NARS in TruePAL: a Trusted and Explainable AGI Partner for First Responders

Christian Hahm*[1], Mina Gabriel[1], Patrick Hammer[2], Peter Isaev[1] and Pei Wang†[1]

[1]*Department of Computer and Information Sciences, Temple University*
[2]*Department of Psychology, Stockholm University*

October 10, 2023

Abstract

The artificial general intelligence system, NARS, was experimentally tested in an on-board AI assistant for first responders. The system, named TruePAL, passed 16 test cases related to first responder safety. TruePAL utilizes multiple techniques including vehicle simulation, object recognition, intelligent reasoning, and so on. NARS provides the ability of real-time reasoning, learning, decision making, and integrating the sensory data with background knowledge.

Contents

1 Introduction **1**
1.1 Background . . . 1
1.2 NARS . . . 2
1.3 TruePAL Ecosystem . . . 2

2 TruePAL **3**
2.1 System Overview and Module Integration . . . 3
2.2 Driving Simulator . . . 4
2.3 Object Detection and Tracking . . . 5
2.3.1 Visual Object Detection in Camera Feed . . . 5
2.3.2 Object Detection in RADAR feed . . . 5
2.3.3 Object Tracking . . . 6
2.4 Non-Axiomatic Reasoning System (NARS) . . . 6
2.5 TruePAL Mobile Application . . . 8

3 Test Cases **11**
3.1 List of Test Cases . . . 11
3.2 Discussion . . . 12

4 Conclusion **16**

1 Introduction

1.1 Background

First responders (FRs) are responsible for managing emergency situations (e.g., violent crimes, medical emergencies, destructive fires) in a timely manner so as to effectively minimize damage to life and property. As a consequence of handling time-critical dangerous emergency situations, a first responder's job is fraught with personal risks.

*Contact: christian.hahm@temple.edu
†Contact: pei.wang@temple.edu

One particularly dangerous aspect of responding to an emergency situation is the vehicle travel from the FR's station to the scene of the emergency. Indeed, on average, one police officer dies per week on U.S. roads [Gray, 2020, p.12], and ambulances are involved in thousands of accidents each year [Fetto Law Group, 2018]. Overall, emergency vehicle accidents cost $35 Billion per year in the US and $250 Billion per year worldwide [Fetto Law Group, 2018].

As part of an initiative to address this issue of emergency traffic safety, we explored the use of explainable AI and the latest AI technologies, in the Trusted and explainable Artificial Intelligence for Saving Lives (TruePAL) project. This project was a collaboration between Temple University and NASA Jet Propulsion Laboratory, to explore how state-of-the-art AI technologies, including NARS that has been developed by the Temple team [Wang, 2013, Hammer and Lofthouse, 2020], could be used to assist first responders.

TruePAL was developed and tested in computer simulation. Hardware-wise, the simulated FR vehicle was equipped with various sensors, including camera, RADAR, vehicle-to-vehicle communication, and database access. The AI system performs fusion on the data from various sensor sources to perceive the environment and make predictions. Driving assistance was only a small part of the project; TruePAL was also able to assist with different types of incidents (such as medical emergencies, chemical spills, etc.). TruePAL was not restricted to the on-board vehicle, but also works for first responders operating on foot outside of vehicles. TruePAL runs on a smartphone app, aggregating and disseminating relevant information from all TruePAL-equipped first responders in the area. For example, first responders would be aware of each others location by viewing it in the TruePAL mobile app's real-time map.

The AI technology developed in this project was designed to assist first responders by providing contextually relevant information in general, and real-time situational alerts when it is critical to do so. Importantly, the AI system was able to *explain* its reasons for giving alerts; this property of "explainability" is different from almost all other modern AI systems which have the issue of "unexplainability". A modern deep learning system functions like a "black box", lacking human-understandable reasons for its decisions; in contrast, NARS is a reasoning system, which means it has an explainable chain of reasons for its actions which can be audited.

1.2 NARS

A key technology which enabled the TruePAL project was NARS (Non-Axiomatic Reasoning System) [Wang, 1995, Wang, 2013, Wang et al., 2020], an artificial general intelligence (AGI) system. Thanks to its general-purpose ability, the system can integrate sensor data with background knowledge provided by human users. Due to its underlying framework as a reasoning system, NARS memory content and "thought process" are fully transparent and human-understandable.

