ASSESSING ROBUSTNESS OF AI-BASED SYSTEMS IN THE FACE OF HUMAN-BASED EXPLOITATIVE UNCERTAINTY

By

Kenneth H. Chan

AN ABSTRACT OF A THESIS PROPOSAL

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science and Engineering—Doctor of Philosophy

2025

Advisor: Dr. Betty H.C. Cheng

ABSTRACT

As research continues to expand and improve the capabilities of artificial intelligence, their integration in safety-critical learning-enabled systems is increasingly ubiquitous. Machine learning models extract patterns directly from large collections of data, reducing human developer burden by automating intensive tasks. The ability and performance of these machine learning models are closely tied to the completeness of the training data, which often include only the expected operating contexts due to the sparseness of edge cases. However, systems deployed in the real-world often involve operating contexts that include humans. Human-involved operating contexts introduce a large source of uncertainty due to the inadequate coverage of relevant training data that comprise unpredictable, erratic, and even malicious human behaviors. Yet, when deployed in safety-critical settings, these systems must demonstrate safe behaviors and mitigate potential damages in the face of uncertainty in addition to satisfying operational requirements. This doctoral work proposes automated testing methods that address the robustness of learning-enabled systems against a spectrum of human-based exploitative uncertainties. First, we define the spectrum of exploitative uncertainty based on the intentions of the external human agent(s). Second, we attack the problem from several different angles using a multipronged approach, integrating and synthesizing a number of distinct and disparate techniques to assess and improve the robustness of the system under study. We demonstrate the capabilities of our approaches by applying the proposed research to various state-of-the-art, machine-learning-based, and safety-critical cyber physical systems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Recent advances in Artificial Intelligence (AI) have promoted their use in many applications to automate previously human-intensive and tedious tasks with large volumes of data. However, a key challenge for these systems is the inability to include complete data that captures all operating contexts that the system may encounter during its lifetime, which can cause the AI to produce incorrect or unintended results. The lack of sufficient training data raises concerns regarding the ability of the system to provide acceptable and safe behavior when facing operating contexts to which it has not previously been exposed to. We define a Learning-Enabled Component (LEC) as a software system component whose behavior is constructed and optimized based on data and training experiences (e.g., machine learning-based object detectors, image classifiers, speech recognizers, etc.). Subsequently, a Learning-Enabled System (LES) is an AI-based software system that involves one or more LEC(s). When LESs are deployed in a safety-critical setting [5] (e.g., Autonomous Vehicles (AVs), medical devices, cyber physical systems, etc. [6, 7]), unexpected failures can lead to injury to humans, loss of life, damage to the operating environment, damage to the LES, and financial loss. As a result, LESs must demonstrate correct and safe behaviors (i.e., *robustness*) in the presence of *uncertainty* in addition to satisfying operational constraints. Uncertainty describes elements that may affect the correctness of the LES(s), including those that originated from input noise, environmental conditions, interaction with humans, etc. This work synthesizes a number of multidisciplinary strategies to propose techniques and frameworks to address the assurance of LESs in the context of human-based uncertainty. Specifically, we are assessing the robustness of the LES to deliver acceptable behavior even when operating under uncertainty.

In order to provide stakeholders with confidence that the LES will deliver safe behavior at all times, developers must demonstrate that they exhibit acceptable behavior when operating in the presence of uncertainty. However, significant costs and potentially safety concerns limit the feasibility of run-time field-testing of LESs (i.e., testing the LES in deployment) [8]. As such, additional techniques are needed during design time to assess the assurance of LESs to

proactively discover edge cases that may lead to undesirable behaviors at run time. While a number of existing approaches have addressed the testing and assurance of LESs in expected operating contexts [9, 10, 11, 12], recent ongoing research efforts are investigating their assurance in the face of uncertainty [11, 13, 14, 15, 16]. However, these approaches tend to focus on environmental uncertainty (e.g., weather, road conditions, etc.). Consequently, additional techniques that address human-based uncertainty are essential to ensure that LESs will respond appropriately to undesirable situations that arise from interactions with humans in the environment.

This document proposes a multipronged approach to assess the robustness of an LES when exposed to different types of human-based uncertainty. We propose to explore the automated testing of an LES for a spectrum of human-based uncertainty, ranging from malicious attacks to non-malicious, but exploitative[1] interactions that lead to unintended behaviors in the system under study. We address human-based uncertainty at multiple levels of abstraction, including at the component level (e.g., testing the LEC) and at the system level (e.g., testing the LES as a whole). Furthermore, we propose a multidisciplinary approach that leverages technologies from different fields of study to support automated approaches to discover the impact of human-based uncertainties, which can be used to assess and improve the robustness of LECs and LESs. This dissertation proposal also includes our preliminary investigations that demonstrate the efficacy of the proposed approach.

1.1 Problem Description

When deployed in safety-critical systems, LESs must operate safely and correctly in the face of uncertainties. These uncertainties are caused by a lack of complete information in the operating context of the LES in the presence of humans, environmental factors, and other previously unseen elements that challenge the assurance of the system. These challenges are exacerbated by the fact that existing AI systems may be confidently incorrect in their response, or "hallucinate" fabricated information. This work proposes to address uncertainties that arise from humans in the operating context, as the failure of the LES in these contexts (e.g., AVs, drones, large industrial robotic arms,

[1] We use the term *exploitative* to describe human intentions that take advantage of one or more conditions, which may lead to the violation of safety-related properties of the LES.

etc.) may lead to injury and fatality in surrounding personnel. This challenge is amplified as humans often exhibit unpredictable, selfish, or even malicious intentions, all of which must be correctly identified, assessed, and/or mitigated by the LES in order to deliver acceptable (safe) behavior.

Existing techniques have focused on testing the ability of LESs to operate correctly in expected operating contexts and well-defined uncertainties (e.g., environmental, data, etc.) [11, 13, 14, 15, 16]. However, limited work has addressed human-based uncertainty. One (brute force) approach to guarantee operational correctness and safety is to test the LES under study on all possible combinations of uncertainty and human interactions that it may encounter during its lifetime. However, complete identification of all scenarios in which an LES may operate is mathematically infeasible, both in terms of time and computational resource constraints [17]. It is equally challenging for developers to anticipate the types of interactions that may arise from human-based uncertainty and avoid testing bias. Thus, new research techniques for the automated assessment of LESs by intelligently and efficiently exploring and discovering edge cases for human uncertainty are needed.

This work proposes strategies and techniques to address the assurance of *exploitative* human-based uncertainty. We define *exploitative uncertainty* as uncertainty due to a spectrum of human intentions that lead to the compromise of the assurance vulnerability in the system (i.e., safety-related properties of the LES). The spectrum captures a range of human intentions that either directly or indirectly (results in system behaviors that) exploits assurance vulnerabilities. Figure 1.1 shows an overview of the spectrum of exploitative uncertainty described in this dissertation proposal. The left side of the uncertainty spectrum (red) describes malicious or intentional exploitations, where a human agent may deliberately and maliciously compromise the system through security attacks or exhibit intentional behavior to cause damage to the LES. The middle of the uncertainty spectrum (purple) describes selfish exploitations, where a human agent acting in their own interests leads to undesirable behaviors in the system (e.g., an aggressive driver exploits the gaps between the leading vehicle and an AV in order to merge swiftly). Finally, the right side of the uncertainty spectrum

(blue) describes incidental exploitations, where a human agent's non-malicious actions may lead to degraded performance in the system or unintentionally take advantage of an assurance vulnerability (e.g., an AV crashing as it tries to avoid the erratic driving pattern of a nearby distracted driver).

Figure 1.1 Spectrum of exploitative uncertainty, based on the motivations and intentions of the human agent in the operating context.

1.2 Thesis Statement

This research is intended to explore a multipronged approach to assess and discover test cases for LESs involving human-based uncertainty. In particular, a number of disparate techniques from several distinct disciplines are synthesized to assess (and improve) the robustness of LESs based on intentional (i.e., malicious) and non-intentional (e.g., side effects of an LES's interaction with humans) human-based uncertainty.

Thesis Statement. *Search-based methods can serve as a foundation and be synergistically integrated with other techniques to assess and facilitate the improvement of the robustness of AI-based systems when exposed to exploitative-based uncertainty from external human agents.*

1.3 Anticipated Research Contributions

The proposed dissertation research is guided by three main objectives. First, we intend to provide techniques to assess human-based uncertainty that can discover edge cases that may not be discovered using traditional techniques. Second, we intend to maximize the use of automated testing techniques to assure safety and reduce developer testing bias. Third, we intend to focus on systematic and diversity-driven techniques to discover a wide spectrum of non-intuitive test cases. To achieve these objectives, the proposed techniques harness the capabilities of Evolutionary Computing (EC), goal modeling, game theory, and Reinforcement Learning (RL) to support the automatic testing and discovery of human-based LES uncertainty.

Our research contributions to date based on preliminary investigations are as follows.

1. EvoAttack: An automated method to discover malicious and human-imperceptible noise that can hinder the ability of an LEC to correctly identify objects in an image [18, 19]. This method produces *adversarial examples* (i.e., expected input images with malicious perturbations added) with minimal perturbations that lead to the complete suppression of the object detection capability in state-of-the-art object detection models.
2. Expound: A two-step automated method to discover diverse adversarial examples [20]. First, we discover a number of incorrect labels for adversarial examples generated on a given input image. Second, for each incorrect label, we discover a set of diverse and distinct perturbations that lead to the same incorrect label. This method adopts the exploration and exploitation paradigm used in search optimization to discover diverse solutions.
3. SafeDriveRL: An automated top-down approach that combines non-cooperative game theory and RL to discover unexpected interactions between an AV (i.e., LES under study) and human-operated vehicles (that exhibit one or more human driving styles) [21].

We present **E**xploitative **U**ncertainty **R**obustness fram**E**wor**K** for **A**ssessment (EUREKA), a risk assessment-inspired framework to categorize exploitative uncertainty and support different automated testing techniques to address different types of exploitative uncertainty. For each uncertainty source, we identify the following: 1) the asset to be protected (e.g., the LEC or the LES), 2) the threat (e.g., a type of human uncertainty from the exploitative spectrum), and 3) the type of assessment and robustness strategy for the asset with respect to the identified threat. Figure 1.2 shows the high-level overview of our multipronged approach to address human-based uncertainty. The vertical axis shows the type of human-based exploitative uncertainty addressed. They include a spectrum of inputs, ranging from malicious and intentional attacks that negatively impact the behavior of the system to incidental or non-malicious behaviors that negatively impact the system. The horizontal axis shows the category of test data produced. They range from a single targeted test case to a diversity-driven archive of test cases. Finally, yellow regions show techniques that address automated assessment at an LEC level (i.e., testing the AI-based component) while

green regions show automated assessment at an LES level (i.e., testing the system as a whole).

We next describe our preliminary investigations. First, EvoAttack [19] addresses malicious adversarial noise that may be used to hinder the perception capabilities of our system at the LEC level (e.g., the perception unit). This testing approach generates a single test case per execution, leading to human-imperceptible noise that prevents the correct identification of any object(s) in a given image. Second, Expound [20] also addresses malicious adversarial noise, but abandons the objective of minimal perturbation in favor of diversity at the LEC level. While the adversarial perturbation is still human-imperceptible, this testing approach focuses on generating diverse perturbations on two different dimensions. Specifically, the proposed technique leverages the exploration and exploitation paradigm commonly used in search-based testing [22]. Expound first discovers a number of diverse incorrect labels that the LEC may be susceptible to for a given image. Then for each label identified, Expound discovers a diverse collection of noise, all of which leads to the given same misbehavior in the system. Third, SafeDriveRL [21] focuses on incidental or non-malicious human-based uncertainty at the LES level. This testing technique identifies how selfish behaviors of human agents sharing an operating context may negatively impact the safety of the LES under study. Lastly, we plan to explore as remaining work techniques that may discover test cases involving incidental uncertainty from humans in the operating context, with diversity as the objective. We also plan to explore different model-based approaches to support systematic assessment of the impact of exploitative uncertainty.

1.4 Organization of Dissertation

The remainder of this dissertation proposal is organized as follows. Chapter 2 provides background information and summarizes key technologies used in this work. Chapter 3 describes EvoAttack, an approach to suppress the ability of object detection algorithms from correctly identifying objects in an image. Chapter 4 describes Expound, an approach to discover the full suite of failures in an LEC, each of which contains different faults that lead to the same failure. Chapter 5 describes SafeDriveRL, a game-based testing approach to identify test cases that lead to previously unseen responses in an LES when exposed to different types of external human driving

patterns. Chapter 6 summarizes our preliminary work and proposes the remaining investigations.

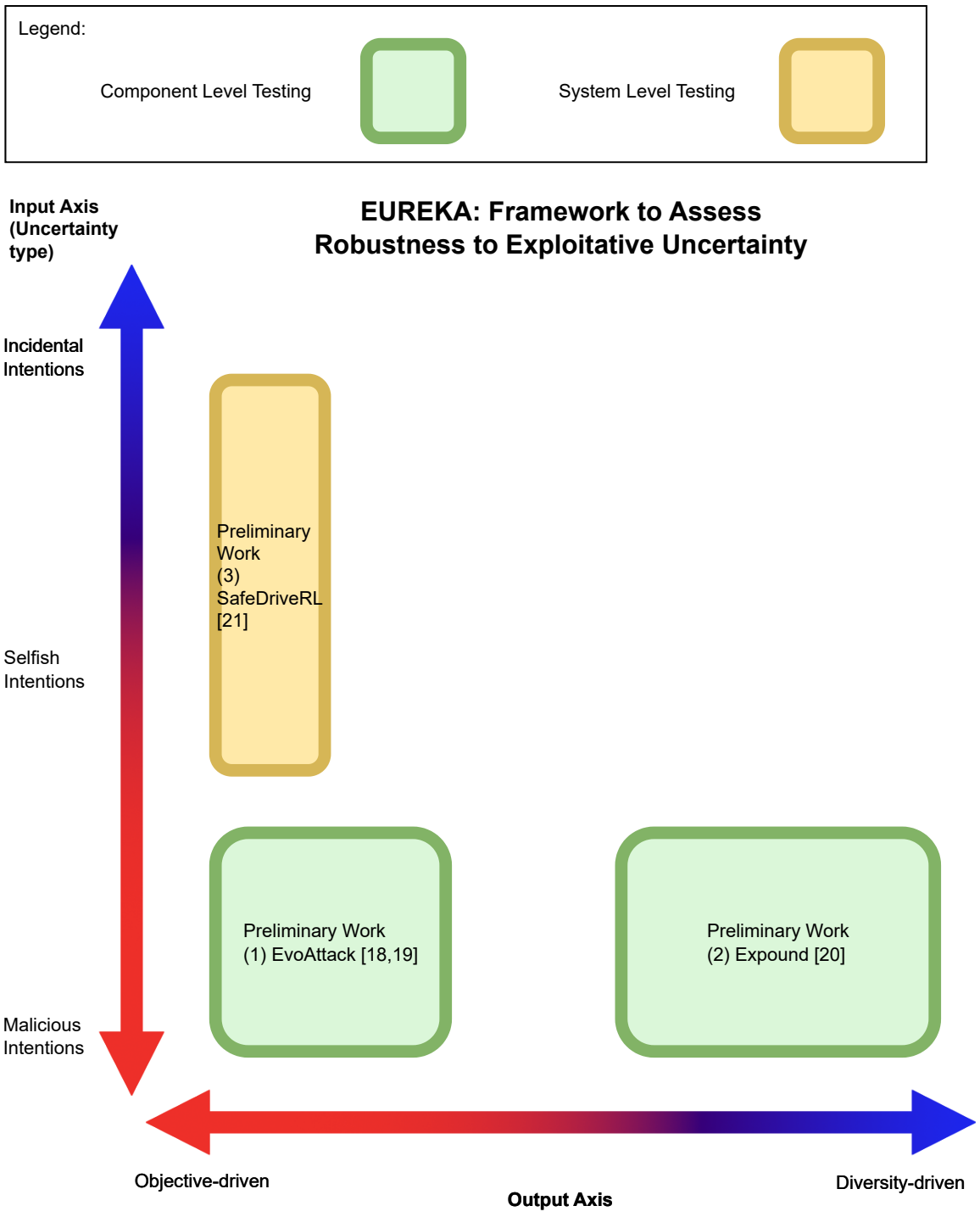


Figure 1.2 The EUREKA framework for addressing human-induced uncertainty. This figure shows the testing techniques along two different axes. The vertical axis shows the type of human-based uncertainty as an input, ranging from malicious, selfish, to incidental. The horizontal axis shows the type of test data produced. They range from a single targeted and objective-driven test case to an archive of diversity-driven test cases.

CHAPTER 2

BACKGROUND

This chapter provides background information and related technologies used in this dissertation proposal. Section 2.1 describes Deep Neural Networks (DNNs), challenges inherent to their use, and adversarial attacks on DNNs. Section 2.2 describes two search-based technologies used in this work: Genetic Algorithm (GA) and novelty search. Section 2.3 describes elements of game-based testing, including game theory and RL.

2.1 Deep Neural Network

This section describes DNNs, the fundamental statistical machine learning models behind modern AI-based systems such as AVs or language models. DNNs are artificial neural networks that are inspired by the human brain. They are capable of automatically discovering patterns that separate data in a dataset into multiple classes. DNNs distinguish between different classes by learning *decision boundaries* that separate features between classes. This dissertation proposal uses two popular machine learning tasks as proof-of-concept testbeds. First, *image classification* involves identifying the type of a single object in a given image. Second, *object detection* involves identifying up to n number of objects in a given image, and then correctly identifying the types of all objects in the image.

2.1.1 Elements of Learning-enabled Components

DNNs are used in many applications due to their ability to extract patterns directly from sufficient training data. Specifically, the universal approximation theorem [23, 24, 25] states that for any continuous mathematical function, there exists a DNN network that can approximate the function. As most problems and tasks can be represented by mathematical functions, such as identifying decision boundaries between classes in a dataset, there theoretically exists an instance of a neural network that can approximate any given function and solve any given task. By iteratively training over a large collection of data and tuning the structure or weights of a DNN, they can extract the features (or functions) which separate classes in a dataset directly.

DNNs are often represented as input transformers. Figure 2.1 shows a high-level illustration

of a DNN model, where circles denote neurons and edges denote connections between neurons. Fundamentally, DNNs consist of connected hidden layers, each containing a number of stacked neurons [23, 26]. For a given data point (x_i) comprising any number of input features, the DNN maps it to a corresponding output label (y_i). *Input features* can contain any numerical property, such as pixel values, values for historical weather data, onboard vehicle sensor values, etc. In contrast, *output labels* are mathematical values that represent information extracted by the DNN, such as object classification type, number of objects, actuator inputs, etc. In order to calculate the output label, the input features pass through a series of *hidden layers*, each applying a number of mathematical operations (based on the weights $(\mathbf{W})$ and biases $(\mathbf{b})$ associated with the nodes) on the values passed from the previous layer. An *activation function* determines the information to be transmitted to the next layer, filtering or amplifying the information collected. The weights and biases of a DNN are automatically calculated during the training phase of a DNN. During each *epoch* of training, a *loss function* evaluates the error between the DNN's current output and the ground truth (the correct output). *Gradient descent* is used to determine the optimal adjustments to the weights and biases to improve the model's performance, then *back propagation* updates the weights and biases of the model.

