# ML-based Fault Injection for Autonomous Vehicles:

## A Case for Bayesian Fault Injection

Saurabh Jha*, Subho S. Banerjee*, Timothy Tsai+, Siva K. S. Hari+, Michael B. Sullivan+, Zbigniew T. Kalbarczyk*, Stephen W. Keckler+ and Ravishankar K. Iyer*

*University of Illinois at Urbana Champaign, +NVIDIA Corporation

2019 IEEE Dependable Systems and Networks (DSN)

Jonathan Lee, Secure Autonomous Systems, October 20th 2022

# MOTIVATION

"Why Uber's self-driving car killed a pedestrian," 2018

- Autonomous system not trained to classify objects as a pedestrian unless near a crosswalk
- Due to misclassification, it could not correctly predict her path
- First recorded death by a self-driving vehicle
- Led Uber to cancel its autonomous testing in Arizona

"Research group demos why Tesla Autopilot could crash into a stationary vehicle," 2018

- Freeway, lead vehicle changes lanes without warning
- Tesla accelerates to match the allowed highway speed
- However, the autopilot has no knowledge of stationary vehicle in front
- Too late for autopilot to recognize car and slow down in time to avoid an accident

# APPROACH

## GOAL

Identify hazardous situations that can lead to collisions and accidents.

## FAULT INJECTION

Test behavior by purposefully introducing errors into the cyber-physical system.

# CONTRIBUTIONS

**DriveFI**

1. A **fault injection engine** that can modify the software and hardware states of an autonomous driving system

2. A **ML-based fault selection engine**, that can find situations and faults that are most likely lead to violations of safety conditions

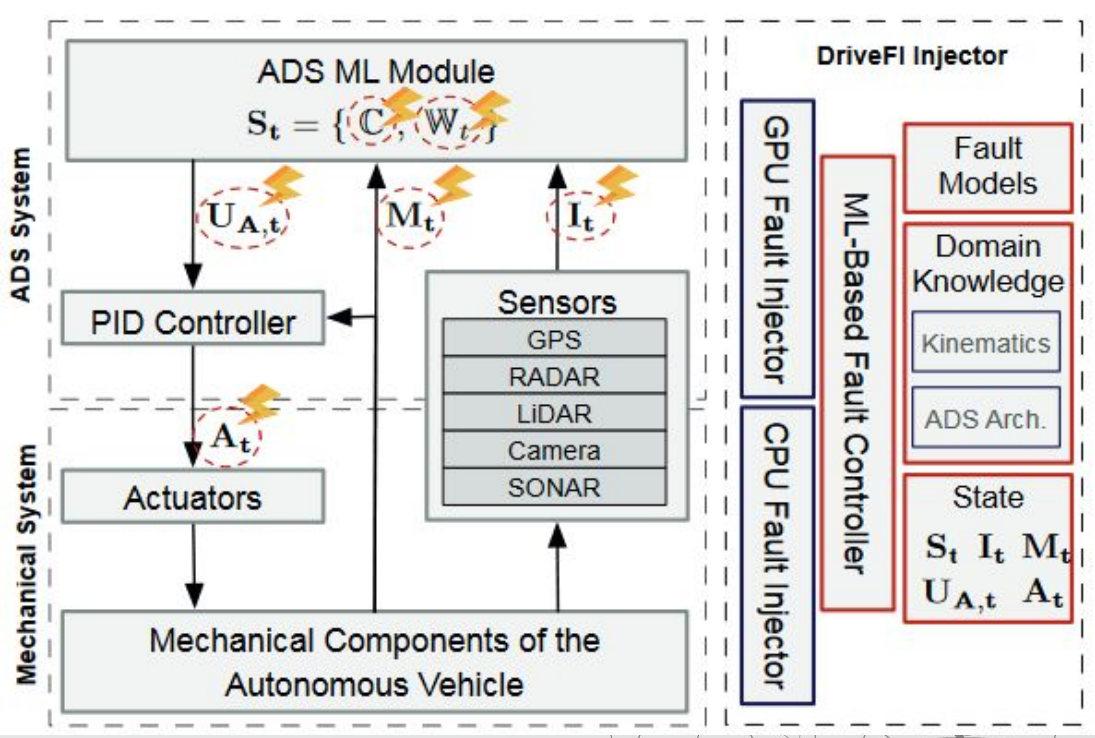
# DRIVE FI

**Autonomous Driving System (ADS)**

- ADS system
- Mechanical components

**DriveFI**

- **Fault injection engine**
  - GPU Fault Injector
  - CPU Fault Injector
- **ML-based fault selection engine**
  - Fault selection policy



