Mike Papadakis
Silvia Regina Vergilio (Eds.)

LNCS 13711

Search-Based Software Engineering

14th International Symposium, SSBSE 2022
Singapore, November 17–18, 2022
Proceedings

Springer

Lecture Notes in Computer Science 13711

EvoAttack: An Evolutionary Search-Based Adversarial Attack for Object Detection Models

Kenneth Chan(✉) and Betty H. C. Cheng

Department of Computer Science and Engineering, Michigan State University, 428 S Shaw Ln, East Lansing, MI 48824, USA
{chanken1,chengb}@msu.edu

Abstract. State-of-the-art deep neural networks in image classification, recognition, and detection tasks are increasingly being used in a range of real-world applications. Applications include those that 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 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 unexplored. This paper explores 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 and Faster R-CNN). A proof-of-concept implementation is provided to demonstrate how evolutionary search can generate adversarial examples that existing models fail to correctly process. We applied our approach to benchmark datasets, Microsoft COCO and Waymo Open Dataset, applying minor perturbations to generate adversarial examples that prevented correct model detections and classifications on areas of interest.

Keywords: Evolutionary search · Adversarial examples · Machine learning

1 Introduction

Many popular machine learning techniques, such as Deep Neural Networks (DNNs), are susceptible to carefully crafted malicious inputs [3,17]. These malicious inputs are known as *adversarial examples* [17]. DNNs are artificial neural networks with multiple layers of activation neurons that can be used for feature learning. DNNs have numerous real-world applications, such as malicious file detection [19,21], fraud detection [11,24], and autonomous vehicles [6,26]. In safety-critical systems [5], commercially-deployed DNNs may have significant consequences should they fail, leading to potential injury, serious harm, death,

M. Papadakis and S. R. Vergilio (Eds.): SSBSE 2022, LNCS 13711, pp. 83–97, 2022.
https://doi.org/10.1007/978-3-031-21251-2_6

and/or financial loss. To prevent serious harm or injuries, DNNs deployed in safety-critical systems must be robust against adversarial attacks. Therefore, a challenge is how machine learning model robustness can be assessed and improved to correctly process inputs in the face of adversarial attacks. This paper introduces EvoAttack, a black-box evolutionary search-based technique, to assess the robustness of object detection algorithms against a diverse collection of adversarial examples.

Over the past decade, several research efforts have addressed adversarial examples for image classification techniques [3,13,17,18]. 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) [3,17]. 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, object detection techniques (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 label them accordingly) have had limited research [25,28,29]. Compared to image classification, attacking object detection techniques is significantly more difficult as the images are larger in size, contain more dimensions, and contain multiple numbers of potential objects. Existing techniques for generating adversarial examples against object detection algorithms [25,28] 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. 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. Furthermore, while black-box approaches have been applied to image classification [12,23], they typically introduce a large amount of visible perturbation when applied to images with large dimensions.

As a means to assess model robustness against adversarial attacks, this paper introduces a black-box evolutionary search-based testing technique, EvoAttack, to generate adversarial examples to compromise object detection algorithms by adversely impacting the detection of objects. The evolutionary search-based adversarial attack used in this work does not require access or estimates of hidden model parameters or model architecture, does not require additional training of surrogate models, and is model and data agnostic. This work contributes two key insights. First, we leverage the output of the object detection model in the previous generation to guide the mutation process and the structure of the fitness function. Second, we propose an adaptive mutation scheme that dynamically scales the mutation rate to reduce perturbation while promoting convergence.

Two key strategies are used to enable our approach to generate adversarial examples while minimizing human-perceptible perturbations. To generate adversarial examples, we use a generational Genetic Algorithm (GA), where individuals in a population evolve towards a global optimum (i.e., a perturbed

image that adversely impacts model 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, the output of the model in the previous generation enables our approach 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 detections. During mutation, we apply a fine-grained approach for generating perturbation by focusing on pixels in areas of interest. We introduce an adaptive mutation scheme, where we promote minor perturbations for each object in the image, while encouraging misdetection from the model. In the proposed adaptive mutation scheme, we mutate a small number of pixels when the generation count is low. In order to promote convergence, the number of modified pixels is dynamically scaled up as the number of generations increases.

