Expound: a Diversity-driven Black-box Approach for Categories of Model and Domain-agnostic Adversarial Examples

Kenneth H. Chan[0000−0001−5014−3411] and Betty H.C. Cheng[0000−0001−9825−5359]

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

Abstract. Deep neural networks (DNNs) have been increasingly used in safety-critical systems for decision-making (e.g., medical applications, autonomous vehicles, etc.). Early work with adversarial examples shows that they can hinder a DNN's ability to correctly process inputs. These techniques largely focused on identifying individual causes of failure, emphasizing that minimal perturbations can cause a DNN to fail. Recent research suggests that *diverse* adversarial examples, those that cause different erroneous model behaviors, can better assess the robustness of DNNs. These techniques, however, use white-box approaches to exhaustively search for diverse DNN misbehaviors. This paper proposes a black-box, model, and data-agnostic approach to generate diverse sets of adversarial examples, where each set corresponds to one type of model misbehavior (e.g., misclassification of an input to a specific label), thus termed a *failure category*. Furthermore, each failure category comprises a diverse set of perturbations, all of which produce the same type of model misbehavior. As such, this work provides both breadth and depth-based information for addressing robustness with respect to multiple categories of failures due to adversarial examples. We illustrate our approach by applying it to different combinations of benchmark datasets and models from various application domains (e.g., image and audio data), generating diverse collections of adversarial examples.

Keywords: Adversarial Example · Novelty Search · Machine Learning · Search-based Software Engineering

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 [22]. However, the existence of adversarial examples [48] 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 [5][57]. DNNs

used in safety-critical autonomous systems (e.g., medical imaging, autonomous vehicles [11][47], 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 paper introduces a black-box, diversity-focused search-based framework, applicable to multiple media (e.g., image and audio), 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 [1][42][55]. Traditional approaches to generating adversarial examples typically discover the "nearest" adversarial examples. In contrast, Rozsa *et al.* [42] promoted the use of adversarial examples that 1) have non-minimal perturbations and 2) that result in different model misbehaviors. Aghababaeyan *et al.* [1] 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 triggers 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 paper 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* [20] 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 leads 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 [33] in two complementary ways. In contrast to traditional evolutionary search techniques, novelty search 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 objec-

[1] This work uses the term *diversity* to describe different DNN model behaviors.
[2] One definition for *expound* is to explain systematically or in detail.

tives. 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 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 1a). The objective of evolutionary search-based adversarial attacks, such as GenAttack [4] or EvoAttack [14], is to generate *one* adversarial example per execution (Figure 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 1c).

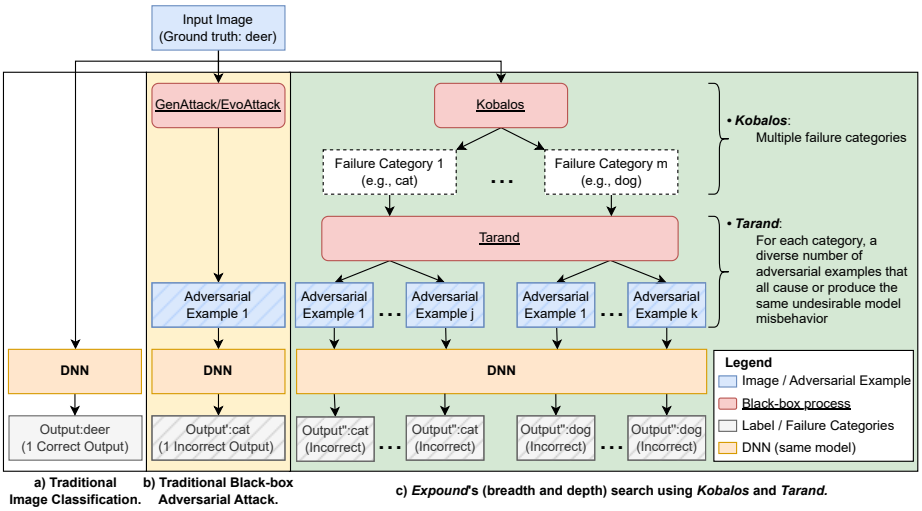


Fig. 1: Overview of image classification, adversarial example, and EXPOUND.

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

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 [26] and MobileNet-V2 [43], on two different benchmark datasets, CIFAR-10 [29] and German Traffic Sign Recognition Benchmark (GTSRB) dataset [44]. We have extended our preliminary work [15] by performing additional validation studies that demonstrate how EXPOUND can also be used to generate diverse adversarial examples against audio machine learning datasets and models (to help demonstrate that EXPOUND is application agnostic). We show that EXPOUND can be used to perturb audio waveform data by generating adversarial examples against an Audio Spectrogram Transformer (AST) model [23] trained on the Google Speech Commands dataset [56], a popular audio classification benchmark dataset. The remainder of the paper is organized as follows. Section 2 overviews background and related information for adversarial examples and novelty search. Section 3 describes the methodology for our proposed framework. Section 4 provides the implementation details and the results of a proof-of-concept validation experiments. Section 5 discusses the results, where we provide insights and consider implications of our results. Section 6 reviews related work. Section 7 summarizes the work presented in this paper and overviews future directions.

2 Background

This section provides background information and enabling technologies used in this work. First, we overview DNNs, the fundamental technology supporting the modern surge of artificial intelligence. Second, we describe adversarial examples and how they impact image classification algorithms. Finally, we also overview the key differences between traditional fitness-based evolutionary searches and novelty search.

