More experimental results and hyperparameters analysis of DCA and SCA are also shown in the supplement material.

COCO. For cross-network attacks, we evaluate with not only anchor-free object detector CornerNet [9], but also two-stage anchor-based detectors, Faster-RCNN [8] and SSD300 [22]. Faster-RCNN and SSD300 are tested on PascalVOC. CornerNet is tested on MS-COCO with backbone Hourglass [20].

To simulate a real-world attack transferring scenario, we generate adversarial examples on the CenterNet and save them in the JPEG format, which may cause them to lose the ability to attack target models [23] as some key detailed information may get lost due to the lossy JPEG compression. Then, we reload them to attack target models and compute mAP. This process has a more strict demand on adversarial examples but should improve their transferability.

i) *Attack transferability on PascalVOC.* Adversarial examples are generated on CenterNet with Resdcn18 and DLA34 backbones for both SCA and DCA. For comparison, DAG is also used to generate adversarial examples on Faster-RCNN. These adversarial examples are then used to attack the other four models. All the five models are trained on PascalVOC. Table 2 shows the experimental results. We can see from the table that adversarial examples generated by our method can successfully transfer to not only CenterNet with different backbones but also completely different types of object detectors, Faster-RCNN and SSD. We can also see that DCA is more robust to the JPEG compression than SCA, while SCA achieves higher ATR than DCA in the black-box test. Table 2 indicates that DAG is sensitive to the JPEG compression, especially when its adversarial examples are used to attack Faster-RCNN, and has a very poor transferability in attacking CenterNet and SSD300. We conclude that both DCA and SCA perform better than DAG on both transferability and robustness to the JPEG compression.

ii) *Attack Transferability on MS-COCO.* Similar to the above experiments, adversarial examples are generated on Centernet with Resdcn18 and DLA34 backbones and then used to attack other object detection models. The experimental results are summarized in Table 3. The table indicates that generated adversarial examples can attack not only CenterNet with different backbones but also CornerNet.

Perceptibility. The perceptibility results of adversarial perturbations of DCA and SCA are shown on Table 4. We can see from Table 4 that P_{L_0} of SCA is lower than 1%, meaning that SCA can fool the detectors by changing only a few number of pixels. Although DCA has a higher P_{L_2} than DAG, perturbations generated by DCA are still hard to be distinguished by humans. We also provide qualitative examples for comparison in Fig. 5.

4. CONCLUSION

In this paper, we propose a category-wise attack to attack anchor-free object detectors. It manifests in two forms, SCA and DCA, when minimizing the L_0 and L_∞ norms, respectively. Both SCA and DCA focus on global and high-level semantic information to generate adversarial perturbations. Our experiments with CenterNet on two public object detection benchmarks indicate that both SCA and DCA achieve the state-of-the-art attack performance and transferability.

5. REFERENCES

[1] Nicholas Carlini and etc. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE SP*, 2017.

[2] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li, "Boosting adversarial attacks with momentum," in *CVPR*, 2018.

[3] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille, "Improving transferability of adversarial examples with input diversity," in *CVPR*, 2019.

[4] Francesco Croce and Matthias Hein, "Minimally distorted adversarial examples with a fast adaptive boundary attack," in *ICML*, 2020.

[5] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu, "Evading defenses to transferable adversarial examples by translation-invariant attacks," in *CVPR*, 2019.

[6] Avishek Joey Bose and Parham Aarabi, "Adversarial attacks on face detectors using neural net based constrained optimization," in *MMSP*.

[7] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Chau, "Robust physical adversarial attack on faster r-cnn object detector," in *ECMLKDD*, 2018.

[8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NIPS*, 2015.

[9] Hei Law and Jia Deng, "Cornernet: Detecting objects as paired keypoints," *IJCV*, 2019.

[10] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl, "Objects as points," *CVPR*, 2019.

[11] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille, "Adversarial examples for semantic segmentation and object detection," in *ICCV*, 2017.

[12] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman, "The pascal visual object classes challenge: A retrospective," *IJCV*, vol. 111, no. 1, 2015.

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "Microsoft coco: Common objects in context," in *ECCV*. Springer, 2014.

[14] Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard, "Sparsefool: a few pixels make a big difference," in *CVPR*.

[15] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *CVPR*, 2016.

[16] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu, "Towards deep learning models resistant to adversarial attacks.," *ICLR*, 2018.

[17] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, "Explaining and harnessing adversarial examples," 2015.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[19] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell, "Deep layer aggregation," in *CVPR*, 2018.

[20] Alejandro Newell, Kaiyu Yang, and Jia Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*. Springer, 2016.

[21] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR*, 2014.

[22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, "Ssd: Single shot multibox detector," in *ECCV*. Springer, 2016.

[23] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy, "A study of the effect of jpg compression on adversarial images," 2016.