# D2MON: Detecting and Mitigating Real-Time Safety Violations in Autonomous Driving Systems

Bohan Zhang, Yafan Huang, Rachael Chen, Guanpeng Li
Computer Science Department, University of Iowa
Iowa City, IA, USA

***Abstract*—This paper proposes D2MON, a data-driven real-time safety monitor, to detect and mitigate safety violations of an autonomous vehicle (AV). The key insight is that traffic situations that lead to AV safety violations fall into patterns and can be identified by learning from existing safety violations. Our approach is to use machine learning techniques to model the traffic behaviors that result in safety violations and detect their symptoms in advance before the actual crashes happen. If D2MON detects surroundings as dangerous, it will take safety actions to mitigate the safety violations so that the AV remains safe in the evolving traffic environment. Our steps are twofold: (1) We use software fuzzing and data augmentation techniques to generate efficient safety violation data for training our ML model. (2) We deploy the model as a plug-and-play module to the AV software, detecting and mitigating safety violations of the AV in runtime. Our evaluation demonstrates our proposed technique is effective in reducing over 99% of safety violations in an industry-level autonomous driving system, Baidu Apollo.**

***Index Terms*—Autonomous Vehicles, Safety-Critical Applications, Machine Learning, Software Fuzzing**

## I. INTRODUCTION

The development of autonomous vehicles (AVs) is recently at a booming speed, which attracts massive research attention from both academia and industrial companies such as Tesla and Waymo. However, current AV products still face many safety issues. For example, in the accident report of L2 driver assistance systems released by NHTSA in June 2022, 367 accidents caused by AVs happened from May 2021 to June 2022 [1], leaving extreme hazard on human lives and properties. Therefore, enabling AV safety has become a crucial task in the reliability research community.

Traditional methods to improve AV safety are through massive testing processes either on the road or in simulators. The search process involves AV under test in an evolving traffic scenarios and observes if the AV will reveal any safety violations [2]–[4]. However, these methods have two critical defects: (1) They are designed to minimize the risky scenarios that an AV may face and fail to guarantee safety when accidents are going to happen. (2) Deploying these methods is exceedingly labor-intensive and time-consuming. Specifically, in the AV testing phase, developers have to spend massive amounts of time finding safety violation cases and localizing the corresponding bugs in large code bases. Besides, such modifications to code may change the original internal logic and even introduce new bugs, incurring extra testing efforts.

To tackle these challenges, we propose a **D**ata-**D**riven AV safety **mon**itor D2MON which can automatically detect and mitigate safety violations of an AV without requiring developers to modify any code. Figure 1[1] explains the general idea of how D2MON cooperates with an AV in a cut-in scenario. D2MON evaluates real-time risks in the surrounding of the AV and hence determines its later actions. Once a nearby NPC (the vehicles on the road other than AV) tries to perform a dangerous action (e.g. cut-in), the safety monitor can detect current surroundings as "danger" and alerts AV to brake in the next time step to avoid such a safety violation. Besides, D2MON is a complete independent module and parallel to Autonomous Driving System (ADS). Such plug and play design principle significantly reduces the time cost compared with existing human-supervised solutions.

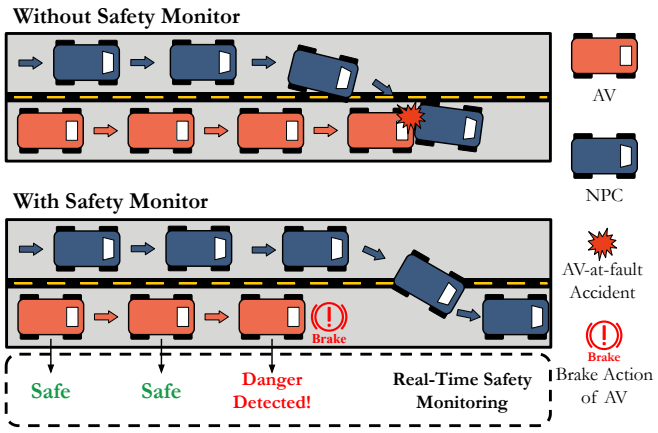



Fig. 1. Mitigating safety violations with AV safety monitor.