NARS, unlike most traditional reasoning systems, is designed to work in real-time. One of the latest and most advanced implementations of NARS, called OpenNARS-for-Applications (ONA) [Hammer and Lofthouse, 2020], was designed in particular for working in real-time applications, and so ONA was selected for TruePAL. As we will see, the system's ability to learn in real-time (while operating) can also enhance the warning behavior of the system, such as to reduce the generation of warnings under conditions when they are unnecessarily triggered and must be adjusted. The system not only generates warnings, but also explains why the warnings were given, which is a crucial requirement of explainability.

We implemented TruePAL as proof-of-concept in a driving simulator (called CARLA) rather than on a real world vehicle, though theoretically, the simulation was realistic enough that TruePAL would transfer over to a real world vehicle with little to no changes necessary in the code.

1.3 TruePAL Ecosystem

The TruePAL ecosystem consists of 3 parts:

- **TruePAL Desktop Application**: this application runs on the vehicle's on-board computer, monitoring the vehicle sensors to provide real-time alerts.
- **TruePAL Mobile Application**: this application runs on the first responder's smartphone. It can optionally act as a heads-up display for the desktop application, and provides extra capabilities like a chatbot, hazardous-material sign identification, and more.
- **The Database**: every TruePAL agent communicates data with a global database. This allows for communication between vehicles and devices equipped with TruePAL.

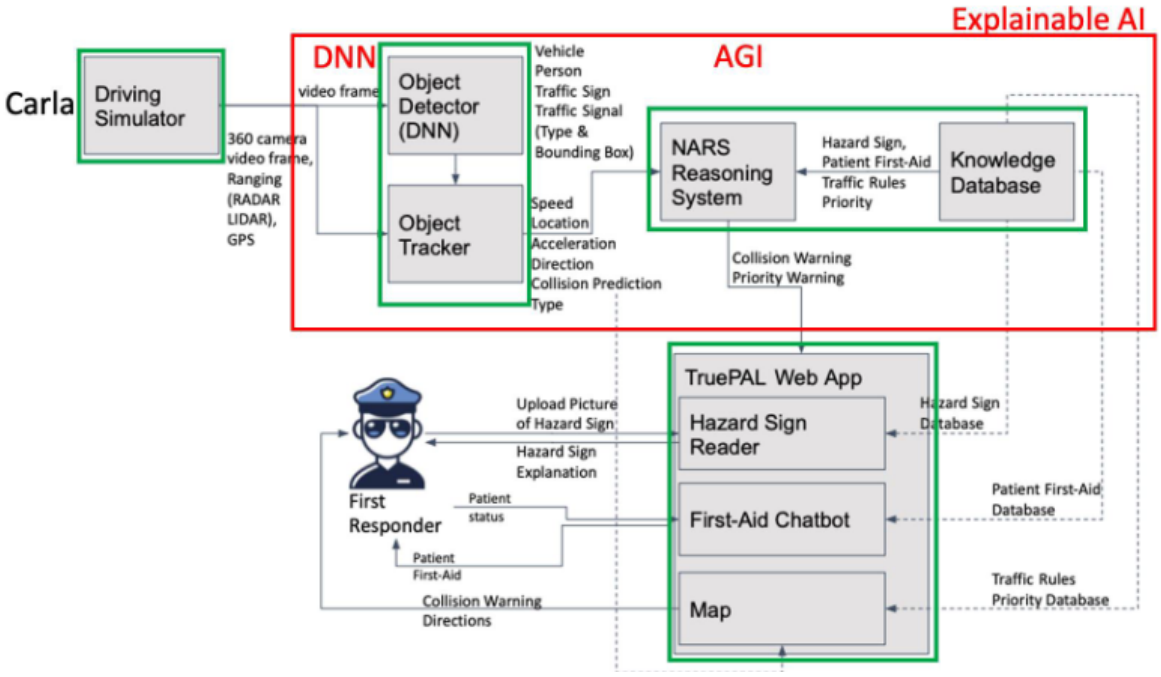


Figure 1: The entire TruePAL ecosystem.

These components interact to provide First Responders with mobile computer tools and a certain level of automated situational awareness around their vehicle.

