Testing DNN-based Autonomous Driving Systems under Critical Environmental Conditions

Zhong Li [1 2] **Minxue Pan** [1 3] **Tian Zhang** [1 2] **Xuandong Li** [1 2]

Abstract

Due to the increasing usage of Deep Neural Network (DNN) based autonomous driving systems (ADS) where erroneous or unexpected behaviours can lead to catastrophic accidents, testing such systems is of growing importance. Existing approaches often just focus on finding erroneous behaviours and have not thoroughly studied the impact of environmental conditions. In this paper, we propose to test DNN-based ADS under different environmental conditions to identify the critical ones, that is, the environmental conditions under which the ADS are more prone to errors. To tackle the problem of the space of environmental conditions being extremely large, we present a novel approach named TACTIC that employs the search-based method to identify critical environmental conditions generated by an image-to-image translation model. Large-scale experiments show that TACTIC can effectively identify critical environmental conditions and produce realistic testing images, and meanwhile, reveal more erroneous behaviours compared to existing approaches.

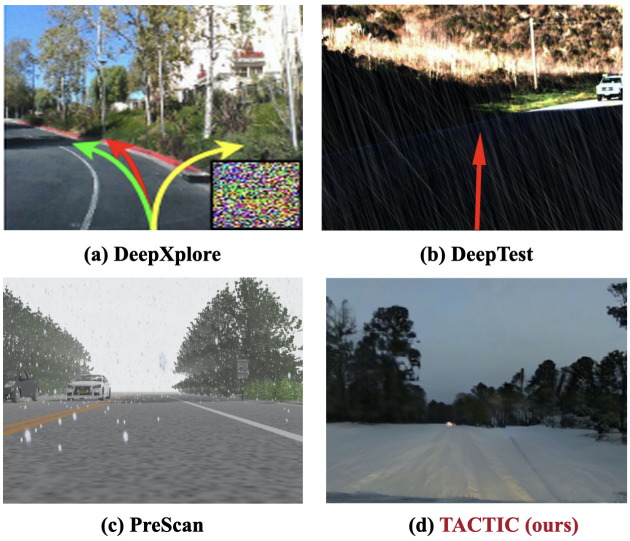


Figure 1. Testing driving scenes generated by DeepXplore (Pei et al., 2017), DeepTest (Tian et al., 2018), PreScan (Marketakis et al., 2009), and our approach TACTIC. Note that the coloured arrows in Figure 1 (a) and (b) are attached to present the predicted steering angles. TACTIC produces more realistic driving scenes compared to the other three approaches.

1. Introduction

Autonomous vehicles have achieved tremendous success over the past decade due to the significant advances in Deep Neural Networks (DNNs). We have witnessed a number of successful DNN-based autonomous driving systems (ADSs) (Grzywaczewski, 2017; Bojarski et al., 2016), where DNNs directly generate the control decisions (e.g., steering and braking) for the systems after processing inputs received from sensors (e.g., camera, Radar, Lidar). Unfortunately, such systems often demonstrate incorrect/unexpected behaviours that can lead to catastrophic accidents when deployed in real-world environments, for instance, the Tesla/Uber autonomous vehicle accidents (Boudette, 2017; Gibbs, 2018). There is an urgent need for a better approach to comprehensively and effectively testing these systems.

Recently, various approaches have been proposed for testing DNN-based ADSs which mostly rely on affine image transformation (Pei et al., 2017; Tian et al., 2018) or high-fidelity simulation (Marketakis et al., 2009; Abdessalem et al., 2016; 2018; Fremont et al., 2019) to generate independent driving scenes that can cause erroneous behaviours. However, there is little attention on studying the impact of environmental conditions (e.g., time-of-day, illumination, and weather, etc.) for DNN-based ADSs. Since DNN-based ADSs must be able to conduct proper operations under all possible environmental conditions in the real world, it is critical to understand which environmental conditions will

[1]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China [2]Department of Computer Science and Technology, Nanjing University, Nanjing, China [3]Software Institute, Nanjing University, Nanjing, China. Correspondence to: Minxue Pan <mxp@nju.edu.cn>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

cause more erroneous behaviours.

In this paper, we first propose the problem of testing DNN-based ADS under different environmental conditions with the goal of identifying the critical environmental conditions under which the ADS are more prone to errors. This is a difficult problem, since it is infeasible to test ADS under the many environmental conditions in real-world. Even if only considering one environmental type, there can be a huge number of environmental conditions. For example, for the environmental type of rainy weather, there can be many different rainy conditions due to the different degrees of light and amounts of rain, etc. An alternative is to adopt synthesised test driving scenes. However, the driving scenes synthesised by most existing approaches can be unrealistic or even never happen in real-world environments. For instances, Figure 1(a)-(c) show three test scenes generated by the state-of-the-art testing techniques, i.e., DeepXplore (Pei et al., 2017), DeepTest (Tian et al., 2018), and PreScan (Marketakis et al., 2009). We can see that all these scenes lack the richness and authenticity of real-world images, and can rarely happen under real-world environments. Besides the problem of synthesised scenes lacking the authenticity of real-world scenes, it is difficult to fully specify an environmental condition by configuring the parameters for affine image transformation or high-fidelity simulation, for an environmental condition is very complex in nature. For example, one cannot exactly specify the accumulation of snow on the street or the amounts of rain for a driving scene. Even worse, the critical environmental conditions may be corner cases and only occupy a small fraction of the whole environmental condition space, making them difficult to be identified.

We address these challenges and propose TACTIC (**T**esting **A**DS under **C**ri**TI**cal **C**onditions), a novel testing framework that can identify critical environmental conditions for different environmental types and generate the corresponding driving scenes for effectively testing DNN-based ADSs. To specify the environmental conditions of an environmental type, instead of using configurations, TACTIC employs the Multimodal Unsupervised Image-to-Image Translation (MUNIT) (Huang et al., 2018) to deep-learn the features of environmental conditions from a set of sample scenes belonging to the same environmental type and encode them as a high-dimensional vector space called *style space*. A *style*, which is a vector in the style space corresponding to an environmental condition, contains extremely rich and complex environmental features. To identify the styles corresponding to the critical environmental conditions in the complex style space, TACTIC employs the search-based method with a carefully designed search objective accounting for both the number and diversity of erroneous behaviours. When styles are obtained, the style-learning process of MUNIT is reversed: styles are applied on existing test driving scenes to synthesise new ones. Because of the richness of styles, the synthesised scenes are realistic enough (as shown by Figure 1(d)) to test whether the DNN-based ADS can perform correctly under the environmental conditions corresponding to the obtained styles.

To evaluate TACTIC, we conducted a large-scale study on three well-known open-source DNN-based ADSs. The results demonstrate that TACTIC is effective in identifying critical environmental conditions for various environmental types. The driving scenes synthesised by TACTIC using the critical environmental conditions are not only more realistic, but also cause the DNN-based ADSs to produce more serious erroneous behaviours, compared to other approaches.

