Automated Software Engineering (2025) 32:3
https://doi.org/10.1007/s10515-024-00470-9

Evoattack: suppressive adversarial attacks against object detection models using evolutionary search

Kenneth H. Chan[1] · Betty H.C. Cheng[1]

Received: 31 March 2024 / Accepted: 22 September 2024

Abstract
State-of-the-art deep neural networks are increasingly used in image classification, recognition, and detection tasks for a range of real-world applications. Moreover, many of these applications are safety-critical, where the failure of the system may cause serious harm, injuries, or even deaths. Adversarial examples are expected inputs that are maliciously modified, but difficult to detect, such that the machine learning models fail to classify them correctly. While a number of evolutionary search-based approaches have been developed to generate adversarial examples against image classification problems, evolutionary search-based attacks against *object detection* algorithms remain largely unexplored. This paper describes EvoAttack that demonstrates how evolutionary search-based techniques can be used as a black-box, model- and data-agnostic approach to attack state-of-the-art object detection algorithms (e.g., RetinaNet, Faster R-CNN, and YoloV5). A proof-of-concept implementation is provided to demonstrate how evolutionary search can generate adversarial examples that existing models fail to correctly process, which can be used to assess model robustness against such attacks. In contrast to other adversarial example approaches that cause misclassification or incorrect labeling of objects, EvoAttack applies minor perturbations to generate adversarial examples that *suppress* the ability of object detection algorithms to detect objects. We applied EvoAttack to popular benchmark datasets for autonomous terrestrial and aerial vehicles.

Keywords Black-box Adversarial Attacks · Cybersecurity · Evolutionary Computing · Object Detection · Cybersecure AI · Deep Neural Networks

✉ Kenneth H. Chan
chanken1@msu.edu

Betty H.C. Cheng
chengb@msu.edu

1 Department of Computer Science and Engineering, Michigan State University, 428 S. Shaw Lane, East Lansing, MI 48824, USA

1 Introduction

The ability of machine learning algorithms to learn complex and high-dimensional features directly from data has promoted their integration into real-world safety critical systems (Knight 2002) (i.e., those that have significant consequences should they fail, leading to injury, death, and/or financial loss). Many popular machine learning techniques, such as Deep Neural Networks (DNNs), are susceptible to carefully-crafted malicious inputs, known as *adversarial examples* (Goodfellow et al. 2015; Szegedy et al. 2014). A fundamental objective and definition of adversarial examples is to minimize the human-perceptible perturbations (i.e., both the number of changed pixels and degree of pixel change), thus enabling them to avoid human detection while causing the model to misbehave (Metzen et al. 2017; Carlini and Wagner 2017; Ren et al. 2020). To prevent serious harm or injuries, DNNs deployed in safety-critical systems (e.g., malicious file detection (Nandita and Chandra 2021; Rudd et al. 2018), fraud detection (Marie-Sainte et al. 2020; Wang and Xu 2018), and autonomous vehicles (Kocić et al. 2019; Wu et al. 2017)) must be robust against adversarial attacks. Therefore, an existing challenge is to assess a machine learning model's robustness to better understand the limitations and weaknesses of the model in the face of adversarial attacks. This paper describes EvoAttack, a black-box evolutionary search-based technique, to assess the robustness of object detection algorithms against adversarial examples.

Over the past decade, several research efforts have addressed adversarial examples for image classification techniques (Eykholt et al. 2018; Goodfellow et al. 2015; Moosavi-Dezfooli et al. 2016; Szegedy et al. 2014). Adversarial examples are expected input data (often part of the original dataset) with a small amount of human-imperceptible perturbations introduced to cause model failure (e.g., misclassification of class labels) (Goodfellow et al. 2015; Szegedy et al. 2014). Adversarial example research has largely focused on techniques that challenge the robustness of image classification techniques (i.e., given an image of an object, correctly label the object). However, limited research has been done on the impact of adversarial attacks on object detection algorithms (i.e., given an image with up to n number of objects, correctly identify the object(s) by drawing a bounding box around them and labeling them accordingly), a vital component of many safety critical systems (e.g., autonomous driving, autonomous drones, etc.) (Wei et al. 2019; Xie et al. 2017; Yuan et al. 2019; Wang et al. 2020). Compared to image classification, attacking object detection techniques is significantly more difficult as the images are larger in size, contain more dimensions, and potentially contain multiple objects to be identified and classified. Existing techniques for generating adversarial examples against object detection algorithms (Wei et al. 2019; Xie et al. 2017) largely assume a white-box model, where sensitive or critical model parameters are known to the adversary. While existing approaches can be used as weak black-box attacks (i.e., transfer from a white-box attack to a black-box model with similar architectures), such approaches often involve additional training overhead to produce a surrogate model to attack. Black-box approaches are particularly noteworthy since they typically require little to no domain knowledge for the attacker on the domain space, data, or model information. As such, a true black-box, model- and data-agnostic approach (i.e., does not depend on model or data specific information) for object detection algorithms is still needed to assess and potentially improve their robustness.

This paper describes EvoAttack, a black-box evolutionary search-based testing technique that generates adversarial examples to compromise object detection algorithms, which can be used to assess model robustness against adversarial attacks. EvoAttack does not require access to model information (e.g., hidden model parameters, model architecture, etc.), does not require additional training of surrogate models, and is model- and data-agnostic. EvoAttack provides three key capabilities. First, EvoAttack uses evolutionary search to generate *suppressive adversarial attacks*, a class of attacks that hinders the detection of objects for object detection algorithms while minimizing perturbations. Traditional attacks for adversarial examples generate perturbations that cause the incorrect labeling of objects in image classification or object detection algorithms (Xie et al. 2017). As the objective of an object detection algorithm is to correctly draw bounding boxes around objects in an image and classify them correctly, we define the term *suppressive adversarial attacks* to describe adversarial attacks that do not seek to deceive the model to produce an incorrect label for a given object, but, instead, cause the model to fail to detect the objects altogether. With respect to object detection algorithms used in safety-critical systems, such as obstacle avoidance, failing to identify an object may be considered a more adverse system behavior when compared to misclassifying an object. For example, an object detection algorithm for an autonomous vehicle may still brake appropriately even if a pedestrian is misclassified as a cyclist. However, the same system may result in a collision if the object detection algorithm fails to detect the pedestrian. Second, as an optimization, we *localize perturbations* to regions of interest by leveraging the output of the object detection model from the previous generation of evolution to guide the mutation process and the structure of the fitness function. Third, we propose a dynamic mutation scheme that dynamically adjusts the mutation rate to further reduce perturbations while promoting convergence.

Two key strategies are used to enable our approach to generate suppressive adversarial examples for object detection. To generate adversarial examples, we use a generational Genetic Algorithm (GA) (Sivanandam and Deepa 2008), where individuals in a population evolve towards a global optimum (i.e., a perturbed image that adversely impacts object detection). Compared to image classification problems, object detection images contain multiple classification sub-problems. As such, we developed a fitness function that simultaneously accounts for all objects detected by the model during the inference stage. Specifically, by using the output of the model from the previous generation, our approach is able to localize the perturbation region by ignoring pixels that do not directly affect the output of the model. Additionally, our fitness function dynamically adapts based on the number of bounding boxes and confidence scores from the previous generation's detection results. During mutation, we apply a fine-grained approach for generating perturbations by focusing only on pixels in areas of interest. We introduce a dynamic mutation scheme, where we promote minor perturbations for each object in the image, while encouraging misdetection from the model. We have extended our preliminary work (Chan and Cheng 2022) in three key ways. First, we have introduced the term "suppressive adversarial attack" and formalized its definition. Second, we replace our *adaptive mutation scheme* from our preliminary work that did not consider object size (i.e., number of pixels) for the mutation process with a new *dynamic mutation scheme* that mutates a small percentage of pixels when the generation count is low. The percentage of modified pixels is dynamically increased as the number of generations increases, thereby promoting convergence. Our new dynamic mutation scheme enables EvoAttack to introduce minimal perturbations required to prevent the object detection model from correctly identifying objects, while maintaining similar computation time for images of different sizes across various domain application space (e.g., terrestrial vehicles and aerial vehicles). In addition, our new dynamic mutation scheme minimizes the need to retune evolutionary parameters between models and datasets. Third, we have conducted a new demonstration study to further illustrate the model- and data-agnostic properties of EvoAttack, while also demonstrating the domain-agnostic property. Specifically, we applied EvoAttack to the popular YoloV5 object detection algorithm trained to detect objects in the VisDrone benchmark dataset (Cao et al. 2021). The VisDrone dataset contains images captured from a camera mounted on an aerial drone. Compared to previous datasets for terrestrial vehicles and common objects explored in our preliminary work, the VisDrone dataset contains many more objects with small dimensions in each image (other examples from this domain include satellite imagery). Perturbations have higher impact on the model's results when introduced to objects with small dimensions than on objects with large dimensions, as objects with small dimensions have fewer pixels (i.e., information) for the purposes of object detection and classification.

In our experiments, we empirically demonstrate that EvoAttack can produce adversarial examples that suppress object detection. We implemented the proposed approach and generated adversarial examples against a number of state-of-the-art object detection models, such as RetinaNet (Lin et al. 2017), Faster R-CNN (Ren et al. 2015), and the YOLOv5 model (Jocher et al. 2022). To illustrate the potential impact of adversarial examples against object detection models, we apply our technique to attack a set of images from several popular benchmark datasets, including safety-critical applications such as autonomous driving and autonomous drone flying. They include the Microsoft Common Object in Context (COCO) dataset (Lin et al. 2014), the Waymo Open Dataset (Sun et al. 2020) for autonomous vehicles, and the VisDrone (Cao et al. 2021) dataset for autonomous drones to show that our approach is data- and domain-agnostic. Results show that our algorithm can cause models to deviate from the expected output, while maintaining a low degree of visible perturbations (i.e., minimizing the number of pixels changed and the degree of change). This work shows that black-box evolutionary search-based adversarial examples can be generated against object detection tasks, which has not been previously achieved to the best of our knowledge. The remainder of this paper is organized as follows. Section 2 overviews background material and enabling technologies. Next, Sect. 3 describes the details of the proposed approach. Section 4 describes the validation work of our approach. Section 5 reviews related work for adversarial attacks against object detection algorithms. Section 6 discusses our threats to validity. Finally, Sect. 7 concludes the paper and discusses future directions.

2 Background

This section provides background information for the paper. First, we overview DNNs and their architectures. Next, we overview the foundations of adversarial examples, as well as describe different types of adversarial attacks based on their objectives and information used. Then we compare the image classification problem with the object detection problem. Finally, we overview the basics of evolutionary algorithms.

2.1 Deep neural networks

DNNs are artificial neural networks that are inspired by the human brain (Hardesty 2017). DNNs are becoming increasingly popular and used in safety-critical systems due to their ability to extract complex and high-dimensional features directly from data. DNNs have demonstrated promising results on traditionally difficult problems, such as object detection, image classification, speech recognition, etc. For example, a developer cannot easily "teach" a software to distinguish between different types of road users (e.g., cars, trucks, pedestrians, cyclist, etc.). However, DNNs can extract common features of the road users directly from data and identify these objects in an image by drawing bounding boxes around them. As such, a number of autonomous systems process data from onboard cameras using DNNs (Feng et al. 2021) for key decision-making processes (e.g., object detection).

Figure 1 shows a high-level illustration of a DNN. Circles denote neurons, while edges denote connections between neurons. A DNN can use any numeric property (e.g., pixel values in an image) as input. The values then pass through multiple hidden layers of neurons, each applying weights and biases to the values. Finally, the values are converged into an output layer. For image classification algorithms, the output layer often consists of neurons equal to the number of classes. The j^{th} neuron of the output layer denotes the probability that the input image is of class j. In order to learn the (optimal) weights and bias, DNNs are iteratively trained with training data to minimize a loss function (i.e., a function that measures the difference between the DNN's current output and the expected output). During each epoch (i.e., one iteration of the algorithm through a batch of inputs) (Hardesty 2017), the loss function is calculated for a batch of training data and the gradient descent algorithm is used to discover the direction of change for the weights and biases to minimize the loss. The back propagation algorithm (LeCun et al. 1988) is then used to update the weights and biases accordingly. Through iterative training, DNNs learn to perform their intended tasks by adjusting its parameters (i.e., the weights of the neurons). Once the training phase is finished, the DNN model is verified using the testing data.

