arXiv:1902.02067v2 [cs.CV] 15 Oct 2019

Daedalus: Breaking Non-Maximum Suppression in Object Detection via Adversarial Examples

Derui Wang*
Swinburne University of technology
deruiwang@swin.edu.au

Chaoran Li†
Swinburne University of technology
chaoranli@swin.edu.au

Sheng Wen
Swinburne University of technology
swen@swin.edu.au

Xiaojun Chang
Monash University
cxj273@gmail.com

Surya Nepal
CSIRO's Data61
surya.nepal@data61.csiro.au

Yang Xiang
Swinburne University of technology
yxiang@swin.edu.au

ABSTRACT

We demonstrate that Non-Maximum Suppression (NMS), which is commonly used in Object Detection (OD) tasks to filter redundant detection results, is no longer secure. Considering that NMS has been an integral part of OD systems, thwarting the functionality of NMS can result in unexpected or even lethal consequences for such systems. In this paper, we propose an adversarial example attack which triggers malfunctioning of NMS in end-to-end OD models. Our attack, namely `Daedalus`, compresses the dimensions of detection boxes to evade NMS. As a result, the final detection output contains extremely dense false positives. This can be fatal for many OD applications such as autonomous vehicle and surveillance system. Our attack can be generalised to different end-to-end OD models, such that the attack cripples various OD applications. Furthermore, we propose a way to craft robust adversarial examples by using an ensemble of popular detection models as the substitutes. Considering the pervasive nature of model reusing in real-world OD scenarios, Daedalus examples crafted based on an *ensemble of substitutes* can launch attacks without knowing the parameters of the victim models. Our experiments demonstrate that the attack effectively stops NMS from filtering redundant bounding boxes. As the evaluation results suggest, Daedalus increases the false positive rate in detection results to 99.9% and reduces the mean average precision scores to 0, while maintaining a low cost of distortion on the original inputs. With the widespread applications of OD, our work shows that there are serious vulnerabilities in the fundamental components of such systems and further investigation on them is required in this area.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Computer vision**; **Object detection**;

KEYWORDS

Deep neural network, adversarial example, object detection, cyber security

*Derui is also with CSIRO's Data61.
†Chaoran is also with CSIRO's Data61.

Anonymous Submission to ACM CCS 2019, Due 15 May 2019, London, United Kingdom 2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Figure 1: A demo of the Daedalus attack on YOLO-v3. The upper-left image is an original image taken from COCO dataset [31]. The OD results are shown in the upper-right image. An adversarial example as well as the OD results of the adversarial example are shown in the second row. The adversarial example significantly increases the number of final detection boxes after NMS and makes the detector non-viable. However, the perturbation on the adversarial example is still imperceptible.

1 INTRODUCTION

Object Detection (OD) is a fundamental operation in the field of computer vision and is repeatedly used in the areas of autonomous vehicles, robotics, surveillance system, and biometric authentications. Convolutional Neural Network (CNN) is embedded in the core of many state-of-the-art OD models. These models can be classified into two broad categories: single-stage detection models (e.g. SSD [33], RetinaNet [30], You-Only-Look-Once (YOLO)[47], YOLO-9000 [48], and YOLO-v3 [49]) and two-stage detection models (e.g. R-CNN [15] and Fast R-CNN [14]). Single-stage ones have become successful since they perform end-to-end detection which has low

computational overhead and is competitively accurate. Current end-to-end detection models are mostly built upon Fully Convolutional Network (FCN) backbones. However, FCNs output a large number of redundant detection results that shall be filtered before the final results are produced.

To filter the redundant detection boxes, Non-Maximum Suppression (NMS) is performed as a key step in the OD procedure [43]. NMS has been an integral component in many OD algorithms for around 50 years. It relies on non-differentiable, hard-coded rules to discard redundant boxes. It is used in both single-stage and two-stage detection algorithms. Beyond OD, NMS is also being used in face recognition, face verification and keypoint detection [50]. NMS has also inspired refinement algorithms in two-stage detection models [6]. Some recent works tried to use a differentiable model to learn NMS [22–24, 46]. Another work suggested using soft-NMS to improve the filtering performance [3]. Therefore, NMS is very important to OD systems, once the NMS component in an OD system does not work properly, the system fails.

CNN is known to be vulnerable to adversarial example attacks [1]. To a human eye, an adversarial image can be indistinguishable from the original one, but it can cause misclassification in a CNN based classifier. Recently, an adversarial example has been crafted for OD and segmentation tasks [58]. This attack follows the same principles as the ones that craft adversarial examples for classifiers. Such attacks on detection models, result in the misclassification of the detected objects. All the above adversarial examples attack by triggering misclassification in the victim models. Thus, we refer to these examples as misclassification adversarial examples. However, false positive rate is another critical metric in OD missions such as pedestrian detection [13] and military object recognition [60]. Therefore, an attack that boosts the false positive rate of an OD system is as lethal as the misclassification adversarial examples.

In this paper, we propose a novel adversarial example attack for NMS, which can be universally applied to any other end-to-end differentiable OD model. A demonstration of our attack is shown in Fig.1. As it can be seen, a properly crafted adversarial example can trigger malfunctioning of the NMS component in YOLO-v3, in a way that it outputs dense and noisy detection results. We name this attack `Daedalus` since it creates a maze composed of false positive detection results.

We first evaluate and analyse how Daedalus reduces the OD performance by breaking NMS[1]. Furthermore, Daedalus is adapted for launching black-box attack. Based on a census in Section 2.2, we observe two facts: 1) current OD tasks are thriving on a few types of OD models, even many models have been proposed; 2) current OD tasks often load trained parameters into a model instead of reparameterising the model. Therefore, if an attacker can generate an universal adversarial example which works effectively against all the popular models, the example has a high chance to successfully sabotage a black-box model. Herein, we propose using an *ensemble of substitutes* (i.e., an ensemble of popular detection models as the substitutes) in Daedalus attack to break the black-box OD models. We exploit the latest version of YOLO (i.e. YOLO-v3 [49]), SSD [33], and RetinaNet [30] as the victims. As suggested by our experiments, universal Daedalus examples generated by using an ensemble of popular OD models can launch attacks in a black-box scenario. In brief, we summarise our contributions as follows:

- *We propose a novel adversarial example attack which aims at breaking NMS (unlike the existing state-of-the art attacks that merely target misclassification). As a consequence, the attack creates a large number of false positives in the detection results;*
- *We propose new adversarial loss functions to generate NMS adversarial examples. We further propose an ensemble-of-substitutes attack that generates robust adversarial examples which can make NMS malfunction in multiple models, simultaneously. In such way, we can partially convert black-box attacks to white-box attacks;*
- *We adopt two adaptive defences aiming to mitigate the attack. We evaluate our attack and show that it is effective against the state-of-the-art OD models with/without the defences.*

The paper is organised as follows: Section 2 introduces the background knowledge of adversarial example, object detection, and NMS. Section 3 describes the design of Daedalus attack. Section 4 compares and selects loss functions from the candidates, and systematically evaluates the performance of our attack. Section 5 discusses the possibility of applying Daedalus adversarial examples on soft-NMS, RetinaNet, and black-box models. We also post the dilemma of defending against Daedalus attack in this section. Section 6 reviews the related work relevant to our study. Section 7 draws a conclusion on the paper and presents our future work.

