Fast Local Attack: Generating Local Adversarial Examples for Object Detectors

Quanyu Liao[1], Xin Wang* *Member, IEEE*[2],
Bin Kong[2], Siwei Lyu *Senior Member, IEEE*[3], Youbing Yin[2], Qi Song[2], and Xi Wu*[1]

[1]Chengdu University of Information Technology, Chengdu, China
[2]CuraCloud Corporation, Seattle, USA
[3]SUNY Albany, NY, USA

***Abstract*—The deep neural network is vulnerable to adversarial examples. Adding imperceptible adversarial perturbations to images is enough to make them fail. Most existing research focuses on attacking image classifiers or anchor-based object detectors, but they generate globally perturbation on the whole image, which is unnecessary. In our work, we leverage higher-level semantic information to generate high aggressive local perturbations for anchor-free object detectors. As a result, it is less computationally intensive and achieves a higher black-box attack as well as transferring attack performance. The adversarial examples generated by our method are not only capable of attacking anchor-free object detectors, but also able to be transferred to attack anchor-based object detector.**

***Index Terms*—adversarial attack, object detection, fast local attack**

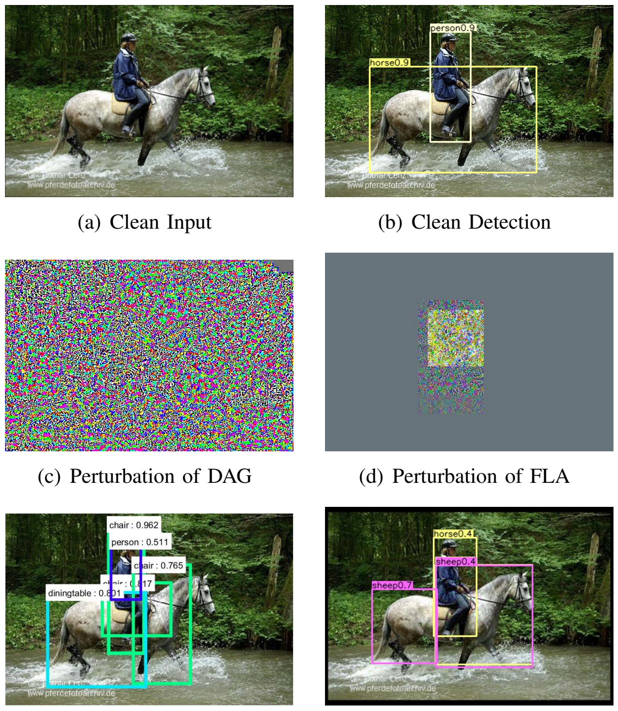
I. Introduction

The development of deep neural networks (DNNs) supports researchers to achieve unprecedented high performance in various computer vision problems. Nevertheless, these deep learning-based algorithms are notoriously vulnerable to adversarial examples [8]: adding some imperceptible adversarial perturbations is enough to make them fail. This phenomenon can be found in different applications [1], [2], [5], [11], [13], [17], [21], [24], including classification, object detection, etc. In this paper, we specifically focus on the adversarial attack of object detectors.

Existing object detectors can be broadly categorized into two groups: anchor-based or anchor-free detectors. Recent anchor-free object detectors [10], [12], [26], [27] achieve competitive performance with traditional anchor-base detectors. Additionally, anchor-free detectors are structurally simpler and more computationally efficient than anchor-based detectors. Anchor-based detectors have dominated object detection due to their superior performance. However, adversarial perturbations to this type of detectors have not been explored.

All the existing research focus on the adversarial attack of anchor-based detectors, such as DAG [23] and UEA [22]. However, Existing methods suffer from three major shortcomings: **1)** Most of the attack methods [8], [16], [18], [23] generate global perturbations, including the background. However, most of the pixels of these perturbations are useless to fool the detectors. On the contrary, they increase the perceptibility of the perturbations. **2)** The generated adversarial examples have the poor transferring ability, *e.g.,* the adversarial examples generated by DAG on Faster-RCNN can only attack Faster-RCNN models and the generated examples can hardly be transferred to attack other object detectors. **3)** They only attack one proposal at one time or relies on training, which is extremely computational consumption. Thus, it is desired to investigate a special attacking scheme for anchor-free detectors that have more local perturbations.

In this paper, inspired by the fact that the key information



(a) Clean Input (b) Clean Detection
(c) Perturbation of DAG (d) Perturbation of FLA
(e) Attack Result of DAG (f) Attack Result of FLA

Fig. 1. Comparison between global (DAG) and local (FLA) attack methods for object detectors. (a) and (b) are the clean input and its detection result on Centernet. In the global attack method, the adversarial examples are generated for all image pixels (c), including the background. Our Fast Locally Attack (FLA) generates perturbations only local to the target objects (d). Using this technique, the attack result of FLA (f) is superior to DAG (e). Especially in (d), the FLA only generate small-scale locally perturbation around the center that completes the attack of both human and horse.