2.1 Deep neural networks

Recent success of DNNs has promoted a large surge in the use of artificial intelligence, leading to their widespread adoption in many applications [8][45]. These applications include those that are safety-critical, where their failure may lead to injury or loss of life. DNNs are artificial neural networks comprising many layers of hidden *neurons* and connecting *edges*. They take numeric properties as inputs (e.g., pixel values of an image or the amplitude of a waveform in audio data) and pass the values through the hidden layers, with each layer applying its corresponding weights, biases, or mathematical operations to the values. For a classification problem, the output of a DNN is typically a layer of neurons based on the maximum number of output labels, where the neuron with the

highest value is the classification result of the model. Weights and biases in DNNs are automatically determined through a process called *training*, where the DNN learns optimal weights by iteratively and repeatedly sampling a large dataset and maximizing the number of correct classifications. After each training step (i.e., epoch), the weights and biases are automatically updated through a process of gradient descent and back-propagation to minimize a loss function (i.e., minimizing the number of incorrect classifications). By extracting patterns directly from training samples, DNNs can analyze and correlate complex and high-dimensional features directly without human supervision.

2.2 Adversarial examples

Adversarial examples are expected inputs for a machine learning model but modified with minor perturbations (i.e., noise). They hinder a DNN model's ability to correctly process inputs. For example, an image classification model may correctly classify an object in an image as a *deer*. However, when carefully crafted (human-imperceptible) perturbation is applied to the image, the model may incorrectly classify it as a *frog*. An *adversarial direction* [42] is defined as the class of perturbations for adversarial examples that lead to the same misclassified label. An image dataset with a total of n class labels has $n-1$ adversarial directions for each image, as each image can be incorrectly classified as one of $n-1$ labels.

Due to their computational efficiency, *white-box* attacks are popular approaches to generate adversarial examples [13][24][48]. In white-box attacks, the adversary has full access to the model's parameters, weights, and architecture. Traditional approaches for generating adversarial examples analyze a model's gradient information and introduce perturbations that are likely to induce a failure in the model. In practice, the model's hidden parameters are often proprietary (i.e., only the compiled model is available for public use). In contrast, black-box attacks assume that the adversary has no *a priori* knowledge of the model [46]. Instead, they may query the model with any valid input and obtain the model's output. The direction of perturbation must be inferred from the change in behavior caused by the perturbation during each query. Thus, black-box attacks are more realistic but more challenging to successfully generate adversarial examples when compared to white-box attacks [14][4].

2.3 Search-based testing for DNNs

In traditional software engineering, a developer can systematically test their code and analyze the software execution flow to trace bugs, both of which can involve visual inspection of the code. However, DNNs often lack interpretability, the degree a human can understand the reasoning behind the DNN's output [5][10][36]. Additionally, DNNs must be robust against minor perturbations in safety-critical settings to ensure safety. Thus, a comprehensive set of test cases that is representative of all possible operating contexts is vital to ensure that the model behaves correctly and faults are discovered before deployment. Search-based techniques

are popular approaches to generate test cases for DNNs [2]. Search-based techniques explore a developer-defined search space to discover test cases based on various fitness criteria, such as the number of faults or neuron coverage [40][51]. In search-based testing, the parameters used to define an individual in the search space are known as *genomes*. In contrast, the observable characteristics or properties of the individual (e.g., the output of a DNN model, the color of the organism, the structure of the entity, etc.) for each genome is known as its *phenome*.

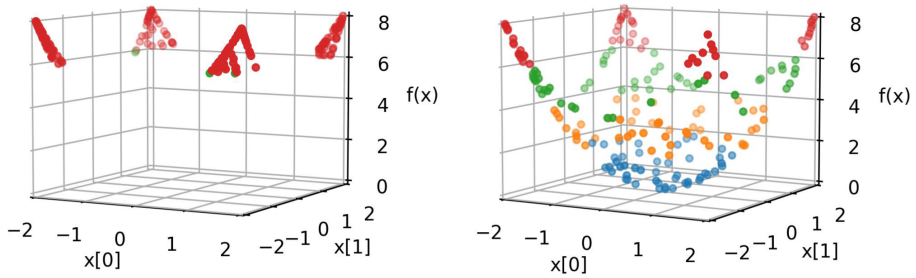
Popular search methods to generate test cases involve Monte Carlo search, grid search, and evolutionary algorithms [9][25]. In Monte Carlo search, genomes are randomly selected as test cases. As Monte Carlo is driven by non-determinism, a solution is not guaranteed and solutions discovered may not be optimal [58]. In contrast, grid search builds an archive of individuals for each combination of genomes and exhaustively searches them for the best phenome. While grid search always finds the global optima, it does not scale computationally as the size and range of genomes increase, or when a high computation cost is associated with the evaluation of individual fitness. In contrast, evolutionary algorithms use techniques analogous to those found in nature [7]. By enabling individuals that are more fit to reproduce, a population of individuals iteratively evolves together towards an optimum using genetic operators such as *selection*, *crossover*, and *mutation* [6].

2.4 Novelty search

In contrast to traditional evolutionary search techniques (e.g., genetic algorithms [27]), whose objective is to seek solutions that optimize specific criteria, *novelty search* is an evolutionary search technique that focuses on the diversity of the evolved solutions. Traditional evolutionary algorithms focus on optimizing a fitness function (e.g., test case metrics, prediction confidence, etc.) and producing similar output(s). These algorithms are efficient and effective for simple problems, but are prone to be trapped in suboptimal regions [34]. Additionally, when there is no clear fitness metric or multiple solutions exist in different search regions/directions, fitness-based evolutionary search is no longer effective for such problem spaces. In order to explore the large search space and discover multiple interesting (unexpected) solutions, novelty search abandons the notion of fitness [33]. Specifically, novelty search promotes individuals in a population to be different from their neighbors. Example diversity criteria for novelty search include the diverse output labels of a DNN or diverse perturbation noise for adversarial examples. Finally, rather than searching for a single solution (i.e., one individual), novelty search produces and maintains an archive of the most diverse individuals by replacing similar individuals with less similar candidates in the archive during each generation.