2.2 Adversarial examples

Adversarial examples describe machine learning model inputs that are maliciously perturbed to cause a failure in the model (Goodfellow et al. 2015; Szegedy et al. 2014; Serban et al. 2020; Xu et al. 2020). Figure 2 shows an example of an adversarial example generated against an object detection algorithm on an image extracted from

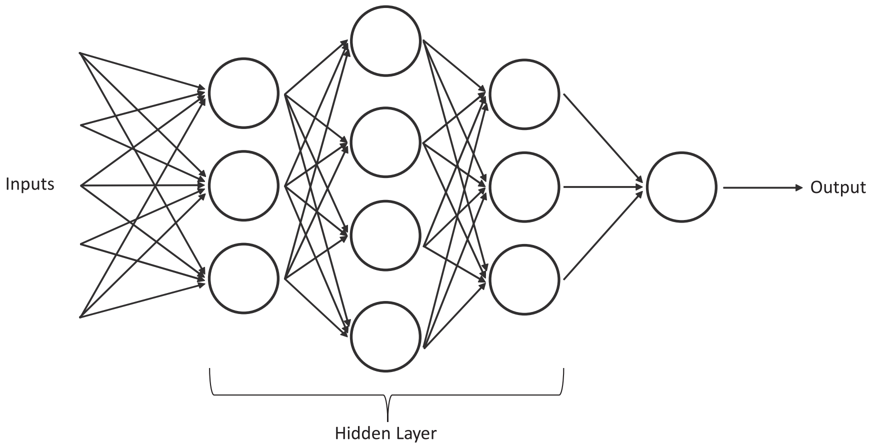


Fig. 1 High-level illustration of a DNN model. Circles denote neurons, while edges denote the flow of data between neurons. The DNN comprises an input layer, multiple hidden layers of neurons, and an output layer

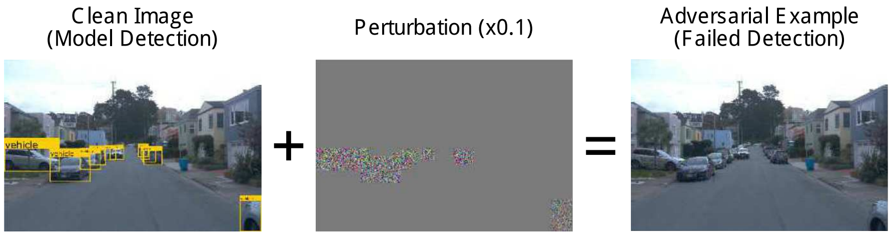


Fig. 2 Example of an adversarial example, where the original clean input image from the Waymo Open Dataset (Sun et al. 2020) with malicious perturbations prevents object detection

the Waymo Open Dataset (Sun et al. 2020). The original input image is shown on the left. The object detection model is capable of identifying objects of interest, drawing yellow bounding boxes around the objects with their corresponding labels. However, when the malicious perturbation noise (scaled by a factor of 10 for visibility purposes) is added to the input image, the combined image (i.e., adversarial example) prevents the correct detection of the objects. A fundamental objective of adversarial examples is that they closely resemble the original images, and thus are not human distinguishable (Metzen et al. 2017; Carlini and Wagner 2017). When safety-critical systems such as autonomous vehicles encounter adversarial examples, the output of the model may lead to dangerous decisions in the autonomous vehicle if it fails to correctly detect obstacles such as pedestrians or other vehicles. An active area of research is to generate adversarial examples that can cause machine learning algorithms to fail in order to identify vulnerabilities and assess the robustness of the system (during the design and testing phases), which then informs robustification, defense, and mitigation strategies against such attacks (Metzen et al. 2017; Serban et al. 2020). We next overview different classifications of existing approaches for adversarial examples.

2.2.1 Targeted and non-targeted adversarial attacks

Adversarial examples may be classified according to the objective of the adversary. For example, an adversary may seek to deceive the model to misclassify an object into an incorrect label in a *non-targeted attack* (Yuan et al. 2019). Specifically, the adversary seeks to prevent the correct classification of the input by introducing carefully crafted perturbations to the original input image. Formally, let F_{img} be a DNN model that takes an input and returns a predicted label, and data (x, y) represents the input x and the ground truth label y. The problem of generating an adversarial perturbation δ in a non-targeted attack can be formulated as Expression (1), where p is any mathematical norm. The adversary seeks to find a minimum perturbation δ such that the DNN misclassifies the input image x (i.e., $F_{img}(x + \delta) \neq y$).

$$\delta_{min} = \underset{\delta}{\operatorname{argmin}} ||\delta||_p \text{ s.t. } F_{img}(x + \delta) \neq y. \tag{1}$$

Another adversarial example attack objective may be to deceive the model and cause it to misclassify an image as a specific (adversary-chosen) label. The problem of generating an adversarial perturbation δ in a *targeted attack* can be formulated as Expression (2). The adversary seeks to find a minimum perturbation δ such that the DNN misclassifies the image into a specific incorrect label, represented by $\hat{y}$. The problem of generating an adversarial example for a targeted attack is more difficult than a non-targeted attack. In a non-targeted attack, an adversary simply discovers a perturbation filter that leads to any incorrect prediction. In a targeted attack, the set of possible adversarial examples and the search space are restricted into the sub-region containing only $\hat{y}$.

$$\delta_{min} = \underset{\delta}{\operatorname{argmin}} ||\delta||_p \text{ s.t. } F_{img}(x + \delta) = \hat{y}. \tag{2}$$

2.2.2 White-box and black-box attacks

Adversarial attacks on machine learning algorithms may also be classified by the assumption of the adversary's access and understanding of the underlying model's architecture and parameters. *White-box* attacks assume that the adversary has access to sensitive information of the model (Yuan et al. 2019). For example, the adversary may have information about the model, weights, gradient information, and/or the architecture of the model. Traditional attack methods such as Szegedy et al.'s Limited Memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimization technique (Szegedy et al. 2014), Fast Gradient Sign Method (FGSM) (Goodfellow et al. 2015), and Dense Adversary Generation (DAG) (Xie et al. 2017) exploit gradient information of the model to be attacked and modify the image by inducing noise based on the gradient information. In contrast, *black-box* attacks assume that the adversary has no prior knowledge of the model to be attacked (Yuan et al. 2019). The adversary has access to a compiled model and may query the model with any input to obtain the model's output, but does not have access to the underlying weights and architecture of the model. In order to attack the model, an adversary must infer the direction of change required by repeatedly querying the model. A black-box attack closely resembles a real-world attack scenario where the development of the DNN model may be proprietary, and only the compiled model is publicly available.

While black-box attacks are more difficult to develop from an adversary's perspective due to the limited information about the model, they also pose unique challenges for model developers to defend the model against such attacks. Specifically, black-box attacks that perturb input images on a pixel level (i.e., modify pixels of the image) require little to no domain knowledge on the underlying DNN model and domain space. This challenge is analogous to the traditional cybersecurity *phishing* attack, where an adversary attempts to extract sensitive information (e.g., passwords) from victims through social engineering. While the adversary may also launch different types of brute-force password attacks to obtain the victim's passwords, brute-force password attacks require the adversary to have existing domain knowledge, such as the intricacies of the encryption algorithm or the structure of the password database. In contrast, a phishing attack requires little to no domain knowledge, and can be executed by any perpetrator with little to no information on the target system. Analogously, black-box attacks that perturb images on a pixel level do not require the adversary to understand the architecture or intricacies of a DNN model, nor do they require additional information of the model. Instead, an adversary simply modifies pixels in the input image until the model misclassifies the image (Alzantot et al. 2019; Vidnerová and Neruda 2020). It is precisely this potential to do harm with such attacks that motivates research into black-box approaches to identify the most adverse adversarial examples (i.e., least perceptible) to enable us to better harden and protect safety-critical systems that use machine learning components.

2.3 Adversarial examples for object detection algorithms

Compared to the image classification problem, object detection is an inherently more difficult problem for both model inference and attacks (Xie et al. 2017). In image classification, an input image consists of exactly one object of interest. The model returns a prediction label with a confidence score, denoting the probability that the object is of the corresponding label. Since the input consists of one object, every pixel in the image may contribute to the output of the model. In object detection algorithms, input images contain multiple objects of interest with different dimensions (i.e., sizes). The objective of the model is to correctly identify objects in the image, draw bounding boxes, and provide the object types. Thus, most regions of the input image may not contribute to the output of the model. If we allow all pixels of the image to be mutated, then the objective of minimal perturbations of adversarial examples may not be satisfied (Metzen et al. 2017). As such, alternative approaches for selecting the perturbation space must be developed for attacking object detection algorithms.

2.4 Evolutionary algorithms

Evolutionary algorithms are a class of optimization algorithms that are inspired by biological evolution observed in nature (Sivanandam and Deepa 2008). Specifically, GAs promote individuals in a population to evolve towards an optimum using evolutionary operators (e.g., mutation, replications, etc.). Individuals (e.g., perturbation filters) are mapped to *genomes*, defining the search space. The corresponding behavior of the individual for the genomes are known as the *phenomes* (e.g., the misbehavior of the model). GAs use techniques analogous to those found in nature to promote and discover genomes that lead to the undesirable phenome (e.g., preventing the correct model output).

In order to evolve solutions using evolutionary search, a developer first initializes a population of random individuals. Next, individuals that are deemed "fit" (i.e., those that have the best performance) are selected for reproduction. During each generation, offspring are created through the crossover operator, where the parent's genomes are combined. Offspring then have a chance to mutate, thereby introducing diversity to the population. By enabling individuals that are more fit to reproduce, a population of individuals iteratively evolves towards an optimum using selective pressure.

3 Methodology

This section describes our proposed evolutionary search-based approach, EvoAttack, to attack object detection models. The remainder of this section describes the progression of our investigations as we explored different strategies to minimize the perturbations of adversarial examples for object detection models. First, we mathematically formulate the expression for a suppressive adversarial attack. Next, we demonstrate that evolutionary search can be used to generate adversarial examples against object detection algorithms. Then, we show that adversarial examples for object detection models can optimize the objective of minimal perturbations by localizing the perturbations added to only pixels in bounding boxes. Finally, we introduce a *dynamic mutation scheme* that enables significant reduction in the number of changed pixels and the degree of change in adversarial examples.

3.1 Suppressive adversarial examples

Object detection algorithms consist of multiple image classification subproblems. Specifically, the algorithm must detect and identify up to n number objects in the image and then label each object. Existing literature for white-box and transfer-based attacks deceive object detection algorithms by causing the model to incorrectly label objects identified (Xie et al. 2017; Wei et al. 2018). They can apply both targeted and non-targeted attacks to generate incorrect labels of objects in the image. This paper presents a variation of an attack, where an adversary's objective is to "suppress" the detection of objects in the image for a given object detection model. We coin the term "*suppressive adversarial attacks*" to categorize these attacks, where the objective of the attacker is to prevent the correct detection of objects in images. Let F_{obj} be an object detector DNN model that takes an input image and returns an annotation[1] containing the set of pairs of (*bounding box, label*) information for each object detected. Then

[1] This paper uses the term annotation to refer to the output of an object detection model.

$len(F_{obj}(x))$ represents the number of objects detected by the DNN model for image x. Data (x, y) represents the input image x and the ground truth label y. The objective of a suppressive adversarial attack is to discover an adversarial example (i.e., the original input image x with added perturbations δ) with minimal perturbations and that causes the DNN to fail to detect any objects (i.e., $len(F_{obj}(x+\delta)) = 0$). Formally, Expression (3) formulates a suppressive adversarial attack.

$$\delta_{min} = \underset{\delta}{\operatorname{argmin}} ||\delta||_p \text{ s.t. } len(F_{obj}(x + \delta)) = 0. \tag{3}$$

Suppressive adversarial attacks lead to different deviant behaviors in the object detection models than traditional targeted or non-targeted attacks. Suppressive adversarial attacks pose a significant challenge, as the behavior of the system may be more adverse if it fails to detect an object as compared to misclassifying an object. For example, if an object detector model mislabels a pedestrian as a static obstacle in an autonomous vehicle, then the autonomous vehicle may still avoid the pedestrian correctly. However, if the same object detector fails to detect the pedestrian completely, then the resulting behavior may be that the autonomous vehicle collides with the pedestrian instead, thereby leading to more adverse outcomes than a misclassification of the pedestrian.

3.2 Applying evolutionary search-based techniques to adversarial examples for object detection models

This section describes our framework, EvoAttack that iteratively uses evolutionary operators to search for perturbations that adversely impact the model's ability to detect objects. We started our exploratory investigation by asking the research question RQ1 of whether evolutionary search-based techniques can be harnessed to generate adversarial examples against object detection models (i.e., suppress the model's ability to detect objects).

Research Question

RQ1: Can evolutionary search be used to generate adversarial examples against object detection algorithms?