*Xin Wang (xinw@curacloudcorp.com) and Xi Wu (xi.wu@cuit.edu.cn) are corresponding authors.

lies in or around objects, we propose a new method, **F**ast **L**ocally **A**ttack (FLA), to generate locally adversarial perturbation for anchor-free object detectors. "Fast" denotes that our method generates adversarial examples with lower time consumption than previous work. For this purpose, we focus on high-level semantic information to generate adversarial examples. We attack all objects in each iteration instead of a single object in each iteration, which can reduce the amount of iteration and improve the transferring attack performance of our method. As for "Locally", which means our method generates locally perturbation that only changes little-scale pixels around the detected objects. Locally perturbation can increase the imperceptible of perturbation without reducing attack performance. Examples in Fig. 1 demonstrates the adversarial examples from our method and compare with DAG's examples. Experimental results show that **FLA** improves the performance over the published state-of-the-art on both white and black box attack tasks.

In comparison with the previous works, the main contributions of this work can be summarized as follows:

- **FLA** generates locally perturbations only around target objects, which increases the imperceptibility of the generated perturbations.
- **FLA** is efficiently on attacking the SOTA object detectors. It achieves higher white-box attack performance than previous methods with lower computational consumption.
- The adversarial examples generated by **FLA** achieve higher black-box attack performance than previous methods. They are not only capable of attacking anchor-free detectors but also can be transferred to attack anchor-based detectors.

This paper is organized as follows. We first discuss the related work in section II. Then, we provide the details of **FLA** in section III. The detailed setup and results of our experiment are described in Section IV. Finally, we conclude the paper in section V.

II. Related Work

A. Object Detection

There are some great progress has been made in object detection. With the deep convolutional neural network's development, many valuable approaches have been proposed. One of the most popular object detection categories is the RCNN [6] family, such as Faster-RCNN [20]. The first process of RCNN's pipeline is generating a large number of proposals which is base on the anchor. Then use a different classifier to classifying the proposals. At last, use post-processing algorithms, such as NMS, to reduce redundancy proposals.

There are also other object detectors that rely on the anchor, like YOLOv2 [19], SSD [15]. Anchor-base detector has high detected accuracy but also has three shortcomings. Slow, hard to apply a new dataset and more difficult to training. Such as Faster-RCNN.

To solve these shortcomings, some new object detectors have been proposed. Such as CornerNet [12] and CenterNet [26]. These new object detectors detect the objects by detecting the keypoints of objects. CornerNet detects the objects by detecting the two corners of the objects. CenterNet relies on find the center points of objects to detect objects. These two methods can complete the training without preset anchor, which we call as the anchor-free detector. These two detectors are not only faster and simpler to training than anchor-base detectors, but they also achieve SOTA detect performance. Both CornerNet and CenterNet can use multiple convolutional neural networks as the backbone network to extract the semantic features of the input image. Then locate the keypoint of the object through these features. Normally, the keypoint include the size and category information of the object. At last, use some post-processing algorithms to remove the redundancy key-point.

B. Adversarial Example for object detection

Goodfellow [8] first showed the adversarial example problem of the deep neural network. The adversarial example means deliberately generate imperceptibly perturbation to add on the original input dataset.

The adversarial example is aimed to make the deep neural network output the wrong result. Almost existing adversarial attack methods are focus on minimizing the L_p norm of the adversarial perturbation. In the most attack methods $p = 2$ or ∞ that can generate imperceptible perturbation.

The most classical attack methods are the FGSM family, such as Fast Gradient Sign Method (FGSM) [8], Project Gradient Descent (PGD) [16]. The first of the pipeline of the FGSM is to get the loss value of the deep neural network, then compute the gradient of the input image. At last, use the gradient and the sign function to generate the adversarial perturbation. The principle can be summarized as follows:

$$x' = x + \epsilon \cdot sign(\bigtriangledown_x f(x, y)) \tag{1}$$

where f is the classifier, x is the input image for the classifier, ϵ is to constrain the L_∞ of the perturbation.

The difference of the PGD is to add the iterative module to the FGSM. Use many small and accurate perturbation instead of one big perturbation. The PGD achieve higher attack success rate and generate smaller L_p norm perturbation. The principle can be summarized as follows:

$$x_{t+1} = \Pi_{x+s}(x_t + \epsilon \cdot sign(\bigtriangledown_x f(x, y))) \tag{2}$$

There is another attack method can generate lower L_2 perturbation than FGSM family, Deepfool [18]. Deepfool uses the generated hyperplane to approximate the decision boundary, and compute the lowest Euclidean distance between the input image and the hyperplane iterative. Then use the distance to generate the adversarial perturbation. Deepfool achieve state-of-art attack performance while has a lower L_2 norm of perturbation than the FGSM-base attack method.

