DeepManeuver: Adversarial Test Generation for Trajectory Manipulation of Autonomous Vehicles

Meriel von Stein, David Shriver, and Sebastian Elbaum
University of Virginia
Charlottesville, VA, USA
{meriel, dls2fc, selbaum}@virginia.edu

***Abstract*—Adversarial test generation techniques aim to produce input perturbations that cause a DNN to compute incorrect outputs. For autonomous vehicles driven by a DNN, however, the effect of such perturbations are attenuated by other parts of the system and are less effective as vehicle state evolves. In this work we argue that for adversarial testing perturbations to be effective on autonomous vehicles, they must account for the subtle interplay between the DNN and vehicle states. Building on that insight, we develop DeepManeuver, an automated framework that interleaves adversarial test generation with vehicle trajectory physics simulation. Thus, as the vehicle moves along a trajectory, DeepManeuver enables the refinement of candidate perturbations to: (1) account for changes in the state of the vehicle that may affect how the perturbation is perceived by the system; (2) retain the effect of the perturbation on previous states so that the current state is still reachable and past trajectory is preserved; and (3) result in multi-target maneuvers that require fulfillment of vehicle state sequences (e.g. reaching locations in a road to navigate a tight turn). Our assessment reveals that DeepManeuver can generate perturbations to force maneuvers more effectively and consistently than state-of-the-art techniques by 20.7 percentage points on average. We also show DeepManeuver's effectiveness at disrupting vehicle behavior to achieve multi-target maneuvers with a minimum 52% rate of success.**

***Index Terms*—test generation, adversarial testing, autonomous systems**

I. Introduction

As vehicles with varying levels of autonomy take to the road, their failures become more noticeable and impactful [1], [2], [3], [4]. Recent failures caused by environmental features such as Burger King signs, lit-up emergency vehicles, and the full moon have garnered significant attention. Among the subsequent efforts to identify underlying faults of these systems, adversarial testing approaches targeting learned components have emerged as fundamental to assess system robustness. These approaches generate input perturbations for deep neural networks (DNN) that force the network to compute erroneous outputs. Though effective at uncovering inputs that may lead to problematic DNN behavior, translating the successes of adversarial testing into system-level failures in autonomous vehicles remains challenging.

Consider a DNN that consumes an image and produces a steering angle for the vehicle in Figure 1.

White-box adversarial testing approaches for DNNs [5], [6], [7] analyze that vehicle's DNN and an input image to generate a realistic image perturbation that cause the DNN to produce an erroneous steering angle. Such perturbations are termed

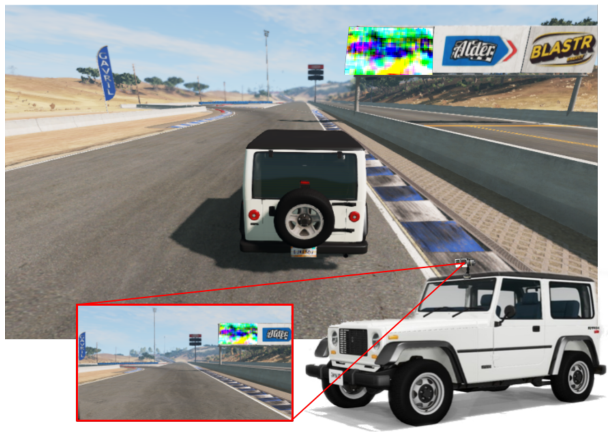
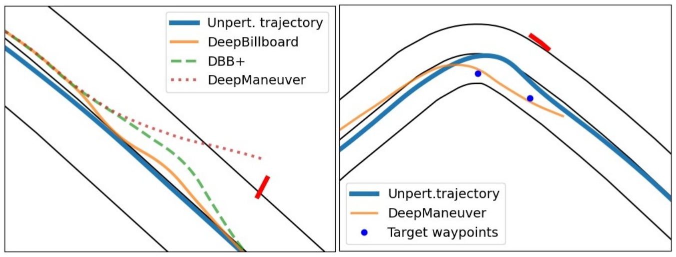


Fig. 1: Autonomous vehicle driving by a roadside billboard with perturbation generated by a patch attack.

patch attacks when performed on physical attack surfaces (see Section IV). Figure 1 illustrates such an attack surface as the leftmost billboard on the right-hand side of the road. A single erroneous DNN output caused by an input image, however, is unlikely to cause a sustained steering deviation of the vehicle for a couple of reasons. First, the system will have constraints on maximum changes in steering angle, bounding the impact of DNN output. Second, error caused by a single perturbed image will be quickly overwritten as dozens of images are processed per second [8], [9], [10]. To address these weaknesses, more sophisticated DNN analyses [11], [12], [13] consume multiple images from across a vehicle's trajectory. This produces a more general perturbation that can be effective under the multiple vehicle states from which those images were captured. Paradoxically, as we demonstrate in Section III, the more the perturbation pushes the vehicle away from the original trajectory on which images were collected, the less effective the perturbation becomes, as it was not generated on images derived from those new vehicle states.

Black box testing approaches typically take the scenario configuration as the attack surface (e.g., [14], [15], [16], [17], [18]), operating at a higher level of abstraction by manipulating simulated entities (e.g.,traffic vehicles, pedestrians, road topology, weather) rather than individual pixels. After a new test scenario configuration is executed, these techniques analyze the state or series of states of the ego vehicle to judge the test outcome and guide the generation of future tests. While the analysis of vehicle state ensures that the tests are



(a) Single-target perturbations forcing a left turn. (b) Multi-target perturbation forcing a tight turn.

Fig. 2: Billboard shown in red, unperturbed trajectory in thick blue, perturbed trajectories as thin multicolored lines.

consequential to system performance, by abstracting the DNN as a black box these approaches are ineffective in exploring the large input space to generate adversarial tests.

Underlying the limitations of both white and black box testing approaches is the idea that errors have compounding, unanticipated effects on subsequent system states. The white box, DNN-driven adversarial perturbation generation process is decoupled from changes in vehicle state, whereas the black box, system-driven testing process fails to analyze vulnerable learned system components and their connection to system state. This decoupling of test generation and secondary effects of perturbed inputs on system state precludes the necessary refinement of the perturbation as the features of the inputs to the vehicle change during its trajectory.

Building on that insight, we develop DeepManeuver, the first **state-adaptive** adversarial testing approach for autonomous vehicles. DeepManeuver is an automated framework that interleaves adversarial test generation with vehicle simulation. To enable state-adaptive test generation, DeepManeuver embeds a simulator into the adversarial generation cycle such that, instead of collecting states and then generating a perturbation, state collection and perturbation generation are interleaved. Thus, as the vehicle travels a trajectory, DeepManeuver is able to generate candidate perturbations to: (1) account for changes in the vehicle's state that may affect how the perturbation is perceived (i.e., the position and heading of the vehicle in the road), (2) retain the effect of the perturbation on previous states so that the current state is still valid, and (3) result in maneuvers – sequences of target system states – that require complex evolution of state and disrupt the trajectory of the vehicle in a predetermined, specifiable way. This paper explores a range of maneuvers, such as making a tight turn, running off the road, or crashing into an obstacle.

State-adaptive adversarial testing allows for the goals of perturbation generation to adapt with changes in the current state and state-dependent capabilities of the target system to better induce future state changes. In this way, we are able to generate test cases that are feasible within the constraints of the system and environment, while still concentrating our efforts on the weaknesses inherent in the DNN.

As we shall show, by accounting for system state during perturbation generation, DeepManeuver can induce complex maneuvers more effectively than state of the art techniques. Figure 2a illustrates an original vehicle trajectory in blue, and three trajectories caused by perturbations aiming to achieve a single-target maneuver of steering left off the road. The red segment next to the road is the billboard where perturbations are injected, the green and yellow lines depict trajectories caused by perturbations generated with existing techniques that do not fully account for state, and the dotted red line depicts the trajectory influenced by DeepManeuver. Note how existing techniques can generate a perturbation that causes a temporary deviation but eventual recover, whereas DeepManeuver generates a perturbation that causes the vehicle to leave the road. Equally significant, DeepManeuver extends the capabilities of adversarial perturbations to enact multi-target maneuvers. Figure 2b illustrates a multi-target perturbation forcing the vehicle to hit waypoints (blue circles) for a tighter turn than the original trajectory.

The contributions of the paper are:

- An adversarial testing framework for autonomous vehicles, DeepManeuver, that generates perturbations while accounting for vehicle state changes and capabilities.
- A study showing that DeepManeuver can generate perturbations that are more effective than state-of-the-art techniques by 20.7 percentage points. It also demonstrates DeepManeuver's effectiveness at achieving multi-target maneuvers with at minimum 52% success.
- An open-source experiment and tool repository that includes a modularized implementation of the approach, pretrained model, and extended results, available at https://figshare.com/s/36c08b9f7b6a0e0a6ae3.

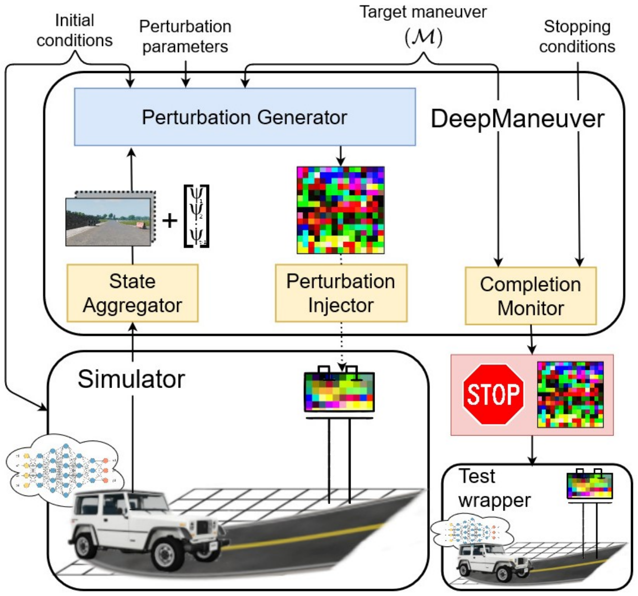
II. Approach

The goal of DeepManeuver is to generate realistic environment perturbations via a patch attack strategy that forces an autonomous vehicle relying on a DNN to follow a trajectory that fulfills an intended adversarial objective.