The contributions of this work are presented as follows:

- **We propose a data-driven AV safety monitor D2MON by combining software fuzzing and machine learning (ML) techniques.** The key insight of D2MON is that an AV can learn from past AV-at-fault accidents and then evaluate the risk of current surroundings. In light of this, we first extract features from those accidents, and then transfer these features to low-dimensional vectors. To process such vector data effectively, we propose a network structure with a sequence-to-sequence (Seq2Seq) model and several fully-connected layers. Seq2Seq is an encoder-decoder framework that takes series vectors as inputs and generates an output vector with similar

[1]We refer the definition of AV-at-fault accident to Section III.

length, while fully-connected layers can further enhance model's fitting capability. Software fuzzing technique is also adopted to localize AV-at-fault accidents (i.e. training data for ML model) with a low cost.

- **We design a flexible framework to create and deploy the safety monitor D2MON into an AV.** Creating the safety monitor D2MON can be conducted totally offline, which indicates this overhead is just a one time cost. Deploying D2MON technique just needs to link two independent components (safety monitor and mitigator) to an AV.
- **We conduct comprehensive experiments to evaluate the prediction accuracy and accident mitigation efficiency of D2MON.** We first analyze the prediction accuracy of the ML-based monitor, including the sensitivities of false positive (FP) and false negative (FN) to the user-defined threshold. We traverse the threshold from 0 to 1, finding an optimal value of this threshold. This threshold will be utilized in D2MON to judge whether the AV is in a dangerous state. Then, we evaluate the accident mitigation efficiency of D2MON. We evaluate two AVs with and without D2MON technology, respectively. In 500 trials, the AV without D2MON causes 313/500 AV-at-fault accidents, while the other AV with D2MON equipped causes only 3/500 AV-at-fault accidents.

The rest of this paper is organized as follows: In Section II, we introduce the background knowledge of this work. In Section III, we present the detailed design of our proposed D2MON. In Section IV, we evaluate this work from two perspectives: the prediction accuracy and accident mitigation efficiency. Finally, we conclude this work in the last section.

## II. BACKGROUND

In this section, we briefly introduce the background knowledge of AV safety, including the autonomous driving system, high-fidelity simulator, fuzzing technique, and sequential data processing algorithms.

### A. Autonomous Driving Systems

The autonomous driving system (ADS) is a decentralized and highly collaborative combination of all AV-related software and hardware [5]. AVs utilize autonomous driving system (ADS) technology to replace human driving [6]–[8]. A modern ADS infrastructure consists of a sensor layer and six modules [9], which can be listed as below.

**Sensor Layer** contains several sensor units such as IMU, GPS, camera, Radar, and Lidar. These sensors can provide raw data such as pictures, point clouds, GPS locations, etc. All the generated data will be sent to and processed by later modules.

**Localization Module** obtains the current position information of the vehicle by processing the coordinate system in the high-definition map, GPS information, and the point cloud provided by the Lidar, etc.

**Perception Module** processes the raw data through hardware such as cameras and Lidars carried by the vehicle itself, including data pre-processing (deep learning model) and post-processing. This module can obtain obstacle, lane line, and traffic light information around the vehicle.

**Prediction Module** obtains the information of obstacles around the vehicle through the perception module and the positioning module and predicts the running trajectory of these obstacles. Those trajectories will be sent to the planning module for calculation for avoiding potential obstacles.

**Routing Module** calculates the long-term travel route of the vehicle based on current vehicle position, a high-fidelity map, and the destination provided by user.

**Planning Module** navigates the short-term route, such as avoiding obstacles and following traffic lights, by obtaining the location information and other surrounding environment information provided by other modules.

**Control Module** is based on the instructions from Planning and Routing modules. It also connects the vehicle hardware, steering wheel, accelerator brake, and other city and county controls of the vehicle through the CAN bus.

Baidu Apollo [9], which is one of the most advanced ADSs in the industry [9]–[11], has gone through 7 iterations since its first official release in 2017, yet has implemented a relatively complete functionality. As Baidu Apollo has been commonly-used in this literature, we use its popular branch Apollo 3.5 in this work.