In our experiments, we verify that EvoAttack can produce adversarial examples that prevent object detection. We implemented the proposed approach and generated adversarial examples against existing object detection models, such as RetinaNet [10] and Faster R-CNN [20]. To illustrate the potential impact of adversarial examples against object detection models, we apply our technique to attack a set of images obtained from the Microsoft COCO dataset [14] and the Waymo Open Dataset [16] to show how adversarial examples can be generated against two different benchmark datasets. Preliminary results show that our algorithm can cause the model to deviate from the expected output, while maintaining a low degree of visible perturbations (i.e., L0 and L2 norms). This work shows that black-box evolutionary search-based adversarial examples can be generated against object detection tasks, a domain not yet explored to the best of our knowledge. The remainder of this paper is organized as follows. Section 2 overviews background material and reviews related work. Next, Sect. 3 describes the details of the proposed approach. Section 4 describes the validation work of our approach. Finally, Sect. 5 concludes the paper and discusses future directions.

2 Background

This section provides background information for the paper. First, we describe adversarial examples. Next, we compare the image classification problem with the object detection problem. Finally, we overview related work.

2.1 Adversarial Examples

Adversarial examples describe machine learning model inputs that are maliciously perturbed to cause a failure in the model. Figure 1 shows an example of an adversarial example. The original input image (i.e., an image with the corresponding identified objects) is shown on the left. When the malicious perturbation noise (scaled by a factor of 10 for readability purposes) is added to the

input image, the resulting image prevents the detection of the objects. Adversarial examples closely resemble original images, and thus are not human distinguishable.

Different adversaries may have different types of 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 [29]. For example, the adversary may have information about the type of model, weights, gradient information, and/or the architecture of the model. Traditional attack methods such as L-BFGS [17], Fast Gradient Sign Method (FGSM) [3], and Dense Adversary Generation (DAG) [28] 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 [29]. 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. Thus, 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.

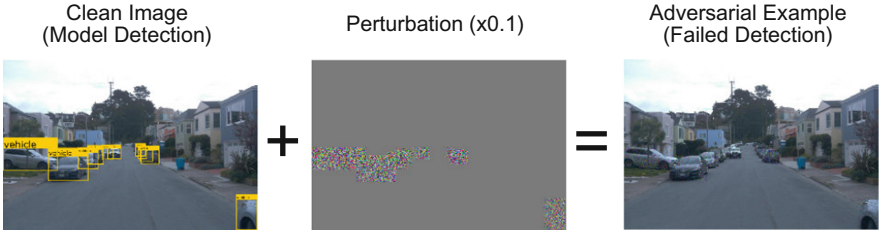


Fig. 1. Example of an adversarial example, where the original clean input with malicious perturbations prevents model detection.

2.2 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 [28]. In image classification, an input image consists of exactly one object. 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 are often large in dimensions with multiple objects of interest. 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. As such, alternative approaches for

selecting the perturbation space must be developed for attacking object detection algorithms.

2.3 Related Work

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

Szegedy et al. [17] introduced the first adversarial examples, revealing the existence of malicious images that machine learning models fail to predict correctly. Carlini and Wagner [1] introduced the C&W attack similar to that of Szegedy et al.'s attack [17]. Goodfellow et al. [3] 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. [22] proposed a one-pixel attack using Differential Evolution (DE). Alzantot et al. [12] proposed GenAttack, which applies a variation of GA to discover adversarial examples. Vidnerová et al. [23], Chen et al. [2], Wu et al. [27], and Han et al. [4] 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.

Finally, a number of research efforts have explored generating adversarial examples against object detection models. Xie et al. [28] 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 [20] on the PascalVOC dataset. In contrast, Wei et al. [25] proposed a transfer-based attack based on a Generative Adversarial Network (GAN). However, their approach requires the training of a surrogate model, thus resulting in additional training overhead. Furthermore, transferability attacks do not guarantee success. As such, current existing state-of-the-art techniques do not provide a true black-box, model- and data-agnostic approach for object detection algorithms.