2 PRELIMINARY

2.1 Adversarial examples

Given a DNN model F and a benign data example x, an attacker can craft an adversarial examples of x by altering the features of x by adding a perturbation δ, such that $F(x+\delta) \neq F(x)$. Generating adversarial examples can be interpreted as optimising the examples features towards adversarial objectives. Gradients-based optimisation methods are commonly employed to search for adversarial examples. The optimisation objective can be generally parsed as follows:

$$\begin{aligned} &\underset{\delta}{\operatorname{argmin}}\, |\delta|_p + c \cdot L(F(x+\delta), y) \\ &s.t.\ \ x+\delta \in [0,1]^n, \end{aligned} \tag{1}$$

wherein , L is an adversarial objective set by the attacker, and y is either a ground truth output or a desired adversarial output for F. $x+\delta$ produces an adversarial example x_{adv}. $|\delta|_p$ is the allowed distortion bounded by the p-norm of δ. A constant c is adopted to balance the adversarial loss and the distortion. This box-constrained non-convex optimisation problem is typically solved by gradient descent methods.

2.2 A census of object detection models

To better understand the impact of attacks against OD tasks, we first need to overview the distribution of models used in OD. We conducted a census of OD projects on Github. We crawled project names, project descriptions, and readme files from 1, 696 OD-related

[1]In this paper, the term *breaking NMS* means that NMS cannot filter redundant bounding boxes detected from an adversarial example.

Table 1: Object detection Models on Github

Rank	Model	Popularity
1	SSD	605
2	YOLO-v3	251
3	Faster R-CNN	236
4	R-CNN	178
5	YOLO-v2	164
6	RetinaNet	75
7	YOLO-v1	52
Popular models/Total		1,561/1,696

repositories on Github. Next, based on the reviewed models in [32], 45 released OD models were selected as keywords. For each repository, we counted the keywords in the readme file, and selected the most frequent one as the model adopted for the repository. Finally, we grouped the repositories by models. The ranking of models that appeared more than 50 times is presented in Table 1. Based on our analysis, $1,561$ out of $1,696$ repositories use models from Table 1. The observation suggests that models and backbones used for OD tasks are highly limited in number. Moreover, we analysed the source codes from the repositories to search for operations of loading trained models and parameters. We observed that almost all the source codes provide an interface for loading pre-trained parameters into the model backbones. We further investigated names of the pre-trained models and parameters. We found that many repositories use the same pre-tained parameters in their models.

Based on the observations, black-box attacks towards OD models can be partially converted to white-box attacks. An attacker can first craft a universal adversarial example which breaks all the popular white-box OD models, and then launch attacks on black-box models using the example. The attack has a high potential to break the black-box models since that the black-box models may have the same parameters with the white-box models.

2.3 Object detection

Given an input frame, an end-to-end OD model outputs detected regression boxes and classification probabilities through a FCN. Herein, we adopt the third version of You Only Look Once detector (YOLO) [47] as an example to introduce the kownledge of end-to-end detection algorithms. YOLO has achieved the state-of-the-art performance on standard detection tasks such as PASCAL VOC [11] and COCO [31]. Over the past years, YOLO has been updated to YOLO-v2/YOLO-9000 [48], and the current YOLO-v3 [49]. YOLO has been applied in video OD [52] and robotics [36]. YOLO has been studied extensively in the research of road sign detection and recognition for autonomous car [61].

The backbone of YOLO-v3 is a 53-layer FCN (i.e. Darknet-53). YOLO-v3 segments the input image using three mesh grids in different scales (i.e., 13×13, 26×26, 52×52). The model outputs detection on these three scales. Each grid contains three anchor boxes, whose dimensions are obtained by K-means clustering on the dimensions of ground truth bounding boxes. Each grid outputs three bounding box predictions. Henceforth, there are totally 10647 bounding boxes in the output of YOLO-v3. The Darknet-53 backbone outputs a feature map which contains bounding box parameters, box confidence, and object class probabilities. Specifically, for each bounding box, the feature map includes its height t_h, width t_w, center coordinates (t_x, t_y), as well as class probabilities $p_1, p_2, ..., p_n$ and the objectness t_0. t_0 is the confidence that a box contains an object. Given the feature map, the position of each detection box is then calculated based on anchor box dimension priors p_w and p_h, and centre offsets (c_x, c_y) from the top-left corner of the image. The final box dimension and position will then be:

$$\begin{aligned} b_x &= c_x + \theta(t_x) \\ b_y &= c_y + \theta(t_y) \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \end{aligned} \tag{2}$$

Herein, b_w is the box width, and b_h is the box height. b_x and b_y are the box center coordinates. The box confidence b_0 is calculated as:

$$b_0 = t_0 \cdot \max\{p_1, p_2, ..., p_n\} \tag{3}$$

YOLO-v3 finally outputs the bounding boxes, the box confidences, and the class probabilities. The outputs will then be processed by NMS to generate the final detection results.

2.4 Non-Maximum Suppression (NMS)

NMS is integrated into OD algorithms to filter detection boxes. NMS makes selections based on the Intersection over Union (IoU) between detection boxes. IoU measures the ratio of the overlapped area over the union area between two boxes. NMS works in two steps: 1) For a given object category, all of the detected bounding boxes in this category (i.e., candidate set) are sorted based on their box confidence scores from high to low; 2) NMS selects the box which has the highest box confidence score as the detection result, and then it discards other candidate boxes whose IoU value with the selected box is beyond the threshold. Then, within the remaining boxes, NMS repeats the above two steps until there is no remaining box in the candidate set. Suppose the initial detection boxes are $B = b_1, b_2, ..., b_n$, the corresponding box confidence scores are $S = s_1, s_2, ..., s_n$. Given an NMS threshold N_t, we can write the NMS algorithm as Algorithm 1:

Algorithm 1: Non-Maximum Suppression

```
Input: B, S, N_t
Initialisation:
D ← {}
while B ≠ empty do
    m ← argmax S
    M ← b_m
    D ← D ∪ B
    B ← B − M
    for b_i ∈ B do
        if IoU(M, b_i) ≥ N_t then
            B ← B − b_i
            S ← S − s_i
Output: D, S.
```

We can observe from the Algorithm 1 that NMS recursively discards the redundant detection boxes from the raw detection proposals B, until all raw proposals have been processed.

3 DAEDALUS ATTACK

We introduce our attack, namely Daedalus, in this section. To clarify the intuition behind the attack, we first analyse the vulnerability that we exploit to break NMS. Then, we introduce the general optimisation techniques used for making adversarial examples. Subsequently, we propose a series of adversarial loss functions to optimise in order to generate Daedalus adversarial examples. At last, we assemble the algorithms together to present the Daedalus attack, which can force a detection network to output an adversarial feature map that causes the malfunctioning of NMS.

3.1 Attacking NMS

We first introduce the vulnerability, which enables our attack, in NMS. NMS filters boxes based on the IoU between boxes. Specifically, after the raw detection boxes are obtained, the boxes with a confidence score below a given threshold will be discarded. Next, for a given object class and a box of this class with the highest box confidence score (i.e., the final detection box), NMS will discard other remaining boxes based on their IoUs with the final detection box. This mechanism makes NMS vulnerable to attacks that suppress the IoUs between output bounding boxes. Once the IoUs are suppressed below the required threshold for NMS filtering, NMS can no longer function normally. In this case, most of the redundant bounding boxes will be kept in the final detection results.