Figure 2 contrasts the results of a traditional evolutionary algorithm and a novelty search algorithm to optimize a two-dimensional sphere function, $f(x) = \sum_{i=1}^{2} x_i^2$ with $x_i \in [-2, 2]$. The objective of the evolutionary algorithm is to *maximize the output value* of the function, while the objective of the novelty search is to *maximize the diversity* of the output. The colors of individuals in the plots

denote the four quartile regions of the output (e.g., blue denotes a low fitness score while red denotes a high fitness score). Figure 2(a) shows an example of individuals in the population of a converged traditional evolutionary search. As the algorithm seeks to maximize the output of the sphere function, the majority of the individuals in the population converge towards the optima (denoted in red). Figure 2(b) shows the results of novelty search on the same search space. Novelty search instead discovers individuals that behave differently from each other, thus creating an archive that is evenly distributed throughout the different quartiles of the search space (represented by different colors), including individuals that are similar to those found by the traditional search. As such, novelty search is better able to capture different and more diverse observable behaviors for a given system to discover the full range of possible behaviors.



(a) Evolutionary search to maximize the sphere function, f(x)

(b) Novelty search for output diversity of the sphere function, f(x)

Fig. 2: Comparison of traditional fitness-based search and novelty search on the two-dimensional sphere function, $f(x)$. The inputs $x[0]$ and $x[1]$ for $f(x)$ range from $[-2, 2]$. Each color denotes one of the four quartile regions of the output.

3 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 3 shows a Data Flow Diagram (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 data sample and the black-box classification model 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 key steps of EXPOUND in turn.

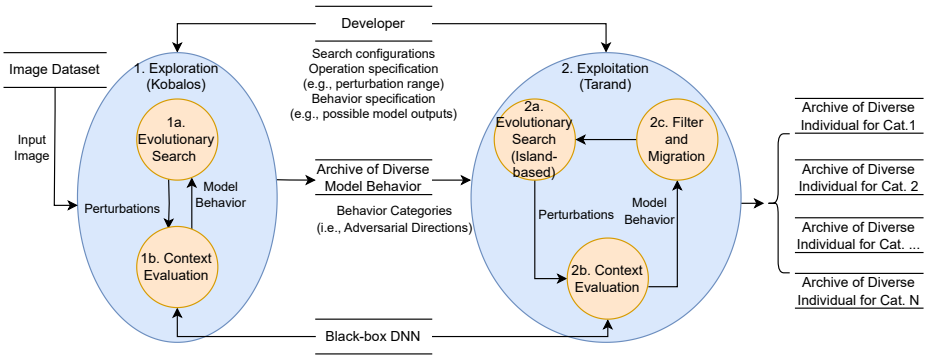


Fig. 3: EXPOUND's process to identify failure categories, each comprising diverse adversarial examples.

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., the largest number of mutually exclusive class labels). The left portion of Figure 4 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.

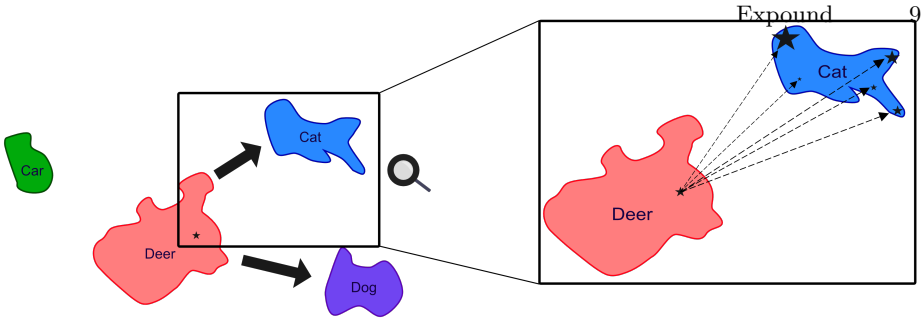


Fig. 4: High level interpretation of EXPOUND. Stars denote images. KOBALOS (left) discovers the failure categories while TARAND (right) exploits the information to discover diverse adversarial examples for each category identified.

Algorithm 1 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 a 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 3). Individuals are then ranked by computing a novelty score based on the *Euclidean distance* between phenomes with other individuals in the archive (line 13). Expression 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).

$$\text{novelty}(\text{p}, \text{P}, k) = \text{mean}(\text{min-k}(\, \|\text{p} - \text{p}_i\|^2 \, \forall \text{p}_i \in \text{P} : \text{p}_i \neq \text{p}, k)) \qquad (1)$$

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.

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 classification score diversity (i.e., different combi-

nations 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 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 contrasts the respective confidence scores of the DNN for the image [42]. 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).

Algorithm 1: Pseudo-code for Kobalos and Tarand [31][33].

```
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
```

To discover a range of perturbations for each category of misclassification, TARAND uses a parallelized novelty search that maintains multiple demes of genomes [37], where a deme is a population of individuals that only evolves with other individuals in the same deme. The number of demes maintained in TARAND is based on the number of unique model outputs from KOBALOS, where each deme is associated with a unique model behavior. In addition, TARAND maintains a separate archive for each deme, each categorizing the most unique individuals exhibiting the same observable behavior (i.e., output). For each deme, TARAND generally follows the same evolutionary process used by KOBALOS, but differs in the diversity metric. Specifically, KOBALOS optimizes a diversity metric based on the classification of the model's output (i.e., the incorrect labels), while TARAND optimizes the diversity with respect to confidence scores of the model output (i.e., the confidence scores of the incorrect labels). Essentially, rather than focusing on input diversity (e.g., structure of the genomes) that lack interpretability in a black-box setting, TARAND identifies the most diverse set of confidence scores for each incorrect label to account for the range of effects that

the adversarial noise filters may induce. First, a population of perturbation filters is 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 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 a deme that exhibit behavior different from other individuals on the same deme are filtered and migrated to the deme with the corresponding behavior (blue line 10 in Algorithm 1). For example, an adversarial example with the label *dog* on a deme of *cats* will be filtered and migrated to the deme 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 (2). Specifically, categories that contain several confidently misclassified images will be assigned a higher score, close to 1.0.

$$\text{MisclassificationScore}(\text{archive}) = \text{mean}(\text{archive}[0.5 : 1].\text{getConfidenceScores}()) \quad (2)$$

4 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. Additionally, we demonstrate that our approach is application agnostic by applying EXPOUND to an audio machine learning dataset and model. 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:

RQ1: Can EXPOUND identify distinct failure categories using KOBALOS?
RQ2: Can EXPOUND identify more faults using TARAND for each failure category?

