Explaining Errors in Autonomous Vehicles

A Diagnosis Tool and Testing Framework for Robust Decision Making

Leilani H. Gilpin, PhD
Dept. of CSE, UC Santa Cruz
JHU CaSE Graduate Seminar
November 11, 2021
Agenda

Motivate problem: Autonomous Vehicles are Prone to Failure

Anomaly Detection through Explanations (ADE): a Diagnosis Tool for AVs.

Adversarial Examples as a Testing Framework for Autonomous System Robustness.

Future work: Explainable Tasks for Robust and Secure Hybrid Systems.

Question: How to develop self-explaining architectures that can help anticipate failures instead of after-the-fact?
Autonomous Vehicles are Prone to Failure

Autonomous Vehicle Solutions are at Two Extremes

Cautious

Comfort

Very comfortable

Not comfortable

Problem: Need better common sense and reasoning

Serious safety lapses led to Uber’s fatal self-driving crash, new documents suggest

My Herky-Jerky Ride in General Motors' Ultra-Cautious Self Driving Car

GM and Cruise are testing vehicles in a chaotic city, and the tech still has a ways to go.
Complex Systems Include People

Misalignment of Expectations

Solution: Built-in structures to deal with flaws and failures

Lack of communication

Expectation
Architecture Inspired by Human Organizations
Communication and Sanity Checks

1. Hierarchy of overlapping committees.
2. Continuous interaction and communication.
3. When failure occurs, a story can be made, combining the members’ observations.
An Architecture to Mitigate Common Problems

Synthesizer to reconcile inconsistencies between parts.

Local Sanity Checks

Reconcile conflicting reasons. Justify new examples.
An Existing Problem

The Uber Accident
The best option is to veer and slow down. The vehicle is traveling too fast to suddenly stop. The vision system is inconsistent, but the lidar system has provided a reasonable and strong claim to avoid the object moving across the street.
Agenda

Motivate problem: Autonomous Vehicles are Prone to Failure

Anomaly Detection through Explanations (ADE): a Diagnosis Tool for AVs.

Adversarial Examples as a Testing Framework for Autonomous System Robustness.

Future work: Explainable Tasks for Robust and Secure Hybrid Systems.
Limited Internal Reasoning

A Google self-driving car caused a crash for the first time

A bad assumption led to a minor fender-bender

Serious safety lapses led to Uber’s fatal self-driving crash, new documents suggest

My Herky-Jerky Ride in General Motors' Ultra-Cautious Self Driving Car

GM and Cruise are testing vehicles in a chaotic city, and the tech still has a ways to go.
Reconciling Internal Disagreements
With an Organizational Architecture

- Monitored subsystems combine into a system architecture.
- Explanation synthesizer to deal with inconsistencies.
  - Argument tree.
  - Queried for support or counterfactuals.
Anomaly Detection through Explanations
Reasoning in Three Steps

1. Generate Symbolic Qualitative Descriptions for each committee.
2. Input qualitative descriptions into local “reasonableness” monitors.
3. Use a synthesizer to reconcile inconsistencies between monitors.

VISION  LiDAR  TACTICS

Synthesizer

Brakes  Steering  Power
Use a synthesizer to reconcile inconsistencies between monitors.

- Explanation synthesizer to deal with inconsistencies.
- Argument tree.
- Queried for support or counterfactuals.

Priority Hierarchy

1. Passenger Safety
2. Passenger Perceived Safety
3. Passenger Comfort
4. Efficiency (e.g. Route efficiency)

A passenger is safe if:

- The vehicle proceeds at the same speed and direction.
- The vehicle avoids threatening objects.
3. Use a synthesizer to reconcile inconsistencies between monitors.

\[(\forall s, t \in \text{STATE}, v \in \text{VELOCITY}) \land
((\text{self}, \text{moving}, v), \text{state}, s) \land
(t, \text{isSuccessorState}, s) \land
((\text{self}, \text{moving}, v), \text{state}, t) \land
(\forall x \in \text{OBJECTS} \text{ s.t.})
((x, \text{isA}, \text{threat}), \text{state}, s) \lor
((x, \text{isA}, \text{threat}), \text{state}, t))\]