### B. LGSVL Simulator

LGSVL (**LG**-**S**ilicon **V**alley **L**ab) [12] is a real-time simulator (shown in Figure 2) based on Unity engines and completely simulates our experiments. It can simulate the cars, environments, and traffic participants in real-time. By linking this simulator and ADS through a bridge, LGSVL can return the information of AV driving status, such as speed, position, and steering angle. The ego car's action in LGSVL simulator can also be controlled by Python API. In this work, we first build the bridge between LGSVL and Apollo, then run the Python APIs provided by LGSVL to navigate ego car. The emergency braking function used in D2MON is also implemented on this API.


Fig. 2. Illustration of LGSVL simulator (left) and Baidu Apollo ADS (right).

### C. Software Fuzzing and AV-FUZZER

Originated from an operating system academia project [13], software fuzzing has been well studied and widely recognized in test case generation and software vulnerability detection

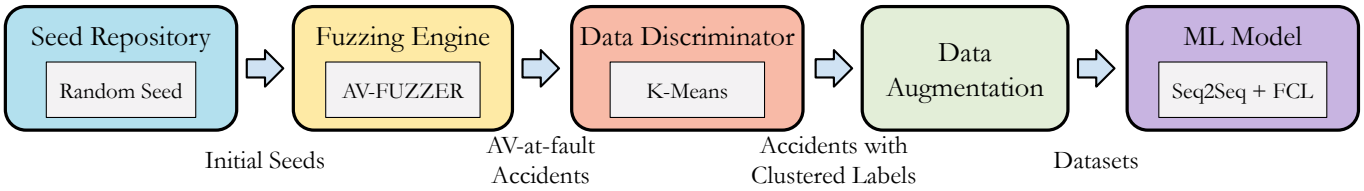



Fig. 3. D2MON creation workflow, where FCL in ML Model indicates fully-connected layers.

in the past years [14]–[18]. By utilizing searching algorithms such as genetic algorithm during the software testing phase, researchers can localize vulnerable cases and hence boost the application.

AV-FUZZER [4] is the first work to use software fuzzing technique in AV safety to find AV-at-fault accidents with high efficiency. AV-FUZZER is based on meta-heuristic algorithms and implements a genetic algorithm to find NPC control instructions that may lead to AV-at-fault accidents. The reasons we utilize AV-FUZZER to generate training data are threefold: (1) Parameters such as initial seed, number of NPCs, and initial locations of AV can be highly-customized in AV-FUZZER. (2) AV-FUZZER is implemented on LGSVL and Baidu Apollo, which are compatible with our testing environments. (3) Compared with random fuzzing technique, AV-FUZZER has a unique fitness function and mutation strategy that can significantly speed up finding the safety violation cases. As a result, we use AV-FUZZER as a black box to generate efficient training data for D2MON with a relatively low cost.

### D. Sequence Data Processing

There are many techniques to process sequence data such as moving-average and exponential smoothing. Among those methods, recurrent neural network (RNN) -based models [19] are the most popular methods since their temporal sequence structures can capture more latent information in series. As a branch of RNN-based models, sequence-to-sequence (Seq2Seq) [20] transfers one sequence into another sequence. It does so by use of RNNs (usually LSTM [21] or GRU [22]) as encoder and decoder. The encoder turns each input item into a corresponding hidden encode state, while the decoder reverses the process and turns this hidden state to sequence data again. Since our designed AV safety monitor predicts current surroundings by history information, usually time-series data, we utilize an LSTM-based Seq2Seq model as the main structure of D2MON.

## III. METHODOLOGY

We present the detailed design of D2MON in this section. In this work, we detect and analyze AV-at-fault accidents since we can only mitigate AV's behaviors and cannot control other traffic participants (i.e. NPCs). AV-at-fault accident is defined as those accidents when AV's head collides the NPC's rear bumper, in which AV bears the primary liability. Generally, this work can be divided into two parts: **D2MON Creation** and **D2MON Deployment**. The first part presents how an ML-based safety monitor is created, while the second part introduces how to integrate this safety monitor into a running AV and hence mitigate AV-at-fault accidents. Note that all data collection and model training in this section can be executed offline, which also indicates these processes are just one-time costs.

### A. D2MON Creation

Figure 3 shows a high-level workflow of the creation process of D2MON. Starting from selecting the original random seed, the selected seed will be set as the initial status and sent to AV-FUZZER to generate a certain number of AV-at-fault accidents. Then, a K-Means clustering algorithm is utilized to generate labels for those accidents. We also perform data augmentation for each type of clustered accident to enlarge the dataset with confidence. Finally, the generated dataset will be fed into our proposed ML model, which is also the core part of D2MON, for real-time reasoning.