4.1 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 [39]. 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 [29], 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 [44]. 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 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 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

4.2 Coarse-grained search for distinct failure categories (RQ1)

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:

$RQ1 - H_0$: Kobalos does not identify more failure categories.
$RQ1 - H_1$: 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 2 shows the average number of unique labels identified by each approach when used to discover failure categories. The standard deviation is denoted in parentheses, 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 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 5 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.

4.3 Fine-grained search for diverse adversarial examples (RQ2)

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:

$RQ2 - H_0$: TARAND does not identify more faults for each failure category.
$RQ2 - H_1$: TARAND identifies more faults for each failure category.

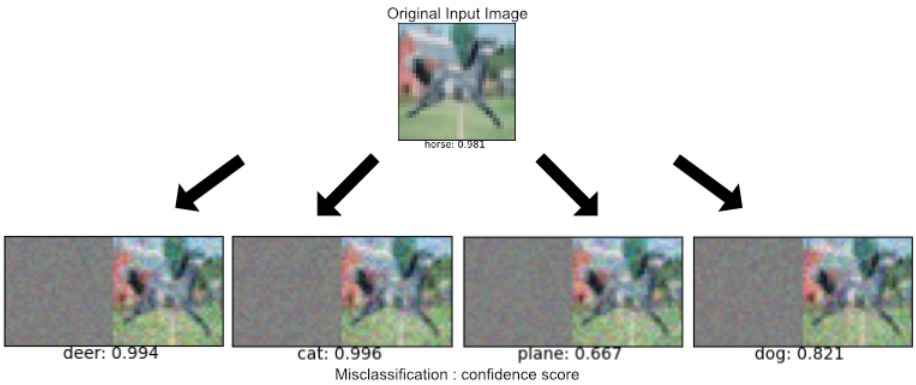


Fig. 5: Example output of KOBALOS on an input image of a *horse*. KOBALOS discovered four different incorrect model behaviors when exposed to perturbations.

Table 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 parentheses. 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.

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 misclassification score based on Expression (2) defined in Section 3. Figure 6 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 parentheses 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.

Table 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

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]

Fig. 6: 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 Applications of Expound for audio data

This section demonstrates that EXPOUND is application agnostic by applying EXPOUND to audio data, which stores information differently than image data. For image data, the inputs of the DNN are images in the tensor shape $[RGB\_channel, i, j]$, where i and j denote the row and column of the pixel, respectively. In contrast, audio data is represented as a waveform when stored digitally, capturing the signal as a function of time in the tensor shape $[w]$, where w is the measured magnitude of the signal at the corresponding timestep. Figure 7 shows an example of an audio sample from the Google Speech Commands dataset [56], where the x-axis represents time and the y-axis represents the magnitude of the signal. We trained an AST model [23] with a base accuracy of 97.96% on the Google Speech Commands dataset [56] to conduct our experiments. While EXPOUND can be applied to perturb the audio sample in both the waveform and the spectrogram format, we demonstrate that EXPOUND is application agnostic by perturbing the waveform directly in order to remain faithful to the nature of the audio file format (in contrast to perturbing the spectrogram, which has a similar tensor shape as an image). As such, we have modified the original AST model's approach to decouple the data loading as a waveform and spectrogram conversion, thereby allowing perturbations generated by EXPOUND to be introduced to the input waveform directly. Figure 8 shows how we have adapted the information flow from the Google Speech Commands dataset to the AST Model for our experiment. Then the perturbed waveform is converted to its spectrogram format. Finally, the AST model is used to classify the spectrogram. We selected to use an AST model for our validation study since it is convolution-free. Instead, it uses a transformer layer, the underlying technology behind the current prevalent and popular large language models [53]. As such, our studies will enable us to study the impact of such adversarial examples on increasingly used transformer-based models (e.g., large language models, vision transformers, reinforcement learning, etc.).

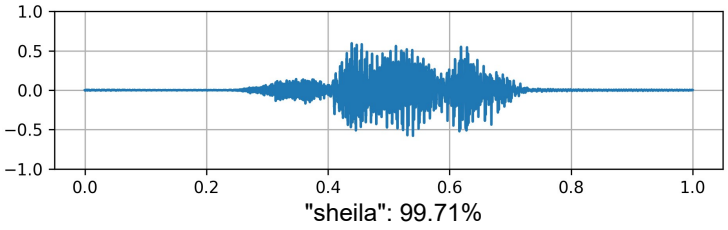


Fig. 7: An example waveform from the Google Speech Commands dataset, where the AST model correctly classified the input as *"Sheila"*.

