Adversarial Examples for Semantic Segmentation and Object Detection

Cihang Xie[1*], Jianyu Wang[2*], Zhishuai Zhang[1*], Yuyin Zhou[1], Lingxi Xie[1 (✉)], Alan Yuille[1]
[1]Department of Computer Science, The Johns Hopkins University, Baltimore, MD 21218 USA
[2]Baidu Research USA, Sunnyvale, CA 94089 USA
{cihangxie306, wjyouch, zhshuai.zhang, zhouyuyiner, 198808xc, alan.l.yuille}@gmail.com

Abstract

It has been well demonstrated that adversarial examples, i.e., natural images with visually imperceptible perturbations added, cause deep networks to fail on image classification. In this paper, we extend adversarial examples to semantic segmentation and object detection which are much more difficult. Our observation is that both segmentation and detection are based on classifying multiple targets on an image (e.g., the target is a pixel or a receptive field in segmentation, and an object proposal in detection). This inspires us to optimize a loss function over a set of targets for generating adversarial perturbations. Based on this, we propose a novel algorithm named Dense Adversary Generation (DAG), which applies to the state-of-the-art networks for segmentation and detection. We find that the adversarial perturbations can be transferred across networks with different training data, based on different architectures, and even for different recognition tasks. In particular, the transfer ability across networks with the same architecture is more significant than in other cases. Besides, we show that summing up heterogeneous perturbations often leads to better transfer performance, which provides an effective method of black-box adversarial attack.

1. Introduction

Convolutional Neural Networks (CNN) [14][30][31][13] have become the state-of-the-art solution for a wide range of visual recognition problems. Based on a large-scale labeled dataset such as ImageNet [6] and powerful computational resources like modern GPUs, it is possible to train a hierarchical deep network to capture different levels of visual patterns. A deep network is also capable of generating transferrable features for different tasks such as image classification [7] and instance retrieval [28], or being fine-tuned to deal with a wide range of vision tasks, including object detection [11][10], visual concept discovery [34], semantic segmentation [18][37][3], boundary detection [29][35], etc.

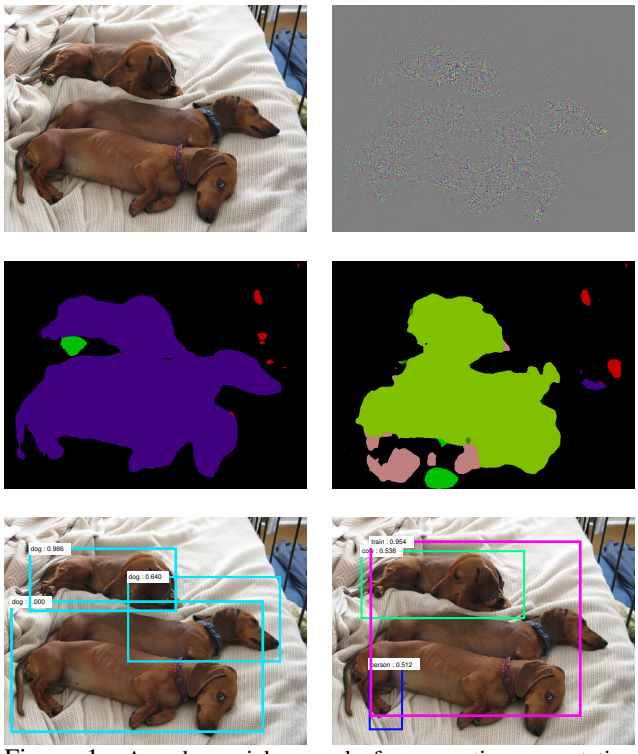

Figure 1: An adversarial example for semantic segmentation and object detection. FCN [18] is used for segmentation, and Faster-RCNN [27] is used for detection. Left column: the original image (top row) with the normal segmentation (the purple region is predicted as *dog*) and detection results. Right column: after the adversarial perturbation (top row, **magnified by 10**) is added to the original image, both segmentation (the light green region as *train* and the pink region as *person*) and detection results are completely wrong. Note that, though the added perturbation can confuse both networks, it is visually imperceptible (the maximal absolute intensity in each channel is less than 10).

Despite their success in visual recognition and feature representation, deep networks are often sensitive to small perturbations to the input image. In [32], it was shown

The first three authors contributed equally to this work. This work was done when Jianyu Wang was a Ph.D. student at UCLA.

that adding visually imperceptible perturbations can result in failures for image classification. These perturbed images, often called *adversarial examples*, are considered to fall on some areas in the large, high-dimensional feature space which are not explored in the training process. Thus, investigating this not only helps understand the working mechanism of deep networks, but also provides opportunities to improve the robustness of network training.

In this paper, we go one step further by generating adversarial examples for semantic segmentation and object detection, and showing the transferability of them. To the best of our knowledge, this topic has not been systematically studied (*e.g.*, on a large dataset) before. Note that these tasks are much more difficult, as we need to consider orders of magnitude more targets (*e.g.*, pixels or proposals). Motivated by the fact that each target undergoes a separate classification process, we propose the Dense Adversary Generation (DAG) algorithm, which considers all the targets simultaneously and optimizes the overall loss function. The implementation of DAG is simple, as it only involves specifying an adversarial label for each target and performing iterative gradient back-propagation. In practice, the algorithm often comes to an end after a reasonable number of, say, 150 to 200, iterations. Figure 1 shows an adversarial example which can confuse both deep segmentation and detection networks.

We point out that generating an adversarial example is more difficult in detection than in segmentation, as the number of targets is orders of magnitude larger in the former case, *e.g.*, for an image with K pixels, the number of possible proposals is $O(K^2)$ while the number of pixels is only $O(K)$, where $O(\cdot)$ is the big-O notation. In addition, if only a subset of proposals are considered, the perturbed image may still be correctly recognized after a new set of proposals are extracted (note that DAG aims at generating recognition failures on the original proposals). To increase the robustness of adversarial attack, we change the intersection-over-union (IOU) rate to preserve an increased but still reasonable number of proposals in optimization. In experiments, we verify that when the proposals are dense enough on the original image, it is highly likely that incorrect recognition results are also produced on the new proposals generated on the perturbed image. We also study the effectiveness and efficiency of the algorithm with respect to the *denseness* of the considered proposals.

Following [32], we investigate the transferability of the generated perturbations. To this end, we use the adversarial perturbation computed on one network to attack another network. Three situations are considered: (1) networks with the same architecture but trained with different data; (2) networks with different architectures but trained for the same task; and (3) networks for different tasks. Although the difficulty increases as the difference goes more significant, the perturbations generated by DAG is able to transfer to some extent. Interestingly, adding two or more heterogeneous perturbations significantly increases the transferability, which provides an effective way of performing black-box adversarial attack [25] to some networks with unknown structures and/or properties.