In order to answer this question, we use a generational GA (Sivanandam and Deepa 2008). Figure 3 shows a Data Flow Diagram (DFD) for EvoAttack. In the DFD, parallel lines represent external data stores, rectangles specify external entities, green circles denote computational processes, and arrows indicate data flow between processes. The inputs for the EvoAttack process are a (black-box) object detection model and an input image (i.e., the original, non-perturbed image). The object detection model is represented by an external entity that takes image(s) as input(s) and returns the annotation(s) (i.e., bounding boxes and labels of the objects in the image) for each respective image. Next, we describe each of the steps in the DFD used to generate adversarial examples.

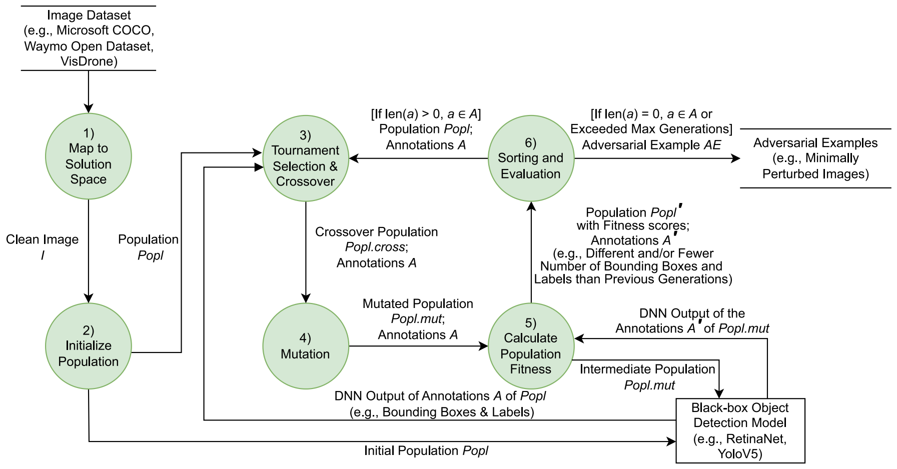


Image Dataset (e.g., Microsoft COCO, Waymo Open Dataset, VisDrone)
1) Map to Solution Space
Clean Image *I*
Population *Popl*
2) Initialize Population
3) Tournament Selection & Crossover
[If len(*a*) > 0, *a* ∈ *A*] Population *Popl*; Annotations *A*
6) Sorting and Evaluation
[If len(*a*) = 0, *a* ∈ *A* or Exceeded Max Generations] Adversarial Example *AE*
Adversarial Examples (e.g., Minimally Perturbed Images)
Crossover Population *Popl.cross*; Annotations *A*
Population *Popl'* with Fitness scores; Annotations *A'* (e.g., Different and/or Fewer Number of Bounding Boxes and Labels than Previous Generations)
4) Mutation
Mutated Population *Popl.mut*; Annotations *A*
5) Calculate Population Fitness
DNN Output of the Annotations *A'* of *Popl.mut*
Intermediate Population *Popl.mut*
DNN Output of Annotations *A* of *Popl* (e.g., Bounding Boxes & Labels)
Black-box Object Detection Model (e.g., RetinaNet, YoloV5)
Initial Population *Popl*

Fig. 3 DFD for EvoAttack's evolutionary process used to generate adversarial examples against object detection models

Step 1) **Map to Solution Space:** Potential solutions are mapped to a genome representation for evolutionary search. In our work, individuals (i.e., images from the datasets) are represented as 3D matrices of the following form: $[RGB\_channel, i, j]$, where each element of the matrix denotes the value of an RGB channel (within the range [0, 255]) for the i^{th}, j^{th} pixel, respectively.

Step 2) **Initialize Population:** The population $Popl$ is initialized by querying the object detection model on the given input image (see the data flow of initial population $Popl$ to obtain the annotations A from "Black-box Object Detection Model" in our DFD) for the population. Annotations contain information regarding object bounding boxes and label information, both of which are used to calculate the fitness score.

Step 3) **Tournament Selection & Crossover:** A point crossover operator is used to create a children population ($Popl.cross$), attempting to find regions of perturbations that are the most effective in suppressing the model's detection capabilities by combining different variations of perturbations. For example, a crossover operator between individual a who successfully suppressed the detection of object 1 and individual b who successfully suppressed the detection of object 2 may lead to individual c that suppresses both objects.

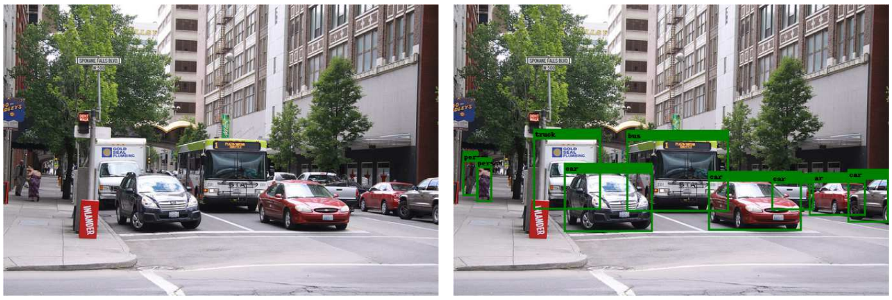
Step 4) **Mutation:** For each member of the population, a percentage of pixels, defined by a probability value P_{mut}, are mutated to form the population $Popl.mut$. This step enables the evolutionary search process to introduce new perturbations to the current population, discovering noise to hinder the detection capabilities of the model. If a pixel is selected for mutation, then perturbations sampled from the range $(\delta_{min}, \delta_{max})$ are added to each channel of the pixel (i.e., increase or reduce the given pixel's color intensity). The values $(\delta_{min}, \delta_{max})$ determine the degree of perturbations introduced. Higher δ values introduce perturbations that are more likely to cause misdetections, but are more likely to be human perceptible.

Step 5) **Calculate Population Fitness:** Once the population has created offspring, their new fitness scores must be computed to assess their performance. This step introduces an optimization strategy, where the model's object detection results are used to configure the structure of the fitness function. First, we evaluate the performance of the individuals for the generation by querying the model (i.e., the data store "Black-box Object Detection Model" in the DFD) to produce their corresponding annotations. Let annotations A' represent the collection of annotations for the population $Popl'$ and annotation $a' \in A'$ denote the annotation for a given individual. We denote the intermediate population with the newly calculated fitness score and DNN annotations with the prime ($'$) symbol. The annotation a' contains the set of pairs of (*bounding box, label*) information for each object detected in a given image. The cardinality of the set a' (i.e., $len(a')$) denotes the number of objects detected by the DNN model for a given individual in the population. The fitness scores for the population are calculated using Expression (4). Specifically, we use the sum of the confidence scores of all objects identified by the model as the fitness score, enabling the automatic adjustment of fitness scores as objects become suppressed from the model's output. As an object is no longer detected by the model when its confidence score is reduced below the detection threshold, our fitness score promotes perturbations that iteratively lower detection confidences until the objects are no longer detected by the model (i.e., each generation should promote different and/or fewer number of bounding boxes and labels when compared to the previous generation).

$$\text{FitnessScore} = \sum_{i=0}^{\text{len(a}')} \text{a}'\text{[i][confidence\_score]}. \tag{4}$$

Step 6) **Sorting and Evaluation:** Finally, the updated population $Popl'$ is sorted by the fitness score. If an adversarial example is found that suppresses all object detections (i.e., no objects are detected), then the algorithm terminates and returns the adversarial example AE. Otherwise (i.e., objects(s) are still detected), the new population $Popl'$ and the corresponding annotations A' return to Step 3) for further iterations of EvoAttack's evolutionary loop. If the maximum number of generations is reached without convergence, then the algorithm fails and is terminated.

While this section presents our basic approach, the remainder of the paper describes refinements and optimizations we developed to augment the mutation strategy (Step 4)) in order to develop the most adverse adversarial examples (i.e., minimal perturbations).



(a) Example of a clean input image from the Microsoft COCO dataset [25].

(b) The same image with the bounding boxes from the object detection model.

Fig. 4 Example image from the Microsoft COCO dataset and the corresponding output of the object detection model for the given image. Bounding boxes are drawn around identified objects discovered by the model

3.3 Effectiveness of localizing perturbations

Several key innovations enable our work to optimize the evolutionary search process and reduce the perturbations in adversarial examples. Our first innovation is the localization of adversarial perturbations added to the image. Figure 4 shows an example input image from the Microsoft COCO dataset (Lin et al. 2014) and the output of an object detection model on the image. The previous subsection described how we used evolutionary search to generate adversarial examples against object detection models, where each pixel in the entire image (Fig. 4a) has an equal chance to mutate. However, since object detection images often contain multiple objects of various sizes, we found that pixels that do not belong to an object (i.e., background pixels) generally do not contribute to the object detection capabilities of an object detection model (i.e., pixels not in bounding boxes of Fig. 4b). Thus, we ask the following research question, RQ2, to study the impact of localizing perturbations while striving for fewer perturbations in the adversarial examples.

Research Question

RQ2: Does localizing perturbations to only pixels in bounding boxes reduce the perturbations needed to generate an adversarial example?

In EvoAttack, instead of allowing any pixels in the entire image to undergo mutation during Step 4), we propose to use the object detection model's annotations from the previous generation to guide the mutation process. The perturbations introduced by our work are localized to only pixels in bounding boxes, as these pixels have significant impact on the model's detection capabilities compared to background pixels. This approach enables EvoAttack to minimize the perturbations required to attack an image, thus preventing the addition of perturbations with low effectiveness to the image.

3.4 Dynamic mutation scheme

Our second key innovation introduces a *dynamic mutation scheme* to further minimize the perturbations added by evolutionary search. While the *adaptive mutation scheme* that appeared in our preliminary work showed that strategically modifying the number of changed pixels can further reduce perturbations, our dynamic mutation scheme is able to further reduce perturbations while reducing the need for hyperparameter tuning. Specifically, the objective of EvoAttack is to introduce perturbations to an image with the objective of finding "ideal" perturbations (i.e., fewest number of changed pixels and smallest degree of change) that hinder object detection. The mutation operator introduced in Step 4) of Sect. 3.2 modifies each pixel with a small mutation rate, P_{mut}, during the mutation step of each generation (Alzantot et al. 2019; Vidnerová and Neruda 2020). This approach can quickly generate adversarial examples that cause a failure in the model's detection ability, where the emphasis is on optimizing computation time. However, we discovered that when applied to images of large dimensions, perturbations would be introduced to many pixels even with a small mutation rate (e.g., $P_{mut} = 0.01$), thus making the attack more likely to be human perceptible (i.e., violating the objective of minimal perturbations (Metzen et al. 2017; Carlini and Wagner 2017)). For example, a pedestrian with a bounding box of dimension 160x200 (i.e., 10% of a 640x500 image) would mutate 320 pixels on average every generation using a standard P_{mut} of 0.01. After 100 generations, every pixel is expected to be mutated at least once. As not all perturbations negatively affect the object detection model's capabilities to the same degree, we propose to perturb the image by adding small perturbations to each object, thereby allowing the selective pressure of an evolutionary system to discover the "ideal" perturbations. However, a challenge for adding a small amount of perturbations in each generation is that the algorithm may not converge in a reasonable amount of time due to the required amount of adversarial noise to attack a given image. Furthermore, we observed that objects in the images of the datasets from different application domains (i.e., terrestrial vs. aerial) had different properties. Specifically, the sizes and resolutions of the objects vary drastically, leading to a disproportional amount of perturbations added to objects of different sizes in the adversarial examples. In order to account for this disparity, we pose the following research question, RQ3.

Research Question

RQ3: Can a dynamic mutation scheme that strategically chooses the number of mutating pixels based on a progress-based criterion of the evolutionary search process be used to reduce perturbations introduced to adversarial examples?

To this end, this work proposes a *dynamic mutation scheme* that has two phases. In the first phase, minimal perturbations (approximately 5% of the original P_{mut}) are added to each bounding box. We linearly scale the added perturbations as the number of generations increases. Specifically, once we reach a given generation threshold (specified by a hyperparameter β) and an adversarial example is not found yet, we can adopt a more aggressive mutation strategy (i.e., each pixel has an increasing chance to mutate) in the second phase to promote convergence.

Algorithm 1 shows the pseudocode for the dynamic mutation scheme used in our framework, where the yellow highlighted text captures the decision logic to transition from the first phase to the second phase. During the mutation step of each generation (i.e., Step 4) of the DFD), we introduce minor perturbations to pixels in bounding boxes. The number of pixels in bounding boxes and coordinates of the bounding boxes are obtained from the annotations of the object detection model's output from the previous generation (Line 2). In the first phase of our dynamic mutation scheme, EvoAttack adds minimal perturbations to a small percentage of pixels in bounding boxes, scaling linearly with the number of generations (Lines 4-5). If the algorithm does not discover an adversarial example before the end of the first phase (number of generations determined by the developer through the hyperparameter β), then the second phase of the dynamic mutation scheme adopts an incrementally aggressive search technique to promote convergence (Lines 6-7). The number of changed pixels n for each object in a bounding box is determined by Expression (5), where the constant hyperparameters α, β, and P_{mut} are empirically determined by the developer based on a trade-off between time and the objective of minimal perturbations. Finally, n number of pixel(s) in each bounding box (bounding box location information obtained in Line 9) are mutated by increasing or decreasing the pixel value(s) by $(\delta_{min}, \delta_{max})$ (Line 10).