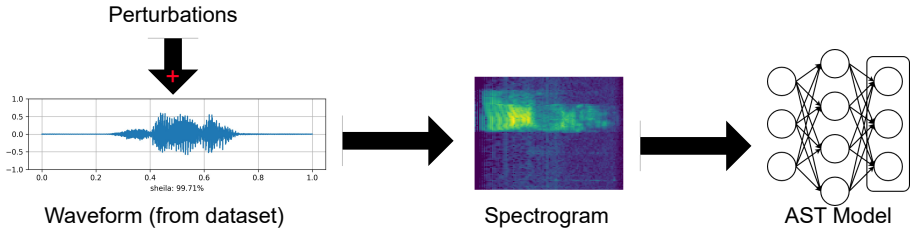


Fig. 8: The data pipeline of our audio validation study. While perturbations are introduced directly to the waveform, EXPOUND can also generate perturbations against the spectrogram.

This experiment largely follows the same algorithm and experimental setup as our previous validation studies for image data, but we update the perturbation space to the waveform for the audio data. The evolutionary parameters based on the number of possible classes are also updated based on the Google Speech Commands dataset (35 total labels). Table 4 shows the corresponding configuration settings for the audio validation experiments of this work.

Table 4: Configuration settings for EXPOUND in the audio experiments.

	Google Speech Commands	
Parameter	Value(Kobalos)	Value(Tarand)
Archive Size	35	10
Population Size	10	35
Num Generations	100	500
Mutation Rate	0.15	0.15
Mutation Shift	0.2	0.2

In the coarse-grained search for distinct failure categories (RQ1), we also select 30 randomly-sampled input audio samples and attempt to generate an archive of 35 diverse individuals for each sample. We repeat the experiment ten times to account for statistical variance. Table 5 shows the results for KOBALOS's search for failure categories, which are comparable to the results of our image experiments. Compared to Monte Carlo, KOBALOS consistently discovers more unique labels, with an average increase of 286%. We found strong evidence ($p \leq 0.01$) to also reject H_0 and support H_1 for RQ1 of the audio experiment. Figure 9 shows the waveform of an example input from the Google Speech Commands dataset and the corresponding unique (incorrect) labels identified by KOBALOS. The AST model correctly classified the original input audio sample as the label *"four"* with a confidence score of 75.34% (highlighted in yellow). We applied KOBALOS and successfully discovered different perturbations that led to 4 unique incorrect labels with varying degrees of confidence scores, including one that resulted in a higher confidence score than the original correct input (the label *"go"* with a confidence score of 76.90% denoted in red text, Figure 9). These adver-

sarial examples introduced static noise that led to distinct model misbehaviors, while maintaining the same overall waveform structure (i.e., the noise does not prevent human perception of the word).

Table 5: Results for Monte Carlo sampling and EXPOUND's search for failure categories against the AST model trained on the Google Speech Commands dataset. We find consistent results compared to image data, where KOBALOS is able to find more unique labels than Monte Carlo.

DNN and Dataset	**Number of Average Unique Labels**		p-value significance
	Monte Carlo	KOBALOS	
AST Model - Google Speech Commands	1.32 ($\sigma = 0.07$)	**5.10** ($\sigma = 0.17$)	< 0.01

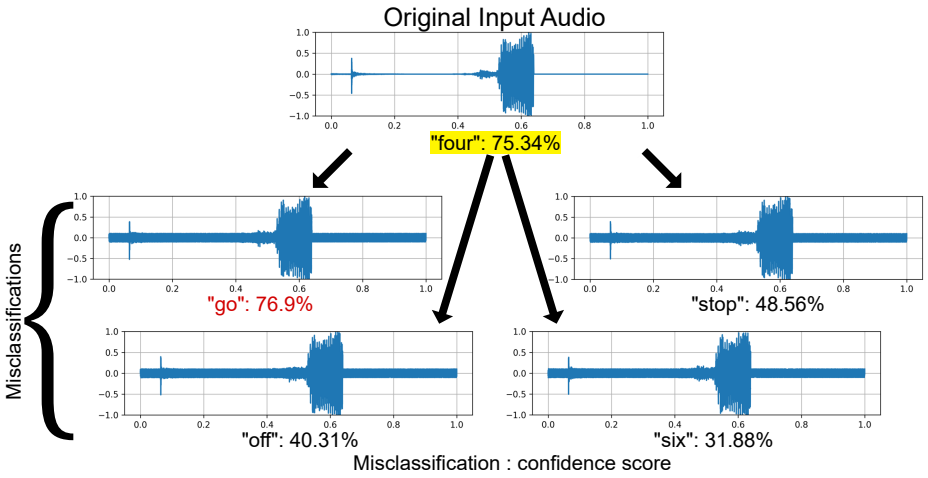


Fig. 9: Example output of KOBALOS on audio input data. The original input audio waveform is perturbed, leading to 4 unique and incorrect labels with different confidence scores; red indicates the highest confidence score.

After the diverse archive of misclassification labels is discovered using KOBALOS, we use TARAND to apply a fine-grained search for archives of adversarial examples for each unique label (RQ2). Table 6 shows the results of TARAND for the audio classification experiment for the same 30 input samples. The results show that TARAND can successfully fill most archives (82.09%) for the failure categories identified by KOBALOS (i.e., discovering ten increasingly different adversarial examples for each incorrect label). In contrast, Monte Carlo failed to consistently fill the same archives for the failure categories identified by KOBALOS. Therefore, we found strong evidence ($p \leq 0.01$) to reject H_0 and support H_1 for RQ2.

Table 6: 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	
AST Model - Google Speech Commands	5.97% ($\sigma = 0.62\%$)	**82.09%** ($\sigma = 3.71\%$)	< 0.01

4.5 Threats to validity

The results in this paper 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 inputs were sampled from different datasets. Additionally, each experiment was repeated ten times with different randomly-generated seeds to ensure comparable results.

5 Discussion

This section discusses the findings of our experiments and offers insights for the results.