2 TruePAL

The TruePAL Desktop Application was the primary aspect of TruePAL, which runs on the vehicle's on-board computer. The application was an integrated AI system; the process was controlled by a centralized AGI system, NARS, which uses other algorithms as tools to pre-process the information which streams from the vehicle's sensors (e.g., camera, RADAR, etc.). Raw sensor information was converted into symbols and sentences upon which NARS can reason. NARS' goal was to provide automated situational awareness around the entire vehicle, by alerting the first responder *before* a dangerous situation occurs. The first responder may then react to the alert (e.g., by braking) to prevent injury or death.

TruePAL agents continuously communicate GPS information to and from a centralized database. The NARS system pools all of this information and processes it using reasoning. NARS selectively triggers motor operations when it detects danger, providing situational awareness and danger alerts to the First Responder.

2.1 System Overview and Module Integration

Figure 2 shows the high-level software architecture of the Desktop Application. The Desktop Application consists of various modules. All information flows through these modules to the AGI module, which acts as the central controller of TruePAL. Read *Figure 2* following the arrows from left to right: the RADAR and camera sensors (orange) feed information into the object detector (red) and tracker (blue) modules which keep track of objects in the surrounding area. These tracked objects are fed as updates into NARS (green), which also takes into account information pulled from the database and mobile phone's GPS sensor.

Each module runs independently from one another. Rather than passing sensor data the full length down a sequential pipeline, which might introduce lag (which is unacceptable in a real-time application such as this), each module runs independently and simply offers up their output for grabs by other modules. In this way, the speed of the application was only bottlenecked by the slowest module, rather than running sensor data through the entire pipeline at every single timestep. Consider a Module B which requires data from Module A; Module B simply grabs the most recently completely-processed data offered up by Module A, even if Module A is in the middle of processing a new set of sensor data.

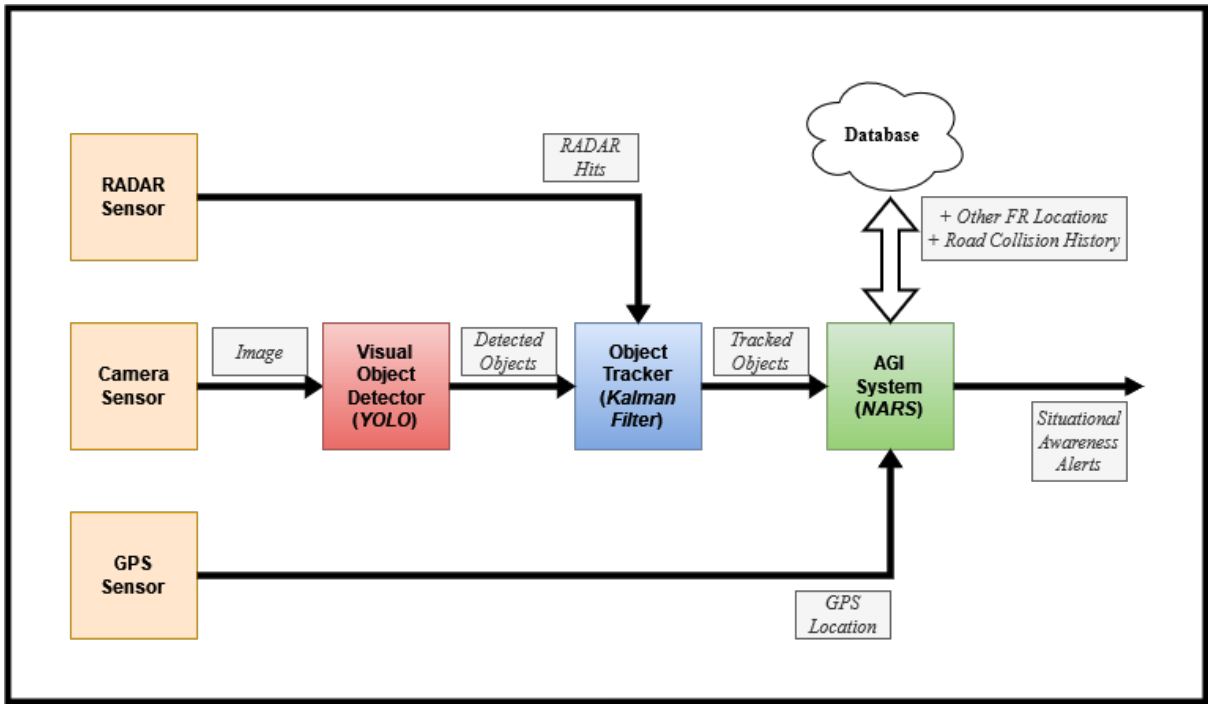


Figure 2: The integrated modules of the TruePAL Desktop Application's architecture. Data flow between components is labeled on the arrows.