We consider three elemental schemes to break NMS. First, for algorithms such as YOLO-v3, it is necessary to make most of the bounding boxes survive the first round of filtering that discards boxes based on the box confidences. Hence, we need to maximise the box confidences. Second, to compress the IoUs, we can directly minimise the IoU for each pair of boxes. Alternatively, we can minimise the expectation of the box dimension over all the bounding boxes, and maximise the expectation of the Euclidean distance between box centres over all pairs of boxes. The schemes are illustrated in Fig.2.

3.2 Generating adversarial example

We denote a benign example as x. An adversarial example of x is denoted as x'. The adversarial perturbation is $\delta = x' - x$. Therefore, the distortion calculated based on p-norm of the perturbation is $\|\delta\|_p$. We can then formulate the attack as the following optimisation problem:

$$\begin{aligned} &\underset{\delta}{\operatorname{argmin}} \ \|\delta\|_p + c \cdot f(x+\delta) \\ &s.t. \ \ x + \delta \in [0,1]^n, \end{aligned} \tag{4}$$

wherein, f is an adversarial loss function. We will later select f from the loss functions presented in the previous section. c is a constant to balance the distortion and the adversarial loss f. δ is the adversarial perturbation. An adversary optimises δ to search for an adversarial example whose pixel values are bounded between 0 and 1. This is a typical box-constraint optimisation problem. To bound the pixel values of the generated adversarial examples between 0 and 1, we adopt the idea of changing variables from the paper [5]. We change the optimisation variables from δ to ω. There are different functions we can use for changing the variables (e.g. sigmoid function, arctan function, hyperbolic tangent function etc.). Among these functions, the hyperbolic tangent function generally produces higher gradients than others. Henceforth, to make the optimisation converge faster, we select the hyperbolic tangent function to change the variables. For δ_i in δ, we can apply a hyperbolic tangent function such that:

Figure 2: The schemes we used to attack NMS. The scheme on the left side directly minimises IoUs between all box pairs. The second scheme takes a low-cost approximation of the first scheme. It minimises the dimension for all boxes, and maximises the Euclidean distance between box centres for all boxes.

$$\delta_i = \frac{1}{2}(tanh(\omega_i) + 1) - x_i. \tag{5}$$

We then optimise the above task based on the new variables ω. We apply binary search to find the best c during the optimisation. Finally, the adversarial perturbation δ can be calculated easily through the inverse of function 5.

3.3 Adversarial loss functions

In this section, we design loss functions that are used to find adversarial examples, which trigger malfunctioning of NMS. Based on the discussion in Section.3.1, we formulate our attack into three potential adversarial loss functions. We will evaluate each loss function in Section.4. There are totally n detection boxes (e.g., $n = 10647$ in YOLO-v3). Supposing the object detector can be regarded as a function F. Given an input image x, the outputs are $F(x) = \{B^x, B^y, B^w, B^h, B^0, P\}$. Herein, $B^x = \{b_0^x, b_1^x, ..., b_n^x\}$, $B^y = \{b_0^y, b_1^y, ..., b_n^y\}$, $B^w = \{b_0^w, b_1^w, ..., b_n^w\}$, and $B^h = \{b_0^h, b_1^h, ..., b_n^h\}$. They are the dimensions and coordinates of the n output bounding boxes. $B^0 = \{b_0^0, b_1^0, ..., b_n^0\}$ are the objectness scores and $P = \{p_0, p_1, ..., p_n\}$ are the class probabilities as introduced in Section 2.3.

Our attack can specify an object classified as category λ to attack. If we want to attack multiple object categories, we can include these categories in a set Λ. Based on the above discussion, we define three loss functions, f_1, f_2, and f_3, as follows:

$$f_1 = \frac{1}{\|\Lambda\|} \sum_{\lambda \in \Lambda} \mathop{\mathbb{E}}_{i:argmax(p_i)=\lambda} \{[b_i^0 \cdot max(p_i) - 1]^2 + \mathop{\mathbb{E}}_{j:argmax(p_j)=\lambda} IoU_{ij}\}, \tag{6}$$

$$f_2 = \frac{1}{\|\Lambda\|} \sum_{\lambda \in \Lambda} \mathop{\mathbb{E}}_{i:argmax(p_i)=\lambda} \{[b_i^0 \cdot max(p_i) - 1]^2 + (\frac{b_i^w \cdot b_i^h}{W \times H})^2 + \mathop{\mathbb{E}}_{j:argmax(p_j)=\lambda} \frac{1}{(b_i^x - b_j^x)^2 + (b_i^y - b_j^y)^2}\}, \tag{7}$$

$$f_3 = \frac{1}{\|\Lambda\|} \sum_{\lambda \in \Lambda} \mathop{\mathbb{E}}_{i:argmax(p_i)=\lambda} \{[b_i^0 \cdot max(p_i) - 1]^2 + (\frac{b_i^w \cdot b_i^h}{W \times H})^2\}, \tag{8}$$

wherein, IoU_{ij} is the IoU between the i-th and the j-th bounding boxes. Box dimensions are scaled between 0 and 1, dividing b^w and b^h by the input image width W and height H, such that the term is invariant towards changing input dimension.

In f_1, we first minimise the expectation of mean squared error between the box confidence and 1 for all the boxes in which the detected objects are in the attacked category set Λ. Henceforth, the boxes will not be discarded due to low box confidence scores. In the second term, f_1 minimises the expectation of IoUs for all pairs of bounding boxes in Λ. When the IoUs fall below the NMS threshold, the boxes can evade the NMS filter. Alternatively, we minimise the expectation of box dimensions and maximise the expectation of box distances in f_2. f_2 approximates the effect of directly minimising IoU by compressing box dimensions and distributing boxes more evenly on the detected region.

Pairwise computation for obtaining IoUs and box distances could be expensive if the output feature map contains too many boxes. Therefore, we also propose f_3 as a more efficient loss function. f_3 only minimises the expectation of box dimensions instead of directly minimising the expectation of the IoUs. Minimising f_3 leads to smaller bounding boxes. When the boxes become small enough, there will be no intersection between the boxes. In other words, the IoUs between the boxes will be suppressed to zero. As a consequence, the boxes can avoid being filtered by NMS.

3.4 Ensemble of substitutes

In this section, we design a loss function that converts black-box attacks into white-box ones by observing the current OD tasks (q.v. Section 2.2). Transferability of adversarial examples is usually considered as a pathway for launching black-box attacks. The transferability of misclassification adversarial examples has been investigated in [20]. However, instead of causing misclassification, our attack takes a different attacking purpose (i.e. disabling NMS). It is difficult to transfer Daedalus examples between OD models with different backbones since the feature maps extracted by the backbones are highly divergent. Herein, we take another pathway to attack black-box models in the fields of OD.

To launch a black-box attack using Daedalus, we suggest using an ensemble of popular OD models as the substitute to generate adversarial examples. This is largely based on the observed fact that current OD tasks tend to reuse few popular OD models with pre-trained backbones. As a consequence, the generated adversarial examples can be effective against all the popular models. Such a universal adversarial example has a high chance to break NMS of a black-box model. To find such an example, we can optimise a loss function as an expectation over OD models:

$$f = \mathop{\mathbb{E}}_{m \sim M} f^m, \tag{9}$$

herein, M is a set of OD models with popular backbones, which includes the popular OD models and backbones. f^m is the loss value of the m-th model in M, calculated based on one of Equations 6,7, and 8.