Overall, the main contributions of this paper are:

- To our knowledge, this is the first work that proposes to test DNN-based ADSs with the goal of identifying critical environmental conditions.
- We propose a novel approach named TACTIC, which employs the search-based method to effectively identify critical environmental conditions from the environmental condition space generated by MUNIT model.
- We perform a large-scale evaluation with TACTIC, and the results show that TACTIC is effective in revealing more serious erroneous behaviours for DNN-based ADS with the help of the obtained critical environmental conditions.

Our implementation of TACTIC and all the experimental data are publicly available[1] to facilitate future studies.

2. Related Work

Testing of DNN-based ADS Recently, there are various works emerging on testing DNN-based ADS (Pei et al., 2017; Tian et al., 2018; Zhang et al., 2018; Li et al., 2019; Huang et al., 2019; Odena et al., 2019; Zhou et al., 2020). DeepXplore (Pei et al., 2017) utilises a gradient-based differential testing techniques to detect behaviour inconsistencies among multiple DNN-based ADSs with neuron coverage guidance. As a following work of DeepXplore, DeepTest (Tian et al., 2018) further leverages neuron coverage to guide test generation for DNN-based ADS, which adopts domain-specific image transformations on driving scenes. DeepRoad (Zhang et al., 2018) proposes a GAN-based approach to generate realistic driving scenes for testing DNN-based ADS. DeepBillboard (Zhou et al., 2020) designs an algorithm to generate printable adversarial billboard that can trigger erroneous behaviours of DNN-based ADS. However, these works mainly focus on introducing erroneous behaviours to prove that the ADS under test contain

[1] https://github.com/SEG-DENSE/TACTIC

defects. They do not study the environmental conditions which may be the root causes for the erroneous behaviours. Differently, the goal of TACTIC is to identify critical environmental conditions under which the DNN-based ADS are prone to errors, and generate the corresponding driving scenes to such conditions for testing.

DNN Coverage Criteria. A variety of testing coverage criteria for DNNs have been proposed in order to guide test generation (Pei et al., 2017; Ma et al., 2018; Sun et al., 2018; Ma et al., 2019; Du et al., 2019; Odena et al., 2019). For example, DeepXplore (Pei et al., 2017) designs neuron coverage for DNNs to measure the percentage of neurons that are activated by a given set of test data. DeepGauge (Ma et al., 2018) generalises the concept of neuron coverage and proposes a set of multi-granularity coverage criteria which take the distribution of training data into consideration. TensorFuzz (Odena et al., 2019) debugs DNNs with coverage-guided fuzzing, which is to find adversarial inputs by mainly adding independent noise to individual inputs. It proposes to measure whether coverage has increased on a given test input by using an approximate nearest neighbour (ANN) algorithm to check how far away the nearest neighbour of the input is. In this work, we use two neuron-level coverage criteria selected from DeepGauge to measure the diversity of erroneous behaviours. Note that TACTIC can be easily generalised to other coverage criteria.

Metamorphic Relations for DNNs. In DNN testing, metamorphic relations (Chen et al., 2020) are widely studied to ameliorate the test oracle problem. A common metamorphic relation in testing DNN-based ADS is stated as the driving behaviours of a self-driving car should not change significantly or stay the same for the transformed images (Tian et al., 2018; Zhang et al., 2018; Han & Zhou, 2020). In this work, we leverage a metamorphic relation for steering angle control as the test oracle for detecting erroneous behaviours, and further explore its efficacy by preforming a large-scale study on three open-source DNN-based autonomous driving models and five real-world environmental types.

3. The TACTIC Approach

In this paper, we aim to test DNN-based ADS under different environmental conditions with the goal of identifying the critical environmental conditions under which the ADS are more prone to erroneous behaviours. We focus on DNN-based ADSs that perform end-to-end steering angle control, i.e., the DNN-based ADS can output the steering angles by taking the driving scenes captured by the car-mounted camera as inputs. We choose to study the steering angle control because it is a fundamental function of all kinds of ADS, and high-quality test datasets constructed and used by many existing approaches are available for experiments. TACTIC is not restricted to the steering angle control but can be extended to support the testing of other functions, e.g., car-braking, as discussed in Section 3.2.2.

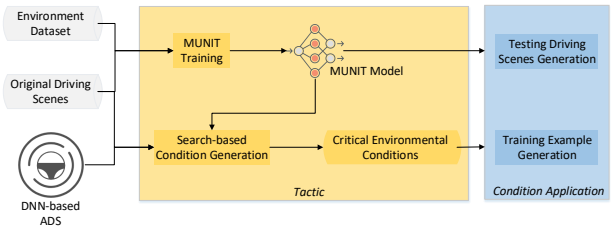


Figure 2. The overview of TACTIC. For a subject DNN-based ADS and a given environmental type, TACTIC first trains a MUNIT model to encode the environmental condition space of the environmental type. Then, TACTIC employs a search-based method, e.g., (1+1) ES in this work, to identify the critical environmental conditions based on the MUNIT model. Finally, testing driving scenes (or more training examples) are generated using the critical environmental conditions. For details see Section 3.

The workflow of TACTIC is shown in Figure 2. For a subject DNN-based ADS and a given environmental type, TACTIC first trains a MUNIT model using two sets of images: the original driving scenes which are used to synthesise testing scenes and the environmental dataset which contains driving scenes belong to the given environmental type. When the training process completes, the environmental condition space of the environmental type is encoded as a style space in the MUNIT model. Details are elaborated in Section 3.1. Then TACTIC sets a search objective which is to maximise the number and diversity of erroneous behaviours and employs a search-based method, e.g., (1+1) Evolution Strategy (ES) (Rechenberg, 1978) in this work, to search the style space for the environmental conditions that reach the search objective, i.e., the critical environmental conditions. Details are presented in Section 3.2 and Section 3.3. Finally, the original driving scenes are transformed by the MUNIT model into synthesised scenes under the critical environmental conditions, to test whether the subject DNN-based ADS can output consistent steering angles under the critical environmental conditions. It is worth mentioning that besides testing, the critical environmental conditions of TACTIC can also be used to synthesise more training examples to improve the performance of DNN-based ADSs by adversarial training (Pei et al., 2017; Tian et al., 2018), which we plan to study in future work.

