2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops

AVFI: Fault Injection for Autonomous Vehicles

Saurabh Jha[§], Subho S. Banerjee[§], James Cyriac[†], Zbigniew T. Kalbarczyk[†] and Ravishankar K. Iyer[†§]
[§]Department of Computer Science, [†]Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign, Urbana IL 61801, USA.

I. Introduction

Autonomous vehicle (AV) technology is rapidly becoming a reality on U.S. roads, offering the promise of improvements in traffic management, safety, and the comfort and efficiency of vehicular travel. With this increasing popularity and ubiquitous deployment, resilience has become a critical requirement for public acceptance and adoption. Recent studies into the resilience of AVs have shown that though the AV systems are improving over time, they have not reached human levels of automation [1]. Prior work in this area has studied the safety and resilience of individual components of the AV system (e.g., testing of neural networks powering the perception function [2], [3]). However, methods for holistic end-to-end resilience assessment of AV systems are still non existent.

This paper presents AVFI (the Autonomous Vehicle Fault Injector), an important step towards constructing a methodology for end-to-end resilience assessment of AV systems using fault injection. The tool empirically validates the robustness of an AV system by introducing faults to test AV resilience in situations that might otherwise be rarely tested. AVFI leverages a state-of-the-art AV simulation framework presented in [4], and can perform fault injections in sensor inputs (e.g., following camera or LIDAR fault models), in neural networks controlling the motion of the AV (e.g., to identify susceptibility to random and adversarial noise in the training procedure), and in hardware/software components (e.g., transient/permanent faults in processing fabric). The AVFI approach uniquely quantifies meaningful domain-specific failure metrics, e.g., number of traffic violations per kilometer driven, mission success rate and time to traffic violation. By using those metrics to evaluate safety, we demonstrate their comprehensive value. AVFI achieves those goals so by simulating real worlds, describing behavior of cars and pedestrians moving in that world, and evaluating resilience metrics. Overall, we believe that AVFI can positively influence the development and holistic testing of AV systems.

Our preliminary results validate AVFI's ability to introduce faults that lead to traffic violations. Those results are supported by failure characterization studies of AVs in the real world [1]. Our findings reiterate the need for experimentation and analysis of failure models and modes for AVs.

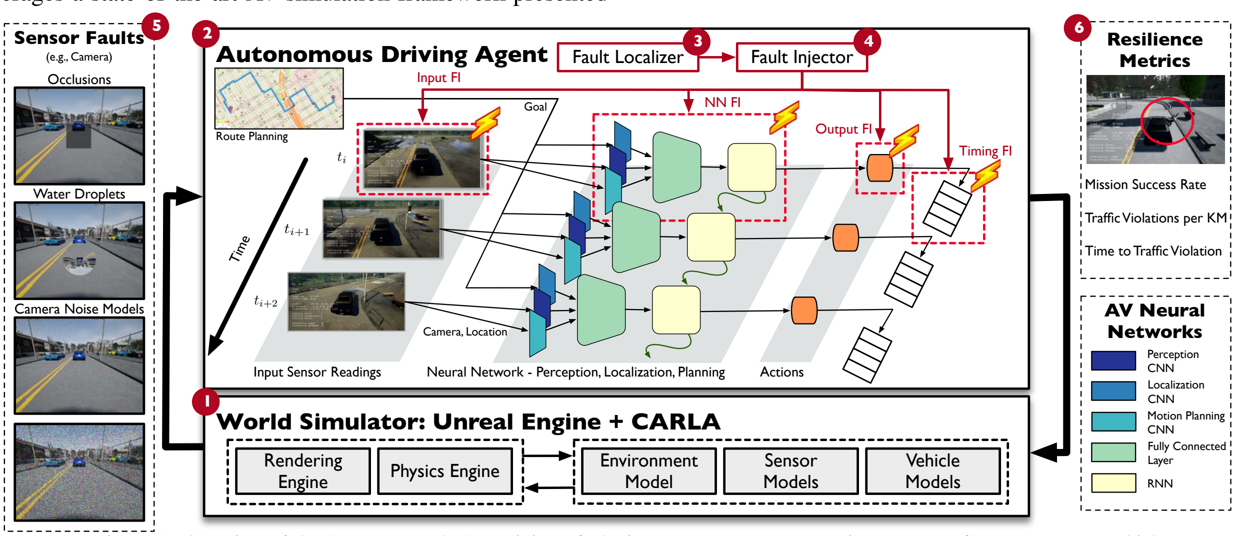

Figure 1. Overview of the AVFI approach: AVFI injects faults into sensor-compute-actuation systems of an autonomous vehicle.