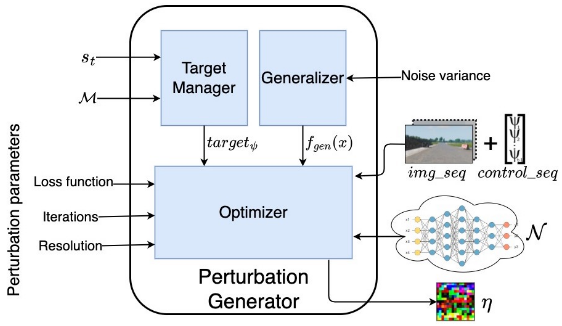
A. *Problem Definition*

Given 1) a vehicle $\mathcal{V}$ with actuation controlled by a DNN $\mathcal{N}$ that consumes a sensed image img_t and predicts an actuation command $\hat{\psi}_t$ such as steering at timestep t, 2) a physical environment $\mathcal{E}$ in which $\mathcal{V}$ operates and which contains an attack surface $\mathcal{O}$ visible to the vehicle's perception subsystem; and 3) a target maneuver $\mathcal{M}$ defined by a sequence of states $\vec{m}_{0,n} \in \mathcal{S}$, where $\mathcal{S}$ is the set of states inhabitable by $\mathcal{V}$ from timesteps 0 to n. $\vec{m}_{0,n}$ characterizes a system-level maneuver that is defined through spatio-temporal relationships between vehicle and environment, such as turning, running off the road, or crashing into an obstacle. Over the timesteps 0 to n, $\mathcal{V}$ will inhabit a sequence of states $\vec{s}_{0,n}$ under the control of $\mathcal{N}$ as it navigates $\mathcal{E}$ within view of $\mathcal{O}$. In the context of $\mathcal{V}$, a state s_t at timestep t includes the vehicle's pose, the image it perceives through its camera img_t, and its steering angle ψ_t. $\mathcal{N}$ output $\hat{\psi}_t$ is attenuated by s_t to produce ψ_t, the control signal that the vehicle can actuate given its current state and the time to actuate such a change.

With these givens, the problem is to generate a perturbation η that when applied to $\mathcal{O}$ induces $\mathcal{V}$ to satisfy $\mathcal{M}$, that is, whether $\vec{s}_{0,n} \models \vec{m}_{0,n}$. In this work, we argue and empirically



(a) Conceptual Overview of DeepManeuver.



(b) DeepManeuver's Perturbation Generator (expanded view of blue box in Figure 3a).

Fig. 3: Overview of DeepManeuver approach.

Algorithm 1: DeepManeuver

```
1  Given $\mathcal{V}$, $\mathcal{N}$ , $\mathcal{E}$, $\mathcal{M}$, init_condit, stop_condit, pert_params
2      simulator = initialize($\mathcal{V}$, $\mathcal{E}$, init_condit);
3      pg = PerturbationGenerator(pert_params);
4      sa = StateAggregator(simulator);
5      inj = PerturbationInjector(simulator);
6      cm = CompletionMonitor(simulator, stop_condit);
7      image_seq, control_seq = [], [] ;
8      $s_t$ = sa.capture(simulator.$\mathcal{V}$.state);
9      img_seq = sa.add($s_t$.img);
10     while ¬ cm.stop_condit do
11         $target_{\psi,t}$ = pg.determine_target($\mathcal{M}$, $s_t$);
12         $\eta$ = pg.optimize_$\eta$($\mathcal{N}$ , img_seq, control_seq, $target_{\psi,t}$);
13         $\mathcal{O}_{\eta,t}$ = inj.inject($\mathcal{E}$, $\eta$, pert_params.$\mathcal{O}$);
14         simulator.step();
15         $s_{t+1}$ = sa.capture(simulator.$\mathcal{V}$.state);
16         img_seq = sa.add($s_{t+1}$.img);
17         control_seq = sa.add($s_{t+1}.\psi$);
18         $s_t = s_{t+1}$
19     end
20     return $\eta$;
```

show that to satisfy $\mathcal{M}$, the generation of η must: 1) account for compounding effects of η on each state in $\vec{s}_{0,n}$, as η may force a new trajectory with a distinct sequence of states $\vec{s'}_{0,n}$, and 2) ensure past states $\vec{s}_{0,t-1}$ are preserved under the effect of η to maintain the validity of the current state s_t, thus ensuring η has a similar effect under test.

B. Overview

Figure 3a provides a conceptual overview of DeepManeuver applied to the motivating example in Section I. The main components DeepManeuver and the Simulator iteratively update a perturbation alongside the vehicle system state, enabling a state-adaptive approach. DeepManeuver takes four sets of inputs: the initial conditions for the simulation and the vehicle, the perturbation parameters such as the location and size of the perturbation surface, the target maneuver, and the stopping conditions for the perturbation generation loop. Once the stopping conditions are met, the perturbation is passed to the test wrapper component to assess its effectiveness.

Algorithm 1 presents the approach key steps. An initialization procedure starts the simulator with $\mathcal{V}$ in $\mathcal{E}$, connects DeepManeuver to the simulator, and initializes each of its components in lines 2-9. In line 11, the Perturbation Generator takes in $\mathcal{M}$ and s_t to determine the next $target_{\psi,t}$ (target steering at time t) for $\mathcal{V}$ to attempt to actuate. Note that $\mathcal{M}$ may be specified in terms of one or more state variables but $target_{\psi,t}$ can only be specified in terms of the actuation controlled by $\mathcal{N}$. In line 12, the Perturbation Generator uses the image and control sequences provided by the State Aggregator (see line 15, next paragraph), $\mathcal{N}$, and $target_{\psi,t}$ to optimize the perturbation η over the updated state sequence. Line 12 relies on $pert\_params$ to parameterize the objective functions used for optimization in the Perturbation Generator. The Perturbation Generator jointly optimizes two objective functions: one for maximizing the likelihood of satisfying $\mathcal{M}$ in the next step, and one for minimizing the effect of the optimization of η on the already-traversed trajectory (Section II-C).

Line 13 injects the updated η into the environment through $\mathcal{O}$. In line 14 the simulator steps forward by one timestep; $\mathcal{V}$ perceives the updated $\mathcal{O}_{\eta,t}$ and advances in its trajectory, with η affecting the new s_{t+1}. The State Aggregator then captures s_{t+1} (line 15) and adds the image and control output into the sequences utilized by the optimization process (lines 16-17) to preserve previous effects of η in future optimizations. At each loop iteration, the Completion Monitor checks whether stopping conditions are met (line 10), such as driving off the road, crashing, or passing $\mathcal{O}_{\eta,t}$ so it is no longer in view. Once a stopping condition has been met, the final perturbation has been generated and is ready for deployment to the test wrapper.

We highlight here how DeepManeuver differs from existing adversarial test generation approaches. Typically, existing techniques [5], [11], [13] consist of two phases: a collection phase to gather a set of inputs to the DNN, and a generation phase to perturb these inputs. DeepManeuver instead interleaves collection and generation phases to ensure that effects of the perturbation are reflected in future states and perturbation updates are tailored to trajectory changes. Also worth highlighting is DeepManeuver's ability to generate perturbations for multiple target states, where the target state is updated with vehicle state and the fulfillment of previous target states. As a vehicle progresses along a trajectory, DeepManeuver will update $target_{\psi,t}$ state-adaptively based on s_t and $\vec{m}_{0,n}$ to redirect the perturbation optimization as $\mathcal{V}$ satisfies $\mathcal{M}$.

C. Perturbation Generation

The Perturbation Generation component performs the critical operations of DeepManeuver to optimize the perturbation at each timestep. Figure 3b depicts it in detail. This component consumes img_seq and $control\_seq$ gathered from the simulator to generate a perturbation that maximizes the likelihood that the vehicle states satisfy a target maneuver $\mathcal{M}$. The workflow of this component is as follows.

The Target Manager produces target steering angle $target_{\psi,t}$ by assessing s_t against the specification of $\mathcal{M}$. Recall from Section II-A that $\mathcal{M}$ is specified through a sequence of target states $\vec{m}_{0,n}$ that can be compared to the vehicle state for similarity, and which characterize system-level behaviors. If $\mathcal{M}$ is single-target, $target_{\psi,t}$ remains constant for all $s_t \in \vec{s}_{0,n}$. For example, the maneuver "drive off the left side of the road" can set $target_{\psi,t} = 1$ (constant maximum left steering) at every step. Whereas, a maneuver like "gradually drift into the left lane" might set $target_{\psi,t} = 0.25$ at every step, until the entirety of the vehicle is within the left lane.

If $\mathcal{M}$ is multi-target (e.g., swerving back and forth at specified waypoints), $target_{\psi,t}$ is recalculated at every step based on s_t in relation to the environment, as defined by the maneuver. When a target state $m \in \mathcal{M}$ is reached, the Target Manager moves to the next target state, or it indicates that the perturbation has reached the final target state and fulfilled $\mathcal{M}$. A maneuver like "swerve back and forth" requires the vehicle to swerve left for some distance. Once that distance is travelled, the $target_{\psi,t}$ switches to swerve back in the other direction. For the maneuver "hit the billboard" at location (x,y), the $target_{\psi,t}$ is calculated according to the turning radius of the vehicle at its current speed and the relative angle between the current heading of the vehicle and the billboard. At each execution of line 11, the Target Manager uses the current m to determine the next control signal for the vehicle to attempt to actuate, $target_{\psi,t}$.

Once an image sequence is received, the Generalizer uses the noise variance parameter to create a function to produce variants of that sequence, increasing the adversarial strength [19] of the perturbation.[1] This component applies random noise to the collected images, a well-known data augmentation and regularization technique [20] to prevent models from overfitting to a small number of pixels in favor of the more general features of the image. The Optimizer then consumes the $\mathcal{N}$, img_seq and $control\_seq$, and $target_{\psi,t}$, and generates a perturbation using projected gradient descent [21], accounting for the following two properties.