The above attack methods are mainly to attack the classifier, there are only a few research for attack object detector. Such as DAG and UEA. Both DAG and UEA attack anchor-base object detectors, such as Faster-RCNN and the SSD. Both

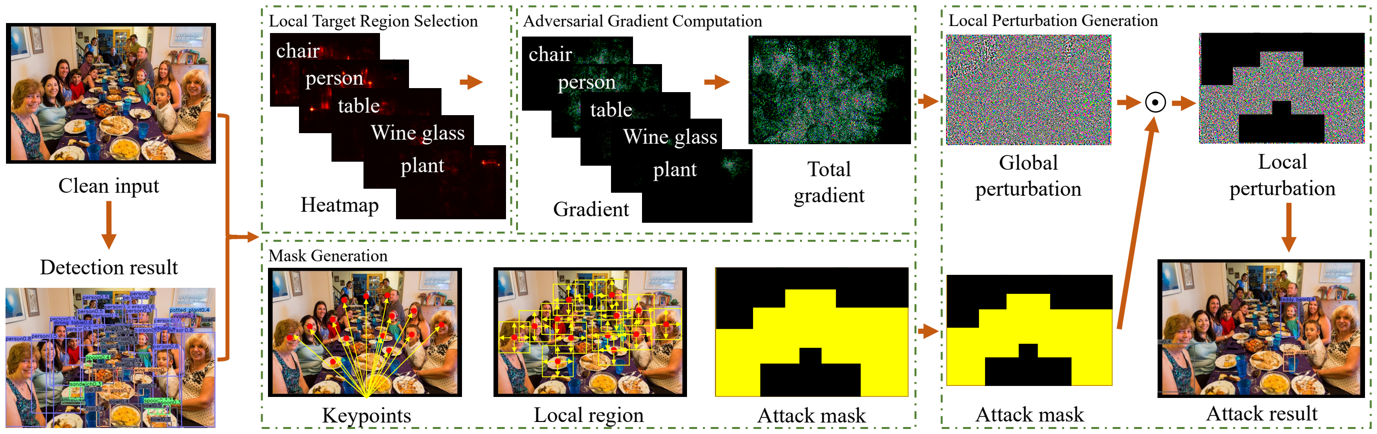


Fig. 2. Illustration of Each Iteration of FLA: First of each iteration, we extract heatmap of each detected categories. Second, use the heatmap to compute the adversarial gradient information of each detect categories. Then normalizing the adversarial gradients with L_∞ and add up all gradients. After normalizing gradients, add up all gradients to generate globally adversarial perturbation by further applying sign operation on it. Finally, generate locally perturbation by the dot product of global perturbation and mask.

DAG and UEA generate globally adversarial perturbation. The main shortcoming of DAG is slow, average consume $10-20s$ to complete ones attack, and weakness on transferring attack. UEA is base on the Generate Adversarial Network [7], it also achieves high attack performance and better transferring attack performance than DAG. But UEA need retraining to attack new dataset or new detector, which is more complex than the optimization-based methods.

III. Method

The Fast Locally Attack (FLA) consists of three parts (as shown in Fig. 2), i.e., (a) Local Target Region Selection (b) Compute Adversarial Gradient and (c) Local Perturbation Generation. We first formulate the FLA as a constraint optimization problem in Section (III-A), then introduce each part in Sections III-B, III-C and III-D, respectively.

A. Problem Definition

The problem of generating adversarial perturbation for object detection can be formulated to the following constraint optimization problem:

$$\begin{aligned} \underset{r}{minimize} \quad & \|r\|_p \\ subject \quad & \widehat{t}(x+r) \cap \widehat{t}(x) = \emptyset \\ & min \le x+r \le max \end{aligned} \tag{3}$$

where x is the origin input image, r is the adversarial perturbation. $\widehat{t}(x)$ denotes the object set that detected by the object detector. $max, min \in \mathbb{R}^n$ denote the maximum and minimum pixel intensity to constraint the pixel of the $x+r$.

There are two ways to satisfy that the intersection of $\widehat{t}(x+r)$ and $\widehat{t}(x)$ is empty. The first one is attacking each object of $\widehat{t}(x)$ individually, the second one is attacking the whole image to make all object categories incorrect. We found the second way is more efficient, thus constraint in the Eq. (3) can be reformatted as follows:

$$\forall t_n \in \widehat{t}(x), f(x+r, t_n) \ne f(x, t_n) \tag{4}$$

where t_n denotes the n-th object of the object set $\widehat{t}(x)$ that detected by the detector, and the $f(x, t_n)$ denotes the category of the object t_n.

B. Local Target Region Selection

We select the local target region using an anchor-free objection method, i.e. CenterNet [26]. In the CenterNet, the detector removes the classifier module and use the keypoint heatmap $\widehat{Y} \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$ to predict the category of the object directly. Where W and H represents the width and height of the heatmap. C means the number of channels. R represents the multiple of downsampling from image to heatmap.

The CenterNet use several different CNN networks as the backbone to construct the complete object detector, and output $\widehat{Y}$ of the input image x. The $\widehat{Y}_{w,h,c} = 1$ indicates the point $\widehat{Y}_{w,h}$ is a detected keypoint which belonging category c, in contrast, the $\widehat{Y}_{w,h,c} = 0$ indicates the point do not belonging c. where w and h denotes the abscissa and ordinate of the point. The CenterNet is base on the keypoints detection which regards the center point of the object as the keypoint. Each detected keypoint $\widehat{Y}_{w,h,c}$ denotes the center point of the object. The keypoints include the information of the object's category, it also includes the scale and the offset of the detection box of the object.