\Rightarrow (\text{passenger, hasProperty, safe})

\[(\forall s \in \text{STATE}, x \in \text{OBJECT}, v \in \text{VELOCITY})
((x, \text{moving}, v), \text{state}, s) \land
((x, \text{locatedNear}, \text{self}), \text{state}, s) \land
((x, \text{isA}, \text{large}\_\text{object}), \text{state}, s)
\leftrightarrow ((x, \text{isA}, \text{threat}), \text{state}, s))\]
3. Use a synthesizer to reconcile inconsistencies between monitors.

\[(\forall s, t \in \text{STATE}, v \in \text{VELOCITY}) \]
\[\left( (\text{self, moving}, v), \text{state}, s \right) \land \]
\[\left( t, \text{isSuccessorState}, s \right) \land \]
\[\left( (\text{self, moving}, v), \text{state}, t \right) \land \]
\[\left( \forall x \in \text{OBJECTS} \text{ s.t.} \right) \]
\[\left( (x, \text{isA, threat}), \text{state}, s \right) \lor \]
\[\left( (x, \text{isA, threat}), \text{state}, t \right) \]
\[\Rightarrow (\text{passenger, hasProperty, safe})\]

Abstract Goal Tree

'passenger is safe',
AND(
  'safe transitions',
  NOT('threatening objects'))
3. Use a synthesizer to reconcile inconsistencies between monitors.

**Abstract Goal Tree**

'passenger is safe',
AND(
  'safe transitions',
  NOT('threatening objects'))

**List of Rules**

IF (AND('moving (?v) at state (?y)'),
     '(?z) succeeds (?y)',
     'moving (?v) at state (?z)'),
THEN('safe driving at (?v) during (?y) and (?z)')

IF (OR('obj is not moving',
       'obj is not located near',
       'obj is not a large object'),
THEN('obj not a threat at (?x)')

IF (AND('obj not a threat at (?y)',
       'obj not a threat at (?z)',
       '(?z) succeeds (?z)'),
THEN('obj is not a threat between (?y) and (?z)')

**Backwards Chain**

passenger is safe at V between s and t
AND (AND (moving V at state s
           t succeeds s
           moving V at state t )
     AND ( OR ( obj is not moving at s
                  obj is not locatedNear at s
                  obj is not a large object at s )
          OR ( obj is not moving at t
               obj is not locatedNear at t
               obj is not a large object at t ) ) )
The best option is to veer and slow down. The vehicle is traveling too fast to suddenly stop. The vision system is inconsistent, but the lidar system has provided a reasonable and strong claim to avoid the object moving across the street.

Abstract Goal Tree

'passenger is safe',
AND(
  'safe transitions',
  NOT('threatening objects')
)

The best option is to veer and slow down. The vehicle is traveling too fast to suddenly stop. The vision system is inconsistent, but the lidar system has provided a reasonable and strong claim to avoid the object moving across the street.
Uber Example in Simulation

Evaluation of Error Detection is Difficult

Real-world Inspired Scenarios

- Detection: Generate logs from scenarios to detect failures.
- Insert errors: Scrambling *multiple* labels on existing datasets.
- Real errors: Examining errors on the validation dataset of NuScenes leaderboard.

<table>
<thead>
<tr>
<th>Priority</th>
<th>Correctness</th>
<th>False Positives</th>
<th>False Negatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>No synthesizer</td>
<td>85.6%</td>
<td>7.1%</td>
<td>7.3%</td>
</tr>
<tr>
<td>Single subsystem</td>
<td>88.9%</td>
<td>7.9%</td>
<td>3.2%</td>
</tr>
<tr>
<td>Safety</td>
<td>93.5%</td>
<td>4.8%</td>
<td>1.7%</td>
</tr>
</tbody>
</table>
Agenda

Motivate problem: Autonomous Vehicles are Prone to Failure

Anomaly Detection through Explanations (ADE): a Diagnosis Tool for AVs.

Adversarial Examples as a Testing Framework for Autonomous System Robustness.