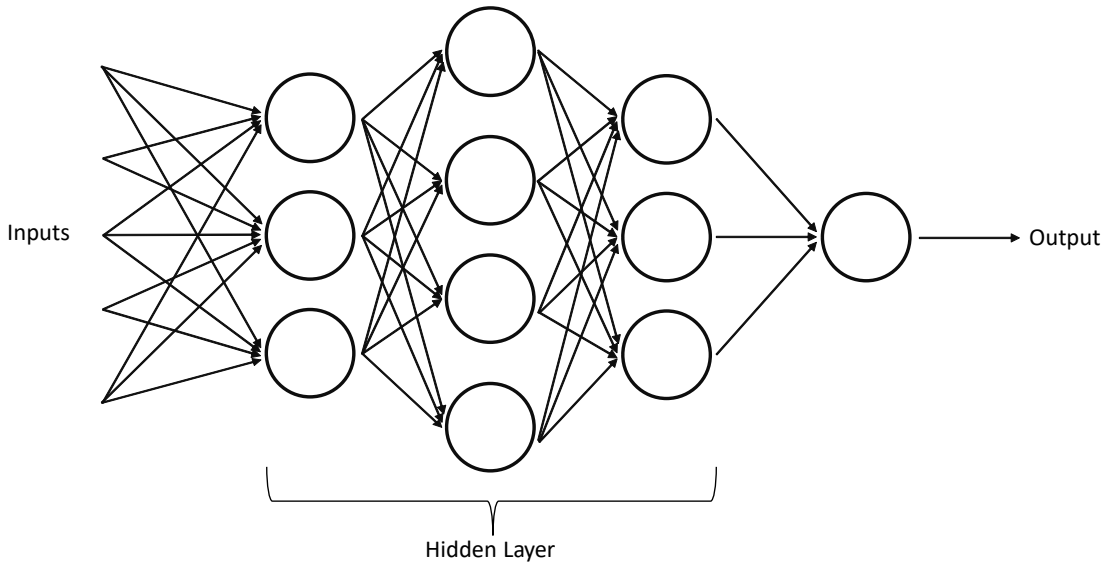


Figure 2.1 High-level illustration of a DNN model. Circles denote neurons, while edges denote the flow of data between neurons. The DNN comprises an input layer, multiple hidden layers of neurons, and an output layer.

DNNs have shown major progress in recent years. While they have demonstrated significant advantages and diverse use cases, they also pose unique challenges. While they enable traditionally difficult problems for software engineering (e.g., teaching a software to classify between different animals) and remove development overhead, the features and information used to infer those relationships cannot be controlled or tuned by a developer directly. In addition, the lack of code logic and the representation of information in numeric weights and biases introduce additional challenges with respect to the interpretability and transparency of the model's decisions [27].

2.1.2 Challenges in Uncertainty for Learning-enabled Systems

Existing research has demonstrated that DNNs can be applied to a wide range of disciplines (e.g., medical applications, drone control, autonomous driving, robotic controllers, etc.) with impressive performance in controlled settings. However, when deployed in settings which involve previously unseen context or environmental conditions that may introduce noise to the input, the ability of the LEC to operate correctly as expected is uncertain. Uncertainties can be described as epistemic or aleatoric [28, 29]. Epistemic uncertainty, the focus of this dissertation proposal, describes uncertainties that arise from incomplete information. For example, a dataset collected during sunny days may lack data samples from rainy conditions, and driving data collected may not include aggressive or severely distracted drivers (due to the infrequency of the drivers). As such, the correct behavior of the LEC in the face of the uncertainty is not guaranteed. Aleatoric uncertainty describes uncertainties that arise due to inherent randomness, such as input noise or measurement errors. While aleatoric uncertainty cannot be reduced, epistemic uncertainty can often be addressed by gaining more knowledge, such as discovering edge cases and including them as additional training data.

2.1.3 Adversarial Examples

Despite their technical promises, DNNs face a number of challenges with respect to their assurance and robustness. Szegedy *et al.* discovered that DNNs are sensitive to minor perturbations (i.e., noise), deceiving DNN models to produce an incorrect output on expected input data. These inputs are known as *adversarial examples*. Adversarial examples pose a large challenge for DNNs,

as the original inputs fall within the expected input distribution and the perturbation is often human-imperceptible. As such, a developer or human supervisor may not be able to distinguish whether a given input is malicious or not.

Figure 2.2 shows an example of an adversarial example generated against an object detection algorithm on an image extracted from the Waymo Open Dataset [2]. 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 [30, 31]. 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 [30, 32].

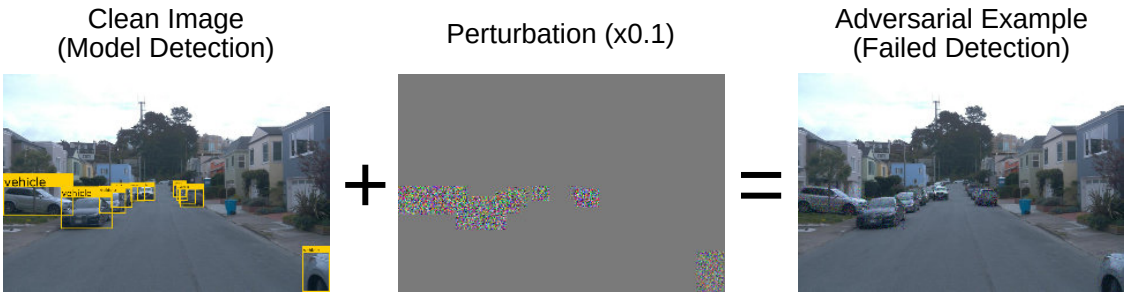


Figure 2.2 Example of an adversarial example, where the original clean input image from the Waymo Open Dataset [2] with malicious perturbations prevents object detection.

2.1.3.1 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 [33]. 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 (2.1), 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., $F_{img}(x + \delta) \neq y$).

$$\delta_{min} = \underset{\delta}{\operatorname{argmin}} ||\delta||_p \text{ s.t. } F_{img}(x + \delta) \neq y. \tag{2.1}$$

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 (2.2). 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 $\hat{y}$.

$$\delta_{min} = \underset{\delta}{\operatorname{argmin}} ||\delta||_p \text{ s.t. } F_{img}(x + \delta) = \hat{y}. \tag{2.2}$$

2.1.3.2 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 [33]. 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 [34], Fast Gra-

dient Sign Method (FGSM) [35], and Dense Adversary Generation (DAG) [36] 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 [33]. 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.

2.2 Search-based Software Engineering

This section overviews the search-based engineering approaches used for test case generation in this work.

2.2.1 Genetic Algorithm

GA is a type of evolutionary search technique that is inspired by Darwinian evolution and natural selection, used to solve optimization problems. Evolutionary search techniques are efficient solvers for large and complex search spaces, discovering solutions through an iterative process where a population of individuals compete to reproduce and evolve together. To apply evolutionary search, a data structure is defined to contain the information that may be used to construct potential solutions (e.g., array, matrix, etc.), known as the *genome*. A *population* of individuals is then initialized randomly. Next, the population undergoes the evolution cycle until the maximum number of generations is reached, executing the following operations.

[Crossover] - New offspring are generated in GAs by combining genetic information from existing individuals in the population using a crossover operator. Figure 2.3 shows an example of a single point crossover operator. The single point crossover operator creates offspring by selecting a random point within two individuals and swapping the contents of both parents' genes from the selected point to the end of the gene. The result is two offspring, each inheriting partial information from both parents.

[Mutation] - After offspring are created, they each have a probability to undergo mutation. Figure 2.4 shows an example of the mutation operator. The mutation operator modifies any units of genetic information in the offspring. This process often adjusts the value of a number of genes by increasing or decreasing them at a fixed rate. The mutation operator introduces diversity into the population, allowing the population to potentially escape local optimum.

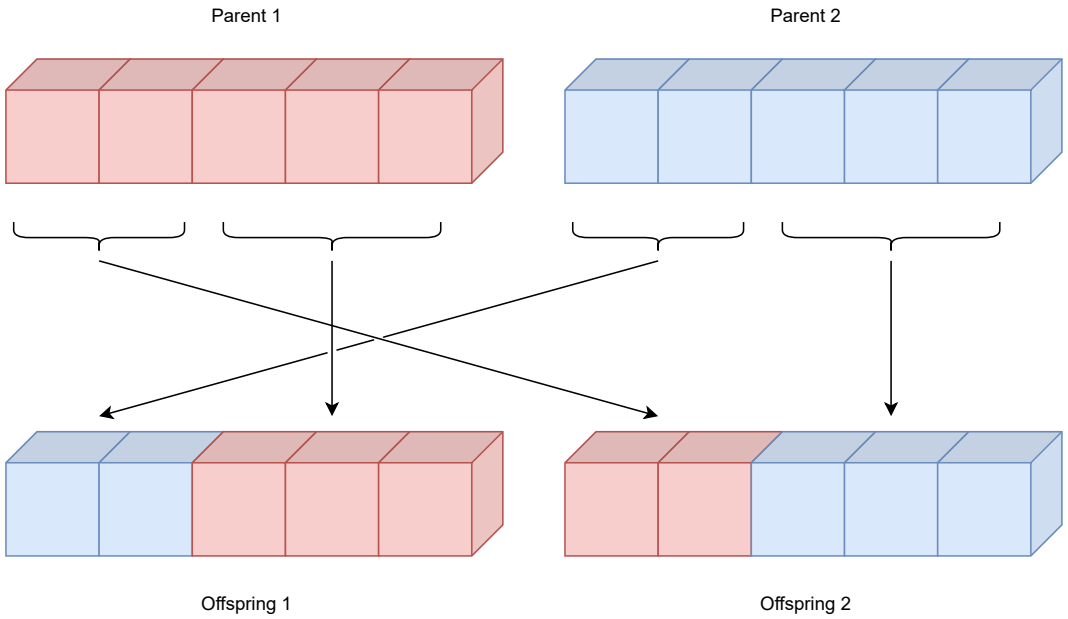


Figure 2.3 A single point crossover operator. Genetic information from both parents (red and blue) are used to form new offspring (shared colors).

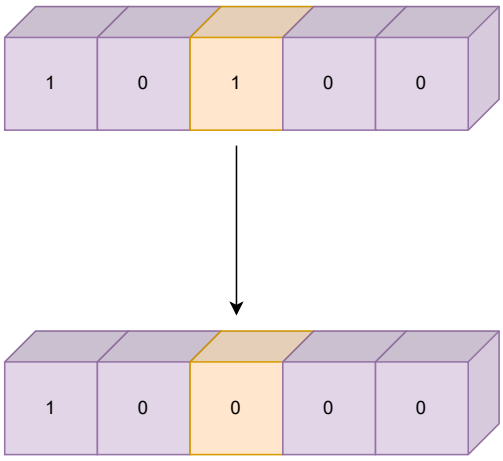


Figure 2.4 A single point mutation operator. Genetic information is randomly changed (the peach-colored cell that mutated from 1 to 0 in this instance) to introduce diversity to the population.

[Evaluation] - This step evaluates the fitness score (i.e., performance) of individuals in the population. Individuals are often executed in a simulator to observe their *phenome*, assessing how well the behavior of the genome performs for a given task. Higher fitness scores are assigned to individuals that perform the given task well (i.e., individuals that are closer to the optimum). The best-performing individuals then continue to the next generation and create offspring, iteratively moving the population towards the optimum.

By leveraging selective pressure and the stochastic properties of the evolutionary operators, evolutionary search discovers optimum solutions in a vast solution space. In the context of software testing, GAs are commonly used to evolve test cases that satisfy developer-specified criteria [37].

2.2.2 Novelty Search

Novelty search [38] is a special case of evolutionary search that optimizes for population diversity rather than discovering an individual solution. Novelty search follows a similar evolutionary loop as GAs. However, they abandon the notion of objective fitness in favor of population diversity [39]. Novelty search maintains an archive of the most diverse individuals during the search. During the evaluation stage of each generation, the fitness score for each individual is assessed based on the average distance (Euclidean, Hamming, etc.) to their k-nearest neighbors. New individuals that exhibit different and novel behaviors than the existing population are committed to the archive, replacing the less diverse individuals. The premise is that by searching for individuals that exhibit increasingly different behaviors, novelty search can avoid local optimum, a weakness of traditional greedy search-based approaches, and explore the entire search space to discover non-intuitive and interesting solutions.

2.3 Game-based Testing

This section describes game-based testing. Game-based testing is a field of testing driven by modeling the context as a game. Players with different objectives interact with the environment and each other to maximize individual scores.

2.3.1 Game Theory

Game theory is a field of study concerned with the strategic interactions of players in a game. A game consists a number of rational decision-making players interacting in an environment, each seeking to optimize their own objectives. Players in a game may share a number of similar or conflicting objectives, but each aims to maximize their individual objectives or rewards. By studying the players and their interactions, developers can often identify undesirable interactions or decisions in their systems. For example, game theory is often applied to network or software security, modeling the objectives and rewards for adversarial attackers to better understand the strategies and approaches they may use to attack the system.

Games are typically categorized into two main categories in game theory: cooperative games and non-cooperative games [40]. In cooperative games [41], players in the environment may communicate and form coalitions, if deemed beneficial, to maximize their individual rewards. In contrast, players in a non-cooperative game do not communicate nor form coalitions with each other [42]. Instead, players take actions in the game which optimize their individual rewards. Counterintuitively, players in a non-cooperative game may take actions that *appear* to benefit or form coalition(s) with other players. However, they do not explicitly form coalitions with binding agreements, only seeking actions which optimize their rewards.

2.3.2 Reinforcement Learning

RL is a subfield of machine learning focused on training rational agents that make decisions to maximize a reward function. Figure 2.5 overviews the high-level operating loop of RL. An *agent* in RL learns a set of *actions* by interacting with other agents and the *environment* in a simulation. Specifically, agents are given a developer-specified reward function, which describes their given goal in a mathematical function. By repeatedly selecting actions and observing the resulting state, agents in RL learn a set of optimal actions given the environmental states (i.e., policy) to achieve their given goals through trial-and-error. For example, an RL agent learning to park in a dense parking lot may be rewarded based on the distance to the desired parking spot, orientation and alignment to the parking lines, time to completion, and penalized for collisions with other objects

in the environment. By repeatedly executing the parking maneuver in a simulation and attempting to maximize the reward, the agent learns to park based on training experiences.

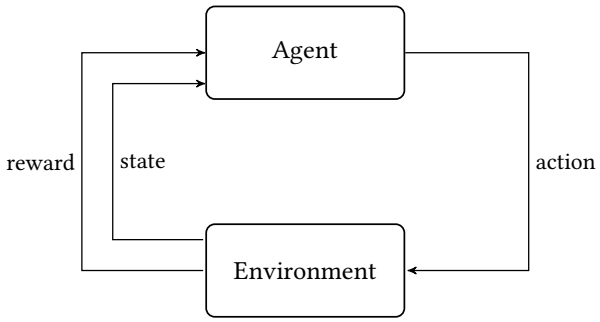


Figure 2.5 An overview of the RL loop at a high-level. An agent first takes an action, changing the environment. The resulting state of the environment is then used by the agent to calculate its current reward, allowing the agent to associate positive actions with higher scores.

CHAPTER 3

SUPPRESSIVE ADVERSARIAL ATTACKS AGAINST LEARNING-ENABLED COMPONENTS

DNNs have demonstrated their application in many domains, increasingly used in critical tasks with performance that can even surpass that of humans [43]. However, researchers have discovered an intriguing property. DNNs are susceptible to minor perturbations, leading to incorrect model behavior, even though the model is able to process the normal input with high accuracy. This chapter presents an approach that addresses malicious human exploitative uncertainty using an objective-driven approach.

Based on the EUREKA assessment framework, this chapter assesses the robustness of an LEC as follows:

- **Asset to be protected:** LEC
- **Threat:** Malicious human exploitative uncertainty
- **Type of assessment and robustness strategy:** This work assesses the robustness of LECs by discovering adversarial examples that can suppress the ability of object detection algorithms from correctly identifying any objects in the image.

3.1 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 [5] (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 DNNs, are susceptible to carefully-crafted malicious inputs, known as *adversarial examples* [34, 35]. 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 [30, 31, 44]. To prevent serious harm or injuries, DNNs deployed in safety-critical systems (e.g., malicious file detection [45, 46], fraud detection [47, 48], and AVs [49, 50]) 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 chapter 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 [34, 35, 51, 52]. 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) [34, 35]. 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.) [33, 36, 53, 54]. 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 [36, 53] 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 chapter 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 [36]. 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 GA [55], 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 per-

turbation 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.

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 [56], Faster R-CNN [57], and the YOLOv5 model [3]. 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 [1], the Waymo Open Dataset [2] for autonomous vehicles, and the VisDrone [4] 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 chapter 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 chapter is organized as follows. Section 3.2 describes the details of the proposed approach. Section 3.3 describes the validation work of our approach. Section 3.4 reviews related work for adversarial attacks against object detection algorithms. Section 3.5 discusses our threats to validity. Finally, Section 3.6 concludes the paper and discusses future directions.

3.2 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 investi-

gations 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.

3.2.1 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 [36, 53]. They can apply both targeted and non-targeted attacks to generate incorrect labels of objects in the image. This chapter 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 $len(F_{obj}(x))$ 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 (3.1) formulates a suppressive adversarial attack.

$$\delta_{min} = \underset{\delta}{\operatorname{argmin}} ||\delta||_p \text{ s.t. } len(F_{obj}(x+\delta)) = 0. \tag{3.1}$$

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

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.

3.2.2 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 *RQ 3.1* 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

RQ 3.1: Can evolutionary search be used to generate adversarial examples against object detection algorithms?

In order to answer this question, we use a generational GA [55]. Figure 3.1 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.

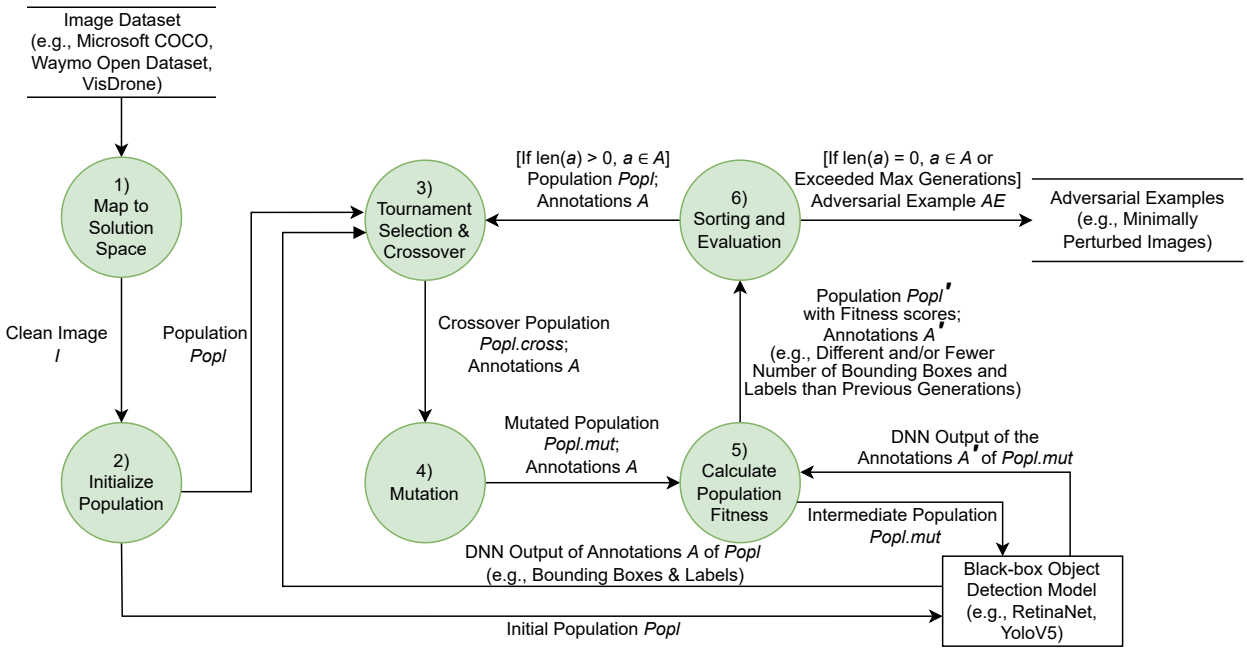


Figure 3.1 DFD for EvoAttack's evolutionary process used to generate adversarial examples against object detection models.