Due to the CenterNet relies on the detected keypoints, the attack method can directly attack the detected keypoints to fail the detector. The constraint in Eq. (4) can be written as the following functions.

$$\begin{aligned} & \widehat{Y} = Centernet(x) \\ & P = \{p_n = \widehat{Y}_{w,h} \mid \widehat{Y}_{w,h,c} = 1, \widehat{Y}_{w,h,c} \in \widehat{Y}\} \\ & \forall n, f(x+r, p_n) \ne f(x, p_n), p_n \in P \end{aligned} \tag{5}$$

where p_n denotes the n-th detected keypoints, all detected keypoints construct the target point set P.

After attacked all detected keypoints, the CenterNet should fail to detect any object on the adversarial example. But we find it still detects some same object after all the keypoints are attacked. We check the heatmap where all the keypoints are changed to the incorrect category, but the neighbor points around the attacked keypoints are also modified. Some neighbor background points' confidence level is increased that makes them belong to the correct category. Due to the new neighbor keypoints location is near to the old keypoints, the newly detected object is in the same category as the old one and the position and the size of the detected bounding box just change little. These two problems make the CenterNet capable to detect the correct object on the adversarial example. To solve those two problems, we can directly add the neighbor keypoints into the target point set. Then the Eq. (5) is further extended to the following function.

$$\begin{aligned}
&\widehat{Y} = Centernet(x) \\
&P = \{p_n = \widehat{Y}_{w,h} \mid \widehat{Y}_{w,h,c} = 1, \widehat{Y}_{w,h,c} \in \widehat{Y}\} \\
&P_{neighbor} = \{p_k \mid p_k \in N(p_n), p_n \in P\} \\
&P = P \cup P_{neighbor} \\
&\forall n, f(x + r, p_n) \neq f(x, p_n), p_n \in P
\end{aligned} \tag{6}$$

where $N(p_n)$ indicates the point set which constructed by the neighbor points around the detected keypoint p_n.

After selecting P composed of detected points and neighbor points, we divide P into different P_j according to different categories j which has detected object.

$$P_j = \{p_n \mid f(x, p_n) = j, p_n \in P\} \tag{7}$$

We generate the adversarial gradient on each P_j, the process of computing adversarial gradient is summarized in the next section.

C. Adversarial Gradient Computation

Our method is base on the PGD [16], which generates perturbation iteratively. In each iteration, the perturbation is generated from the adversarial gradient. Unlike DAG, which only computes gradient on a single object and generates perturbation for a single object in each iteration, our method computes gradient on each P_{target} and add up all the gradients to generate perturbation that can reduce the time consumption and increase the transferring attack performance.

As shown in Algorithm. 1, during each iteration, we first generate target points set P_j for each detected category j which has detected objects. Then, we compute $loss_{sum}$ of P_j of each detected category j and compute adversarial gradient information r_j for each category j. Each r_j is normalized by L_∞ and obtain adversarial gradient information r'_j. After all, we add up all r'_j to obtain total gradient r_i. (An example is shown in Fig. (2).) After obtaining r_i, we generate local perturbation in the next section.

Algorithm 1 Fast Locally Attack (FLA)

```
Input: image $x$, target points set $P$, number of category $C$
    attack radius $R^*$
Output: perturbation $r$
  Initialize: $x^0 \leftarrow x, i \leftarrow 0, j \leftarrow 0, P_0 \leftarrow P, r_{adv} \leftarrow 0$
  while $P_i \cap P \neq \emptyset$ and $i < M_D$ do
      $r_i \leftarrow 0, j \leftarrow 0$
      $mask_i \leftarrow GenerateMask(x, P_i, R^*)$
      while $j < C$ do
          $P_j = \{p_n \mid f(x, p_n) = j, p_n \in P\}$
          if $P_j \neq \emptyset$ then
              $loss_{sum} \leftarrow \sum_{p_n \in P_j} CrossEntropy(x^{(i)}, p_n)$
              $r_j = \nabla_{x^{(i)}} loss_{sum}$
              $r'_j = \frac{r_j}{\|r_j\|_\infty}$
              $r_i \leftarrow r_i + r'_j$
          end if
          $j \leftarrow j + 1$
      end while
      $x^{i+1} \leftarrow x^i + \frac{\epsilon'}{M_D} \cdot sign(r_i) \cdot mask_i$
      $P_{i+1} \leftarrow RefreshPoints\ (x^{i+1}, P_i)$
      $i \leftarrow i + 1$
  end while
  return $r = x^i - x^0$
```

D. Local Perturbation Generation

To generate locally perturbation, we use attack mask $mask_i$ to keep perturbation that around detected objects and remove perturbation in the background. The attack mask $mask_i$ is generated from the P_j by *GenerateMask* step which is demonstrated in Fig. 2. At first, we generate a zero matrix $mask$ that has the same size as the input image. We relocated all points p_i of P_j on the input image. Then get the location of each p_i and set the same location points of $mask$ as 1. After relocate all p_i, then set all points in the box with the size of attack radius R^* and centered on p_i's location of $mask$ as 1. After that, we obtain a locally attack mask. In the final of each attack, we use $mask$ and global perturbation to obtain locally perturbation. Different R^* will lead to different attack performance, we will quantitative analysis of this relation in Section IV-E.