3 Methodology

This section introduces our proposed evolutionary search-based approach to attack object detection algorithms. We first describe the image datasets used

in our experiments. Next, we overview how we use evolutionary search to generate adversarial examples with the objective of minimizing perturbations. Finally, we introduce an adaptive mutation scheme that reduces the number of changed pixels and the degree of perturbations in adversarial examples.

3.1 Object Detection Benchmark Datasets

This work uses two benchmark datasets to validate the proposed technique to illustrate that evolutionary search-based attacks are not model or dataset-dependent. First, the Common Object in Context (COCO) [14] 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 [16] 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.

3.2 Evolutionary Search-Based Approach

This section describes how we harness evolutionary search to generate adversarial examples against object detection algorithms. Figure 2 shows a Data Flow Diagram (DFD) for EvoAttack, where parallel lines represent external data stores, green circles denote process bubbles, and arrows indicate data flow between processes. The inputs for the algorithm are a (black-box) object detection model and an input image (i.e., the original, non-perturbed image). The algorithm searches for perturbations that adversely impact the model's ability to detect objects. In order to apply evolutionary search, potential solutions are mapped to a genome representation in Step 1. In our work, individuals (i.e., images) 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 (ranging from [0, 255]) for the i, j-th pixel, respectively.

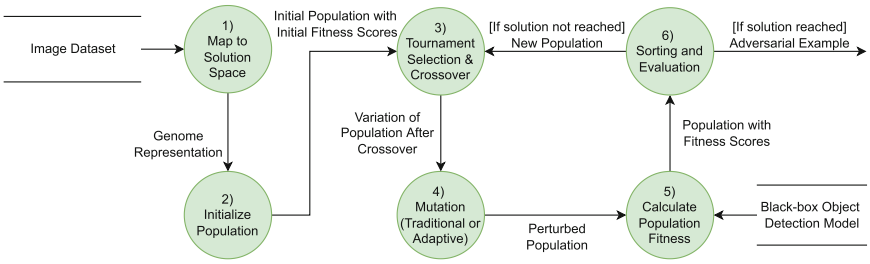


Fig. 2. DFD for the evolutionary process used to generate adversarial examples against object detection models.

We largely follow a standard generational GA process to identify adversarial examples. A point crossover operator is used in Step 3 to create children to maintain regions of perturbation that cause failures in the model's detection. Several

key innovations enable EvoAttack to optimize the evolutionary search process and reduce the perturbations in adversarial examples. First, EvoAttack uses an adaptive mutation scheme that enables the trade-off of minimizing visible perturbation with computational time to find adversarial examples (Sect. 3.3 describes the adaptive mutation scheme in detail). Second, we introduce an optimization strategy in Step 5, where the previous generation's model results are used to configure the structure of the fitness function and identify regions of interest to perturb. The fitness scores are calculated using the following expression:

$$\text{FitnessScore} = \sum_{i=0}^{\text{len(detection)}} \text{detection[i]['confidence']}$$

The fitness score represents the model's "degree of correctness" as the sum of the confidence scores for objects identified by the model. As an object is no longer detected by the model when its confidence score is reduced below the detection threshold, the fitness score promotes the evolutionary search to introduce perturbations that iteratively lowers detection confidences until the objects are no longer detected by the model.

Finally, the population is sorted by the fitness score in Step 6. If an adversarial example is found that prevents model detection, then the algorithm terminates and returns the adversarial example. Otherwise, the new population returns to Step 3 for the algorithm to iteratively add perturbations. If the maximum number of generations is reached without convergence, then the algorithm fails and is terminated.

3.3 Adaptive Mutation Scheme

The mutation operation in Step 4 introduces perturbations to an image with the objective of finding "ideal" perturbations (i.e., fewest number of changed pixels and smallest degree of changes) that hamper model detection. The traditional approach to the mutation scheme [12,23] in evolutionary search-based attacks is to modify each pixel with a small mutation rate, P_{mut}, during the mutation step of each generation. This approach can quickly generate adversarial examples that cause a failure in the model's detection ability, where the emphasis is on optimizing computational time. However, 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. For example, an object in a bounding box of dimension 160×200 (i.e., 10% of a 640×500 image) would mutate 320 pixels on average every generation using a small P_{mut} of 0.01. After 100 generations, every pixel is expected to be mutated at least once. As such, we found that this approach would introduce perturbations to the image that are too easily perceived by humans.