Algorithm 1: Pseudo-code for EvoAttack's dynamic mutation scheme.

```
   /* Modify $n$ pixel(s) in the bounding boxes of objects in the image, based
      on Expression (5).                                                    */
1  function DYNAMIC- MUTATION($popl$, $annotations$, $num\_gen$, const $\alpha$, const $\beta$, const $P_{mut}$, const $\delta_{min}$, const
   $\delta_{max}$):
       /* Get the number of pixels in bounding boxes.                       */
2      $num\_of\_pixels = annotations$.getNumOfPixels() ;
3      /* Calculate $n$ based on Expression (5).                            */
4      if $num\_gen <= \beta$ then
5          $n = (num\_of\_pixels * P_{mut} * num\_gen * \alpha)$
6      else
7          $n = (num\_of\_pixels * P_{mut} * (num\_gen/\beta))$
8      /* Get bounding box information from the annotations.                */
9      $boundingboxes = annotations$.getBoundingBoxLocations() ;
       /* Mutate $n$ number of pixels in each bounding box by adding noise
          between $(\delta_{min}, \delta_{max})$.                           */
10     $popl' \leftarrow$ MUTATE($n$, $popl$, $boundingboxes$, $\delta_{min}$, $\delta_{max}$);
11     return $popl'$
```

Specifically, in EvoAttack, we choose small α and P_{mut} values such that the percentage of modified pixels n is small during early generations. The value α enables the developer to adjust the percentage of pixels changed (i.e., rate of growth). A small α value promotes minimal perturbations, while a large α value promotes convergence. For example, consider the pedestrian in an image discussed above with dimension 160x200 with $\alpha = 0.0005$ and $P_{mut} = 0.01$. During early exploration (e.g., generation 1), EvoAttack introduces minor perturbations to the detected object by modifying 1 pixel. By the 500th generation, 80 pixels in the bounding box of the object are mutated. The value β defines the number of generations EvoAttack explores before the algorithm adopts a more aggressive search strategy. A lower β value results in faster convergence (and more perturbations), while a higher β value results in slower convergence (but fewer perturbations). After β number of generations, n is instead based on the number of pixels in the bounding box multiplied by a mutation rate (i.e., P_{mut}) that increases dynamically based on the generation count. Using a dynamic mutation scheme, images that do not require large perturbations to cause a misdetection will not have unnecessary perturbations added, while images that are difficult to perturb will still be promoted to converge as the number of generations increases.

$$n = \begin{cases} \text{num\_of\_pixels} * P_{mut} * (\text{generation} * \alpha) & \text{if generation} \leq \beta \\ \text{num\_of\_pixels} * P_{mut} * (\text{generation}/\beta) & \text{if generation} > \beta \end{cases} \tag{5}$$

4 Empirical studies

We evaluate the efficacy of EvoAttack against state-of-the-art object detection algorithms. First, we empirically establish that evolutionary search can be used as a black-box attack against object detection algorithms, which has not been previously demonstrated. Second, we demonstrate that by localizing the perturbations to pixels in bounding boxes, the perturbations required to suppress the model's correct detection of objects in the image can be significantly reduced. Third, we demonstrate that our dynamic mutation scheme can further reduce the required perturbations, thus leading to even more adverse test data. Finally, we discuss several properties of our approach (i.e., model-, data-, and domain-agnosticism).

4.1 Preliminaries and experimental setup

This section overviews the datasets, models, and evaluation metrics used in our validation work. We also discuss evolutionary parameters used in our experiments.

4.1.1 Object detection benchmark datasets

This work uses three benchmark datasets to validate the proposed technique and to illustrate that our evolutionary search-based attack is not dataset dependent. First, the COCO (Lin et al. 2014) dataset is a large-scale dataset created by Microsoft to promote machine learning progress in object detection, segmentation, and captioning. We also use the Waymo Open Dataset (Sun et al. 2020) for autonomous driving in our studies. The Waymo dataset contains high-quality images taken from a camera mounted atop a vehicle to obtain real-world driving scenarios for object detection. Finally, we also validate our framework on a dataset from a different application domain, VisDrone (Cao et al. 2021), for object detection using a camera mounted atop an aerial drone. The VisDrone dataset contains aerial images (i.e., images with different object counts and resolutions), which belongs in a different application domain than the datasets for terrestrial images.

4.1.2 Object detection models

In our validation studies, we use object detection DNNs implemented using the PyTorch deep learning research platform (Paszke et al. 2019). To demonstrate that our attack is model agnostic, we use a number of different popular model architectures. We demonstrate our experiments using Faster R-CNN MobileNetV3 (Ren et al. 2015) and RetinaNet (Lin et al. 2017) trained using a ResNet-50-FPN backbone (Langford and Cheng 2021) for the Microsoft COCO experiments. For Waymo images, we train an object detector using a RetinaNet with a ResNet-50-FPN backbone (Langford and Cheng 2021). For the VisDrone dataset (Cao et al. 2021), we train and validate our experiments using an object detector with a YoloV5 (Jocher et al. 2022) architecture.

4.1.3 Evaluation metrics for experiments

Several metrics are used to evaluate adversarial examples. First, the number of generations serves as a metric that can be used to differentiate between the different versions of evolutionary search-based and random-based adversarial examples, denoting the number of iterations of evolution required to discover an adversarial example for a given image. Second, we use the number of changed pixels and degree of change, two standard metrics used for adversarial examples in image classification and object detection (Serban et al. 2020; Yuan et al. 2019; Xu et al. 2020), as criteria for the objective of minimal perturbation. Expressions (6) and (7) show the equations used to calculate the number of changed pixels (i.e., L0 norm) and the degree of pixel change (i.e., L2 norm or Euclidean distance), respectively. The variable $\boldsymbol{x}$ represents the modified image, while $\boldsymbol{x_0}$ represents the clean input image.

$$\mathbf{L0} : \text{Number\_Pixels\_Changed}(\boldsymbol{x}, \boldsymbol{x_0}) = \|\boldsymbol{x} - \boldsymbol{x_0}\|_0 \tag{6}$$

$$\mathbf{L2} : \text{Euclidean\_Distance}(\boldsymbol{x}, \boldsymbol{x_0}) = \sqrt{\sum_{i=1}^{\mathrm{n}} (\boldsymbol{x}_i - \boldsymbol{x_0}_i)^2}. \tag{7}$$

The objective of minimizing perturbations is important for adversarial examples, as human agents are often assigned to supervise machine learning components (Kaszas and Roberts 2023; Otsu et al. 2020; Schiaretti et al. 2017; Han et al. 2022) (e.g., an airport agent manually reviewing the object detection algorithm's output on an X-ray image (Liang et al. 2018)). For example, several researchers demonstrated that physical objects can contain adversarial noise (e.g., clothing), causing a failure in an object detection model's ability (Thys et al. 2019; Wu et al. 2020). If such techniques were to be applied to prohibited objects at an airport, then they can potentially deceive and bypass security checkpoints. However, if the adversarial perturbations are visible to the human eye, then the human agent may notice the attack, thus leading to a failed attack (Yuan et al. 2019; Carlini and Wagner 2017). Similarly, a human driver of record for an autonomous vehicle may fail to recognize an attack affecting their vehicle if the added perturbations are not perceptible. For humans tasked with the job of labeling data for DNN training, adversarial noise for attacks must be imperceptible to bypass human detection, but cause deviant model behavior(s). As human-perceptibility is a subjective metric (Wang et al. 2004; Rozsa et al. 2016) (i.e., different people can perceive different levels of perturbations), state-of-the-art techniques use two objective metrics (i.e., number and degree of pixel change) to evaluate the performance of adversarial examples (Serban et al. 2020; Metzen et al. 2017; Carlini and Wagner 2017). Finally, we also provide the computation time used to generate each adversarial example.

4.1.4 Experimental setup

For evolutionary search parameters, we started with values used in state-of-the-art image classification attacks (Alzantot et al. 2019; Vidnerová and Neruda 2020). We experimentally fine-tuned these parameters for object detection. The maximum number of generations is set to 2000, with 16 individuals in each population. Based on commonly used values for these hyperparameters, the hyperparameters for crossover rate, $P_{crossover}$, and mutation rate, P_{mut}, are set to 0.6 and 0.01, respectively. In order to provide a baseline comparison for EvoAttack, we have implemented a random search algorithm that iteratively adds perturbations to a random number of pixels each generation. The random search is not population based and does not use any evolutionary operators other than random mutation. For each experiment, 30 randomly sampled images from the selected datasets are used for comparison. All experiments are performed on a Tesla V100s GPU with an Intel Xeon CPU.

4.2 E1: applying evolutionary computation to adversarial examples for object detection

In our first experiment, *E1*, we explore whether evolutionary search can successfully attack images for object detection models. In order to address this question, we apply the evolutionary search-based algorithm described in Sect. 3.2 to images from an object detection dataset, termed EvoAttack.v0. During each mutation step, perturbations are introduced to the image based on a fixed mutation rate, P_{mut}. If a pixel is chosen for mutation, then perturbations sampled over a uniform distribution between $(\delta_{min}, \delta_{max})$ are introduced to each RGB channel of the pixel, thereby increasing or decreasing the selected pixel's intensity. For our experiments, we used $\delta_{min} = 0.025$ and $\delta_{max} = 0.05$. In order to compare our technique, we also apply random search to the same set of images. For random search, perturbations between $(\delta_{min}, \delta_{max})$ are introduced to a random number of pixels during each generation's mutation step. The random algorithm also terminates after an adversarial example has been found for the given image. We evaluate whether EvoAttack.v0 can successfully hinder the ability of the object detection model to correctly draw bounding boxes and suppress the object detection capabilities. Thirty randomly sampled images from the Microsoft COCO dataset (Lin et al. 2014) and the RetinaNet model (Lin et al. 2017) are used to validate our results. We use the following null and alternate hypotheses to test our research question below.

Research Question

RQ1: Can evolutionary search be used to generate adversarial examples against object detection algorithms?

$H_0(\mu_{\text{diff}} = 0)$: There is no difference in the adversarial examples generated between EvoAttack.v0 and random search.

$H_1(\mu_{\text{diff}} > 0)$: There is a difference in the adversarial examples generated between EvoAttack.v0 and random search.

Table 1 shows the average metrics for the adversarial examples generated against the set of images. This table shows that evolutionary search can successfully discover adversarial examples against the object detection model. Compared to random search, which introduced perturbations to almost every pixel in the images (99.89% of pixels), EvoAttack.v0 only introduced perturbations to approximately 89.04% of pixels. When analyzing the degree of perturbations (i.e., degree of pixel change), EvoAttack.v0 significantly outperformed random search in the objective of minimal perturbation by reducing the degree of perturbations by approximately 3.8 times. We found strong evidence ($p \leq 0.01$ using the Mann–Whitney U test), based on both the number of and degree of pixel change, to reject our null hypothesis and support the alternate hypothesis for RQ1.

Figure 5 shows an example image from the dataset with the corresponding adversarial examples generated by random search and EvoAttack.v0. The model correctly drew bounding boxes around the suitcases in the original input image, shown on the left. The first row shows the adversarial perturbations added to the image (middle), forming the adversarial example (right) for random search. This adversarial example contains visible perturbations, changing almost every pixel in the image by a large degree (168.59). In contrast, while the adversarial example generated by EvoAttack.v0 in the second row still modified many pixels (90.16% of the image), it contains significantly lower degree of perturbations (29.98). While both adversarial examples are able to *completely suppress the ability of the model to detect objects*, this example shows that the visible perturbations (both the number of changed pixels and degree of change) added by random search is significantly higher than evolutionary search.

4.3 E2: localizing perturbations to bounding boxes

In our second experiment, *E2*, we explore whether localizing perturbations to only pixels in bounding boxes can reduce the perturbations required to generate an adversarial example. Our hypothesis is that since pixels not in bounding boxes are not likely to affect the model's decision on the objects, introducing perturbations to pixels not in bounding boxes will add perturbations that do not contribute to the suppression of object detection capabilities. To validate our hypothesis, we ask the following research question and use the following null and alternate hypotheses.

