Improving Robustness of DNN-Dependent ADS: Transforming Input Distributions to Account for Inference and System State

PhD Proposal

Meriel von Stein
July 15, 2024

A dissertation thesis proposal presented for the degree of
Doctor of Philosophy

Department of Computer Science
University of Virginia

Abstract

Autonomous Driving Systems (ADSs) are becoming more advanced and ubiquitous, enabled by increasingly sophisticated deep neural networks (DNNs). As ADSs' autonomy levels rise, so does the cost and complexity of their failures. Often, these failures arise when these DNNs are less robust than expected. In studying these systems, I found that these failures can occur due to unexplored inputs from the long tail of driving scenarios or when the system evolution affects the input distribution. These circumstances are common but challenging to accommodate because their effects are difficult to anticipate and solutions may not generalize, leaving us with a brittle system. I postulate that the impact of input distribution shifts on the robustness of a DNN-dependent system can be manipulated through the careful design and encoding of transformations that account for their effects on DNN predictions, analysis of their compounding effects on system state, and naturalness. To overcome these threats to robustness, I have developed two types of techniques. First, I have engineered techniques to mitigate robustness-related failures when the cause is known but the effect on the DNN prediction is not, specifically for the common scenario when a sensor component used to collect the training dataset for a DNN onboard an ADS is swapped out. Second, I have extended adversarial test generation techniques, which aim to produce input perturbations that cause a DNN to compute incorrect outputs and estimate DNN robustness, to consider how the effect of perturbations are attenuated by other ADS subsystems and are less effective as ADS state evolves. However, these perturbations are often not in distribution and appear unnatural, making them easy to spot and dismantle or less likely to resemble deployment conditions. I will investigate two approaches to increase naturalness while retaining perturbation strength: constraining perturbation generation techniques to only reproduce features seen during DNN training, and manipulating unconstrained perturbations with diffusion models to appear naturalistic to humans. Overall, the family of proposed techniques takes a significant step to improving ADS robustness.

1 Introduction

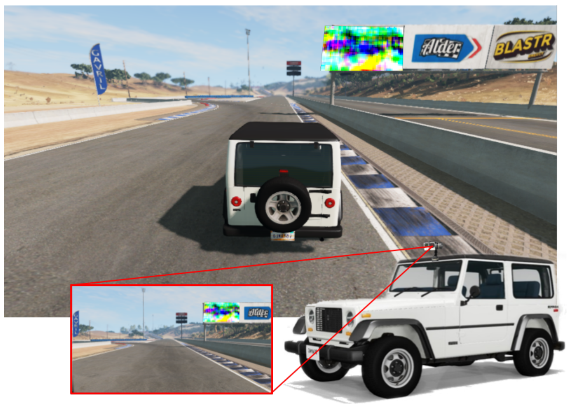
Autonomous driving systems (ADS) are increasingly normalized in our day-to-day lives. This trend towards automation is projected to continue as its advantages, reliability, and cost grow [52]. ADS often rely on deep neural networks (DNNs) to automate tasks that humans could otherwise reliably perform. Producing these DNNs is expensive, requiring massive amounts of data, labelling, training, and iterations of refinement [76, 86, 96]. Despite their critical role in autonomy and cost-intensive production, DNNs are often brittle to shifts in their input distribution, even ones imperceptible to humans [15]. In previously studying these systems [83] I have shown that even the DNNs of sophisticated, road-tested commercial driver assistance systems are susceptible to perturbations that force violations of properties integral to system safety.

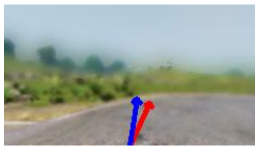
More generally, these DNNs suffer from a lack of robustness to commonly occurring shifts to the DNN's input distribution. Albarghouthi defines robustness as a desirable DNN property where 'small perturbations to inputs should not result in changes to the output of the neural network'. Clearly, 'small' is context-dependent, and a slightly generalized version of that definition accounts for cases where small changes in the input may cause changes in the output [5]. I examine two such changes.

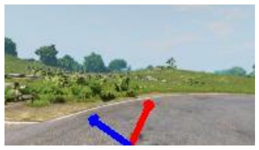
One type of change that threatens system robustness stems from the presence of unexpected objects in the deployment environment. This can happen accidentally by inputs from the current deployment environment drifting out of distribution with the intended deployment context [9], the occurrence of a rare event that the DNN's training set did not capture [18, 41, 46], or an intentional reconfiguration of the environment [32]. These unexpected features can also be artificially constructed, easily integrated into the environment [33, 44] and can fool a variety of sensors [14, 54]. Artificially constructed adversarial examples for autonomous mobile robots often directly reuse problem setups from the machine learning community, with existing techniques failing to account for the unique considerations of these systems, such as system state, which has a cascading effect on the DNN input distribution shift as the adversarial example takes effect and on the spatiotemporal dependency of inputs and outputs. While these artificially constructed adversarial features can mislead systems with varying degrees of effectiveness, they are also limited in that they do not resemble features for which the DNN was trained, they are not optimized for the surrounding deployment environment (resulting in mixed success or success limited to a single environment), and they are visually identifiable by humans.

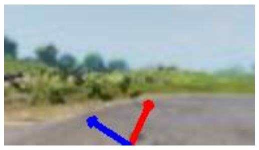
Another type of change originates in changes to the system itself, such as sensor hardware migration. Sensor hardware migration is where one sensor is replaced by another, for example to reduce cost or increase sensing quality [85]. Sensor hardware migrations occur frequently in practice (see Table 1). These migrations are problematic because they may render data different from that employed in the ADS development, potentially affecting the system performance and mitigating their downstream effects on DNN predictions can be costly [49]. Although these migrations are extremely common and the problematic changes to the sensor readings between hardware is well-documented, relatively little work has emerged to mitigate these changes in ADS or other safety-critical systems. This leaves a key piece of the ADS pipeline exposed to failures with high cost to mitigate this common occurrence during development.

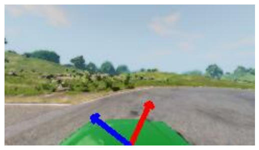
I postulate that the impact of input distribution shifts on the robustness of a DNN-dependent system can be manipulated through the careful design and encoding of transformations that account for their effects on DNN predictions, analysis of their compounding effects on system state, and naturalness. Section 1.1 provides an illustrative example of the types of challenges I plan to tackle. Section 1.2 outlines completed and proposed work.


(a) Motivating example of an ADS


(b) Shallow Depth


(c) Resolution Increase


(d) Resolution Decrease


(e) Fisheye Lens

(f) Image transformations that result from camera migrations and the corresponding change in steering prediction. (red arrow is the original prediction, blue arrow is the new).

Figure 1: Motivating examples for changes in ADS.

1.1 Motivation

Even the navigation DNNs of sophisticated commercial ADS have been proven to be insufficiently robust. I conducted the first falsification study of a real-world commercial ADS OpenPilot [83] and its end-to-end DNN `supercombo`. `supercombo` takes multiple inputs including two images and outputs a $(1, 6472)$ vector predicting lane lines, lead cars, ego trajectory, and prediction confidence, among other things. In order to successfully drive an OpenPilot system, `supercombo` output must be robust to small changes in input, e.g. small changes to an input image should not change lane line predictions by more than $\frac{1}{4}$ standard lane width (see Figure 2). With multiple high-dimensional inputs and outputs, a recurrent architecture, and over 100 hidden layers, `supercombo` itself is highly complex which can make it difficult to test. Falsification can only provide a measure of local robustness to pixel-level input manipulation, and the complexity of `supercombo`, the OpenPilot system, and ever-changing driving environments can present a myriad of threats to system robustness that are difficult to uncover through state-of-the-art falsification techniques.

In spite of the falsification study findings, pixel-level adversarial manipulation is not as powerful when considering the full scope of real-world ADS development and deployment. These techniques are limited in that: (1) pixel-level perturbations found during falsification often cannot be applied under deployment as they assume full control over the sensor reading; (2) they do not model realistic system changes that threaten robustness such as sensor hardware migration; (3) they lack knowledge of their compounding effect on ADS state and so these local robustness failures may not translate to system-level failures; and (4) they lack naturalness in that they do not mimic the circumstances of real-world ADS failures.

To reduce the assumption of full control, existing adversarial example generation techniques have taken a variety of approaches. Adversarial example generation is a testing technique that helps to determine the robustness of a DNN-dependent system, especially perception-dependent systems due to their exceptionally

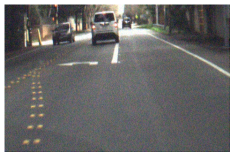
Systems	Camera hardware migration	Feature change
Level 5 / Woven Planet	100m to 60m stereo depth accuracy[66]	Depth of field
Waymo	Pinhole camera to fisheye lens camera [87]	Fisheye
Waymo	Rolling shutter to global shutter [87]	Removed artifacting
Argo v1.0 to 2.0 (ring)	(1920 x 1200) to (2048 x 1550) resol. [20, 88]	Resolution increase
Argo v1.0 to 2.0 (front stereo)	(2056 x 2464) to (2048 x 1550) resol. [20, 88]	Resolution decrease
Argo v1.0 to 2.0	Landscape to portrait orientation [20, 88]	Orientation
Tesla AutoPilot 2 to 2.5/3.0	RCCC filter to RCCB.8 filter [77]	Filter color shift
Tesla AutoPilot 1.0 to 2.0	Black and white to RGB color [77]	Dimensionality inc.