Each module runs on a separate process (thus each having its own set of resources), accomplished using Python's "*multiprocessing*" module.

When TruePAL detected danger, an alert sign was displayed visually on the screen, and the text explaining the alert was also displayed. The alert was also delivered via text-to-voice notification to the user's cell phone.

2.2 Driving Simulator

The *Car Learning to Act* (CARLA) simulator [Dosovitskiy et al., 2017] is an open-source 3D simulator designed for experimenting with autonomous vehicles. CARLA provides a variety of functionalities for developers to use, including maps of various towns, vehicle types, NPC vehicle behaviors, types of sensors, etc. and a Python API allowing external programs (such as TruePAL) to communicate with the CARLA application.

The CARLA application consists of two parts: 1.) a *server*, which is the ultimate source of truth regarding the simulated world and, 2.) a *client*, which is an application that allows the user to drive a vehicle around the simulated world. The TruePAL program was attached to a CARLA vehicle, and the user could drive the vehicle around the simulation. The vehicle has a set of front-facing and back-facing sensors. These sensors are a *camera* (with a wide field of view) and *RADAR* (simulated by raycasting). Additionally, the vehicle was equipped with a *GPS* sensor (simulated simply by using the vehicle's location in the virtual world).

A supplemental application to CARLA called *Scenario Runner* was used to build so-called "scenarios" in CARLA — effectively, scripted events. In these scenarios, vehicles are in certain locations and move in certain ways as specified by the script. We built one scenario for each of our test cases.

2.3 Object Detection and Tracking



Figure 3: YOLOv4 was trained on our custom dataset to detect the vehicles, pedestrians, and traffic lights in this image.

In order to monitor for threats, TruePAL had to perceive the objects in its surrounding environment. The environment was sensed using 2 types of sensors: camera and RADAR, which were attached to the front and back of the vehicle. These sensor data were pre-processed using special-purpose algorithms in order to detect objects. Then, the results were entered into the AGI system to make a holistic analysis.

2.3.1 Visual Object Detection in Camera Feed

The camera sensor data was processed using You-Only-Look-Once version 4 (YOLOv4) [Bochkovskiy et al., 2020]. YOLOv4 was a state-of-the-art deep neural network architecture for detecting objects in images, achieving excellent accuracies yet performing fast enough to work in real-time applications. It takes as input a single image. It outputs a set of bounding boxes. Each bounding box contains an object that was detected in the image.

We trained a YOLOv4 network using the Microsoft *Common Objects in Context* (COCO) 2017 dataset [Lin et al., 2014], in order to detect a wide range of object classes (80, to be exact). Then, we further refined the model on our own custom-generated CARLA dataset. This allowed the model to better recognize simulated objects: in particular, vehicles and traffic lights.

YOLOv4 was continuously fed frames from the camera feed. Once the algorithm was done processing a frame, the next most up-to-date camera frame was input into it. The outputs of YOLOv4 were offered up for the Object Tracker to process.

Additionally, a separate algorithm [Vaishnav, 2020] using Canny Edge Detection with Hough Transform was implemented for detecting the lane markings on the road. For the results, see *Figure 12*.

2.3.2 Object Detection in RADAR feed

CARLA simulated RADAR sensor using raycasting. Raycasting works as follows: from a certain point in space, an imaginary ray is cast outwards in a certain direction. The ray stops when it hits an object, similarly to a RADAR wave. As such, if many of these rays are shot outward in many directions, raycasting can be used to simulate a RADAR. RADAR detections were offered up for the Object Tracker to process.