First, **the perturbation must maximize the vehicle state change toward the current** m**.** This is accomplished by minimizing the error between a control signal ($target_{\psi,t}$) determined by the Target Manager and $\mathcal{N}$ output ($\hat{\psi}$). Unlike existing work, the effect of the perturbation is not measured in terms of $\mathcal{N}$ output ($\hat{\psi}$), but rather in terms of the vehicle actuation (ψ). This improves the likelihood for the perturbation to affect the system because optimization for previous steps is performed on actuation values that have been attenuated by the system according to s_t (ψ) rather than the control input ($\hat{\psi}$), which does not necessarily represent vehicle actuation.

Second, **perturbation effects over** $s_{0,t}$ **must be consistent over time and space**. The current state of the vehicle s_t depends on the effects of early perturbations. Subsequent optimization must result in a perturbation that consistently causes the sequence of previous states, thus preserving the validity of the trajectory up to s_t. Moreover, we do not want to sacrifice the effect of perturbations achieved in previous steps for the sake of maximizing the current perturbation.

These two properties are enforced jointly at each iteration of DeepManeuver by finding an η minimizing the loss functions $\mathcal{L}_1$ and $\mathcal{L}_2$ derived from the two error terms in:

$$\underset{\eta}{argmin}(\mathcal{L}_1(\mathcal{N}(img_n+\eta), target_{\psi,n}) + \sum_{t=0}^{n-1} \mathcal{L}_2(\mathcal{N}(img_t+\eta), \psi_t)) \tag{1}$$

where t is a timestep, $(img_t + \eta)$ is the image perceived by $\mathcal{V}$ at t that includes the perturbation η, and ψ is the actuation taken by $\mathcal{V}$ conditioned on its state. The first term of this objective function corresponds to the first property seeking to cause $\mathcal{N}$ to maximize the latest vehicle change at timestep N towards target state m, and the second term enforces that the perturbation maintains the attributes that caused the previous states of the sequence. This joint optimization allows for perturbations to induce single- or multi-target maneuvers and uses gradient descent to balance the actuation error across the sequence of images. The perturbation is used to adjust the input according to the gradient during gradient descent. The objective function's parameterization is further detailed in Section II-F4.

D. Injection

Line 13 in Algorithm 1 specifies that the perturbation is injected onto the attack surface. There are at least two generic approaches to injecting the perturbation: 1) superimposing a warped perturbation onto the sensed image, or 2) creating a new "skin" for the attack surface object and re-rendering the object in simulation for the vehicle perception system to capture. These two options offer different tradeoffs in terms of quality and overhead. Superimposition only requires changing captured images and has no bearing on the simulated representations of environment objects, but its approximations may not have the quality of high-end simulations. This simpler approach may be sufficient to render a perturbation on a billboard, but it will struggle with more complex multiplanar attack surfaces such as the surface of a car or other environment features like lighting and weather conditions. Conversely, changing the appearance of an object in simulation maintains the rendering quality, but requires re-rendering the current and previous images during optimization to change the appearance of the attack surface, which can require significant overhead. Given the complexity of the target surfaces, our study implements perturbation injection using superimposition.

[1]Note that strength refers to the ability of the perturbation to fool DNNs. This stands in contrast to robustness, which refers to a DNN's ability to handle small changes to inputs.

E. Simulator-in-the-Loop

DeepManeuver relies on a simulator that models $\mathcal{V}$ in $\mathcal{E}$ to incorporate η in the simulated $\mathcal{E}$ in a targeted and realistic manner and to observe the effects of η on the vehicle state s_t. In turn, acting on those observations is what makes DeepManeuver state-adaptive. The simulator's modeling of the vehicle behavior also means that DeepManeuver can optimize the η based on their effects on the vehicle. This is important because $\mathcal{N}$ outputs may be dampened by the state-dependent capabilities of $\mathcal{V}$ at the time the control signal is given. For example, a steering angle can only change so much in a given number of timesteps, and a value greater than the max steering angle will be truncated to the max value. So, even a perturbation that is dramatically effective on $\mathcal{N}$ may only have a small effect on $\mathcal{V}$. More formally, at any timestep t:

$$\psi_t = \mathcal{V}.actuate(\hat{\psi}_t \mid s_t), \text{ where } \hat{\psi}_t = \mathcal{N}(img_t + \eta) \quad (2)$$

DeepManeuver requires that a simulator meets three basic requirements. First, it must step forward incrementally such that DeepManeuver can run between steps to optimize η for new s_t. Second, the attack surface $\mathcal{O}$ must be modifiable such that η can be injected. Third, s_t must be queryable and contain features relevant to $\vec{m}_{0,n}$ such as the vehicle's pose, onboard camera sensor images, and the steering angle. Our current implementation uses BeamNG [22], an open-source high-fidelity driving simulator that meets these requirements. Figure 1 shows an instance of an autonomous vehicle in BeamNG. We interface between DeepManeuver and the simulator using BeamNGpy [23], a Python API for BeamNG.

F. Parameterization

Our technique is parameterized across a range of variables that contribute to high configurability, briefly discussed here.

1) Noise variance: A scalar parameter of a Gaussian distribution to control variance around a zero-centered mean. At each iteration of gradient descent, noise is independently sampled and applied to each image in the sequence.

2) Iterations of gradient descent: The number of steps taken to explore the input gradient when generating a perturbation. Rather than executing until convergence, bounding the iterations can mitigate the cost to run and facilitate comparison across techniques.

3) Sequence Collection: The **cut-on** is the vehicle state at which to begin generating the perturbation. Cut-on is specified through a state predicate such as distance from the vehicle to the attack surface, or by a set of state variables such as wheel speed or whether the vehicle is centered on the road. Similarly, the **cut-off** is a state predicate that determines the end of the perturbation generation run. For example, when the vehicle has deviated from the original trajectory by a specified value may trigger the cut-off. In our study, we compute the vehicle's reachable set [24] to determine when the vehicle is reaching unrecoverable conditions. Sequence collection occurs in two stages: driving by the attack surface normally, producing the original trajectory, and driving by while undergoing perturbation generation, producing the state-adaptive trajectory.

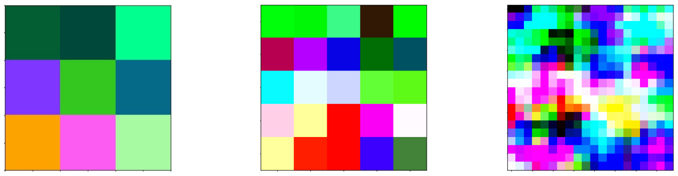

(a) 3×3 (b) 5×5 (c) 20×20

Fig. 4: Billboard perturbations across levels of resolution.

Perturbation generation can be parameterized to include the original trajectory as part of the image and control sequences.

4) Maneuvers, target states, and objective functions: The Target Manager retrieves the current target state m from $\mathcal{M}$ based on the vehicle state s_t and uses it to calculate $target_{\psi,t}$. For a single-target maneuver, $target_{\psi,t}$ remains constant across a trajectory irrespective of t. Our first study illustrates a single-target maneuver by aiming to steer as hard as possible in one direction ($target_{\psi,t} = \mathcal{V}.maxLeft|\mathcal{V}.maxRight$) to run the car off the road. For a multi-target maneuver, $target_{\psi,t}$ may vary at each step depending on the current vehicle state in relation to m. For example, if m is defined by a particular road location but the vehicle has drifted to the left of the target state at timestep t, then $target_{\psi} = \mathcal{V}.right_{\psi}$ at $t+1$, towards the direction of the target location specified by m. Section III-D in the study provides further examples of multi-target maneuvers.

DeepManeuver supports some common objective functions such as mean squared error (MSE) and mean absolute error. The choice of objective function depends on several contextual factors like variability in the image and control sequences or the precision of the specification of $\mathcal{M}$. For single-target maneuvers, one might tailor the objective function to be directly proportional to the steering angle, so that its minimization steers as far as possible in the desired direction at each timestep. For a more complex maneuver, however, where small differences in the preceding sequence may have a large effect on later behavior, an objective function such as MSE may be preferable to penalize the largest deviations from the control sequence. These functions are minimized when two vectors are equal, so minimizing them will reduce the difference between $\mathcal{N}$'s predicted outputs and the previously seen actuations. Additionally, the objective function can encode heuristics, such as weighting error terms at different timesteps. Our motivating example may require states towards the end of $\mathcal{M}$ to have a lower loss, as $\mathcal{V}$ gets close to satisfying $\mathcal{M}$, so we can design the loss functions with a decay factor for each term.

5) Perturbation surface attributes: These parameters account for three aspects of the perturbation surface: position and angle with respect to the vehicle's initial location, size, and perturbation resolution. Perturbation resolution, as illustrated in Figure 4, determines the number of grid spaces in the perturbation that can be manipulated by joint optimization. These attributes can affect the perturbation strength.

G. Test Wrapper

Separate from DeepManeuver, our infrastructure includes a test wrapper to inject the perturbation into the environment, execute a suite of test runs, and assess the overall effectiveness of the perturbation in forcing vehicle maneuvers that satisfy

$\mathcal{M}$. During this test phase, the wrapper allows the execution of many simulation runs initialized with $\mathcal{V}$, $\mathcal{E}$, and $\mathcal{O}_\eta$, and the collection of detailed state traces from these test runs to compare against $\mathcal{M}$. We track the effectiveness of the perturbation over multiple test runs to form an understanding of performance in the average case, and to deflake a non-deterministic system subject to environmental noise from the simulator. Metrics computed include mean angle error, average distance from original trajectory, as well as number of crashes, deviations from the road surface, and ms fulfilled.