3.1. Environmental Conditions

To encode the environmental condition space of an environmental type, we leverage MUNIT, a multimodal unsupervised image-to-image translation framework based on Generative Adversarial Networks (GAN). One insight of

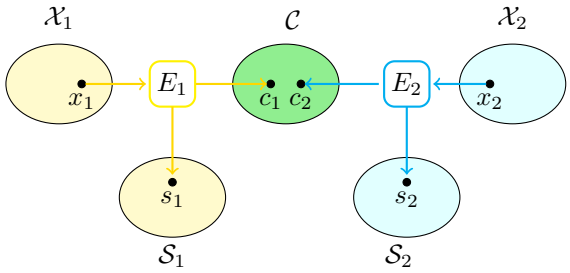


Figure 3. Structure of MUNIT (Huang et al., 2018). Images in each domain $\mathcal{X}_i$ are encoded to a shared content space $\mathcal{C}$ and a domain-specific style space $\mathcal{S}_i$ through an encoder E_i. Each encoder has an inverse decoder G_i omitted from this figure.

MUNIT is that the representation of images can be decomposed into a content space that is domain-invariant and a style space that captures domain-specific characteristics, as shown in Figure 3. Accordingly, when training a MUNIT model based on two sets of driving scenes that belong to two different environmental types, the domain-invariant content space will encode the information shared by scenes of the two different environmental types, such as the road information like the shapes of road and the roadside trees. The specific characteristics of each environmental type, e.g., the unique degrees of illumination, amounts of rain, and cloud patterns for the rainy weather type, would be encoded by the style space. Therefore, we can use the style space of an environmental type in a well-trained MUNIT model to represent the environmental condition space of the type.

After obtaining a MUNIT model, driving scenes under different environmental conditions of an environmental type can be realistically generated. Specifically, suppose that we have a MUNIT model M trained on two sets of driving scenes that belong to two environmental types: the environmental type $\mathcal{X}_1$ and the environmental type $\mathcal{X}_2$. To transform a driving scene x_1 in $\mathcal{X}_1$ to a driving scene x_2 in $\mathcal{X}_2$, the MUNIT model M first uses the encoder E_1 of $\mathcal{X}_1$ to decompose x_1 into a content vector c_1 and a style vector s_1, i.e., $(c_1, s_1) = E_1(x_1)$. Then, to produce x_2, M recombines the content c_1 with a style vector s_2 sampled from the style space $\mathcal{S}_2$ of $\mathcal{X}_2$ by the decoder G_2 of $\mathcal{X}_2$, i.e., $x_2 = G_2(c_1, s_2)$. In the above transformation process, by using another style vector s_2', we can generate a driving scene x_2', i.e., $x_2' = G_2(c_1, s_2')$, which has the same content as x_2 but a different environmental condition. Furthermore, we can also transform the entire set of driving scenes under different environmental conditions into those under the same environmental condition by using the same style vector. More specifically, let $\mathbf{X}_1 = \{x_1, x_2, ..., x_n\}$ be a set of diving scenes under different environmental conditions of $\mathcal{X}_1$. Through the encoder E_1, each scene $x_i \in \mathrm{X}_1$ is decomposed into a content vector c_i and a style vector s_i, i.e., $(c_i, s_i) = E_1(x_i)$. Then, we recombine each content vector c_i with the same style vector s sampled from the style space $\mathcal{S}_2$ of $\mathcal{X}_2$ by the decoder G_2 of $\mathcal{X}_2$ to generate a driving scene x_i', i.e., $x_i' = G_2(c_i, s)$. Finally, we obtain a set of driving scenes under the same environmental condition of $\mathcal{X}_2$, i.e., a set $\mathbf{X}_2 = \{x_1', x_2', ..., x_n'\}$. Figure 4 shows some driving scenes synthesised by respectively using the above two types of transformations, based on the MUNIT models used in TACTIC. More details about MUNIT can be found in Appendix A.

3.2. Search Objective for Critical Environmental Conditions

As introduced in Section 3.1, TACTIC uses the style space in a MUNIT model to represent the environmental condition space of a target environmental type. However, identifying the critical environmental conditions (i.e., critical style vectors) from the style space is still challenging. First, the style space in MUNIT is a high-dimensional vector space, and hence, a style vector in the style space is difficult for interpretation. For example, when transforming a sunny scene to a rainy scene, a style vector may encode information about the light effects, amounts of rain, cloud patterns, and even a mixture of all of them and many more, unlike the configurations in simulators where each configurable variable controls a certain simulation object. Second, the critical environmental conditions may be the corner cases which only occupy a small fraction of the entire space.

To alleviate these challenges, we use the search-based approach, which is proven to be effective for complex optimisation problems, to search the style space for the critical style vectors. The key to the success of the search is a proper fitness function that measures the fitness of the style vectors. The number of erroneous behaviours is certainly important metrics, since it indicates that the ADS are prone to errors. However, using such metrics alone may lead to the same type of errors repeatedly happening. Therefore, we design the fitness function for the style search to consider both the number and the types of errors, i.e., it is the combination of two effectiveness measures: (1) the ability to detect more diverse erroneous behaviours; and (2) the ability to detect more erroneous behaviours.

3.2.1. TESTING COVERAGE CRITERIA

In this work, we employ DNN coverage criteria to measure the ability of a style vector to detect diverse erroneous behaviours. Prior work has proved that increasing the DNN coverage can lead to more diverse behaviours (Pei et al., 2017; Tian et al., 2018; Xie et al., 2019; Huang et al., 2019), which, consequently, can increase the chance to detect more diverse erroneous behaviours. Therefore, a style vector is

Figure 4. Driving scenes synthesised using MUNIT. **Left:** We transform one original driving scene to driving scenes under different environmental conditions of an environmental type by using different styles. **Right:** We transform two driving scenes that have different environmental conditions to those under the same environmental condition of an environmental type by using the same style.

considered to have the potential to find defects of more diversity, if the driving scenes obtained by applying this style vector have higher DNN coverage measures. Many different DNN coverage metrics have been proposed. Since we need to distinguish a large number of style vectors by coverage, we carefully select two fine-grained coverage metrics on the sub-neuron level (Ma et al., 2018) for TACTIC to support, i.e., k-multisection Neuron Coverage (KMNC) and Neuron Boundary Coverage (NBC). It is worth mentioning, however, TACTIC is not bound to any specific metrics and new metrics can be easily supported.