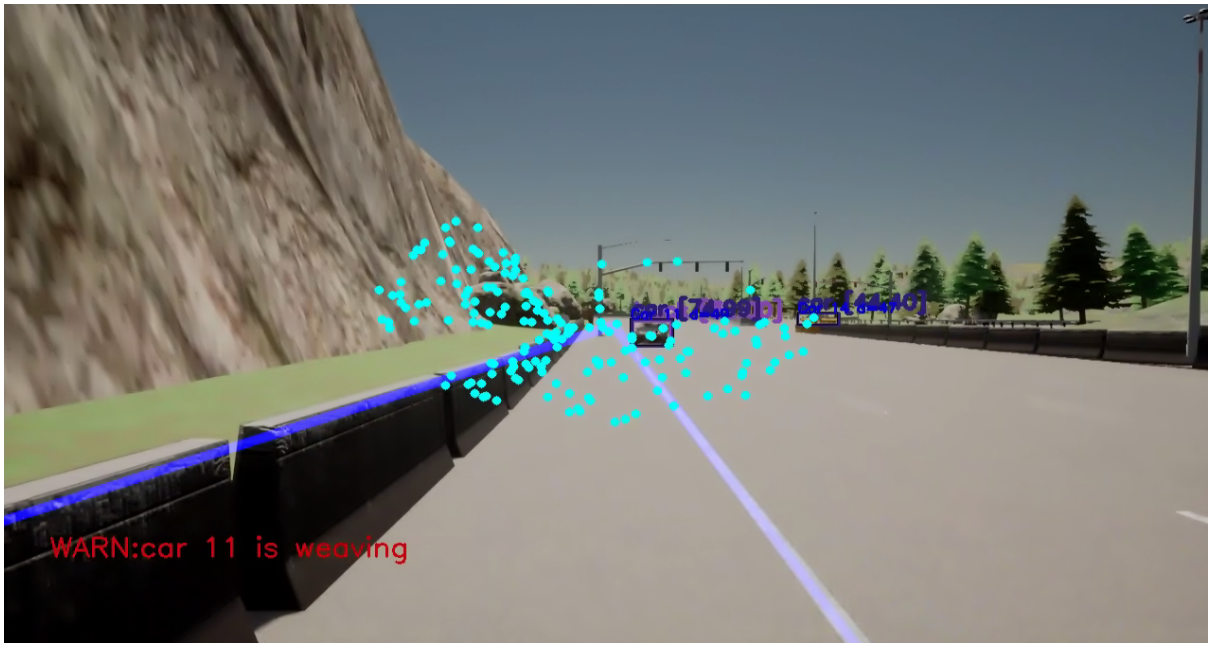


Figure 4: A car is tracked using RADAR and camera. TruePAL monitors for dangerous weaving between lanes, and determines if the car is on a trajectory to cross the lane markings.

2.3.3 Object Tracking

The detected objects are passed to the Object Tracker module. This module estimates the speed, movement velocity, and angle of objects, as well as to assign an instance ID to the object instance which remains stable across time / across multiple frames. The camera was mounted on the front of the first responder vehicle, not fixed in space; hence, the speed of the FR vehicle (to distinguish own movement from object movement) and the distance to detected objects (as estimated by RADAR) was taken into account by the Object Tracker to reach higher accuracies of object tracking. This tracking approach can be considered a direct extension of Simple Online and Realtime Tracking (SORT) [Bewley et al., 2016], using a Kalman filter for tracking, but with a linear dynamics model which also takes the first responder's vehicle speed and depth values (obtained by RADAR and LIDAR) into account in addition to the bounding box coordinates. Here, any matches with a depth difference above a certain threshold are rejected, which significantly improves the accuracy over a purely screen-space tracking approach.

2.4 Non-Axiomatic Reasoning System (NARS)

Non-Axiomatic Reasoning System (NARS) [Wang, 1995, Wang, 2013] is an advanced and innovative reasoning system that has general-purpose ability: a NARS agent has the capacity to learn any knowledge during its lifetime, and pursue any goal. This is achieved via real-time reasoning and learning.

In the past, NARS has primarily been used in reasoning problems that do not critically require a real-time response or contain such components. To advance the NARS project, we have been recently focusing on real-time applications, such as using NARS for video analysis and robot control. Major progress was made using a fixed-location camera in the "smart city" domain [Hammer et al., 2021], where the system was mounted to a traffic light, and monitored a video feed to detect various traffic anomalies including "jaywalking pedestrian", "cyclist in danger of car", etc.