H. Applicability and Generality

DeepManeuver is intended to validate the robustness of the system as a whole while considering the system state. Since it requires a vehicle $\mathcal{V}$ controlled by a DNN $\mathcal{N}$ operating in a simulated environment $\mathcal{E}$, it is likely to be applied in the later phases of the validation cycle. However, the application of DeepManeuver is not constrained to a specific type of vehicle, DNN, simulator, environment, or attack surface. For instance, it could be applied to drones, boats, and industrial systems such as grippers and assembly-line robotic arms, each operating in environments with distinct attack surfaces.

III. Study

Our study aims to answer the following research questions:
RQ1) *How effective is DeepManeuver at generating perturbations that cause an autonomous vehicle to leave the road?* To answer this we compare DeepManeuver to two versions of a state-of-the-art technique on 6 scenarios across 3 road topologies, and we explore the effect of some key parameters on the effectiveness of our approach.
RQ2) *How effective is DeepManeuver at generating perturbations that cause an autonomous vehicle to fulfill maneuvers involving multiple target states?* To answer this question, we explore the ability of DeepManeuver to achieve three multi-target maneuvers: hit a target, change lanes, and cut a corner.

A. Setup

1) System Under Test: Our study uses the BeamNG driving simulator [22] for its pedigree as a customizable high-fidelity beam-node simulator and its use in research on realistic simulated driving [15], [25], [26], [27]. For the autonomous vehicle under test, we equip BeamNG's prepackaged "hopper" vehicle (Figure 1) with an onboard camera matching established self-driving setups [28], [10]. Images are collected from the simulator at 15 Hz as 240 $\times$ 135 pixels (ratio 16:9) to match related work. Steering inputs are also collected. The steering input range of $[-1, 1]$ sets wheel rotation to the left or right, respectively, which determines the turning radius of the car. With a steering input of ± 1 at 40kph, the hopper can change its yaw relative to the world frame by $29.96°$ per second. For the steering model, we re-implemented the canonical Nvidia DNN architecture, DAVE2 [29]. We trained with data augmentations on 82,193 images collected over multiple prebuilt BeamNG driving environments with a final training loss of 0.0022 MSE. The system can follow the centerline of the road indefinitely in the racetrack environment used in this study.

2) Environments: We manipulate several variables associated with the environment. For **RQ1**, we perform a full factorial study on the three techniques across 6 scenarios, 2 maneuvers, and 3 billboard resolutions. Each of the 6 scenarios is set up on one of 3 straight and curved road topologies in the "industrial" prepackaged environment within BeamNG. A billboard is inserted near the track to serve as the attack surface. We fixed the physical size of the billboard to occupy at least 1% or about 400 pixels of the image when the car is approximately 28-30m away from the board. Note that aspects of our study mirror the setup of the DeepBillboard [11] study to make a fair comparison (see Section III-B). DeepBillboard indirectly specified the billboard resolution according to pixel overlap and the shape of each billboard; in this work, we explore the performance of each technique on billboards with three different resolutions: 5×5, 10×10, 15×15. For **RQ2**, we evaluate DeepManeuver on 3 road topologies with 3 maneuvers: hit a target, change lanes, and cut a corner.

We spawn the vehicle in the center of the track slightly outside of the range in which it can perceive the billboard to allow it to approach $\mathcal{O}$ under $\mathcal{N}$ control. Note that due to the built-in nondeterminism and noise in the BeamNG simulator, test runs are not identical and can demonstrate how well the perturbation would work under real-world circumstances and generalize to similar trajectories. During the testing phase, we track the variation of the effectiveness of the perturbation over multiple test runs (specified under each question).

B. Design

We apply DeepManeuver and two baseline techniques to various scenarios in an attempt to generate perturbations that cause the system under test to satisfy a specified maneuver. We assess the performance of each technique by measuring its success in fulfilling the maneuver. In the rest of this section, we discuss the techniques and parameter configurations considered, how they are assessed, and the scenarios explored.

1) Techniques: The techniques to generate the perturbations constitute the main treatments we evaluate.

For **RQ1**, those techniques are the proposed DeepManeuver, the baseline technique DeepBillboard [11], and DBB+ which incorporates enhancements from DeepManeuver into DeepBillboard, notably the inclusion of noise variance and supplementing the original trajectory with a second collection of the unperturbed action and image sequence. DeepBillboard inspired this work and serves as a baseline for environmentally-situated adversarial perturbations that influence the actuation of autonomous vehicles. We have undertaken several measures to make a fair comparison to DeepBillboard. First, due to limitations and inconsistencies of the publicly available DeepBillboard implementation, we have made a best effort re-implementation of the DeepBillboard technique. Second, we configure the three techniques with the same parameters, from the loss function to the initial color of the billboard (all grey). Third, for independent variables not explored in the DeepBillboard evaluation, we either justify their value or explore several values to understand their effect.

For **RQ2**, we have one treatment, DeepManeuver, since there are no available techniques that can perform multi-target perturbations. Because the value of $target_{\psi,t}$ is no longer constant and depends on the current state, we disable inclusion of the unperturbed original trajectory for this treatment.

2) Technique Parameters: The techniques take on multiple parameters, discussed here and under each RQ in more detail.

Noise variance (`noise_var`): Following existing practices [30], we explore six noise variances $\{0, \frac{1}{25}, \frac{1}{20}, \frac{1}{15}, \frac{1}{10}, \frac{1}{5}\}$, where 0 indicates no noise. Max variance was selected such that the mean pixel value begins to shift due to clipping. Intermediate values are based on an inverse log scale.

Gradient Descent Iterations (`iters`): set to 400 as it was shown to be sufficient in related work [11].

Cut-on: determines when collection begins. We explore distances of 20, 24, and 28 meters from the billboard as cut-on points. Closer than 20m, the billboard may not be visible to a vehicle in the center of the road, and beyond 28m the billboard corners are not distinguishable in the image.

Cut-off: We used the vehicle's reachable set as an indicator that the deviation is large enough to stop collection. At least 60% of the area reachable by the vehicle is within the track when driving under normal conditions. Collection stops when less than 60% of the reachable set remained on the track. The reachable set is overestimated by bounding the area reachable by max steering angles and current speed in 1 second.

Objective Functions: The loss for single- and multi-target maneuvers must be constructed such that it satisfies Equation 1. For the single-target perturbations under **RQ1**, $target_{\psi,t}$ was set to -1 or 1 when perturbing towards the left or right respectively, and the objective function was defined as $argmin_{\eta}(-target_{\psi,t} * mean(x-y))$. We take the mean of the loss at all steps and direct it according to the opposite sign of $target_{\psi,t}$ so as to not prioritize minimizing the loss for any given step, as MSE loss does for larger terms in the loss function. For multi-target maneuvers in **RQ2**, $target_{\psi,t}$ was set to a steering command to meet the desired pose state and the objective function was defined as $argmin_{\eta}(MSE(x,y))$. In both cases, x represents the output of $\mathcal{N}$ on an image sequence with the current η injected (including the image collected at the current step), and y represents all steering angles actuated in previous steps, plus the $target_{\psi,t}$ in the current step. For reference, Equation 1 represents previous steps with $\mathcal{L}_2$ and current step with $\mathcal{L}_1$.

3) Dependent Variables: Dependent variables are associated with the vehicle behavior under test. Our primary consideration is the effect on the overall system behavior, so we count whether the target maneuvers are met, and we measure the average distance from the original trajectory (ADOT) in meters throughout the run once the perturbation began to take effect. We also report the average angle error (AAE) of each image in the test run, which represents the average per-image effect on the DNN behavior calculated as $\sum_{t=0}^{N} \frac{\psi_{orig} - \psi_{pert}}{N}$. A negative AAE value indicates steering angles were shifted to the right on average; positive indicates a shift to the left.

Reproducibility. The trained model and the framework are available at https://figshare.com/s/36c08b9f7b6a0e0a6ae3.

C. RQ1: DeepManeuver Single-Target Maneuvers

We assess the effectiveness of DeepBillboard, DBB+, and DeepManeuver in forcing a single-target maneuver of max steering in one direction at each timestep (as done by DeepBillboard [11]), with the goal of crashing or leaving the road. In the first part of this section, we study the performance across a set of road topologies, approximating the setup of related work. In the second part, we manipulate key parameters to understand their effect.

1) Baseline Parameters on Varying Road Topologies: First, we approximated the context explored by evaluations in related work. We revisit four aspects mentioned in Section III-B1. First, the cut-on: DeepBillboard's physical study began the data collection sequence when the billboard takes up at least 400 pixels in the image, which here begins at approximately 28m. Second, we used a grey starting billboard with RGB value (128, 128, 128) for all techniques rather than the gold-yellow billboard used by DeepBillboard, as initial maximum values will hamper optimization. Third, we explicitly explore the effects of billboard resolution (i.e., 5×5, 10×10, 15×15) rather than basing the resolution on the size of the billboard in the image, and we use perspective warping to place the perturbation into the image. Fourth, we set up 3 distinct road topologies (see Figure 5) to test on a straightaway and left- and right-hand turns instead of the single straightaway road DeepBillboard used in its physical test. To account for randomness in perturbation generation and simulation, we generate 50 perturbations per technique and parameter combination and perform 10 test runs on each perturbed billboard.

Table I summarizes the performance of the three techniques under the original parameters of the related work through a full factorial study of the combination of three road topologies, two maneuvers, and three resolution levels. Topology-maneuver pairs are presented as "scenarios" in the leftmost column. Odd-numbered scenarios are left-turn maneuvers and even-numbered scenarios are right-turn maneuvers. Odd-even pairs of scenarios are performed on the same road topology.