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 success-

fully 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 (3.2). 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).

$$\text{FitnessScore} = \sum_{i=0}^{\text{len(a}')} \text{a}'[\text{i}][\text{confidence\_score}]. \tag{3.2}$$

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).

3.2.3 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 3.2 shows an example input image from the Microsoft COCO dataset [1] 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 (Subfigure 3.2a) 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 Subfigure 3.2b). Thus, we ask the following research question, *RQ 3.2*, to study the impact of localizing perturbations while striving for fewer perturbations in the adversarial examples.

Research Question

RQ 3.2: 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.

(a) Example of a clean input image from the Microsoft COCO dataset [1].

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

Figure 3.2 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.

3.2.4 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 [18] 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 Section 3.2.2 modifies each pixel with a small mutation rate, P_{mut}, during the mutation step of each generation [58, 59]. 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 [30, 31]). 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, *RQ 3.3*.

Research Question

RQ 3.3: 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 (3.3), 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).

Specifically, in EvoAttack, we choose small α and P_{mut} values such that the percentage

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 (3.3).                                                        */
function DYNAMIC-MUTATION(popl, annotations, num_gen, const α, const β, const P_mut,
  const δ_min, const δ_max):
    /* Get the number of pixels in bounding boxes.                         */
    num_of_pixels = annotations.getNumOfPixels() ;
    /* Calculate n based on Expression (3.3).                              */
    if num_gen <= β then
        n = (num_of_pixels * P_mut * num_gen * α)
    else
        n = (num_of_pixels * P_mut * (num_gen/β))
    /* Get bounding box information from the annotations.                  */
    boundingboxes = annotations.getBoundingBoxLocations() ;
    /* Mutate n number of pixels in each bounding box by adding noise between
       (δ_min, δ_max).                                                     */
    popl' ← MUTATE(n, popl, boundingboxes, δ_min, δ_max);
    return popl'
```

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 500$^{\text{th}}$ 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.

$$n = \begin{cases} \text{num\_of\_pixels} * P_{mut} * (\text{generation} * \alpha) & \text{if generation} \leq \beta \\ \text{num\_of\_pixels} * P_{mut} * (\text{generation}/\beta) & \text{if generation} > \beta \end{cases} \tag{3.3}$$

3.3 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).

3.3.1 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.

3.3.1.1 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 [1] 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 [2] 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 [4], 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.

3.3.1.2 Object Detection Models

In our validation studies, we use object detection DNNs implemented using the PyTorch deep learning research platform [60]. 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 [57] and RetinaNet [56] trained using a ResNet-50-FPN backbone [14] for the Microsoft COCO experiments. For Waymo images, we train an object detector using a RetinaNet with a ResNet-50-FPN backbone [14]. For the VisDrone dataset [4], we train and validate our experiments using an object detector with a YoloV5 [3] architecture.

3.3.1.3 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 [32, 33, 61], as criteria for the objective of minimal perturbation. Expressions (3.4) and (3.5) 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.

$$\textbf{L0: } \text{Number\_Pixels\_Changed}(\boldsymbol{x}, \boldsymbol{x_0}) = \|\boldsymbol{x} - \boldsymbol{x_0}\|_0 \tag{3.4}$$

$$\textbf{L2: } \text{Euclidean\_Distance}(\boldsymbol{x}, \boldsymbol{x_0}) = \sqrt{\sum_{i=1}^{\text{n}} (\boldsymbol{x}_i - \boldsymbol{x_0}_i)^2}. \tag{3.5}$$

The objective of minimizing perturbations is important for adversarial examples, as human agents are often assigned to supervise machine learning components [62, 63, 64, 65] (e.g., an airport agent manually reviewing the object detection algorithm's output on an X-ray image [66]). 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 [67, 68]. 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 [31, 33]. 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 [69, 70] (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 [30, 31, 32]. Finally, we also provide the computation time used to generate each adversarial example.

3.3.1.4 Experimental Setup

For evolutionary search parameters, we started with values used in state-of-the-art image classification attacks [58, 59]. 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.

3.3.2 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 Section 3.2.2 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 [1] and the RetinaNet model [56] are used to validate our results. We use the following null and alternate hypotheses to test our research question below.

Research Question

RQ 3.1: 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 3.1 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 *RQ 3.1*.

Figure 3.3 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.

Table 3.1 Comparison of perturbations measured for adversarial examples generated over thirty input images against a RetinaNet model for the Microsoft COCO dataset. The first column represents the average performance of adversarial examples generated by random search, while the second column represents evolutionary search. Blue numbers denote the results for the best performing approach, while red numbers denote results for the worst performing approach.

***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	279,279.31	248,950.52
% of pixels in full image changed	99.89%	89.04%
Degree of perturbations	187.22	49.45
Computation time (sec)	624.64	991.47

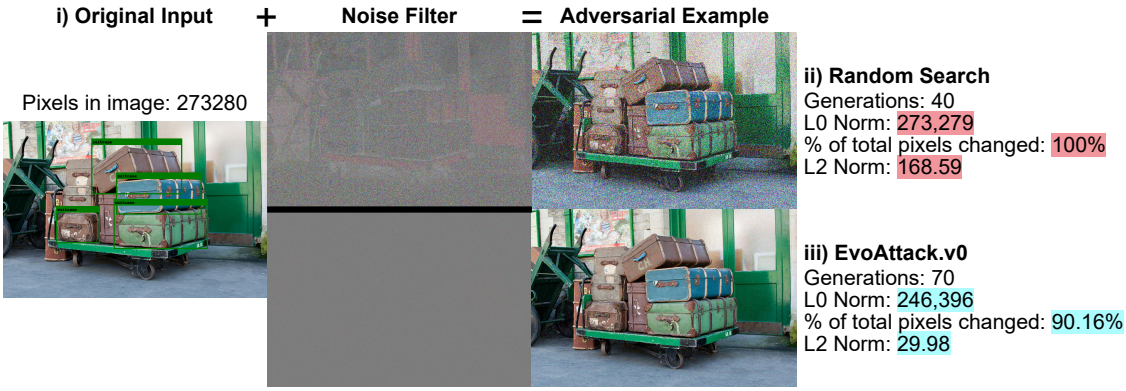


Figure 3.3 Two adversarial examples generated by random search (top row) and EvoAttack.v0 (bottom row). The first figure on the left shows the model's output on the original unperturbed input image. The number of pixels in the image is shown above the image. The middle column shows the noise added to 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).

3.3.3 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.

Research Question

RQ 3.2: 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 and discussion purposes, we term the intermediate algorithm version with localization EvoAttack.v1.

Table 3.2 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 *RQ 3.2*. 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, Figure 3.4 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.

Table 3.2 Comparison of perturbations measured for adversarial examples for Experiment *E2*. 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. Blue numbers denote the results for the best performing approach, while red numbers denote results for the worst performing approach. Boxed numbers highlight the improvement that localization (EvoAttack.v1) provides over EvoAttack.v0 and random.

***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	248,950.52	58,807.84
% of pixels in full image changed	89.04%	21.03%
Degree of perturbations	49.45	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	279,278.31	106,816.17
% of pixels in full image changed	99.89%	38.20%
Degree of perturbations	187.22	80.14
Computation time (sec)	624.64	154.51

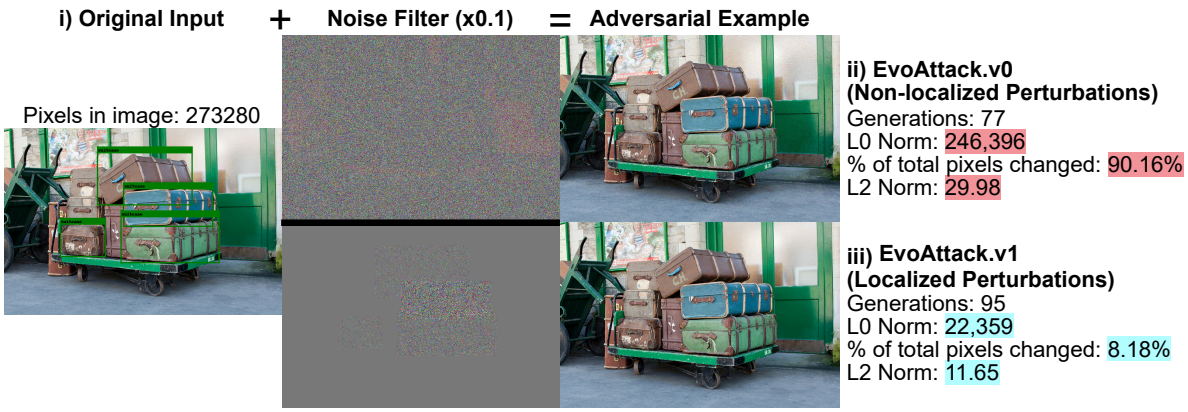


Figure 3.4 Comparison of adversarial example generated between localizing and non-localizing perturbations. The first row shows the result for EvoAttack.v0 (non-localized perturbations), where perturbations are seen in the entire image. The second row shows the result for EvoAttack.v1, where perturbations 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.

3.3.4 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. 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

RQ 3.3: 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 3.5 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.

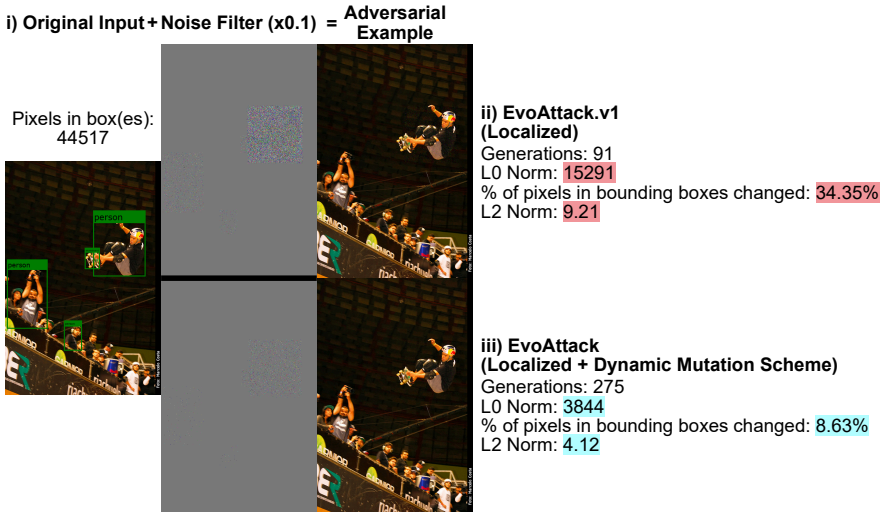
Figure 3.5a.i) shows the original image of a scene at a skateboarding event. The model identified multiple objects of interest in the image and drew bounding boxes around them. While both approaches generated adversarial examples, the adversarial example generated by EvoAttack (see Figure 3.5a.iii) contains significantly fewer perturbations when compared to the adversarial example generated by EvoAttack.v1 (see Figure 3.5a.ii). Since EvoAttack converged on an adversarial example before the β number of generations, the original image in Figure 3.5a.i) is considered to have "converged early".

Figure 3.5b.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

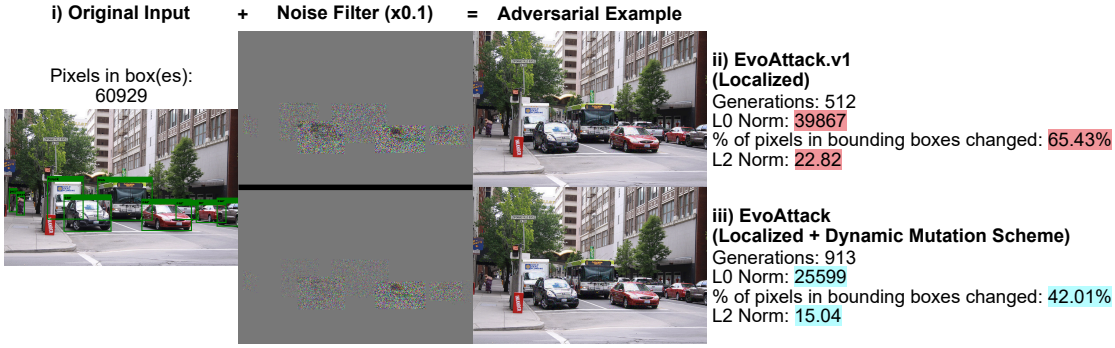
to the input image in Figure 3.5a.i), the input image in Figure 3.5b.i) required more than β number of generations to converge for both (b.ii)) and (b.iii)), where EvoAttack only moderately outperforms EvoAttack.v1 in minimizing perturbations. These figures illustrate that adversarial examples generated by EvoAttack contain fewer perturbations, especially those that converged early before the β number of generations.

The first section of Table 3.3 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 3.3 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 *RQ 3.3*. 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.



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



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

Figure 3.5 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.

Table 3.3 Comparison of perturbations measured for adversarial examples generated over thirty input images against a RetinaNet model for the Microsoft COCO dataset [1]. 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. Blue numbers denote the results for the best performing approach, while red numbers denote results for the worst performing approach.

***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	106,816.17	58,807.84	37,089.50
% 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	103,957.09	51,005.42	23,631.71
% 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

3.3.5 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.

3.3.5.1 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 [57] model and show that we can produce comparable results as an attack against the RetinaNet [56]. 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 3.4 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 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.

3.3.5.2 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 [14] trained over the Waymo Open Dataset [2]. The trained model predicts vehicles, pedestrians, and cyclists 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 3.5 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 an adversarial example is significantly reduced from 18.84 to 3.72 (approximately 5 times less).

Figure 3.6 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.

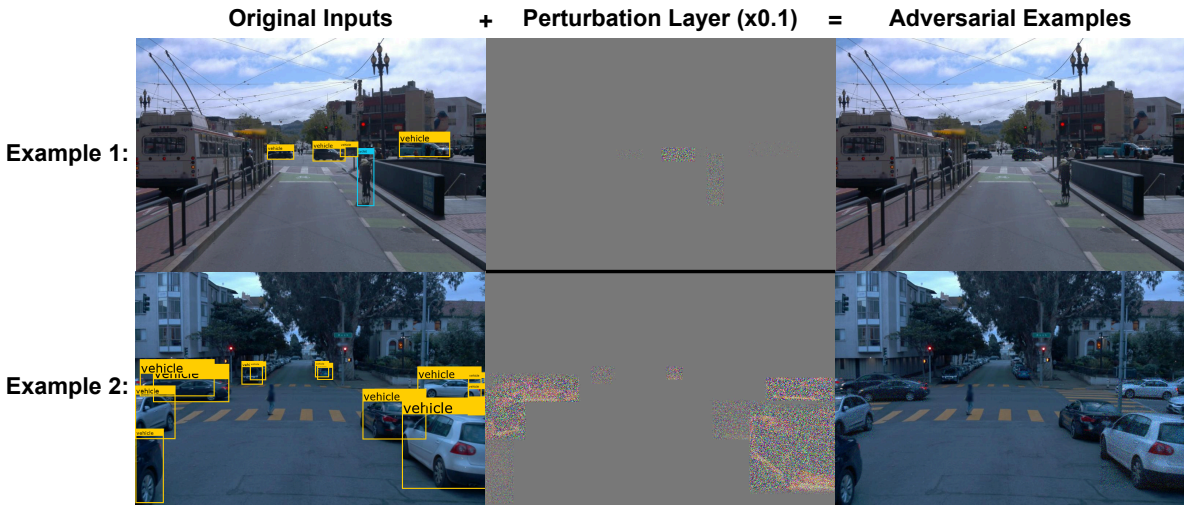


Figure 3.6 Adversarial examples generated against the Waymo Open Dataset [2]. The perturbation layer is amplified by a factor of ten for visualization. The object detection algorithm fails to detect any object in the adversarial example.

Table 3.4 Comparison of perturbations for adversarial examples against a Faster R-CNN model. 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. Blue numbers denote the results for the best performing approach, while red numbers denote results for the worst performing approach.

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	138,829.08	97,014.17
% of pixels in bounding boxes changed	94.87%	66.29%
Degree of perturbations	112.85	40.16
Computation time (sec)	347.67	1,092.88
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	120,375.00	46,414.67
% of pixels in bounding boxes changed	112.20%	43.26%
Degree of perturbations	82.48	13.44
Computation time (sec)	270.85	216.16

Table 3.5 Comparison of perturbations measured for adversarial examples generated over thirty input images against a RetinaNet model for the Waymo Open Dataset [2]. Blue numbers denote the results for the best performing approach, while red numbers denote results for the worst performing approach.

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	56,690.64	36,951.07
% 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	21,889.46	4,678.36
% of pixels in bounding boxes changed	86.83%	18.56%
Degree of perturbations	21.74	3.72
Computation time (sec)	6.00	134.24

3.3.5.3 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, 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 [2] 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 [3] trained on the VisDrone dataset [4], 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 3.6 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 3.7 shows several adversarial examples generated against images sampled from the VisDrone dataset [3]. 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 3.6 Comparison of perturbations measured for adversarial examples generated over thirty input images against a YoloV5 model [3] for the VisDrone Dataset [4]. Blue numbers denote the results for the best performing approach, while red numbers denote results for the worst performing approach.

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	14,504.27	8,900.53
% 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	4,502.73	1,092.53
% of pixels in bounding boxes changed	90.29%	21.91%
Degree of perturbations	8.56	2.36
Computation time (sec)	1.87	143.90

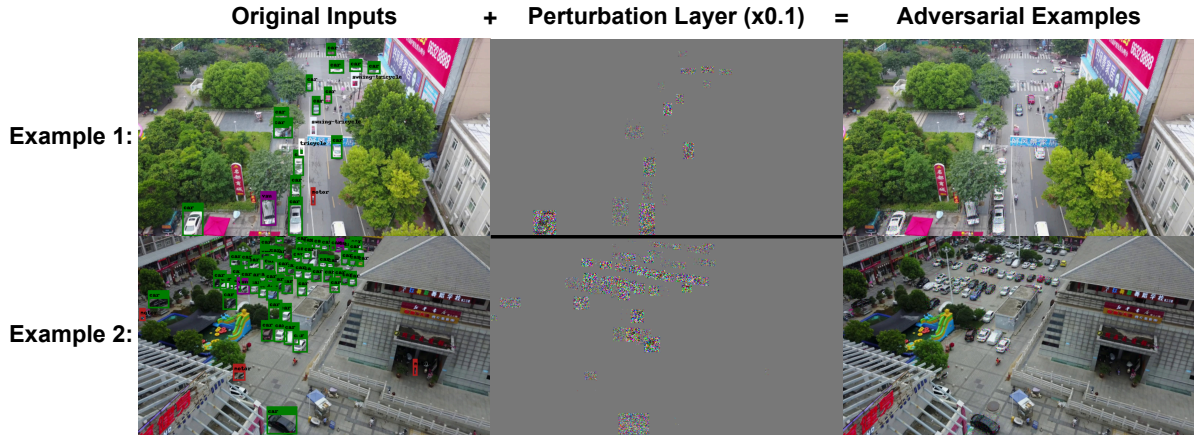


Figure 3.7 Adversarial examples generated against the VisDrone dataset [4], a dataset for aerial drone images (bounding boxes delimited in various colors). Compared to the Waymo Open Dataset [2] for autonomous driving, objects in these images are smaller in dimensions and contains more objects per image.

3.3.6 Discussion

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

3.3.6.1 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 3.3) and the Faster R-CNN model (Table 3.4) 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.

3.3.6.2 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 [31] 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 [18] to better account for various object sizes, thereby avoiding the need to retune parameters across the different application domains.

3.3.6.3 Hyperparameter Tuning