# AUTONOMOUS DRIVING SYSTEMS (ADS)

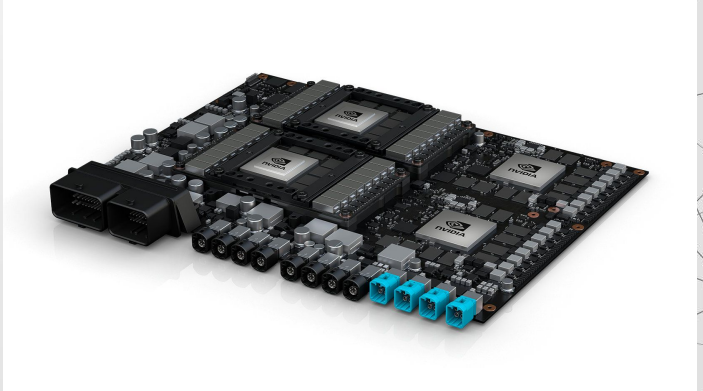
Tested two level-4 Autonomous Driving Systems

DriveAV (NVIDIA)

- NVIDIA AGX Pegasus platform

Apollo 3.0 (Baidu)

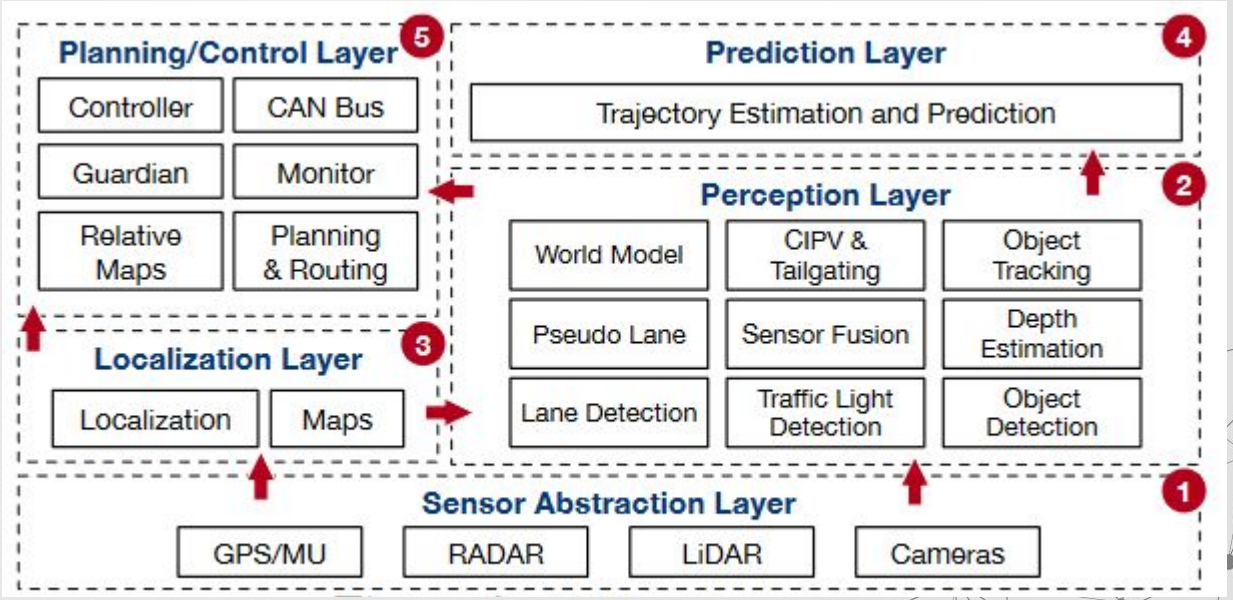
- x86 workstation with two NVIDIA Titan Xp GPUs



# ADS ARCHITECTURE

1. **Sensor abstraction layer**
   a. preprocessing
   b. noise filtering
2. **Perception layer**
   a. object detection
   b. sensor fusion
   c. temporal tracking
3. **Localization layer**
   a. vehicle location in scene
4. **Prediction layer**
   a. trajectories of objects
   b. identify obstacles
5. **Planning & control layer**
   a. collision-free trajectory
   b. actuator control command