In Algorithm. 1. We generate globally perturbation by applying $sign$ operation to the r_i. After obtaining global perturbation we generate locally perturbation by the dot product of global perturbation and $mask$. After generating perturbation, we refresh the P by $RefreshPoints$, which remove the points p_n which has been attacked successfully.

Normally, the DAG needs $150 - 200$ iterations to generate perturbation and the FLA only needs $10 - 50$ iterations.

Method	Network	Dataset	mAP(Clean)	mAP(Attack)	ASR	Time(s)
DAG [23]	Faster-RCNN	PascalVOC	0.70	0.05	0.92	10.0
UEA [22]	Faster-RCNN	PascalVOC	0.70	0.05	0.92	—
FLA	Resdcn18	PascalVOC	0.67	0.07	0.90	**0.8**
FLA	DLA34	PascalVOC	0.77	0.06	0.93	1.1
FLA	Resdcn18	MS-COCO	0.29	0.006	0.98	2.7
FLA	DLA34-1x	MS-COCO	0.38	0.008	**0.98**	4.7

TABLE I
RESULTS OF WHITE-BOX ATTACK (MEASURED BY MAP, %). IN THE TABLE, CLEAN MEANS THE MAP OBTAINED FROM THE CLEAR INPUT. ATTACK DENOTES THE MAP OBTAINED FROM THE ADVERSARIAL EXAMPLE. IN THE TIME COLUMN, WE SHOW THE AVERAGE ATTACK TIME OF THE ATTACK METHOD.

Network	Resdcn18		Resdcn101		DLA34-1x		DLA34-2x		CornerNet	
	mAP	ATR	mAP	ATR	mAP	ATR	mAP	ATR	mAP	ATR
Clean	0.29	—	0.36	—	0.38	—	0.39	—	0.43	—
DLA34-1x	0.10	0.86	0.12	0.87	**0.09**	1.00	0.11	0.94	0.13	0.92
DLA34-2x	**0.10**	**0.88**	**0.12**	**0.90**	0.11	0.96	**0.10**	1.00	**0.13**	**0.94**

TABLE II
TEST BLACK-BOX ATTACK PERFORMANCE ON THE COCO DATASET. THE FIRST COLUMN MEANS WHICH NETWORK THAT ADVERSARIAL EXAMPLES GENERATE FROM. THE FIRST ROW MEANS WHICH NETWORK THAT BEEN ATTACKED IN THE BLACK-BOX ATTACK.

Network	Resdcn18		DLA34		Resdcn101		Faster-RCNN		SSD300	
	mAP	ATR	mAP	ATR	mAP	ATR	mAP	ATR	mAP	ATR
Clean	0.67	None	0.77	None	0.76	None	0.71	None	0.77	None
DAG [23]	0.65	0.19	0.75	0.16	0.74	0.16	0.60	1.00	0.76	0.08
DLA34-384	0.50	0.30	0.1	1.00	0.62	0.22	0.53	**0.35**	0.67	0.15
DLA34-512	**0.48**	**0.32**	**0.07**	1.00	**0.60**	**0.24**	**0.51**	0.37	**0.66**	**0.16**

TABLE III
TEST BLACK-BOX ATTACK PERFORMANCE ON THE PASCAL DATASET. THE FIRST COLUMN MEANS WHICH NETWORK THAT ADVERSARIAL EXAMPLES GENERATE FROM. THE FIRST ROW MEANS WHICH NETWORK THAT BEEN ATTACKED IN THE BLACK-BOX ATTACK. THE ROW 'DAG' DENOTES THE BLACK-BOX ATTACK RESULT OF DAG. THE ROW 'DLA34-384' AND 'DLA34-512' DENOTES THE BLACK-BOX ATTACK RESULT OF FLA ON DLA34 BACKBONE CENTERNET WITH DIFFERENT INPUT SCALE.

IV. EXPERIMENT

In this section, we first introduce the detailed setup of the experiment. Then, we report both the white-box and black-box attack results. Finally, we evaluate the perceptibility of the generated adversarial examples and the attack radius R^*.

A. Experimental Details

1) Attacked Object Detectors: All adversarial examples are generated on CenterNet with two different backbones: ResNet-18 [9] and DLA-34 [25]. CenterNet with backbone ResNet-18 is highly efficient. In contrast, CenterNet with backbone DLA-34 is more computationally intensive but more accurate.

2) Dataset: The above two networks are trained on the training set of PascalVOC [4] and MS-COCO [14]. The training set of PascalVOC includes the trainval sets of PascalVOC-2007 and PascalVOC-2012. In this paper, both the white-box and black-box attack performance are reported on the testing set of PascalVOC and MS-COCO.

3) Metrics: We compare the white-box attack performance with the DAG and the UEA. It is evaluated by computing the decreased percentage of mean average precision (mAP), which is referred to as **Attack Success Ratio(ASR)** in this paper:

$$ASR = 1 - \frac{mAP_{attack}}{mAP_{clean}} \quad (8)$$

where mAP_{attack} denotes the mAP of the targeted object detector on adversarial examples. mAP_{clean} denotes the mAP of clean input. Higher ASR means better white-box attack performance.