In our validation experiments, we selected parameters for evolutionary search based on existing work on image classification algorithms [58, 59]. 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.

3.4 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 chapter, 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. [34] introduced the first adversarial examples, revealing the existence of malicious images that machine learning models fail to predict correctly. Carlini and Wagner [31] introduced the C&W attack similar to that of Szegedy et al.'s attack [34]. Goodfellow et al. [35] 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. [71] proposed a one-pixel attack using differential evolution. Alzantot et al. [58] proposed GenAttack, which applies a variation of GA to discover adversarial examples. Vidnerová et al. [59], Chen et al. [72], Wu et al. [73], and Han et al. [74] 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 et al. [20] 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. [36] 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 [57] on the PascalVOC dataset. In contrast, Wei et al. [53] proposed a transfer-based attack based on a Generative Adversarial Network (GAN). Li et al. [75] 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. [76] 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. [54] 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. [77] and Sun et al. [78] proposed a similar approach, but leverages differential evolution instead. Lapid and Sipper [79] 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).

3.5 Threats to Validity

The results in this chapter 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

chapter. 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.

3.6 Conclusion

This chapter 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 [80]) and parallel GAs [81]. 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 [13, 38] may be leveraged to discover a collection of adversarial examples that causes diverse behaviors in the model. Finally, Enlil [14] (i.e., behavior oracles) may be used to predict the uncertain behavior of the object detection model when exposed to adversarial examples.

CHAPTER 4

EXPOUND: A BLACK-BOX APPROACH FOR GENERATING DIVERSITY-DRIVEN ADVERSARIAL EXAMPLES

This chapter describes our preliminary investigation in the development of a framework that can assess diversity-driven exploitative uncertainty with malicious intentions. Based on the EUREKA assessment framework, this chapter's work assesses the robustness of an LEC as follows:

- **Asset to be protected:** LEC
- **Threat:** Malicious human exploitative uncertainty
- **Type of assessment and robustness strategy:** This approach assesses the robustness of LECs towards adversarial noise in two dimensions. First, we assess the range of failures that an adversarial example is likely to be misclassified as. Second, for each failure identified, we present a diverse range of faults that can lead to that given failure.

As such, this chapter explores an approach that provides developers with valuable insights regarding the limitations, extremes, and regions of failures found in their LEC due to adversarial noise.

4.1 Introduction

Due to their ability to learn abstract patterns and representations directly from data, an increasing number of autonomous systems are using machine learning, DNNs, and related technologies for decision making [82]. However, the existence of adversarial examples [34] suggests that they are prone to unexpected failures, such as uncertainty factors and minor perturbations possibly due to adversarial attacks. The *robustness* of DNNs describes their ability to correctly operate in the face of minor noise perturbations or uncertainty factors [10, 83]. DNNs used in safety-critical autonomous systems (e.g., medical imaging, AVs [6, 7], etc.) must be robust against minor data perturbations, as the failure of these systems may lead to potential injuries, financial damages, or loss of life. This chapter introduces a black-box diversity-focused search framework to discover failure categories and enable the automated assessment of the severity of each failure category.

Recent state-of-the-art research has suggested the use of *diverse*[1] adversarial examples to address the inability of a DNN to generalize to previously unseen or unknown data deviations [9, 70, 84]. Traditional approaches to generate adversarial examples typically discover the "nearest" adversarial examples. In contrast, Rozsa *et al.* promoted the use of adversarial examples that 1) have non-minimal perturbations and 2) that result in different model misbehaviors. Aghababaeyan *et al.* showed that the robustness of a DNN model can be better assessed and improved using inputs that trigger different faults. Currently, state-of-the-art research identifies diverse adversarial examples by using white-box techniques to exhaustively search for perturbations, each of which trigger a unique given model misbehavior. However, white-box techniques require access to hidden model information and may not work if the gradient is obfuscated. Additionally, brute-force approaches do not scale as the number of model behaviors (e.g., number of class labels) in a dataset increases.

This chapter introduces **EXP**loration and expl**O**itation **U**sing **N**ovelty search for **D**iversity (EXPOUND),[2] a black-box search-based approach to generate diverse adversarial examples to identify the failure categories for a DNN model. Our work contributes several key insights. As the space of possible perturbations for adversarial examples is large, our work leverages the *exploration and exploitation paradigm* [22] in order to strategically discover diverse adversarial examples for different failure categories. More specifically, EXPOUND first uses a coarse-grained *exploration* search to discover the categories of diverse erroneous behaviors for a DNN model, each of which is termed a *failure category*. Second, for each failure category, EXPOUND then applies a localized fine-grained *exploitation* search to identify a diverse secondary collection of adversarial examples, each of which lead to similar model misbehavior (i.e., different perturbations that lead to the same misclassification). Developers can then analyze the adversarial examples and focus their efforts to improve the model's performance against the most relevant adverse failure categories for their applications.

In order to generate diverse adversarial examples, EXPOUND uses novelty search [38] in two complementary ways. In contrast to traditional evolutionary search techniques, novelty search

[1] We use the term *diversity* to describe different DNN model behaviors.

[2] One definition for *expound* is to explain systematically or in detail.

abandons the notion of objective fitness by promoting a diversity metric instead. Specifically, novelty search rewards individuals that are different from each other and produces an archive of diverse individuals. We developed two novelty-based techniques to support our objectives. First, KOBALOS[3] uses novelty search to *explore* the search space in order to *discover* the diverse set of possible failure categories for a DNN, where the diversity metric is based on the diversity in the *output of the DNN model* (e.g., image classification label). KOBALOS *explores* and reduces the search space to diverse types of faults that are likely to induce a failure in the model. Next, TARAND[4] *exploits* the failure categories identified by KOBALOS. TARAND applies novelty search for each failure category, generating a *diverse set of faults* (e.g., perturbations) that cause the same type of model misbehavior. TARAND enables developers to assess the failure categories to determine which categories might be confidently misclassified. The breadth and depth-based approach to generate a diverse collection of image perturbations can better inform developers as to how a model's performance and robustness can be improved (e.g., select specific training data, robustness analysis, etc.). Figure 4.1 overviews the differences between image classification, existing black-box techniques for adversarial examples, and EXPOUND, where elements with hatched shading denote adversarial artifacts. Image classification algorithms seek to correctly label an object in an image (Figure 4.1a). The objective of evolutionary search-based adversarial attacks, such as GenAttack [58] or EvoAttack [18], is to generate *one* adversarial example per execution (Figure 4.1b) to establish the existence of adversarial examples. In contrast, EXPOUND identifies multiple failure categories, each of which comprises a number of distinct adversarial examples (Figure 4.1c).

Preliminary results indicate that EXPOUND can discover adversarial examples that lead to different (undesirable) model behaviors that would otherwise not be discovered by existing evolutionary search-based approaches. We implemented the proposed two-phase framework and demonstrated the results on existing image classification algorithms and benchmark datasets. To illustrate that our approach is model and data-agnostic, we used two different DNN architectures, ResNet-20 [85] and MobileNet-V2 [86], on two different benchmark datasets, CIFAR-10 [87] and German Traffic

[3]Kobalos is a mischievous sprite in Greek mythology.

[4]Tarand is a legendary creature with chameleon-like properties in Greek mythology.

Sign Recognition Benchmark (GTSRB) dataset [88]. The remainder of the paper is organized as follows. Section 4.2 describes the methodology for our proposed framework. Section 4.3 provides the implementation details and the results of a proof-of-concept validation experiment. Section 4.4 discusses related work. Section 4.5 discusses our threats to validity. Section 4.6 concludes the paper and discusses future directions.

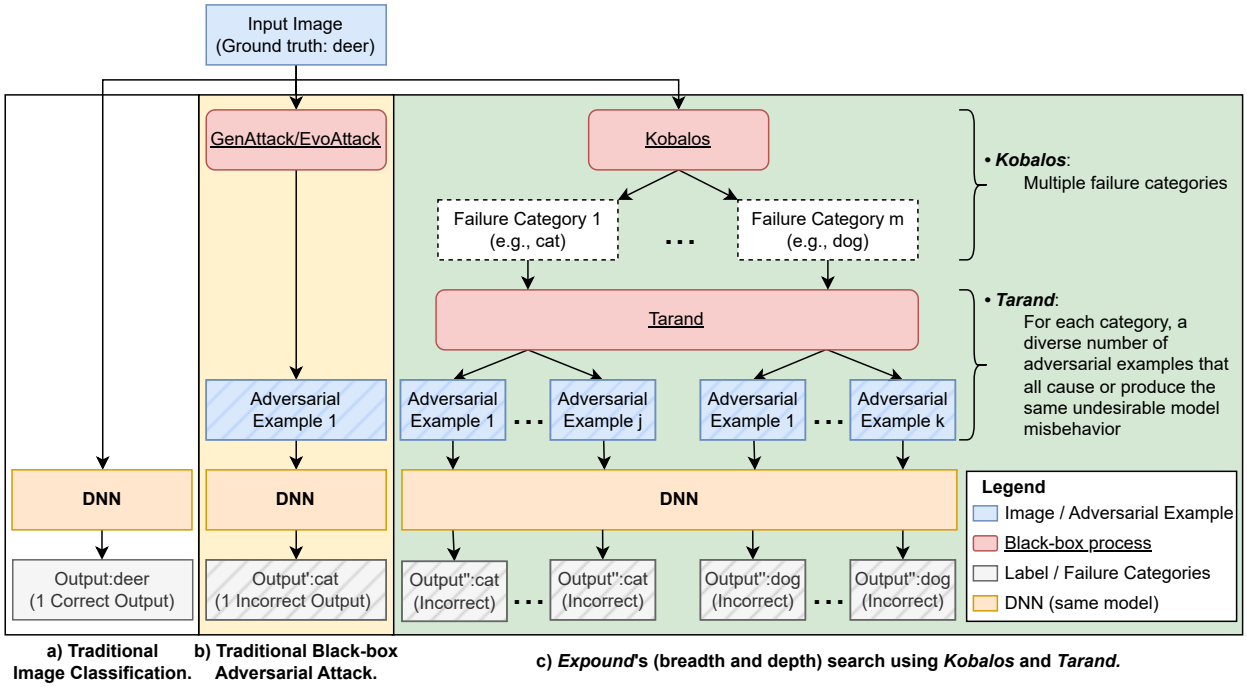


Figure 4.1 Overview of image classification, adversarial example, and EXPOUND.

4.2 Methodology

This section describes our EXPOUND framework, a two-phase search-based technique used to generate diverse archives of adversarial examples that result in different failure categories of the DNN model. Figure 4.2 shows a DFD for EXPOUND, where circles indicate process bubbles, arrows indicate data flow between processes, and parallel lines indicate external data stores. EXPOUND takes a non-perturbed input image and the black-box image classification algorithm as inputs. First, a developer provides the evolutionary parameters (e.g., mutation rate, population size, etc.), operating context, and behavior specification for EXPOUND. The operating context defines the range of perturbations that EXPOUND can search to discover different model behaviors (i.e., the genome

range). For example, a developer may define the operating context to generate adversarial examples with non-minimal perturbation, restricted to 10% of a pixel's maximum value. The behavior specification defines the possible range of behaviors exhibited by the model (i.e., the phenome range), such as image classification labels. Next, we describe the steps of EXPOUND in turn.

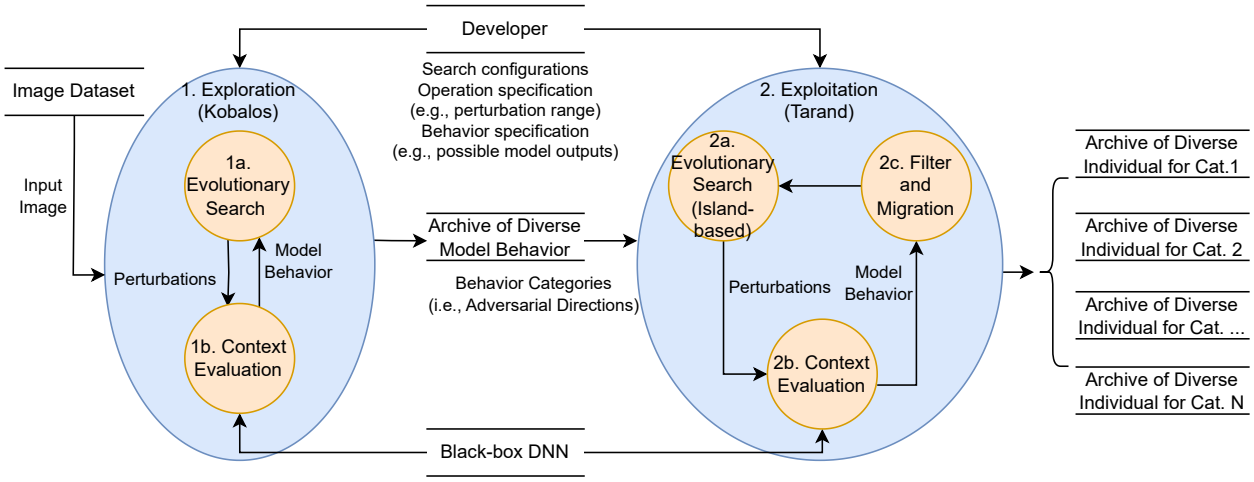


Figure 4.2 EXPOUND's process to identify failure categories, each comprising diverse adversarial examples.

4.2.1 Step 1. Exploration

During the *exploration* phase, EXPOUND uses a coarse-grained novelty search to identify the broad adversarial directions when exposed to perturbations. For discussion purposes, we use the term *failure category* in this work to describe the diverse collection of adversarial examples that lead to the same adversarial direction. Specifically, different incorrect model outputs (e.g., misclassifying a *dog* as a *cat* versus a *dog* as a *deer*) are considered different failure categories, as the model incorrectly classified the image in different ways. For image classification problems, the number of potential failure categories increases based on the number of labels in the dataset. As such, it is computationally inefficient to exhaustively search every label to determine whether the model will misclassify an input image as the corresponding label when exposed to perturbations. Given an unperturbed input image and the DNN classifier, KOBALOS addresses this issue by exploring the search space and generating a collection of individuals that result in the most diverse model behaviors (i.e., most number of mutually exclusive class labels). The left portion of Figure 4.3

provides a visual representation with colored regions denoting classification categories to illustrate KOBALOS's search on an input image of a *deer*. In this example, KOBALOS identifies two adversarial examples with different incorrect class labels, *dog* and *cat*, for the clean input image that has been exposed to perturbation. The bold arrows point towards incorrect class labels, indicating the failure categories identified by KOBALOS.

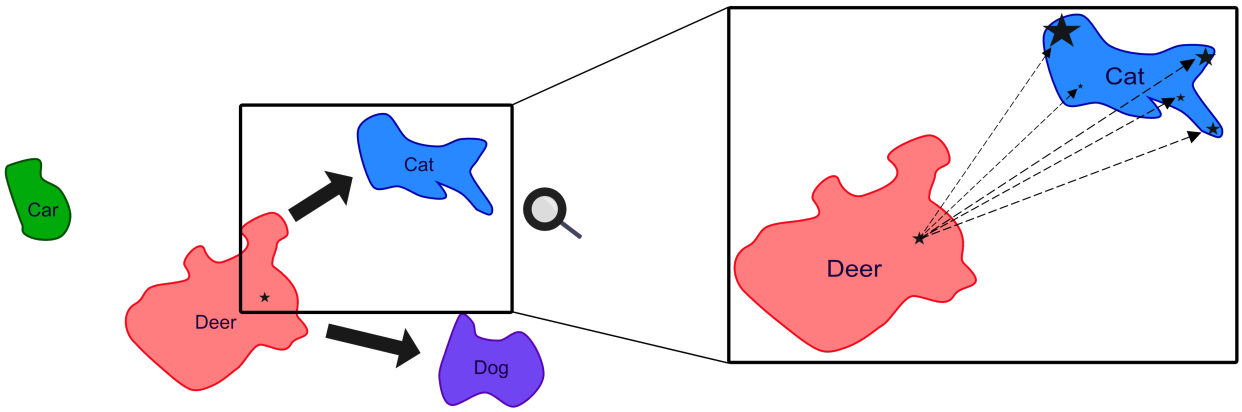


Figure 4.3 High level interpretation of EXPOUND. Stars denote images. KOBALOS (left) discovers the failure categories while TARAND (right) exploits the information to discover diverse adversarial examples for each category identified.

Algorithm 2 shows the pseudo-code of KOBALOS to identify the broad failure categories using novelty search. KOBALOS largely follows a standard evolutionary algorithm approach in Step 1. First, a population of perturbation filters is randomly generated (i.e., filters are initialized with randomly sampled uniform noise). During each generation, individuals in the population are selected to reproduce, favoring those that result in phenome diversity. Specifically, the outputs of the DNN (i.e., vectors consisting of the label probabilities) are used as the phenomes. Parents create offspring using a single-point crossover operator and creep mutation operator. The creep mutation operator slightly increases or decreases the value of elements randomly sampled between a *mutation shift* variable with a *mutation rate* probability. Next, KOBALOS evaluates the individuals with the DNN to obtain the probability vectors (Step 1b in Figure 4.2). Individuals are then ranked by computing a novelty score based on the *Euclidean distance* between phenomes with other individuals in the archive (line 14). Expression 4.1 shows the novelty score metric used in EXPOUND, computing the average distance between an individual (p) with the k-nearest neighbors

in the archive (P).

$$\text{novelty(p, P, } k) = \text{mean(min-k(} \left\| \text{p} - \text{p}_i \right\|^2 \forall \text{p}_i \in \text{P} : \text{p}_i \neq \text{p}, k)) \tag{4.1}$$

The archive of individuals (i.e., the output of KOBALOS) is then updated to maintain the most diverse individuals (i.e., those with different labels) discovered by replacing individuals with low novelty scores with individuals with higher scores (line 15). KOBALOS uses phenome diversity (i.e., classification output labels) to strategically explore the search space and identify the probable types of faults, instead of using a brute-force approach to explore all possible labels.

Algorithm 2: Pseudo-code for KOBALOS and TARAND [13, 38].