2. Related Work

2.1. Deep Learning for Detection and Segmentation

Deep convolutional neural networks have been applied to object detection [11][10][27][5][16] and semantic segmentation [18][3][37] successfully. Currently, one of the most popular object detection pipeline [27][5][16] involves first generating a number of proposals of different scales and positions, classifying each of them, and performing post-processing such as non-maximal suppression (NMS). On the other hand, the dominating segmentation pipeline [18] works by first predicting a class-dependent score map at a reduced resolution, and performing up-sampling to obtain high-resolution segmentation. [3] incorporates the "atrous" algorithm and the conditional random field (CRF) to this pipeline to improve the segmentation performance further.

2.2. Adversarial Attack and Defense

Generating adversarial examples for classification has been extensively studied recently. [32] first showed that adversarial examples, computed by adding visually imperceptible perturbations to the original images, make CNNs predict a wrong label with high confidence. [12] proposed a simple and fast gradient sign method to generate adversarial examples based on the linear nature of CNNs. [23] proposed a simple algorithm to compute the minimal adversarial perturbation by assuming that the loss function can be linearized around the current data point at each iteration. [22] showed the existence of universal (image-agnostic) adversarial perturbations. [1] trained a feedforward network to generate adversarial examples for a particular target model (without using gradients). [17] studied the transferability of both non-targeted and targeted adversarial examples, and proposed an ensemble-based approaches to generate adversarial examples with stronger transferability. [24] generated images using evolutionary algorithms that are unrecognizable to humans, but cause CNNs to output very confident (incorrect) predictions. This can be thought of as in the opposite direction of above works.

In contrast to generating adversarial examples, there are some works trying to reduce the effect of adversarial examples. Defensive distillation [26] was proposed as a defense to against adversarial examples, while [2] developed stronger attacks which break defensive distillation easily. [15] trained the network on adversarial examples using the large-scale dataset, ImageNet, and showed this brings

robustness to adversarial attack. This is improved by [33], which proposed an ensemble adversarial training method to increase the network robustness to black-box attacks. [20] trained a detector on the inner layer of the classifier to detect adversarial examples. [19] proposed a forveation-based mechanism to alleviate adversarial examples.

There are two concurrent works [9][21] studying adversarial examples in semantic segmentation on the Cityscapes dataset [4]. [9] showed the existence of adversarial examples, and [21] showed the existence of universal perturbations. We refer interested readers to their papers for details.

3. Generating Adversarial Examples

In this section, we introduce DAG algorithm. Given an image and the recognition targets (proposals and/or pixels), DAG generates an adversarial perturbation which aims at confusing as many targets as possible.

3.1. Dense Adversary Generation

Let $\mathbf{X}$ be an image which contains N recognition *targets* $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$. Each target t_n, $n = 1, 2, \ldots, N$, is assigned a ground-truth class label $l_n \in \{1, 2, \ldots, C\}$, where C is the number of classes, *e.g.*, $C = 21$ (including the *background* class) in the PascalVOC dataset [8]. Denote $\mathcal{L} = \{l_1, l_2, \ldots, l_n\}$. The detailed form of $\mathcal{T}$ varies among different tasks. In image classification, $\mathcal{T}$ only contains one element, *i.e.*, the entire image. Conversely, $\mathcal{T}$ is composed of all pixels (or the corresponding receptive fields) in semantic segmentation, and all proposals in object detection. We will discuss how to construct $\mathcal{T}$ in Section 3.2.

Given a deep network for a specific task, we use $\mathbf{f}(\mathbf{X}, t_n) \in \mathbb{R}^C$ to denote the classification score vector (before softmax normalization) on the n-th recognition target of $\mathbf{X}$. To generate an adversarial example, the goal is to make the predictions of all targets go wrong, *i.e.*, $\forall n$, $\arg\max_c \{f_c(\mathbf{X} + \mathbf{r}, t_n)\} \neq l_n$. Here $\mathbf{r}$ denotes an adversarial perturbation added to $\mathbf{X}$. To this end, we specify an adversarial label l'_n for each target, in which l'_n is randomly sampled from other incorrect classes, *i.e.*, $l'_n \in \{1, 2, \ldots, C\} \setminus \{l_n\}$. Denote $\mathcal{L}' = \{l'_1, l'_2, \ldots, l'_n\}$. In practice, we define a random permutation function $\pi : \{1, 2, \ldots, C\} \to \{1, 2, \ldots, C\}$ for every image independently, in which $\pi(c) \neq c$ for $c = 1, 2, \ldots, C$, and generate $\mathcal{L}'$ by setting $l'_n = \pi(l_n)$ for all n. Under this setting, the loss function covering all targets can be written as:

$$L(\mathbf{X}, \mathcal{T}, \mathcal{L}, \mathcal{L}') = \sum_{n=1}^{N} \left[f_{l_n}(\mathbf{X}, t_n) - f_{l'_n}(\mathbf{X}, t_n) \right] \quad (1)$$

Minimizing L can be achieved via making every target to be incorrectly predicted, *i.e.*, suppressing the confidence of the original correct class $f_{l_n}(\mathbf{X} + \mathrm{r}, t_n)$, while increasing that of the desired (adversarial) incorrect class $f_{l'_n}(\mathbf{X} + \mathrm{r}, t_n)$.

Algorithm 1: Dense Adversary Generation (DAG)

Input : input image $\mathbf{X}$;
the classifier $\mathbf{f}(\cdot, \cdot) \in \mathbb{R}^C$;
the target set $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$;
the original label set $\mathcal{L} = \{l_1, l_2, \ldots, l_N\}$;
the adversarial label set $\mathcal{L}' = \{l'_1, l'_2, \ldots, l'_N\}$;
the maximal iterations M_0;
Output: the adversarial perturbation $\mathbf{r}$;

$\mathbf{X}_0 \leftarrow \mathbf{X}$, $\mathbf{r} \leftarrow \mathbf{0}$, $m \leftarrow 0$, $\mathcal{T}_0 \leftarrow \mathcal{T}$;
while $m < M_0$ **and** $\mathcal{T}_m \neq \varnothing$ **do**
  $\mathcal{T}_m = \{t_n \mid \arg\max_c \{f_c(\mathbf{X}_m, t_n)\} = l_n\}$;
  $\mathbf{r}_m \leftarrow \sum_{t_n \in \mathcal{T}_m} \left[\nabla_{\mathbf{X}_m} f_{l'_n}(\mathbf{X}_m, t_n) - \nabla_{\mathbf{X}_m} f_{l_n}(\mathbf{X}_m, t_n) \right]$;
  $\mathbf{r}'_m \leftarrow \frac{\gamma}{\|\mathbf{r}_m\|_\infty} \mathbf{r}_m$;
  $\mathbf{r} \leftarrow \mathbf{r} + \mathbf{r}'_m$;
  $\mathbf{X}_{m+1} \leftarrow \mathbf{X}_m + \mathbf{r}'_m$;
  $m \leftarrow m + 1$;
end
Return: $\mathbf{r}$