In order to address the high perturbation problem, we propose an adaptive mutation scheme for EvoAttack that begins by adding minimal perturbations and scales the added perturbations as the number of generations increases.

During the mutation step of each generation (i.e., Step 4), we introduce minor perturbations to pixels in bounding boxes identified by the model by adding perturbations sampled over $(\delta_{min}, \delta_{max})$ to n pixels for each object detected. The values $(\delta_{min}, \delta_{max})$ determine the degree of perturbation introduced. Higher δ values introduce perturbations that are more likely to cause misdetections, but are more likely to be human perceptible. The number of changed pixels n is determined by the following formula, where α, β, and P_{mut} are hyperparameters:

$$n = \begin{cases} max(\text{generation}/\alpha, 1) & \text{if generation} \leq \beta \\ \text{num\_of\_pixels} * P_{mut} * (\text{generation}/\beta) & \text{if generation} > \beta \end{cases}$$

Specifically, n is increased incrementally every α number of generations. The value α determines the rate of growth for the number of pixels perturbed. The value β defines the number of generations EvoAttack explores before the algorithm adopts a more aggressive search strategy to promote convergence. 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. For example, consider the object in an image discussed above with dimension 160×200 and $\alpha = 15, \beta = 750, P_{mut} = 0.01$. EvoAttack introduces perturbations to each detected object that incrementally increase by 1 every 15 generations (i.e., 1 pixel is changed from generations 1–15, 2 pixels are changed from generations 15–30, etc.) until the 750^{th} generation. By the 750^{th} generation, 320 pixels in the bounding box of each object are mutated. The algorithm then modifies a number of pixels based on a scaling mutation rate, P_{mut}. Using an adaptive mutation scheme, images that do not require large perturbations to cause a misdetection will not have unnecessary perturbations, while images that are difficult to perturb will still be promoted to converge as the number of generations increases.

4 Empirical Studies

We evaluate the efficacy of EvoAttack against state-of-the-art object detection algorithms. First, we demonstrate that the adaptive mutation scheme can generate more adverse test data (less perturbations) than the traditional mutation scheme (i.e., all pixels in bounding boxes are eligible for mutation with a fixed chance). Second, we show that our approach is model agnostic by attacking models of different architectures. Finally, we demonstrate the potential negative impacts of such attacks by attacking the Waymo Open Dataset [16], while also demonstrating that it is data agnostic.

4.1 Experimental Setup for Evolutionary Search-Based Approaches

For the validation work, we use object detection DNNs implemented using the Pytorch [15] deep learning research platform. To show that our attack is model

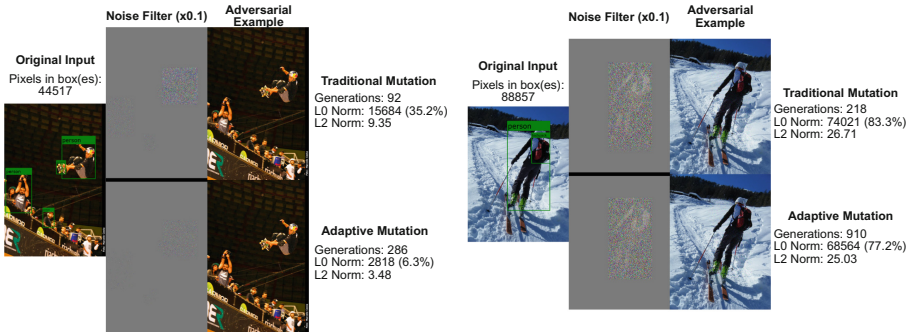
agnostic, we use various model architectures with weights pretrained by Pytorch, such as a Faster R-CNN MobileNetV3 [20] and RetinaNet [10] trained using a ResNet-50-FPN backbone. For Waymo images, we train an object detector using a RetinaNet with a ResNet-50-FPN backbone [8]. The trained model has a mean recall of 95%. During model inference, we set the detection confidence threshold to be 0.7 to provide a proof-of-concept demonstration of EvoAttack. However, EvoAttack can generate adversarial examples using any threshold score for model testing. For evolutionary parameters, we started with values used in state-of-the-art image classification attacks [12,23]. We experimentally fine-tuned these parameters for object detection. The maximum number of generations is set to 2000, with 16 individuals in each population. Finally, $P_{crossover}$ and 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. Finally, since EvoAttack is intended to be used as a testing technique before the model is deployed, we use the objective of minimizing perturbations as the primary metric to gauge attack efficiency in order to obtain higher quality adversarial examples (i.e., less perturbation) over quantity (i.e., time). Additionally, the number of generations cannot be used to adequately compare adversarial examples, as each algorithm adds a different amount of perturbation to the image per generation. All experiments are performed on a NVIDIA GeForce GTX 1080 GPU with an Intel Core i7-7700K CPU.