3.5 Daedalus attack against NMS

In this section, we assemble the above methods to form Daedalus attack. Actually, our attack can be applied to all FCN-based detectors. FCN-based detectors output the detected box dimension, box position and classification probabilities in a fully differentiable, end-to-end manner. Therefore, our attack can compute end-to-end adversarial gradients and optimise adversarial examples for FCN-based detectors.

Algorithm 2: L_2 Daedalus attack

Input: $x, \Lambda, \gamma, binary_steps, \eta, max_iteration, c_{max}, c_{min}$
Initialisation:
$c \leftarrow 10$
$loss_{init} \leftarrow f(x)$
$\delta \leftarrow 0$
$x^* \leftarrow x + \delta$
for *n starts from 0 to* $binary_steps$ **do**
  for *i starts from 0 to* $max_iteration$ **do**
    select boxes in Λ to calculate loss $f(x^*)$
    $\delta \leftarrow \delta - \eta \nabla_\delta [\|\delta\| + c \cdot f(x^*)]$
    $x^* \leftarrow x^* + \delta$
  $x' \leftarrow$ the best x^* found.
  if $f(x^*) <= loss_{init} \cdot (1 - \gamma)$ **then**
    $c_{max} = min(c, c_{max})$
    $c \leftarrow 0.5 \cdot (c_{max} + c_{min})$
  else
    $c_{min} = max(c, c_{min})$
    $c \leftarrow 0.5 \cdot (c_{max} + c_{min})$
Output: Adversarial example x'.

Given an OD task with an image x as an input, we are going to maximise the number of bounding box output from NMS. As an example, YOLO-v3 outputs three feature maps that are in different scales. The three feature maps are concatenated into one final feature map. The details of the YOLO-v3 feature map are discussed in Section 2.3. We add a transformation layer after the feature map to obtain the final detection results. The transformation layer applies sigmoid transformations on the t_x and t_y in the feature map to get box centre coordinates b_x and b_y. Exponential transformations are applied on t_w and t_h to obtain box width b_w and box height b_h, respectively. We then calculate the values of the loss functions defined in Section 3.3 and Section 3.4. Next, we minimise the loss value together with the distortion to generate our adversarial examples. During the optimisation, we introduce a hyper-parameter γ to control the strength of the attack.

We proposed our attack based on two distortion metrics, namely L_2-norm and L_0-norm. Herein, an L_p-norm is defined as $L_p =$

Algorithm 3: L_0 Daedalus attack

Input: x, Λ, γ, $binary_steps$, η, $max_iteration$, c_{max}, c_{min}
Initialisation:
$c \leftarrow 10$
$loss_{init} \leftarrow f(x)$
$\delta \leftarrow 0$
$x^* \leftarrow x + \delta$
for *n starts from 0 to* $binary_steps$ **do**
  for *i starts from 0 to* $max_iteration$ **do**
    select boxes in Λ to calculate loss $f(x^*)$
    $\lambda = \text{argmax}_{\lambda \in \delta} \nabla_\delta [\|\delta\| + c \cdot f(x^*)]$
    $\delta \leftarrow \delta - \eta \nabla_\lambda [\|\delta\| + c \cdot f(x^*)]$
    $x^* \leftarrow x^* + \delta$
  $x' \leftarrow$ the best x^* found.
  if $f(x^*) <= loss_{init} \cdot (1 - \gamma)$ **then**
    $c_{max} = min(c, c_{max})$
    $c \leftarrow 0.5 \cdot (c_{max} + c_{min})$
  else
    $c_{min} = max(c, c_{min})$
    $c \leftarrow 0.5 \cdot (c_{max} + c_{min})$
Output: Adversarial example x'.

$(\sum_i^n |\delta_i|^p)^{\frac{1}{p}}$, where δ_i is the perturbation on the i-th pixel. Therefore, L_2-norm attack limits the perturbations added on all the pixels, while L_0-norm attack limits the total number of pixels to be altered. We develop these two versions of attack to study the effectiveness of all-pixel perturbation and selective-pixel perturbation while generating Daedalus adversarial examples.

For the L_2 Daedalus attack, we can directly minimise Equation 4. However, in L_0 Daedalus attack, since L_0-norm is a non-differentiable metric, we alternatively select the pixels that have the highest adversarial gradients to perturb in each iteration, and then we clip other gradients to be 0. The detailed algorithm of L_2 Daedalus attack is presented in Algorithm 2, and the L_0 Daedalus attack is presented in Algorithm 3. Notice that Daedalus requires only the box parameters to launch the attack. This means that our attack can be applied to detection models that output feature maps which contain the geometry information of bounding boxes, even if they use different backbone networks.

4 EVALUATION

We carried out extensive experiments to evaluate the performance of our attack. We run our experiments on a machine with four RTX 2080ti GPUs and 128G of memory. The code of Daedalus attack is available on Github[2].

4.1 Experiment settings

In our evaluations, we pick YOLO-v3 as the model to attack. We first select the best loss function, and then we quantify the performance of our attack based on YOLO-v3 with NMS. We use YOLO-v3 as the substitute to generate 416×416 adversarial perturbations in our experiments. During the binary search to find the best constant c in our optimisation algorithm, we set the maximum number of search steps to 5.

We make adversarial examples of images in the COCO 2017val dataset [31]. We generate those using both our L_0 and L_2 attacks. For each attack, first, to investigate the effectiveness of different confidence values (γ), we sweep γ from 0.1 to 0.9 and select 10 COCO examples to perturb under each γ. Second, we randomly select 100 COCO examples to perturb under both a low (0.3) and a high (0.7) value for γ. To unify the results in our evaluation, we include all 80 object categories of COCO dataset in the object category set Λ to be attacked. We also select the categories of 'person' and 'car' to attack. The results are shown in the appendix.

To prove that our approach can be applied on other detectors, we also demonstrate the results of attack to RetinaNet-ResNet-50. RetinaNet-ResNet-50 uses ResNet-50 [21] as the backbone. The backbone differs from that of YOLO-v3. The output feature map is different as well. In this way, we demonstrate that our Daedalus adversarial examples can attack detectors with different backbone structures. Moreover, we study our attack in black-box scenarios by using an *ensemble of substitutes* to craft robust adversarial examples. The visualisations of the detection results are presented in the appendix.

4.2 Loss function selection

We propose three adversarial loss functions to craft adversarial examples (Equations 6, 7, and 8). We compare these three and select the best one in this section. We select 10 random images from COCO dataset. With each function, we generate adversarial examples of these 10 images. We record the runtime per example and the volatile GPU utilisation of each loss functions. We measure the performance of each function by looking at the success rate of finding adversarial examples at different confidences. Since our L_0 attack essentially has the same optimisation process as L_2 does, we only run L_2 attack for the comparison purpose. The results are reported in Table 2.

The recorded runtime is averaged over the runtime of generating all examples under the same loss function. The GPU utilisation is also averaged over all examples. We use only one GPU to generate an adversarial example in this section. It can be observed that f_3 (i.e. Equation 8) has the best success rate of finding adversarial examples under different γ values. Moreover, f_3 has the lowest runtime as well as the lowest GPU utilisation compared to f_1 and f_2.

Based on the comparisons, we select f_3, which is simple yet effective, as the loss function for Daedalus attack. We continue evaluating Daedalus attack in the following sections with f_3 (Equation 8).