We apply a gradient descent algorithm for optimization. At the m-th iteration, denote the current image (possibly after adding several perturbations) as $\mathbf{X}_m$. We find the set of correctly predicted targets, named the *active target set*: $\mathcal{T}_m = \{t_n \mid \arg\max_c \{f_c(\mathbf{X}_m, t_n)\} = l_n\}$. Then we compute the gradient with respect to the input data and then accumulate all these perturbations:

$$\mathbf{r}_m = \sum_{t_n \in \mathcal{T}_m} \left[\nabla_{\mathbf{X}_m} f_{l'_n}(\mathbf{X}_m, t_n) - \nabla_{\mathbf{X}_m} f_{l_n}(\mathbf{X}_m, t_n) \right] \quad (2)$$

Note that $|\mathcal{T}_m| \ll |\mathcal{T}|$ when m gets large, thus this strategy considerably reduces the computational overhead. To avoid numerical instability, we normalize $\mathbf{r}_m$ as

$$\mathbf{r}'_m = \frac{\gamma}{\|\mathbf{r}_m\|_\infty} \cdot \mathbf{r}_m \quad (3)$$

where $\gamma = 0.5$ is a fixed hyper-parameter. We then add $\mathbf{r}'_m$ to the current image $\mathbf{X}_m$ and proceed to the next iteration. The algorithm terminates if either all the targets are predicted as desired, *i.e.*, $\mathcal{T}_m = \varnothing$, or it reaches the maximum iteration number, which is set to be 200 in segmentation and 150 in detection.

The final adversarial perturbation is computed as $\mathbf{r} = \sum_m \mathbf{r}'_m$. Note that, in practice, we often obtain the input image $\mathbf{X}$ after subtracting the mean image $\widehat{\mathbf{X}}$. In this case, the adversarial image is $\mathrm{Trunc}\left(\mathbf{X} + \mathbf{r} + \widehat{\mathbf{X}}\right)$, where $\mathrm{Trunc}(\cdot)$ denotes the function that truncates every pixel value by $[0, 255]$. Although truncation may harm the adversarial perturbation, we observed little effect in experiments, mainly because the magnitude of perturbation $\mathbf{r}$ is very small (see

Section 3.5.3). The overall pipeline of DAG algorithm is illustrated in Algorithm 1.

3.2. Selecting Input Proposals for Detection

A critical issue in DAG is to select a proper set $\mathcal{T}$ of targets. This is relatively easy in the semantic segmentation task, because the goal is to produce incorrect classification on all pixels, and thus we can set each of them as a separate target, *i.e.*, performing dense sampling on the image lattice. This is tractable, *i.e.*, the computational complexity is proportional to the total number of pixels.

In object detection, target selection becomes a lot more difficult, as the total number of possible targets (bounding box proposals) is orders of magnitudes larger than that in semantic segmentation. A straightforward choice is to only consider the proposals generated by a sideway network, *e.g.*, the regional proposal network (RPN) [27], but we find that when the adversarial perturbation $\mathbf{r}$ is added to the original image $\mathbf{X}$, a different set of proposals may be generated according to the new input $\mathbf{X}+\mathbf{r}$, and the network may still be able to correctly classify these new proposals [19]. To overcome this problem, we make the proposals very dense by increasing the threshold of NMS in RPN. In practice, when the intersection-over-union (IOU) goes up from 0.70 to 0.90, the average number of proposals on each image increases from around 300 to around 3000. Using this denser target set $\mathcal{T}$, most probable object bounding boxes are only pixels away from at least one of the selected input proposals, and we can expect the classification error transfers among neighboring bounding boxes. As shown in experiments, this heuristic idea works very well, and the effect of adversarial perturbations is positively correlated to the number of proposals considered in DAG.

Technically, given the proposals generated by RPN, we preserve all *positive* proposals and discard the remaining. Here, a positive proposal satisfies the following two conditions: 1) the IOU with the closest ground-truth object is greater than 0.1, and 2) the confidence score for the corresponding ground-truth class is greater than 0.1. If both conditions hold on multiple ground-truth objects, we select the one with the maximal IOU. The label of the proposal is defined as the corresponding confident class. This strategy aims at selecting high-quality targets for Algorithm 1.

3.3. Quantitative Evaluation

Following some previous work [32][23], we evaluate our approach by measuring the drop in recognition accuracy, *i.e.*, mean intersection-over-union (mIOU) for semantic segmentation and mean average precision (mAP) for object detection, using the original test images and the ones after adding adversarial perturbations[1].

[1]For implementation simplicity, we keep targets with ground-truth class label *background* unchanged when generating adversarial examples.

Network	ORIG	ADVR	PERM
FCN-Alex	48.04	3.98	48.04
FCN-Alex*	48.92	3.98	48.91
FCN-VGG	65.49	4.09	65.47
FCN-VGG*	67.09	4.18	67.08
FR-ZF-07	58.70	3.61	58.33
FR-ZF-0712	61.07	1.95	60.94
FR-VGG-07	69.14	5.92	68.68
FR-VGG-0712	72.07	3.36	71.97

Table 1: Semantic segmentation (measured by mIOU, %) and object detection (measured by mAP, %) results of different networks. Here, ORIG is the accuracy obtained on the original image set, ADVR is obtained on the set after the adversarial perturbations are added, and PERM is obtained after the randomly permuted perturbations are added. Please see Section 3.3 for details.

- For semantic segmentation, we study two network architectures based on the FCN [18] framework. One of them is based on the **AlexNet** [14] and the other one is based on the 16-layer **VGGNet** [30]. Both networks have two variants. We use **FCN-Alex** and **FCN-VGG**, which are publicly available, to denote the networks that are trained on the original FCN [18] training set which has 9610 images, and use **FCN-Alex*** and **FCN-VGG*** to denote the networks that are trained on the DeepLab [3] training set which has 10582 images. We use the validation set in [18] which has 736 images as our semantic segmentation test set.

- For object detection, based on the Faster-RCNN [27] framework, we study two network architectures, *i.e.*, the **ZFNet** [36] and the 16-layer **VGGNet** [30]. Both networks have two variants, which are either trained on the **PascalVOC-2007** trainval set, or the combined **PascalVOC-2007** and **PascalVOC-2012** trainval sets. These four models are publicly available, and are denoted as **FR-ZF-07**, **FR-ZF-0712**, **FR-VGG-07** and **FR-VGG-0712**, respectively. We use the **PascalVOC-2007** test set which has 4952 images as our object detection test set.

Results are summarized in Table 1. We can observe that the accuracy (mIOU for segmentation and mAP for detection) drops significantly after the adversarial perturbations are added, demonstrating the effectiveness of DAG algorithm. Moreover, for detection, the networks with more training data are often more sensitive to the adversarial perturbation. This is verified by the fact that **FR-ZF-07** (from 58.70% to 3.61%) has a smaller performance drop than **FR-ZF-0712** (from 61.07% to 1.95%), and that **FR-VGG-07** (from 69.14% to 5.92%) has a smaller performance drop than **FR-VGG-0712** (from 72.04% to 3.36%).