Scenario 1 (Left turn on straight road). This Scenario shows the effectiveness of DeepBillboard, DBB+, and DeepManeuver to force a car to turn left on a straight road. DeepBillboard perturbations had little success, producing only 3 crashes across all 1500 test runs for this topology (50 billboards, 3 resolution levels, 10 test runs each). Figure 5a shows test trajectories concentrated around the centerline of the road with little deviation. DBB+ shows improvement over DeepBillboard in success rate and ADOT. Because this is a leftward perturbation, AAE should trend positive as explained in Section III.B.3. At all resolutions, DBB+ has success rates in the single digits with a decrease across success rate and ADOT as resolution increases, and minimal change to AAE. For DeepManeuver, we see double-digit success rates between 21.0% and 30.4%. Still, as shown in Figure 5b for 10×10 resolution, the perturbation has a strong effect on successful trajectories. Because DeepManeuver uses weighting in the objective function, this could mean that it does not enact high errors in steering angle later in the sequence, preventing large changes in trajectory past certain states. We again see a drop in

		DeepBillboard			DBB+			DeepManeuver		
		`noise_var=0, cut-on=28m`			`noise_var=`$\frac{1}{15}$`, cut-on=28m`			`noise_var=`$\frac{1}{15}$`, cut-on=28m`		
	`Resol.`	**Success rate**	**ADOT**	**AAE**	**Success rate**	**ADOT**	**AAE**	**Success rate**	**ADOT**	**AAE**
Scenario 1:	**5×5**	0.6%	2.18	**0.000**	8.8%	2.42	-0.015	**26.4%**	* **3.06**	-0.056
left turn on	**10×10**	0.0%	1.99	**0.013**	5.8%	2.21	0.012	* **30.4%**	* **2.62**	-0.017
straight road	**15×15**	0.0%	1.96	**0.022**	0.2%	2.01	0.011	* **21.0%**	* **2.71**	-0.033
Scenario 2:	**5×5**	99.3%	2.35	-0.198	**99.4%**	2.31	-0.201	99.2%	* **2.50**	* **-0.204**
right turn on	**10×10**	99.4%	**2.35**	-0.180	99.6%	2.34	-0.201	**99.8%**	2.34	* **-0.215**
straight road	**15×15**	98.7%	2.33	-0.176	**99.0%**	2.33	-0.181	88.6%	* **2.35**	* **-0.186**
Scenario 3:	**5×5**	0.2%	1.09	0.012	6.4%	1.27	* **0.034**	* **54.6%**	* **1.86**	0.060
left turn on	**10×10**	8.8%	1.11	0.022	* **12.9%**	1.32	* **0.030**	10.8%	**1.35**	0.021
right-hand curve	**15×15**	**17.6%**	1.27	**0.032**	16.6%	1.31	0.030	3.4%	* **1.43**	0.015
Scenario 4:	**5×5**	47.2%	2.55	-0.367	99.6%	3.09	-0.766	* **100.0%**	* **3.17**	* **-0.819**
right turn on	**10×10**	15.0%	2.48	-0.283	79.0%	2.69	-0.599	* **94.2%**	**2.87**	* **-0.678**
right-hand curve	**15×15**	6.3%	2.34	-0.258	65.0%	2.50	* **-0.533**	* **81.6%**	**2.60**	-0.515
Scenario 5:	**5×5**	1.4%	0.93	0.011	0.6%	0.90	0.009	* **9.2%**	* **1.65**	* **0.054**
left turn on	**10×10**	0.0%	1.22	0.025	0.0%	1.08	0.017	**1.4%**	* **1.54**	* **0.047**
left-hand curve	**15×15**	0.0%	1.27	0.026	0.0%	1.21	0.021	0.0%	* **1.38**	* **0.033**
Scenario 6:	**5×5**	67.6%	1.95	-0.111	**88.2%**	**2.19**	**-0.152**	85.6%	2.17	-0.151
right turn on	**10×10**	8.2%	1.39	0.041	8.0%	1.40	0.041	* **31.2%**	* **1.64**	* **-0.015**
left-hand curve	**15×15**	0.0%	1.33	0.056	0.2%	1.34	0.056	* **6.0%**	* **1.40**	**0.047**

TABLE I: All techniques on 6 scenarios (topology-maneuver pairs) at 3 resolutions. Metrics are the success rate, AAE under test, and ADOT. Best-performing values per row are in bold. Statistically significant best-performing values are starred.

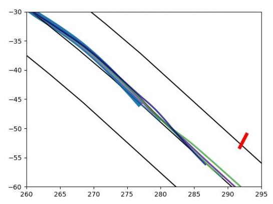
success rate and AAE for resolution=15×15 meaning that the technique performed well on the collection sequence, but not on test runs, an indication of overfitting at higher resolutions. It is also worth noting that the 69.6% of unsuccessful trajectories show a slight deviation to the right in Figure 5b, which would explain the slightly negative AAE for DeepManeuver. Overall, DeepManeuver performs best by success rate and ADOT over all resolutions, with the DBB+ success rate less than one-third that of DeepManeuver. DeepBillboard had little effect on success rate or ADOT.

Scenario 2 (Right turn on straight road) keeps the same, straight road topology and reverses the direction of the maneuver. This change significantly increases the effectiveness of all three techniques. All success rates show a 88.6% success rate or above and all but one success rate is 98.7% or above, with an average of 98.1% success rate over all techniques and all parameter combinations. Although no success rates are statistically significantly different, all but one ADOT and AAE measure are statistically significant better and belong to DeepManeuver. This large disparity between right- and left-turn maneuvers on a straight road appears due to overfitting, as there are a greater number of right- than left-turn datapoints in the training set for our DNN. This makes right-hand turn maneuvers easier because overfitting will create high variance for that subset of datapoints, allowing for more opportunities to shift the output of the DNN with small changes to the image.

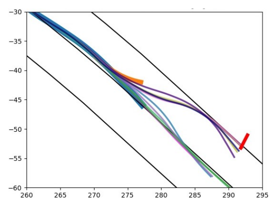
Scenario 3 (Left turn on right-hand curve). The middle third of Table I shows the effectiveness of the three techniques on a right-hand curve road topology. Scenario 3 forces the vehicle to turn left. DeepBillboard and DBB+ show similar low success rates between 0.2% and 17.7%. By contrast, DeepManeuver achieves success rates of 3.4% to 54.6%. Both baseline techniques show monotonically increasing success rates with increasing billboard resolution, whereas DeepManeuver shows decreasing effectiveness with increasing billboard resolution. Although DeepManeuver AAE is low compared to the baseline techniques, DeepManeuver ADOT is higher across resolutions. ADOT remaining high indicates that, even for test trajectories that did not produce a crash, DeepManeuver perturbations were able to produce more consistent deviation from the original trajectory than either of the two baseline techniques.

Scenario 4 (right turn on right-hand curve) retains the right-hand curve topology and reverses the direction of the maneuver, forcing the car to make an early hard right turn on a right-hand curve. This significantly increases the effectiveness of all three techniques. DeepManeuver again outperforms both techniques with a success rate of 100% at resolution=5×5. DeepBillboard is more successful on Scenario 4 than Scenarios 1 and 3 for all billboard resolutions, but still far weaker than DBB+ and DeepManeuver, both of which approach saturation for at least one resolution. Comparing Figures 5c and 5d, DeepBillboard trajectories show higher variation than DeepManeuver trajectories, which consistently veer off the right road edge. DBB+ shows high sensitivity to resolution, with a 14 percentage point (pp) or greater decrease in success rate for each increase in resolution, compared to a 12.6pp or smaller decrease for each resolution increase for DeepManeuver. The improved performance across techniques indicates either a weakness in the DNN in this area of the track, or that features of images collected from normal driving are similar enough to the early hard right to cause a failure without a state-adaptive technique. Additionally, the track used to train the DNN has five right-hand curves and one left-hand curve, which could make perturbing to the left more difficult given the training data bias. Overall, DeepManeuver outperforms both DeepBillboard and DBB+ in all values of resolution for this topology in both success rate and ADOT.

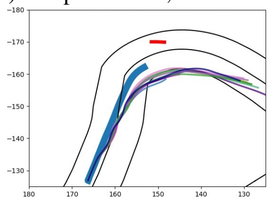
Scenario 5 (Left turn on left-hand curve). The bottom of Table I tests each technique on a left-handed curve, first at forcing a car to make a left-hand turn such that the car leaves the road. All techniques show a reduction in success rate for this topology, with values below 10% for DeepManeuver, and below 2% for the other techniques. ADOT provides hints about the weaker performances. 5×5 is the best resolution for all techniques in terms of success rate, with other resolutions having a success rate of zero or near



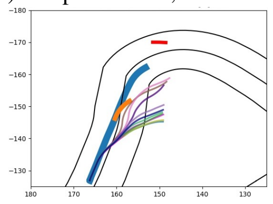
(a) DeepBillboard, Scenario 1



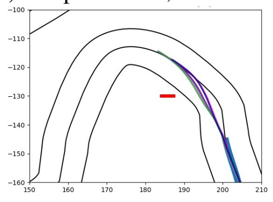
(b) DeepManeuver, Scenario 1



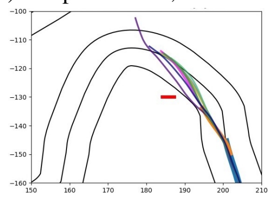
(c) DeepBillboard, Scenario 4



(d) DeepManeuver, Scenario 4



(e) DeepBillboard, Scenario 5



(f) DeepManeuver, Scenario 5

Fig. 5: Perturbations generated by DeepBillboard and DeepManeuver for 10×10 billboard. Billboard is shown in red, original trajectory in thick blue, DeepManeuver collection sequence in thick orange, and test trajectories are thin lines.

zero. As seen in Figures 5e-5f, the test trajectories' deviation from the center of the road increases as the vehicle approaches the billboard. The trajectories then either return to the center after passing the billboard or fail to recover and leave the track in Figure 5f. Although DeepManeuver performs best, this topology illustrates how challenging it can be to generate perturbations that generalize across states.

Scenario 6 (Right turn on left-hand curve). Conversely to Scenario 5, shows a decrease in success rate and ADOT with an increase in resolution for all three techniques. DeepManeuver still performs the best for all performance metrics, with statistical significance. All techniques see a monotonic decrease in performance across all metrics with increasing resolution. This is the least successful right turn for all topologies and all techniques. This contributes to our hypothesis that overfitting is the cause, as left turns are the least common road topology in the DNN's training dataset. This leaves less of an opportunity for the DNN to overfit to this topology, causing the increased robustness on this topology as compared to the straight or right-hand curved road topologies.