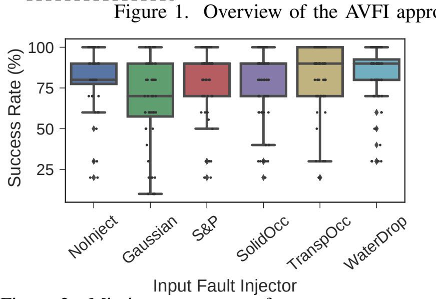

Figure 2. Mission success rate for an autonomous vehicle with different input fault injectors.

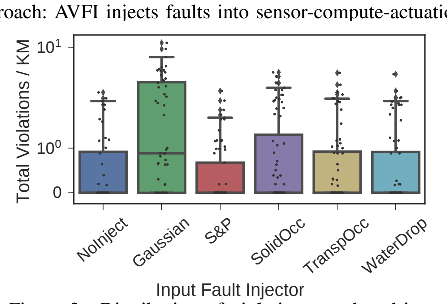

Figure 3. Distribution of violations per km driven with different input fault injectors.

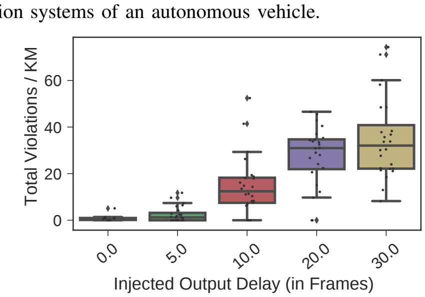

Figure 4. Distribution of violations per km with increasing output delay between ADA and actuation.

2325-6664/18/$31.00 ©2018 IEEE
DOI 10.1109/DSN-W.2018.00027

II. AVFI: Autonomous Vehicle Fault Injector

Our approach (see Fig. 1) uses (a) CARLA [4] as an urban driving simulator, (b) the approach described in [5] as an ADA (Autonomous Driving Agent) to control the AV and (c) AVFI as fault injection-based assessment engine for the ADA. AVFI has inbuilt fault models and provides methods for statistical analysis of traffic violations.

Autonomous Driving Agent & World Simulator. CARLA is an open urban driving simulator (1[1]). CARLA operates by running two components, the server and the client. The server is responsible for generating the virtual urban environments, and the client functions as an ADA. The server leverages Unreal Engine (popularly used in video games) as its rendering and physics engine. CARLA has an inbuilt library of urban layouts, buildings, pedestrians, vehicles, and weather conditions (e.g., sunny, rainy, and foggy) that can be used to simulate an urban environment. Further, it provides a variety of sensors (e.g., camera, GPS, LIDAR) to use in AV simulations.

The ADA uses the approach described in [5] as the controller; that in turn, uses an imitation learning-based convolution neural network (IL-CNN) for perception, planning, and localization (2). In our test environment, the client is fed from a forward-facing RGB camera sensor on the hood of the AV. The server sends sensor data, along with other measurements of the car (e.g., speed, location) to the client. The controller is responsible for perception of sensor inputs and for producing an action that describes the behavior of the AV. Its decisions are then sent from the client to the server, which applies those commands to the AV's actuators. In that way, an AV can complete missions, i.e., navigating between way points in the simulated world.

Fault Models and Injector. AVFI runs fault injection campaigns in two steps: (a) selecting the location of faults (3) (e.g., choosing specific neurons and layers in the IL-CNN) and (b) injecting the faults into the chosen locations using the fault models mentioned below (4). Broadly, AVFI can inject the following four classes of faults into the ADA.