Table 1 Comparison of perturbations measured for adversarial examples generated over thirty input images against a RetinaNet model for the Microsoft COCO dataset

E1: Can Evolutionary Search be Applied? Microsoft COCO - RetinaNet Average Statistic	All Inputs	
Number of objects in input	2.73	
Number of total pixels in image	279,581.60	
Total number of images	30.00	
Average Values (per image)	Random	EvoAttack.v0
Number of generations	67.13	279.54
Number of changed pixels	*279279.31*	***248950.52***
% of pixels in full image changed	*99.89%*	***89.04%***
Degree of perturbations	*187.22*	***49.45***
Computation time (sec)	***624.64***	*991.47*

The first column represents the average performance of adversarial examples generated by random search, while the second column represents evolutionary search.
Bolditalic numbers denote the results for the best performing approach, while italic numbers denote results for the worst performing approach

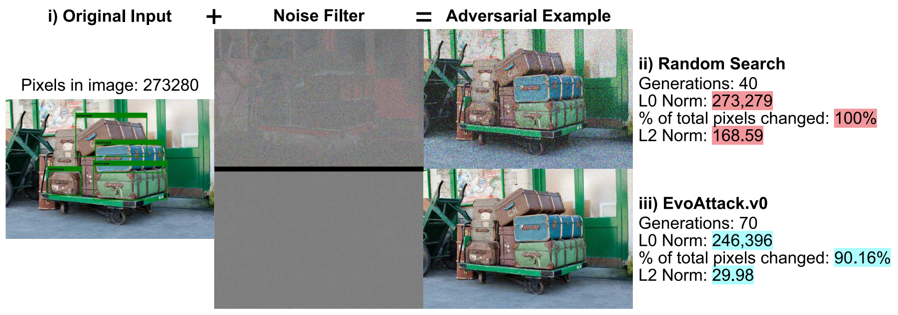


Fig. 5 Two adversarial examples generated by random search (top row) and EvoAttack.v0 (bottom row). The first figure on the left shows the model's output on the original unperturbed input image. The number of pixels in the image is shown above the image. The middle column shows the noise added to the images. The final right column shows the adversarial examples, where the model fails to identify the objects in the images. The adversarial example generated by EvoAttack.v0 contains significantly fewer perturbations (both number of pixels and degree of change)

Research Question

RQ2: Does localizing perturbations to only pixels in bounding boxes reduce the perturbations needed to generate an adversarial example?

$H_0(\mu_{\text{diff}} = 0)$: There is no difference when the perturbations are localized.
$H_1(\mu_{\text{diff}} > 0)$: There is a difference when the perturbations are localized.

We apply our localization technique to the same 30 images sampled from the Microsoft COCO dataset and RetinaNet model in our previous experiment for comparison, thereby enabling a side-by-side comparison of the results from random search and evolutionary search with and without localizing the perturbations. For comparison

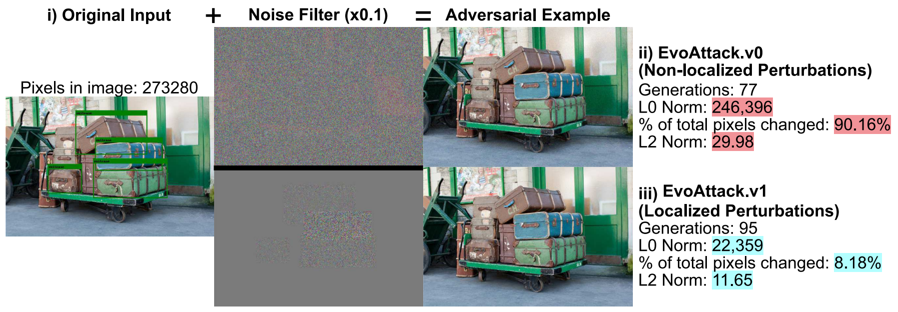


Fig. 6 Comparison of adversarial example generated between localizing and non-localizing perturbations. The first row shows the result for EvoAttack.v0 (non-localized perturbations), where perturbations are seen in the entire image. The second row shows the result for EvoAttack.v1, where perturbations are only added to pixels originally identified as part of a suitcase. The noise filters shown in the middle column are amplified by a factor of 10 for readability

and discussion purposes, we term the intermediate algorithm version with localization EvoAttack.v1.

Table 2 shows the result of Experiment *E2*. The first section of the table (above the bold horizontal line) shows the results of localization for evolutionary search, while the second section (below the bold horizontal line) shows the results of localization for random search. The results indicate that localizing the perturbations have a significant effect on the amount of perturbations in adversarial examples for both random and EvoAttack.v1. Specifically, we found that the localization of perturbations (EvoAttack.v1) leads to an approximately 76.38% reduction in the number of changed pixels and 55.05% reduction in the degree of pixel change in evolutionary search for adversarial examples. For random search, the number of changed pixels is reduced by 61.76% while the degree of pixel change is reduced by 57.19%. We also found strong evidence ($p \leq 0.01$ using the Mann–Whitney U test), based on both the number of and degree of pixel change, to reject our null hypothesis and support the alternate hypothesis for RQ2. As such, this result supports our hypothesis that pixels not in bounding boxes do not contribute significantly to the object detection model's decision-making process. We found that both the evolutionary search-based approach and random-based approach improved with localization. Evolutionary search-based approach still performed better than random search-based approach for both the localized and non-localized experiments.

Finally, Fig. 6 shows two example adversarial examples between non-localized (EvoAttack.v0) and localized (EvoAttack.v1) perturbations, where the adversarial example with localized perturbations contains significantly less noise. In the localized version, only 8.18% of the pixels in the entire image were modified to suppress the detection of the suitcases, compared to 90.16% for the non-localized version.

4.4 E3: Demonstration of the dynamic mutation scheme

In our third experiment, *E3*, we demonstrate how our dynamic mutation scheme can further reduce the perturbations for evolutionary search-based adversarial examples.

Table 2 Comparison of perturbations measured for adversarial examples for Experiment *E2*

E2: Localizing Perturbations to Bounding Boxes Microsoft COCO - RetinaNet Average Statistic	All Inputs	
Number of objects in input	2.73	
Number of total pixels in image	279,581.60	
Number of pixels in bounding boxes	114,670.98	
Percentage of pixels in bounding boxes	41.0%	
Total number of images	30.00	
Average Values (per image)	EvoAttack.v0 (Non-localized)	EvoAttack.v1 (Localized)
Number of generations	279.5	231.67
Number of changed pixels	*248950.52*	$\boxed{\textbf{\textit{58,807.84}}}$
% of pixels in full image changed	*89.04%*	***21.03%***
Degree of perturbations	*49.45*	$\boxed{\textbf{\textit{22.23}}}$
Computation time (sec)	*991.47*	***352.31***
Average Values (per image)	Random.v0 (Non-localized)	Random.v1 (Localized)
Number of generations	67.13	56.46
Number of changed pixels	*279278.31*	***106816.17***
% of pixels in full image changed	*99.89%*	***38.20%***
Degree of perturbations	*187.22*	***80.14***
Computation time (sec)	*624.64*	***154.51***

The first column represents non-localized perturbations, while the second column represents localized perturbations. The first section shows the comparisons between localizing perturbations and non-localizing perturbations for evolutionary search, while the second section shows the comparisons for random search. Bolditalic numbers denote the results for the best performing approach, while italic numbers denote results for the worst performing approach. Boxed numbers highlight the improvement that localization (EvoAttack.v1) provides over EvoAttack.v0 and random

This algorithm is termed EvoAttack to reflect the comprehensive version of our approach that includes all the optimizations described (localized perturbations and dynamic mutation scheme). Specifically, we illustrate the notable impact of EvoAttack's dynamic mutation scheme by comparing its adversarial examples with those generated by random search, and with those generated by the evolutionary search (EvoAttack.v1) from Experiment *E2*. In this experiment, the perturbations introduced are localized to only pixels in bounding boxes, as Experiment *E2* demonstrated that the localization of perturbations results in fewer perturbations. As Experiment *E2* has demonstrated that the localization of perturbations significantly reduces perturbations and computation time for adversarial examples, we only include comparisons with random search and the results from EvoAttack.v1 in subsequent experiments. Values for α and β are selected empirically based on multiple runs of the experiment, set to $5 * 10^{-4}$ and 750, respectively. We set β such that EvoAttack searches for adversarial examples using approximately ten minutes of computation time for each image. The selected values are reaffirmed with consistent results using repeated trials of the experiments. We ask the following research question.

Research Question

RQ3: Can a dynamic mutation scheme that strategically chooses the number of mutating pixels based on a progress-based criterion of the evolutionary search process be used to reduce perturbations introduced to adversarial examples?

$H_0(\mu_{\text{diff}} = 0)$: There is no difference between EvoAttack.v1 and EvoAttack.
$H_1(\mu_{\text{diff}} > 0)$: There is a difference between EvoAttack.v1 and EvoAttack.

Figure 7 shows two adversarial examples generated over a randomly sampled set of input images obtained from the COCO testset against the RetinaNet model. The first column shows the model's output on the clean input image. Bounding boxes are drawn and labeled over objects identified with a confidence score of ≥ 0.7. Next, the noise filters show the differences between the original input and the adversarial examples, amplified by a factor of 10 for readability. The adversarial examples in the rightmost column of images suppress object detection, thereby causing the model to fail to draw bounding boxes around the objects of interest. The adversarial examples shown on top (a.ii)) and bottom (a.iii)) are generated by EvoAttack.v1 (i.e., localization) and EvoAttack (i.e., localization and dynamic mutation), respectively. The perturbation information shows the number of generations required for convergence, number of changed pixels (L0 norm), and degree of change (L2 norm or Euclidean distance) of the adversarial examples.

Figure 7a.i) shows the original image of a scene at a skateboarding event. The model identified multiple objects of interest in the image and drew bounding boxes around them. While both approaches generated adversarial examples, the adversarial example generated by EvoAttack (see Fig. 7a.iii) contains significantly fewer perturbations when compared to the adversarial example generated by EvoAttack.v1 (see Fig. 7a.ii). Since EvoAttack converged on an adversarial example before the β number of generations, the original image in Fig. 7a.i) is considered to have "converged early".

Figure 7b.i) shows an input image of a road scene with a number of cars and people. The generated adversarial examples also caused failures in the model's detections. However, compared to the input image in Fig. 7a.i), the input image in Fig. 7b.i) required more than β number of generations to converge for both (b.ii)) and (b.iii)), where EvoAttack only moderately outperforms EvoAttack.v1 in minimizing perturbations. These figures illustrate that adversarial examples generated by EvoAttack contain fewer perturbations, especially those that converged early before the β number of generations.

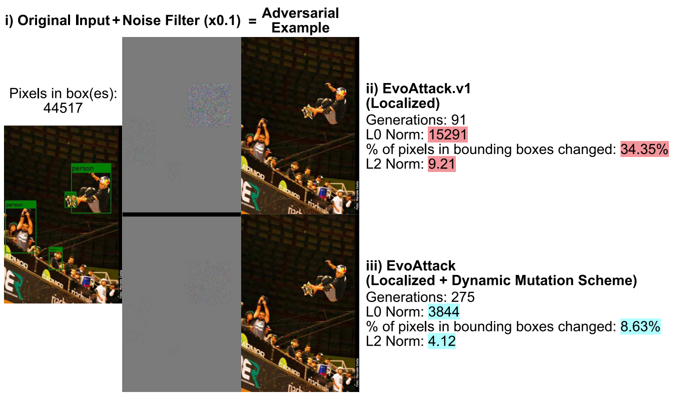
The first section of Table 3 shows the average number of changed pixels and degree of change for adversarial examples generated against 30 randomly sampled input images for the Microsoft COCO dataset used in Experiments *E1* and *E2*. The table shows that both EvoAttack.v1 and EvoAttack perform better than random search, as random search introduces significant perturbations to the image before the model fails to identify the objects. Adversarial examples generated by EvoAttack contain fewer perturbations (both the number of and the degree of change) when compared to adversarial examples generated by EvoAttack.v1. Specifically, the average number of changed pixels is reduced by 36.93% and the average degree of change is reduced by 27.40% using EvoAttack. Upon further inspection, we found a significant number of images (i.e., 24) that converged before β number of generations, where EvoAttack switches to an aggressive search strategy. These adversarial examples are significantly more adverse compared to the random search, require less resources, and less time. The second section of Table 3 shows the measured metrics for adversarial examples that converged early before β number of generations. The metrics for the same set of images (i.e., images that converged early) for random search and EvoAttack.v1 are also provided for comparison. The results indicate that EvoAttack is able to find adversarial examples with significantly fewer perturbations for objects that converged early, with 53.67% reduced number of changed pixels and 48.21% reduced degree of change on average. Compared to random search, EvoAttack found adversarial examples with 77.27% reduced number of changed pixels and 86.23% reduced degree of change. The results of this experiment show that EvoAttack is able to generate adversarial examples with low degree of change and small number of pixels changed. We found strong evidence ($p \leq 0.01$ using the Mann–Whitney U test) to reject our null hypothesis and support the alternate hypothesis for RQ3. Furthermore, the dynamic mutation scheme generated more adverse adversarial examples for a significant majority of the images, while using less resources. As such, given the superior performance of EvoAttack over its intermediate versions, we perform the remaining experimental studies with EvoAttack, comparing its results to random with localization.