4.3 Quantitative performance

In this section, we evaluate our attack from two perspectives. First, we evaluate how Daedalus attack performs under different NMS thresholds. Second, we investigate how the attack confidence level (γ) affects detection results. We assess the performance of our adversarial example on a YOLO-v3 detector. Herein, we quantify the attack performance based on three metrics: False Positive (*FP*) rate, mean Average Precision (*mAP*), and distortion of the example.

First, to investigate the effect of NMS threshold on our attack, we obtain the detection results of each adversarial/benign example under a series of NMS thresholds, which range between 0.5 and 0.95. Second, to assess the effect of confidence value (γ) on attack performance, we craft adversarial examples for confidences from 0.1 to 0.9.

False positive rate. To quantify the performance of our attack, we use the False Positive (*FP*) rate as an indicator. *FP* rate is defined as

[2]Anonymised for double-blind review.

Table 2: Loss function comparison

Loss Function	*Metric*	*Value of* γ								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
f_1	*Runtime*	2375.62 s								
	GPU-Util	79%								
	Success rate	**100%**	**100%**	**100%**	90%	40%	40%	20%	0%	0%
f_2	*Runtime*	1548.57 s								
	GPU-Util	77%								
	Success rate	**100%**	**100%**	**100%**	**100%**	**100%**	90%	40%	20%	0%
f_3	*Runtime*	1337.06 s								
	GPU-Util	73%								
	Success rate	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	70%	40%	40%

Figure 3: The false positive rate at IoU threshold of 0.5 with respect to NMS threshold and confidence of attack. The NMS threshold ranges from 0.5 **to** 0.95**. The confidence ranges from** 0.1 **to** 0.9 **in** L_2 **attack,** 0.1 **to** 0.8 **in** L_0 **attack.**

Figure 4: The false positive rate at IoU threshold of 0.75 with respect to NMS threshold and confidence of attack. The NMS threshold ranges from 0.5 **to** 0.95**. The confidence ranges from** 0.1 **to** 0.9 **in** L_2 **attack,** 0.1 **to** 0.8 **in** L_0 **attack.**

the ratio of false positive detection boxes with respect to the total number of detection boxes. *FP* rate can be defined as below:

$$FP\ rate = \frac{N_\phi + 1}{N + 1}, \tag{10}$$

wherein N_ϕ is the number of false positive detection boxes, and N is the total number of bounding boxes found by the model. We add 1 to both the numerator and the denominator to avoid ill definition when there is no detected box. *FP* rate measures the proportion of redundant boxes that are not filtered by NMS. *FP* is maximally 1 and that is when there is no correct detection result. A higher value of *FP* indicates a higher success rate for the attack.

We vary IoU threshold in NMS from 0.5 to 0.95 with steps of 0.05 to obtain detection outputs under different thresholds. We then obtain the detection results of Daedalus examples made under confidences from 0.1 to 0.9 (0.1 to 0.8 for L_0 since it cannot find examples with 0.9 confidence). This is because L_0 cannot further discard perturbations from L_2 examples under a high γ. We have reported the *FP* rates in the results based on IoU thresholds of 0.5 and 0.75 in Fig. 3 and Fig. 4, respectively. Each *FP* rate is averaged over 10 Daedalus examples. Next, we measure the average *FP* rate of 100 examples crafted under a confidence value of 0.3/0.7. We plot the trends of the average *FP* rate at IoU thresholds of 0.5 and 0.75 with respect to NMS threshold in Fig. 5 and Fig. 6, respectively.

It can be observed that both L_0 and L_2 attacks achieve *FP* rates of at least 90% under different NMS thresholds, even with the lowest attack confidence of 0.1. Increasing the NMS threshold slightly

Figure 5: L_0 **attack false positive rate at IoU threshold of 0.5 with respect to NMS threshold. Each** *FP* **rate is averaged from the detection results of** 100 **Daedalus examples.**

reduces the *FP* rate of low-confidence attacks. However, changing the NMS threshold barely has an impact on the performance of high-confidence attacks. *FP* rate increases with the increase of attack confidence i.e. γ. It can get as high as 99.9% for a high-confidence Daedalus attack, even with a stricter NMS threshold.

Mean average precision. With this criterion, we want to investigate how our adversarial example changes *mAP* of the detection results. *mAP* is commonly used to evaluate the performance of OD model. It averages the Average Precisions (*AP*s) over all detected object categories. This metric is defined as follows:

Figure 6: L_0 attack false positive rate at IoU threshold of 0.75 with respect to NMS threshold. Each *FP* rate is averaged from the detection results of 100 Daedalus examples.

Figure 7: $mAP^{IoU=.50}$ of Daedalus example detection results with respect to NMS threshold and attack confidence. The threshold ranges from 0.5 to 0.95.

Figure 8: $mAP^{IoU=.75}$ of Daedalus example detection results with respect to NMS threshold and attack confidence. The threshold ranges from 0.5 to 0.95.

$$mAP = \frac{1}{M}\sum_{i=1}^{M} AP_i, \tag{11}$$

in which M is the number of object categories. We adopted the all-point interpolation AP from Pascal VOC challenge [11] to calculate mAP.

Figure 9: $mAP^{IoU=.50}$ of L_0 and L_2 Daedalus attacks with respect to NMS threshold. The threshold ranges from 0.5 to 0.95. Each mAP is averaged from the detection results of 100 Daedalus examples.

Figure 10: $mAP^{IoU=.75}$ of L_0 and L_2 Daedalus attacks with respect to NMS threshold. The threshold ranges from 0.5 to 0.95. Each mAP is averaged from the detection results of 100 Daedalus examples.

We use the recommended COCO evaluation metrics $mAP^{IoU=.50}$ and $mAP^{IoU=.75}$, which are the mAP values calculated at IoU thresholds of 0.5 and 0.75, respectively, to measure the detection precisions on benign examples and Daedalus examples. Fig. 7 and Fig. 8 show the mAP values of the detection results when we change the NMS threshold and γ. We also plotted the average $mAP^{IoU=.50}$ and $mAP^{IoU=.75}$ for 100 examples made under confidence values of 0.3 and 0.7 in Fig. 9 and Fig. 10, respectively.

Based on the results, both L_0 and L_2 attacks decrease mAP of the detector from 45% $\sim$ 59% to a value between 0% to 17.8%. mAP drops to 0% when the confidence of the attack surpasses 0.2.

Distortion of the examples. We evaluate the distortion of the Daedalus adversarial examples crafted based on L_2-norm and L_0-norm. For a fair comparison, the distortions are measured using L_2-norm. We record the maximum, the minimum and the average distortion of 10 adversarial examples for each attack confidence value. The results are summarised in Table 3. It can be seen that the distortion in adversarial examples has a positive correlation with the confidence of attacks. The distortion increases slowly before the confidence reaches 0.6. Once the confidence goes beyond 0.6,

there is a rapid increase in the distortion amount. Additionally, compared to L_2 attack, L_0 can not find any adversarial examples at a confidence of 0.9.

Based on the evaluation, both L_0 and L_2 attacks require moderate distortions to make low-confidence examples. Compared to L_0, L_2 attack introduces less distortion when the confidence is between 0.2 and 0.7, and it is slightly better in terms of finding adversarial examples with a confidence of 0.9. However, L_0 attack can reduce the distortion when the confidence goes above 0.8. It is known that FCN-based object detectors usually segment the input image by a mesh grid. Each grid cell has its own responsibility for detecting objects. Low-level perturbation based on L_0-norm may leave some of the grid cells unperturbed. Therefore, these cells can still output normal detection results, which can be filtered by NMS. As a result, L_0 attack tries to perturb most of the pixels to generate a well-qualified adversarial example, which implies having high distortions in the example. High-level perturbation is imposed on each pixel when the confidence goes above 0.8. However, our L_0 attack can discard the perturbation on some pixels to reduce the overall distortion.