```
1  function NOVELTY-SEARCH(n_generations, image):
2      archive ← ∅
3      popl ← RANDOM-POPULATION()          // Initialize population with random noise.
4      for ( n to n_generations ) {
5          popl ← SELECTION(popl)          // Select genomes via tournament selection.
6          popl ← RECOMBINATION(popl)      // Recombine via single-point crossover.
7          popl ← MUTATION(popl)           // Mutation via creep mutation.
8          popl ← EVALUATE(popl)           // Evaluate individuals using DNN.
9          archive, popl ← COMMIT-TO-ARCHIVE(archive, popl)
10         popl ← FILTER(popl)             // TARAND ONLY: filter and migrate individuals.
11     }
12     return archive
13 function COMMIT-TO-ARCHIVE(archive, popl):
14     scores ← RANK-INDIVIDUALS(archive, popl)   // Calc. novelty score (Eq.(1)).
15     archive ← TRIM(archive ∪ popl)             // Combine and truncate to desired size.
16     return archive, popl
```

4.2.2 Step 2. Exploitation

Once the failure categories have been identified in the exploration phase, EXPOUND uses TARAND for the *exploitation* phase in **Step 2** to exploit the KOBALOS-discovered information. For each failure category identified in **Step 1**, TARAND uses novelty search to generate secondary archives of adversarial examples based on genome diversity (i.e., different combinations of noises that lead to the same misclassification). The archives of individuals (i.e., diverse types of faults) enable a more informed assessment of the types of perturbations within a given failure category. The right portion

of Figure 4.3 visually illustrates TARAND's search process to exploit the information identified by KOBALOS. Stars denote the original input image (i.e., deer) and adversarial examples generated by TARAND (e.g., an individual misclassified image). The relative size of individual stars contrast the respective confidence scores of the DNN for the image [70]. In this example, TARAND discovered a number of diverse adversarial examples that lead to the same *cat* misclassification. Adversarial examples discovered by TARAND in the archive have different types of perturbations, yet cause the same type of model misbehavior, denoted by thin dashed arrows. Understanding the nature and confidence scores for each failure category enables developers to focus their efforts to improve the robustness of the DNN (e.g., identify additional training data that addresses the misclassifications).

To discover a range of perturbations for each category of misclassification, TARAND maintains multiple islands of genomes. TARAND largely follows the same evolutionary process used by KOBALOS, but differs in the diversity metric. Specifically, KOBALOS optimizes a diversity metric based on the model's output (i.e., phenome defined in terms of image classification labels), while TARAND optimizes the diversity with respect to the perturbations that yield the same model output (i.e., genome). First, a population of perturbation filters are randomly generated using uniform sampling. During **Step 2a**, individuals are selected to evolve using single-point crossover and creep mutation operators based on their genome diversity. **Step 2b** evaluates the individuals in the population. The novelty score is computed using Expression 4.1, where P is based on the *genome* diversity of the population instead (i.e., TARAND promotes different perturbation filters). TARAND also includes an additional step, **Step 2c**, where individuals on an island that exhibit behavior different from other individuals on the same island are moved to the island with corresponding behavior (blue line 10 in Algorithm 2). For example, an adversarial example with the label *dog* on an island of *cats* will be filtered and migrated to the island containing *dogs*.

Misclassification score. As novelty search promotes diversity, adversarial examples generated by TARAND can provide developers with insight on the boundaries of errors and confidence scores for the misclassifications in each failure category. For example, a developer may want to identify

and prioritize categories that are confidently misclassified. Confidently misclassified inputs may pose a higher risk to safety-critical systems, as the system may mistakenly trust the model's output with higher weight based on the confidence score. Identifying confidently misclassified categories enables developers to include additional training data that may help the model distinguish between the two types of images. In this work, we first calculate and sort the archive of adversarial examples by confidence scores using the output of the model. We define the misclassification scores for each category using the average confidence score of the top 50% of sorted adversarial examples (i.e., archive$[0.5 : 1]$.getConfidenceScores()), denoted by Expression (4.2). Specifically, categories that contain several confidently misclassified images will be assigned a higher score, close to 1.0.

$$\text{MisclassificationScore(archive)} = \text{mean(archive}[0.5 : 1].\text{getConfidenceScores())} \tag{4.2}$$

4.3 Validation studies

This section demonstrates the EXPOUND framework using an empirical evaluation. We used two image classification DNNs with different architectures on two different image benchmark datasets to conduct our experiments. In order to assess whether EXPOUND can identify more failure categories for the DNN models, we compared KOBALOS with a Monte Carlo sampling method. Then, we compared the ability of TARAND and Monte Carlo sampling to generate archives of diverse adversarial examples for each failure category. This section addresses the following research questions:

Research Question

RQ 4.1: Can EXPOUND identify distinct failure categories using KOBALOS?
RQ 4.2: Can EXPOUND identify more faults using TARAND for each failure category?

4.3.1 Experimental setup

This section describes the experimental setup and evolutionary parameters used in our validation experiments. For our experiments, we used existing image classification algorithms implemented

using the Pytorch deep learning research platform [60]. In order to illustrate that our approach is model and data-agnostic, we use two different benchmark datasets for our validation work. First, the CIFAR-10 benchmark dataset is commonly used for image classification algorithms [87], comprising ten evenly distributed categories. For the CIFAR-10 dataset, we used a ResNet-20 model with a base accuracy of 91.58% and a MobileNet-V2 model with a base accuracy of 91.52%. Second, we also use the GTSRB dataset [88]. Unlike the CIFAR-10 dataset, the GTSRB dataset consists of 43 classes of traffic signs with an uneven distribution of labels. For the GTSRB dataset, we used a ResNet-20 model with a base accuracy of 96.28% and a MobileNet-V2 model with a base accuracy of 90.77%.

In order to provide a baseline evaluation for our framework, we implemented a Monte Carlo sampling approach (i.e., randomly sample noise). Each approach generates an archive size based on the total number of labels in the dataset (i.e., 10 for CIFAR-10 and 43 for GTSRB). Table 4.1 shows the evolutionary parameters used in EXPOUND, where the parameters are based on empirical studies. The maximum perturbation for each approach is set to $[-0.1, +0.1]$ of the maximum pixel value. All experiments are conducted on a NVIDIA GeForce GTX 1080 GPU with an Intel Core i7-7700K CPU on Ubuntu 22.04. Finally, each experiment is repeated ten times using different seeds. We show the average value of the experiments, their standard deviations, and provide statistical analysis to ensure the feasibility of our approach. For the results, we verify their statistical significance using the Mann-Whitney U test.

Table 4.1 Configuration settings for EXPOUND

	CIFAR10		GTSRB	
Parameter	Value(Kobalos)	Value(Tarand)	Value(Kobalos)	Value(Tarand)
Archive Size	10	10	43	10
Population Size	10	10	10	43
Num Generations	100	500	100	500
Mutation Rate	0.15	0.15	0.15	0.15
Mutation Shift	0.2	0.2	0.2	0.2

4.3.2 Coarse-grained search for distinct failure categories (RQ 4.1)

To address our first research question, we evaluate KOBALOS's ability to discover distinct failure categories for DNN models by comparing the results with Monte Carlo sampling. We define our

null and alternate hypotheses as follows:

Research Question

RQ 4.1$-H_0$: KOBALOS does not identify more failure categories.
RQ 4.1$-H_1$: KOBALOS identifies more failure categories.

For each experiment and dataset, 30 randomly-sampled input images were used. For each image, a collection of 10 individuals is generated for the CIFAR-10 dataset while a collection of 43 individuals is generated for the GTSRB dataset, based on the number of possible labels for each dataset.

In order to ensure the feasibility of our approach, results are shown as the mean over ten trials to account for statistical variance. Table 4.2 shows the average number of unique labels identified by each approach when used to discover failure categories. The standard deviation is denoted in parenthesis, while the best performing technique is indicated in bold. The p-value for each experiment is shown in the last column. For each combination of DNN architectures and datasets, EXPOUND consistently discovered more failure categories than Monte Carlo sampling (on average 80% more for CIFAR-10 and 270% more for GTSRB). We found strong evidence ($p \leq 0.01$) to reject H_0 and support H_1. These results indicate EXPOUND can consistently discover more distinct failure categories.

Table 4.2 Results for Monte Carlo sampling and EXPOUND's search for failure categories in the DNN's output. EXPOUND consistently identified more failure categories.

DNN and Dataset	**Number of Average Unique Labels**		p-value significance
	Monte Carlo	KOBALOS	
ResNet-20 - CIFAR-10	2.02 ($\sigma = 0.10$)	**3.62** ($\sigma = 0.17$)	< 0.01
MobileNet-V2 - CIFAR-10	1.83 ($\sigma = 0.09$)	**3.33** ($\sigma = 0.13$)	< 0.01
ResNet-20 - GTSRB	1.94 ($\sigma = 0.08$)	**7.11** ($\sigma = 0.19$)	< 0.01
MobileNet-V2 - GTSRB	1.73 ($\sigma = 0.09$)	**6.45** ($\sigma = 0.09$)	< 0.01

Figure 4.4 shows an example output of EXPOUND's exploration phase for an image randomly sampled in the CIFAR-10 dataset against the ResNet-20 model. The top image shows the clean input image and the model's correct classification output. Next, the collection of noise filters

generated by KOBALOS (left) and the corresponding adversarial examples (right) are shown in the second row. As indicated by the number of misclassifications, adversarial examples generated by KOBALOS induce different distinct misbehaviors in the model's output.

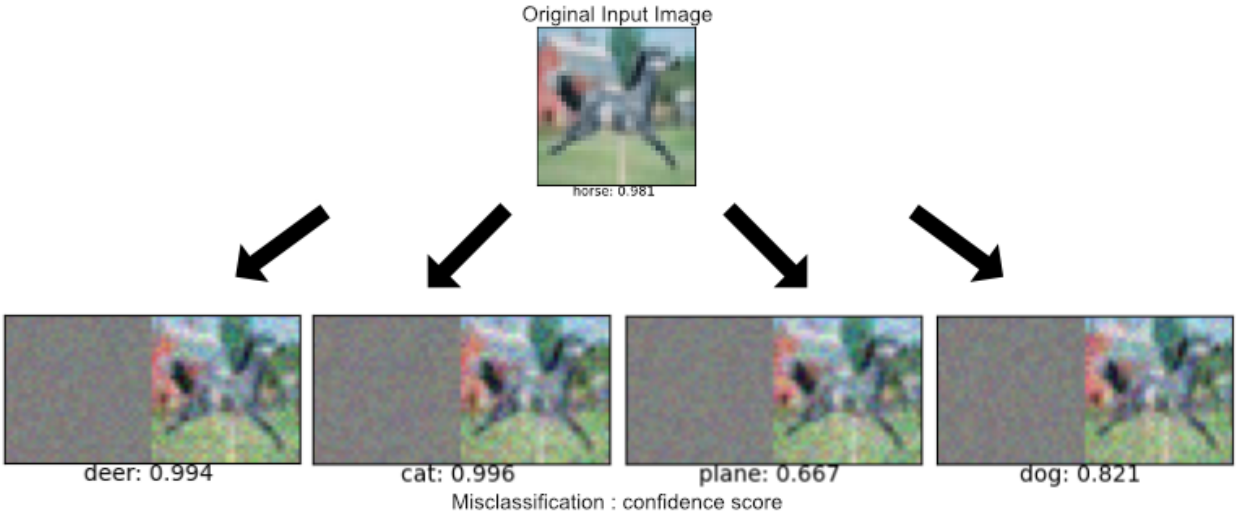


Figure 4.4 Example output of KOBALOS on an input image of a *horse*. KOBALOS discovered four different incorrect model behaviors when exposed to perturbations.

4.3.3 Fine-grained search for diverse adversarial examples (RQ 4.2)

In this experiment, we demonstrate that TARAND can generate a diverse archive of adversarial examples for each failure category identified by KOBALOS. Specifically, we illustrate the ability of TARAND to exploit the information from KOBALOS and generate an archive of ten individuals for each category of misbehavior identified. For example, if six failure categories have been identified by KOBALOS, then TARAND attempts to generate six secondary archives of adversarial examples, one for each failure category. We compare the results of TARAND with Monte Carlo sampling. Monte Carlo sampling generates adversarial examples equal to the number of adversarial examples generated by TARAND. The adversarial examples generated by Monte Carlo sampling are then sorted into their corresponding failure categories. We define our null and alternate hypotheses to answer this question as follows:

Research Question

RQ 4.2$-H_0$: TARAND does not identify more faults for each failure category.
RQ 4.2$-H_1$: TARAND identifies more faults for each failure category.

Table 4.3 shows the experimental results for the fine-grained exploitation search. The table shows how well a given technique under study is able to generate distinct and diverse adversarial examples, each resulting in similar misclassification labels. For example, a full archive (i.e., 100%) for an image from the CIFAR-10 dataset denotes that 10 adversarial examples were discovered for each failure category identified. The standard deviation is shown in parenthesis. The final column of the table shows the corresponding p-value using the Mann-Whitney U test. When applied to the same set of thirty images, TARAND is able to successfully generate a (relatively full) archive of adversarial examples for most failure categories. In contrast, Monte Carlo sampling consistently did not generate adversarial examples for the failure categories. For each experiment, we found strong evidence ($p \leq 0.01$) to reject H_0 and support H_1. The results indicate that TARAND can generate more and different perturbations for a given failure category, a new capability not offered by existing approaches.

Table 4.3 Comparing TARAND's generation of diverse archives of adversarial examples for the categories of misbehavior against Monte Carlo sampling.

DNN and Dataset	Percentage of Archives Filled		p-value significance
	Monte Carlo	TARAND	
ResNet-20 - CIFAR-10	26.36% ($\sigma = 2.69\%$)	**93.73%** ($\sigma = 1.35\%$)	< 0.01
MobileNet-V2 - CIFAR-10	33.43% ($\sigma = 1.09\%$)	**91.59%** ($\sigma = 1.20\%$)	< 0.01
ResNet-20 - GTSRB	14.48% ($\sigma = 1.05\%$)	**81.16%** ($\sigma = 1.09\%$)	< 0.01
MobileNet-V2 - GTSRB	11.12% ($\sigma = 0.95\%$)	**71.93%** ($\sigma = 1.43\%$)	< 0.01

The archives of adversarial examples generated by TARAND can be used to assess the model's robustness towards each of the incorrect labels identified by EXPOUND. Specifically, each archive of adversarial examples generated by TARAND promotes genome diversity that produces the same model behavior failure. Thus, we can identify relevant and confidently misclassified failure categories based on the confidence scores of the adversarial examples. We calculate the misclassifi-

cation score based on Expression (4.2) defined in Section 4.2. Figure 4.5 shows a sample output of TARAND randomly sampled from the CIFAR-10 dataset. The ground truth for the clean input image is *truck*. Each row denotes a failure category identified by EXPOUND. Individual scores show the confidence scores for the diverse archive of adversarial examples generated by TARAND. Finally, the misclassification score of each category is computed and shown in parenthesis next to the class label. The failure categories are sorted based on the misclassification score, in descending order. As the results indicate, adversarial examples with the labels *bird* and *ship* are not commonly misclassified with high confidence scores (i.e., less than 0.7). However, the model confidently misclassifies adversarial examples as *frogs*. Thus, developers may use this information to include additional training samples that distinguish frogs from trucks.