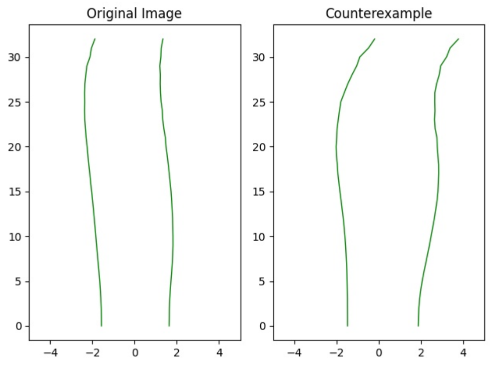
Table 1: Common hardware migrations and resulting image transformations. See Figure 1 for image examples captured through a subset of these cameras.

large and complex input space. More recent works have adjusted their threat model to only manipulate a limited set of features [44, 79]. However, these techniques do not account for changes in state once the perturbation takes effect during generation or only evaluate on single-image classification tasks.

To exemplify robustness to system changes, Table 1 lists real-world commercial ADSs similar to the Figure 1 ADS, the camera hardware migration they underwent between versions, and the resulting feature change in the camera sensor readings. Figures 1b-e show camera sensors with varying intrinsic parameters corresponding to Table 1 rows 1, 2, 5, and 4 respectively and the resulting steering DNN prediction change. The computer vision community has proposed limited solutions to mitigate some of these feature changes individually. Image transformations such as depth of field, camera motion, or lossy artifacting such as lens flare each require a unique mitigation technique [2, 60, 69, 74]. Moreover, they are evaluated on reconstruction error (a pixel-wise metric) [7], or inception score based on a learned discriminator [36]. These metrics are problematic for ADS robustness due to the safety critical function of the downstream DNN. Because camera hardware changes persist from timestep to timestep, there is a high probability that the resulting prediction errors will result in a crash.

(a) Original Image


(b) Counterexample



(c) Change in `supercombo` lane line predictions between original image and counterexample.

Figure 2: Original comma.ai dataset image and counterexample with low-level noise within ϵ=10 and corresponding output from `supercombo`.

To incorporate knowledge of state, existing techniques have taken preliminary steps (see Section 2.2). However, adversarial testing does not consider system state during attack optimization. This results in attacks that are physically problematic [12, 28] or self-defeating attacks that push the ADS into a region of the environment for which the attack was not optimized [44, 98], leading to an inconsistent input distribution later in the test. This inconsistent input distribution between generation and testing results in reduced efficacy of the adversarial attack and difficulty gauging that efficacy from metrics produced during the generation phase.

And lastly, these perturbations do not appear natural, in that they are dissimilar from the training set of the DNN, the surrounding environment, and the distribution of natural images [37, 97]. Real-world ADS failures are often caused by higher-level feature changes, such as roadside signs, turning on an emergency vehicle's flashing lights [41], or unusual lighting conditions [9]. State-of-the-art work on natural perturbations still assumes control over large areas of the image, has diverging definitions and metrics for what qualifies as a natural perturbation, and do not consider systems with spatiotemporal aspects. As a result, we need a strategic way to identify and test a range of possible features of DNN inputs that violate expected system behavior and conform to the possible deployment environments of the system and therefore also conform to the possible input distribution of the DNN.

Section	Technique	Input Distribution Shift Type	Transformation Mechanism
1	OpenPilot DNN safety property falsification [83]	adversarial	–
3.2	Transformations for supporting camera sensor hardware migration [85]	sensor migration	autoencoder
3.3	State-adaptive in-situ adversarial perturbations [84]	adversarial	state-adaptive optimization
3.4	Natural adversarial perturbations (features, disguised)	adversarial	distribution density, stable diffusion

Table 2: An overview of the completed work, proposed work, and which challenge it tackles. Rows 1-3 are completed work and Row 4 is proposed work.

1.2 Contributions

Table 2 provides an overview of the novel contributions described in this proposal, related sections and how they address each of these two axes. Columns 1 and 2 list the techniques described in this proposal and their sections. Column 3 lists the origin of the distribution shift that occurs to the DNN inputs during the ADS development lifecycle and threatens system robustness. Column 4 describes the transformation mechanism each technique uses or will use to manipulate the distribution shift.

This dissertation proposal includes two completed pieces from my body of work: (1) the mitigation of feature deformations in sensor readings to preserve DNN predictions in the presence of sensor hardware migrations; and (2) the introduction of state-adaptive adversarial perturbations that increase the strength of in-situ adversarial patch attacks and the versatility of the maneuvers they can induce. Then, I propose an exploration of and extensions to techniques for natural adversarial patch generation. The expected contributions of this dissertation are:

- A study of a real-world system that shows input distribution related failures with system-level consequences are common and reproducible [83];
- A technique to provide a low-cost, effective, and highly generalizable remedy to sensor hardware migration that is hardware independent and an implementation and comparison against state-of-the-art techniques to mitigate the impact of sensor hardware migrations, specifically camera sensors in commercial ADS [85];
- An adversarial testing framework for autonomous vehicles that generates perturbations while accounting for vehicle state changes and capabilities and makes subtle changes viable in real world contexts that can make previously safe environments unsafe and cause system-level failures [84];
- A proposed extension to the previous contribution that allows for natural perturbations within the training distribution of the DNN and natural perturbations in terms of noticeability to humans;
- Open-source repositories of results and tools for all the above techniques: `https://missmeriel.github.io/artifacts/`; and
- Broader impact through mentorship of 8 undergrads in my project that iteratively built up an end-to-end robotics and deep learning pipeline for better reproducibility and comparison of research techniques. `https://github.com/MissMeriel/ROSbot_data_collection/tree/rosbotXL`

2 Background

This section provides descriptions of key areas of related work, namely input transformations to support Section 3.2, adversarial testing to support Section 3.3, and naturalness to support Section 3.4.

2.1 On Encoding and Transforming Input Distributions

This section covers previous work on how to encode image distributions and how to engineer transformations from one distribution to another. These previous works often employ learned components to encode these distributions or learn transformations and use a variety of architectures.

Encoding the features of image distributions is a wide field of study [13, 30, 42]. An encoder [59, 72] is a type of neural network that learns efficient embeddings of unlabeled data. The set of embeddings is often referred to as a dictionary. An input image is passed through the encoder to generate a dictionary embedding, then the decoder uses the embedding to generate the reconstruction. My completed and proposed techniques primarily use variational auto-encoders (VAEs) [7, 47] which use an encoder, dictionary, and a decoder to reconstruct the original data using its lower-dimensional embedding. VAEs can suffer from gradient collapse and have difficulty with non-categorical data. Vector-quantized VAEs (VQ-VAEs) [82] were designed to overcome such deficiencies by using an autoregressive (learned rather than static) prior and discretizing the embedding dictionary, which is a more natural representation modality for languages, image features, and planning. VQVAEs are well-suited to encode images correlated with a continuous output distribution from the original DNN. A discrete dictionary also circumvents "posterior collapse" where the decoder overpowers the latent encoding and leads to poor reconstructions. Encoders encompass many different architectures beyond VAEs and VQVAEs, but the exploration of what these encoders can learn has been limited to the pixel-level image reconstruction. State-of-the-art encoders do not try to encode other (state-related) information like the steering angle associated with images as a way to guide reconstruction or group the latent space of embeddings. See Section 2.3 for other generative architectures that can encode image distributions.

Using custom image transformations to approximate a feature space mapping has been explored in the computer vision community [2, 7, 36, 69, 74]. However, producing mappings between sensor changes is a relatively new problem for engineers of cyber-physical systems [16, 17]. Current research [43] shows that engineers can introduce transformations to "clean" images, but cannot always translate between them. For example, novel view synthesis [19] can support the transformation between images due to differences in camera extrinsics by interpolating between perspectives. However feature deformations due to changes in camera intrinsics prevent this transformation. Moreover, the computer vision community tends to evaluate these transformations on image similarity metrics rather than their prediction fidelity by a downstream image processing neural network [7, 36]. This setup, sometimes referred to as network chaining [67, 92], appears in commercial ADS [65] and is a critical success metric for image transformations used by ADS.

2.2 On Adversarial Testing

This section covers common deep learning adversarial testing strategies for images, as well as software-based strategies for testing autonomous robotic systems like ADS. These testing strategies vary from low-level adjustments to individual pixels, to changing the appearance of features, to reconfiguring the entire driving environment including the road surface.

The machine learning community has pioneered specialized mechanisms to judge DNN robustness through the generation of adversarial inputs [4, 31]. Many of these techniques have been extended to apply to the ADS context. The seminal work DeepXplore [64] generates adversarial perturbations using gradient ascent that vary a minimal number of pixels in single images and diffs the prediction accuracy of an autonomous vehicle steering model. These pixel-level perturbations can give a measure of local robustness around the original images used to generate a perturbation, but are unlikely to occur in practice because they assume control over the entire image space.

Other techniques apply adversarial perturbations at the feature level of images, a higher level abstraction than pixel-wise manipulation [57, 78, 94, 101]. These techniques generate perturbations of varying sophistication, mimicking effects like weather conditions or camera distortion, while assessing their impact across various trained models and architectures. However, these techniques still generate and test perturbations on static images or even single images, and as a result do consider system state. As stated previously, the effectiveness of DNN robustness testing does not necessarily translate to system failures. 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) [34].