4.2 E1: Demonstration of the Adaptive Mutation Operator

In our first experiment, we demonstrate how adversarial examples can be generated against object detection algorithms using EvoAttack. Specifically, we illustrate the notable impact of EvoAttack's adaptive mutation scheme by including a comparison for adversarial examples generated with random search, EvoAttack using the traditional mutation scheme (denoted as EvoAttack.Trad), and EvoAttack using the adaptive mutation scheme (denoted as EvoAttack). During each mutation operation after crossover, we introduce minor perturbations to pixels in the bounding boxes identified by the object detection model in the previous generation. 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. For experiment E1, we used $\delta_{min} = 0.025$ and $\delta_{max} = 0.05$. Values for α and β are selected empirically based on from multiple runs of the experiment on the COCO dataset.

Figure 3 shows two adversarial examples generated over a randomly sampled set of input images obtained from the COCO testset against the RetinaNet model. The first image 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 read-

ability. The adversarial examples in the rightmost column of images prevent model detection, where the model failed to draw bounding boxes around the objects of interest. The adversarial examples shown on top and bottom are generated using EvoAttack.Trad and EvoAttack, respectively. The perturbation information shows the number of generations required for convergence, number of changed pixels (L0 norm), and degree of perturbations (L2 norm or Euclidean distance) of the adversarial examples. The numbers in parenthesis denote the percentage of pixels in the original bounding boxes that have been modified in the adversarial example. Figure 3a shows a scene at a skateboarding event. The model identified multiple objects of interest in the image and drew bounding boxes around them. Both mutation schemes generated an adversarial example that caused the model to fail to detect the objects. Compared to the adversarial example generated by EvoAttack.Trad, the adversarial example generated by EvoAttack contains significantly fewer perturbations. Since EvoAttack converged on an adversarial example before the β number of generations, the adversarial example is considered to be "easy to perturb". In contrast, Fig. 3b shows an input image of a skier. The adversarial examples generated also caused a failure in the model's detection. However, compared to Fig. 3a, this input image required more than β number of generations to converge, thus EvoAttack only slightly outperforms EvoAttack.Trad in minimizing perturbations.



(a) Example of an easy to perturb image (b) Example of a difficult to perturb image

Fig. 3. Comparisons of adversarial examples generated using EvoAttack and EvoAttack.Trad. The original predicted inputs, noise filters, adversarial examples, and perturbation information are shown for each image.

Table 1 shows the average number of changed pixels and degree of perturbations for adversarial examples generated against thirty randomly sampled input images. Using traditional mutation and adaptive mutation, EvoAttack performs better than random search. In Table 1, the percentage of pixels in bounding boxes changed metric for random search exceeds 100%, since the algorithm perturbs a random number of pixels in the entire image (i.e., modifies pixels beyond the bounding boxes and does not localize the perturbation). With the