To verify the importance of the spatial structure of adversarial perturbations, we evaluate the accuracy after ran-

Figure 2: Examples generated by DAG for semantic segmentation. The *adversarial* image is on the left and the *fooling* image is on the right. From top to bottom: the original image, the perturbation (**magnified by 10**), the adversarial image after adding perturbation, and the segmentation results. The red, blue and black regions are predicted as *airplane*, *bus* and *background*.

domly permuting the rows and/or columns of $\mathbf{r}$. In Table 1, we find that permuted perturbations cause negligible accuracy drop, indicating that it is the spatial structure of $\mathbf{r}$, instead of its magnitude, that indeed contributes in generating adversarial examples. For permutation results, we randomly permute $\mathbf{r}$ for three times and take the average.

3.4. Adversarial Examples

Figure 1 shows an adversarial example that fails in both detection and segmentation networks. In addition, we show that DAG is able to control the output of adversarial images very well. In Figure 2, we apply DAG to generating one *adversarial* image (which humans can recognize but deep networks cannot) and one *fooling* image [24] (which is completely unrecognizable to humans but deep networks produce false positives). This suggests that deep networks only cover a limited area in the high-dimensional feature space, and that we can easily find adversarial and/or fooling examples that fall in the unexplored parts.

3.5. Diagnostics

3.5.1 The Denseness of Proposals

We first observe the impact brought by the denseness of the proposals for generating adversaries. To this end, we use different IOU rates in the NMS process after RPN [27], which directly affects the number of proposals preserved in Algorithm 1. As we can see in Figure 3, the mAP value goes down (*i.e.*, stronger adversarial perturbations are generated) as the IOU rate increases, which means that fewer proposals are filtered out and thus the set of targets $\mathcal{T}$ becomes larger. This is in line of our expectation, since DAG only guarantees misclassification on the targets in $\mathcal{T}$. The denser sampling on proposals allows the recognition error to propagate to other possible object positions better. Therefore, we choose a large IOU value (0.90) which produces good results.

3.5.2 Convergence

We then investigate the convergence of DAG, *i.e.*, how many iterations are needed to find the desired adversarial perturbation. Figure 4 shows the number of active targets, *i.e.*, $|\mathcal{T}_m|$, with respect to the number of iterations m. In general, the training process goes smoothly in the early rounds, in which we find that the number of active proposals is significantly reduced. After the algorithm reaches the maximal number of iterations, *i.e.*, 200 in segmentation and 150 in detection, only few (less than 1%) image fail to converge. Even on these cases, DAG is able to produce reasonable adversarial perturbations.

Another interesting observation is the difficulty in generating adversarial examples. In general, the detection networks are more difficult to attack than the segmentation networks, which is arguably caused by the much larger number of potential targets (recall that the total number of possible bounding boxes is one or two orders of magnitudes larger). Meanwhile, as the IOU rate increases, *i.e.*, a larger set $\mathcal{T}$ of proposals is considered, convergence also becomes slower, implying that more iterations are required to generate stronger adversarial perturbations.

3.5.3 Perceptibility

Following [32][23], we compute the perceptibility of the adversarial perturbation $\mathbf{r}$ defined by $p = \left(\frac{1}{K}\sum_k \|\mathbf{r}_k\|_2^2\right)^{1/2}$, where K is the number of pixels, and $\mathbf{r}_k$ is the intensity vector (3-dimensional in the RGB color space, $k = 1, 2, 3$) normalized in $[0, 1]$. We average the perceptibility value over the entire test set. In semantic segmentation, these values are 2.6×10^{-3}, 2.5×10^{-3}, 2.9×10^{-3} and 3.0×10^{-3} on

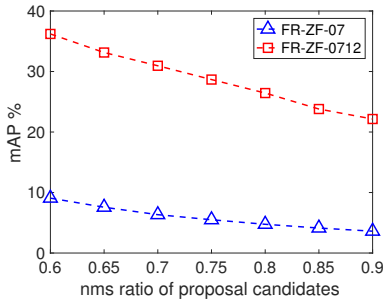

Figure 3: The mAP of using adversarial perturbations on **FR-ZF-07** to attack **FR-ZF-07** and **FR-ZF-0712**, with respect to the IOU rate. A larger IOU leads to a denser set of proposals.

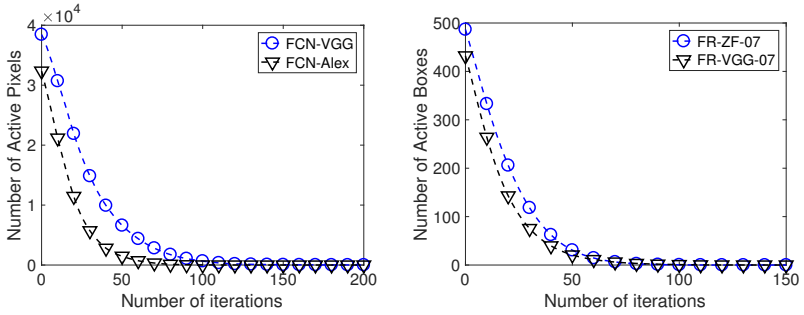

Figure 4: The convergence of DAG measured by the number of active targets, *i.e.*, $|\mathcal{T}_m|$, with respect to the number of iterations. Over the entire dataset, the average numbers of iterations are 31.78 and 54.02 for **FCN-Alex** and **FCN-VGG**, and these numbers are 47.05 and 41.42 for **FR-ZF-07** and **FR-VGG-07**, respectively.

FCN-Alex, **FCN-Alex***, **FCN-VGG** and **FCN-VGG***, respectively. In object detection, these values are 2.4×10^{-3}, 2.7×10^{-3}, 1.5×10^{-3} and 1.7×10^{-3} on **FR-ZF-07**, **FR-ZF-0712**, **FR-VGG-07** and **FR-VGG-0712**, respectively. One can see that these numbers are very small, which guarantees the imperceptibility of the generated adversarial perturbations. The visualized examples (Figures 1 and 2) also verify this point.

4. Transferring Adversarial Perturbations

In this section, we investigate the transfer ability of the generated adversarial perturbations. We use the adversarial perturbation computed on one model to attack other models. The attacked model may be trained on a different network architecture, or even targeted at a different vision task. Quantitative results are summarized in Tables 2–4. In the following parts, we analyze these results by organizing them into three categories, namely *cross-training* transfer, *cross-network* transfer and *cross-task* transfer.

4.1. Cross-Training Transfer

By *cross-training* transfer, we mean to apply the perturbations learned from one network to another network with the same architecture but trained on a different dataset. We observe that the transferability *largely* exists within the same network structure[2]. For example, using the adversarial perturbations generated by **FR-ZF-07** to attack **FR-ZF-0712** obtains a 22.15% mAP. This is a dramatic drop from the performance (61.07%) reported on the original images, although the drop is less than that observed in attacking **FR-ZF-07** itself (from 58.70% to 3.61%). Meanwhile, using the adversarial perturbations generated by **FR-ZF-0712** to attack **FR-ZF-07** causes the mAP drop from 58.70% to 13.14%, We observe similar phenomena when **FR-VGG-07** and **FR-VGG-0712**, or **FCN-Alex** and **FCN-Alex***, or **FCN-VGG** and **FCN-VGG*** are used to attack each other. Detailed results are shown in Tables 2 and 3.