Although classical adversarial attacks for DNNs do not measure the effect on the surrounding system, ADS software testing approaches often employ search-based techniques that test the system as a whole in a state-aware manner, focusing on interactions of features or properties of the driving environment. Many of these efforts target the whole system but use insights about high-level component mechanisms to better generate inputs that might cause behaviors that violate system safety [10, 68, 80, 81]. Others focus on searching the space of environment features to create new adversarial environments for system tests [29, 90]. 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. However, they are using search-based exploration for a near-infinite configuration space – the space of inputs to the ADS. Furthermore, the search process treats the ADS as a black box, is open-loop and does not adapt to the evolving system state, and gives no indication that the generated test will provoke a system failure until it is actually run, only that the newly generated test is similar in some properties to previous failing tests.

Other approaches focus on feature-level perturbations, analyzing the DNN like other adversarial mechanisms but augmented by system state [44, 61, 62, 98]. These approaches all use a grey-box model of a regression network-dependent ADS to change the appearance of a billboard within view of the ADS perception sensor object in the driving environment in order to influence system behavior. These works took the first steps in generating perturbations that account for the system state. Yet, they are still open-loop in that their perturbation generation processes do not update for the impact that the perturbation may have on the system state, which may render the perturbation ineffective over time as the vehicle trajectory diverges from the one for which generation was optimized. Patch attacks designed to defeat classification and detection networks usually use a similar state-aware open-loop approach to generation but attack the physical objects directly, such as stop signs or lane markers [12, 28], which are highly regulated objects in the driving environment and illegal to deface. Although these state-aware techniques show greater promise than white-box feature-level perturbations on single images or black-box search-based software testing techniques, they are still open-loop and can be self-defeating as the perturbation takes effect.

2.3 On Naturalness

This section covers previous work on definitions of image naturalness, why natural adversarial examples are worthwhile to generate, commonly used strategies to generate them, and the various metrics with which naturalness is judged.

Zhao et al. [97] determine classic pixelwise adversarial examples [31, 50, 75] to be unnatural in that they are statistically unlikely to occur under deployment, and thus they do not expose useful "blind spots" in learned components and do not offer helpful insights into the fundamental decision behavior inside the black-box learned component. In other words, they do not help answer the question, "why is the model's inference different for the adversarial example versus the unperturbed input?" Zhao et al. apply their technique to very small (100×100 or less) images and textual natural language examples and search for adversaries in a dense and continuous representation of the training data, a common strategy [24, 25] but one that can prove difficult for large and complex image distributions [73]. Note that similar work [3, 40] defines natural adversarial examples as semantic adversaries that manipulate image properties like color shift or artifacting derived from sensing hardware. This proposal uses Zhao et al.'s definition of naturalness.

Natural adversarial patches are an emerging type of attack. For example, MVPatch [100] creates "vivid and aggressive" adversarial patches for single-image person detection tasks. These patches are generated by obtaining the gradient for a patch using several losses for backpropagation, including object detection confidence and a pixelwise Minkowski distance to constrain naturalness. The patches are intended to be natural in that they appear on their own to be standalone natural images. However, they are not stealthy in relation to their surroundings [27]. MVPatch introduces the naturalness score which relies on cosine similarity score of the original image and the adversarial patch under several different generalization-related image transformations. However, the changes to these features must be significant enough to affect ADS behavior. While there are results on similar techniques, which involve human-imperceptible changes to roadside billboards, the studies were limited and no artifacts were available [44].

Diffusion models [38] have become popular for their ability to sample data distributions with high fidelity and diversity. Detecting synthetic images generated by stable diffusion models has shown to be less prone to producing distinctive patterns in their generated images than GANs and can be difficult to distinguish from real images [22, 70]. AdvDiffuser [21] harnesses the power of diffusion model to perturb classification examples by denoising between an original image and an image that has been attacked using projected gradient descent (PGD), iteratively removing unnatural components from the injected noise. AdvDiffuser restricts distance from the original image by restricting perturbations to regions that are not the salient object and using image comparison metrics [26, 39, 95] to compare to the unperturbed image. Although this technique assumes control over all aspects of the image that are not the salient object, the technique shows promise on classification tasks and could be extended to the ADS context.

Naturalness metrics are a known difficult problem, and the gold standard of naturalness is considered human adjudication [27, 100]. In a seminal work on naturalness, Hendrycks et al. [37] select natural adversarial examples that adhere to the Zhao definition by manually culling through several natural image classification datasets. While this approach has been replicated [63], the process is expensive and time-consuming and not feasible for all models, as classification is a clearer task than "correct" driving or other regression tasks with fuzzy comparison.

3 Research Progress

In this section I will provide a roadmap of my completed and pending work, starting with a general problem definition and following with each of my research threads. For each research thread, I refine the problem definition, summarize the technical approach, and highlight the major empirical findings.

3.1 Common Problem Definition

An ADS $\mathcal{S}$ is equipped with sensor c, a navigation DNN $\mathcal{N}$, and a system state space A. At each timestep t, c produces sensor reading x_t, which is consumed by $\mathcal{N}$ to produce control signal $\hat{\psi}_t$. That control signal $\hat{\psi}_t$ is then passed to $\mathcal{S}$ and attenuated by $\mathcal{S}$'s current state $a_t \in A$ to produce the signal it actuates, ψ_t. x is sampled from distribution $X_{\mathbb{R}} \subset X$, the set of all realistic inputs to $\mathcal{N}$ and a subset of all possible values the sensor reading can take. $X_{\mathbb{R}}$ is parameterized by c intrinsics and extrinsics, as well as the set of environments $\mathcal{E} = \{e_1, e_2, ...e_n\}$ that $\mathcal{S}$ is designed to navigate.

Over the course of the system lifecycle, $X_{\mathbb{R}}$ can shift due to intrinsic and extrinsic changes to c and changes to $\mathcal{E}$. We characterize that shift through function $T(X_{\mathbb{R}}) = X'_{\mathbb{R}}$ that produces a new inconsistent input distribution $X'_{\mathbb{R}}$ such that $\mathcal{N}(x \in X_{\mathbb{R}}) \neq \mathcal{N}(x' \in X'_{\mathbb{R}})$. This shift and resulting mispredictions $\mathcal{N}(T(X_{\mathbb{R}})) \neq \mathcal{N}(X_{\mathbb{R}})$ can lead $\mathcal{S}$ to inhabit states outside of the safe operation of the system, including failure states. The problem I address in this dissertation is how to approximate or invert T such that the system can reach desired states.

Refer to the Glossary (Appendix A) for a shortlist of terms and symbols introduced here.

3.2 Learning Transformations to Mitigate ADS Sensor Migration

3.2.1 Problem Definition

Given an ADS that relies on an $\mathcal{N}$ that is reliable but costly to produce, a sensor hardware migration from c to a new sensor c' that shifts $\mathcal{N}$'s input distribution from X_c to $X_{c'}$, the problem is to reduce the cost to produce $\mathcal{N}$ by finding a transformation that can map an $x_{c'}$ to x_c to produce a reconstruction image $\hat{x} \in \hat{X}$ such that:
(1) $T_{c':c}(X_{c'}) = \hat{X} \approx X_c$: the transformation of the input distribution from the new sensor must consistently approximate the input distribution from the original sensor c.
(2) $\text{argmin}_{T_{c':c}} \mathcal{N}(T_{c':c}(X_{c'}) = \hat{X}) - \mathcal{N}(X_c)$: the error between the prediction produced by $\mathcal{N}$ on the original sensor reading and $\mathcal{N}$ on the transformed new sensor reading is minimized.

Note this setup could consider a_t as an supplement of the inputs to $\mathcal{N}$ or the type of transformation $T_{c':c}$.

3.2.2 Technical Approach

To address this problem, previous techniques based on image reconstruction do not demonstrate that they can be applied to safety critical systems, and previous techniques based on retraining do not show that they are resource-efficient for ADS tasks (see Section 2.1). To address this failing, I developed a technique called PreFixer. PreFixer can systematically learn transformations $T_{c':c}$ to mitigate $T_{c:c'}$ for many types of sensor hardware migrations during the ADS development lifecycle. For example, $T_{c'c:}$ can reconstruct any of the sensor readings Figure 1c-e to approximate Figure 1a in a way that preserves the $\hat{\psi}$ of Figure 1a.

My approach relies on two key components to automatically compute the transformation $T_{c':c}$: a flexible variable encoder architecture and the ability to capture a small dataset with both sensors running concurrently. These components play a role in the collection, configuration, training, and deployment stages of the development lifecycle.