*1) Seed Repository:* Our safety monitor starts with the random selection of seeds. Seed concept is recognized as the initial status of software fuzzing. Specifically, a seed usually contains surrounding NPC speeds and their turning commands. In random selection phase, for example, the speed range can be selected within 0∼30 miles/hour, and the turning command is randomly generated from 0 to 2, where 0 = stay in the same lane, 1 = turn left, and 2 = turn right).

*2) Fuzzing Engine:* The randomly selected seeds will be taken as input to the AV-FUZZER. We also need to configure the AV-FUZZER as the instruction such as the liability judgment, fitness function, etc. AV-FUZZER continuously mutates the given random seed through the genetic algorithm and continuously improves the fitness score for the scenario to find AV-at-fault accidents. Compared with other fuzzing techniques such as random fuzzing, AV-FUZZER can generate more AV-at-fault accidents with only a few time costs (within only 10 hours), which is also a one time cost.

*3) Data Discriminator:* Data discriminator is designed to label AV-at-fault accidents that are generated from AV-FUZZER, so that they can be used for ML model training. To discriminate the labels of data, we cluster all the AV-at-fault accidents by K-Means algorithm. For each cluster, we consider that this class is AV sensitive at fault accident.

*4) Data Augmentation:* After we complete the clustering, we perform data augmentation for each category (i.e. data with the same label) The purpose of this is to find as many NPC

trajectories that are similar to this cluster as possible, including seeds that can cause NPC at fault accident, or seeds that almost cause NPC at fault accident. The key observation is that AV-at-fault accidents are very sensitive to a nearby NPC speed. Thus, for each category, we randomly add or reduce the NPC speeds by ten percent and generate more raw data with very high probabilities ($> 70\%$) to cause accidents. All the raw data will be rechecked in the LGSVL simulator. We also record the NPC and EGO statues every 0.25 seconds, for speed, steering angle, and location. In this way, more convincing data can be generated and trained by the ML model so that the prediction accuracy can be improved.

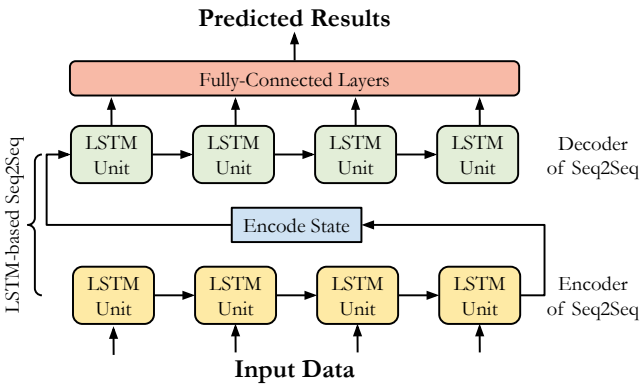



Fig. 4. The network structure of ML model, which is also the key component of AV safety monitor D2MON.

*5) ML Model:* The key component of AV safety monitor D2MON is to use the ML model to evaluate real-time surrounding risks. The ML model is a classifier to conduct such computations. If the output of D2MON is larger than a user-set threshold (i.e. a hyperparameter in Section IV), it will be predicted as "dangerous". Figure 4 presents the network structure of this ML model. Specifically, this model consists of two components: a sequence-to-sequence (Seq2Seq) model and fully-connected layers (FCLs). The Seq2Seq is composed of two LSTM-based encoder and decoder and hence is good at processing time-series data. FCLs are introduced to further improve fitting capability. The output of this model is the predicted results (i.e. safety or not) of the input time slot. We will evaluate the prediction accuracy in the following section.