5 ANALYSIS OF THE PROPOSED ATTACK

In this section, we first investigate whether our attack can be extended to a variation of NMS algorithm, namely soft-NMS [3] or not. Second, we demonstrate the universality of Daedalus attack by applying it to attack RetinaNet-Resnet-50. Third, we propose using an *ensemble of substitutes* to create robust Daedalus examples that can attack multiple object detectors. The proposed ensemble attack makes Daedalus examples transferable so that they can be used in black-box scenarios. Finally, we discuss the problem of defending against Daedalus attack.

5.1 Breaking Soft-NMS

In addition to breaking NMS, we test our attack on soft-NMS, which iteratively decreases the confidence score of overlapping boxes instead of discarding them [3]. Soft-NMS has linear and Gaussian versions. We apply our attack on both to investigate whether soft-NMS can handle it or not. We randomly select one of L_0 or L_2 Daedalus examples to attack a soft-NMS-equipped YOLO-v3 detector. The detection results for L_0 adversarial examples are displayed in Fig. 11a. For L_2 adversarial examples, the detection results are similarly illustrated in Fig. 11b. From the results one can find that both L_0 and L_2 can break linear soft-NMS as well as Gaussian soft-NMS.

We also measure the FP rate of our attack for soft-NMS. The results are shown in Fig. 12. Our L_0 adversarial examples which are made with different confidence values result in a FP rate of 100%. Our L_2 attack also leads to 100% FP rate when the attack confidence is above 0.2.

5.2 Attacking RetinaNet

Daedalus attack can be applied to different FCN-based detection algorithms. To demonstrate this capability, we launch it against another popular OD algorithm, RetinaNet [30]. By default, we select ResNet-50 as the backbone network of RetinaNet. RetinaNet extracts the feature maps of detection boxes and object classes by using a feature pyramid network (FPN) [29]. The features will then be processed by regression models to obtain the position and the dimension of the bounding boxes. The object classification results will be obtained from the feature maps by classification models.

We follow the same procedure for attacking RetinaNet as we did for YOLO-v3. We calculate the adversarial loss based on Equation 8, and generate adversarial examples by optimising Equation 4. We only test our L_2 attack since a successful L_2 attack guarantees a successful L_0 attack, according to the structure of the attacks. We generate ten 416×416 adversarial examples under a confidence of 0.3. We plot one of the detection results from RetinaNet in Fig. 13. It can be seen that the example triggers malfunctioning of NMS in RetinaNet, and it leads to extremely dense detection results.

Figure 11: (a) L_0 Daedalus attack against soft-NMS. The top left image is an L_0 Daedalus example crafted under a confidence of 0.3. The top right one shows the detection results from NMS, as a reference. The left bottom one shows the results of linear soft-NMS, and the right bottom one shows the results of Gaussian soft-NMS. (b) L_2 Daedalus attack against soft-NMS. The top left image is an L_2 Daedalus example crafted under a confidence of 0.3. The top right one shows the detection results from NMS, as a reference. The left bottom one shows the results of linear soft-NMS, and the right bottom one shows the results of Gaussian soft-NMS.

Figure 12: The FP rate of Daedalus attack against YOLO-v3 with soft-NMS. We evaluated the FP rate at IoU thresholds of 0.5 and 0.75.

Table 3: Distortions under different confidences

Attack	*Distortion*	*Value of γ*								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
L_2 *Daedalus*	*Maximum*	157.49	259.25	113.24	324.53	598.45	4138.47	5915.90	25859.67	19553.05
	Average	130.59	218.86	96.51	166.27	344.75	1670.35	1835.04	9953.59	14390.76
	Minimum	94.30	175.07	75.24	95.30	185.25	351.55	366.49	210.24	11894.38
L_0 *Daedalus*	*Maximum*	162.74	513.54	555.11	1200.48	1001.86	42144.96	3891.80	3364.29	N/A
	Average	87.81	346.71	440.21	651.77	736.40	12765.82	1992.30	1501.58	N/A
	Minimum	53.26	165.50	339.44	402.48	436.15	3713.67	729.38	848.36	N/A

Figure 13: The detection results of a Daedalus adversarial example made by our L_2 attack towards RetinaNet-ResNet-50.

We also measure the *FP* rate and the *mAP* value of RetinaNet for the 10 Daedalus examples. The *FP* rate with respect to NMS threshold is plotted in Fig. 14, and *mAP* with respect to NMS threshold is plotted in Fig. 15. More detection results of Daedalus examples from RetinaNet are presented in the appendix.

Fig. 14 shows that Daedalus attack can increase the *FP* rate of RetinaNet to 99%. According to Fig. 15, our attack decreases $mAP^{IoU=.50}$ of RetinaNet from around 0.27 to below 0.05 and also $mAP^{IoU=.75}$ to 0. Therefore, our attack is effective against RetinaNet.

5.3 Making universal adversarial examples

Based on the census in Section 1, we take three popular models, namely YOLO-v3, SSD, and RetinaNet, to verify the idea of ensemble attack. We do not introduce more models as substitutes, simply due to the limited GPU memory. According to Table 1, a universal Daedalus example for YOLO-v3, SSD, and RetinaNet can already break 54.9% of the 1,696 repositories. We investigate whether a universal Daedalus example can be found to break all the models. In our experiments, YOLO-v3 employs Darknet-53 as its backbone network. RetinaNet and SSD have different backbone networks and use ResNet-50 and VGG-16, respectively. Since the three models

Figure 14: $FP\ rate^{IoU=.50}$ and $FP\ rate^{IoU=.75}$ of our L_2 Daedalus attack against RetinaNet, with respect to NMS threshold. The NMS threshold ranges from 0.5 to 0.95.

Figure 15: $mAP^{IoU=.50}$ and $mAP^{IoU=.75}$ of our L_2 Daedalus attacks against RetinaNet, with respect to NMS threshold. The threshold ranges from 0.5 to 0.95.

are backed by different backbones, they generate divergent feature maps.

The detection results of the adversarial example are displayed in Fig.16. We can observe that the adversarial example can trigger NMS malfunctioning for all the models. The detection results for more adversarial examples crafted using *ensemble of substitutes* can be found in the appendix of this paper. Furthermore, when there is a large number of OD models in the ensemble, an attacker may

Figure 16: The detection results of a Daedalus adversarial example made by ensemble-of-substitutes approach. We detect the adversarial example using YOLO-v3, RetinaNet, and SSD. The first row contains a benign example and its detection results. The second row contains an adversarial example and the detection results of the adversarial example.

adopt methods such as [51] to find proper coefficient for each term in the loss function to enhance attack performance.

5.4 Adaptive defences against Daedalus attack

Daedalus minimises detection boxes to trigger a NMS malfunction. Henceforth, a straight forward defence might be limiting the minimal allowed detection box size during NMS (i.e. *NMS defence*). To investigate this possible defence, we first study the outputs of YOLO-v3 given 100 benign examples and corresponding 100 Daedalus examples. We visualise the distribution of the detection box dimension in Figure 17. According to Figure 17, we select an empirical threshold of $10^{3.62}$ to filter adversarial boxes from detected boxes during NMS. Given this defence, we again evaluate the FP rate and the mAP of YOLO-v3 on 10 Daedalus examples. The evaluation results are plotted in Figure 18. According to the results, *NMS defence* is not an effective defence.