Figure 3 provides an overview of the PreFixer technique applied to the Figure 1 ADS. Step 1 is the collection of a collocated dataset using both sensors c and c' as well as $\mathcal{N}$'s predictions on c which are also being used to drive $\mathcal{S}$ through a subset of environments in $\mathcal{E}$. Step 2 is the configuration of the encoder that will perform the transformations. Step 3 is the training of the encoder using the collocated dataset and a loss function that utilizes predictions from the navigation network. This loss function uses the reconstruction loss of the encoder's reconstruction compared to the original $x_c \in X_c$ using $x_{c'}$, an accuracy loss of $\mathcal{N}(x_c)$ and the $\mathcal{N}(\hat{x})$, and an embedding loss to guide the embedding dictionary vectors. The training portion of Figure 3 shows training for one input. The c' reading $x_{c'}$ is passed to our augmented encoder, a VQVAE [82]. The resulting reconstruction $\hat{x} \in X_c$ is then used to calculate the terms of the *loss* function:

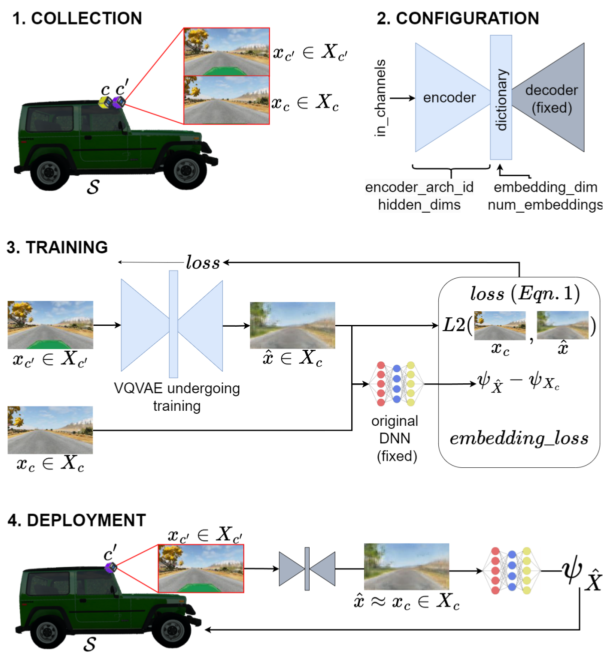


Figure 3: PreFixer overview of technique

$$loss = L_1(\psi_{\hat{x}} - \psi_x) + L_2(x_c - \hat{x}) + embedding\_loss \tag{1}$$

Step 4 is deployment where the original hardware c is replaced with the new sensor hardware c' and the encoder-decoder module is integrated into the system software as a preprocessing step to $\mathcal{N}$. Under deployment, the states inhabited by $\mathcal{S}$ $\vec{a}_{0,n} \in \mathcal{A}$ using the original software configuration are compared to those inhabited while using PreFixer and the new sensor to determine the efficacy of the learned $T_{c':c}$.

I developed several instantiations of the approach to support a diverse range of camera hardware migrations commonly occurring in practice, to be resource-efficient in that it is easier, faster, and cheaper to apply than existing techniques such as designing custom image transformations or retraining a DNN, and which provides a set of configuration principles that require minimal setup, parameter tweaking, and domain knowledge. To support a range of camera migrations, I tailor a VQVAE encoder to each transformation to avoid gradient collapse and leverage the output distribution of the original DNN. I augment the encoder architecture to be flexible enough to support variations in input image size and feature deformation, ensuring that all images can be encoded to the same low-dimensional space and reducing parameter tweaking once a sufficient embedding dimension has been found. To ensure resource efficiency, I use an augmented VQVAE training loss function to optimize PreFixer for the reconstruction of features important to $\mathcal{N}$, improving prediction accuracy on $\hat{x}$. This practice is rooted in recent results suggesting that a well-trained image processing model will help with the training of another [6, 67, 91]. Finally, to minimize investment in setup, parameter exploration, or domain knowledge, the VQVAE offers a single automated mechanism for unsupervised learning to automate the mapping between feature spaces of the new and old sensors, transformation regardless of type, eliminating the need for developer time to craft bespoke transformations or domain knowledge beyond the dimensions of the images.

As mentioned in Section 3.2.1, PreFixer can consider state as well but in this implementation it does not to better determine the difficulty of image transformations alone.

3.2.3 Results

My study on the PreFixer technique aims to answer the following research questions:

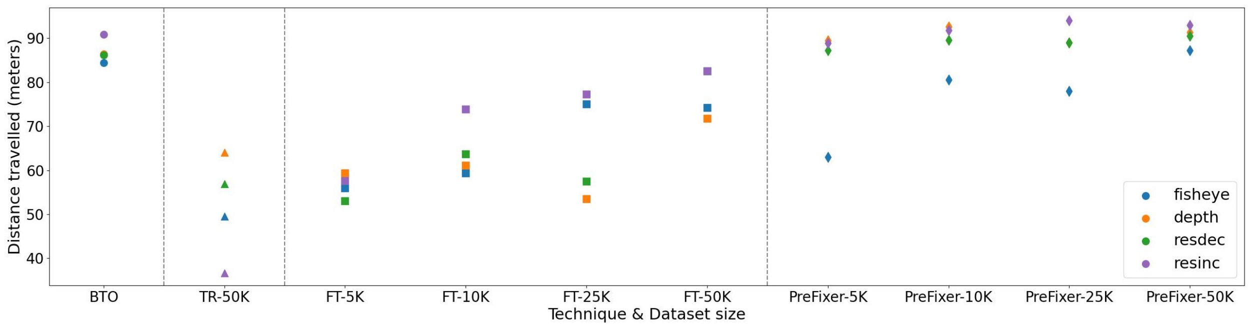


Figure 4: Effectiveness of PreFixer and baseline techniques with varying dataset sizes from 5K to 50K samples. My results show, on average, PreFixer is more successful than all other techniques, with all transformations achieving between 87.2 and 93.0m travelled. Moreover, PreFixer can learn an effective transformation with $\frac{1}{3}$ or less data than it takes to retrain a network.

RQ1) *How effective is PreFixer compared to other techniques in supporting common camera migrations?* To answer this question, I explore 4 image transformations related to camera migrations observed in real-world systems. I then test PreFixer on 10 validation road segments not seen in training.

RQ2) *How much data does PreFixer need to successfully learn each image transformation?* To answer this question, I compare performance across 4 dataset sizes for all 4 image transformations.

My study uses the BeamNG high-fidelity driving simulator [8]. For the autonomous vehicle under test, I equip the prepackaged "hopper" vehicle with an onboard camera at the top edge of the windshield. The camera has a 50° field of view angled upwards 5° relative to the vehicle pitch and collects images at 15 Hz to mimic established self-driving setups [1]. For vehicle control, I re-implemented the DAVE2 architecture [11], which consumes camera images to steer a vehicle. I trained DAVE2 with 145,521 images collected over multiple prebuilt BeamNG driving environments. The trained network final loss was 0.012 MSE between predicted and ground truth steering angles. A PID regulates throttle. This system is able to traverse all road segments in this study with the original camera configuration. I test all techniques in simulation on 10 unseen 100-meter road segments and performance was measured by average distance travelled.

Table 1 examples from rows 1, 2, 4 and 5 motivate the transformations I chose for this study. The fisheye transformation increases the field of view from 50° to 75°. The depth of field transformation decreases the depth in focus from 1000m to 100m. Depth and fisheye maintain the image size of 108×192. The resolution increase transformation increases original image dimensions from 108×192 pixels to 270×480 pixels. The resolution decrease transformation adjusts the original dimensions to 54×96 pixels.

As shown in Figure 4, I compare PreFixer against three baselines. First, Bespoke Transformation Only (BTO) employs a custom and domain-specific inverse transformation that maps the new sensor feature space back to the old sensor feature space ($T_{c':c}$) before prediction by $\mathcal{N}$. In practice, $T_{c':c}$ is designed and tuned over time by the developer using domain knowledge and validation results. Second, Transform and Retrain (TR) retrains the DAVE2 model architecture using a transformed version of the dataset where existing images are mapped to the feature space of the new sensor ($T_{c:c'}$). $T_{c:c'}$ is developed using a similar pipeline to BTO, but the transformation is inversed. Lastly, Fine-tuning (FT) uses warmstarting by keeping the weights of the trained DAVE2 model and fine-tuning those weights using new sensor data [6, 48]. Ash et al. mention this practice consistently hurts generalization, while deceptively having little effect on training accuracy. Completely retraining a DNN on a new dataset is costly [35, 45, 51, 53], so warm-starting and fine-tuning are viable options for some applications. For transformations involving changes to input size, the model's last convolutional layer was adjusted.

My results show that PreFixer can learn an effective transformation with $\frac{1}{3}$ or less data than it takes to retrain a network. Given a 50K dataset, PreFixer outperforms all baselines by a margin of 3m travelled or more. PreFixer also outperforms *TR* and *FT* for all transformations when comparing identical dataset sizes. Figure 4 shows that, on average, PreFixer is more successful than all other techniques, with all transformations achieving between 87.2 and 93.0m travelled. Even when compared with the best-performing alternative technique, *BTO*, PreFixer can equal or surpass the performance for three of the four transformations with just a 5K dataset. My technique can improve upon BTO for 3 out of 4 common image transformations using only a 5K dataset. Moreover, PreFixer is more affordable than retraining a DAVE2 network, even without considering the cost of collecting a new dataset for the new DAVE2.

3.3 DeepManeuver

3.3.1 Problem Definition

This work extends the common problem definition to include an attack surface $\mathcal{O}$ in $e \in \mathcal{E}$ visible to the vehicle's perception subsystem, as well as a target maneuver $\mathcal{M}$ defined by a sequence of states $\vec{m}_{0,n} \in \mathcal{A}$ inhabitable by $\mathcal{S}$ 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, $\mathcal{S}$ will inhabit a sequence of states $\vec{a}_{0,n}$ under the control of $\mathcal{N}$ as it navigates e within view of $\mathcal{O}$. $\mathcal{S}$'s state a_t at timestep t includes $\mathcal{S}$'s pose, the image it perceives through its camera x_c, and its actuated steering angle ψ_t. $\mathcal{O}$ is captured in the images $x_{c_{0,n}}$ embedded in states $\vec{a}_{0,n}$. The problem is to find a function $T_{orig:pert}(x_{c_{0,n}} \in \vec{a}_{0,n})$ by manipulating the appearance of $\mathcal{O}$ such that $argmin_{\mathcal{O}} \vec{m}_{0,n} - \vec{a}_{0,n}$.