[2]We also studied training on strictly non-overlapping datasets, *e.g.*, the model **FR-ZF-07** trained on **PascalVOC-2007** trainval set and the model **FR-ZF-12val** trained on **PascalVOC-2012** val set. The experiments deliver similar conclusions. For example, using **FR-ZF-07** to attack **FR-ZF-12val** results in a mAP drop from 56.03% to 25.40%, and using **FR-ZF-12val** to attack **FR-ZF-07** results in a mAP drop from 58.70% to 30.41%.

4.2. Cross-Network Transfer

We extend the previous case to consider the transferability through different network structures. We introduce two models which are more powerful than what we used to generate adversarial perturbations, namely DeepLab [3] for semantic segmentation and R-FCN [5] for object detection. For DeepLab [3], we use **DL-VGG** to denote the network based on 16-layer **VGGNet**[30], and use **DL-RN101** to denote the network based on 101-layer **ResNet**[13]. Both networks are trained on original DeepLab [3] training set which has 10582 images. For R-FCN [5], we use **R-FCN-RN50** to denote the network based on 50-layer **ResNet**[13], and use **R-FCN-RN101** to denote the network based on 101-layer **ResNet**[13]. Both networks are trained on the combined trainval sets of **PascalVOC-2007** and **PascalVOC-2012**. The perturbations applied to these four models are considered as black-box attacks [25], since DAG does not know the structure of these networks beforehand.

Detailed results are shown in Tables 2 and 3. Experiments reveal that transferability between different network structures becomes weaker. For example, applying the perturbations generated by **FR-ZF-07** leads to slight accuracy drop on **FR-VGG-07** (from 69.14% to 66.01%), **FR-VGG-0712** (from 72.07% to 69.74%), **R-FCN-RN50** (from 76.40% to 74.01%) and **R-FCN-RN101** (from 78.06% to 75.87%), respectively. Similar phenomena are observed in using different segmentation models to attack each other. One exception is using **FCN-VGG** or **FCN-VGG*** to attack **DL-VGG** (from 70.72% to 45.16% for **FCN-VGG** attack, or from 70.72% to 46.33% by **FCN-VGG*** attack), which results in a significant accuracy drop of **DL-VGG**. Considering the cues obtained from previous experiments, we conclude that adversarial perturbations are closely related to the architecture of the network.

Adversarial Perturbations from	**FR-ZF-07**	**FR-ZF-0712**	**FR-VGG-07**	**FR-VGG-0712**	**R-FCN-RN50**	**R-FCN-RN101**
None	58.70	61.07	69.14	72.07	76.40	78.06
FR-ZF-07 ($\mathbf{r}_1$)	**3.61**	22.15	66.01	69.47	74.01	75.87
FR-ZF-0712 ($\mathbf{r}_2$)	13.14	**1.95**	64.61	68.17	72.29	74.68
FR-VGG-07 ($\mathbf{r}_3$)	56.41	59.31	**5.92**	48.05	72.84	74.79
FR-VGG-0712 ($\mathbf{r}_4$)	56.09	58.58	31.84	**3.36**	70.55	72.78
$\mathbf{r}_1 + \mathbf{r}_3$	**3.98**	21.63	**7.00**	44.14	68.89	71.56
$\mathbf{r}_1 + \mathbf{r}_3$ (permuted)	58.30	61.08	68.63	71.82	76.34	77.71
$\mathbf{r}_2 + \mathbf{r}_4$	13.15	**2.13**	28.92	**4.28**	63.93	67.25
$\mathbf{r}_2 + \mathbf{r}_4$ (permuted)	58.51	61.09	68.68	71.78	76.23	77.71

Table 2: Transfer results for detection networks. **FR-ZF-07**, **FR-ZF-0712**, **FR-VGG-07** and **FR-VGG-0712** are used as four basic models to generate adversarial perturbations, and **R-FCN-RN50** and **R-FCN-RN101** are used as black-box models. All models are evaluated on the **PascalVOC-2007** test set and its adversarial version, which both has 4952 images.

Adversarial Perturbations from	**FCN-Alex**	**FCN-Alex***	**FCN-VGG**	**FCN-VGG***	**DL-VGG**	**DL-RN101**
None	48.04	48.92	65.49	67.09	70.72	76.11
FCN-Alex ($\mathbf{r}_5$)	**3.98**	7.94	64.82	66.54	70.18	75.45
FCN-Alex* ($\mathbf{r}_6$)	5.10	**3.98**	64.60	66.36	69.98	75.52
FCN-VGG ($\mathbf{r}_7$)	46.21	47.38	**4.09**	16.36	45.16	73.98
FCN-VGG* ($\mathbf{r}_8$)	46.10	47.21	12.72	**4.18**	46.33	73.76
$\mathbf{r}_5 + \mathbf{r}_7$	**4.83**	8.55	**4.23**	17.59	43.95	73.26
$\mathbf{r}_5 + \mathbf{r}_7$ (permuted)	48.03	48.90	65.47	67.09	70.69	76.04
$\mathbf{r}_6 + \mathbf{r}_8$	5.52	**4.23**	13.89	**4.98**	44.18	73.01
$\mathbf{r}_6 + \mathbf{r}_8$ (permuted)	48.03	48.90	65.47	67.05	70.69	76.05

Table 3: Transfer results for segmentation networks. **FCN-Alex**, **FCN-Alex***, **FCN-VGG** and **FCN-VGG*** are used as four basic models to generate adversarial perturbations, and **DL-VGG** and **DL-RN101** are used as black-box models. All models are evaluated on validation set in [18] and its adversarial version, which both has 736 images.

Adversarial Perturbations from	**FR-ZF-07**	**FR-VGG-07**	**FCN-Alex**	**FCN-VGG**	**R-FCN-RN101**
None	56.83	68.88	35.73	54.87	80.20
FR-ZF-07 ($\mathbf{r}_1$)	**5.14**	66.63	31.74	51.94	76.00
FR-VGG-07 ($\mathbf{r}_3$)	54.96	**7.17**	34.53	43.06	74.50
FCN-Alex ($\mathbf{r}_5$)	55.61	68.62	**4.04**	54.08	77.09
FCN-VGG ($\mathbf{r}_7$)	55.24	56.33	33.99	**4.10**	73.86
$\mathbf{r}_1 + \mathbf{r}_3 + \mathbf{r}_5$	**5.02**	**8.75**	**4.32**	37.90	69.07
$\mathbf{r}_1 + \mathbf{r}_3 + \mathbf{r}_7$	**5.15**	**5.63**	28.48	**4.81**	65.23
$\mathbf{r}_1 + \mathbf{r}_5 + \mathbf{r}_7$	**5.14**	47.52	**4.37**	**5.20**	68.51
$\mathbf{r}_3 + \mathbf{r}_5 + \mathbf{r}_7$	53.34	**5.94**	**4.41**	**4.68**	67.57
$\mathbf{r}_1 + \mathbf{r}_3 + \mathbf{r}_5 + \mathbf{r}_7$	**5.05**	**5.89**	**4.51**	**5.09**	64.52

Table 4: Transfer results between detection networks and segmentation networks. **FR-ZF-07**, **FR-VGG-07**, **FCN-Alex** and **FCN-VGG** are used as four basic models to generate adversarial perturbations, and **R-FCN-RN101** are used as black-box model. When attacking the first four basic networks, we use a subset of the **PascalVOC-2012** segmentation validation set which contains 687 images. In the black-box attack, we evaluate our method on the non-intersecting subset of 110 images.