**Autonomous Driving System (ADS) Architecture:**



# HARDWARE & SOFTWARE FAULT INJECTION

**Hardware** - Injecting into computational elements

- Inject bit-flips into the outputs of executing instructions (flip-bits in the destination registers)
- Unprotected SRAM structures of the GPU processor

**Software** - Source code level injection

- Maximum & minimum values
- Doubling or halving current output values
- Object detection:: "do not care", "pedestrian", "cyclist", "vehicle"

**Table I:** Examples of SLI-supported ADS module outputs.

| FI Target (Output Variables) |
|---|
| **Path Perception Module** |
| lane_type, lane_width |
| **Object Perception Module** |
| camera_object_distance, camera_object_class, lidar_object_distance, lidar_object_class, sensor_fused_obstacle_distance, sensor_fused_obstacle_class |
| **Planning & Control Module** |
| vehicle_state_measurements $(pos, v, a)$, obstacle_state_measurements $(pos, v, a)$, actuator_values $(\zeta, b, \phi)$, pid_measured_value, pid_output |

# MODELING SAFETY

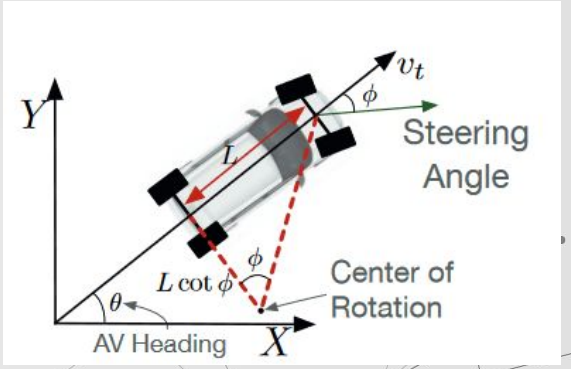
Kinematics-Based Model of Safety

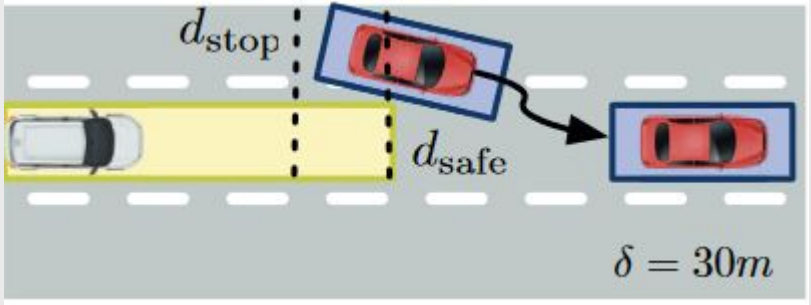
- position, speed, heading, steering angle

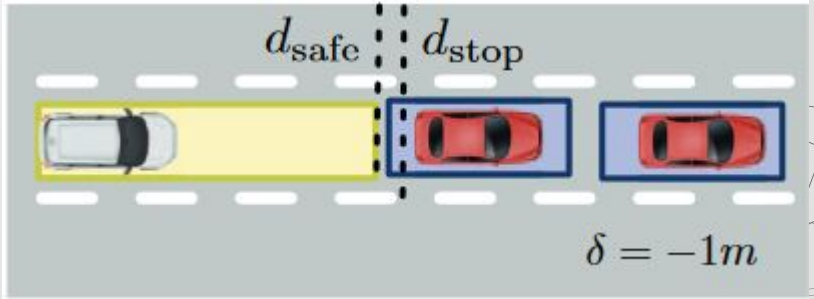
Stopping distance - distance vehicle will travel before coming to a stop

Safety envelope - distance vehicle can travel without colliding with an object

δ = d_safe - d_stop > 0



# SIMULATING DRIVING SCENARIOS

Simulation: Carla and DriveSim (Unreal Engine based simulation platforms)

6 Scenarios: 500-2400 time intervals

For each scenario:

- Control run with no faults (golden run)
- Inject one or more transient faults per run
- Time interval of injection selected randomly