4.5 Demonstrating agnostic properties of EvoAttack

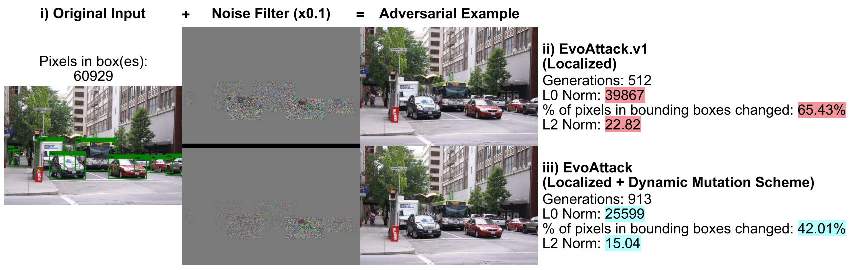
This section applies EvoAttack to a number of different datasets, models, and domain applications to demonstrate that our black-box technique is model, data, and domain agnostic. As our previous experiments (i.e., *E1, E2, E3*) demonstrated that EvoAttack performs better in the objective of minimal perturbations than both previous versions EvoAttack.v0 and EvoAttack.v1, we only include the results from EvoAttack in the following experiments for brevity.

4.5.1 Demonstration that EvoAttack is model agnostic

To demonstrate that EvoAttack is model agnostic, we show that our approach can attack a model of a different architecture. Specifically, we apply EvoAttack to a Faster R-CNN MobileNetV3 (Ren et al. 2015) model and show that we can produce comparable results as an attack against the RetinaNet (Lin et al. 2017). Against the same set of 30 input images, our approach is also able to suppress the ability of the DNN model to detect objects. The first section of Table 4 shows the average perturbations of adversarial examples generated against the Faster R-CNN model, while the second section of the table shows the average perturbations of adversarial examples that have converged before β number of generations. Compared to the RetinaNet model, the Faster R-CNN model is more robust on average against EvoAttack, as it requires more perturbations to suppress object detection. In our studies, we found



(a) Example of an adversarial example that converged *before* β number of generations.


i) Original Input
+
Noise Filter (x0.1)
=
Adversarial Example
Pixels in box(es): 60929
ii) EvoAttack.v1 (Localized)
Generations: 512
L0 Norm: 39867
% of pixels in bounding boxes changed: 65.43%
L2 Norm: 22.82
iii) EvoAttack (Localized + Dynamic Mutation Scheme)
Generations: 913
L0 Norm: 25599
% of pixels in bounding boxes changed: 42.01%
L2 Norm: 15.04

(b) Example of an adversarial example that converged *after* β number of generations.

Fig. 7 Comparisons of adversarial examples generated by EvoAttack.v1 and EvoAttack. The original predicted input image, noise filters (amplified by a factor of 10), adversarial examples, and perturbation information are shown for each image. The Subfigure (i) shows the original input image and the number of pixels in bounding boxes, Subfigure (ii) shows the metrics measured for the evolutionary search (i.e., EvoAttack.v1), and Subfigure (iii) shows the metrics measured for EvoAttack

that the Faster R-CNN model requires almost twice the number of changed pixels and degree of change when compared to the RetinaNet model. Furthermore, the number of adversarial examples that were generated before the β number of generations reduced significantly from 24 to 9 for the Faster R-CNN using EvoAttack, implying that there are fewer images that require low perturbations to cause a misdetection. Thus, this experiment shows that EvoAttack is model agnostic and demonstrates EvoAttack as a testing technique to determine that the Faster R-CNN model is more robust than the RetinaNet model.

4.5.2 Demonstration that EvoAttack is data agnostic

The purpose of this experiment is to demonstrate that EvoAttack is data agnostic within the same application domain and illustrate the potential impact of such attacks on real-world scenarios (e.g., contexts relevant to autonomous vehicles). We apply EvoAttack to a RetinaNet (Langford and Cheng 2021) trained over the Waymo Open Dataset (Sun et al. 2020). The trained model predicts vehicles, pedestrians, and cyclists

Table 3 Comparison of perturbations measured for adversarial examples generated over thirty input images against a RetinaNet model for the Microsoft COCO dataset (Lin et al. 2014)

E3: Demonstration of the Dynamic Mutation Operator			
Microsoft COCO - RetinaNet			
Average Statistic	All Inputs		
Number of objects in input	2.93		
Number of pixels in bounding boxes	114,670.98		
Total number of images	30.00		
Average Values (per image)	Random.v1 (Localized)	EvoAttack.v1 (Localized)	EvoAttack (Localized + Dynamic)
Number of generations	60.90	231.67	535.47
Number of changed pixels	*106816.17*	58807.84	***37089.50***
% of pixels in bounding boxes changed	*93.15%*	51.28%	***32.34%***
Degree of perturbations	*80.14*	22.23	***16.14***
Computation time (sec)	***154.51***	352.31	*526.94*
Average Statistic	Early Convergence		
Number of objects in input	2.58		
Number of pixels in bounding boxes	114,429.82		
Total number of images	24.00		
Average Values (per image)	Random.v1 (Localized)	EvoAttack.v1 (Localized)	EvoAttack (Localized + Dynamic)
Number of generations	56.46	149.42	382.08
Number of changed pixels	*103957.09*	51005.42	***23631.71***
% of pixels in bounding boxes changed	*90.85%*	44.57%	***20.65%***
Degree of perturbations	*67.25*	17.88	***9.26***
Computation time (sec)	***122.53***	215.65	*286.09*

The metrics for the same set of images from a random search, EvoAttack.v1 from Experiment *E2*, and EvoAttack are provided for comparison. The first section of the table labeled "All Inputs" show the average metrics for all 30 input images. The lower section of the table labeled "Early Convergence" shows the average metrics for the (24) adversarial examples that converged early before β number of generations. Bolditalic numbers denote the results for the best performing approach, while italic numbers denote results for the worst performing approach

for a camera mounted atop a vehicle. Thus, if an adversary successfully prevents correct object detection, then the resulting behavior of the system may lead to significant consequences such as serious injuries or even deaths. We apply EvoAttack to 30 randomly chosen images sampled over the Waymo Open Dataset and demonstrate the results for random search and EvoAttack. Table 5 shows the experiment results for the Waymo Open Dataset. Similar to our previous experiments, the adversarial examples generated by random search show large perturbations, modifying most of the pixels in bounding boxes. In contrast, adversarial examples generated by EvoAttack show significantly fewer perturbations, both for the number of changed pixels and for the degree of change. For the adversarial examples that converged early in the Waymo experiment, we found that the degree of pixel change required to generate

Table 4 Comparison of perturbations for adversarial examples against a Faster R-CNN model

Demonstration that EvoAttack is Model Agnostic Microsoft COCO - Faster R-CNN MobileNet V3 Average Statistic	All Inputs	
Number of objects in input	2.93	
Number of pixels in bounding boxes	146,338.12	
Total number of images	30.00	
Average Values (per image)	Random	EvoAttack
Number of generations	73.20	964.83
Number of changed pixels	*138829.08*	***97014.17***
% of pixels in bounding boxes changed	*94.87%*	***66.29%***
Degree of perturbations	*112.85*	***40.16***
Computation time (sec)	***347.67***	*1092.88*
Average Statistic	Early Convergence	
Number of objects in input	1.67	
Number of pixels in bounding boxes	107,290.56	
Total number of images	9.00	
Average Values (per image)	Random	EvoAttack
Number of generations	59.44	414.00
Number of changed pixels	*120375.00*	***46414.67***
% of pixels in bounding boxes changed	*112.20%*	***43.26%***
Degree of perturbations	*82.48*	***13.44***
Computation time (sec)	*270.85*	***216.16***

The first section of the table labeled "All Inputs" shows the average metrics for all thirty input images. The second section of the table labeled "Early Convergence" shows the average metrics for the adversarial examples that converged early before β number of generations.
Bolditalic numbers denote the results for the best performing approach, while italic numbers denote results for the worst performing approach

an adversarial example is significantly reduced from 18.84 to 3.72 (approximately 5 times less).

Figure 8 shows two adversarial examples. This result shows that our black-box approach successfully introduced adversarial perturbations to cause the model to fail to draw correct bounding boxes around vehicles, pedestrians, and cyclists in the images. If an object detection model used in a safety-critical application, such as an autonomous vehicle, is compromised using such attacks, then the behavior of the vehicle may result in a collision with surrounding vehicles or people.

4.5.3 Demonstration that EvoAttack is domain agnostic

This section shows the results of applying EvoAttack to object detection models for aerial drone data. Aerial drone perception data differs from terrestrial autonomous vehicle perception, as drones operate in the air and do not face the same safety-critical constraints of collisions with other road users (e.g., pedestrians, vehicles, cyclist,

Table 5 Comparison of perturbations measured for adversarial examples generated over thirty input images against a RetinaNet model for the Waymo Open Dataset (Sun et al. 2020)

Demonstration that EvoAttack is Data Agnostic Waymo Open Dataset - RetinaNet Average Statistic	All Inputs	
Number of objects in input	5.00	
Number of pixels in bounding boxes	57,083.42	
Total number of images	30.00	
Average Values (per image)	Random	EvoAttack
Number of generations	32.70	535.47
Number of changed pixels	*56690.64*	***36951.07***
% of pixels in bounding boxes changed	*99.31%*	***64.73%***
Degree of perturbations	*37.58*	***18.84***
Computation time (sec)	***30.24***	*472.79*
Average Statistic	Early Convergence	
Number of objects in input	2.73	
Number of pixels in bounding boxes	25,209.30	
Total number of images	10.00	
Average Values (per image)	Random	EvoAttack
Number of generations	13.55	327.00
Number of changed pixels	*21889.46*	***4678.36***
% of pixels in bounding boxes changed	*86.83%*	***18.56%***
Degree of perturbations	*21.74*	***3.72***
Computation time (sec)	***6.00***	*134.24*

Blue numbers denote the results for the best performing approach, while red numbers denote results for the worst performing approach

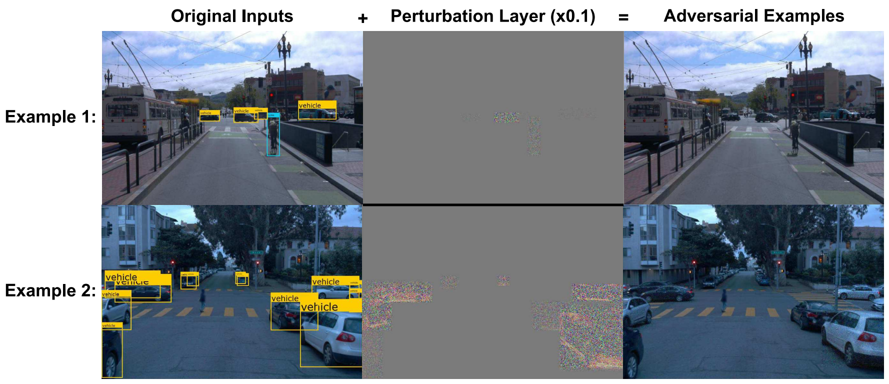


Fig. 8 Adversarial examples generated against the Waymo Open Dataset (Sun et al. 2020). The perturbation layer is amplified by a factor of ten for visualization. The object detection algorithm fails to detect any object in the adversarial example

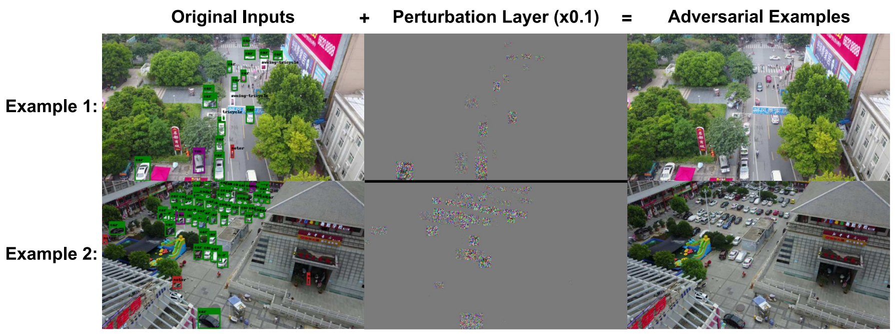


Fig. 9 Adversarial examples generated against the VisDrone dataset (Cao et al. 2021), a dataset for aerial drone images (bounding boxes delimited in various colors). Compared to the Waymo Open Dataset (Sun et al. 2020) for autonomous driving, objects in these images are smaller in dimensions and contains more objects per image

etc.). Additionally, objects captured in drone images are typically smaller in dimensions compared to those captured onboard of an autonomous vehicle, as the drone captures images from high altitudes (and thus further away from the objects). Finally, drone-captured data typically contains more objects per image (15 objects on average, compared to 5 objects on average in the Waymo Open Dataset (Sun et al. 2020) for our experimental images). As such, this experiment highlights the flexibility of EvoAttack across different domains and applications, showing the ability of our dynamic mutation scheme to account for objects of different resolutions. We apply EvoAttack to attack a YoloV5 model (Jocher et al. 2022) trained on the VisDrone dataset (Cao et al. 2021), a popular large-scale dataset for aerial drone data. VisDrone contains a wide range of urban and country scenes with various object densities captured using drone-mounted cameras. We randomly selected 30 images from the VisDrone dataset. Table 6 shows the comparison of adversarial examples generated against the VisDrone dataset and the comparison for those that converged early for random search and EvoAttack. Similar to the previous experiments, EvoAttack generated adversarial examples that changed the fewest number of pixels with the lowest degree of change. When compared to terrestrial data, adversarial examples for the VisDrone aerial data show that the percentage of changed pixel and degree of pixel change is notably lower in the VisDrone experiments, as the objects are smaller in size (i.e., they have lower resolution), and thus are more sensitive to perturbations.