3.3.2 Technical Approach

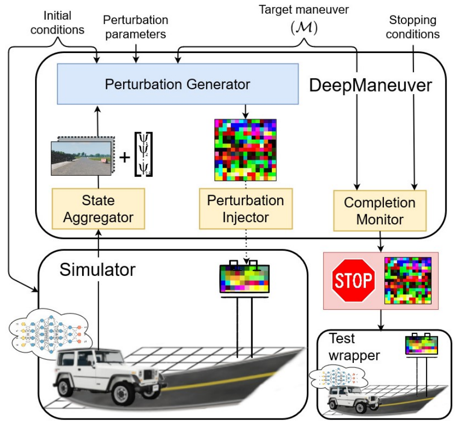
My approach DeepManeuver [84] attempts to address a key failing in the state of the art for adversarial attacks on ADS viable in a real-world deployment context [12, 44, 61, 99], namely that they do not optimize for the compounding effects of the perturbation on vehicle state. These in-situ patch attacks take a physical object in the deployment environment and apply some function $T_{orig:pert}$ to perturb its appearance in an attempt to induce a system-level failure. In this work I present a state-adaptive approach that interleaves perturbation generation and simulation, jointly updating the a_t of $\mathcal{S}$ and target state m_t that the perturbation seeks to induce in $\mathcal{S}$. The $T_{orig:pert}$ function, which in my study is applied to physical roadside billboards, is refined over each timestep as the perturbation of the previous timestep takes effect and weights the expected perturbation error according to size of the perturbation in the sensor reading image, which is a function of distance from the vehicle to the billboard $\mathcal{O}$. This allows for the prioritization of high-strength perturbations for more effective timesteps, and allows the capture of the full input distribution $X_{a_{0,n}}$ for which the perturbation generation function $T_{orig:pert}$ must be optimized in order to have a measurable effect on $a_{0,n}$ when the patch is deployed.

Figure 5a provides a conceptual overview of DeepManeuver applied to the motivating example in Section 1.1. The main components DeepManeuver and the Simulator iteratively update a perturbation *pert* 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. As $\mathcal{S}$ navigates e in view of $\mathcal{O}$, the State Aggregator aggregates a_t at each timestep and passes it to the Perturbation Generator. The Perturbation Generator uses this state sequence (specifically x_t and $\hat{\psi}_t$), $\mathcal{N}$, and $target_{\psi,t}$ to optimize *pert* over the updated state sequence. This component of the technique determines target steering value $target_{\psi,t}$ for $\mathcal{S}$ to attempt to actuate using $\vec{m}_{0,n} = \mathcal{M}$ at time t and a_t. Note that $\mathcal{M}$ may be specified in terms of any state variables but $target_{\psi,t}$ can only be specified in terms of the actuation controlled by $\mathcal{N}$. 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 perturbation optimization on the already-traversed trajectory. After perturbation optimization, the Perturbation Injector injects the updated *pert* into e through $\mathcal{O}$. The simulator then steps forward one timestep; $\mathcal{S}$ perceives the updated $\mathcal{O}_{pert,t}$ and advances in its trajectory, with $\mathcal{N}(img_{\mathcal{O}_{pert,t}})$ affecting the new a_{t+1}. The State Aggregator then captures a_{t+1} and adds the new x_{t+1} and ψ_t into the sequences utilized by the optimization process to preserve previous effects of $\mathcal{O}_{pert,t}$ in future optimizations. At each loop iteration, the Completion Monitor checks whether stopping conditions are met, such as driving off the road, crashing, or passing $\mathcal{O}_{pert,t}$ so it is no longer in view. Once a stopping condition has been met, the final perturbation is ready for deployment to the test wrapper.

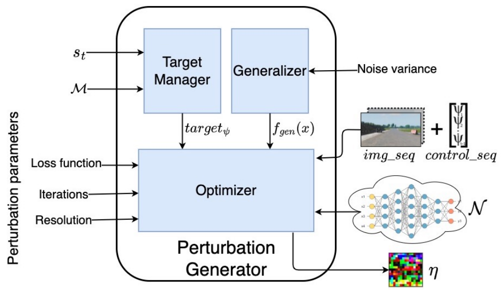
The Perturbation Generator in Figure 5b performs the critical operations of DeepManeuver to optimize the perturbation at each timestep. Once an image sequence is received, the Generalizer uses the noise variance parameter to create a function to produce variants of that sequence, increasing the adversarial strength [27] of the perturbation.[1] The Optimizer then consumes $\mathcal{N}$, img_seq and $control\_seq$, and $target_{\psi,t}$, and generates $T_{orig:pert}$ using projected gradient descent [15], accounting for two properties. First, the perturbation must maximize the vehicle state change toward the current m. Second, perturbation effects over $a_{0,t}$ must be consistent over time and space, as the current a_t depends on the effects of early perturbations. These two properties are enforced jointly at each iteration of perturbation generation by finding a $T_{orig:pert}$ that minimizes the loss functions $\mathcal{L}_1$ and $\mathcal{L}_2$ derived from two error terms:

$$\underset{T_{orig:pert}}{argmin}\,(\mathcal{L}_1(\mathcal{N}(img_n + T_{orig:pert}), target_{\psi,n}) + \sum_{t=0}^{n-1} \mathcal{L}_2(\mathcal{N}(img_t + T_{orig:pert}), \psi_t)) \tag{2}$$

Typically, existing techniques [44, 62, 98] 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 by embedding a simulator into the adversarial generation cycle, ensuring that effects of the perturbation are reflected in future states and perturbation updates are tailored to trajectory changes. 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 target maneuvers that require complex evolution of state and disrupt the vehicle trajectory in a predetermined, specifiable way. In summary, DeepManeuver is the first state-adaptive adversarial testing approach for autonomous vehicles.



(a) Conceptual Overview of DeepManeuver.



(b) DeepManeuver's Perturbation Generator (expanded view of blue box in Figure 5a).

Figure 5: Overview of DeepManeuver approach

3.3.3 Results

The DeepManeuver 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 I compare DeepManeuver to two versions of a state-of-the-art technique on 6 scenarios across 3 road topologies, and I 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, I explore the ability of DeepManeuver to achieve three multi-target maneuvers: hit a target, change lanes, and cut a corner.

Our study uses the same BeamNG driving simulator [8] and vehicle, camera setup, and DAVE2 architecture as Section 3.2. The system can follow the centerline of the road indefinitely in the racetrack environment used in this study.

I manipulate several variables associated with the environment. For **RQ1**, I perform a full factorial study 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

[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.

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 [98] study to make a fair comparison. DeepBillboard indirectly specified the billboard resolution according to pixel overlap and the shape of each billboard; in this work, I explore the performance of each technique on billboards with three different resolutions: 5×5, 10×10, 15×15. For **RQ2**, I evaluate DeepManeuver on 3 road topologies with 3 maneuvers: hit a target, change lanes, and cut a corner. Due to space limitations, only the "cut a corner" maneuver is discussed here.

Our study shows DeepManeuver can consistently produce perturbations that affect the vehicle differently based on state, with high accuracy in perturbing towards multiple target states. We compare DeepManeuver and two baseline comparison techniques, DeepBillboard and DBB+. DeepBillboard [99] inspired my technique and serves as a baseline for environmentally-situated adversarial perturbations that influence the actuation of autonomous vehicles. DBB+ 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 (but without the simulator in the loop). Table 3 summarizes the performance of these three techniques 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. Our primary consideration is the effect on the overall system behavior, so I count whether the target maneuvers are met (success rate), and I measure the average distance from the original trajectory (ADOT) in meters throughout the run once the perturbation began to take effect.[2]

		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**	**Success rate**	**ADOT**	**Success rate**	**ADOT**
Scenario 1: left turn on straight road	**5×5**	0.6%	2.18	8.8%	2.42	**26.4%**	**3.06**
	10×10	0.0%	1.99	5.8%	2.21	**30.4%**	**2.62**
	15×15	0.0%	1.96	0.2%	2.01	**21.0%**	**2.71**
Scenario 2: right turn on straight road	**5×5**	99.3%	2.35	99.4%	2.31	99.2%	2.50
	10×10	99.4%	2.35	99.6%	2.34	99.8%	2.34
	15×15	98.7%	2.33	99.0%	2.33	88.6%	2.35
Scenario 3: left turn on right-hand curve	**5×5**	0.2%	1.09	6.4%	1.27	54.6%	1.86
	10×10	8.8%	1.11	12.9%	1.32	10.8%	1.35
	15×15	17.6%	1.27	16.6%	1.31	3.4%	1.43
Scenario 4: right turn on right-hand curve	**5×5**	47.2%	2.55	99.6%	3.09	**100.0%**	**3.17**
	10×10	15.0%	2.48	79.0%	2.69	**94.2%**	**2.87**
	15×15	6.3%	2.34	65.0%	2.50	**81.6%**	**2.60**

Table 3: All techniques on 4 out of the 6 scenarios (topology-maneuver pairs) at 3 resolutions. Metrics are the success rate and ADOT. Best-performing values per row are in bold.