### B. D2MON *Deployment*

Figure 5 presents how we deploy D2MON to an AV to mitigate real-time AV-at-fault accidents. After the training process of the ML model is complete, it can be offloaded to an AV for later use. When AV starts running, D2MON first collects 1.25 seconds of data as the initial input. Such a cold start cost is very small and can almost be neglected. Then, D2MON will collect the real-time surrounding information and perform inference through the ML model every 0.25 seconds. This data can be stored with a user-defined period for later possible posthoc analysis. Each cluster has an independent model, and all the models are contained in the D2MON

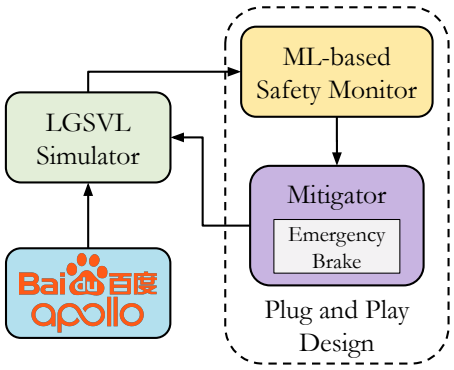



Fig. 5. D2MON deployment. D2MON refers to ML-based safety monitor.

mitigator and will be inferred parallelly while AV is running. Once the predicted value from any cluster is higher than the threshold, we will activate the mitigator for the mitigation. we take over the control of the current AV in the simulator and conduct a maximum AV braking (Emergency Brake) command by predefined Python API. These operations will be performed again after restarting AV. As shown in Figure 5, the deployment of D2MON includes the ML-based safety monitor and mitigator, which are independent of ADS in an AV. As a result, our technique can be configured very flexibly and is easy to use.

## IV. EVALUATION

In this section, we present the evaluation for D2MON.

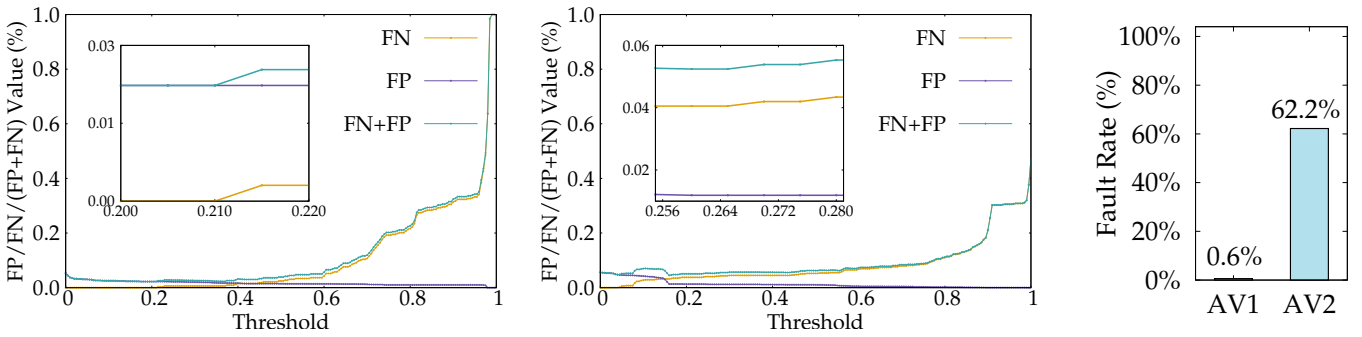
### A. Experiment Settings

*1) Machine Specification:* Our experiments were conducted on an Ubuntu 18.04 machine with 32GB RAM. This machine is also equipped with an AMD 5900X CPU (12-core/24-thread) and an NVIDIA GTX 1080 Ti GPU card.

*2) Driving Environment Setup:* For ADS and high-fidelity simulator, Baidu Apollo 3.5 and LGSVL linux-64-2021.3 were used in our experiments. We used a simple 2-lane map of LGSVL for the map selection and the vehicle configuration was also built into the simulation. We started with AV-FUZZER. Given a random seed, we fuzzed 10 hours by AV-FUZZER, and 9 AV-at-fault accident trials were found. We recorded the NPC behaviors and adopted K-Means algorithm to automatically classify them into two clusters. Data augmentation is also conducted on these two clusters, respectively. Each class ran out of 500 AV-at-fault accidents in this stage. The detailed settings of the ML model training will be presented in the later section.