Future work: Explainable Tasks for Robust and Secure Hybrid Systems.
Vision: Real World Adversarial Examples

Vision: Real World Adversarial Examples
Anticipatory Thinking Layer for Error Detection

The traffic lights are on top of the truck. The lights are not illuminated. The lights are moving at the same rate as the truck, therefore this is not a “regular” traffic light for slowing down and stopping at.
The traffic lights are on top of the truck. The lights are not illuminated. The lights are moving at the same rate as the truck, therefore this is not a “regular” traffic light for slowing down and stopping at.
Lack of Data and Challenges for AVs

- Existing Challenges
  - Targeted as optimizing a mission or trajectory and not safety.
  - Data is hand-curated.
  - Failure data is not available
    - Unethical to get it (cannot just drive into bad situations).
  - Want the data to be realistic (usually difficult in simulation).

Data from NuScenes
Existing Challenges and Benchmarks
Not Focused on Out of Domain Errors

NHTSA-inspired pre-crash scenarios

Traffic Scenario 01: Control loss without previous action
- Definition: The ego-vehicle loses control due to bad conditions on the road and it must recover, coming back to its original lane.

Traffic Scenario 02: Longitudinal control after leading vehicle’s brake
- Definition: The leading vehicle decelerates suddenly due to an obstacle and the ego-vehicle must react, performing an emergency brake or an avoidance maneuver.

Traffic Scenario 03: Obstacle avoidance without prior action
- Definition: The ego-vehicle encounters an obstacle/unexpected entity on the road and must perform an emergency brake or an avoidance maneuver.
Other Challenges Not Anticipatory
Not Focused on Error Detection
Approach: Content Generation

Anticipatory Thinking Layer for Error Detection

DALL-E Generates “A chair in the shape of an avocado”
Approach: Content Generation
Anticipatory Thinking Layer for Error Detection

- Generate images with shadows before tunnels.
- Generate images with fallen signs.
- Generate images with trucks carrying traffic lights.
Approach: Content Generation

Anticipatory Thinking Layer for Error Detection

Generate images with shadows before tunnels.
Generate images with fallen signs.
Generate images with trucks carrying traffic lights.
Need for Context

“Realistic” Adversarial
Approach: How it Works

Use Adversarial Images in Dev Testing

• Solution: Use a cognitive architecture that helps to anticipate and understand these failure cases.

• Assess autonomous vehicles for their risk management capabilities before being deployed and provide incident level risk management explanations in human readable form.
**Isolated error detection**

- Error detection
- Error detection
- Error detection

**Integrated error detection**

- Explanatory Error Detection
- Content generation

---

**Dev**

- Observed errors

**Deploy**

- All errors
  - Observed errors
  - Content generated errors
Impact
Anticipatory Thinking Layer for Error Detection

• Goal - Develop methods that a priori can explain an autonomous vehicle’s ability to manage the risks stemming from errors in perceiving their environment.

• One possible solution is to explain why the autonomous behavior is safe (or risky, trustworthy, etc.) or not.

• Impact - Consumer confidence and safety features, appropriate legal and regulatory oversight.
Agenda

Motivate problem: Autonomous Vehicles are Prone to Failure

Anomaly Detection through Explanations (ADE): a Diagnosis Tool for AVs.

Adversarial Examples as a Testing Framework for Autonomous System Robustness.

Future work: Explainable Tasks for Robust and Secure Hybrid Systems.
Hybrid Systems with Humans and Machines
Working Together on Shared Tasks

Explanations are a debugging language.

- Debugging: humans can improve complex systems.
- Education: complex systems can “improve” or teach humans.
Ex post facto explanations

Log data
Input
Sensor data

Learning system

“Explanation”
Debugging
Learning system

Saliency map

Symbolic system

The object to the left is 5 ft tall, moving towards the right. It's the salient feature of neurons.

Feedback
Validation
Help on tasks

Humans

Contextual justification: “This is a person because they have the right shape and movement.”

Log data

Sensor data

Input

Dev testing

Game adversaries

Security
Impact
Confidence and Integrity of Systems

**Society**
Systems that articulately communicate with humans on shared tasks.

**Liability**
Systems that can testify, answer questions, and provide insights.

**Robustness**
Dynamic detection of failure and intrusion with precise mitigation.
Contributions

The problem: Autonomous Vehicles are Prone to Failure.

Anomaly Detection through Explanations (ADE): a Diagnosis Tool for AVs.

Adversarial Examples as a Testing Framework for Autonomous System Robustness.

Explainable Tasks for Robust and Secure Hybrid Systems.