Overall, DeepManeuver provides a 20.7pp average improvement over DeepBillboard and a 8.6pp improvement over DBB+. DeepManeuver also holds 38 of the 54 best metrics in Table I, with 6 belonging to DeepBillboard, 9 belonging to DBB+, and the success rate for Scenario 5, resolution 15×15 having no best metric as all techniques have a rate of 0.0%.

In terms of statistical significance, the Kruskal-Wallis one-way test was used to determine whether the performance differences across techniques was statistically significantly. Each performance metric was compared for each resolution across all three techniques. Overall, Table I shows a concentration of starred metrics on the right side of the table under DeepManeuver, which has 31 statistically significant best-performance out of the 54 shown in Table I.

We note that DeepBillboard's performance is less impressive than originally reported. A potential reason is that DeepBillboard was assessed only on vehicle perception and DNN steering output. So, measures that require vehicle actuation such as success rate or ADOT were not provided. Instead, the assessment included a measure similar to AAE, which is reported as an average steering angle error of 8.8 degrees. However, we note again that since the vehicle did not act on the DNN's output, the vehicle was following a test trajectory very similar to the original trajectory on which the perturbation was generated. This is an ideal case for DeepBillboard, but one that is unlikely to occur in practice. For further discussion of factors contributing to diminished DeepBillboard performance, occurrences of overfitting, and how to interpret average angle error, see Section III-E and extended studies in the supplementary material (https://figshare.com/s/36c08b9f7b6a0e0a6ae3).

In terms of cost, DeepManeuver is significantly more expensive to run than either DeepBillboard or DBB+ by a factor of N, where N is the final length of the sequence. This is because perturbation optimization is run at each timestep in DeepManeuver, compared to only once for DeepBillboard and DBB+. For example, under Scenario 1 and the earliest cut-on=20m, DeepBillboard takes an average of 33 seconds to generate a perturbation, compared to 156 seconds for DeepManeuver. With the latest cut-on=28m, DeepBillboard takes 53 seconds to generate a perturbation, compared to 405 seconds for DeepManeuver. Detailed cost assessment and optimization of DeepManeuver is an avenue for future work.

Across all road topologies, DeepManeuver leads to statistically comparable or greater ADOT and frequency of vehicle crashes and road surface departures than baseline techniques for all but one combination of scenario, maneuver, and resolution.

2) Parameter Exploration and Key Tradeoffs: In our investigation, we found that parameterization can have a significant effect on performance. Hence, we performed a partial ablation study through a fractional factorial exploration of some key parameters. To supplement our study of resolution in Table I, we investigate value ranges for cut-on and noise variance in Tables II and III, keeping other parameters constant. We reuse Scenario 1 as the success rate does not saturate or desaturate for any technique. Best-performing values are highlighted in bold, and statistically significant best-performing values as determined by a Kruskal-Wallis one-way test are starred.

Table II shows the effect of cut-on values. All techniques are susceptible to cut-on differences. DeepManeuver is strengthened by earlier cut-ons, with a 20pp or greater increase in success rate for cut-ons of 24 or more. The benefits of earlier cut-ons is less noticeable DeepBillboard and DBB+, as all cut-ons show success rates close to zero. DeepManeuver appears to have a much higher success rate than the other techniques for all cut-ons, but notably the earlier cut-ons of 24m and 28m. As expected, cut-on limits the space that the perturbation is optimized for, so earlier cut-ons are advantageous.

cut-on	DeepBillboard noise_var=0		DBB+ noise_var=$\frac{1}{15}$		DeepManeuver noise_var=$\frac{1}{15}$	
	Success rate	ADOT	Success rate	ADOT	Success rate	ADOT
20m	0.0%	1.99	0.0%	1.95	*** 5.4%**	*** 2.17**
24m	1.0%	2.11	0.0%	2.04	*** 27.2%**	*** 2.42**
28m	0.0%	2.02	2.8%	2.22	*** 31.4%**	*** 2.63**

TABLE II: Preliminary exploration of the effects of cut-on on a resolution=10 × 10 billboard. Metrics are the success rate and ADOT under test.

noise_var	DBB+		DeepManeuver	
	Success rate	ADOT	Success rate	ADOT
0	0.0%	1.93	*** 3.4%**	*** 2.11**
$\frac{1}{25}$	0.0%	*** 2.07**	*** 1.4%**	2.05
$\frac{1}{20}$	0.0%	1.96	*** 25.8%**	*** 2.42**
$\frac{1}{15}$	0.0%	2.10	*** 31.2%**	*** 2.45**
$\frac{1}{10}$	0.8%	2.15	*** 45.4%**	*** 2.77**
$\frac{1}{5}$	1.0%	2.13	*** 9.4%**	*** 2.19**

TABLE III: Preliminary exploration of the effect of noise variance. Metrics are the success rate and ADOT under test.

Table III shows the effect of noise-variance with a constant median cut-on value of 24m. We compare only DBB+ and DeepManeuver as DeepBillboard does not incorporate noise. As a commonly applied regularization technique, we expected it to have a similar effect across techniques. DBB+ and DeepManeuver react positively to the introduction of noise, with $\frac{1}{10}$ variance showing the greatest performance boost. DeepManeuver also shows a significant jump for noise_var=$\frac{1}{10}$ compared to other values, indicating that this technique is sensitive to high and low noise variance. The skewed normal distribution of all metrics, centered around $\frac{1}{10}$, suggests an optimal noise variance for this technique. A more comprehensive parameter exploration to address issues of overfitting and other questions brought up by Table I is presented in the supplementary material (https://figshare.com/s/36c08b9f7b6a0e0a6ae3).

The baseline techniques did not improve within the explored parameter space. DeepManeuver showed improvement with low cut-on parameters, and performance varied widely within the range of noise.

D. RQ2: DeepManeuver Multi-Target

To evaluate the effectiveness of DeepManeuver at inducing multi-target maneuvers, we set up three maneuvers: crashing into the billboard, changing lanes, and cutting a corner on the inside of the track. These maneuvers are multi-target because they require the vehicle reach multiple objective states, and require updating the output constraint based on the state of the vehicle at each timestep.

For this study we keep the same configuration parameter values used in the first portion of the previous study as reported in Table I: resolution=10×10, noise_var=$\frac{1}{15}$, iters=400, cut-off=0.60, and cut-on=28m. However, we make two adjustments. First, to better optimize across multiple objectives, we change the loss function to MSE as explained in Section II-F5. Second, we determine that a maneuver is satisfied when every spatial objective m is reached within a threshold of 1 meter. The threshold of 1m is measured with respect to the center mass of the vehicle rather than its bounding box. In other words, as long as the center mass of the car passes within 1m of the point, then the objective is met. Because the hopper vehicle, like most vehicles, is over 2m wide and 2m long, some portion of the car chassis will pass through the point defining that part of the maneuver.

We generate 25 billboards per objective and perform 10 test runs on each billboard. The results and trajectories are summarized in Figure 6.

The “hit the billboard” maneuver in Figure 6a shows a near 100% success rate. Contrary to the single-target maneuvers in Section III-C, this maneuver does not attempt to have the vehicle turn right away. Instead, the vehicle follows the centerline for most of its trajectory, then turns directly into the billboard near the edge of the road. Since most of the trajectory thus adheres to normal driving, ADOT and AAE are low. AAE from the collection and test runs are comparable, suggesting a high similarity between test and collection runs. This similarity is a likely reason for the high success rate.

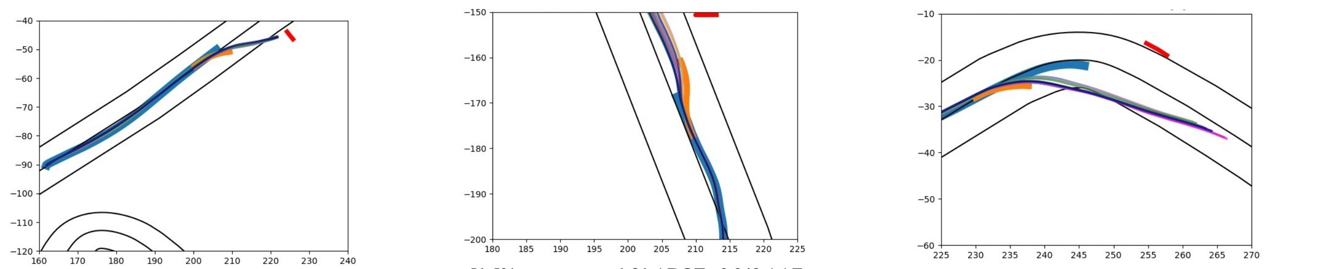
In Figure 6b, we see that most trajectories achieve a lane change. However, the success rate is little more than 50%. Further inspection reveals that this is due to our acceptance threshold of 1m. Relaxing the threshold to 2m improves the success rate to over 85%. Since this maneuver requires small but constant deviation from the original trajectory, the ADOT is 1.21m, and AAE metrics for collection and test runs are low. This multi-target maneuver shows that DeepManeuver can effectively enact subtle perturbations that require two clear objectives – turning right, and then turning parallel with the road – with a high rate of success.

In Figure 6c, the “cut the corner” maneuver turns close to the inside edge of the road, instead of following the curve like the original trajectory. The success rate for this multi-target maneuver is 68.4%, with a low trajectory variance across all successful test runs. However, the trajectories show some unintended effects, such as forcing the car to stay straight when entering the curve in 3 test runs. The cause of this unintended effect may lie in the variation in trajectories between test runs and is a candidate for future work. Additionally, ADOT results for this maneuver are higher than other multi-target maneuvers because the distance from the center of the road to the edge is farther away so it is possible to drive in a valid area farther from the original trajectory.

DeepManeuver can consistently produce perturbations that affect the vehicle differently based on state, with high accuracy in perturbing towards multiple target states.