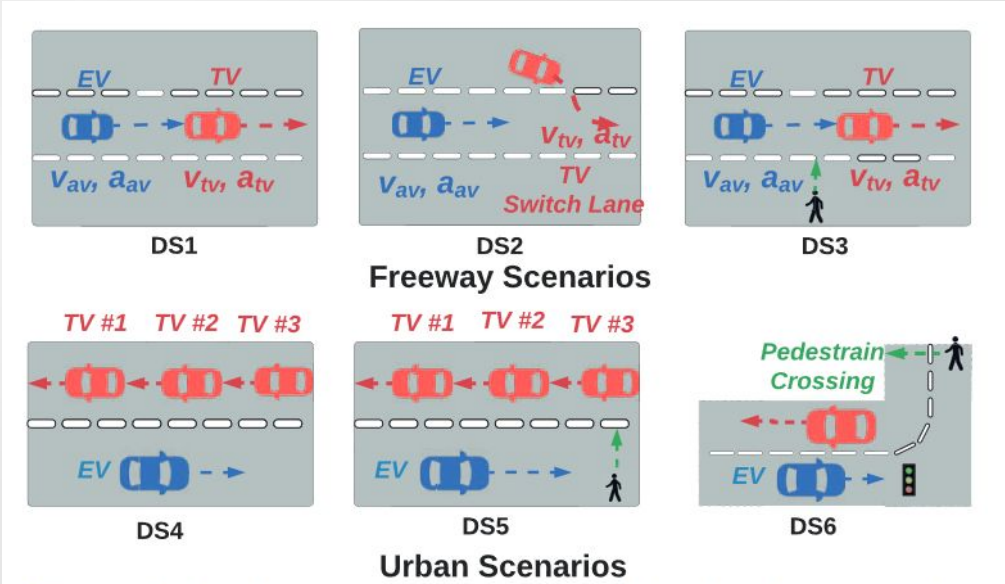
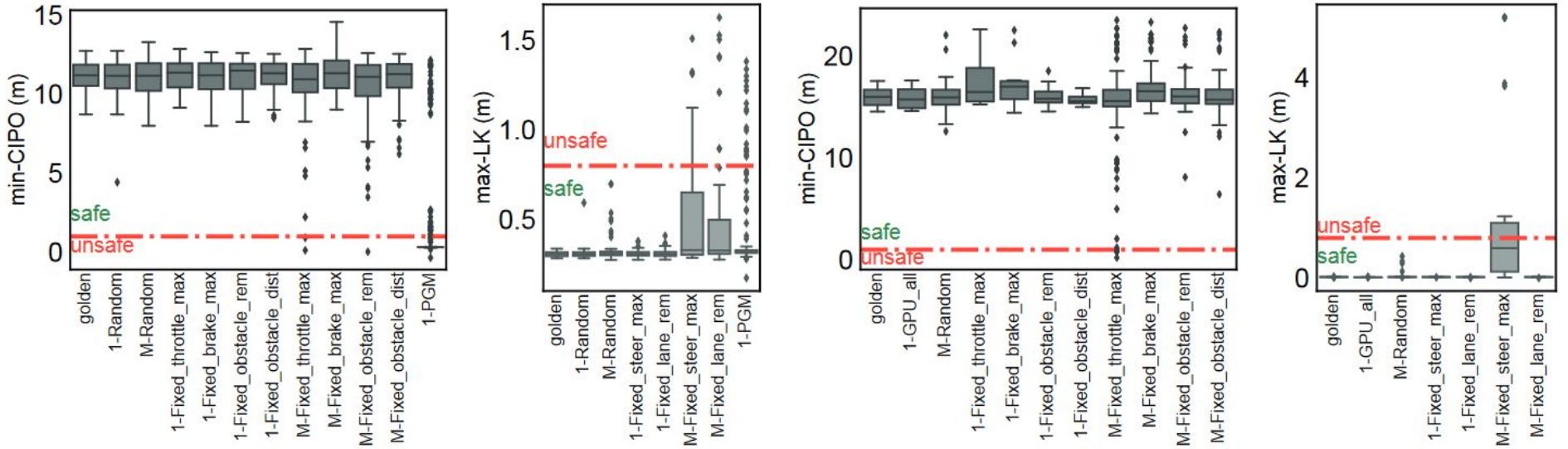



**Figure 9:** Driving scenarios supported by simulation engine.

# SIMULATION OF RANDOM FAULTS




**(a)** Apollo min-CIPO **(b)** Apollo max-LK **(c)** DriveAV min-CIPO **(d)** DriveAV max-LK

**Figure 11:** Fault/error impact characterization using FI campaigns. (a) & (b) use DS6; (c) & (d) use DS1.

Only 1.9% led to data corruption, and 0.02% led to object misclassification errors (2400 experiments)

None of these led to an unsafe state.