Figure 9 shows several adversarial examples generated against images sampled from the VisDrone dataset (Jocher et al. 2022). Compared to previous applications, images in the VisDrone dataset show objects of small dimensions. The images show that EvoAttack is capable of generating adversarial examples that suppress the ability of the YoloV5 model to correctly draw bounding boxes around any objects previously identified (i.e., prior to the adversarial attack).

Table 6 Comparison of perturbations measured for adversarial examples generated over thirty input images against a YoloV5 model (Jocher et al. 2022) for the VisDrone Dataset (Cao et al. 2021)

Demonstration that EvoAttack is Domain Agnostic VisDrone Dataset - YoloV5 Average Statistic	All Inputs	
Number of objects in input	**15.30**	
Number of pixels in bounding boxes	15,001.78	
Total number of images	30.00	
Average Values (per image)	Random	EvoAttack
Number of generations	38.40	851.80
Number of changed pixels	*14504.27*	***8900.53***
% of pixels in bounding boxes changed	*96.68%*	***59.33%***
Degree of perturbations	*17.79*	***10.12***
Computation time (sec)	***13.82***	*393.33*
Average Statistic	Early Convergence	
Number of objects in input	6.47	
Number of pixels in bounding boxes	4987.19	
Total number of images	15.00	
Average Values (per image)	Random	EvoAttack
Number of generations	18.87	397.33
Number of changed pixels	*4502.73*	***1092.53***
% of pixels in bounding boxes changed	*90.29%*	***21.91%***
Degree of perturbations	*8.56*	***2.36***
Computation time (sec)	***1.87***	*143.90*

Blue numbers denote the results for the best performing approach, while red numbers denote results for the worst performing approach

4.6 Discussion

This section discusses the findings of our experiments. We provide insights for our results and lessons learned.

4.6.1 Experiment findings and insights

In our progression of experiments, we explored several properties of adversarial examples generated by evolutionary search. First, we find that perturbations affect different objects from various datasets and models differently. Specifically, the confidence scores for different objects are reduced at different rates using the same amount of perturbations. This implies that some objects may be more robust towards perturbations (e.g., adding a given amount of perturbations does not have a large effect on the model's confidence score), while other objects are more susceptible to perturbations (e.g., adding the same amount of perturbations does have a large effect on the model's confidence score). We believe this is because the trained model may be better at identifying some object classes, while other object classes may perform poorly.

Since the confidence scores are reduced at different rates, we found that objects are incrementally suppressed by EvoAttack (i.e., objects disappear one at a time and are not dependent on each other), rather than disappearing together during the same evolutionary step. As perturbations have different degrees of effect for different objects, we did not find any direct correlation between the number of objects initially detected by the object detection model and the execution time of EvoAttack to suppress the detection of all objects.

Because EvoAttack localizes its perturbations according to annotations generated by the DNN (i.e., object boundaries and classification labels), it can also be used to generate targeted suppression attacks, where only a specific adversary-provided type of objects are suppressed. For example, an image from the Waymo Open Dataset for autonomous vehicles may contain a number of pedestrians, cyclists, and vehicles. EvoAttack can leverage the output of the DNN and suppress only the detection of "pedestrians" (i.e., only add perturbations to the bounding boxes with the given label), causing the autonomous vehicle to fail to detect only pedestrians.

Finally, we have demonstrated the use of EvoAttack as a testing technique to assess and compare the robustness of two DNNs. When comparing the adversarial examples of the RetinaNet model (Table 3) and the Faster R-CNN model (Table 4) for the same 30 input images of the Microsoft COCO dataset, we find that the Faster R-CNN model is more robust on average than the RetinaNet, requiring more perturbations and resources to cause the model to misbehave.

4.6.2 Utility of the dynamic mutation scheme

To illustrate the flexibility of EvoAttack, we have demonstrated its ability to attack various object detection datasets and models. We demonstrated EvoAttack's *data* agnostic property by attacking three different datasets, Microsoft COCO dataset, Waymo Open Dataset, and the VisDrone dataset. We demonstrated EvoAttack's *model* agnostic property by attacking three different models of various architectures and sizes, the RetinaNet, Faster R-CNN, and the YoloV5 model. Finally, we demonstrated EvoAttack's *domain* agnostic property by applying it to datasets from different application domains (specifically terrestrial and aerial). Images from different domains contain different properties that may challenge adversarial examples in different ways. For example, the terrestrial data explored in our work contains few but large objects in the images (e.g., a centered picture of a person), while the VisDrone aerial data contains many but small objects (e.g., a picture of a parking lot from the sky). Similarly, satellite images may also contain many objects with small resolutions, thereby leading to a different application domain than terrestrial data. Since the smaller objects contain lower resolution, pixels in those objects are more important for the decision-making of the object detection model (i.e., there are fewer pixels from which to obtain information). If equal amounts of perturbations are added to the objects from the different datasets, then the objective of minimal perturbations (Carlini and Wagner 2017) may not be satisfied for the dataset with small objects. As a result, we were motivated to reconsider and revise our adaptive mutation scheme in our preliminary work (Chan and Cheng 2022) to better account for various object sizes, thereby avoiding the need to retune parameters across the different application domains.

4.6.3 Hyperparameter tuning

In our validation experiments, we selected parameters for evolutionary search based on existing work on image classification algorithms (Vidnerová and Neruda 2020; Alzantot et al. 2019). For hyperparameters used in our dynamic mutation algorithm, we empirically tuned our parameters α and β based on an experimental balance between computation time and perturbations added. The value α is set to a small number to promote minimal perturbations to be introduced to the image during early generations (if $\alpha = 1$, then the search effectively becomes a basic evolutionary search with mutation rate P_{mut}). Future work may explore a number of techniques, such as grid search or Bayesian optimization, to better tune the hyperparameters used in our work. Finally, the number of generations EvoAttack searches before switching to an aggressive search strategy, β, may be further optimized in future work to automatically configure based on a plateau of the population's average performance in a sequence of generations. Specifically, if the fitness score of the population does not show improvements above a threshold ϵ for a set number of generations, then EvoAttack may switch to the aggressive search strategy to promote convergence.

5 Related work

This section overviews related work in the area of adversarial examples, black-box approaches, and current research on adversarial examples applied to object detection. While other works have explored evolutionary search-based adversarial examples for classification problems, this paper, in contrast, examines how evolutionary search-based approaches can be used to attack object detection algorithms.

A number of researchers have proposed work on white-box adversarial examples. Szegedy et al. (2014) introduced the first adversarial examples, revealing the existence of malicious images that machine learning models fail to predict correctly. Carlini and Wagner (2017) introduced the C&W attack similar to that of Szegedy et al.'s attack (Szegedy et al. 2014). Goodfellow et al. (2015) proposed the FGSM algorithm to perturb the image based on the signed gradient. However, most of the adversarial example generation techniques explore white-box attacks, where the gradient and other sensitive information of the underlying model are not hidden from the adversary. Our approach assumes a black-box model where the adversary does not have access to model weights and architecture.

Several researchers have explored the use of black-box evolutionary approaches to generate adversarial examples for image classification algorithms, but to the best of our knowledge, these techniques have not targeted object detection algorithms. Su et al. (2019) proposed a one-pixel attack using differential evolution. Alzantot et al. (2019) proposed GenAttack, which applies a variation of GA to discover adversarial examples. Vidnerová and Neruda (2020); Chen et al. (2019); Wu et al. (2021), and Han et al. (2019) proposed similar GA approaches. These approaches use different evolutionary search techniques (e.g., evolutionary algorithms, GAs, multi-objective GAs etc.) and target different applications and datasets. Chan and Cheng (2023) proposed a novelty search-based approach to generate diverse adversarial examples against image classification algorithms.

Other research has explored generating adversarial examples against object detection models. Xie et al. (2017) proposed the DAG algorithm that calculates the gradients with respect to all correctly-labeled objects and accumulates perturbations that reduce the model's output confidence. The authors applied their technique to previous state-of-the-art networks, such as the FCN framework and Faster R-CNN Ren et al. (2015) on the PascalVOC dataset. In contrast, Wei et al. (2019) proposed a transfer-based attack based on a Generative Adversarial Network (GAN). Li et al. (2021) proposed a similar GAN-based technique to generate adversarial examples against black-box algorithms. However, these approaches require the additional training of a GAN model for each dataset and model, and thus are not model- or data-agnostic. Furthermore, transferability attacks do not guarantee success. Cai et al. (2022) proposed an attack that generates context-specific adversarial examples. However, their approach leverages the Projected Gradient Descent (PGD) algorithm to generate perturbations, and therefore is not black-box. In addition, their attack seeks to change the correct labeling of object(s) in an image, rather than preventing the correct detection of objects.

Finally, a few recent research efforts have explored black-box adversarial example approaches for object detection algorithms. However, these approaches have only explored biomimetic techniques (i.e., techniques that mimic phenomena observed in organisms in nature, rather than undergoing an evolutionary process to evolve solutions). Wang et al. (2020) proposed a particle swarm-based algorithm to generate adversarial examples. Their algorithm first generates a number of candidate adversarial examples with large perturbations, and then use particle swarm algorithms to reduce the perturbations iteratively by optimizing only the degree of pixel change. However, their approach does not localize the perturbations to pixels in bounding boxes, thereby introducing more perturbations than necessary to attack the image. Additionally, their attack requires a large query count in order to reduce perturbations. Ye et al. (2022) and Sun et al. (2023) proposed a similar approach, but leverages differential evolution instead. Lapid and Sipper (2023) proposed an adversarial attack against object detection algorithms using adversarial patches, preventing the model from detecting a single object per attack using a physical object (i.e., attacking physical objects by attaching images of visible adversarial noise). These approaches do not use an evolutionary search-based technique for a true black-box, model- and data-agnostic approach to generate an adversarial example for object detection algorithms with the objective of minimal perturbations (i.e., minimizing both the number of pixels perturbed and the degree of change).

6 Threats to validity

The results in this paper are limited to adversarial examples generated using evolutionary search on DNNs for object detection algorithms. The results of the experiments may vary with each run, as evolutionary search-based algorithms rely on non-determinism to evolve solutions. To ensure the feasibility of the approach, a wide variety of randomly sampled images were chosen from a number of different datasets and disciplines. Additionally, the measured Coefficient of Variation (CV) for a wide variety of inputs and models over multiple repetitions of the experiments of EvoAttack are all less than 0.15, indicating that multiple runs of EvoAttack on the same image produce adversarial examples with similar and comparable degree of perturbations and number of generations. For random search, a high variance is measured in repeated experiments due to the broad variation in the number of pixels changed. However, the perturbations (i.e., L0 and L2 norms) of the best performing adversarial examples of the repeated random searches are still significantly worse than EvoAttack's adversarial examples. The images selected as examples for display are also chosen randomly. There is no post-selection process applied.