4.3. Cross-Task Transfer

Finally, we investigate *cross-task* transfer, *i.e.*, using the perturbations generated by a detection network to attack a segmentation network or in the opposite direction. We use a subset of **PascalVOC-2012** segmentation validation set as our test set. Note that there are training images of **FR-ZF-07**, **FR-VGG-07**, **FCN-Alex** and **FCN-VGG** included in the **PascalVOC-2012** segmentation validation set, so we evaluate on the non-intersecting set of 687 images. Results are summarized in Table 4. We note that if the same network structure is used, *e.g.*, using **FCN-VGG** (segmentation) and **FR-VGG-07** (detection) to attack each other, the accuracy drop is significant (the mIOU of **FCN-VGG** drops from

54.87% to 43.06%, and the mAP of **FR-VGG-07** drops from 68.88% to 56.33%). Note that this drop is even more significant than *cross-network* transfer on the same task, which verifies our hypothesis again that the adversarial perturbations are related to the network architecture.

4.4. Combining Heterogeneous Perturbations

From the above experiments, we assume that different network structures generate roughly *orthogonal* perturbations, which means that if $\mathbf{r}_{\mathbb{A}}$ is generated by one structure $\mathbb{A}$, then adding it to another structure $\mathbb{B}$ merely changes the recognition results, *i.e.*, $\mathbf{f}^{\mathbb{B}}(\mathbf{X}, t_n) \approx \mathbf{f}^{\mathbb{B}}(\mathbf{X} + \mathbf{r}_{\mathbb{A}}, t_n)$. This motivates us to combine heterogeneous perturbations towards better adversarial performance. For example, if both $\mathbf{r}_{\mathbb{A}}$ and $\mathbf{r}_{\mathbb{B}}$ are added, we have $\mathbf{f}^{\mathbb{A}}(\mathbf{X} + \mathbf{r}_{\mathbb{A}} + \mathbf{r}_{\mathbb{B}}, t_n) \approx \mathbf{f}^{\mathbb{A}}(\mathbf{X} + \mathbf{r}_{\mathbb{A}}, t_n)$ and $\mathbf{f}^{\mathbb{B}}(\mathbf{X} + \mathbf{r}_{\mathbb{A}} + \mathbf{r}_{\mathbb{B}}, t_n) \approx \mathbf{f}^{\mathbb{B}}(\mathbf{X} + \mathbf{r}_{\mathbb{B}}, t_n)$. Thus, the combined perturbation $\mathbf{r}_{\mathbb{A}} + \mathbf{r}_{\mathbb{B}}$ is able to confuse both network structures.

In Tables 2–4, we list some results by adding multiple adversarial perturbations. Also, in order to verify that the spatial structure of combined adversarial perturbations is the key point that leads to statistically significant accuracy drop, we randomly generate three permutations of the combined adversarial perturbations and report the average accuracy. From the results listed in Tables 2–4, we can observe that adding multiple adversarial perturbations often works better than adding a single source of perturbations. Indeed, the accuracy drop caused by the combined perturbation approximately equals to the sum of drops by each perturbation. For example, the adversarial perturbation $\mathbf{r}_2 + \mathbf{r}_4$ (combining **FR-ZF-0712** and **FR-VGG-0712**) causes significant mAP drop on all **ZFNet**-based and **VGGNet**-based detection networks, and the adversarial perturbation $\mathbf{r}_5 + \mathbf{r}_7$ (combining **FCN-Alex*** and **FCN-VGG***) causes significant mIOU drop on all **AlexNet**-based and **VGGNet**-based segmentation networks. However, permutation destroys the spatial structure of the adversarial perturbations, leading to negligible accuracy drops. The same conclusion holds when the perturbations from different tasks are combined. Table 4 shows some quantitative results of such combination and Figure 5 shows an example. Note that, the perceptibility value defined in Section 3.5.3 remains very small even when multiple adversarial perturbations are combine (*e.g.*, 4.0×10^{-3} by $\mathbf{r}_1 + \mathbf{r}_3 + \mathbf{r}_5 + \mathbf{r}_7$).

4.5. Black-Box Attack

Combining heterogeneous perturbations allows us to perform better on the so-called *black-box attack* [25], in which we do not need to know the detailed properties (architecture, purpose, *etc.*) about the defender network. According to the above experiments, a simple and effective way is to compute the sum of perturbations from several

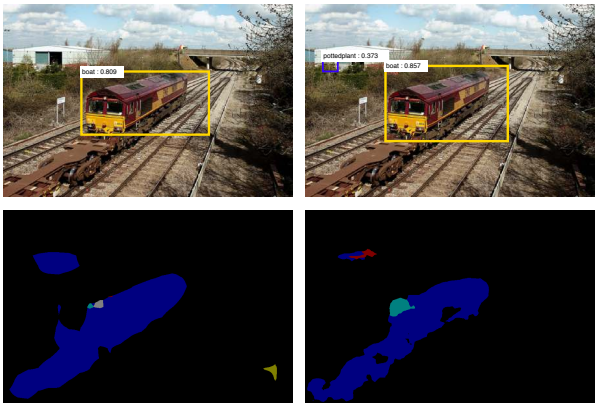

Figure 5: Adding the fused adversarial perturbation ($\mathbf{r}_1 + \mathbf{r}_3 + \mathbf{r}_5 + \mathbf{r}_7$, see Table 4) confuses four different networks. The top row shows **FR-VGG-07** and **FR-ZF-07** detection results, and the bottom row shows **FCN-Alex** and **FCN-VGG** segmentation results. The blue in segmentation results corresponds to *boat*.

of known networks, such as **FR-ZF-07**, **FR-VGG-07** and **FCN-Alex**, and use it to attack an unknown network. This strategy even works well when the structure of the defender is not investigated before. As an example shown in Table 4, the perturbation $\mathbf{r}_1 + \mathbf{r}_3 + \mathbf{r}_5 + \mathbf{r}_7$ leads to significant accuracy drop (from 80.20% to 64.52%) on **R-FCN-RN101**[5], a powerful network based on the deep **ResNet** [13].