Table 3 shows 4 out of the 6 road topologies from the original study. Across all 6 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. Overall, my study finds that DeepManeuver is more effective on average by 20.7pp in comparison to state-of-the-art technique 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. [3]

To evaluate the effectiveness of DeepManeuver at inducing multi-target maneuvers, I 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 I keep the same configuration parameter values as reported in Table 3: `resolution`=10×10, `noise_var`=$\frac{1}{15}$, `iters`=400,

[2]The full paper also reports 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.

[3]Note that the full paper includes an ablative exploration of the parameter space including billboard resolution, noise variance and cut-on that due to space limitations cannot be included here. Table 3 is a coarse summary of results.

`cut-off`=0.60, and `cut-on`=28m. A multi-objective maneuver is satisfied when every spatial objective m is reached within a threshold of 1 meter.

The "cut the corner" maneuver shown in Figure 6 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 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.

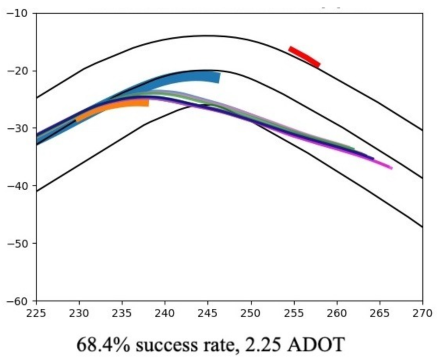


Figure 6: "Cut the corner" multi-target perturbation. Billboard is shown in red, original trajectory in thick blue, collection sequence in thick orange, and test trajectories are thin multicolored lines.

3.4 Natural Adversarial Patch Generation

3.4.1 Proposed Technical Approach

Section 3.3 demonstrates that adversarial patches can be successfully designed to consistently enact system-level failures. However, DeepManeuver adversarial patches do not attempt to blend into the surrounding environment, they are dissimilar to billboards in the DNN training dataset, and they appear unnatural to humans, making them easy to spot and dismantle, less likely to resemble deployment conditions, and inconsistent with the training distribution.

The proposed approach seeks to address the problem of adversarial attacks appearing out-of-place, semantically inconsistent, or insufficiently stealthy [100] compared with $\mathcal{N}$'s training distribution X_{train}, and out of distribution with the set of all semantically realistic inputs $X_{\mathbb{R}} \subset X$. We want to find some function $T_{orig:pert}$ to apply to a spatiotemporal series of inputs to $\mathcal{N}$ that maximizes the difference between $\hat{\psi}_{orig}$ and $\hat{\psi}_{pert}$ and conforms to one of two typologies: (1) $T_{orig:pert}$ uses only features in X_{train} to maximize this difference; or (2) $T_{orig:pert}$ maximizes this difference by generating perturbations that may be outside X_{train} but within $X_{\mathbb{R}}$.

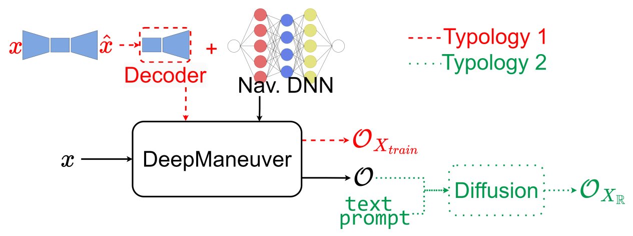


Figure 7: Overview of natural adversarial example generation approaches for 2 typologies.

This work will expand on my previous work to produce two distinct typologies of natural adversarial examples. Figure 7 shows the pipeline for each of these typologies as they interface with the DeepManeuver perturbation generation loop. The first typology will employ a decoder trained on X_{train} using the PreFixer loss function. The decoder is attached to the beginning of the navigation DNN so that, during perturbation generation, the decoder and embedding dictionary can minimize the distance between the perturbation and similar features in X_{train}, producing an adversarial patch $pert_{X_{train}}$. The second typology of techniques will function as a post-processing step for DeepManeuver patches. The DeepManeuver-generated patch $pert$ undergoes a style transfer using a text prompt and an image-to-image stable diffusion model, to produce a patch in the set of realistic images.

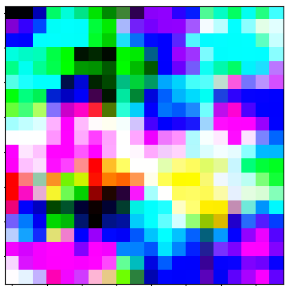
Typology 1 explores methods to generate examples that are constrained within X_{train}. My general approach expands on PreFixer and DeepManeuver. First, I will examine whether a decoder trained on X_{train} could be used to bring adversarial examples back in distribution with the DNN itself . I will investigate training the decoder with several architectures, X_{train} subsets, and inference data from the navigation DNN to determine what X_{train} features are preserved by the decoder. The backpropagation during adversarial test generation will rely on the code base used in the Perturbation Generation step of DeepManeuver. Thus the expectation is that the resulting patch-attacked images are in distribution and preserve the output of $\mathcal{N}$.

Typology 2 will explore extensions to the patches generated by DeepManeuver that modify the appearance of existing patches with high adversarial strength to constrain them to $X_{\mathbb{R}}$. First, I will reuse patches that have demonstrated a high success rate in the DeepManeuver study, i.e. a patch that has led to a high percentage of crashes per test run in simulation (see Section 3.3 and Table 3). I will then investigate whether image-to-image stable diffusion models are sensitive enough to critical aspects important to perturbation strength, e.g. the structure and color of the patch image, such that they can preserve perturbation strength in the generated image while still making it appear like a natural image one might see on a roadside billboard.

3.4.2 Preliminary Work

Developing the proposed techniques requires an $\mathcal{N}$ that can drive $\mathcal{S}$ within $\mathcal{E}$, an available X_{train}, and a simulator into which adversarial patches can be injected to test $T_{orig:pert}(X_{\mathbb{R}})$'s effect on $\mathcal{S}$. It also requires a patch generation loss function that can incorporate predictions of $\mathcal{N}$ and the state of $\mathcal{S}$. DeepManeuver provides significant parts of this infrastructure. First, it has a large training dataset of 150,000 images, a sophisticated simulation framework and codebase to use that simulator, several DNNs that can traverse a variety of roads in that simulator, and a stable of adversarial patches that can mislead those DNNs. These aspects can be reused by both typologies.

Additionally, my first typology of techniques can reuse parts of PreFixer. Typology 1 techniques seek to investigate naturalness by leveraging known features and their effect on prediction accuracy by applying the inputs of my previous work on feature deformation mitigation in the presence of sensor hardware migrations. Thus the technique can reuse and refine the generation strategy of the PreFixer work in the construction of patch attacks that are within distribution of the original training set of the navigation DNN and preserve the perturbation strength of the patch attack. Reduced attack strength is a typical result of attempting to generate more natural perturbations that conform to the features present in X_{train} [97]. This proposed technique does not promise to necessarily retain the full strength of perturbations but to better leverage known features in perturbation generation.


(a) DeepManeuver 20 × 20 degrees-of-freedom patch perturbation.

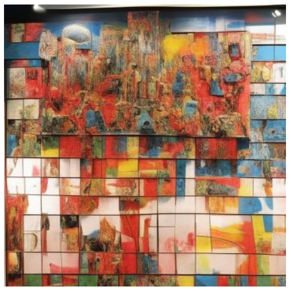

(b) Stable Diffusion of Figure 8a with prompt "museum, advertisement, detailed, colorful, artistic, modern art".

Figure 8: Example natural adversarial perturbation using stable diffusion method.

The second typology of techniques will investigate re-utilizing patches generated in the DeepManeuver work with a high success rate in order to "naturalize" them and explore features in the long tail of driving scenarios. This also allows the system to be tested for unanticipated inputs not captured by the training or validation sets. Figure 8 shows an example high-strength DeepManeuver adversarial patch for application to a roadside billboard, and a style transformation using a stable diffusion image-to-image pretrained model from huggingface [89]. The structural similarity is high but Figure 8b is significantly more realistic and semantically meaningful in appearance. This image would replace the less natural perturbation on the billboard in Figure 1a. The wide range of possible prompt tuning (e.g. the distance between the input image and output image) that these stable diffusion models can take in, the variety of features they can reproduce, and use of tune-able denoising shows promise for high strength but natural perturbations.

3.4.3 Study and Expected Value

This study aims to answer the following research questions:

RQ1) *How effective are in-distribution features in producing high-strength, stealthy adversarial tests?* To answer this I will assess the relationship between perturbation strength and naturalness constraints for constructing adversarial patches from features in X_{train}. [4]

RQ2) *How much of the perturbation strength can stable diffusion models preserve in realistically transformed pre-generated adversarial patches?* To answer this question, I compare the perturbation strength-naturalness tradeoff for high-strength perturbations generated by DeepManeuver and then realistically transformed using image-to-image diffusion.

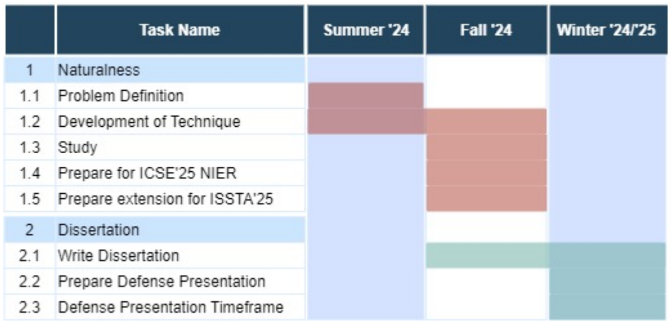
The exploration of RQ1 will use generative model architectures to encode X_{train} and its correlation to $\mathcal{N}$ predictions $\hat{\psi}$, similar to the approach of PreFixer. It will also break down X_{train} into subsets according to the location of each run and semantic segmentation to determine whether features common to these subsets can be successfully reproduced by the decoder and to what extent the training set can control the features reproduced by the decoder. The study will also use the decoder and latent space distance similar to [23]. This study will better explore the range of known features that the DNN misinterprets in certain contexts which could likely appear in these contexts under deployment because they are present in the training data.