The black-box attack signifies the transferability of the generated adversarial examples to other object detectors. In this paper, black-box adversarial examples are generated on Centernet with DLA-34 [25] backbone and tested on Centernet with different backbones (Resdcn18 and Resdcn101). We also test these adversarial examples on other object detectors, including anchor-free (CornerNet) and anchor-based detectors (Faster-RCNN and SSD300). In this paper, the performance of the black-box attack is evaluated by the ASR ratio between the targeted detector and the original detector on which the adversarial examples are generated. It's referred to as **Attack**

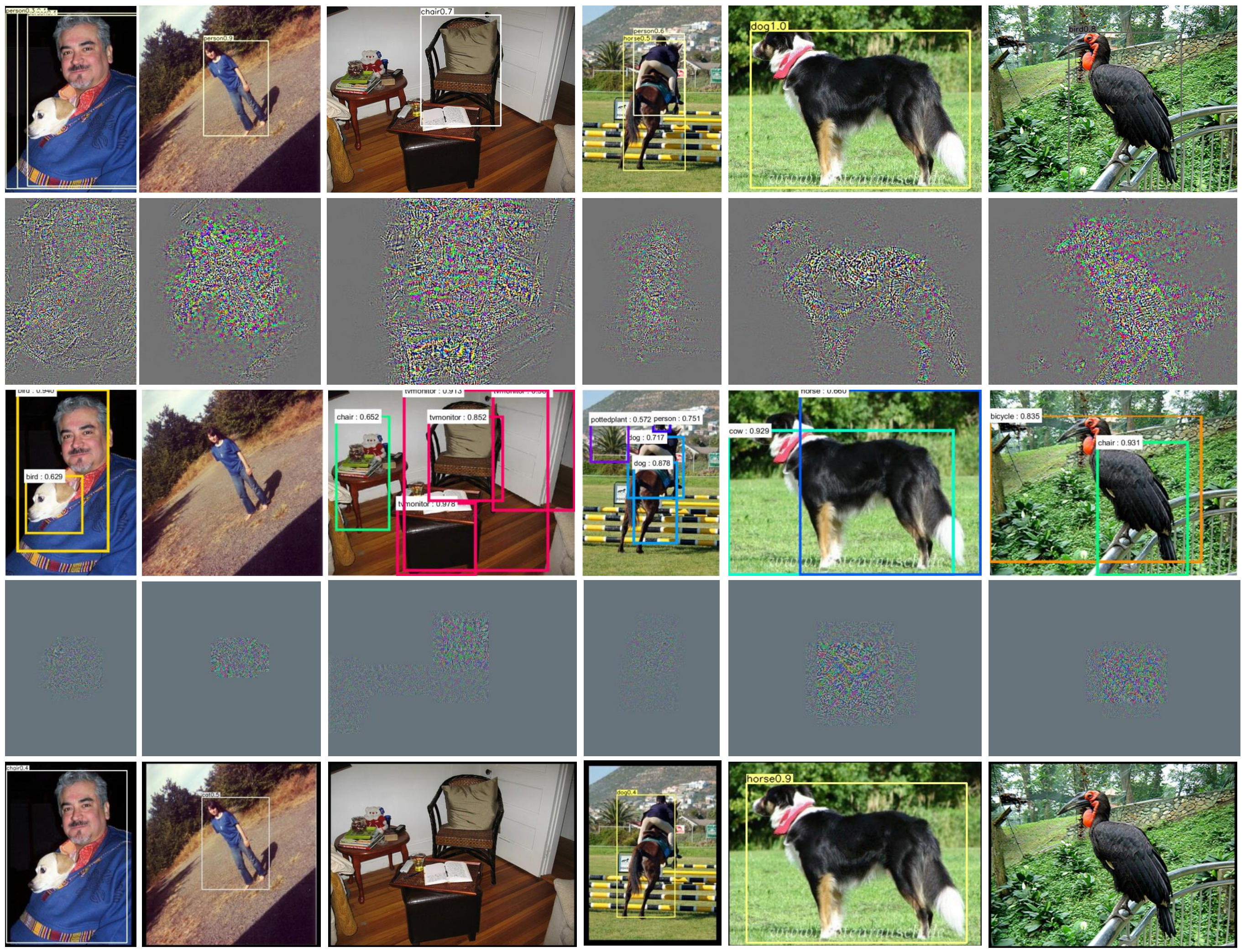


Fig. 3. Each column is an example. **Row 1:** Detection results of clean inputs on CenterNet. **Row 2&3:** DAG perturbations and DAG attacked results on Faster-RCNN. **Row 4&5:** FLA perturbations and FLA attacked results on CenterNet. Note that in **Row 4**, from left to right, the percentage of the changed pixels for each FLA perturbations are: 17% 8%, 26%, 25%, 26%, 14%. We can see that the perturbations of FLA are smaller than the DAG. To better show the perturbation, we have multiplied the intensity of all perturbation images by 10.

Transfer Ratio (ATR) in this paper:

$$ATR = \frac{ASR_{target}}{ASR_{origin}} \tag{9}$$

where ASR_{target} represents the ASR of the targeted detector and ASR_{origin} denotes the ASR of the detector on which the adversarial examples are generated. Higher ATR denotes better transferability.