```
Frog-(Score=0.78): [0.39, 0.40, 0.46, 0.50, 0.64, 0.67, 0.71, 0.81, 0.81, 0.90]
 Cat-(Score=0.68): [0.31, 0.35, 0.43, 0.47, 0.48, 0.53, 0.59, 0.61, 0.71, 0.94]
Bird-(Score=0.65): [0.37, 0.39, 0.46, 0.49, 0.54, 0.56, 0.65, 0.67, 0.68, 0.69]
Ship-(Score=0.57): [0.32, 0.35, 0.38, 0.39, 0.47, 0.50, 0.52, 0.58, 0.58, 0.68]
```

Figure 4.5 An analysis of adversarial examples generated with TARAND for an image of a *truck*. The (sorted) misclassification score is shown in parenthesis. The numbers in brackets show the confidence scores of individual adversarial examples.

4.4 Related Work

Early work with adversarial examples was generated using white-box approaches [89]. These approaches leverage gradient information of the model to be attacked and introduce noise to hinder the model's capability. Szegedy *et al.* [34] first introduced adversarial examples generated using the L-BFGS algorithm. Carlini and Wagner [31] proposed a similar approach with a modified objective function and constraint, known as the C&W attack. Goodfellow *et al.* [35] introduced the FGSM that perturbs the target image based on the signed gradient at each step. However, these techniques are all white-box, where gradient information and other model parameters are assumed to be accessible.

A number of researchers have also explored black-box evolutionary search-based approaches to generate adversarial examples. Compared to white-box approaches, black-box approaches closely

resemble a real attack scenario, where the development of the model is proprietary. Papernot *et al.* [90] demonstrated that white-box adversarial attacks can transfer to other models. In contrast, Alzantot *et al.* [58], Vidnerová *et al.* [59], Chen *et al.* [72] proposed various genetic algorithm approaches to generate adversarial examples against image classification algorithms. Chan and Cheng proposed EvoAttack [18] that showed adversarial attacks also apply to object detection algorithms, an important component for perception sensing in autonomous vehicles. However, these approaches generate adversarial examples that do not reveal diverse model behaviors.

Several existing work has explored the use of diversity for DNNs. Rozsa *et al.* [70] introduced the use of a diverse set of adversarial examples to evaluate and retrain models. However, their approach generates adversarial examples using a white-box approach and does not discover failure categories. Aghababaeyan *et al.* [9] proposed the use of the geometric diversity metric to identify a diverse set of test cases from the original input dataset based on their similarity to other samples. Langford and Cheng [13, 14] proposed several novelty search techniques to explore the effect of environmental uncertainties on DNNs. However, their approach does not address the robustness of the DNN against noise perturbation or adversarial examples. In their work, Enki [13] is used to generate environmental uncertainties against a DNN model applied over a dataset. The identified environmental uncertainties demonstrate the inability of the DNNs to process the occluded inputs, including those that prevent human classification. They also proposed Enlil [14], a novelty search framework to discover operating contexts that can lead to different categories of performance degradation. To the best of our knowledge, EXPOUND is the first black-box approach to explore the use of novelty search to generate diverse adversarial examples that can discover a number of distinct failure categories, providing both breadth and depth-based information about the types of perturbations that cause model misbehaviors.

4.5 Threats to Validity

The results in this chapter show that a diverse collection of adversarial examples can be generated against DNNs using novelty search. The results of the experiment may vary with repeated executions, as novelty search and evolutionary search-based algorithms rely on randomness. To

ensure the feasibility of our approach, a wide range of sample of input images were sampled from different datasets. Additionally, each experiment was repeated ten times with different randomly-generated seeds to ensure comparable results.

4.6 Conclusion

This chapter proposed EXPOUND, a novelty search-based framework to generate diverse adversarial examples. We showed that our two-phase framework first identifies failure categories for image classification algorithms that would otherwise not be discovered by existing approaches. Furthermore, EXPOUND then generates secondary archives of diverse adversarial examples for each failure category identified, enabling the assessment for the misclassification score of each category. We conducted a series of experiments to show that our approach can successfully generate adversarial examples against different datasets and models without access to model weights, architectures, and gradient information.

Future work will explore additional datasets and models for validation results. Additional studies may be performed to explore whether noise generated from EXPOUND transfers to other images. Future studies may also explore whether novelty search might be used to discover if "universal" adversarial perturbations exist and can be used to improve a DNN's robustness. Additional work may explore the ability of EXPOUND to generate diverse adversarial examples against models with defense mechanisms [91]. Finally, we will investigate whether EXPOUND can be applied to other types of machine learning algorithms, such as speech analysis, natural language processing, or object detection algorithms.

CHAPTER 5

ASSESSING LES ROBUSTNESS TOWARDS NON-INTENTIONAL HUMAN EXPLOITATIVE UNCERTAINTY

When deployed in operating contexts with human agents, LESs must behave correctly in the face of uncertainty posed by humans. While most humans may not exhibit malicious behaviors, their selfish or non-intentional behaviors due to their personal objective(s) may lead to previously unseen human actions that result in uncertainty regarding the LES's ability to respond correctly or safely. This chapter explores incidental human exploitative uncertainty using an objective-driven technique. Based on the EUREKA assessment framework, this chapter assesses the robustness of an LES as follows:

- **Asset to be protected:** LES (e.g., AV)
- **Threat:** Incidental human exploitative uncertainty
- **Type of assessment and robustness strategy:** This work explores an automated testing technique that can discover previously unseen human maneuvers, which can then be used to assess the ability of the LES to handle the uncertainty.

5.1 Introduction

Recently, significant and rapid advancements in deep learning (e.g., object detection algorithms for onboard cameras) have facilitated their use in AVs [92]. These LESs are safety critical [5], where their failure can result in loss of life, injuries, and financial damage. LESs may not operate as expected if the training data does not sufficiently capture run-time contexts [16]. For example, an AV may be trained with external agents such as pedestrians or other vehicles, whose training behavior conforms to defined rules [93]. In contrast, human agents may exhibit unexpected behaviors (e.g., suddenly braking, unexpected lane changes, etc.), thereby introducing a new source of uncertainty for AVs. We use the term "ego vehicle" to describe the AV under study and "non-ego vehicle" to describe other vehicles. We propose a non-cooperative game theory[1] [42] and RL [94] framework to discover unexpected behavior exhibited by a trained (selfish) non-ego vehicle and the unexpected

[1]Non-cooperative game theory describes a game theory setup where players are only aware of and attempt to optimize their individual objectives [42].

responses from a *naïve*[2] ego vehicle, which can be used to enable the robustification of the ego vehicle to prevent and/or mitigate unsafe situations.

In order to guarantee a high level of safety during operation, LESs must operate safely and correctly in various operating contexts (i.e., environmental conditions, pedestrians, obscured lane markings, etc.). Existing research has largely addressed well-defined uncertainties and their effects on LESs, including environmental uncertainties [13, 14, 16] and adversarial examples [18, 95]. In contrast, human behaviors provide a source of uncertainty that cannot be explicitly defined and modeled. Specifically, human behaviors that are often motivated by various personal factors, such as temperament, schedules, attention level, etc., lead to unpredictable actions from the human driver and unknown responses in the AV [96].

This chapter introduces SafeDriveRL, an automated synthetic test data generation and analysis framework used to discover unexpected non-ego behavior, assess the undesirable response of the ego vehicle, and inform the reconfiguration of the ego LES to improve its robustness against the discovered behaviors. In order to achieve these goals, SafeDriveRL contributes three key insights. First, we use non-cooperative game theory to define and model the relationship and interactions between an ego AV and a non-ego human-driven vehicle on the road in order to capture real-world driving scenarios. Second, SafeDriveRL discovers uncertain and unexpected behaviors from non-ego vehicles whose objective is to optimize its human-based objective using RL. Specifically, a developer can declaratively specify different types of human-driver personas [97] to capture the non-ego vehicles' objectives (e.g., aggressive, distracted, overly-cautious drivers, etc.), then RL is used to discover how a non-ego player may learn unexpected maneuvers (i.e., sequence of atomic driving actions) that adversely affect an ego AV's performance, such as swerving, sudden braking, etc. Third, the learned non-ego behavior information can be subsequently used to assess the robustness of the ego vehicle. The discovered information enables the developer to improve or fine-tune the ego vehicle against the unexpected behavior(s) of the non-ego vehicle.

SafeDriveRL uses RL to support non-cooperative gaming between the ego and non-ego

[2]Basic driving behavior with no experience with uncertainty.

vehicles to explore uncertain behavior(s) and discover appropriate mitigation strategies for the ego vehicle. We define a two-player non-cooperative game between the ego and non-ego vehicle, where the players compete to learn their respective optimal policies. The objective of the ego and non-ego vehicle is to complete a navigation task subject to operational and safety constraints. Additionally, the non-ego vehicle must also capture a given human driver persona and unique motivations. SafeDriveRL leverages RL to reduce the search space and solve the two-player game, approximating the best strategies for the players that maximize their respective rewards without changing strategies [42].

Preliminary results indicate that SafeDriveRL can successfully discover previously unknown behaviors that a non-ego vehicle may use to optimize their human-based objective, discover erroneous or undesirable behaviors in the naïve ego vehicle induced by the non-ego vehicle's unexpected maneuvers, and enable the developer to reconfigure or improve the ability of the ego vehicle to handle the uncertainty. We implemented a proof-of-concept prototype for SafeDriveRL and demonstrated our approach on a sample use case scenario using different driver personas. The remainder of the paper is organized as follows. Section 5.2 overviews the methodology used in the proposed approach. Section 5.3 describes the proof-of-concept implementation and demonstration of the SafeDriveRL framework. Section 5.4 discusses related work. Section 5.5 discusses our threats to validity. Finally, Section 5.6 concludes the paper and discusses future directions.

5.2 Methodology

This section introduces the SafeDriveRL framework. Figure 5.1 shows a DFD of SafeDriveRL, where circles denote computational steps, parallel lines represent persistent data stores, and arrows show the data flow. Although SafeDriveRL can be applied to a variety of traffic scenarios, we use a *merge* scenario as the running example to illustrate the key elements of our framework.

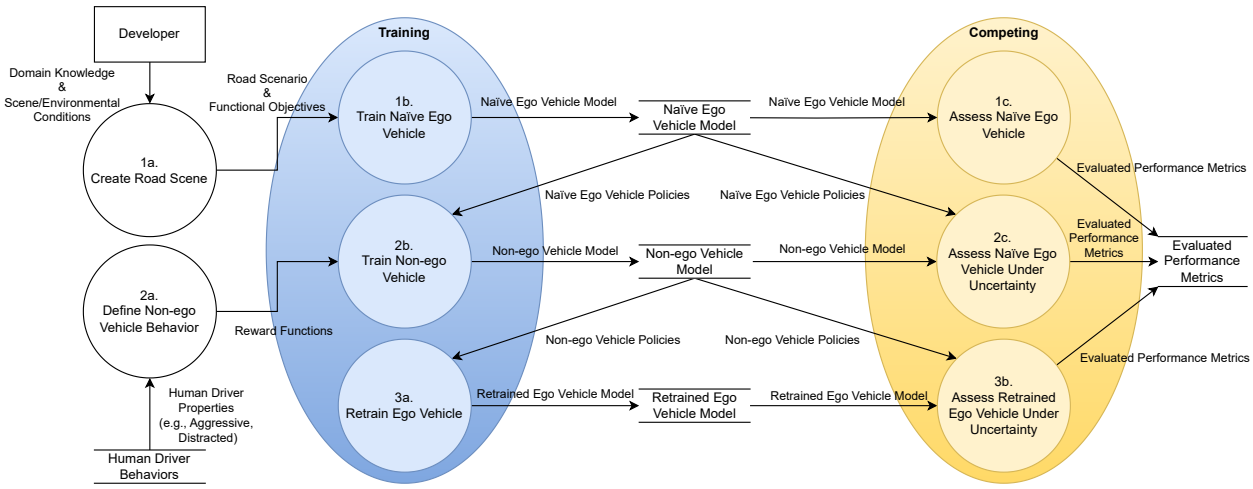


Figure 5.1 A high-level data flow diagram for the SAFEDRIVERL framework.

5.2.1 Preliminaries

SAFEDRIVERL uses a non-cooperative game-theoretic formulation of vehicles and their interactions to discover uncertain human behaviors in a simulation environment. We consider a two-player game between an ego vehicle and a non-ego vehicle. This game comprises a player set I, each player's strategy space S_i, and a payoff function π^i, where the term i corresponds to player i [98, 99]. This game-theoretic formulation is realized as an RL system. Each player i uses a DNN, $f^i : \texttt{state} \rightarrow \texttt{action}$, for decision-making, where f^i is a real-valued function that maps the current state (i.e., internal state such as position, velocity, and environment state such as external agents, roadways etc.) of the game to an action (i.e., actuator inputs such as acceleration, braking, and steering). Specifically, we consider the learned function f^i to represent the strategy s_i of player i [100]. Each player i is only aware of their own objective and the state of the current environment. Thus, player i in our non-cooperative game seeks to find a "best" strategy s_i^* that maximizes *individual* payoff $\pi^i(s_i)$. The payoff function π^i is constructed based on the weighted summation of a given player i's safety constraints (e.g., traffic rule violations, distance to other vehicles), operational constraints (e.g., offset from optimal navigation path), and human-based constraints (e.g., distracted driver behaviors) (Expression 5.1), aggregated over each timestep over an entire game [98].

$$\pi^i(s_i) = \alpha_0 \mathcal{R}_{\text{safety}}(i) + \alpha_1 \mathcal{R}_{\text{operational}}(i) + \alpha_2 \mathcal{R}_{\text{human}}(i) \tag{5.1}$$

The functions $\mathcal{R}$ are real-valued functions that map a player i's state to an RL reward value based on behavioral constraints (e.g., behaviors of an aggressive driver). The coefficients $\{\alpha_k : 0 \leq k \leq 2\}$ can be tuned by developers or generated from analyzing real-world driving data. After training both players, we consider the resulting players' DNNs to have approximated the best strategy selection s^*, where $\forall i \in I, \forall s_i \in S_i$:

$$\pi^i(s^*) \geq \pi^i(s_i, s^*_{-i}) \quad \text{[99]} \tag{5.2}$$

(s_i, s^*_{-i}) denotes a strategy selection where player i is the only player that does not use a strategy from s^* (i.e., any change in strategy for player i, assuming all other players use s^*, will result in a lower payoff) [99].

5.2.2 Step 1: Initialization

SafeDriveRL assesses the ability of a naïve ego vehicle to handle human-induced uncertainty and discovers undesirable behaviors. As such, **Step 1** involves training or creating a base player. **Step 1** is analogous to a novice driver who has just completed their driver's education course. This driver has learned basic driving maneuvers and can successfully drive from one location to another designated location with minimal failures (e.g., crashes). However, the novice driver has not been exposed to a variety of behaviors from other road users, including uncertain or dangerous behaviors. In SafeDriveRL, a developer can use different sources for the naïve ego vehicle, such as traditional software implementation of Advanced Driver-Assistance System features, RL models, or a digital twin that reflects the behavior of an AV under study [101, 102, 103].

Step 1a. Create Road Scene. Developers must first specify the road scene and simulation environment for a given scenario. Namely, a developer should specify the road network configuration (e.g., road edges), initial vehicle configurations (e.g., locations, velocity, orientation), and environment parameters (e.g., speed limit, time delta). Figure 5.2 shows an example road scene setup for

a lane changing merge scenario, where a non-ego vehicle (denoted in orange) is driving alongside an ego vehicle (denoted in blue) on a two-lane road that converges to a single-lane, thereby forcing the non-ego vehicle to merge into the ego vehicle's lane.

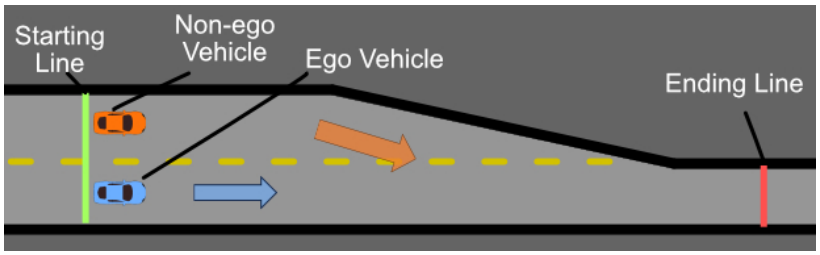


Figure 5.2 Graphical depictions of a merge road scenario.

Step 1b. Develop Naïve Ego Vehicle. The naïve ego-vehicle is developed (e.g., using rule-based behavior or is trained with RL) with the objective of completing the basic scenario. For example, an RL-trained ego vehicle is trained to complete the navigation task while optimizing a safety-oriented reward function. The ego vehicle seeks to maintain a safe distance from other vehicles in the environment, avoid collisions with other objects and road boundaries, and navigate to the desired location.

Step 1c. Assess Naïve Ego Vehicle Performance. The final initialization step assesses the naïve ego vehicle's performance using a set of performance metrics defined by the developers. For example, a developer may assess the average speed of the naïve ego vehicle, average trailing distance, number of crashes, etc. These values provide a basis for performance evaluation in the subsequent steps.

5.2.3 Step 2: Training the Non-ego Vehicle

The goal of this step is to identify two types of uncertain behaviors: unexpected maneuvers that a non-ego vehicle may exhibit on the road and unexpected responses from the naïve ego vehicle. This step captures Round 1 of the gaming setup, including both the training phase (of non-ego vehicle) and the competing phase (between ego and non-ego vehicles) to identify respective unexpected behaviors.

Step 2a. Define Non-ego Vehicle Behavior. In order to model and simulate human driver maneuvers, the developers define an RL reward function for the non-ego vehicle based on existing human driver behavior models. For example, the developer may be interested in the AV's behavior in the presence of fast or distracted drivers. As such, a developer may define a reward function that promotes the non-ego vehicle to merge into the naïve ego vehicle's lane, while minimizing the time required to perform the maneuver. In contrast to existing non-cooperative game theory approaches where the malicious intent of the non-ego vehicle is to minimize the naïve ego vehicle's reward [104], the non-ego vehicle in SafeDriveRL only seeks to maximize its own reward. This approach closely captures the relationship between road users in practice, as drivers seek to prioritize their self-serving goals but do not typically maliciously sabotage the objectives of other vehicles.

Step 2b. Train Non-ego Vehicle Next, the non-ego vehicle is trained using RL to discover potential behaviors (e.g., maneuvers) that it may employ to maximize the reward objectives. Specifically, RL simulates and enables a non-ego vehicle to explore and test a variety of maneuvers over many episodes (i.e., instances of the road scenario). During each episode, the non-ego vehicle explores a number of states, or road scenario configurations (e.g., positions, speeds of the vehicle, etc.) at a given timestep, and approximates the best next action. By disregarding maneuvers that result in low rewards and learning maneuvers that result in high rewards, RL discovers an optimal policy for the non-ego vehicle to maximize the given human reward function based on the defined driver persona. The policy learned by the non-ego vehicle represents a mechanism for selecting optimal actions based on the state of the environment at a given timestep. For example, a non-ego vehicle may learn over time to quickly (and aggressively) merge in front of the naïve ego vehicle in order to complete its navigation task in minimal time.