### B. Evaluation Metrics

There are three metrics for evaluating D2MON in mitigating AV-at-fault accidents. The calculation formula and descriptions can be found in Table I. To evaluate the prediction accuracy, we adopt false positive (FP) and false negative (FN) values. The lower the better. The reason is that both these two




(a) ML prediction accuracy for cluster1. (b) ML prediction accuracy for cluster2. (c) Accident mitigation efficiency

Fig. 6. Evaluation of D2MON. In (c), "AV1" represents "AV with D2MON " and "AV2" represents "AV without D2MON ".

types of faulty predictions can lead to severe results in real-world scenarios. To evaluate the overall accident mitigation efficiency, we adopt a fault rate. The lower the fault rate, the more accidents the AV can avoid.

TABLE I
EVALUATION METRICS

| Metrics | Formula | Descriptions |
|---|---|---|
| FP Val. (False Positive) | (FP Case) / (Total Case) | FP: Predict the safe situation as dangerous. |
| FN Val. (False Negative) | (FN Case) / (Total Case) | FN: Predict the dangerous situation as safe. |
| Fault Rate (%) | (Fault Trial) / (Total Trial) | Number of trials that leads to accidents. |

### C. Prediction Accuracy

Before we present the prediction accuracy, we first present the related settings. Back to the data augmentation phase in D2MON creation workflow, we collected nearly one thousand trials of AV-at-fault accidents or other actions for each cluster. If one trial is very close to an accident but finally executes in normal, we define it as a close trial. On the one hand, we use 400 AV-at-fault accident trials, 200 close trials, and 200 random trials for training the ML predictor for each cluster. On the other hand, we use another 100 AV-at-fault accident trials, 50 close trials, and 50 random trials for testing. We balance the training and testing data to obtain a higher prediction accuracy and a fair evaluation, respectively. We decompose each trial into several sliding windows. There are 20 timeslots in each trial. The first 5 timeslots are the model input (i.e. cold-start stage) for D2MON, while the output is the safety prediction for the later 8 timeslots. Since timeslot has a 5 dimension feature, the input is a $5 \times 5$ matrix. The output of the ML model is an 8 dimension vector, where each number in this vector represents a safety prediction for its corresponding timeslot. Specifically, this predicted number is a floating-point number between 0 and 1. We set a hyperparameter threshold for each cluster here. If this number exceeds the user-defined threshold, the corresponding timeslot will be regarded as dangerous, otherwise safe. **False positive**: If any of the 8 predicted numbers exceeds the threshold and the ground truths of them are all 0, such will be recorded as a false positive case. **False negative**: If all of the 8 predicted numbers do not exceed the threshold while there is one number or more has a ground truth 1, such will be recorded as a false negative case. For each cluster, we traverse the thresholds from 0 to 1 and choose a value that can lead to the smallest FP+FN rate. Such an optimized value will be selected for inference in the ML model. If one ML model of any cluster evaluates NPC actions in the current surrounding as dangerous, the message will be sent to the mitigator. And D2MON will take over control from ADS and conduct an emergent brake.

Figure 6(a) and 6(b) presents the ML prediction results for these two clusters. For cluster1, the FP keeps at a slow value ($< 0.1$) when we traverse the threshold from 0 to 1. The FN is small when the threshold is close to 0, but it increases rapidly when this threshold is larger than 0.6. We set the threshold as 0.21 to obtain a minimum FN+FP, of which FN and FP are 0.0061 and 0.0167, respectively. The curves in cluster2 have similar trends. We then set the threshold as 0.265 to minimize the value of FN+FP, of which FN and FP are 0.0405 and 0.0118. We can know that our ML models can achieve promising prediction accuracy for both clusters when optimized thresholds are selected.

### D. Accident Mitigation Efficiency

In this section, we evaluate the AV-at-fault accident mitigation efficiency of D2MON, including the AV safety monitor and the mitigator. Because 2 clusters obtained the data discriminator phase in the D2MON creation workflow, we also collect data from 2 clusters. For each of these 2 clusters, we use AV-FUZZER to get a base trial and perform data augmentation on each trial. At last there will be in total 500 trials to evaluate the D2MON accident mitigation efficiency. Because of the speed settings in data augmentation, all the trials can or are very likely to lead to accidents.

Figure 6(c) presents the mitigation results. "Type1" and "Type2" are running AVs with and without D2MON. We let both of these two AVs run above 500 trials in the LGSVL simulator and check the results. For AV1, since D2MON is equipped to detect real-time risks, only 0.6% cases (3 in 500) finally lead to AV-at-fault accidents. For AV2, since no

protection is granted, 62.2% cases (303 in 500) cause accidents at last. D2MON can detect near 99.01% possible AV-at-fault accidents, which demonstrates the protection of D2MON is effective.