Let O be a set of neurons of a DNN. Let $\phi(x, o)$ be the output value of neuron $o \in O$ for input x. When providing the training dataset D_{train} to the DNN model, the range of the observed neuron activation values of a neuron $o \in O$ is represented as $[low_o, high_o]$, where $low_o = \min_{x \in D_{train}} \phi(x, o)$ and $high_o = \max_{x \in D_{train}} \phi(x, o)$. Moreover, we refer the value range out of $[low_o, high_o]$, i.e., $(-\infty, low_o) \cup (high_o, +\infty)$, as the corner-case regions of the neuron o. Then, KMNC and NBC are defined as follows:

KMNC. Given a neuron $o \in O$, the KMNC measures how thoroughly the given input set T covers the range $[low_o, high_o]$. To quantify this, the range $[low_o, high_o]$ is divided into k equal sections (i.e., k-multisections), for $k > 0$. Let S_i^o be the i-th section with $1 \leq i \leq k$. Then if $\phi(x, o) \in S_i^o$, the i-th section is covered by at least one input $x \in T$. Therefore, the KMNC of a DNN obtained by the test input set T is defined as:

$$\mathsf{KMNC} = \frac{\sum_{o \in O} |\{S_i^o | \exists x \in T : \phi(x, o) \in S_i^o\}|}{k \times |O|}$$

NBC. Given a neuron $o \in O$, the NBC shows to what extent the corner-case regions are covered by the given input set T. If $\phi(x, o)$ belongs to $(-\infty, low_o)$ or $(high_o, +\infty)$, the corresponding corner-case region is covered by at least one input $x \in T$. Therefore, given the test input set T, let $\mathsf{UpperCornerNeuron} = \{o \in O | \exists x \in T : \phi(x, o) \in (high_o, +\infty)\}$ and $\mathsf{LowerCornerNeuron} = \{o \in O | \exists x \in T : \phi(x, o) \in (-\infty, low_o)\}$ be the sets of neurons that ever fall into the corner case regions, respectively. The NBC of a DNN achieved by the test input set T is defined as:

$$\mathsf{NBC} = \frac{|\mathsf{UpperCornerNeuron}| + |\mathsf{LowerCornerNeuron}|}{2 \times |O|}$$

Moreover, we respectively denote the regions that KMNC targets as KMNC regions and the regions that NBC targets as NBC regions in the rest of the paper.

3.2.2. METAMORPHIC ERROR ANALYSIS.

Following the existing work (Tian et al., 2018), we also adopt Metamorphic Relation (MR) (Chen et al., 2020) as the test oracle to determine whether a system behaviour is correct or not. In particular, the MR used in TACTIC is defined as that the steering angles should be consistent among the driving scenes transformed from the same ones by applying different style vectors, i.e., retain behaviour consistency. However, such an MR is too strong to be practical since the acceptable steering angle for a driving scene can be within a range (Tian et al., 2018). Therefore, the MR is further relaxed to a steering angle divergence within an error bound. Formally, let x_o be an original driving scene and x_t be a newly generated driving scene synthesised based on x_o in a target environmental type. The θ_o and θ_t represent the steering angles for x_o and x_t, respectively. Then the MR is defined as $|\theta_o - \theta_t| < \epsilon$, where the ϵ is a user-defined error bound.

Based on the MR, we can assess the tendency of behaviours to be erroneous ones. For a style vector, if the new driving scenes synthesised with this style vector have larger steering

angle divergences compared with the original driving scenes, then the style vector is considered to be able to detect more erroneous behaviours. TACTIC can be easily extended to test other functions of DNN-based ADS by providing a specification depicting the erroneous behaviours for those functions. For example, for the function of car-braking, a simple way to specify the erroneous behaviours can be that the two boolean variables indicating whether or not to brake respectively for the original and synthesised scenes have different values.

3.2.3. FITNESS FUNCTION DEFINITION.

Now we formally define the two effectiveness measures:

The ability to detect diverse erroneous behaviours of a style vector is measured by the increase in the testing coverage measured by KMNC or NBC. Let R_t be the set of uncovered KMNC/NBC regions so far during the search-based testing process and R_s be the set of regions that will be newly covered by the synthesised driving scenes using style vector s. The ability of s to detect diverse erroneous behaviours is calculated as: $F_c(s) = \frac{|R_s|}{|R_t|}$.

The ability to detect more erroneous behaviours of a style vector is measured by the mean steering angle divergences between the original dataset of driving scenes and the newly synthesised driving scenes with that style vector. Formally, let $g(x, s)$ be the function that transforms a driving scene x into another in the target environmental type using style vector s, and $f(x)$ be the function that returns the steering angle for the driving scene x. Let I_o be the original dataset of driving scenes, the ability of a style vector s to detect more erroneous behaviours is calculated as:

$$F_d(s) = \frac{1}{|I_o|} \sum_{x \in I_o} |f(x) - f(g(x, s))|.$$

Based on the above two effectiveness measures, we design the fitness function for TACTIC as follows:

$$F(s) = w_c * F_c(s) + w_d * norm(F_d(s)),$$

where s is a style vector, $norm(x) = \frac{x}{x+1}$ is a normalisation function (Greer & Ruhe, 2004; Zhang et al., 2008) for normalising the value of F_d in the same magnitude with F_c (i.e., within the same range of value $[0, 1]$), and w_c, w_d are the weights assigned to two measure, respectively, which allow for prioritisation of the measures. In this work, we simply treat the two measures as equal by setting $w_c = w_d = 1$, and leave the study of weight optimisation for future work.

3.3. Search of Critical Environmental Conditions

We are now ready to present how TACTIC identifies the critical environmental conditions of a given environmental type for a subject DNN-based ADS. We choose (1+1) Evolution Strategy (ES) (Rechenberg, 1978), which has shown to be effective in previous studies (Ali et al., 2013; Arcuri, 2013; Ji et al., 2018), as the search algorithm in TACTIC to identify the critical environmental conditions. TACTIC receives as inputs the subject DNN-based ADS N , the MUNIT model M trained for the given environmental type, and a set I_o of the original driving scenes. The output of TACTIC is a set S of the critical environmental conditions of the given environmental type.

Specifically, TACTIC iteratively calls the (1+1) ES to identify a critical environmental condition until a user-defined number of such conditions are obtained. Each iteration of TACTIC consists of the following main steps: *First*, the (1+1) ES is initialised with an initial individual, which is a style vector randomly sampled from the style space in the MUNIT model M. *Second*, the (1+1) ES evolves the individual s via selection and reproduction. In particular, a new individual s' is firstly generated based on the current individual s. Then, the fitness values of s and s' is evaluated as described in Section 3.2.3. After that, the individual with higher fitness value is kept as the current individual in the next evolution. *Lastly*, the (1+1) ES terminates the evolution process when the stopping condition is met and the current individual is added to the set S as a critical environmental condition. The stopping condition could be many, for example, the level of improvement during the evolution process or maximum time budget, etc. In this work, the stopping condition is set conservatively to that the fitness value has no improvement in 100 successive iterations.

TACTIC stops when a user-defined number of critical conditions are obtained, at which point TACTIC returns the critical environmental condition set S. In this work, we set the number to be four, as we observed in the experiments that increasing the number of style vectors when it is already above four would not result in a significant increase in the test coverage. Note that this number is related to the number of the driving scenes in the test dataset, and can be easily configured when necessary.

4. Experimental Evaluation

In this section, we present the experimental results to demonstrate the effectiveness of TACTIC. We mainly focus on evaluating whether TACTIC is able to find critical environmental conditions, which are used to synthesise driving scenes that can effectively reveal erroneous behaviours in different environmental types. Additionally, testing ADS under environmental conditions requires that the synthesised driving scenes realistically reflect reality, and therefore, we study the realism of the synthesised driving scenes.