5.1 Experimental findings and insights

This paper has shown that novelty search can be used to generate diverse adversarial examples against DNN models trained on classification tasks. While other existing approaches focus on generating adversarial examples to attack the model by causing a (single) misclassification, our work focuses on understanding the limitations and *range* of failures for a given DNN. Specifically, existing white-box approaches require an exhaustive and brute-force approach to identify different failures, while other black-box approaches typically lead to the same types of failures when repeating the same search due to the homogeneous nature of the search landscape. In contrast, our approach generates a diverse archive of adversarial examples during each execution, identifying both distinct failure categories and generating collections of diverse adversarial examples that lead to each of those failure categories. Finally, while a key objective of adversarial examples in a malicious setting is to minimize the perturbations, we do not impose a minimal perturbation constraint in EXPOUND (but limit maximum perturbation) in order to allow for the discovery and identification of the full range of failures in the DNN.

In order to demonstrate the capabilities of our work and avoid statistical errors, our validation studies used a wide range of randomly selected inputs with various ground truth labels and provided the average results. However, we found that not all input samples are affected by perturbations to the same degree. For

example, in the experiments for the audio dataset, we found that some samples may lead to few or no instances of misclassifications, while some samples may include a large range of misclassifications (up to 13 unique labels out of 35 total labels). As such, we believe that some inputs are more susceptible to adversarial perturbations, while others are more robust to adversarial perturbations. Finally, the number of labels and percentage of novelty archive filled do not appear to have any strong correlation with the dataset, model architecture, or the application domain. This is likely because the discovered adversarial perturbations disrupt the features used by the DNN to correlate between the patterns in the dataset, instead of attacking dataset-specific, model-specific, or application-specific weaknesses.

5.2 Gradient-free approach

The experiments in this work have demonstrated that EXPOUND can be applied to different models and datasets, as well as different application domains. Compared to other white-box approaches that depend on gradient information to generate perturbations that hinder the model's capabilities, EXPOUND (and other similar black-box approaches) introduces perturbations to the inputs of the DNN instead. EXPOUND harnesses evolutionary search and uses selection pressure to discover perturbations that lead to increasingly different model misbehaviors (i.e., EXPOUND is gradient-free). As such, when applying EXPOUND to discover diverse adversarial examples for other classification tasks, developers only need to update the operating context, outlining the inputs that can be perturbed and providing the range of perturbations. Finally, EXPOUND can also be applied to other machine learning tasks, such as object detection. However, developers must define the criteria for output diversity for their desired tasks.

5.3 Coarse-grained and fine-grained search

Our EXPOUND framework has introduced a two-step approach, where novelty search has been demonstrated to discover both coarse-grained adversarial information and corresponding fine-grained adversarial information. KOBALOS first *explores* and discovers the possible failure category for a DNN in a coarse-grained search. The information discovered by TARAND is then used to guide the fine-grained search, where the possible failure categories are *exploited* to discover increasingly different inputs that lead to the same misclassification, with different misclassification confidence scores from the model. A key advantage of TARAND's fine-grained search is the generation of uniformly-distributed archives that lead to various misbehaviors, compared to Monte Carlo that may not discover the full range due to the sparsity of some labels. Finally, compared to other novelty search-based approaches for diverse archives of misbehaviors, where a human developer defines the criteria for behavior categories, our approach automatically determines these behavior categories for TARAND.

To the best of our knowledge, this work is the first to use the *exploration and exploitation paradigm* in novelty search to assess the robustness of DNNs. We

believe similar approaches may be applied in other settings to discover different kinds of uncertainty for AI-enabled systems, such as identifying a diverse range of undesirable behaviors that an autonomous vehicle may exhibit in the presence of human-driven vehicles. This information may then be exploited to discover increasingly different maneuvers from the human-driven vehicle that can lead to critical test cases. Diverse test cases discovered by EXPOUND can facilitate developers to identify common challenges for their system, revise the requirements, and update their system to mitigate the uncertainty accordingly. Compared to traditional test case generation approaches (that generate a single targeted test case), our novelty search-based approach attempts to identify the full range of failures for DNNs, uncovering both depth and breadth-based information on the types of failures.

6 Related work

Early work with adversarial examples was generated using white-box approaches [46]. These approaches leverage gradient information of the model to be attacked and introduce noise to hinder the model's capability. Szegedy *et al.* [48] first introduced adversarial examples generated using the L-BFGS algorithm. Carlini and Wagner [13] proposed a similar approach with a modified objective function and constraint, known as the C&W attack. Goodfellow *et al.* [24] introduced the Fast Gradient Sign Method (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.* [38] demonstrated that white-box adversarial attacks can transfer to other models. In contrast, Alzantot *et al.* [4], Vidnerová *et al.* [54], Chen *et al.* [19] proposed various genetic algorithm approaches to generate adversarial examples against image classification algorithms. Chan and Cheng proposed EvoAttack [14][16] that showed adversarial attacks also apply to object detection algorithms, an important component for perception sensing in autonomous vehicles. Recently, McIntyre-Garcia *et al.* [35] extended their work by introducing a multi-objective approach to further reduce adversarial noise. 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.* [42] 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. Similarly, Tashiro *et al.* [50] proposed a white-box approach to generate diverse outputs to study the effect of transferability for different victim models. However, their work is not gradient-free, as they train a surrogate model to attack before transferring the perturba-

tions. Aghababaeyan *et al.* [1] 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 [31][32] 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 [31] 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 [32], a novelty search framework to discover operating contexts that can lead to different categories of performance degradation. Dong *et al.* [21] proposed the use of pixel value diversity (i.e., spread of pixel values in an image) to detect adversarial examples, but does not consider adversarial example diversity. Zhou *et al.* [60] proposed a multi-objective evolutionary search-based approach to generate adversarial examples for code snippets, but they do not explicitly consider the output behavior diversity.