adaptive mutation scheme, EvoAttack has fewer perturbations when compared to EvoAttack.Trad that uses traditional mutation. Specifically, 26 adversarial examples generated with EvoAttack have fewer number of changed pixels and 19 adversarial examples generated have less degree of perturbations when compared to EvoAttack.Trad. The average number of changed pixels is reduced by 31.41% and the average degree of perturbations is reduced by 21.49% using EvoAttack. The columns labeled "*Easy to perturb inputs*" show the measured metrics for adversarial examples that are perturbed before β number of generations in EvoAttack. The metrics for the same set of images for EvoAttack.Trad are also provided for comparison. The results indicate that EvoAttack is able to find adversarial examples with significantly less perturbations for objects that are easy to perturb, with 81.96% reduced number of changed pixels and 64.67% reduced degree of perturbations on average. The results of this experiment show that EvoAttack is able to generate adversarial examples with low degree of perturbations and few number of pixel changes.

Table 1. Comparison of perturbations measured for adversarial examples generated over thirty input images against a RetinaNet model. The easy to perturb inputs consist of images that converged before β number of generations in EvoAttack. The metrics for the same set of images from an evolutionary search using the traditional mutation scheme are provided for comparison.

Avg. Statistic	**All Inputs**			**Easy to Perturb Inputs**	
Num of objects in input	2.73			2.55	
Num of pixels in bnd. boxes	114,670			92,332	
Total number of images	30			20	
Avg. Values (per image)	**Random**	**EvoAttack (Trad.)**	**EvoAttack (Adapt.)**	**EvoAttack (Trad.)**	**EvoAttack (Adapt.)**
Num of generations	71.37	231.07	625.50	130.50	376.90
Num of changed pixels	277,955.94	59,430.57	40,762.67	33,916.95	6,119.35
% of pixels in bnd. boxes changed	242.37%	51.83%	35.55%	36.73%	6.63%
Degree of perturbations	337.33	22.33	17.53	14.04	4.96
Computational time (sec)	813.11	677.91	1,496.31	329.19	768.47

4.3 E2: Demonstration that EvoAttack is Model Agnostic

In this experiment, we demonstrate that EvoAttack is model agnostic by attacking a model of different architecture. Specifically, we apply EvoAttack to a Faster R-CNN MobileNetV3 [20] model and show that we can produce comparable results as an attack against the RetinaNet [10]. Against the same set of thirty input images, our approach is able to reduce all detected objects below the detection threshold. Table 2 shows the average perturbations of adversarial examples generated against the Faster R-CNN model.

Compared to the RetinaNet model, the Faster R-CNN model is more robust on average against EvoAttack, as it requires more perturbations to prevent

model detection. In our studies, we found that the Faster R-CNN model requires almost twice the number of changed pixels and degree of perturbations when compared to the RetinaNet model. Furthermore, the number of adversarial examples that were generated before the β number of generations reduced significantly from 20 to 6 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.

Table 2. Comparison of perturbations measured for adversarial examples generated over thirty input images against a Faster R-CNN model.

Avg. Statistic	**All Inputs**			**Easy to Perturb Inputs**	
Num of objects in input	2.93			1.67	
Num of pixels in bnd. boxes	146,338			58,428	
Total number of images	30			6	
Avg. Values (per image)	**Random**	**EvoAttack (Trad.)**	**EvoAttack (Adapt.)**	**EvoAttack (Trad.)**	**EvoAttack (Adapt.)**
Num of generations	89.77	509.33	988.73	80.00	435.67
Num of changed pixels	280,679.41	110,466.38	101,815.88	27,136.17	6,410.50
% of pixels in bnd. box changed	191.8%	75.49%	69.58%	46.44%	10.97%
Degree of perturbations	414.45	41.06	39.91	12.36	5.29
Computational time (sec)	862.85	1,008.77	1,145.01	90.96	181.13

4.4 E3: Demonstration that EvoAttack is Data Agnostic

The purpose of this experiment is to demonstrate that EvoAttack is data agnostic 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 [8] trained over the Waymo Open Dataset [16]. The trained model predicts vehicles, pedestrians, and cyclists for a camera mounted atop a vehicle. Thus, if an adversary successfully prevents correct model detection, the resulting behavior of the system may lead to significant consequences such as serious injuries or even deaths. We apply EvoAttack to thirty randomly chosen images sampled over the Waymo Open Dataset. Figure 4 shows several adversarial examples. This result shows that our approach successfully introduced perturbations such that the model fails to draw correct bounding boxes around vehicles, pedestrians, and cyclists in the images. We also show the potential impact of black-box adversarial attacks on real-world safety-critical systems. If an object detection model used in an autonomous vehicle is compromised using such attacks, then the behavior of the vehicle may result in a collision with surrounding vehicles or people.