4.1. Experimental Settings

Datasets and DNN-based ADSs. We conduct experiments on the Udacity dataset (Udacity, 2016). To demonstrate the effectiveness of TACTIC, we consider three popular pre-trained DNN-based ADSs, which have been widely used in previous work (Pei et al., 2017; Tian et al., 2018; Zhang et al., 2018; Zhou et al., 2020), i.e., Dave-orig (0bserver07, 2016), Dave-dropout (Navoshta, 2016), and Chauffeur (Emef, 2016). For each DNN-based ADSs, we study five environmental types (night, sunshine, rain, snow in daytime and snow in night) that are representatives of the runtime environments for DNN-based ADSs. The corresponding dataset of each environmental type is collected from YouTube. See more details about the datasets and targeted DNN-based ADSs in Appendix B.

Baselines. We consider two groups of baselines. (1) For the effectiveness comparison in Section 4.2 , we evaluate the effectiveness of TACTIC against two methods, i.e., an approach using randomly sampled environmental conditions (denoted as $\mathrm{R_c}$) and the state-of-the-art DeepRoad (Zhang et al., 2018). (2) For the image quality comparison in Section 4.3, we compare the realism of the driving scenes synthesised by TACTIC with three methods, i.e., DeepRoad (Zhang et al., 2018), DeepTest (Tian et al., 2018), and PreScan (Marketakis et al., 2009), which are the current state-of-the-art in testing DNN-based ADSs. Note that we do not include DeepTest and PreScan in the effectiveness comparison (c.f., Section 4.2) due to the synthesised driving scenes by these methods are unrealistic, as shown in Section 4.3.

Evaluation Metrics We use two metrics to evaluate the effectiveness of TACTIC: (1) the achieved test coverage (i.e., KMNC and NBC) and (2) the number of detected erroneous behaviours. For the achieved coverage, we focus on measuring the achieved coverage for CNNs used in the ADSs due to the coverage criteria are less effective on testing RNNs (Tian et al., 2018) and we set the number k of KMNC to 1000, consistent with DeepGauge (Ma et al., 2018). For the number of detected erroneous behaviours, we present the number of erroneous behaviours detected under four different error bounds (c.f. Section 3.2.2), which are consistent with the error bounds used in DeepRoad (Zhang et al., 2018). Regarding the quality of synthesised driving scenes, we conduct a user study to evaluate the realism of the synthesised driving scenes, since currently, user studies are the most effective standard to evaluate the realism of objects artificially synthesised.

4.2. Comparison with Baselines on Effectiveness

Setup. We execute TACTIC with two coverage-guiding strategies: KMNC (denoted as $\text{TACTIC}^{\text{KMNC}}$) and NBC (denoted as $\text{TACTIC}^{\text{NBC}}$), respectively, on each of the three subject DNN-based ADSs, under the five environmental types. In each execution, the entire set of driving scenes from the Udacity testing dataset is used as the original driving scenes (i.e., I_o in Section 3.3), and 4 critical environmental conditions are generated (c.f. Section 3.3), which are separately applied on the Udacity testing dataset to synthesise testing driving scenes (i.e., 4×5614 testing driving scenes are totally generated).

For $\mathrm{R_c}$, when testing the three subject systems, we randomly sample 4 style vectors (equal to the number of critical style vectors generated by TACTIC) for each of the environmental types and separately apply the 4 random style vectors on the Udacity testing dataset to synthesise testing driving scenes (i.e., 4×5614 testing driving scenes are totally generated). Moreover, to reduce the randomness of (1+1) ES used in TACTIC and $\mathrm{R_c}$, we conduct 10 runs for each experimental case and average the results.

For DeepRoad, we use the training datasets of the MUNIT models in TACTIC for training DeepRoad, and then generate driving scenes for each of the environmental types. Note that, given the Udacity testing dataset, DeepRoad can only generate the same number (5614) of testing driving scenes as the Udactiy dataset for each environmental type, since it transforms each driving scene into the other in a deterministic way. Therefore, for TACTIC, we apply just one critical style on the Udactiy testing dataset each time and then average the results for a fair comparison.

Due to the space limit, we report the results of NBC-guided TACTIC on Dave-orig in the main body of the paper. Results of KMNC-guided TACTIC on Dave-orig and results of TACTIC on Dave-dropout and Chauffeur can be found in Appendix C.1. The results on all the coverage-guiding strategies and the DNN-based ADSs are consistent.

Results. Table 1 and Table 2 summarise the results of comparing $\text{TACTIC}^{\text{NBC}}$ with $\mathrm{R_c}$ and DeepRoad, respectively. Better results are highlighted with a darker background. We discuss the results from two aspects: (1) the achieved test coverage, and (2) the number of detected erroneous behaviours. In terms of the achieved coverage, compared with $\mathrm{R_c}$, TACTIC achieves higher coverage in all environmental types, demonstrating that TACTIC can detect more diverse erroneous behaviours than $\mathrm{R_c}$. Compared with DeepRoad, TACTIC achieves slightly lower coverage in the environmental type of "Snow in Daytime" while obtaining higher coverage than DeepRoad in all the other four environmental types. We leave the analysis about why DeepRoad achieves higher coverage in Appendix C.1. In terms of the detected erroneous behaviours, compared with $\mathrm{R_c}$ and DeepRoad, TACTIC detects many more erroneous behaviours in all environmental types for both four error bounds. Furthermore, we also observe that, when the error bound increases, $\mathrm{R_c}$ and DeepRoad hardly detects erroneous behaviours while TAC-

Table 1. Results of comparing NBC-guided TACTIC with R_c on Dave-orig. Better results are highlighted with a darker background.

ENV. TYPE		NIGHT		SUNSHINE		RAIN		SNOW IN DAYTIME		SNOW IN NIGHT	
METHOD		TACTIC	R_c	TACTIC	R_c	TACTIC	R_c	TACTIC	R_c	TACTIC	R_c
COVERAGE	KMNC	73.68%	43.55%	56.24%	43.34%	50.71%	42.04%	54.41%	47.15%	72.09%	52.04%
	NBC	35.92%	3.18%	13.81%	1.97%	7.30%	2.00%	8.67%	3.88%	33.30%	10.10%
NUMBER OF ERRORS	10°	18675.1	2971.2	2629.8	1300.5	9795.2	1484.5	13450.8	3605.2	20879.0	7323.4
	20°	13790.0	269.7	196.5	103.8	3479.7	44.9	3396.8	197.7	17978.1	2583.0
	30°	8627.7	25.5	22.7	4.7	1933.8	0.1	845.1	10.6	14561.1	749.7
	40°	4303.7	1.9	0.7	0.0	687.7	0.0	209.5	0.8	12074.7	72.5

Table 2. Results of comparing NBC-guided TACTIC with DeepRoad on Dave-orig. Better results are highlighted with a darker background.

