Accountability Layers
Stress-testing Using Explainable AI for Safety-critical Systems

Leilani H. Gilpin
Assistant Professor
Dept. of Computer Science & Engineering, UC Santa Cruz
Motivate problem: Autonomous Vehicles are Prone to Failure

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

Ongoing Work: Adversarial Examples for as a Stress Testing Framework.

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

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

Problem: Need better common sense and reasoning

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.
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.*

Ongoing Work: Adversarial Examples for as a Stress Testing Framework.
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.

---

Architecture Diagram:

- **Synthesizer**
  - **VISION**
  - **LiDAR**
  - **TACTICS**
    - **Brakes**
    - **Steering**
    - **Power**

Anomaly Detection Through Explanations
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.
3. Use a synthesizer to reconcile inconsistencies between monitors.

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

<table>
<thead>
<tr>
<th>Priority Hierarchy</th>
<th>Abstract Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Passenger Safety</td>
<td>A passenger is safe if:</td>
</tr>
<tr>
<td>2. Passenger Perceived Safety</td>
<td></td>
</tr>
<tr>
<td>3. Passenger Comfort</td>
<td></td>
</tr>
<tr>
<td>4. Efficiency (e.g. Route efficiency)</td>
<td></td>
</tr>
</tbody>
</table>

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

\[(\forall s, t \in \text{STATE}, v \in \text{VELOCITY})
\quad ((\text{self}, \text{moving}, v), \text{state}, s) \land
\quad (t, \text{isSuccessorState}, s) \land
\quad ((\text{self}, \text{moving}, v), \text{state}, t) \land
\quad (\nexists x \in \text{OBJECTS} \hspace{1em} \text{s.t.}
\quad ((x, \text{isA}, \text{threat}), \text{state}, s) \lor
\quad ((x, \text{isA}, \text{threat}), \text{state}, t)))\]

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

A passenger is safe if:

- The vehicle proceeds at the same speed and direction.
- The vehicle avoids threatening objects.

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

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

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

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

Backwards Chain

AND/OR TREE

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)'))

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 ) ) )
3. Use a synthesizer to reconcile inconsistencies between monitors.

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.

Ongoing Work: Adversarial Examples for as a Stress Testing Framework.
Vision: Real World Adversarial Examples

Vision: Real World Adversarial Examples

Anticipatory Thinking Layer for Error Detection

"Realistic" Adversarial examples

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).
Approach: Content Generation
Anticipatory Thinking Layer for Error Detection

DALL-E Generates “A chair in the shape of an avocado”

Synthetic images produced by StyleGAN, a GAN created by Nvidia researchers.
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.

Generate “dangerous driving.”
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.
- Generate “dangerous driving.”
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.
Generate “dangerous driving.”

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.

Generate “dangerous driving.”

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.
Generate “dangerous driving.”
Need for Context and Explanation

“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.
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.
Contributions

The problem: Autonomous Vehicles are Prone to Failure.

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

Ongoing Work: Adversarial Examples for as a Stress Testing Framework.