Step 2c. Assess Naïve Ego Vehicle Under Uncertainty. This step assesses the performance of the naïve ego vehicle as it operates in the context of the non-ego vehicle. We use the vehicle evaluation metrics defined in **Step 1** to evaluate both the naïve ego and non-ego vehicles. By analyzing how the behavior of the naïve ego vehicle changes/reacts in response to the presence

of the non-ego vehicle, a developer can discover and assess the effects of maneuvers used by the non-ego vehicle that impact the safety or performance of the naïve ego vehicle. For example, the aggressive behavior learned by the non-ego vehicle may trigger a previously unknown response from the naïve ego vehicle (e.g., abruptly braking or swerving).

5.2.4 Step 3: Retraining the Ego Vehicle

The final step of SafeDriveRL addresses the ability of the retrained ego vehicle to operate safely in the presence of uncertain human behaviors exhibited by the non-ego vehicle. Analogously, the novice driver gained diverse driving experiences and exposure to different types of uncertain driver behaviors. As the novice driver gains on-road experience, they learn to drive defensively in the presence of other road users, including maneuvers that can reduce the chance of collisions in response to the uncertainties. **Step 3** represents Round 2 of the gaming process, where the ego vehicle is retrained with the (fixed) non-ego vehicle and then the retrained ego competes with the non-ego vehicle.

Step 3a. Retrain Ego Vehicle. This step retrains the ego vehicle in an environment with the non-ego vehicle from **Step 2**. Specifically, the *retraining scenario* involves a retrained ego vehicle learning to complete the intended navigation tasks (e.g., driving from the starting line to the ending line) in the presence of uncertainty posed by the non-ego vehicle. The objective of the non-ego vehicle in this step is to complete the navigation task using the policy learned in **Step 2**, while the objective of the retrained ego vehicle is to robustify its policy by learning new actions associated with states that result from the non-ego vehicle's unexpected behaviors. With RL, the retrained ego vehicle learns new defensive maneuvers, such as preemptively reducing their speed, to improve the performance and safety of its navigation objectives.

Step 3b. Assess Retrained Ego Vehicle Under Uncertainty. Finally, the retrained ego vehicle is re-assessed to compare the performance of the naïve ego vehicle versus the retrained ego vehicle in the presence of the non-ego vehicle. This metric shows whether the retrained ego vehicle is better at safely handling uncertain maneuvers exhibited by the non-ego vehicle.

5.3 Demonstrations

This section describes a proof-of-concept use case for SafeDriveRL. We overview our simulation environment and experimental setup. We discuss the implementation details for the driving scenarios, RL setup, and the results of the experiments.

5.3.1 Simulation Environment

SafeDriveRL requires a simulation environment to model the vehicles and their interactions in order to discover undesirable behavior using non-cooperative game theory and RL. While SafeDriveRL can be applied to any simulation environment that accepts player actions and returns environment states, we developed TinyRoad, a modular 2D simulation environment that models the behavior of vehicle interactions with an emphasis on computational efficiency to demonstrate the feasibility of our approach. TinyRoad takes a configuration file for each scenario. The configuration file comprises all relevant information for instantiating a given scenario, such as the road environment setup (i.e., road edge specification).

5.3.2 Experimental Setup

To evaluate the ability of SafeDriveRL to discover erroneous behavior in the naïve ego vehicle induced by human behaviors, we applied SafeDriveRL with TinyRoad to demonstrate our example use cases. In order to discover the learned policies for both vehicles, we implement a standard deep reinforcement learning (i.e., RL with a DNN as a controller) environment using OpenAI Gym [105]. Specifically, players use an actor-critic-based DNN for decision-making in the environment [106, 107, 108]. The position and velocity of both vehicles are randomly initialized for repeated episodes. While SafeDriveRL accepts any configuration of driver persona as defined by the developer, this work shows two different personas for demonstration purposes. The selected personas are based on the most common types of dangerous driving behaviors from a report by the AAA Foundation for Traffic Safety [109]. The parameters used for RL training are shown in Table 5.1. All experiments were conducted using Python on a computer running Ubuntu 20.04, with 32GB of RAM, Intel I7 CPU, and an NVIDIA GTX 3060 GPU.

For each use case, we address the following research questions and define our null (i.e., H_0)

and alternate (i.e., H_1) hypotheses as follows. To answer our research questions, we describe our numerical results and demonstrate the statistical significance using the paired-samples one-tailed *t*-test [13]. Finally, a demonstration package with trained models is provided for validation of our results.

Research Question

RQ 5.1: Can SAFEDRIVERL train a non-ego vehicle model based on human objectives that induce failures in the ego vehicle compared to the baseline scenario?

$H_0(\mu_{\text{diff}} = 0)$: There is no difference in ego performance.

$H_1(\mu_{\text{diff}} > 0)$: There is a difference in ego performance.

RQ 5.2: Can we use the non-ego vehicle model trained by SAFEDRIVERL to improve the ability of the ego vehicle to operate in the face of the discovered uncertainty?

$H_0(\mu_{\text{diff}} = 0)$: There is no difference between a reconfigured ego vehicle and the naïve ego vehicle.

$H_1(\mu_{\text{diff}} > 0)$: There is a difference between a reconfigured ego vehicle and the naïve ego vehicle

Table 5.1 Overview of RL training parameters.

Hyperparameters	**Value**	**Hyperparameters**	**Value**
RL Training Steps	3×10^6	Clip ϵ	0.1
Early convergence	True	Entropy Coefficient	0.02
Optimizer	Adam	Learning Rate	1×10^{-4}
Discount Factor γ	0.99	Replay Buffer Size	2048

5.3.3 Use case study: Road Merge

Figure 5.2 shows a graphical example of the merge scenario in the use case, where a two-lane road *converges* into a single lane. As the orange non-ego vehicle approaches the end of its lane, it seeks to merge into the blue ego vehicle's lane. In this scenario, a naïve ego vehicle is trained to drive safely from the green starting line to the red ending line using RL. Next, a non-ego vehicle is trained to merge into the naïve ego vehicle's lane with a given human behavior model. The following subsections describe two non-ego vehicle behavior models: aggressive/speedy and distracted. Finally, the retrained ego vehicles assess whether SAFEDRIVERL can improve the robustness of the naïve ego vehicle against the newly discovered uncertainties.

5.3.3.1 Road Merge: Aggressive/Speedy Driver

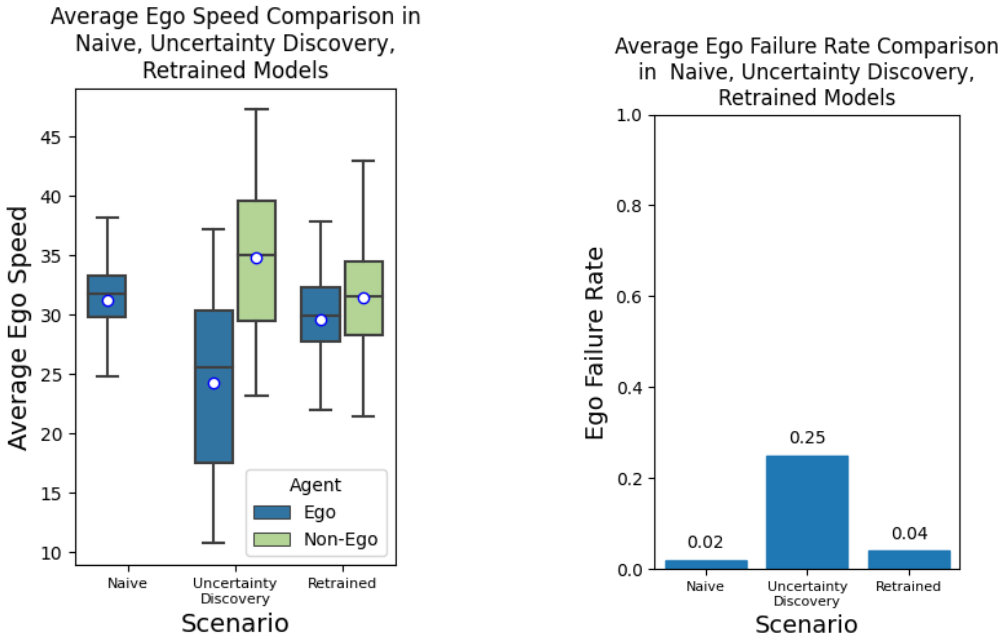
This experiment explores the interaction between a speedy driver and the naïve ego vehicle. Traffic data (i.e., fatal accident causes, driver behavior) reported by the NHTSA [110] and the AAA Foundation for Traffic Safety [109] were used to inform the reward structure. For example, as a speedy driver often does not follow the local speed limit, thus we introduced a fuzzy logic function to promote the speedy non-ego vehicle to exceed the speed limit appropriately. To address our research questions, we evaluate the ability of SafeDriveRL to train a non-ego vehicle model with human-based behaviors and assess if the data can be used to improve the robustness of the ego vehicle.

Figure 5.3 shows a graphical comparison of the average speed (a) and failure rate (b) over 100 episodes. In the presence of the speedy driver, the naïve ego vehicle experiences a high failure rate and shows a slower average speed. Specifically, the presence of the non-ego vehicle induces the naïve ego vehicle to crash 25% of the episodes. We found strong evidence using the paired-samples one-tailed *t*-test (with $p < 0.01$) to reject H_0 and support H_1 for ***RQ 5.1***. In contrast, the retrained ego vehicle demonstrates improved performance, returning to a similar level of speed and failure rate as the scenario without the non-ego vehicle. The results indicate that SafeDriveRL can discover human-induced maneuvers for the non-ego vehicle that degrade the naïve ego vehicle's performance. The maneuvers discovered by SafeDriveRL informed a developer reconfiguration of the naïve ego vehicle that was then able to successfully mitigate the maneuvers of the non-ego vehicle. We found strong evidence ($p < 0.01$) to reject H_0 and support H_1 for ***RQ 5.2***.

5.3.3.2 Road Merge: Distracted Driver

The second player explored in this work is a *distracted driver persona*. In 2021, NHTSA reported that 3,522 accidents were due to distracted driving [111]. A non-ego vehicle is trained using the same rewards provided to the naïve ego vehicle. To capture distracted driving behavior, the non-ego vehicle enters a distracted state and omits atomic actions from the RL model during random intervals, based on NHTSA claims that indicate viewing a text message takes about 5 seconds [111]. Figure 5.4 shows a sample episode of the distracted merge scenario, where a

non-ego driver enters a distracted state as their vehicle is drifting sideways towards the naïve ego vehicle. As the naïve ego vehicle seeks to maintain a large distance between itself and nearby objects, it swerves away and collides with the guard rail.



(a) Average speed.

(b) Average failure rate.

Figure 5.3 Comparison of the naïve ego vehicle and retrained vehicle in the presence of the speedy non-ego vehicle.

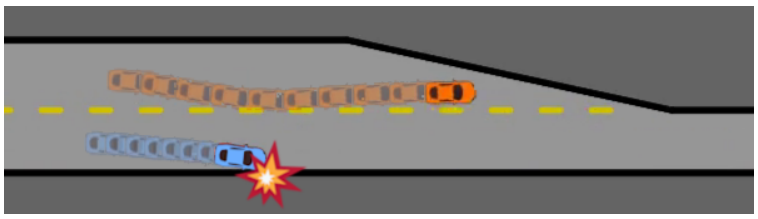
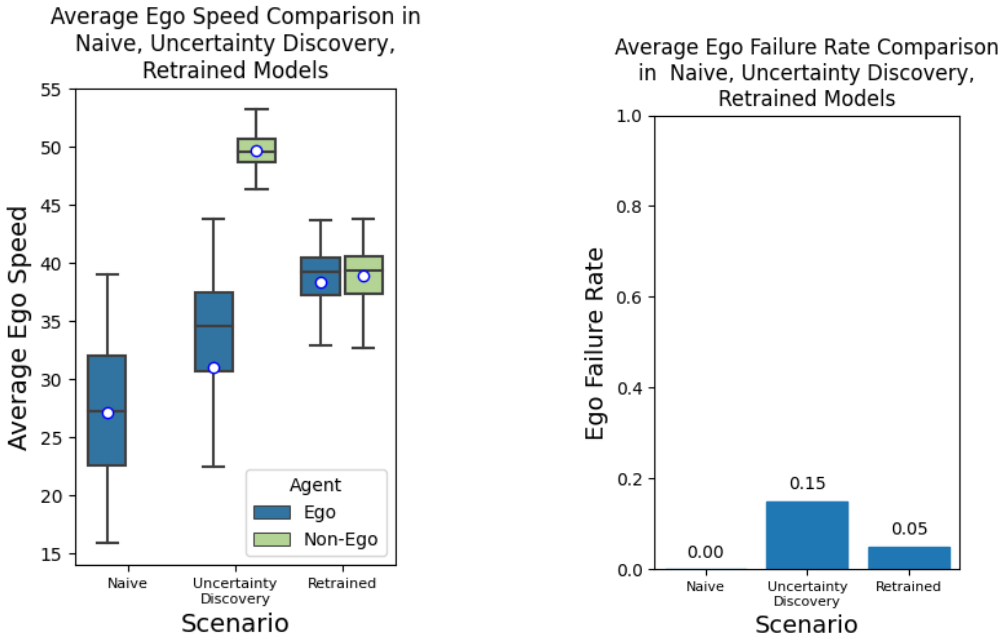


Figure 5.4 Example of discovered *uncertain* maneuver.

Figure 5.5 shows a graphical comparison of the average speed (a) and failure rate (b) over 100 episodes. In the presence of the distracted driver, the naïve ego vehicle's failure rate increases from zero to 15% of episodes. We found strong evidence ($p < 0.01$) to reject H_0 and support H_1 for *RQ 5.1* of the distracted driver merge scenario. After the ego vehicle is retrained with the distracted non-ego vehicle, the retrained ego vehicle is capable of reducing the failure rate to 5%. We also found strong evidence ($p < 0.01$) to reject H_0 and support H_1 for *RQ 5.2*. We found that

the retrained scenarios include scenarios where the distracted driver drives directly into the ego vehicle, causing a failure. This experiment shows that a retrained ego vehicle is capable of learning defensive maneuvers to mitigate the uncertain maneuvers of the distracted non-ego vehicle.



(a) Average speed.

(b) Average failure rate.

Figure 5.5 Comparison of the naïve ego vehicle and retrained ego vehicle in the presence of the distracted non-ego vehicle.

5.4 Related Work

This section overviews related work for RL and applications of game theory related to AVs. To the best of our knowledge, SafeDriveRL is the first to address uncertain human driver behavior using non-cooperative game theory and RL for AVs.

In order to study the strategic interactions between AVs and other road users, a number of researchers have modeled their behavior using game theory. Banjanovic-Mehmedovic *et al.* [112], Michieli and Badia [113], and Wei *et al.* [114] proposed several game theory modeling approaches for AVs and surrounding objects (e.g., pedestrians, road users, other AVs, etc.). Xue *et al.* [115] proposed a game theory approach to address the leader-follower problem for multiple AVs. Bui *et al.* [116] and Wang *et al.* proposed a cooperative game theory approach to improve traffic flows for intersections. Liniger *et al.* [117] defined a non-cooperative game theory approach for autonomous

racing games, but both players share the same reward function. Our approach focuses on the interaction between a variety of human driver behaviors (modeled as a player) and the AV (modeled as the other player), where the two players do not share identical objectives to discover potential strategies that human drivers may exploit against the AV.

Finally, several researchers have explored the use of RL to train or improve the performance of AVs. Ko *et al.* [118] introduced an RL approach to learn how a vehicle system can share driving authority between a human and a vehicle. Bouton *et al.* [119] proposed using RL to learn how AVs can navigate dense merging scenarios with various "levels of cooperation" from other drivers. Gupta *et al.* [104] introduced the PAIN framework to train two dueling RL players, but do not leverage a game theory approach. Our approach uses RL in order to generate road scenario edge cases to test and improve the ego model's behavior using non-cooperative game theory.

5.5 Threats to Validity

This chapter shows that RL and non-cooperative game theory can be synergistically integrated to identify undesirable behaviors in an ego vehicle induced by human uncertainties. The results of the experiments may vary with repeated experiments, as the training of RL rely on a trial-and-error approach using non-determinism, and the decisions of RL are based on a probabilistic estimate to maximize the reward function. Additionally, the learned behavior of the driver may not fully capture human driver behavior. Changes in operational context may require restructuring the reward function. Finally, we acknowledge the possible discrepancies between behaviors observed in simulation and reality (i.e., "reality gap").

5.6 Conclusion

This chapter introduced SafeDriveRL, a non-cooperative game theory and RL approach to discover and mitigate unexpected AV behaviors induced by *uncertain and unexpected* human behaviors. We demonstrated that a non-ego vehicle optimizing their reward functions can learn policies that may leverage or exploit an ego AV's safety properties. Preliminary results show that SafeDriveRL can create more robust systems capable of handling a variety of representative human driving behaviors. Future work will explore how SafeDriveRL can be extended to assess and

improve the robustness of AVs with other external actors (e.g., pedestrians, cyclist, obstacles, etc.), environmental factors, and infrastructures. Additional studies may combine existing technologies for procedurally-generated road scenes in order to automatically test an AV system (e.g., a digital twin [101, 102, 103]) under a variety of road scenarios.

CHAPTER 6

CONCLUSION AND REMAINING INVESTIGATIONS

This chapter summarizes the preliminary investigations for this dissertation proposal and presents our remaining investigations.

6.1 Summary of Preliminary Findings

Recent advances in DNNs have demonstrated impressive capabilities for traditionally difficult software engineering tasks, promoting their use in many real-world applications. Applications that involve humans are often safety-critical, and thus require stakeholder confidence that the LES will be safe and correct during run time. However, humans and interactions with humans in the environment pose a large source of uncertainty, involving a range of actions that are malicious and actions that are non-malicious yet unpredictable. As a result, researchers are exploring new techniques and approaches to discover and assess the robustness of LES against human uncertainty. Discovering undesirable LES behaviors or interactions with humans enables developers to better design mitigation strategies or introduce safeguards to minimize harm to surrounding humans.

Motivated by challenges for LES against exploitative uncertainty, this dissertation proposal and preliminary work described the development of automated methods to discover test cases for LESs and assess their robustness. We introduced techniques that tackled the problem from two distinct perspectives: the type of exploitative uncertainty and the type of test cases to be produced. Chapter 3 and Chapter 4 explored malicious exploitative uncertainty, where adversarial noise is used to hinder the performance of LECs. We demonstrated the efficacy of our approaches on several state-of-the-art models and datasets, including those that involve safety-critical applications (e.g., Waymo driving data and VisDrone aerial drone data). Chapter 5 explored a game-based testing approach to uncover selfish and incidental exploitative uncertainty, where the human agent's behavior indirectly affects the LES's performance. We demonstrate that a game-based testing approach can be used to discover unintentional maneuvers that an AV cannot safely handle.

6.2 Remaining Investigations

The remaining investigations for this doctoral work will focus on developing automated testing frameworks to supplement and complement our preliminary findings by addressing diversity-driven incidental exploitative uncertainty. Figure 6.1 overviews the remaining investigations proposed in this manuscript, based on the EUREKA framework described in Section 1. Specifically, we focus on investigations to address the remaining quadrant of the exploitative uncertainty (dashed regions). We also plan on investigating a model-driven approach (denoted by hashed items) to better define RL agent reward functions, enabling a systemic approach to refine and reuse agent objectives. A model-driven approach decreases developer overhead in the definition of the low-level reward function, thereby facilitating reuse and design at the model level. Specifically, we propose the following remaining investigations:

I. [Generalizable perturbations using EXPOUND] Extend our work with EXPOUND to assess whether novelty search can be used to identify generalizable or reusable perturbations for DNNs that can be used to deceive the model to incorrectly classify any input image as a given adversarial label.

II. [Model-based approach for game-based testing] Investigate whether a goal-based model-driven approach can enable a systematic means to define and decompose high-level non-functional agent objectives into low-level system requirements, which are then discharged to system components to assess for satisficement, thereby forming the reward function of the agent.

III. [Diversity-driven test case generation for incidental exploitative uncertainty] Explore how search-based approaches can be harnessed to automatically generate diverse test cases to assess LES robustness against incidental exploitative uncertainty.

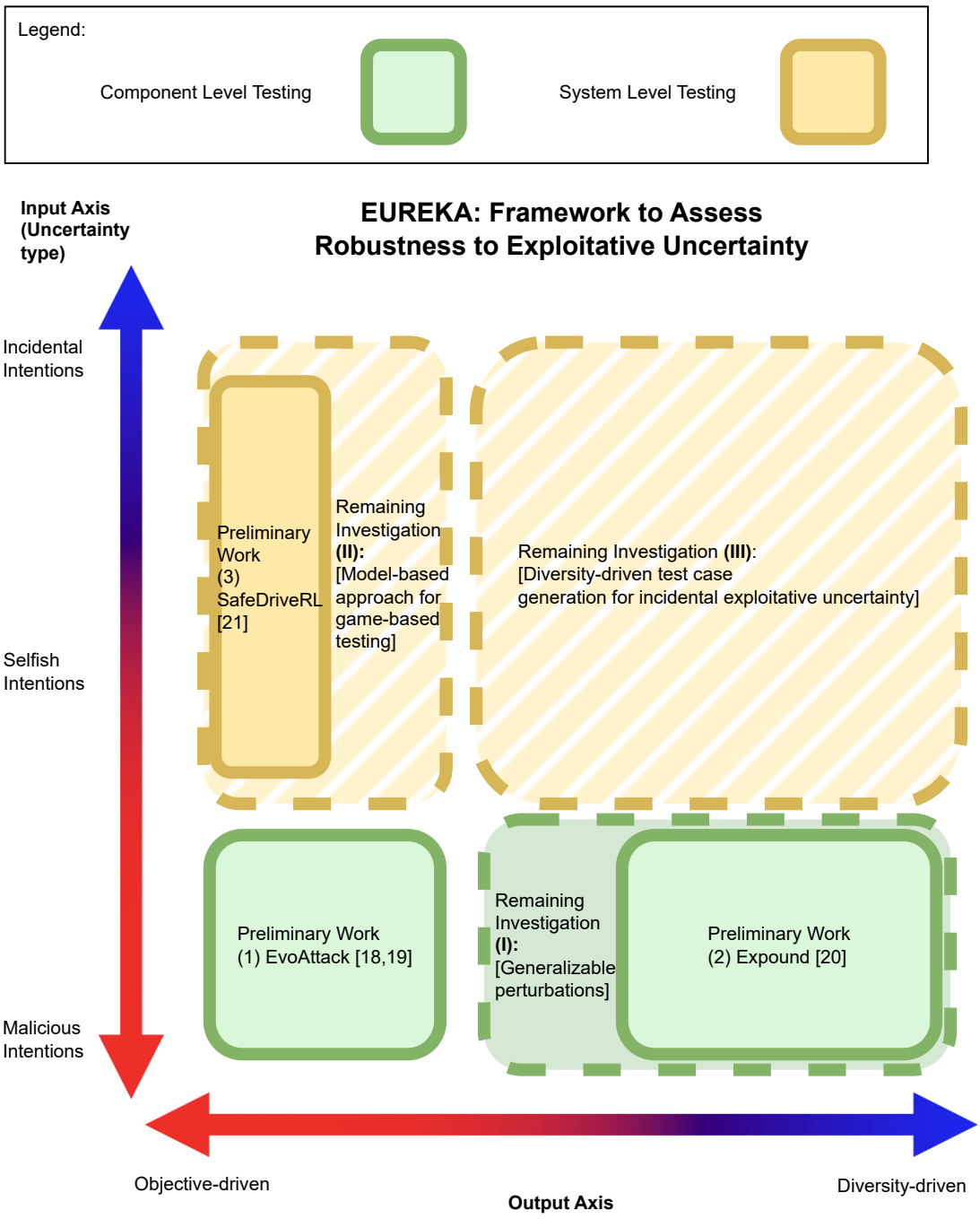


Figure 6.1 Remaining investigations for this dissertation proposal. Quadrants for remaining investigations are represented by dashed regions. Proposed model-based approaches are denoted in hashed fill.

APPENDIX

MANUSCRIPTS AND PUBLICATIONS

Kenneth H Chan and B.H.C. Cheng. EvoAttack: Suppressive adversarial attacks against object detection models using evolutionary search. *Automated Software Engineering*, 2024. Accepted to Special Issue on Advances in Search-based Software Engineering

Kenneth H Chan, Sol Zilberman, Nick Polanco, Joshua E Siegel, and B.H.C. Cheng. SafeDriveRL: Combining non-cooperative game theory with reinforcement learning to explore and mitigate human-based uncertainty for autonomous vehicles. In *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 214–220, 2024

Kenneth H Chan and B.H.C. Cheng. Expound: A black-box approach for generating diversity-driven adversarial examples. In *International Symposium on Search Based Software Engineering*, pages 19–34. Springer, 2023