ENV. TYPE		NIGHT		SUNSHINE		RAIN		SNOW IN DAYTIME		SNOW IN NIGHT	
METHOD		TACTIC	DEEPROAD	TACTIC	DEEPROAD	TACTIC	DEEPROAD	TACTIC	DEEPROAD	TACTIC	DEEPROAD
COVERAGE	KMNC	54.59%	40.99%	45.37%	40.97%	40.71%	40.23%	41.80%	45.21%	60.03%	55.66%
	NBC	21.72%	5.99%	6.62%	2.05%	2.90%	2.05%	3.34%	3.81%	26.64%	21.50%
NUMBER OF ERRORS	10°	4885.0	613.0	708.8	355.0	3035.8	487.0	3708.0	1250.0	5041.0	3189.0
	20°	3898.3	114.0	41.5	40.0	1442.5	20.0	820.5	90.0	4418.0	1996.0
	30°	2662.5	33.0	4.8	3.0	816.5	0.0	154.3	2.0	3716.3	802.0
	40°	1620.5	9.0	0.0	0.0	304.5	0.0	47.3	0.0	2843.0	173.0

TIC still shows strong ability to detect erroneous behaviours. In summary, TACTIC can effectively identify critical environmental conditions, and reveal more diverse and more serious erroneous behaviours than R_c and DeepRoad.

Time Efficiency. We also analysed the efficiency of TACTIC. Compared to R_c and DeepRoad, TACTIC spent more time on testing DNN-based ADS, since TACTIC needs to search the style space to identify the critical environmental conditions. Specifically, about a total of 15 hours, including the time cost for driving scenes generation, model prediction and fitness function calculation, was required to identify a critical environmental condition in our experimental environment. Such a time cost is worthwhile, since TACTIC can obtain the critical environmental conditions under which the DNN-based ADS are more prone to errors by a relatively comprehensive exploration of the condition space of an environmental type. This cannot be done by existing approaches using synthesised driving scenes, and would require substantially more time for real-world tests (Kalra & Paddock, 2016).

4.3. Comparison with Baselines on Image Quality

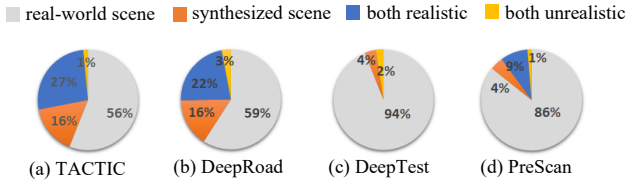
Setup. We conduct a user study to evaluate the realism of the testing driving scenes synthesised by TACTIC through comparing them with the test driving scenes synthesised by DeepRoad (Zhang et al., 2018), DeepTest (Tian et al., 2018), and PreScan (Marketakis et al., 2009). Specifically, we design an online questionnaire consisting of two questions: (1) "Which driving scene is more realistic?", and (2) "Which environmental type does the driving scene belong to?".

In the first questionnaire, 80 image pairs, each containing one synthesised image and one real image, were shown to the participants. Among the 80 synthesised images, each of the four approaches synthesised 20 images. The participants were asked to choose one of the four choices: the first image is more realistic, the second image is more realistic, both are realistic, and both are unrealistic. This question intends to compare the degrees of realism between the real-world driving scenes and the ones synthesised by different approaches.

In the second questionnaire, the same 80 synthesised images that were used in the first questionnaire were shown to the participants. The participants were asked to choose one environmental type for the driving scenes from the seven choices: rain, sunshine, night, fog, snow in daytime, snow in night, and image unrealistic. This question intends to compare the ability to synthesise testing driving scenes to realistically reflect the environmental conditions. All participants were given the same image pairs (or images).

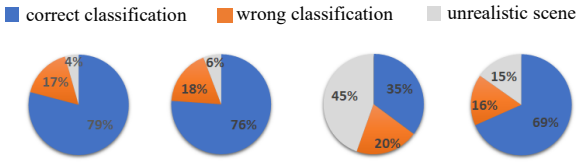
The questionnaire was distributed online, open to both students and industrial practitioners.

Results. We received answers from a total of 34 participants of which 12 are from industry and 22 are students. Figure 5 and Figure 6 present the results of answers to the first and the second questions, respectively. The results demonstrate that the testing driving scenes synthesised by TACTIC are more realistic than the ones synthesised by affine image transformation (e.g., DeepTest) or high-fidelity simulation (e.g., PreScan) and can better reflect real environments. Specifically, from Figure 5, it can be observed that there are nearly half of driving scene pairs where the synthesised driving scenes cannot be distinguished with the



(a) TACTIC (b) DeepRoad (c) DeepTest (d) PreScan

Figure 5. Comparison of realism among real-world driving scenes and synthesised scenes of (a) TACTIC, (b) DeepRoad, (c) DeepTest, and (d) PreScan. The grey sectors denote the ratios that the real-world scenes are selected. Therefore, the smaller the grey sectors are, the more realistic the synthesised scenes are.



(a) TACTIC (b) DeepRoad (c) DeepTest (d) PreScan

Figure 6. Ratios of correct classification of the driving scenes synthesised from (a) TACTIC, (b) DeepRoad, (c) DeepTest, and (d) PreScan. The blue sectors denote the ratios that the synthesised scenes are correctly classified as their corresponding environmental types. Therefore, the larger blue sectors suggest that the synthesised scenes can more realistically reflect the environmental types.

real-world ones in human perception for TACTIC (44%) and DeepRoad (41%). In contrast, for DeepTest and PreScan, there are on average 94% and 86% driving scene pairs where the real-world driving scenes are considered more realistic, respectively. As shown in Figure 6, in general, the testing driving scenes synthesised by TACTIC and DeepRoad can more realistically reflect the environmental types compared to the ones synthesised by DeepTest and PreScan.

Furthermore, we observe that the testing driving scenes synthesised by TACTIC and DeepRoad have comparable results for the degrees of realism. The reason is that both TACTIC and DeepRoad use a type of GAN to generate new testing driving scenes. However, as shown in Section 4.2, compared with DeepRoad, TACTIC manages to detect more diverse and more serious erroneous behaviours.

4.4. Ablation Study

Setup. We perform the ablation study to justify the selection of (1+1) ES in TACTIC. In particular, we implement a new version of TACTIC by replacing the search algorithm (1+1) ES with a random search (RS) and compare the effectiveness of the two versions. Except for the search algorithm, the other implementation and settings, e.g., the fitness function, are not changed. For each experimental case, we also conduct 10 runs and average the results to reduce the randomness in the search. Again, we mainly report the results of NBC-guided TACTIC on Dave-orig in the main body of this paper, and leave the complete results to Appendix C.2.

Table 3. Results of comparing the effectiveness of (1+1) ES and RS in NBC-guided TACTIC on Dave-orig. Better results are highlighted with a darker background.