E. Threats to Validity

Generalization of the findings is limited by the vehicle, DNN, and environments under which we performed the study. We systematically chose these aspects following the setup of previous work, defined in practice when available, and justified throughout the paper. Regarding the network model choice, although used extensively in previous work, the DAVE2 DNN architecture is simpler than today’s architectures [31] and we have trained it with just enough data to perform consistently on a single track. We deemed that sufficient to expose the differences between the techniques, and we expect these techniques’



98.6% success rate, 1.03 ADOT, -0.073 AAE

(a) Hit the billboard

51.5% success rate, 1.21 ADOT, -0.049 AAE

(b) Change lanes

68.4% success rate, 2.25 ADOT, -0.103 AAE

(c) Cut the corner

Fig. 6: Multi-target perturbations: hit billboard, move off center lane, take a sharper turn on a corner. Billboard is shown in red, original trajectory in thick blue, collection sequence in thick orange, and test trajectories are thin multicolored lines.

performance to be weaker with a higher-performing network. Our findings are also limited to the parameter space we explored, which captures key aspects affecting the techniques' performance but is nonetheless limited.

The threats to internal validity include the implementation quality of the DeepBillboard and DBB+ techniques and the effective manipulation of parameters and configurations. We have shared our implementation to facilitate checking and reproduction by others. At the time of publication, DeepBillboard did not have runnable code available; our reimplementation is based on their repo and paper descriptions. Regarding the study setup, we tried to replicate and then extend DeepBillboard's study in simulation to increase the exploration of what we consider fundamental variables to the effectiveness of the techniques such as the topologies, the billboard attributes, and when the billboard is visible. We also consider the non-determinism as part of the simulation to determine perturbation overfitting and address the issue of perspectival warp according to that nondeterminism through the billboard resolution parameter. Ultimately, our study setup compares identical scenarios and parameter combinations, and so we consider this a fair comparison of the two techniques.

Our measures of success, particularly ADOT and AAE, obscure some aspects of vehicle behavior and thus create threats to our findings' construct validity. ADOT does not account for variance, and so a large deviation that is then corrected for or a test trajectory that swerves back and forth might have a low ADOT when there was a significant failure. Similarly, AAE may seem low for a perturbation that generates a high success rate because small deviations add up over time, and a large error at a particular timestep, at the expense of optimization for other timesteps, may be the key to a high success rate. That said, taken together with the success rates, the metrics provide an overall characterization of each scenario. We also note that adjudications of various vehicle failures are based on the deviation threshold chosen to measure the difference between the original and perturbed trajectory, and different threshold selections can lead to different results.

IV. Related Work

We have organized the related work in two sections, with the first set focusing on DNN testing and the second set on testing whole autonomous systems. We note that due to the high cost associated with test scenario manipulation and reproducing system-level failures, most of these testing approaches are performed almost exclusively in simulation, which has been determined sufficient for testing these systems [15], [32], [14].

A. *Testing DNNs*

Many traditional software testing techniques and concepts have been adapted to DNNs to effectively guide the generation of test inputs [33], [34]. In particular we note approaches that use random fuzzing and genetic search [35], [36], [15], those that are coverage-guided [37], [7], [38], [39], [40], [41], and those that consider the input distributions [42], [43], [44], [45].

In parallel to more traditional techniques, the machine learning community pioneered specialized mechanisms to judge DNN robustness through the generation of adversarial inputs [46], [47]. Such input generation is a form of metamorphic testing [48] as realized by Zhou et al. [49], where perturbations to the DNN's input are expected to cause no difference in its output, or to cause specific differences. For example, DeepXplore [7] generates adversarial perturbations using gradient ascent that vary a minimal number of pixels in an image and compares the DNN accuracy for an autonomous vehicle steering model. DeepHunter [39] employs a fuzzer to make changes that preserve the semantic meaning of an image, much like changing the image on a billboard or the color of a building would preserve the road. DeepRoad [50], DeepTest [6], Zhou et al. [49], and Nie et al. [51] apply image perturbations of varying sophistication, mimicking effects like weather conditions or camera distortion, while assessing their impact across various trained models and architectures. To generalize the validity of such adversarial perturbations on single images, efforts like Duan et al. [19] aim to hide adversarial perturbations in the physical environment.

As argued earlier, the effectiveness of DNN testing techniques in detecting faults does not necessarily map to the whole system. DNN faults may not cause a system failure (i.e., a DNN steering error is bounded by the system controller), and behaviors considered benign at the DNN level may become problematic for the system (i.e., an accumulation of small steering errors) [52]. The next set of complementary techniques targets the system.

B. *Testing Autonomous Vehicles*

Autonomous vehicle test strategies can be divided along two axes: black-box or gray-box, state-aware or state-adaptive.

Black-Box testing of AVs treats the system as an input/output function and focuses on generating a challenging environment configuration for the system [14], [15], [16], [17], [53]. These frameworks commonly rely on search-based manipulation of the configuration space defining the environment (e.g., road topology, objects, actors), to find scenarios that cause the system to misbehave. For example, Althoff et al. [17] aims to create critical situations with a small solution space where the autonomous vehicle must avoid a collision. Among these efforts, we note the use of simulation to be pervasive and cost-effective to detect different types of faults [54], [52].

Gray-box testing of AVs peeks into the system to more effectively generate tests. Some approaches target particular system components [18], [5] while others focus on particular interactions [32], [55]. Other efforts target the whole system but use insights about high-level component mechanisms to better generate inputs. DeepJanus [56], for example, accounts for learned components by performing an evolutionary search of the input domain model to find frontier inputs for DNN-dependent systems. Other approaches are more focused on the DNN, analyzing the DNN like other adversarial mechanisms but to derive system tests [57], [58], [11], [13]. DeepBillboard [11], the technique that inspired this work, changes the appearance of roadside billboards based on DNN analysis while accounting for the impact on system behavior. It is also novel in that it aims to generate perturbations based on a sequence of images collected over a trajectory. Yet, as previously explained, it does not account for the impact that the perturbation may have on the system state, which may render the perturbation ineffective as the affected trajectory diverges from the one in which the images were collected.

State-aware techniques consider the state of the vehicle, perceived externally by black-box approaches or through the system internals in gray-box approaches, to evaluate and guide testing [14], [32], [11], [59]. For example, Gambi et al. [15] present an approach to manipulate roads through search-based procedural content generation, tracking out-of-bounds episodes such as vehicle lane departures to use as search criteria. Tuncali et al. [14] use a series of states including collisions and velocity to identify system inputs that falsify autonomous vehicle requirements. Similarly, Abdessalem et al. [32] induce states in which vehicle subcomponents such as cruise control and sign recognition issue conflicting commands through manipulation of the driving environment.

Other works like Patel et al. [58] incorporate a more sophisticated billboard with precise vehicle tracking sensors to update the perturbation displayed based on the pose of the vehicle. Besides the additional assumption of having access to a billboard equipped with a camera sensor to precisely track the vehicle, the approach training stage generates a perturbation for each pose of the vehicle similar to previous work on adversarial attacks rather than a cohesive end-to-end perturbation generation. Sadly, there is no implementation available nor enough implementation or study details in the paper for us to reproduce the approach as part of our assessment. Boloor et al. [57] generates state-aware lane markers to influence a vehicle's pose. The approach assumes that large portions of highly regulated environment features like road surfaces can be changed, and it generates perturbations by exhaustively exploring that space and examining vehicle crashes to determine the next candidate perturbation. These works show that setting the system in the right state and environment matters and that checking the system state at the test completion can be helpful to guide the generation of future tests. Deepbillboard [11], mentioned earlier, took a first step in generating **state-aware** perturbations that account for the system state to the extent that the system state is embedded in the image sequence it analyzes to generate the perturbation. Again, this line of work does not consider how the system state evolves throughout the test execution.

State-adaptive. Our approach is the first state-adaptive testing approach for autonomous vehicles to manipulate the environment while considering vehicle state to more effectively exploit system weaknesses. It is gray-box in that it analyzes the DNN in connection with system-level behavior. State-adaptivity also enables the approach to force complex maneuvers of multiple objectives that rely on the system state.

V. Conclusion

DeepManeuver is the first state-adaptive approach to generate adversarial perturbations that can cause complex maneuvers. Its power comes from a refinement cycle that interleaves perturbation generation and simulation, jointly updating the state of the vehicle and perturbation. Our study finds that DeepManeuver is more effective on average by 20.7pp in comparison to Deepbillboard, and on average by 8.6pp in comparison to DBB+ at generating perturbations that consistently lead to vehicle misbehavior than existing techniques, and that can successfully cause more complex and subtle multi-objective maneuvers.

In future work, we intend to pursue four general directions. First, we will tackle the computational cost of DeepManeuver. The perturbation refinement cycles required to update the system state and the perturbation is costly, yet there are plenty of opportunities to accelerate this process through parallelization, early stoppage for underperforming perturbations, and downsampling optimization timesteps between collected states. Second, we will extend our exploration of factors that may affect DeepManeuver performance. For example, we plan to assess its performance under additional models and vehicles, road topologies, attack surfaces, and parameter configurations. Third, DeepManeuver does not have to just cause detrimental maneuvers. We will explore the multi-target capabilities of DeepManeuver to assist vehicles performing helpful maneuvers in challenging contexts by for which they may not have sufficient training. Lastly, we intend to adapt this approach to a real-world context, expanding on early-stage exploratory studies we have performed on comma.ai's OpenPilot system. Changes to the approach itself, realities of perturbing industrially-produced self-driving networks, and environment conditions will play a large role in adaptation to real-world autonomous driving.

VI. Acknowledgements

This work was supported in part by NSF Award #1924777 and AFOSR Award #FA9550-21-1-0164.