4) Perceptibility Metric: The adversarial perturbation's perceptibility is quantified by its L_p norms. Specifically, P_{L_2} and P_{L_0} are used, which are defined as follows.

i) P_{L_2}**:** L_2 norm of the perturbation. A lower L_2 value usually signifies that the perturbation is more imperceptible for the human. Formally,

$$P_{L_2} = \sqrt{\frac{1}{k}\sum r_k^2} \tag{10}$$

where the k is the number of the pixels. We also normalized the P_{L_2} in $[0, 1]$.

ii) P_{L_0}**:** L_0 norm of the perturbation. A lower L_0 value means that less less images images are changed during the attack. We compute P_{L_0} by measuing the proportion of changed pixels. The whole experiment is conducted with a Intel Core i7-7700k CPU and an Nvidia GeForce GTX-1080ti GPU.

B. White-Box Attack Results

In this subsection, we show the white-box attack result on PascalVOC and MS-COCO. The overall attack results are shown in Table I. It is obvious that the mAPs of different Centernet have dropped dramatically after adversarial attack. In PascalVOC, the ASR of FLA is 0.90 and 0.93 respectively when the backbone is Resdcn18 and DLA-34, outperforming DAG and UEA. Besides, FLA is almost ten times faster than DAG. Regarding UEA, we set the attack time to N/A, as they don't

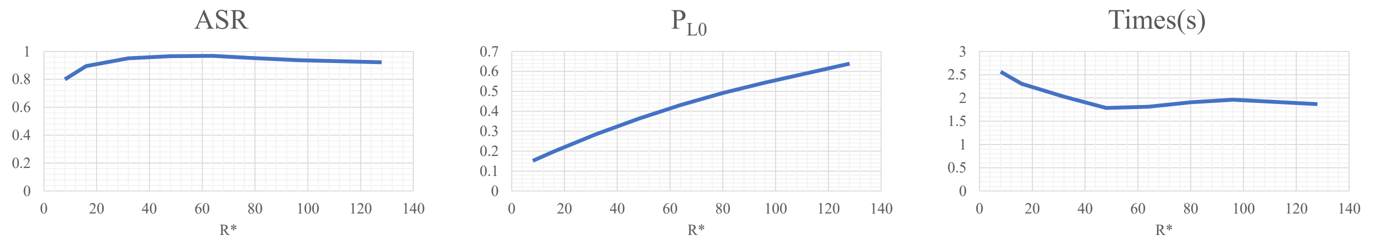


Fig. 4. Correlation between the attack radius R^* and the attack performance, time consumption, and the perceptibility of adversarial examples. The ASR of FLA correlates positively with R^* when R^* is less 16. However, ASR is stable or slightly decreased afterwards. P_{L_0} of the perturbation correlates positively with R^*. This is because higher R^* means bigger attack masks. The mean attack time of FLA correlates negatively with R^* when R^* is lower than 48. However, the mean attack time of FLA become stable afterwards.

Network	Dataset	P_{L_2}	P_{L_0}
DAG [23]	PascalVOC	3×10^{-3}	$> 99\%$
Resdcn18	PascalVOC	5.9×10^{-3}	30%
DLA34	PascalVOC	6.1×10^{-3}	32%
Resdcn18	MS-COCO	6.0×10^{-3}	38%
DLA34	MS-COCO	6.0×10^{-3}	36%

TABLE IV
EVALUATION OF THE PERCEPTIBILITY OF THE GENERATED ADVERSARIAL EXAMPLES. HIGHER P_{L_2} AND P_{L_2} VALUES GENERALLY MEAN THAT THE GENERATED ADVERSARIAL EXAMPLES ARE MORE PERCEPTIBLE TO HUMAN EYES.

provide the source code. Note that UEA requires additional training time. In MS-COCO, the FLA achieves 0.98 ASR on Resdcn18 and DLA34. On Resdcn18, the average attack time required by FLA is 2.7s. On DLA34, the average attack time required by FLA is 4.7s.

C. Black-Box Attack Results

We report the black-box attack results in this subsection. The black-box attack measures the transferability of adversarial examples. All adversarial examples generated for black-box attack experiment are generated on Centernet with backbone DLA34-1x and DLA34-2x.

At first in the experiment, we use the FLA to generate the adversarial example on the CenterNet and save the adversarial example in a common image format, JPG. Then reload the saved adversarial example to compute the mAP. We want to simulate a real transferring attack scenario, so we are abandoning the use of lossless photo formats and use the JPG format to save. Most of transferring test use the lossless float matrix to test, but most of normal image input is $8-bit$ int matrix. So we save the adversarial example in JPG format, this process will better simulate a real transferring attack scenario. Compare with directly compute mAP with lossless adversarial example, save the adversarial example in JPG will lose a small amount of aggressiveness [3], because save in JPG will lose some details of the adversarial example. But in this way we can further guarantee the attack ability of our method. We test the transferability of adversarial examples by generated on one model and compute mAP on other models.