## V. CONCLUSION AND FUTURE WORKS

In this work, we propose an AV safety monitor D2MON to detect and mitigate hazards in real-time. D2MON uses a neural network composed of Seq2Seq and FCLs to evaluate the risks of surroundings. The creation and deployment of D2MON are automatic and flexible, significantly reducing the time cost compared with existing AV protection methods. Evaluation shows that the ML model is accurate in predicting hazards and D2MON is effective in mitigating hazards (more than 99% accident cases) in real-time driving scenarios.

In the future, we will further explore this topic in two possible directions: (1) accelerating the fuzzing engine, and (2) optimizing the mitigation strategies.

## REFERENCES


[1] "Summary report: June 2022 standing general order on crash reporting for level 2 advanced driver assistance systems," https://www.nhtsa.gov/sites/nhtsa.gov/files/2022-06/ADAS-L2-SGO-Report-June-2022.pdf.

[2] W. Zhan, C. Liu, C.-Y. Chan, and M. Tomizuka, "A non-conservatively defensive strategy for urban autonomous driving," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 459–464.

[3] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek, "Generating adversarial driving scenarios in high-fidelity simulators," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8271–8277.

[4] G. Li, Y. Li, S. Jha, T. Tsai, M. Sullivan, S. K. S. Hari, Z. Kalbarczyk, and R. Iyer, "Av-fuzzer: Finding safety violations in autonomous driving systems," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2020, pp. 25–36.

[5] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58 443–58 469, 2020.

[6] A. C. Madriga, "Inside waymo's secret world for training self-driving cars," *The Atlantic*, vol. 23, 2017.

[7] F. Zhu, L. Ma, X. Xu, D. Guo, X. Cui, and Q. Kong, "Baidu apollo auto-calibration system-an industry-level data-driven and learning based vehicle longitude dynamic calibrating algorithm," *arXiv preprint arXiv:1808.10134*, 2018.

[8] S. Jha, S. Banerjee, T. Tsai, S. K. Hari, M. B. Sullivan, Z. T. Kalbarczyk, S. W. Keckler, and R. K. Iyer, "Ml-based fault injection for autonomous vehicles: A case for bayesian fault injection," in *2019 49th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2019, pp. 112–124.

[9] "Baidu apollo github repository," https://github.com/ApolloAuto/apollo.

[10] Z. Hu, S. Guo, Z. Zhong, and K. Li, "Disclosing the fragility problem of virtual safety testing for autonomous driving systems," in *2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2021, pp. 387–392.

[11] "Carla autonomous driving system," https://carla.readthedocs.io//.

[12] "Lgsvl simulator," https://www.lgsvlsimulator.com/.

[13] B. P. Miller, L. Fredriksen, and B. So, "An empirical study of the reliability of unix utilities," *Communications of the ACM*, vol. 33, no. 12, pp. 32–44, 1990.

[14] H. Chen, S. Guo, Y. Xue, Y. Sui, C. Zhang, Y. Li, H. Wang, and Y. Liu, "{MUZZ}: Thread-aware grey-box fuzzing for effective bug hunting in multithreaded programs," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 2325–2342.

[15] Y. Chen, P. Li, J. Xu, S. Guo, R. Zhou, Y. Zhang, T. Wei, and L. Lu, "Savior: Towards bug-driven hybrid testing," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1580–1596.

[16] H. Chen, Y. Xue, Y. Li, B. Chen, X. Xie, X. Wu, and Y. Liu, "Hawkeye: Towards a desired directed grey-box fuzzer," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 2095–2108.

[17] Y. Li, Y. Xue, H. Chen, X. Wu, C. Zhang, X. Xie, H. Wang, and Y. Liu, "Cerebro: context-aware adaptive fuzzing for effective vulnerability detection," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 533–544.

[18] O. Oleksenko, B. Trach, M. Silberstein, and C. Fetzer, "{SpecFuzz}: Bringing spectre-type vulnerabilities to the surface," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1481–1498.

[19] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, pp. 64–67, 2001.

[20] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.