Michael Austin Langford, **Kenneth H Chan**, Jonathon Emil Fleck, Philip K McKinley, and B.H.C. Cheng. MoDALAS: addressing assurance for learning-enabled autonomous systems in the face of uncertainty. *Software and Systems Modeling*, pages 1–21, 2023

Kenneth H Chan and B.H.C. Cheng. EvoAttack: An evolutionary search-based adversarial attack for object detection models. In *Proceedings of the 14th IEEE Symposium on Search-Based Software Engineering*, Singapore, 2022

Michael Austin Langford, **Kenneth H Chan**, Jonathon Emil Fleck, Philip K McKinley, and B.H.C. Cheng. MoDALAS: Model-driven assurance for learning-enabled autonomous systems. In *Proceedings of MODELS 2021: ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 207–216, Fukuoka, JP, 2021. Model Driven Engineering Languages and Systems. (Extended paper invited for special issue journal submission to Software and Systems Modeling (SoSyM))

Kenneth H Chan, Matthew Pasco, and B.H.C. Cheng. Towards a blockchain framework for autonomous vehicle system integrity. *SAE International Journal of Transportation Cybersecurity and Privacy Special Issue on System Safety and Cybersecurity*, 4(11-04-01-0002), 2021

BIBLIOGRAPHY

[1] Lin et al., "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[2] 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.

[3] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, Y. Kwon, K. Michael, J. Fang, Z. Yifu, C. Wong, D. Montes, and others, "ultralytics/yolov5," *Zenodo*, 2022.

[4] Y. Cao, Z. He, L. Wang, W. Wang, Y. Yuan, D. Zhang, J. Zhang, P. Zhu, L. Van Gool, J. Han, and others, "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.

[5] J. C. Knight, "Safety critical systems: challenges and directions," in *Proceedings of the 24th International Conference on Software Engineering. ICSE 2002*, pp. 547–550, 2002.

[6] Z. Cai *et al.*, "A unified multi-scale deep convolutional neural network for fast object detection," in *ECCV 2016: 14th European Conf., Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 354–370, Springer, 2016.

[7] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," *Advances in neural information processing systems*, vol. 26, 2013.

[8] W. Wachenfeld and H. Winner, "The release of autonomous vehicles," *Autonomous Driving: Technical, Legal and Social Aspects*, pp. 425–449, 2016.

[9] Z. Aghababaeyan, M. Abdellatif, M. Dadkhah, and L. Briand, "Deepgd: A multi-objective black-box test selection approach for deep neural networks," *arXiv*, 2023.

[10] F. Yu, Z. Qin, C. Liu, L. Zhao, Y. Wang, and X. Chen, "Interpreting and Evaluating Neural Network Robustness," 2019. Paper presented at 28th International Joint Conf. on Artificial Intelligence (IJCAI 2019).

[11] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th international conference on software engineering*, pp. 303–314, 2018.

[12] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore, "Testing deep neural networks," *arXiv preprint arXiv:1803.04792*, 2018.

[13] M. A. Langford and B.H.C. Cheng, "Enki: a diversity-driven approach to test and train robust learning-enabled systems," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 15, no. 2, pp. 1–32, 2021.

[14] M. A. Langford and B.H.C. Cheng, ""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, IEEE, 2021.

[15] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *proceedings of the 26th Symposium on Operating Systems Principles*, pp. 1–18, 2017.

[16] M. A. Langford, K. H. Chan, J. E. Fleck, P. K. McKinley, and B.H.C. Cheng, "Modalas: addressing assurance for learning-enabled autonomous systems in the face of uncertainty," *Software and Systems Modeling*, pp. 1–21, 2023.

[17] W. R. Adrion, M. A. Branstad, and J. C. Cherniavsky, "Validation, verification, and testing of computer software," *ACM Computing Surveys (CSUR)*, vol. 14, no. 2, pp. 159–192, 1982.

[18] K. Chan and B.H.C. Cheng, "Evoattack: An evolutionary search-based adversarial attack for object detection models," in *Search-Based Software Engineering: 14th International Symposium, SSBSE 2022, Singapore, November 17–18, 2022, Proceedings*, pp. 83–97, Springer, 2022.

[19] K. H. Chan and B.H.C. Cheng, "Evoattack: suppressive adversarial attacks against object detection models using evolutionary search," *Automated Software Engineering*, vol. 32, no. 1, p. 3, 2025.

[20] K. H. Chan and B.H.C. Cheng, "Expound: A black-box approach for generating diversity-driven adversarial examples," in *International Symposium on Search Based Software Engineering*, pp. 19–34, Springer, 2023.

[21] K. H. Chan, S. Zilberman, N. Polanco, J. E. Siegel, and B.H.C. Cheng, "Safedriverl: Combining non-cooperative game theory with reinforcement learning to explore and mitigate human-based uncertainty for autonomous vehicles," in *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 214–220, 2024.

[22] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM computing surveys (CSUR)*, vol. 45, no. 3, pp. 1–33, 2013.

[23] M. A. Nielsen, "Neural networks and deep learning," 2015. Volume: 25.

[24] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[25] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[26] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[27] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable ai: A review of machine learning interpretability methods," *Entropy*, vol. 23, no. 1, p. 18, 2020.

[28] A. Der Kiureghian and O. Ditlevsen, "Aleatory or epistemic? does it matter?," *Structural safety*, vol. 31, no. 2, pp. 105–112, 2009.

[29] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods," *Machine learning*, vol. 110, no. 3, pp. 457–506, 2021.

[30] J. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *Proceedings of the 5th international conference on learning representations*, 2017.

[31] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*, pp. 39–57, IEEE, 2017.

[32] A. Serban, E. Poll, and J. Visser, "Adversarial examples on object recognition: a comprehensive survey," *Acm Computing Surveys*, vol. 53, June 2020. Number of pages: 38 Place: New York, NY, USA Publisher: Association for Computing Machinery tex.articleno: 66 tex.issue_date: June 2020.

[33] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2805–2824, 2019.

[34] Szegedy et al., "Intriguing properties of neural networks," in *International conference on learning representations*, 2014.

[35] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International conference on learning representations*, 2015.

[36] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille, "Adversarial examples for semantic segmentation and object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 1369–1378, 2017.

[37] P. R. Srivastava and T.-h. Kim, "Application of genetic algorithm in software testing," *International Journal of software Engineering and its Applications*, vol. 3, no. 4, pp. 87–96, 2009.

[38] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary computation*, vol. 19, no. 2, pp. 189–223, 2011.

[39] J. Lehman and K. O. Stanley, "Novelty search and the problem with objectives," *Genetic programming theory and practice IX*, pp. 37–56, 2011.

[40] M. J. Osborne *et al.*, *An introduction to game theory*, vol. 3. Oxford university press New York, 2004.

[41] J. M. Staatz, "The cooperative as a coalition: a game-theoretic approach," *American Journal of Agricultural Economics*, vol. 65, no. 5, pp. 1084–1089, 1983.

[42] J. Nash, "Non-cooperative games," *Annals of Mathematics*, vol. 54, no. 2, pp. 286–295, 1951.

[43] S. Dodge and L. Karam, "A study and comparison of human and deep learning recognition performance under visual distortions," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–7, 2017.

[44] K. Ren, T. Zheng, Z. Qin, and X. Liu, "Adversarial attacks and defenses in deep learning," *Engineering*, vol. 6, no. 3, pp. 346–360, 2020. Publisher: Elsevier.

[45] G. Nandita and T. M. Chandra, "Malicious host detection and classification in cloud forensics with DNN and SFLO approaches," *International Journal of System Assurance Engineering and Management*, pp. 1–13, 2021. Publisher: Springer.

[46] E. M. Rudd, R. Harang, and J. Saxe, "MEADE: Towards a malicious email attachment detection engine," in *2018 IEEE international symposium on technologies for homeland security (HST)*, pp. 1–7, 2018.

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

[48] Y. Wang and W. Xu, "Leveraging deep learning with LDA-based text analytics to detect automobile insurance fraud," *Decision Support Systems*, vol. 105, pp. 87–95, 2018.

[49] J. Kocić, N. Jovičić, and V. Drndarević, "An end-to-end deep neural network for autonomous driving designed for embedded automotive platforms," *Sensors*, vol. 19, no. 9, p. 2064, 2019. Publisher: Multidisciplinary Digital Publishing Institute.

[50] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, "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.

[51] 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.

[52] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "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.

[53] X. Wei, S. Liang, N. Chen, and X. Cao, "Transferable adversarial attacks for image and video object detection," 2019. ISBN: 9780999241141 Number of pages: 7 Pages: 954–960 Place: Macao, China Publication title: Proceedings of the 28th international joint conference on artificial intelligence Series: IJCAI'19.

[54] Y. Wang, Y.-a. Tan, W. Zhang, Y. Zhao, and X. Kuang, "An adversarial attack on DNN-based black-box object detectors," *Journal of Network and Computer Applications*, vol. 161, p. 102634, 2020.

[55] S. Sivanandam and S. Deepa, "Introduction to genetic algorithms," 2008.

[56] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[57] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[58] M. Alzantot, Y. Sharma, S. Chakraborty, H. Zhang, C.-J. Hsieh, and M. B. Srivastava, "Genattack: Practical black-box attacks with gradient-free optimization," in *Proceedings of the genetic and evolutionary computation conference*, pp. 1111–1119, 2019.

[59] P. Vidnerová and R. Neruda, "Vulnerability of classifiers to evolutionary generated adversarial examples," *Neural Networks*, vol. 127, pp. 168–181, 2020.

[60] Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach *et al.*, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[61] H. Xu, Y. Ma, H.-C. Liu, D. Deb, H. Liu, J.-L. Tang, and A. K. Jain, "Adversarial attacks and defenses in images, graphs and text: A review," *International Journal of Automation and Computing*, vol. 17, no. 2, pp. 151–178, 2020. Publisher: Springer.

[62] D. Kaszas and A. C. Roberts, "Comfort with varying levels of human supervision in self-driving cars: Determining factors in Europe," *International journal of transportation science and technology*, vol. 12, no. 3, pp. 809–821, 2023. Publisher: Elsevier.

[63] K. Otsu, S. Tepsuporn, R. Thakker, T. S. Vaquero, J. A. Edlund, W. Walsh, G. Miles, T. Heywood, M. T. Wolf, and A.-A. Agha-Mohammadi, "Supervised autonomy for communication-degraded subterranean exploration by a robot team," in *2020 IEEE aerospace conference*, pp. 1–9, IEEE, 2020.

[64] M. Schiaretti, L. Chen, and R. R. Negenborn, "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.

[65] J. Han, J. Davids, H. Ashrafian, A. Darzi, D. S. Elson, and M. Sodergren, "A systematic review of robotic surgery: From supervised paradigms to fully autonomous robotic approaches," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 18, no. 2, p. e2358, 2022. Publisher: Wiley Online Library.

[66] K. J. Liang, G. Heilmann, C. Gregory, S. O. Diallo, D. Carlson, G. P. Spell, J. B. Sigman, K. Roe, and L. Carin, "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.

[67] S. Thys, W. Van Ranst, and T. Goedemé, "Fooling automated surveillance cameras: adversarial patches to attack person detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 0–0, 2019.

[68] Z. Wu, S.-N. Lim, L. S. Davis, and T. Goldstein, "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.

[69] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004. Publisher: IEEE.

[70] A. Rozsa, E. M. Rudd, and T. E. Boult, "Adversarial diversity and hard positive generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 25–32, 2016.

[71] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[72] Chen et al., "POBA-GA: Perturbation optimized black-box adversarial attacks via genetic algorithm," *Computers & Security*, vol. 85, pp. 89–106, 2019.

[73] C. Wu, W. Luo, N. Zhou, P. Xu, and T. Zhu, "Genetic algorithm with multiple fitness functions for generating adversarial examples," in *2021 IEEE congress on evolutionary computation (CEC)*, pp. 1792–1799, 2021.

[74] J. K. Han, H. Kim, and S. S. Woo, "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, (New York, NY, USA), pp. 2593–2595, Association for Computing Machinery, 2019. Number of pages: 3 Place: London, United Kingdom.

[75] X. Li, Y. Jiang, C. Liu, S. Liu, H. Luo, and S. Yin, "Playing against deep-neural-network-based object detectors: A novel bidirectional adversarial attack approach," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 1, pp. 20–28, 2021. Publisher: IEEE.

[76] Z. Cai, S. Rane, A. E. Brito, C. Song, S. V. Krishnamurthy, A. K. Roy-Chowdhury, and M. S. Asif, "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.

[77] J. Ye, Y. Wang, X. Zhang, L. Xu, and R. Ni, "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.

[78] J. Sun, W. Yao, T. Jiang, D. Wang, and X. Chen, "Differential evolution based dual adversarial camouflage: Fooling human eyes and object detectors," *Neural Networks*, vol. 163, pp. 256–271, 2023.

[79] R. Lapid and M. Sipper, "Patch of invisibility: Naturalistic black-box adversarial attacks on object detectors," *arXiv preprint arXiv:2303.04238*, 2023.

[80] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002. Publisher: IEEE.

[81] M. Nowostawski and R. Poli, "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.

[82] O. Gheibi, D. Weyns, and F. Quin, "Applying machine learning in self-adaptive systems: A systematic literature review," *ACM TAAS*, vol. 15, no. 3, pp. 1–37, 2021.

[83] Barredo Arrieta *et al.*, "Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI," *Information Fusion*, vol. 58, pp. 82–115, Jun 2020.

[84] E. Wallace *et al.*, "Trick me if you can: Human-in-the-loop generation of adversarial examples for question answering," *TACL*, vol. 7, pp. 387–401, 2019.

[85] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[86] M. Sandler *et al.*, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conf. on CV and pattern rec*, pp. 4510–4520, 2018.

[87] A. Krizhevsky, G. Hinton, and others, "Learning multiple layers of features from tiny images," 2009.

[88] J. Stallkamp *et al.*, "The gtsrb: a multi-class classification competition," in *The 2011 international joint conference on neural networks*, pp. 1453–1460, IEEE, 2011.

[89] L. Sun *et al.*, "A survey of practical adversarial example attacks," *Cybersecurity*, vol. 1, p. 1, 2018.

[90] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv*, 2016.

[91] A. Kurakin *et al.*, "Adversarial attacks and defences competition," in *The NIPS'17 Competition: Building Intelligent Systems*, pp. 195–231, Springer, 2018.

[92] "Ford's BlueCruise Ousts GM's Super Cruise as CR's Top-Rated Active Driving Assistance System." https://www.consumerreports.org/cars/car-safety/active-driving-assistance-systems-review-a2103632203/, May 2023.

[93] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[94] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[95] P. Sharma, D. Austin, and H. Liu, "Attacks on machine learning: Adversarial examples in connected and autonomous vehicles," in *2019 IEEE International Symposium on Technologies for Homeland Security (HST)*, pp. 1–7, IEEE, 2019.

[96] E. M. Hill, L. T. Ross, and B. S. Low, "The role of future unpredictability in human risk-taking," *Human nature*, vol. 8, pp. 287–325, 1997.

[97] S. Blomkvist, "Persona–an overview," *Retrieved November*, vol. 22, p. 2004, 2002.

[98] T. Fujiwara-Greve, "Nash Equilibrium," in *Non-Cooperative Game Theory* (T. Fujiwara-Greve, ed.), Monographs in Mathematical Economics, pp. 23–55, Tokyo: Springer Japan, 2015.

[99] D. Fudenberg and J. Tirole, "Noncooperative game theory for industrial organization: an introduction and overview," *Handbook of industrial Organization*, vol. 1, pp. 259–327, 1989.

[100] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Perolat, D. Silver, and T. Graepel, "A unified game-theoretic approach to multiagent reinforcement learning," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[101] F. Tao, H. Zhang, A. Liu, and A. Y. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on industrial informatics*, vol. 15, no. 4, pp. 2405–2415, 2018.

[102] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the digital twin: A systematic literature review," *CIRP journal of manufacturing science and technology*, vol. 29, pp. 36–52, 2020.

[103] B.H.C.. Cheng, R. J. Clark, J. E. Fleck, M. A. Langford, and P. K. McKinley, "Ac-ros: assurance case driven adaptation for the robot operating system," in *Proceedings of the 23rd acm/ieee international conference on model driven engineering languages and systems*, pp. 102–113, 2020.

[104] P. Gupta, D. Coleman, and J. E. Siegel, "Towards physically adversarial intelligent networks (pains) for safer self-driving," *IEEE Control Systems Letters*, vol. 7, pp. 1063–1068, 2023.

[105] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[106] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[107] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[108] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017.

[109] A. Gross, "Risky business – more than half of all drivers engage in dangerous behavior." https://newsroom.aaa.com/2023/11/risky-business-more-than-half-of-all-drivers-engage-in-dangerous-behavior/, Nov. 2023.

[110] Media, NHTSA, "Driving behaviors reported for drivers and motorcycle operators involved in fatal crashes." `https://www.iii.org/fact-statistic/facts-statistics-aggressive-driving`, 2021.

[111] Media, NHTSA, "Distracted driving." `https://www.nhtsa.gov/risky-driving/distracted-driving`, 2021.

[112] L. Banjanovic-Mehmedovic, E. Halilovic, I. Bosankic, M. Kantardzic, and S. Kasapovic, "Autonomous vehicle-to-vehicle (v2v) decision making in roundabout using game theory," *International journal of advanced computer science and applications*, vol. 7, no. 8, 2016.

[113] U. Michieli and L. Badia, "Game theoretic analysis of road user safety scenarios involving autonomous vehicles," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1377–1381, IEEE, 2018.

[114] H. Wei, L. Mashayekhy, and J. Papineau, "Intersection management for connected autonomous vehicles: A game theoretic framework," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 583–588, IEEE, 2018.

[115] L. Xue, B. Ma, J. Liu, C. Mu, and D. C. Wunsch, "Extended kalman filter based resilient formation tracking control of multiple unmanned vehicles via game-theoretical reinforcement learning," *IEEE Transactions on Intelligent Vehicles*, 2023.

[116] K.-H. N. Bui and J. J. Jung, "Cooperative game-theoretic approach to traffic flow optimization for multiple intersections," *Computers & Electrical Engineering*, vol. 71, pp. 1012–1024, 2018.

[117] A. Liniger and J. Lygeros, "A noncooperative game approach to autonomous racing," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 884–897, 2019.

[118] S. Ko, "Reinforcement learning based decision making for self driving & shared control between human driver and machine," 2021.

[119] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Cooperation-aware reinforcement learning for merging in dense traffic," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 3441–3447, IEEE, 2019.