# RESILIENT TO TRANSIENT FAULTS

## SENSOR FUSION

Redundancy of sensors compensates. Kalman filtering.

Errors in output masked by new sensor data, collected at regular intervals.

## PERSISTENT UPDATES

## OBSTACLE TRACKING

Don't need to classify object every frame due to object tracking.

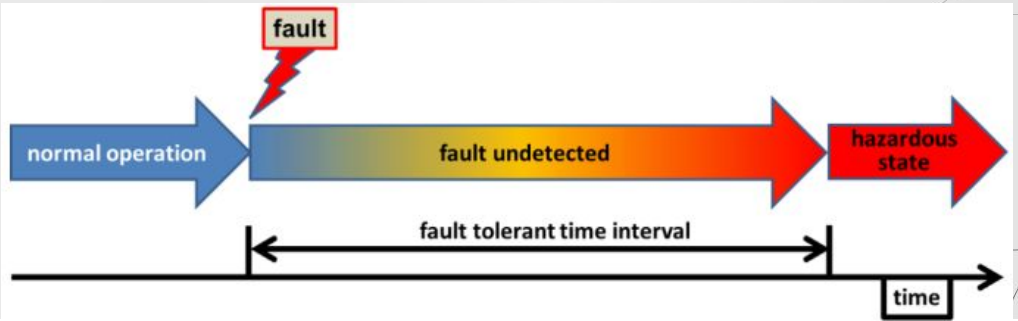
# IS THE SYSTEM SAFE?

No, although these transient faults didn't cause unsafe conditions, this does not mean that such faults are nonexistent.

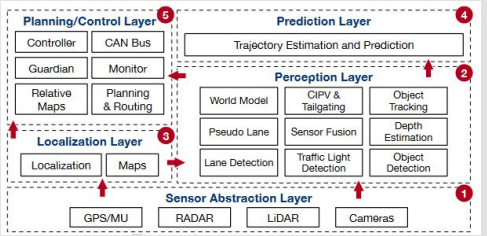
Such critical faults are rare, and situational. They're unlikely to find through random sampling.

An exhaustive search - time consuming (272 days).

**We need a smart method to search for single faults that cause the system to enter an unsafe state from a safe state.**



# BAYESIAN FAULT INJECTION




Bayesian Networks - Directed acyclic graphs. Represents the causal relationships between nodes (random variables), within a system.

Train probability distributions of each node on driving scenarios with random faults

**Key idea - make probabilistic inference on the next kinematic state of the vehicle.**

motion (heading, velocity, accel…)

variables representing ADS modules

actuation (steering, braking, throttle)

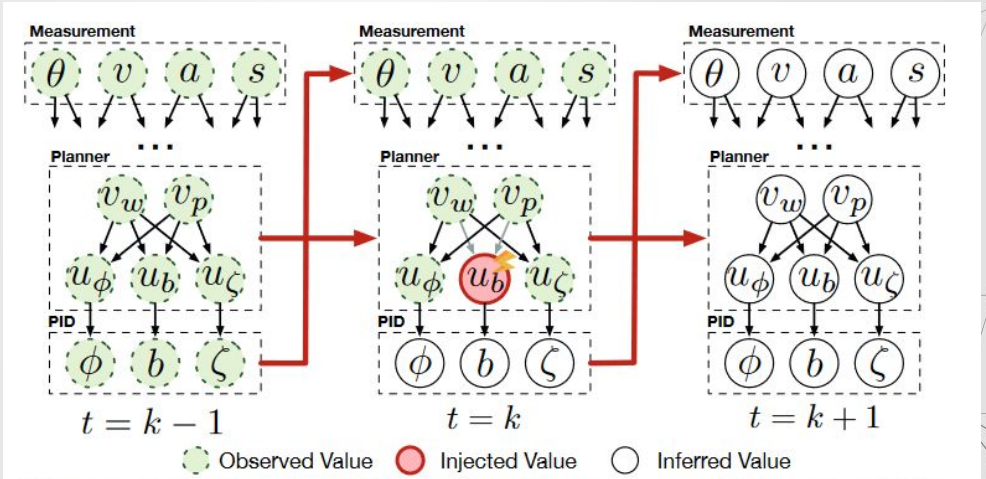



**Figure 6:** 3-Temporal Bayesian Network modeling the ADS.

# MINING FAULTS

Use the Bayesian Network to find the set of critical faults within a scenario.