A second adaptive defence might be MagNet [40]. We train an denoise autoencoder to reform Daedalus examples back to benign ones, and measure the reconstruction error to detect strong perturbations. Therefore, the evaluation of the defence is two-fold: 1) Whether the defence can increase mAP and reduce FP rate via reforming an adversarial example. 2) Whether the reconstruction error can be detected when an adversarial example cannot be turned into a benign one. To answer these two questions, we evaluate the FP rate, the mAP and the Mean Squared Error (MSE) between a Daedalus example and a reformed example (i.e., the reconstruction error) of a defended YOLO-v3 model. The results are plotted in Figure 19. According to the evaluation results, first, reforming the adversarial examples can barely increase the performance of the detection model. Second, The reconstruction error increases only after the confidence of the attack reaches 0.5, which means it is difficult to detect attacks with confidence levels below 0.5. Interestingly, we observe that the detected objects in an image reformed by a denoise autoencoder cannot retain proper positions and categories. This makes the defence inviable.

Figure 17: Distributions of bounding box dimension from detection results of YOLO-v3. The distributions are based on detection results of 100 COCO images and corresponding 100 adversarial examples, respectively. In total, there are 812 boxes in the benign results and 286,469 adversarial boxes in the adversarial results. As a defence, we discarded boxes whose dimension is below $10^{3.62}$ during NMS.

Figure 18: The FP-rate and the mAP of YOLO-v3 with *NMS defence*. The defence barely changes the FP-rate and the mAP.

5.5 Discussion of Daedalus attack

Adversarial examples were initially made for machine learning classifiers. Hence, current defense methods mainly focus on the classification task. Defence mechanisms can be either reactive (e.g. [17, 34, 38, 40, 41, 55, 59]), or proactive (e.g. [7, 8, 19, 37, 39, 45, 53, 56, 57]). In the majority of reactive defense systems, a second model is used to detect examples with adversarial perturbation. In the category of proactive defence, the model is trained to be as robust to adversarial examples as possible. Currently, there exists no

Figure 19: The FP-rate and the mAP of YOLO-v3 with *MagNet defence*. Reforming Daedalus examples by an autoencoder can slightly increase the performance of YOLO-v3. However, the FP-rate@.50 still stays above 0.6 while the FP-rate@.75 stays above 0.7. The mAP@.50 remains below 0.45 while the mAP@.75 remains below 0.3. Furthermore, the model performance drops when the confidence of the attack goes above 0.7.

defense method for the new type of attack introduced by Daedalus examples.

Daedalus attack poses an intimidating threat to OD tasks. Current machine learning applications, especially those that need OD, are developed by reusing existing models [26]. Therefore, an adversary can adopt an *ensemble of substitutes* to launch the attack under black-box setting. He/she only needs to include popular backbone networks in the *ensemble of substitutes*. Alternatively, an adversary can guess the type of the backbone network being used and then launch the attack. Moreover, it is possible to combine Daedalus with attacks such as Expectation over Transformation [2] to produce robust physical adversarial examples. This will further augment the viability of the attack vector.

As revealed by our attack, NMS is no longer a secure solution for OD. Unlike the attacks against classifiers, Daedalus attack compresses the box dimension values in the feature map to undermine the NMS functionality. Currently, there is no intact defence for such attack. However, defences for DNN classifiers cannot be directly applied to OD algorithms to tackle Daedalus attack, which is supported by our evaluation in Section 5.4. This dilemma makes defending against Daedalus an urgent task. We may need to propose new algorithms to filter redundant bounding boxes for OD tasks if no effective defence is found against Daedalus attack.

6 RELATED WORK

Methods for crafting adversarial examples against machine-learning-based classifiers have been extensively studied. The algorithms for generating such examples can be divided into gradient-based and forward derivative-based. The former type of attacks finds adversarial examples by minimising the cost of an adversarial objective set by the attacker, based on gradient descent. For example, L-BFGS was adopted to optimise the adversarial objective functions to generate adversarial examples [5, 56]. Fast gradient sign method was proposed to rapidly find first-order Taylor polynomial approximations of adversarial examples based on gradient descent or gradient sign descent [16]. Basic iterative method relies on multiple steps of gradient descent to generate adversarial examples [28]. Deepfool computes adversarial gradients based on the local linearity of neural networks [42]. Forward derivative based attack perturbs salient features based on the Jacobian between the model inputs and outputs [44]. Based on these algorithms, there are evolved attacks that use different distortion metrics to make adversarial examples more imperceptible to human eyes [10, 25]. Furthermore, there are methods to craft adversarial examples of data that have discrete features (e.g. text) [9, 18, 27].

Beyond the above attacks, some methods can generate robust adversarial examples that can fool real-world classifiers and detectors. For example, dense adversarial generation (DAG) algorithm was proposed to create adversarial examples for OD and segmentation [58]. Lu et al. crafted robust adversarial examples that caused misclassification in object detector [35]. RP_2 was proposed to craft adversarial example of real-world road signs [12]. Later, RP_2 was extended to attack YOLO-v2 [54]. Recently, expectation over transformation (EoT) algorithm has been proposed to synthesise robust adversarial examples [2, 4]. However, current adversarial examples only focus on triggering misclassification in machine learning classifiers/object detectors. Our Daedalus attack creates adversarial examples that causes malfunctioning of NMS, which is different from all the previous attacks in nature.

7 CONCLUSION

In this paper, we propose a novel type of adversarial example which aims at disabling NMS functionality in object detection models. The proposed attack, named Daedalus, can control both the strength of the generated adversarial examples and the object class to be attacked. The attacked model will output extremely noisy detection results such that it is no longer functional. Our attack can reduce the mAP of detection to nearly 0%. The false positive rate for Daedalus examples can go up to 99.9%. Meanwhile, the required distortion to launch Daedalus attack is imperceptible. This attack aims at breaking NMS instead of causing misclassifications. Unlike misclassification-targeting adversarial examples, it is difficult to defend against such attack since the attacked feature map is much more complicated than the classification logits. Transferability of Daedalus attack is affected by the selected substitutes. We rely on minimising the expectation of box dimensions over a set of feature maps to make robust examples, which can make NMS malfunction in multiple detection models. There are some remaining problems we aim to address in future. For example, considering the complexity of OD feature maps, it is difficult to make universal adversarial examples that can launch zero-knowledge attacks. Daedalus attack reveals a vulnerability lies in object detection algorithms, which can have fatal consequences in the real world. Nevertheless, defending method against Daedalus attack is still a missing piece. We will also try to address this problem in our future work.

REFERENCES

[1] Naveed Akhtar and Ajmal Mian. 2018. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *IEEE Access* 6 (2018), 14410–14430.

[2] Anish Athalye and Ilya Sutskever. 2017. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397* (2017).

[3] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. 2017. Soft-nms-improving object detection with one line of code. In *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 5562–5570.

[4] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial Patch. *arXiv preprint arXiv:1712.09665* (2017).

[5] Nicholas Carlini and David Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. https://doi.org/10.1109/sp.2017.49

[6] Zhe Chen, Shaoli Huang, and Dacheng Tao. 2018. Context Refinement for Object Detection. In *The European Conference on Computer Vision (ECCV)*.