The RQ2 techniques will search the input space of various stable diffusion pretrained models to apply a realistic transformation to existing patches from DeepManeuver that show a high success rate. Figure 8 shows such an example transformation. Figure 8a is input to a stable diffusion model and prompted according to certain desired characteristics. The resulting patch (Figure 8b) and prompt can be automatically fuzzed and refined such that the perturbation strength is retained according to a validation set that samples test conditions. This technique will be applied after patch generation and will refine the input prompts along with other parameters such as denoising strength to investigate how stable diffusion networks can be used to preserve critical aspects of the patch while introducing naturalness. This RQ will hopefully test the long tail as well as use the looser constraints of $X_{\mathbb{R}}$ (rather than only features in X_{train}) to better preserve the adversarial strength of the perturbation, a common pitfall of natural adversarial examples.

For both techniques, the generated patches will be tested in simulation and ADS state and trajectory will be compared to DeepManeuver for angle error and system-level failures. The generated patches will also be judged for image naturalness. While some previous work relies on large-scale human studies to compile natural image benchmarks [37], others [21, 100] use single-image comparison metrics [26, 39, 95]. Additionally, similarity metrics for videos [71, 93] can be used to compare to these spatiotemporal series of images to which the adversarial patches will be applied. All of these can use the unperturbed images as a baseline comparison. Dataset comparison metrics [55, 56, 58] could also be useful for comparing spatiotemporal series against X_{train}. I also propose a small authors-only human assessment of a subset of generated examples from each RQ compared against the original patches produced by DeepManeuver. This could be supplemented by an LLM model which can adjudicate naturalness at scale.

4 Work Plan and Timeline

I plan to develop the naturalness approach detailed in Section 3.4 by Fall 2024 and defend my dissertation in Winter 2024/2025. Figure 9 details my timeline to tackle the proposed work and complete the steps of my dissertation. I plan to submit a first iteration of this work to ICSE'25 NIER and the full technique and study to either ISSTA'25 or a high-impact software engineering journal.



	Task Name	Summer '24	Fall '24	Winter '24/'25
1	Naturalness			
1.1	Problem Definition	🟥		
1.2	Development of Technique	🟥	🟥	
1.3	Study		🟥	
1.4	Prepare for ICSE'25 NIER		🟥	
1.5	Prepare extension for ISSTA'25		🟥	
2	Dissertation			
2.1	Write Dissertation		🟩	🟩
2.2	Prepare Defense Presentation			🟩
2.3	Defense Presentation Timeframe			🟩

Figure 9: A Gantt chart outlining my proposed work plan.

[4]Recall that perturbation strength is the ability of the perturbation to mislead $\mathcal{N}$ under test.

Acknowledgment

This material is based in part upon work supported by the National Science Foundation (NSF) Award #1924777 and Air Force Office of Scientific Research (AFOSR) Award #FA9550-21-1-0164.

A Glossary

Term	**Symbol**	**Definition**
DNN	$\mathcal{N}$	Trained neural network used to navigate $\mathcal{S}$ through a deployment environment
ADS	$\mathcal{S}$	Autonomous driving system
system state space	A	the set of states that $\mathcal{S}$ can occupy, including failure states
system state	$a_t \in A$	$\mathcal{S}$'s current state at timestep t
environment	$e \in \mathcal{E}$	An environment within which $\mathcal{S}$ can be safely deployed
control signal	$\hat{\psi}_t$	Output of $\mathcal{N}$ and input to $\mathcal{S}$ for actuation
actuation signal	ψ_t	the value upon which $\mathcal{S}$ can actuate, given its current state a_t
sensor	c	a perception sensor used by $\mathcal{S}$ to navigate $\mathcal{E}$
input distribution	X	the set of possible inputs to $\mathcal{N}$, parameterized by all the values the sensor reading can take
real input distribution	$X_{\mathbb{R}}$	the set of realistic inputs to $\mathcal{N}$, parameterized by intrinsic and extrinsic parameters of c and current e
input distribution shift	$T_{a:b}$	A function applied to $\mathcal{N}$'s input distribution X_a to produce a new non-identical input distribution X_b

References

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

[2] Abdelrahman Abdelhamed et al. A high-quality denoising dataset for smartphone cameras. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1692–1700, 2018.

[3] Akshay Agarwal, Nalini Ratha, Mayank Vatsa, and Richa Singh. Exploring robustness connection between artificial and natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 179–186, June 2022.

[4] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[5] Aws Albarghouthi. Introduction to neural network verification. *Found. Trends Program. Lang.*, 7(1–2):1–157, dec 2021.

[6] Jordan T. Ash and Ryan P. Adams. On the difficulty of warm-starting neural network training. *CoRR*, abs/1910.08475, 2019.

[7] Dor Bank et al. Autoencoders. *CoRR*, abs/2003.05991, 2020.

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

[9] Mike Bedigan. Tesla owner says car in 'full self-driving mode' failed to detect a moving train. *The Independent*.

[10] Raja Ben Abdessalem, Annibale Panichella, Shiva Nejati, Lionel C. Briand, and Thomas Stifter. Testing autonomous cars for feature interaction failures using many-objective search. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 143–154, 2018.

[11] Mariusz Bojarski et al. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

[12] Adith Boloor, Xin He, Christopher Gill, Yevgeniy Vorobeychik, and Xuan Zhang. Simple physical adversarial examples against end-to-end autonomous driving models. In *2019 IEEE International Conference on Embedded Software and Systems (ICESS)*, pages 1–7, 2019.

[13] Hanqun Cao, Cheng Tan, Zhangyang Gao, Yilun Xu, Guangyong Chen, Pheng-Ann Heng, and Stan Z. Li. A survey on generative diffusion models. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):2814–2830, 2024.

[14] Yulong Cao, Chaowei Xiao, Dawei Yang, Jing Fang, Ruigang Yang, Mingyan Liu, and Bo Li. Adversarial objects against lidar-based autonomous driving systems. *arXiv preprint arXiv:1907.05418*, 2019.

[15] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.

[16] Alexandra Carlson et al. Modeling camera effects to improve visual learning from synthetic data. In *European Conference on Computer Vision Workshops*, September 2018.

[17] Alexandra Carlson et al. Sensor transfer: Learning optimal sensor effect image augmentation for sim-to-real domain adaptation. *IEEE Robotics and Automation Letters*, 4(3):2431–2438, 2019.

[18] Rory Cellan-Jones. Uber's self-driving operator charged over fatal crash. *BBC*.

[19] Eric R. Chan et al. Generative novel view synthesis with 3d-aware diffusion models. In *IEEE/CVF International Conference on Computer Vision*, pages 4217–4229, October 2023.

[20] Ming-Fang Chang et al. Argoverse: 3d tracking and forecasting with rich maps, 2019.

[21] Xinquan Chen, Xitong Gao, Juanjuan Zhao, Kejiang Ye, and Cheng-Zhong Xu. Advdiffuser: Natural adversarial example synthesis with diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4562–4572, October 2023.

[22] Riccardo Corvi, Davide Cozzolino, Giada Zingarini, Giovanni Poggi, Koki Nagano, and Luisa Verdoliva. On the detection of synthetic images generated by diffusion models. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

[23] Swaroopa Dola, Matthew B. Dwyer, and Mary Lou 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*, pages 226–237. IEEE, 2021.

[24] Swaroopa Dola, Matthew B. Dwyer, and Mary Lou Soffa. Input distribution coverage: Measuring feature interaction adequacy in neural network testing. *ACM Trans. Softw. Eng. Methodol.*, 32(3), apr 2023.

[25] Swaroopa Dola, Rory McDaniel, Matthew B. Dwyer, and Mary Lou Soffa. Cit4dnn: Generating diverse and rare inputs for neural networks using latent space combinatorial testing. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24, New York, NY, USA, 2024. Association for Computing Machinery.

[26] Richard Dosselmann and Xue Dong Yang. Existing and emerging image quality metrics. In *Canadian Conference on Electrical and Computer Engineering, 2005.*, pages 1906–1913. IEEE, 2005.

[27] Ranjie Duan, Xingjun Ma, Yisen Wang, James Bailey, A. K. Qin, and Yun 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.

[28] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. *CoRR*, abs/1707.08945, 2017.

[29] Alessio Gambi, Marc Mueller, and Gordon 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*, ISSTA 2019, page 318–328, New York, NY, USA, 2019. Association for Computing Machinery.

[30] Harshvardhan GM, Mahendra Kumar Gourisaria, Manjusha Pandey, and Siddharth Swarup Rautaray. A comprehensive survey and analysis of generative models in machine learning. *Comput. Sci. Rev.*, 38(C), nov 2020.

[31] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[32] Alison Griswold. The self-driving cars wearing a cone of shame. *Slate*.

[33] Pia Hanfeld, Marina M. C. Höhne, Michael Bussmann, and Wolfgang Hönig. Flying adversarial patches: Manipulating the behavior of deep learning-based autonomous multirotors, 2023.