References

[1] C. Isidore and P. Valdes-Dapena, "Tesla is under investigation because its cars keep hitting emergency vehicles," *CNN*. [Online]. Available: https://www.cnn.com/2021/08/16/business/tesla-autopilot-federal-safety-probe/index.html

[2] T. Levin, "Tesla's full self-driving tech keeps getting fooled by the moon, billboards, and burger king signs," *Business Insider*. [Online]. Available: https://www.businessinsider.com/tesla-fsd-full-self-driving-traffic-light-fooled-moon-video-2021-7

[3] R. Cellan-Jones, "Uber's self-driving operator charged over fatal crash," *BBC*. [Online]. Available: https://www.bbc.com/news/technology-54175359

[4] M. Harris, "Google reports self-driving car mistakes: 272 failures and 13 near misses," *The Guardian*. [Online]. Available: https://www.theguardian.com/technology/2016/jan/12/google-self-driving-cars-mistakes-data-reports

[5] S. Pavlitskaya, S. Ünver, and J. M. Zöllner, "Feasibility and suppression of adversarial patch attacks on end-to-end vehicle control," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–8.

[6] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, 2018, pp. 303–314.

[7] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," *Commun. ACM*, vol. 62, no. 11, p. 137–145, Oct. 2019. [Online]. Available: https://doi.org/10.1145/3361566

[8] E. Santana and G. Hotz, "Learning a driving simulator," *ArXiv*, vol. abs/1608.01230, 2016.

[9] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[10] "Waymo open dataset: An autonomous driving dataset," 2019.

[11] H. Zhou, W. Li, Z. Kong, J. Guo, Y. Zhang, B. Yu, L. Zhang, and C. Liu, "Deepbillboard: Systematic physical-world testing of autonomous driving systems," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, 2020, pp. 347–358.

[12] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 86–94.

[13] Z. Kong, J. Guo, A. Li, and C. Liu, "Physgan: Generating physical-world-resilient adversarial examples for autonomous driving," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 14 242–14 251.

[14] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1555–1562.

[15] A. Gambi, M. Mueller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 318–328. [Online]. Available: https://doi.org/10.1145/3293882.3330566

[16] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: A language for scenario specification and scene generation," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 63–78. [Online]. Available: https://doi.org/10.1145/3314221.3314633

[17] M. Althoff and S. Lutz, "Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1326–1333.

[18] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing vision-based control systems using learnable evolutionary algorithms," in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, 2018, pp. 1016–1026.

[19] R. Duan, X. Ma, Y. Wang, J. Bailey, A. K. Qin, and Y. Yang, "Adversarial camouflage: Hiding physical-world attacks with natural styles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[20] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[21] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," *CoRR*, vol. abs/1608.04644, 2016. [Online]. Available: http://arxiv.org/abs/1608.04644

[22] BeamNG, "Beamng.drive vehicle simulator," https://www.beamng.com/game/, 2020.

[23] ——, "Beamngpy: Python api to beamng.drive," https://github.com/BeamNG/BeamNGpy, 2020.

[24] Y. Lin and S. Saripalli, "Sampling-based path planning for uav collision avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3179–3192, 2017.

[25] V. Corcoba, Z. Falomir, and N. Mohamed, "Using wearable sensors to detect workload on driving simulated scenarios," in *Artificial Intelligence Research and Development*. IOS Press, 2018, pp. 287–296.

[26] A. Gambi, T. Huynh, and G. Fraser, "Generating effective test cases for self-driving cars from police reports," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 257–267. [Online]. Available: https://doi.org/10.1145/3338906.3338942

[27] A. Adel, M. Mahmoud, N. Sayed, O. Hisham, O. Ossama, P. Adel, Y. Ayman, M. I. Awad, S. A. Maged, S. M. Umer, H. Iqbal, and H. F. Maqbool, "Design of a 6-dof hydraulic vehicle driving simulator," in *2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*, 2020, pp. 170–175.

[28] Udacity, "Udacity self-driving car," https://github.com/udacity/self-driving-car, 2016.

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

[30] J. Chen, X.-W. Li, and Q.-H. Wang, "Deep learning for improving the robustness of image encryption," *IEEE Access*, vol. 7, pp. 181 083–181 091, 2019.

[31] M. von Stein and S. Elbaum, "Finding property violations through network falsification: Challenges, adaptations and lessons learned from openpilot," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '22. Association for Computing Machinery, 2022.

[32] R. Ben Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, "Testing autonomous cars for feature interaction failures using many-objective search," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2018, pp. 143–154.

[33] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," *IEEE Transactions on Software Engineering*, vol. 48, no. 1, pp. 1–36, 2022.

[34] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, p. 100270, 2020.

[35] J. Guo, Y. Jiang, Y. Zhao, Q. Chen, and J. Sun, "Dlfuzz: Differential fuzzing testing of deep learning systems," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 739–743. [Online]. Available: https://doi.org/10.1145/3236024.3264835

[36] J. Wang, J. Chen, Y. Sun, X. Ma, D. Wang, J. Sun, and P. Cheng, "Robot: Robustness-oriented testing for deep learning systems," in *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*. IEEE, 2021, pp. 300–311. [Online]. Available: https://doi.org/10.1109/ICSE43902.2021.00038

[37] Y. Sun, M. Wu, W. Ruan, X. Huang, M. Kwiatkowska, and D. Kroening, *Concolic Testing for Deep Neural Networks*. New York, NY, USA: Association for Computing Machinery, 2018, p. 109–119. [Online]. Available: https://doi.org/10.1145/3238147.3238172

[38] S. Gerasimou, H. F. Eniser, A. Sen, and A. Çakan, "Importance-driven deep learning system testing," in *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, G. Rothermel and D. Bae, Eds. ACM, 2020, pp. 702–713. [Online]. Available: https://doi.org/10.1145/3377811.3380391

[39] X. Xie, L. Ma, F. Juefei-Xu, M. Xue, H. Chen, Y. Liu, J. Zhao, B. Li, J. Yin, and S. See, "Deephunter: A coverage-guided fuzz testing framework for deep neural networks," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2019. New York, NY, USA: Association

for Computing Machinery, 2019, p. 146–157. [Online]. Available: https://doi.org/10.1145/3293882.3330579

[40] F. Tramèr, V. Atlidakis, R. Geambasu, D. J. Hsu, J. Hubaux, M. Humbert, A. Juels, and H. Lin, "Discovering unwarranted associations in data-driven applications with the fairtest testing toolkit," *CoRR*, vol. abs/1510.02377, 2015. [Online]. Available: http://arxiv.org/abs/1510.02377

[41] J. Kim, R. Feldt, and S. Yoo, "Guiding deep learning system testing using surprise adequacy," *CoRR*, vol. abs/1808.08444, 2018. [Online]. Available: http://arxiv.org/abs/1808.08444

[42] S. Dola, M. B. Dwyer, and M. L. Soffa, "Distribution-aware testing of neural networks using generative models," in *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*. IEEE, 2021, pp. 226–237. [Online]. Available: https://doi.org/10.1109/ICSE43902.2021.00032

[43] Y. Tian, Z. Zhong, V. Ordonez, G. E. Kaiser, and B. Ray, "Testing DNN image classifiers for confusion & bias errors," in *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, G. Rothermel and D. Bae, Eds. ACM, 2020, pp. 1122–1134. [Online]. Available: https://doi.org/10.1145/3377811.3380400

[44] J. Uesato, A. Kumar, C. Szepesvári, T. Erez, A. Ruderman, K. Anderson, K. Dvijotham, N. Heess, and P. Kohli, "Rigorous agent evaluation: An adversarial approach to uncover catastrophic failures," *CoRR*, vol. abs/1812.01647, 2018. [Online]. Available: http://arxiv.org/abs/1812.01647

[45] A. Dwarakanath, M. Ahuja, S. Sikand, R. M. Rao, R. P. J. C. Bose, N. Dubash, and S. Podder, "Identifying implementation bugs in machine learning based image classifiers using metamorphic testing," in *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 118–128. [Online]. Available: https://doi.org/10.1145/3213846.3213858

[46] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6572

[47] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[48] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortés, "A survey on metamorphic testing," *IEEE Transactions on Software Engineering*, vol. 42, no. 9, pp. 805–824, 2016.

[49] Z. Q. Zhou and L. Sun, "Metamorphic testing of driverless cars," *Commun. ACM*, vol. 62, no. 3, p. 61–67, feb 2019. [Online]. Available: https://doi.org/10.1145/3241979

[50] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2018, pp. 132–142.

[51] J. Nie, J. Yan, H. Yin, L. Ren, and Q. Meng, "A multimodality fusion deep neural network and safety test strategy for intelligent vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 310–322, 2021.

[52] F. U. Haq, D. Shin, S. Nejati, and L. C. Briand, "Comparing offline and online testing of deep neural networks: An autonomous car case study," in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, 2020, pp. 85–95.

[53] T. Woodlief, S. Elbaum, and K. Sullivan, "Semantic image fuzzing of AI perception systems," in *Proceedings of the 44th International Conference on Software Engineering*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1958–1969. [Online]. Available: https://doi.org/10.1145/3510003.3510212

[54] A. Afzal, C. L. Goues, M. Hilton, and C. S. Timperley, "A study on challenges of testing robotic systems," in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, 2020, pp. 96–107.

[55] C. E. Tuncali, G. Fainekos, D. Prokhorov, H. Ito, and J. Kapinski, "Requirements-driven test generation for autonomous vehicles with machine learning components," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 2, pp. 265–280, 2020.

[56] V. Riccio and P. Tonella, "Model-based exploration of the frontier of behaviours for deep learning system testing," *CoRR*, vol. abs/2007.02787, 2020. [Online]. Available: https://arxiv.org/abs/2007.02787

[57] A. Boloor, X. He, C. Gill, Y. Vorobeychik, and X. Zhang, "Simple physical adversarial examples against end-to-end autonomous driving models," in *2019 IEEE International Conference on Embedded Software and Systems (ICESS)*, 2019, pp. 1–7.

[58] N. Patel, P. Krishnamurthy, S. Garg, and F. Khorrami, "Adaptive adversarial videos on roadside billboards: Dynamically modifying trajectories of autonomous vehicles," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5916–5921.

[59] S. Feng, X. Yan, H. Sun, Y. Feng, and H. X. Liu, "Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment," *Nature Communications*, vol. 12, no. 748, 2021.