5. Conclusions

In this paper, we investigate the problem of generating adversarial examples for semantic segmentation and object detection. We propose DAG algorithm, which is able to effectively generate visually imperceptible perturbation, so that we can confuse the originally correct recognition results in a well controllable manner. An intriguing property of the perturbation generated by DAG is the transfer ability. We show that the perturbation can be transferred across different training sets, different network architectures and even different tasks. Combining heterogeneous perturbations often leads to more effective adversarial perturbations in black-box attacks.

The transfer ability also suggests that deep networks, though started with different initialization and trained in different ways, share some intrinsic structure, which make them sensitive to a similar source of perturbations. This reveals an interesting topic for future research.

Acknowledgements

This work is supported by the Intelligence Advanced Research Projects Activity (IARPA) via DoI/IBC contract number D16PC00007, and also by the grant ONR–N00014-15-1-2356. We thank Dr. Vittal Premachandran, Weichao Qiu, Chenxi Liu, Zhuotun Zhu, Chenxu Luo and Siyuan Qiao for instructive discussions.

References

[1] S. Baluja and I. Fischer. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387*, 2017.

[2] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*. IEEE, 2017.

[3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[4] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Computer Vision and Pattern Recognition*. IEEE, 2016.

[5] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*, 2016.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*. IEEE, 2009.

[7] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*, 2014.

[8] M. Everingham, A. Zisserman, C. K. Williams, L. Van Gool, M. Allan, C. M. Bishop, O. Chapelle, N. Dalal, T. Deselaers, G. Dorkó, et al. The pascal visual object classes challenge 2007 (voc2007) results. 2007.

[9] V. Fischer, M. C. Kumar, J. H. Metzen, and T. Brox. Adversarial examples for semantic image segmentation. In *International Conference on Learning Representations Workshop*, 2017.

[10] R. Girshick. Fast r-cnn. In *International Conference on Computer Vision*. IEEE, 2015.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*. IEEE, 2014.

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*. IEEE, 2016.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[15] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017.

[16] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Computer Vision and Pattern Recognition*. IEEE, 2017.

[17] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations*, 2017.

[18] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition*. IEEE, 2015.

[19] Y. Luo, X. Boix, G. Roig, T. Poggio, and Q. Zhao. Foveation-based mechanisms alleviate adversarial examples. *arXiv preprint arXiv:1511.06292*, 2015.

[20] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff. On detecting adversarial perturbations. In *International Conference on Learning Representations*, 2017.

[21] J. H. Metzen, M. C. Kumar, T. Brox, and V. Fischer. Universal adversarial perturbations against semantic image segmentation. *arXiv preprint arXiv:1704.05712*, 2017.

[22] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *Computer Vision and Pattern Recognition*. IEEE, 2017.

[23] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Computer Vision and Pattern Recognition*. IEEE, 2016.

[24] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Computer Vision and Pattern Recognition*. IEEE, 2015.

[25] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *ACM on Asia Conference on Computer and Communications Security*. ACM, 2017.

[26] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy*. IEEE, 2016.

[27] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015.

[28] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR Workshops*. IEEE, 2014.

[29] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *Computer Vision and Pattern Recognition*. IEEE, 2015.

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition*. IEEE, 2015.

[32] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[33] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[34] J. Wang, Z. Zhang, C. Xie, V. Premachandran, and A. Yuille. Unsupervised learning of object semantic parts from internal states of cnns by population encoding. *arXiv preprint arXiv:1511.06855*, 2015.

[35] S. Xie and Z. Tu. Holistically-nested edge detection. In *International Conference on Computer Vision*. IEEE, 2015.

[36] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*. Springer, 2014.

[37] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *International Conference on Computer Vision*. IEEE, 2015.

A. Supplementary Material

A.1. Transferable Examples for Semantic Segmentation and Object Detection

As shown in the main paper, adversarial examples can be transformed across networks with different training data. based on different architectures, and even for different tasks. Some typical examples are shown in Figure 1, where the adversarial examples from Network 1 are able to transfer to Network 2.

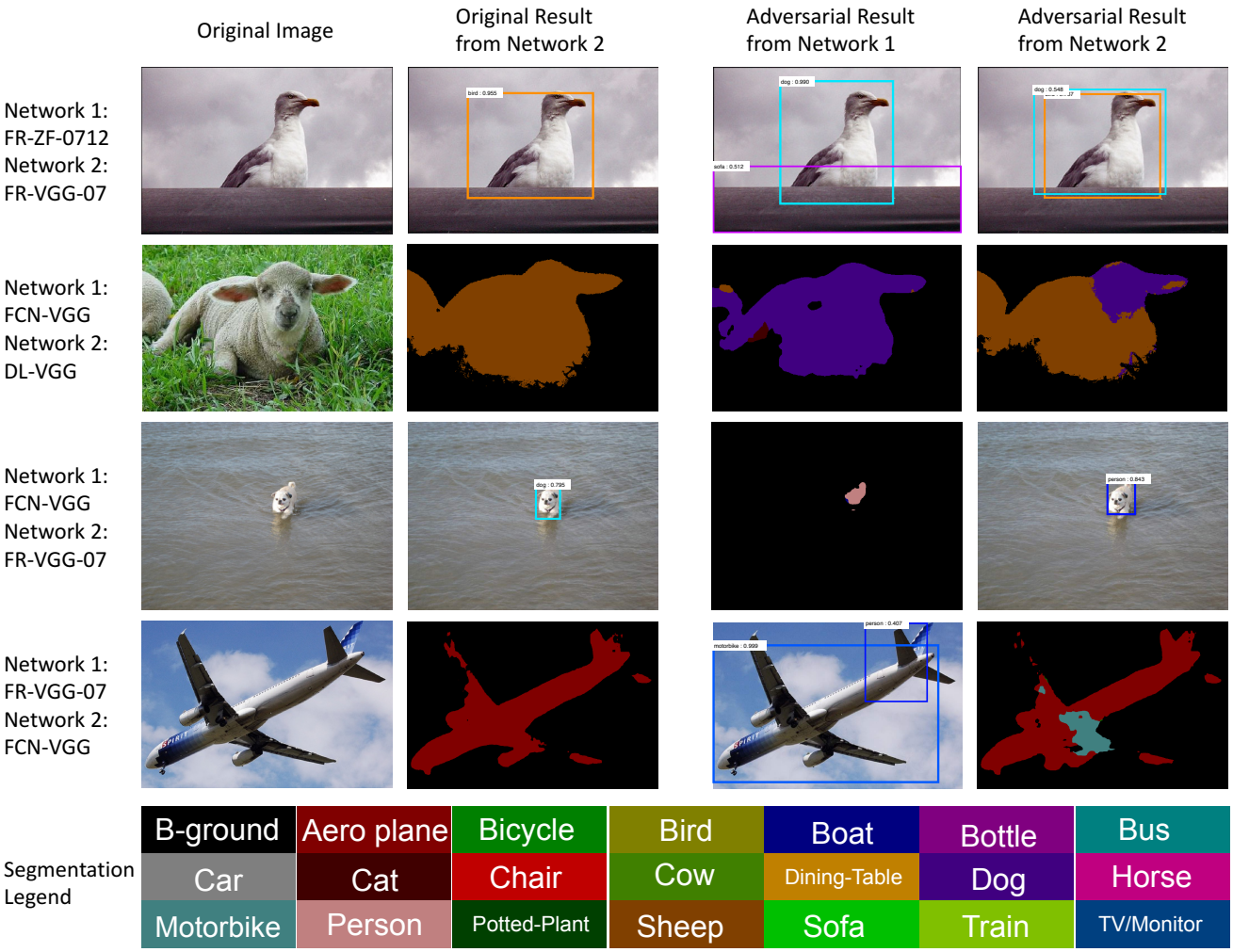

Figure 1. Transferrable examples for semantic segmentation and object detection. These four rows, from top to bottom, shows the adversarial attack examples within two detection networks, within two segmentation networks, using a segmentation network to attack a detection network and in the opposite direction. The segmentation legend borrows that in [1].