As the pervious experiment, we evaluate the black-box attack performance of the adversarial example on PascalVOC and MS-COCO. On the PascalVOC, we generate adversarial example on DLA34-1x and DLA34-2x backbone centernet. As compare, we generate adversarial example on Faster-RCNN by DAG and also save as JPG to transferring to other detector. On the COCO, we also generate adversarial example on DLA34-1x and DLA34-2x backbone centernet. and transferring to other backbone centernet and CornerNet.

The results of black-box attack are summarize in the Table II and Table III. As the shown on Table III. On the PascalVOC, adversarial example that generated by our method has obviously higher black-box attack performance than the DAG. The adversarial example that generated by DAG loss its aggressiveness after JPG compression, hard to attack the faster-rcnn. The adversarial example that generated by FLA keep its aggressiveness after JPG compression, and achieve higher ATR than the DAG. The adversarial example that generated by FLA is also valid to transferring attacking anchor-base detector, Faster-RCNN and SSD300. Meanwhile, the adversarial example generated by DAG is invalid to attack anchor-free detector.

On the Table II, compare with test on PascalVOC, FLA achieve higher ATR on COCO. The average ATR is over 90%. The adversarial example that generated by FLA is valid to different backbone centernet. It is also achieve high black-box attack performance when transferring to CornerNet.

D. Evaluation of Perceptibility

The P_{L_2} and P_{L_0} values of the generated adversarial examples by FLA are summarized in Table IV. Higher P_{L_2} and P_{L_2} values generally mean that the generated adversarial examples are more perceptible to human eyes. Although P_{L_2} of our method is slighly higher than DAG, the perturbation is almost imperceptible to human eyes (see Fig. 3). The P_{L_0} value of FLA is significantly lower than DAG, meaning that the generated perturbations of FLA are much locally constrained than DAG.

E. Evaluation of Attack Radius R^*

R^* correlates with the attack performance, time consumption, and the perceptibility of adversarial examples. We summarize the relation between R^* and adversarial perturbation in Fig. 4. In this experiment, all adversarial examples are generated by attacking centernet with backbone Resdcn18 on MS-

COCO. Based on Fig. 4, we can draw three conclusions. First, the ASR of FLA correlates positively with R^* when R^* is less 16. However, ASR is stable or slightly decreased afterwards. Second, P_{L_0} of the perturbation correlates positively with R^*. This is because higher R^* means bigger attack masks. Finally, the mean attack time of FLA correlates negatively with R^* when R^* is lower than 48. However, the mean attack time of FLA become stable afterwards.

V. Conclusion

In this paper, we propose Fast Locally Attack to generate transferable adversarial example for attacking SOTA anchor-free object detectors. Our method leverages higher-level semantic information to generate locally adversarial perturbation. FLA computes adversarial gradient information for all detected categories and generate locally perturbation, which can improve the attack performance of adversarial example. FLA also achieved SOTA white-box attack performance for attacking Centernet, while being dozens of times faster than DAG. Additionally, it only changes $30\% - 40\%$ image pixels during the attack process. Finally, our method achieved better black-box attack performance and are more robust to JPEG compression.

Acknowledgment

This work was supported by the Sichuan Science and Technology Program under Grant 2019YFG0399.

References

[1] Avishek Joey Bose and Parham Aarabi. Adversarial attacks on face detectors using neural net based constrained optimization. In *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2018.

[2] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Chau. Robust physical adversarial attack on faster r-cnn object detector. *arXiv preprint arXiv:1804.05810*, 2(3):4, 2018.

[3] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853*, 2016.

[4] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision (IJCV)*, 111(1):98–136, 2015.

[5] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945*, 2017.

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014.

[8] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[9] 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 (CVPR)*, pages 770–778, 2016.

[10] Lichao Huang, Yi Yang, Yafeng Deng, and Yinan Yu. Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015.

[11] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[12] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018.

[13] Yuezun Li, Daniel Tian, Xiao Bian, and Siwei Lyu. Robust adversarial perturbation on deep proposal-based models. *British Machine Vision Conference (BMVC)*, 2018.

[14] 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 *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.

[15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37. Springer, 2016.

[16] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[17] Jan Hendrik Metzen, Mummadi Chaithanya Kumar, Thomas Brox, and Volker Fischer. Universal adversarial perturbations against semantic image segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2774–2783. IEEE, 2017.

[18] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2016.

[19] 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 (CVPR)*, pages 779–788, 2016.

[20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[21] Pedro Tabacof, Julia Tavares, and Eduardo Valle. Adversarial images for variational autoencoders. *arXiv preprint arXiv:1612.00155*, 2016.

[22] Xingxing Wei, Siyuan Liang, Ning Chen, and Xiaochun Cao. Transferable adversarial attacks for image and video object detection. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 954–960. AAAI Press, 2019.

[23] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1369–1378, 2017.

[24] Dawei Yang, Chaowei Xiao, Bo Li, Jia Deng, and Mingyan Liu. Realistic adversarial examples in 3d meshes. *arXiv preprint arXiv:1810.05206*, 2, 2018.

[25] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2403–2412, 2018.

[26] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

[27] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krahenbuhl. Bottom-up object detection by grouping extreme and center points. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 850–859, 2019.