ENV. TYPE	METHOD	COVERAGE		NUMBER OF ERRORS			
		KMNC	NBC	10^o	20^o	30^o	40^o
NIGHT	(1+1) ES	73.68%	35.92%	18675.1	13790	8627.7	4303.7
	RS	53.09%	10.73%	15833.3	4783.0	384.0	9.1
SUNSHINE	(1+1) ES	56.24%	13.81%	2629.8	196.5	22.7	0.7
	RS	45.48%	2.85%	1451.4	81.2	5.0	0.0
RAIN	(1+1) ES	50.71%	7.30%	9795.2	3479.7	1993.8	687.7
	RS	42.35%	2.00%	3633.9	188.1	3.8	0.0
SNOW IN DAYTIME	(1+1) ES	54.41%	8.67%	13450.8	3396.8	845.1	209.5
	RS	51.87%	6.80%	9730.0	748.8	20.0	14.3
SNOW IN NIGH	(1+1) ES	72.09%	33.30%	20879.0	17978.1	14561.1	12074.7
	RS	65.45%	25.69%	19341.5	14917.3	9955.4	4966.7

Results. Table 3 summarises the comparison results and we also discuss the results from the achieved test coverage and the number of detected erroneous behaviours. For the achieved test coverage, TACTIC using (1+1) ES achieves higher coverage than TACTIC using RS in both five environmental types. For the number of the detected erroneous behaviours, TACTIC using (1+1) ES obtains significantly more erroneous behaviours on all environmental types. Additionally, we also observe that the difference between the number of erroneous behaviours detected by the two approaches increases with the error bound. Such results demonstrate that (1+1) ES has better performance in identifying critical environmental conditions than random search.

5. Conclusion

In this paper, we propose to study the impact of different environmental conditions on the DNN-based ADS and raise the problem of identifying the critical environmental conditions that would make the system under test more prone to erroneous behaviours. We introduce a novel approach TACTIC, which employs a search-based method to search the environmental condition space of the given environmental type generated by MUNIT. Large-scale experiments demonstrate that TACTIC can effectively identify critical environmental conditions and synthesise realistic testing driving scenes. Compared to the state-of-the-art approaches, TACTIC can reveal more diverse and more erroneous behaviours for the popular DNN-based ADSs, and meanwhile, reach a satisfactory testing coverage.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (Nos. 62032010 and 61972193) and the Fundamental Research Funds for the Central Universities of China (No. 14380027).

References

0bserver07. Nvidia-autopilot-keras: End to end learning for self-driving cars, 2016. URL https://github.com/0bserver07/Nvidia-Autopilot-Keras. [online, accessed 27-May-2021].

Abdessalem, R. B., Nejati, S., Briand, L. C., and Stifter, T. Testing advanced driver assistance systems using multi-objective search and neural networks. In Lo, D., Apel, S., and Khurshid, S. (eds.), *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, ASE 2016, Singapore, September 3-7, 2016*, pp. 63–74. ACM, 2016. doi: 10.1145/2970276.2970311. URL https://doi.org/10.1145/2970276.2970311.

Abdessalem, R. B., Nejati, S., Briand, L. C., and Stifter, T. Testing vision-based control systems using learnable evolutionary algorithms. In Chaudron, M., Crnkovic, I., Chechik, M., and Harman, M. (eds.), *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, pp. 1016–1026. ACM, 2018. doi: 10.1145/3180155.3180160. URL https://doi.org/10.1145/3180155.3180160.

Ali, S., Iqbal, M. Z. Z., Arcuri, A., and Briand, L. C. Generating test data from OCL constraints with search techniques. *IEEE Trans. Software Eng.*, 39(10):1376–1402, 2013. doi: 10.1109/TSE.2013.17. URL https://doi.org/10.1109/TSE.2013.17.

Arcuri, A. It really does matter how you normalize the branch distance in search-based software testing. *Softw. Test. Verification Reliab.*, 23(2):119–147, 2013. doi: 10.1002/stvr.457. URL https://doi.org/10.1002/stvr.457.

Arcuri, A. and Briand, L. C. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *Softw. Test. Verification Reliab.*, 24(3):219–250, 2014. doi: 10.1002/stvr.1486. URL https://doi.org/10.1002/stvr.1486.

Badvboynofear. Driving in the snow, Mar 2018. URL https://www.youtube.com/watch?v=fm5nKWeVNGI. [online, accessed 27-May-2021].

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016. URL http://arxiv.org/abs/1604.07316.

Boudette, N. E. Teslas self-driving system cleared in deadly crash. *The New York Times*, 19, 2017.

Capon, J. A. Elementary statistics for the social sciences: Study guide, 1991.

Chen, T. Y., Cheung, S. C., and Yiu, S. Metamorphic testing: A new approach for generating next test cases. *CoRR*, abs/2002.12543, 2020. URL https://arxiv.org/abs/2002.12543.

Du, X., Xie, X., Li, Y., Ma, L., Liu, Y., and Zhao, J. Deepstellar: model-based quantitative analysis of stateful deep learning systems. In Dumas, M., Pfahl, D., Apel, S., and Russo, A. (eds.), *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26-30, 2019*, pp. 477–487. ACM, 2019. doi: 10.1145/3338906.3338954. URL https://doi.org/10.1145/3338906.3338954.

Emef. Sdc, 2016. URL https://github.com/emef/sdc. [online, accessed 27-May-2021].

Fremont, D. J., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A. L., and Seshia, S. A. Scenic: a language for scenario specification and scene generation. In McKinley, K. S. and Fisher, K. (eds.), *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22-26, 2019*, pp. 63–78. ACM, 2019. doi: 10.1145/3314221.3314633. URL https://doi.org/10.1145/3314221.3314633.

Gibbs, S. Ubers self-driving car saw the pedestrian but didnt swerve–report. *The Guardian*, 2018.

Greer, D. and Ruhe, G. Software release planning: an evolutionary and iterative approach. *Inf. Softw. Technol.*, 46(4):243–253, 2004. doi: 10.1016/j.infsof.2003.07.002. URL https://doi.org/10.1016/j.infsof.2003.07.002.

Grzywaczewski, A. Training ai for self-driving vehicles: the challenge of scale, 2017.

Han, J. C. and Zhou, Z. Q. Metamorphic fuzz testing of autonomous vehicles. In *ICSE '20: 42nd International Conference on Software Engineering, Workshops, Seoul, Republic of Korea, 27 June - 19 July, 2020*, pp. 380–385. ACM, 2020. doi: 10.1145/3387940.3392252. URL https://doi.org/10.1145/3387940.3392252.

Huang, W., Sun, Y., Huang, X., and Sharp, J. testrnn: Coverage-guided testing on recurrent neural networks. *CoRR*, abs/1906.08557, 2019. URL http://arxiv.org/abs/1906.08557.