Finally, recent work has been done to assess the adversarial robustness of DNNs for audio machine learning. Qin *et al.* [41] proposed a white-box attack where they minimize audio perturbations by abandoning the traditional norm distance measure used in image attacks, instead using psychoacoustic principles to inject imperceptible noise. Chang *et al.* [17] introduced a white-box attack method that trains a Recurrent Neural Network to generate adversarial examples. Other researchers [12][52][59] have also explored other white-box approaches to hinder speech recognition and other audio machine learning models. Alzantot *et al.* [3], Taori *et al.* [49], Khare *et al.* [28], and others have proposed the use of evolutionary algorithms to attack natural language models. Chen *et al.* [18] explored diverse (input) waveform transformation and their effect on model robustness, but did not consider the diversity in the model's behavior.

While other researchers have explored image and audio adversarial examples, 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. EXPOUND provides both breadth and depth-based information about the types of perturbations that cause model misbehavior. This work is also the first black-box technique that addresses diverse adversarial examples (for distinct model misbehavior) that is model, data, and application agnostic.

7 Conclusion

This paper 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 [30]. Finally, other audio transformation functions may be used to parameterize adversarial noise and minimize the perturbations introduced to audio machine learning.

Acknowledgements

We greatly appreciate Michael Austin Langford's contributions on our preliminary work. We also greatly appreciate the insightful and detailed feedback from the reviewers. This work was supported in part by funding provided by Michigan State University, the BEACON Center, the Air Force Research Laboratory, and our industrial collaborators.

References

1. Aghababaeyan, Z., Abdellatif, M., Dadkhah, M., Briand, L.: Deepgd: A multi-objective black-box test selection approach for deep neural networks. ACM Transactions on Software Engineering and Methodology **33**(6), 1–29 (2024)
2. Ali, S., Briand, L.C., Hemmati, H., Panesar-Walawege, R.K.: A systematic review of the application and empirical investigation of search-based test case generation. IEEE Transactions on Software Engineering **36**(6), 742–762 (2009)
3. Alzantot, M., Balaji, B., Srivastava, M.B.: Did you hear that? adversarial examples against automatic speech recognition. CoRR **abs/1801.00554** (2018), `http://arxiv.org/abs/1801.00554`
4. Alzantot, M., Sharma, Y., Chakraborty, S., Zhang, H., Hsieh, C.J., Srivastava, M.B.: GenAttack: Practical black-box attacks with gradient-free optimization. In: Proc. of the Genetic and Evol. Comp. Conf. pp. 1111–1119 (2019)
5. Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al.: Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI. Information Fusion **58**, 82–115 (Jun 2020). https://doi.org/10.1016/j.inffus.2019.12.012
6. Back, T.: Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press (1996)
7. Bäck, T., Schwefel, H.P.: An overview of evolutionary algorithms for parameter optimization. Evolutionary computation **1**(1), 1–23 (1993)
8. Balas, V.E., Kumar, R., Srivastava, R., et al.: Recent trends and advances in artificial intelligence and internet of things. Springer (2020)

9. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of machine learning research **13**(2) (2012)
10. Biran, O., Cotton, C.: Explanation and justification in machine learning: A survey. In: IJCAI-17 workshop on explainable AI (XAI). vol. 8, pp. 8–13 (2017)
11. Cai, Z., Fan, Q., Feris, R.S., Vasconcelos, N.: 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)
12. Carlini, N., Mishra, P., Vaidya, T., Zhang, Y., Sherr, M., Shields, C., Wagner, D., Zhou, W.: Hidden voice commands. In: 25th USENIX security symposium (USENIX security 16). pp. 513–530 (2016)
13. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 ieee symposium on security and privacy (sp). pp. 39–57. IEEE (2017)
14. Chan, K., Cheng, B.H.C.: Evoattack: An evolutionary search-based adversarial attack for object detection models. In: Search-Based Software Engineering: 14th Int. Sym., SSBSE 2022, Singapore, November, 2022. pp. 83–97. Springer (2022)
15. Chan, K.H., Cheng, B.H.C.: Expound: a black-box approach for generating diversity-driven adversarial examples. In: International symposium on search based software engineering. pp. 19–34. Springer (2023)
16. Chan, K.H., Cheng, B.H.C.: Evoattack: suppressive adversarial attacks against object detection models using evolutionary search. Automated Software Engineering **32**(1), 3 (2025)
17. Chang, K.H., Huang, P.H., Yu, H., Jin, Y., Wang, T.C.: Audio adversarial examples generation with recurrent neural networks. In: 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC). pp. 488–493. IEEE (2020)
18. Chen, G., Zhao, Z., Song, F., Chen, S., Fan, L., Wang, F., Wang, J.: Towards understanding and mitigating audio adversarial examples for speaker recognition. IEEE Transactions on Dependable and Secure Computing **20**(5), 3970–3987 (2022)
19. Chen, J., Su, M., Shen, S., Xiong, H., Zheng, H.: POBA-GA: Perturbation optimized black-box adversarial attacks via genetic algorithm. Computers & Security **85**, 89–106 (2019)
20. Črepinšek, M., Liu, S.H., Mernik, M.: Exploration and exploitation in evolutionary algorithms: A survey. ACM computing surveys (CSUR) **45**(3), 1–33 (2013)
21. Dong, J., Zhou, P.: Detecting adversarial examples utilizing pixel value diversity. ACM Transactions on Design Automation of Electronic Systems **29**(3), 1–12 (2024)
22. Gheibi, O., Weyns, D., Quin, F.: Applying machine learning in self-adaptive systems: A systematic literature review. ACM TAAS **15**(3), 1–37 (2021)
23. Gong, Y., Chung, Y.A., Glass, J.: AST: Audio Spectrogram Transformer. In: Proc. Interspeech 2021. pp. 571–575 (2021). https://doi.org/10.21437/Interspeech.2021-698
24. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (2015)
25. Harman, M., Mansouri, S.A., Zhang, Y.: Search-based software engineering: Trends, techniques and applications. ACM Computing Surveys (CSUR) **45**(1), 1–61 (2012)
26. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
27. Holland, J.: Genetic algorithms. Scientific american **267**(1), 44–50 (1992)