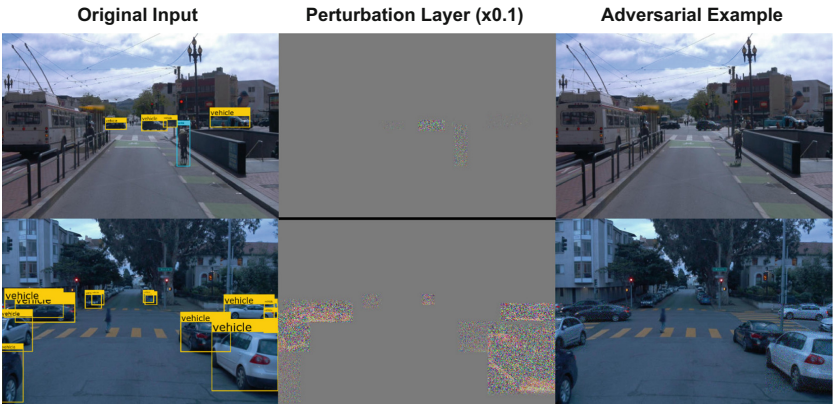


Fig. 4. Adversarial examples generated against the Waymo Open Dataset [16].

4.5 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. 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.

5 Conclusion

This paper introduced 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 an adaptive 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 various datasets and models of various architectures.

Future research will explore potential improvements to our evolutionary search-based attack and explore detection and mitigation strategies. They include potential improvements using multi-objective GAs (e.g., NSGA-II) and parallel GAs. Various 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 explored. Furthermore, novelty search [7,9] may be leveraged to discover a collection of adversarial examples that causes diverse behaviors in the model. Finally, Enlil [8] (i.e., behavior oracles) may be used to predict the uncertain behavior of the object detection model when exposed to adversarial examples.

References

1. 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)
2. 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)
3. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (2015)
4. 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 2019, pp. 2593–2595. Association for Computing Machinery, New York (2019)
5. 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)
6. 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)
7. 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)
8. 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)
9. Lehman, J., Stanley, K.O.: Abandoning objectives: evolution through the search for novelty alone. Evol. Comput. **19**(2), 189–223 (2011)
10. 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)
11. 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.) SAI 2020. AISC, vol. 1229, pp. 301–313. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-52246-9_21

12. Alzantot, M., et al.: GenAttack: practical black-box attacks with gradient-free optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1111–1119 (2019)
13. Eykholt, K., 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)
14. Lin, T.Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
15. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32, pp. 8024–8035. Curran Associates, Inc. (2019)
16. Sun, P., 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)
17. Szegedy, C., et al.: Intriguing properties of neural networks. In: International Conference on Learning Representations (2014)
18. 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)
19. 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). https://doi.org/10.1007/s13198-021-01168-x
20. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, vol. 28 (2015)
21. 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)
22. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Trans. Evol. Comput. **23**(5), 828–841 (2019)
23. Vidnerová, P., Neruda, R.: Vulnerability of classifiers to evolutionary generated adversarial examples. Neural Netw. **127**, 168–181 (2020)
24. Wang, Y., Xu, W.: Leveraging deep learning with LDA-based text analytics to detect automobile insurance fraud. Decis. Support Syst. **105**, 87–95 (2018)
25. Wei, X., Liang, S., Chen, N., Cao, X.: Transferable adversarial attacks for image and video object detection. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019, pp. 954–960. AAAI Press (2019)
26. 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)
27. 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)
28. 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)
29. Yuan, X., He, P., Zhu, Q., Li, X.: Adversarial examples: attacks and defenses for deep learning. IEEE Trans. Neural Netw. Learn. Syst. **30**(9), 2805–2824 (2019)