Applying EvoAttack to generate adversarial examples against object detection algorithms may require re-tuning of parameters, as hyperparameters are sensitive to changes in datasets or models. Further research may be needed to better inform the selection of hyperparameters used in this work. Specifically, setting the hyperparameter β to be too small results in high perturbations, while setting a large β results in long computation time before an adversarial example may be found. Additional analysis is needed to find the optimal β value to balance computation time and perturbations added. Finally, additional research may be required to assess the effectiveness of such attacks in a real-world application setting against cyber-physical systems.

7 Conclusion

This paper described EvoAttack, an evolutionary search-based attack to generate adversarial examples against object detection algorithms. We showed that our approach can attack object detection algorithms without having access to model parameters, architecture, or estimates of the gradient. The search-based process uses the results of previous iterations of the evolutionary process to configure the structure of the fitness function and guide the mutation process. Furthermore, we introduced a dynamic mutation scheme that reduces both the number of perturbations and the degree of change for object detection adversarial examples. We conducted a series of experiments to show how adversarial examples can be generated against images from different datasets and models of different architectures.

Future research will explore potential improvements to our evolutionary search-based attack and investigate detection and mitigation strategies. They include potential improvements using multi-objective GAs (e.g., NSGA-II Deb et al. (2002)) and parallel GAs Nowostawski and Poli (1999). Different hyperparameter tuning approaches may be explored to identify optimal hyperparameters for EvoAttack. We will also perform more empirical studies to compare the effectiveness of EvoAttack with existing white-box attacks. Additionally, research to improve the robustness of object detection models through adversarial training with EvoAttack will be studied. Furthermore, novelty search Langford and Cheng (2021); Lehman and Stanley (2011) may be leveraged to discover a collection of adversarial examples that causes diverse behaviors in the model. Finally, Enlil (Langford and Cheng 2021) (i.e., behavior oracles) may be used to predict the uncertain behavior of the object detection model when exposed to adversarial examples.

Acknowledgements This work was supported in part by funding provided by Michigan State University and the BEACON Center.

Author Contributions K.H.C. and B.H.C. have contributed equally to this work.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no competing interests.

References

Alzantot et al.: GenAttack: Practical black-box attacks with gradient-free optimization. In: Proceedings of the genetic and evolutionary computation conference, pp. 1111–1119 (2019)

Cai, Z., Rane, S., Brito, A.E., Song, C., Krishnamurthy, S.V., Roy-Chowdhury, A.K., Asif, M.S.: Zero-query transfer attacks on context-aware object detectors. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 15024–15034 (2022)

Cao, Y., He, Z., Wang, L., Wang, W., Yuan, Y., Zhang, D., Zhang, J., Zhu, P., Van Gool, L., Han, J., et al.: Visdrone-det2021: the vision meets drone object detection challenge results. In: Proceedings of the IEEE/CVF international conference on computer vision, pp. 2847–2854 (2021)

Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 Ieee symposium on security and privacy (sp), pp. 39–57. IEEE (2017)

Chan, K., Cheng, B.H.C.: Evoattack: an evolutionary search-based adversarial attack for object detection models. In: International symposium on search based software engineering, pp. 83–97. Springer (2022)

Chan, K.H., Cheng, B.H.C.: Expound: a black-box approach for generating diversity-driven adversarial examples. In: International symposium on search based software engineering, pp. 19–34. Springer (2023)

Chen, J., Su, M., Shen, S., Xiong, H., Zheng, H.: POBA-GA: perturbation optimized black-box adversarial attacks via genetic algorithm. Comput. Secur. **85**, 89–106 (2019)

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)

Eykholt et al.: Robust physical-world attacks on deep learning visual classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1625–1634 (2018)

Feng, D., Harakeh, A., Waslander, S.L., Dietmayer, K.: A review and comparative study on probabilistic object detection in autonomous driving. IEEE Trans. Intell. Transp. Syst. **23**(8), 9961–9980 (2021)

Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International conference on learning representations. https://doi.org/10.48550/arXiv.1412.6572. Cited by 21194 (2015)

Han, J.K., Kim, H., Woo, S.S.: Nickel to lego: using Foolgle to create adversarial examples to fool google cloud speech-to-text API. In: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security. CCS '19, pp. 2593–2595. Association for Computing Machinery, New York, NY, USA (2019)

Han, J., Davids, J., Ashrafian, H., Darzi, A., Elson, D.S., Sodergren, M.: A systematic review of robotic surgery: from supervised paradigms to fully autonomous robotic approaches. Int. J. Med. Robot. Comput. Assist. Surg. **18**(2), 2358 (2022)

Hardesty, L.: Explained: neural networks. MIT News **14**, (2017)

Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., Kwon, Y., Michael, K., Fang, J., Yifu, Z., Wong, C., Montes, D., et al.: ultralytics/yolov5. Zenodo (2022)

Kaszas, D., Roberts, A.C.: Comfort with varying levels of human supervision in self-driving cars: determining factors in Europe. Int. J. Transp. Sci. Technol. **12**(3), 809–821 (2023)

Knight, J.C.: Safety critical systems: challenges and directions. In: Proceedings of the 24th International conference on software engineering. ICSE 2002, pp. 547–550 (2002)

Kocić, J., Jovičić, N., Drndarević, V.: An end-to-end deep neural network for autonomous driving designed for embedded automotive platforms. Sensors **19**(9), 2064 (2019)

Langford, M.A., Cheng, B.H.C.: Know What You Know: predicting behavior for learning-enabled systems when facing uncertainty. In: 2021 international symposium on software engineering for adaptive and self-managing systems (SEAMS), pp. 78–89 (2021)

Langford, M.A., Cheng, B.H.C.: Enki: a diversity-driven approach to test and train robust learning-enabled systems. ACM Trans. Auton. Adapt. Syst. (TAAS) **15**(2), 1–32 (2021)

Lapid, R., Sipper, M.: Patch of invisibility: naturalistic black-box adversarial attacks on object detectors. arXiv preprint arXiv:2303.04238 (2023)

LeCun, Y., Touresky, D., Hinton, G., Sejnowski, T.: A theoretical framework for back-propagation. In: Proceedings of the 1988 connectionist models summer school, vol. 1, pp. 21–28 (1988)

Lehman, J., Stanley, K.O.: Abandoning objectives: evolution through the search for novelty alone. Evol. Comput. **19**(2), 189–223 (2011)

Li, X., Jiang, Y., Liu, C., Liu, S., Luo, H., Yin, S.: Playing against deep-neural-network-based object detectors: a novel bidirectional adversarial attack approach. IEEE Trans. Artif. Intell. **3**(1), 20–28 (2021)

Liang, K.J., Heilmann, G., Gregory, C., Diallo, S.O., Carlson, D., Spell, G.P., Sigman, J.B., Roe, K., Carin, L.: Automatic threat recognition of prohibited items at aviation checkpoint with X-ray imaging: a deep learning approach. In: Anomaly Detection and Imaging with X-Rays (ADIX) III, vol. 10632, p. 1063203. SPIE (2018)

Lin et al.: Microsoft coco: common objects in context. In: European conference on computer vision, pp. 740–755. Springer (2014)

Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp. 2980–2988 (2017)

Marie-Sainte, S.L., Alamir, M.B., Alsaleh, D., Albakri, G., Zouhair, J.: Enhancing credit card fraud detection using deep neural network. In: Arai, K., Kapoor, S., Bhatia, R. (eds.) Intelligent computing, pp. 301–313. Springer, Cham (2020)

Metzen, J., Genewein, T., Fischer, V., Bischoff, B.: On detecting adversarial perturbations. In: Proceedings of the 5th international conference on learning representations. https://doi.org/10.48550/arXiv.1702.04267 . Cited by 1140 (2017)

Moosavi-Dezfooli, S.-M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2574–2582 (2016)

Nandita, G., Chandra, T.M.: Malicious host detection and classification in cloud forensics with dnn and sflo approaches. Int. J. Syst. Assur. Eng. Manag. 1–13 (2021)

Nowostawski, M., Poli, R.: Parallel genetic algorithm taxonomy. In: 1999 Third international conference on knowledge-based intelligent information engineering systems. Proceedings (Cat. No. 99TH8410), pp. 88–92. IEEE (1999)

Otsu, K., Tepsuporn, S., Thakker, R., Vaquero, T.S., Edlund, J.A., Walsh, W., Miles, G., Heywood, T., Wolf, M.T., Agha-Mohammadi, A.-A.: Supervised autonomy for communication-degraded subterranean exploration by a robot team. In: 2020 IEEE aerospace conference, pp. 1–9. IEEE (2020)

Paszke, et al.: PyTorch: an imperative style. Curran Associates, Inc, High-Performance Deep Learning Library (2019)

Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. Adv. Neural Inf. Process. Syst. **28**, 2969239 (2015)

Ren, K., Zheng, T., Qin, Z., Liu, X.: Adversarial attacks and defenses in deep learning. Engineering **6**(3), 346–360 (2020)

Rozsa, A., Rudd, E.M., Boult, T.E.: Adversarial diversity and hard positive generation. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 25–32 (2016)

Rudd, E.M., Harang, R., Saxe, J.: Meade: Towards a malicious email attachment detection engine. In: 2018 IEEE international symposium on technologies for homeland security (HST), pp. 1–7 (2018)

Schiaretti, M., Chen, L., Negenborn, R.R.: Survey on autonomous surface vessels: Part i-a new detailed definition of autonomy levels. In: Computational Logistics: 8th international conference, ICCL 2017, Southampton, UK, October 18-20, 2017, Proceedings 8, pp. 219–233. Springer (2017)

Serban, A., Poll, E., Visser, J.: Adversarial examples on object recognition: a comprehensive survey. ACM Comput. Surv. **53**(3), 1–38 (2020)

Sivanandam, S., Deepa, S.: Introduction to genetic algorithms. Springer (2008)

Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Trans. Evol. Comput. **23**(5), 828–841 (2019)

Sun et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 2446–2454 (2020)

Sun, J., Yao, W., Jiang, T., Wang, D., Chen, X.: Differential evolution based dual adversarial camouflage: fooling human eyes and object detectors. Neural Netw. **163**, 256–271 (2023). https://doi.org/10.1016/j.neunet.2023.03.041

Szegedy et al.: Intriguing properties of neural networks. In: International conference on learning representations. https://doi.org/10.48550/arXiv.1312.6199 . Cited by 16845 (2014)

Thys, S., Van Ranst, W., Goedemé, T.: Fooling automated surveillance cameras: adversarial patches to attack person detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. (2019)

Vidnerová, P., Neruda, R.: Vulnerability of classifiers to evolutionary generated adversarial examples. Neural Netw. **127**, 168–181 (2020)

Wang, Y., Xu, W.: Leveraging deep learning with lda-based text analytics to detect automobile insurance fraud. Decis. Support Syst. **105**, 87–95 (2018)

Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)

Wang, Y., Tan, Y.-A., Zhang, W., Zhao, Y., Kuang, X.: An adversarial attack on dnn-based black-box object detectors. J. Netw. Comput. Appl. **161**, 102634 (2020). https://doi.org/10.1016/j.jnca.2020.102634

Wei, X., Liang, S., Chen, N., Cao, X.: Transferable adversarial attacks for image and video object detection. arXiv preprint arXiv:1811.12641 (2018)

Wei, X., Liang, S., Chen, N., Cao, X.: Transferable Adversarial Attacks for Image and Video Object Detection. AAAI Press (2019)

Wu, B., Iandola, F., Jin, P.H., Keutzer, K.: Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 129–137 (2017)

Wu, Z., Lim, S.-N., Davis, L.S., Goldstein, T.: Making an invisibility cloak: real world adversarial attacks on object detectors. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16, pp. 1–17. Springer (2020)

Wu, C., Luo, W., Zhou, N., Xu, P., Zhu, T.: Genetic algorithm with multiple fitness functions for generating adversarial examples. In: 2021 IEEE Congress on evolutionary computation (CEC), pp. 1792–1799 (2021)

Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., Yuille, A.: Adversarial examples for semantic segmentation and object detection. In: Proceedings of the IEEE international conference on computer vision, pp. 1369–1378 (2017)

Xu, H., Ma, Y., Liu, H.-C., Deb, D., Liu, H., Tang, J.-L., Jain, A.K.: Adversarial attacks and defenses in images, graphs and text: a review. Int. J. Autom. Comput. **17**(2), 151–178 (2020)

Ye, J., Wang, Y., Zhang, X., Xu, L., Ni, R.: Adversarial attack algorithm for object detection based on improved differential evolution. In: 6th international workshop on advanced algorithms and control engineering (IWAACE 2022), vol. 12350, pp. 669–678. SPIE (2022)

Yuan, X., He, P., Zhu, Q., Li, X.: Adversarial examples: attacks and defenses for deep learning. IEEE Trans. Neural Netw. Learning Syst. **30**(9), 2805–2824 (2019)