[7] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. 2017. Parseval Networks: Improving Robustness to Adversarial Examples. In *International Conference on Machine Learning*. 854–863.

[8] Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. 2018. Stochastic activation pruning for robust adversarial defense. *arXiv preprint arXiv:1803.01442* (2018).

[9] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of ACL*.

[10] Logan Engstrom, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. 2017. A rotation and a translation suffice: Fooling cnns with simple transformations. *arXiv preprint arXiv:1712.02779* (2017).

[11] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision* 88, 2 (2010), 303–338.

[12] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. 2017. Robust physical-world attacks on machine learning models. *arXiv preprint arXiv:1707.08945* 2, 3 (2017), 4.

[13] Álvaro García-Martín and José María Martínez. 2015. People detection in surveillance: classification and evaluation. *IET Computer Vision* 9, 5 (2015), 779–788.

[14] Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 1440–1448.

[15] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 580–587.

[16] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and Harnessing Adversarial Examples. *Computer Science* (2014).

[17] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. 2017. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280* (2017).

[18] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. 2017. Adversarial examples for malware detection. In *European Symposium on Research in Computer Security*. Springer, 62–79.

[19] Shixiang Gu and Luca Rigazio. 2015. Towards Deep Neural Network Architectures Robust to Adversarial Examples. *Computer Science* (2015).

[20] Raghav Gurbaxani and Shivank Mishra. 2018. Traits & Transferability of Adversarial Examples against Instance Segmentation & Object Detection. *arXiv preprint arXiv:1808.01452* (2018).

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

[22] Paul Henderson and Vittorio Ferrari. 2016. End-to-end training of object class detectors for mean average precision. In *Asian Conference on Computer Vision*. Springer, 198–213.

[23] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. 2016. A convnet for non-maximum suppression. In *German Conference on Pattern Recognition*. Springer, 192–204.

[24] Jan Hendrik Hosang, Rodrigo Benenson, and Bernt Schiele. 2017. Learning non-maximum suppression.. In *CVPR*. 6469–6477.

[25] Hossein Hosseini and Radha Poovendran. 2018. Semantic Adversarial Examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 1614–1619.

[26] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. 2018. Model-Reuse Attacks on Deep Learning Systems. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 349–363.

[27] Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2021–2031.

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

[29] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *CVPR*, Vol. 1. 4.

[30] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence* (2018).

[31] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[32] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. 2018. Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165* (2018).

[33] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*. Springer, 21–37.

[34] Jiajun Lu, Theerasit Issaranon, and David Forsyth. 2017. Safetynet: Detecting and rejecting adversarial examples robustly. *arXiv preprint arXiv:1704.00103* (2017).

[35] Jiajun Lu, Hussein Sibai, and Evan Fabry. 2017. Adversarial Examples that Fool Detectors. *arXiv preprint arXiv:1712.02494* (2017).

[36] Keyu Lu, Xiangjing An, Jian Li, and Hangen He. 2017. Efficient deep network for vision-based object detection in robotic applications. *Neurocomputing* 245 (2017), 31–45.

[37] Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang. 2015. A unified gradient regularization family for adversarial examples. In *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 301–309.

[38] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Michael E Houle, Grant Schoenebeck, Dawn Song, and James Bailey. 2018. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613* (2018).

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

[40] Dongyu Meng and Hao Chen. 2017. MagNet: a Two-Pronged Defense against Adversarial Examples. *arXiv preprint arXiv:1705.09064* (2017).

[41] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On detecting adversarial perturbations. In *The 5th IEEE International Conference on Learning Representation*.

[42] Seyed Mohsen Moosavidezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2574–2582.

[43] Alexander Neubeck and Luc Van Gool. 2006. Efficient non-maximum suppression. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, Vol. 3. IEEE, 850–855.

[44] Nicolas Papernot, Patrick Mcdaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. 2016. The Limitations of Deep Learning in Adversarial Settings. (2016), 372–387.

[45] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 582–597.

[46] Sergey Prokudin, Daniel Kappler, Sebastian Nowozin, and Peter Gehler. 2017. Learning To Filter Object Detections. In *German Conference on Pattern Recognition*. Springer, 52–62.

[47] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 779–788.

[48] Joseph Redmon and Ali Farhadi. 2016. YOLO9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242* (2016).

[49] Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[50] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision & Pattern Recognition*.

[51] Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*. 527–538.

[52] Mohammad Javad Shafiee, Brendan Chywl, Francis Li, and Alexander Wong. 2017. Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video. *arXiv preprint arXiv:1709.05943* (2017).

[53] Uri Shaham, Yutaro Yamada, and Sahand Negahban. 2018. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing* (2018).

[54] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramer, Atul Prakash, and Tadayoshi Kohno. 2018. Physical Adversarial Examples for Object Detectors. In *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*.

[55] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. 2017. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766* (2017).

[56] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *Computer Science* (2013).

[57] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. 2017. Ensemble Adversarial Training: Attacks and Defenses. *arXiv preprint arXiv:1705.07204* (2017).
[58] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. 2017. Adversarial Examples for Semantic Segmentation and Object Detection. In *International Conference on Computer Vision*. IEEE.
[59] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155* (2017).
[60] Zhi Yang, Wei Yu, Pengwei Liang, Hanqi Guo, Likun Xia, Feng Zhang, Yong Ma, and Jiayi Ma. 2018. Deep transfer learning for military object recognition under small training set condition. *Neural Computing and Applications* (2018), 1–10.
[61] Jianming Zhang, Manting Huang, Xiaokang Jin, and Xudong Li. 2017. A Real-Time Chinese Traffic Sign Detection Algorithm Based on Modified YOLOv2. *Algorithms* 10, 4 (2017), 127.

APPENDIX A: DEMONSTRATION

We include more results of Daedalus attack in this appendix.

Figure 20: Adversarial examples made by our L_0 attack. The first row contains the original images. The third row contains our low-confidence ($\gamma = 0.3$) adversarial examples. The fifth row contains our high-confidence ($\gamma = 0.7$) examples. The detection results from YOLO-v3 are in the rows below them. The confidence controls the density of the redundant detection boxes in the detection results.

Figure 21: Adversarial examples made by our L_2 attack. The first row contains the original images. The third row contains our low-confidence ($\gamma = 0.3$) adversarial examples. The fifth row contains our high-confidence ($\gamma = 0.7$) examples. The detection results from YOLO-v3 are in the rows below them. The confidence controls the density of the redundant detection boxes in the detection results.

Figure 22: The detection results of Daedalus adversarial examples made by our L_2 attack towards RetinaNet-ResNet-50. The adversarial examples are crafted based on a confidence of 0.3.

Figure 23: The detection results of Daedalus adversarial examples made by our ensemble L_2 attack towards YOLO-v3, RetinaNet, and SSD. The adversarial examples are crafted based on a confidence of 0.3. The first row displays the benign examples. The second, the third, and the fourth rows are the detection results of the benign examples from YOLO-v3, RetinaNet, and SSD, respectively. The Daedalus examples are displayed in the fourth row. The detection results of the adversarial examples from YOLO-v3, RetinaNet, and SSD are in the following rows.

Figure 24: Attacking the object category which has the most number of the detection boxes before NMS. The attacked model is YOLO-v3.

Figure 25: Demonstration of attacking object in a specified category for YOLO-v3. We select 'person' to attack.

Figure 26: Demonstration of attacking object in a specified category for YOLO-v3. We select 'car' to attack.