28. Khare, S., Aralikatte, R., Mani, S.: Adversarial black-box attacks on automatic speech recognition systems using multi-objective evolutionary optimization. In: Interspeech 2019. pp. 3208–3212 (2019). https://doi.org/10.21437/Interspeech.2019-2420
29. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
30. Kurakin, A., Goodfellow, I., Bengio, S., Dong, Y., Liao, F., Liang, M., Pang, T., Zhu, J., Hu, X., Xie, C., et al.: Adversarial attacks and defences competition. In: The NIPS'17 Competition: Building Intelligent Systems. pp. 195–231. Springer (2018)
31. Langford, M.A., Cheng, B.H.C.: Enki: a diversity-driven approach to test and train robust learning-enabled systems. ACM TAAS **15**(2), 1–32 (2021)
32. Langford, M.A., Cheng, B.H.C.: "Know What You Know": Predicting behavior for learning-enabled systems when facing uncertainty. In: 2021 Int. Symposium on Software Eng. for Adaptive and Self-Managing Systems. pp. 78–89. IEEE (2021)
33. Lehman, J., Stanley, K.O.: Abandoning objectives: Evolution through the search for novelty alone. Evolutionary computation **19**(2), 189–223 (2011)
34. Lehman, J., Stanley, K.O.: Novelty search and the problem with objectives. Genetic programming theory and practice IX pp. 37–56 (2011)
35. McIntyre-Garcia, C., Heymans, A., Borali, B., Lee, W.S., Nejati, S.: Generating minimalist adversarial perturbations to test object-detection models: An adaptive multi-metric evolutionary search approach. In: 2024 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). pp. 9–12. IEEE (2024)
36. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artificial intelligence **267**, 1–38 (2019)
37. Nowostawski, M., Poli, R.: Parallel genetic algorithm taxonomy. In: 1999 Third international conference on knowledge-based intelligent information engineering systems. Proceedings (Cat. No. 99TH8410). pp. 88–92. Ieee (1999)
38. Papernot, N., McDaniel, P., Goodfellow, I.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv (2016)
39. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., et al. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019)
40. Pei, K., Cao, Y., Yang, J., Jana, S.: Deepxplore: Automated whitebox testing of deep learning systems. In: proceedings of the 26th Symposium on Operating Systems Principles. pp. 1–18 (2017)
41. Qin, Y., Carlini, N., Cottrell, G., Goodfellow, I., Raffel, C.: Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In: International conference on machine learning. pp. 5231–5240. PMLR (2019)
42. Rozsa, A., Rudd, E.M., Boult, T.E.: Adversarial diversity and hard positive generation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)
43. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conf. on CV and pattern rec. pp. 4510–4520 (2018)
44. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The german traffic sign recognition benchmark: a multi-class classification competition. In: The 2011 international joint conference on neural networks. pp. 1453–1460. IEEE (2011)

45. Suguna, S.K., Dhivya, M., Paiva, S.: Artificial intelligence (ai): Recent trends and applications (2021)
46. Sun, L., Tan, M., Zhou, Z.: A survey of practical adversarial example attacks **1**, 1 (2018)
47. Szegedy, C., Toshev, A., Erhan, D.: Deep neural networks for object detection. Advances in neural information processing systems **26** (2013)
48. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: International Conference on Learning Representations (2014)
49. Taori, R., Kamsetty, A., Chu, B., Vemuri, N.: Targeted adversarial examples for black box audio systems. In: 2019 IEEE security and privacy workshops (SPW). pp. 15–20. IEEE (2019)
50. Tashiro, Y., Song, Y., Ermon, S.: Diversity can be transferred: Output diversification for white-and black-box attacks. Advances in neural information processing systems **33**, 4536–4548 (2020)
51. Tian, Y., Pei, K., Jana, S., Ray, B.: Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In: Proceedings of the 40th international conference on software engineering. pp. 303–314 (2018)
52. Vaidya, T., Zhang, Y., Sherr, M., Shields, C.: Cocaine noodles: exploiting the gap between human and machine speech recognition. In: 9th USENIX Workshop on Offensive Technologies (WOOT 15) (2015)
53. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
54. Vidnerová, P., Neruda, R.: Vulnerability of classifiers to evolutionary generated adversarial examples. Neural Networks **127**, 168–181 (2020)
55. Wallace, E., Rodriguez, P., Feng, S., Yamada, I., Boyd-Graber, J.: Trick me if you can: Human-in-the-loop generation of adversarial examples for question answering. TACL **7**, 387–401 (2019)
56. Warden, P.: Speech commands: A dataset for limited-vocabulary speech recognition. CoRR **abs/1804.03209** (2018), `http://arxiv.org/abs/1804.03209`
57. Yu, F., Qin, Z., Liu, C., Zhao, L., Wang, Y., Chen, X.: Interpreting and Evaluating Neural Network Robustness (2019). https://doi.org/10.24963/ijcai.2019/583, (IJCAI 2019)
58. Zabinsky, Z.B., et al.: Random search algorithms. Department of Industrial and Systems Engineering, University of Washington, USA (2009)
59. Zhang, G., Yan, C., Ji, X., Zhang, T., Zhang, T., Xu, W.: Dolphinattack: Inaudible voice commands. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. pp. 103–117 (2017)
60. Zhou, S., Huang, M., Sun, Y., Li, K.: Evolutionary multi-objective optimization for contextual adversarial example generation. Proceedings of the ACM on Software Engineering **1**(FSE), 2285–2308 (2024)