At each time interval, measure and input variables and introduce a fault into Bayesian network

Maximum Likelihood Estimation (MLE) of stopping distance from our observations.

Is stopping distance more than the boundary of the safety envelope?

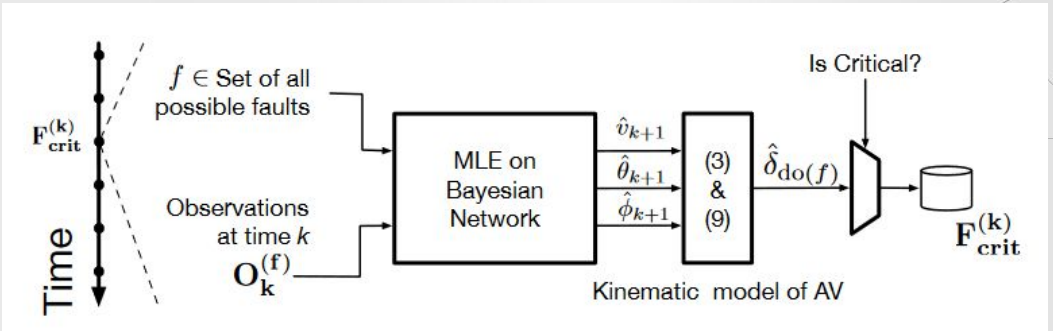


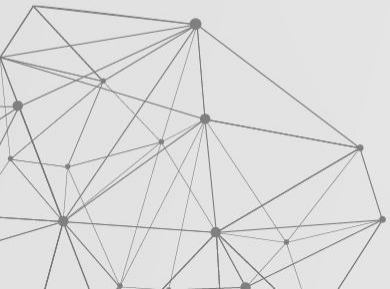

**Figure 7:** BN MLE inference is executed offline for every simulated time point to find the set of critical faults.

# EFFECTIVENESS OF BAYESIAN FI

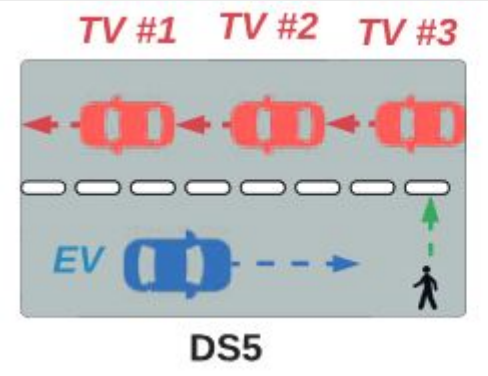
**4** Hours to mine faults

**561** Critical faults identified

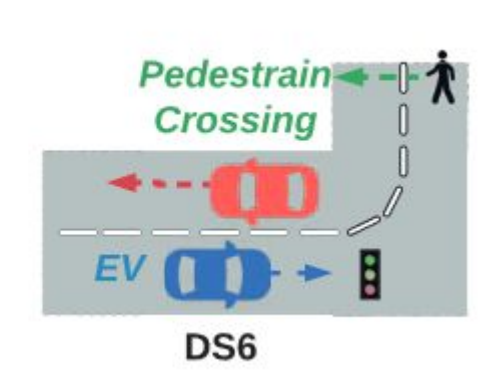
**82%** of faults caused accidents



# CRITICAL FAULT EXAMPLES



DS5



DS6

DS5: Hits pedestrian

1. Pedestrian registered into world model, but fault injection removes/misclassifies the pedestrian
2. Vehicle starts to brake, but fault injection accelerates the vehicle

DS6: Fault around turns

1. Steering value corruption
2. Disappearing lane markings

# Questions?

# DISCUSSION

- At what level of safety would you consider autonomous vehicles to be safe to operate on the road?
  - Less than 0.001% of critical faults in random scenarios?
  - Safer than the average human?

- Do you think all AV testing should be conducted in simulation? On the road? Or a mix of both?

# CLOSING THOUGHTS

Approve

- Contribution to identifying critical faults. Helps to make autonomous systems more resilient.
- Bayesian Fault Injection framework
    - Can be extended to other fields (e.g. surgical robots, airplanes)

The ISO 26262 set of standards was first published in 2011, and applies to the full safety lifecycle (concept, development and operation) of electrical and electronic systems in production-line passenger cars.

# THANKS FOR LISTENING!