[34] Fitash Ul Haq, Donghwan Shin, Shiva Nejati, and Lionel 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)*, pages 85–95, 2020.

[35] Kaiming He et al. Convolutional neural networks at constrained time cost. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[36] Lei He et al. Learning depth from single images with deep neural network embedding focal length. *CoRR*, abs/1803.10039, 2018.

[37] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *CoRR*, abs/1907.07174, 2019.

[38] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239, 2020.

[39] Alain Horé and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, 2010.

[40] Hossein Hosseini and Radha Poovendran. Semantic adversarial examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[41] Chris Isidore and Peter Valdes-Dapena. Tesla is under investigation because its cars keep hitting emergency vehicles. *CNN*.

[42] Yuzhu Ji, Haijun Zhang, Zhao Zhang, and Ming Liu. Cnn-based encoder-decoder networks for salient object detection: A comprehensive review and recent advances. *Information Sciences*, 546:835–857, 2021.

[43] Oğuzhan Fatih Kar et al. 3d common corruptions and data augmentation, 2022.

[44] Zelun Kong, Junfeng Guo, Ang Li, and Cong Liu. Physgan: Generating physical-world-resilient adversarial examples for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14242–14251, 2020.

[45] Matja Kukar et al. Cost-sensitive learning with neural networks. In *European Conference on Artificial Intelligence*, 1998.

[46] Tim Levin. Tesla's full self-driving tech keeps getting fooled by the moon, billboards, and burger king signs. *Business Insider*.

[47] Pengzhi Li et al. A comprehensive survey on design and application of autoencoder in deep learning. *Applied Soft Computing*, 138:110176, 2023.

[48] Haokun Liu et al. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning, 2022.

[49] André Luckow et al. Deep learning in the automotive industry: Applications and tools. In *IEEE International Conference on Big Data*, pages 3759–3768, 2016.

[50] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[51] Rafid Mahmood et al. Optimizing data collection for machine learning. *Advances in Neural Information Processing Systems*, 35:29915–29928, 2022.

[52] Spyros Makridakis. The forthcoming artificial intelligence (ai) revolution: Its impact on society and firms. *Futures*, 90:46–60, 2017.

[53] Dominic Masters et al. Revisiting small batch training for deep neural networks. *CoRR*, abs/1804.07612, 2018.

[54] Enrico Meloni, Matteo Tiezzi, Luca Pasqualini, Marco Gori, and Stefano Melacci. Messing up 3d virtual environments: Transferable adversarial 3d objects. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1–8. IEEE, 2021.

[55] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. No-reference image quality assessment in the spatial domain. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 21(12):4695—4708, December 2012.

[56] Anish Mittal, Rajiv Soundararajan, and Alan C. Bovik. Making a “completely blind” image quality analyzer. *IEEE Signal Processing Letters*, 20(3):209–212, 2013.

[57] Jian Nie, Jun Yan, Huilin Yin, Lei Ren, and Qian Meng. A multimodality fusion deep neural network and safety test strategy for intelligent vehicles. *IEEE Transactions on Intelligent Vehicles*, 6(2):310–322, 2021.

[58] Artem Obukhov and Mikhail Krasnyanskiy. Quality assessment method for gan based on modified metrics inception score and fréchet inception distance. In *Software Engineering Perspectives in Intelligent Systems: Proceedings of 4th Computational Methods in Systems and Software 2020, Vol. 1 4*, pages 102–114. Springer, 2020.

[59] Bruno A. Olshausen et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

[60] Liyuan Pan et al. Bringing a blurry frame alive at high frame-rate with an event camera. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[61] Naman Patel, Prashanth Krishnamurthy, Siddharth Garg, and Farshad 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)*, pages 5916–5921, 2019.

[62] Svetlana Pavlitskaya, Sefa Ünver, and J. Marius 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)*, pages 1–8, 2020.

[63] Anibal Pedraza, Oscar Deniz, and Gloria Bueno. Really natural adversarial examples. *International Journal of Machine Learning and Cybernetics*, 13(4):1065–1077, 2022.

[64] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. *Commun. ACM*, 62(11):137–145, October 2019.

[65] Zi Peng et al. A first look at the integration of machine learning models in complex autonomous driving systems: A case study on apollo. In *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, page 1240–1250, 2020.

[66] Lukas Platinsky et al. Quantity over quality: Training an av motion planner with large scale commodity vision data, 2022.

[67] Jesse Read et al. Classifier chains: A review and perspectives. *CoRR*, abs/1912.13405, 2019.

[68] Vincenzo Riccio and Paolo Tonella. Model-based exploration of the frontier of behaviours for deep learning system testing. *CoRR*, abs/2007.02787, 2020.

[69] David Josué Barrientos Rojas, Bruno José Torres Fernandes, and Sergio Murilo Maciel Fernandes. A review on image inpainting techniques and datasets. In *2020 33rd SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*, pages 240–247. IEEE, 2020.

[70] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *CoRR*, abs/2112.10752, 2021.

[71] Kalpana Seshadrinathan and Alan Conrad Bovik. Motion tuned spatio-temporal quality assessment of natural videos. *IEEE Transactions on Image Processing*, 19(2):335–350, 2010.

[72] Himanshu Sharma. A survey on image encoders and language models for image captioning. *IOP Conference Series: Materials Science and Engineering*, 1116(1):012118, apr 2021.

[73] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[74] Jian Sun, Wenfei Cao, Zongben Xu, and Jean Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[75] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[76] Ardi Tampuu et al. A survey of end-to-end driving: Architectures and training methods. *IEEE Transactions on Neural Networks and Learning Systems*, 33(4):1364–1384, 2022.

[77] TeslaDriver.net. Tesla ap1 vs ap2 vs ap3 – difference between autopilot versions. `https://tesladriver.net/tesla-ap1-vs-ap2-vs-ap3-difference-between-autopilot-versions/`, 2022.

[78] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 303–314, 2018.

[79] Felipe Toledo, David Shriver, Sebastian Elbaum, and Matthew B Dwyer. Distribution models for falsification and verification of dnns. *Automated Software Engineering*, 2021.

[80] Cumhur Erkan Tuncali, Georgios Fainekos, Hisahiro Ito, and James Kapinski. Simulation-based adversarial test generation for autonomous vehicles with machine learning components. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1555–1562, 2018.

[81] Cumhur Erkan Tuncali, Georgios Fainekos, Danil Prokhorov, Hisahiro Ito, and James Kapinski. Requirements-driven test generation for autonomous vehicles with machine learning components. *IEEE Transactions on Intelligent Vehicles*, 5(2):265–280, 2020.

[82] Aäron van den Oord et al. Neural discrete representation learning. *CoRR*, abs/1711.00937, 2017.

[83] Meriel von Stein and Sebastian 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*, ASE '22. Association for Computing Machinery, 2022.

[84] Meriel von Stein, David Shriver, and Sebastian Elbaum. Deepmaneuver: Adversarial test generation for trajectory manipulation of autonomous vehicles. *IEEE Transactions on Software Engineering*, 49(10):4496–4509, 2023.

[85] Meriel von Stein, Hongning Wang, and Sebastian Elbaum. Automated generation of transformations to mitigate sensor hardware migration in ads. *IEEE Robotics and Automation Letters*, 9(7):6480–6487, 2024.

[86] Waymo. The waymo driver's training regimen: How structured testing prepares our self-driving technology for the real world. `https://waymo.com/blog/2020/09/the-waymo-drivers-training-regime.html`, 2020.

[87] Waymo. Waymo open dataset. `https://waymo.com/open/data/perception/`, 2022.

[88] Benjamin Wilson et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *NeurIPS*, 2021.

[89] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019.

[90] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon 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*, ISSTA 2019, page 146–157, New York, NY, USA, 2019. Association for Computing Machinery.

[91] Jason Yosinski et al. Advances in neural information processing systems. volume 27, 2014.

[92] Khobaib Zaamout et al. Improving neural networks classification through chaining. In *International Conference on Artificial Neural Networks and Machine Learning*, page 288–295. Springer-Verlag, 2012.

[93] Han Zhang, Jizheng Xu, and Li Song. Video multimethod assessment fusion based rate-distortion optimization for versatile video coding. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2064–2068, 2021.

[94] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz 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)*, pages 132–142, 2018.

[95] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[96] Xudong Zhang, Yan Cai, and Zijiang Yang. A study on testing autonomous driving systems. In *IEEE International Conference on Software Quality, Reliability and Security Companion*, pages 241–244, 2020.

[97] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. *CoRR*, abs/1710.11342, 2017.

[98] Husheng Zhou, Wei Li, Zelun Kong, Junfeng Guo, Yuqun Zhang, Bei Yu, Lingming Zhang, and Cong Liu. Deepbillboard: Systematic physical-world testing of autonomous driving systems. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 347–358, 2020.

[99] Husheng Zhou, Wei Li, Zelun Kong, Junfeng Guo, Yuqun Zhang, Bei Yu, Lingming Zhang, and Cong Liu. Deepbillboard: Systematic physical-world testing of autonomous driving systems. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 347–358, 2020.

[100] Zheng Zhou, Hongbo Zhao, Ju Liu, Qiaosheng Zhang, Liwei Geng, Shuchang Lyu, and Wenquan Feng. Mvpatch: More vivid patch for adversarial camouflaged attacks on object detectors in the physical world, 2024.

[101] Zhi Quan Zhou and Liqun Sun. Metamorphic testing of driverless cars. *Commun. ACM*, 62(3):61–67, feb 2019.