- *Data Faults*: AVFI injects data faults by manipulating sensor measurements (such as camera images, LIDAR, and GPS) or world measurements (such as car spee or weather type) taken by the AV system. In the real world, sensor inputs can change because of (a) faulty sensors, (b) changes in the external environment (such as fog or rain), and (c) unseen perturbations of images (such as broken road sign posts). For example, AVFI intercepts the RGB camera sensor data from the server, modifies the image according to a sensor-specific fault model (5), and then forwards it to the IL-CNN.
- *Hardware Faults*: AVFI injects hardware faults by injecting single-bit, multiple-bit, and stuck-at faults in the hardware components of the autonomous systems, such as processors, sensors, software, and communication networks. For example, AVFI can intercept and corrupt a control command from the IL-CNN and then forward it to the server.
- *Timing Faults*: AVFI injects timing faults into the communication paths of the network, resulting in (a) delays in flow of data from one component of the AV system to another, (b) loss of data, or (c) out-of-order delivery of the data packets. For example, AVFI pauses the output of IL-CNN for k frames and either replays or drops the outputs.
- *Machine Learning Faults*: Errors in the machine learning models (such as neural networks) during training or at runtime will lead to prediction errors. AVFI injects faults into the neural network by adding noise into the parameters of the machine learning model (e.g., weights of the neural network), which is modeled on real-world hardware failures.

Resilience Assessment. AVFI reports the following resilience metrics (6).

- *Mission Success Rate (MSR)* is the percentage of times that the autonomous agent was able to complete a navigation mission in a fixed amount of time. Higher MSR values are representative of higher resiliency.
- *Traffic Violations Per KM (VPK)* is the number of traffic violations (including lane violations, driving on the curb, and collisions with pedestrians, cars, and other objects on the streets) per kilometer driven in a fault injection campaign. Lower VPK values are representative of higher resiliency.
- *Accidents Per KM (APK)* is the number of accidents (i.e., collisions with pedestrians/cars/etc.) per kilometer driven in a fault injection campaign.
- *Time to Traffic Violation (TTV)* is the time between a fault injection and its manifestation as a traffic violation. Higher values of TTV imply that the system has more time to detect and correct its state to avoid traffic violations.

[1] * refers to annotations in Fig. 1.

III. Preliminary Results

Initial experiments using AVFI on the IL-CNN-based ADA presented in [4] have shown promising results and point to the need for further experimentation and analysis of failure models and modes for deep-learning-based ADA.

Fig. 2 shows the increase in variance of the mission success rate with varying sensor fault models across multiple test scenarios. Fig. 3 shows a similar increase in variability of traffic violations per km driven across a range of sensor fault injectors. The variability suggests that the overall decrease in success rate is correlated to the increase in traffic violations per km. Fig. 4 shows a significant increase in the number of traffic violations per km with the introduction of delays between the generation of output from the agent's neural network and its actuation in the world model (i.e., using AVFI's timing fault injector). Our simulation environment is configured to run at 15 frames per second; hence, a delay of 30 frames in Fig. 4 corresponds to an overall delay of a mere 2 s between decision and actuation. Our results and those in [1] (e.g., data from Nissan), indicate a need to explore the real-time nature and constraints associated with the AV.

References

[1] S. S. Banerjee *et al.*, "Hands off the wheel in autonomous vehicles? A systems perspective on over a million miles of field data," in *Proc. of the 48th Annual IEEE/IFIP International Conf. on Dependable Systems and Networks (DSN)*, June 2018.

[2] G. Li *et al.*, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proc. International Conf. for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 8:1–8:12.

[3] K. Pei *et al.*, "DeepXplore: Automated whitebox testing of deep learning systems," in *Proc. of the 26th Symposium on Operating Systems Principles*, 2017, pp. 1–18.

[4] A. Dosovitskiy *et al.*, "CARLA: An open urban driving simulator," in *Proc. of the 1st Annual Conf. on Robot Learning*, 2017, pp. 1–16.

[5] F. Codevilla *et al.*, "End-to-end driving via conditional imitation learning," in *Proc. of International Conf. on Robotics and Automation (ICRA)*, 2018.