Huang, X., Liu, M., Belongie, S. J., and Kautz, J. Multimodal unsupervised image-to-image translation. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y. (eds.), *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part III*, volume 11207 of *Lecture Notes in Computer Science*, pp. 179–196. Springer, 2018. doi: 10.1007/978-3-030-01219-9_11. URL https://doi.org/10.1007/978-3-030-01219-9_11.

Ji, R., Li, Z., Chen, S., Pan, M., Zhang, T., Ali, S., Yue, T., and Li, X. Uncovering unknown system behaviors in uncertain networks with model and search-based testing. In *11th IEEE International Conference on Software Testing, Verification and Validation, ICST 2018, Västerås, Sweden, April 9-13, 2018*, pp. 204–214. IEEE Computer Society, 2018. doi: 10.1109/ICST.2018.00029. URL https://doi.org/10.1109/ICST.2018.00029.

Kalra, N. and Paddock, S. M. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, 2016.

Li, Z., Ma, X., Xu, C., Cao, C., Xu, J., and Lü, J. Boosting operational DNN testing efficiency through conditioning. In Dumas, M., Pfahl, D., Apel, S., and Russo, A. (eds.), *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26-30, 2019*, pp. 499–509. ACM, 2019. doi: 10.1145/3338906.3338930. URL https://doi.org/10.1145/3338906.3338930.

Ma, L., Juefei-Xu, F., Zhang, F., Sun, J., Xue, M., Li, B., Chen, C., Su, T., Li, L., Liu, Y., Zhao, J., and Wang, Y. Deepgauge: multi-granularity testing criteria for deep learning systems. In Huchard, M., Kästner, C., and Fraser, G. (eds.), *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018*, pp. 120–131. ACM, 2018. doi: 10.1145/3238147.3238202. URL https://doi.org/10.1145/3238147.3238202.

Ma, L., Juefei-Xu, F., Xue, M., Li, B., Li, L., Liu, Y., and Zhao, J. Deepct: Tomographic combinatorial testing for deep learning systems. In Wang, X., Lo, D., and Shihab, E. (eds.), *26th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2019, Hangzhou, China, February 24-27, 2019*, pp. 614–618. IEEE, 2019. doi: 10.1109/SANER.2019.8668044. URL https://doi.org/10.1109/SANER.2019.8668044.

Marketakis, Y., Tzanakis, M., and Tzitzikas, Y. Prescan: towards automating the preservation of digital objects. In Chbeir, R., Badr, Y., Kapetanios, E., and Traina, A. J. M. (eds.), *MEDES '09: International ACM Conference on Management of Emergent Digital EcoSystems, Lyon, France, October 27-30, 2009*, pp. 404–411. ACM, 2009. doi: 10.1145/1643823.1643898. URL https://doi.org/10.1145/1643823.1643898.

McGowan, D. Driving on snow - greenville, nc 1-4-2018 at 7:00am, Jan 2018. URL https://www.youtube.com/watch?v=ps56tnQG8V0. [online, accessed 27-May-2021].

Navoshta. Behavioral cloning: End to end learning for self-driving cars, 2016. URL https://github.com/navoshta/behavioral-cloning. [online, accessed 27-May-2021].

Odena, A., Olsson, C., Andersen, D. G., and Goodfellow, I. J. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4901–4911. PMLR, 2019. URL http://proceedings.mlr.press/v97/odena19a.html.

Pei, K., Cao, Y., Yang, J., and Jana, S. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pp. 1–18. ACM, 2017. doi: 10.1145/3132747.3132785. URL https://doi.org/10.1145/3132747.3132785.

Rechenberg, I. Evolutionsstrategien. In *Simulationsmethoden in der Medizin und Biologie*, pp. 83–114. Springer, 1978.

Sun, Y., Huang, X., and Kroening, D. Testing deep neural networks. *CoRR*, abs/1803.04792, 2018. URL http://arxiv.org/abs/1803.04792.

Tian, Y., Pei, K., Jana, S., and Ray, B. Deeptest: automated testing of deep-neural-network-driven autonomous cars. In Chaudron, M., Crnkovic, I., Chechik, M., and Harman, M. (eds.), *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, pp. 303–314. ACM, 2018. doi: 10.1145/3180155.3180220. URL https://doi.org/10.1145/3180155.3180220.

Tours, D. C. Driving on california freeway from calabasas to universal studios hollywood.no music, Jun 2018. URL https://www.youtube.com/

watch?v=01_SVshk-MY. [online, accessed 27-May-2021].

Udacity. The udacity open source self-driving car project, 2016. URL https://github.com/udacity/self-driving-car. [online, accessed 27-May-2021].

Utah, J. Los angeles 4k - night drive, Feb 2019. URL https://www.youtube.com/watch?v=lTvYjERVAnY. [online, accessed 27-May-2021].

Vargha, A. and Delaney, H. D. A critique and improvement of the cl common language effect size statistics of mcgraw and wong. *Journal of Educational and Behavioral Statistics*, 25(2):101–132, 2000.

Vids, A. Rain on a car roof - 1 hour - asmr, Apr 2014. URL https://www.youtube.com/watch?v=O88fXBx-Qdg. [online, accessed 27-May-2021].

Xie, X., Ma, L., Juefei-Xu, F., Xue, M., Chen, H., Liu, Y., Zhao, J., Li, B., Yin, J., and See, S. Deephunter: a coverage-guided fuzz testing framework for deep neural networks. In Zhang, D. and Møller, A. (eds.), *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2019, Beijing, China, July 15-19, 2019*, pp. 146–157. ACM, 2019. doi: 10.1145/3293882.3330579. URL https://doi.org/10.1145/3293882.3330579.

Zhang, M., Zhang, Y., Zhang, L., Liu, C., and Khurshid, S. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In Huchard, M., Kästner, C., and Fraser, G. (eds.), *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018*, pp. 132–142. ACM, 2018. doi: 10.1145/3238147.3238187. URL https://doi.org/10.1145/3238147.3238187.

Zhang, Y., Finkelstein, A., and Harman, M. Search based requirements optimisation: Existing work and challenges. In Paech, B. and Rolland, C. (eds.), *Requirements Engineering: Foundation for Software Quality, 14th International Working Conference, REFSQ 2008, Montpellier, France, June 16-17, 2008, Proceedings*, volume 5025 of *Lecture Notes in Computer Science*, pp. 88–94. Springer, 2008. doi: 10.1007/978-3-540-69062-7_8. URL https://doi.org/10.1007/978-3-540-69062-7_8.

Zhou, H., Li, W., Kong, Z., Guo, J., Zhang, Y., Yu, B., Zhang, L., and Liu, C. Deepbillboard: systematic physical-world testing of autonomous driving systems. In Rothermel, G. and Bae, D. (eds.), *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, pp. 347–358. ACM, 2020. doi: 10.1145/3377811.3380422. URL https://doi.org/10.1145/3377811.3380422.