A.2. Generating Geometric Patterns

As an additional showcase, the deep segmentation networks can be confused to output some geometric shapes, including *stripes*, *circles*, *triangles*, *squares*, *etc.*, after different adversarial perturbations is added to the original image. Results are shown in Figure 2. Here, the added adversarial perturbation varies from case to case.

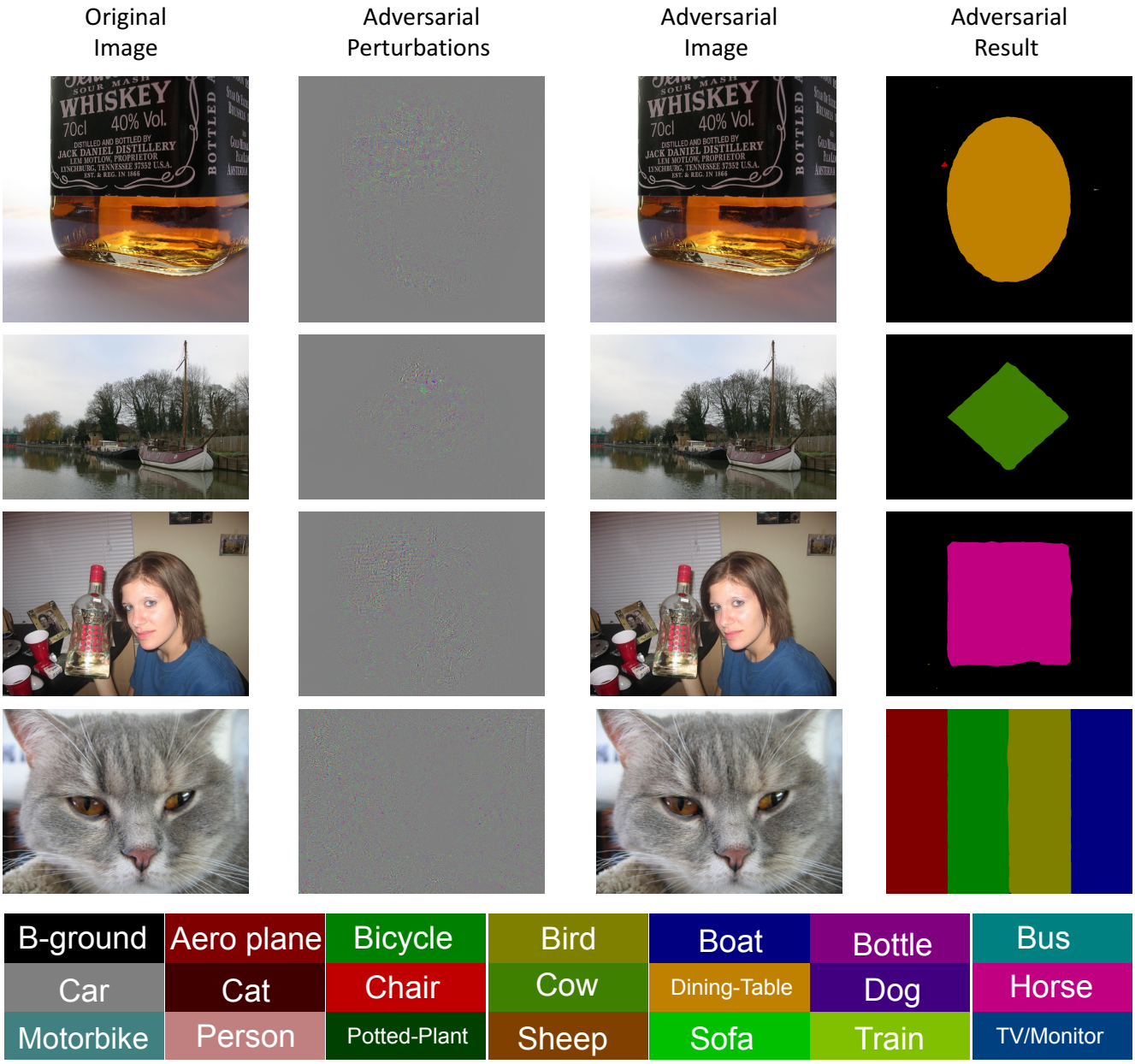

Figure 2. The adversarial perturbations confuse the deep networks to output different geometric patterns as segmentation results, such as a circle (the first row), a diamond (the second row), a square (the third row), and stripes (the fourth row). Here, **FCN-Alex** is used as the baseline network (defender). All the perturbations are **magnified by 10** for better visualization. The segmentation legend borrows that in [1].

A.3. Same Noise, Different Outputs

In Figure 2 of the main article, we show that we can generate some adversarial perturbations to make a deep segmentation network output a pre-specified segmentation mask (*e.g.*, `ICCV` and `2017`). But, the perturbations used to generate these two segmentation masks are different.

Here, we present a more challenging task, which uses the same perturbations to confuse two networks. More specifically, we hope to generate a perturbation $\mathbf{r}$, when it is added to an image $\mathbf{X}$, the **FCN-Alex** and the **FCN-VGG** models are confused to output `ICCV` and `2017`, respectively. To implement this, we apply the locally linear property of the network, and add two sources of perturbations, *i.e.*, $\mathbf{r} = \mathbf{r}_1 + \mathbf{r}_2$, where $\mathbf{r}_1$ is generated on **FCN-Alex** with the mask `ICCV`, and $\mathbf{r}_2$ is generated on **FCN-VGG** with the mask `2017`. As shown in Figure 3, our simple strategy works very well, although the segmentation boundary of each letter or digit becomes somewhat jagged.

Figure 3. We add one adversarial perturbation (**magnified by 10**) to the same original image to generate different pre-specified segmentation masks on two deep segmentation networks (**FCN-Alex** and **FCN-VGG**). This is a more difficult task compared to that shown in Figure 2 of the main article, where two different adversarial perturbations are used to generate two pre-specified segmentation masks. The blue regions in the segmentation masks are predicted as *bus*, a randomly selected class.

References

[1] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *International Conference on Computer Vision*. IEEE, 2015.