Now, with the TruePAL project, we experimented with placing NARS in an even more complex situation: monitoring a video feed, but attached to a moving vehicle. We use OpenNARS for Application (ONA) [Hammer and Lofthouse, 2020], a NARS implementation which was created specifically for working in real-time applications; the ONA system works differently from the traditional NARS implementations, such as OpenNARS [Hammer et al., 2016], though ONA still operates using the same logic (Non-Axiomatic Logic) and general principles.

Event encodings The inputs to NARS (at each time step) are the following event encodings. Each instance which is paired with its class ID (e.g., truck, bike, pedestrian, first responder vehicle, etc.).

- The instance is approaching the first responder vehicle

```
<(ClassID * InstanceID) --> [ApproachingOrLeaving]>.
```

- The instance is in a certain direction

```
<(ClassID * InstanceID) --> [Direction]>.
```

- The instance is exceeding a certain speed

```
<(ClassID * InstanceID) --> [aboveNumberMPH]>
```

- The instance is inside the first responder's traffic lane

```
<(ClassID * InstanceID) --> [InOrOutOfLane]>
```

- The first responder is approaching an intersection with a certain traffic light color.

```
<(intersection * color) --> [approaching]>
```

- The first responder is approaching an intersection, where a different first responder is also expected to cross now.

```
<(intersection * color) --> [expectedFR]>
```

- General risk assessment of the intersection by the reasoner, based on historic accidents and reasoning which utilizes learned temporal relationships.

```
<(intersection * color) --> [risky]>
```

Background knowledge The background knowledge consists of critical situations under which a warning should be triggered:

- Generate a warning in case of a risky intersection to avoid collision:

```
<(<(intersection * #1) --> [risky]> &/ ^warning)
  =/> (! <{SELF} --> [collided]>)>.
```

- Generate a warning if another first responder is expected to cross the same intersection

```
<((<(intersection * #1) --> [approaching]> &/
   <(intersection * #1) --> [expectedFR]>) &/ ^warning)
     =/> (! <{SELF} --> [collided]>)>.
```

- Generate a warning if an entity is approaching the first responder down the center:

```
<(<(#entity * #id) --> [approaching centered]> &/ ^warning)
  =/> (! <{SELF} --> [collided]>)>.
```

- Roadside parking risk warning, from a vehicle which quickly approaches the first responder vehicle in the lane it is parked in:

```
<(<(car * #nr) --> [fast inLane]> &/ ^warning)
  =/> (! <{SELF} --> [collided]>)>.
```

- If we are speeding above a certain KMPH while approaching a red light intersection, then a warning will lead the officer to go below KMPH:

```
<((<{SELF} --> [aboveKMPH]> &/
   <(intersection * red) --> [approaching]>) &/ ^warning)
     =/> <{SELF} --> [belowKMPH]>>.
```

Unlike many modern AI systems, NARS continuously learns while it operates. We used this "speed limit" warning to showcase NARS learning capabilities. Although we provided NARS with a reasonable value of the speed limit (35 miles per hour), it is also possible to allow NARS to learn the first responder's personal speed limit. The process is as follows: The warning by TruePAL will be ignored by the first responder if the speed limit set by TruePAL is set at too low of a speed. Therefore, the implication statement with too low of a speed limit will receive negative evidence when ignored. TruePAL will make a prediction that the first responder will slow down; this prediction will not be confirmed (the first responder will ignore the warning, and continue to drive above K miles per hour). Hence, that piece of knowledge will not be used by TruePAL anymore after a few occasions; thus, TruePAL has learned.

The other previous implications are of course learnable on the fly, and can be supplemented with additional observed relationships and background knowledge, though so far only the adaptation of this speeding warning behavior has been demonstrated. In this case multiple events such as

```
//the FR vehicle moves above 35MPH
<{SELF} --> [above35MPH]>. :|:
//We are approaching a red intersection
<(intersection * red) --> [approaching]>. :|:
//Generation of warning
^warning. :|:
```

enter the system in succession, in which case the system is expected to update the truth value of the hypothesis

```
<((<{SELF} --> [above35MPH]> &/
   <(intersection * red) --> [approaching]>) &/ ^warning)
     =/> <{SELF} --> [below35MPH]>>.
```

Since the event "moving below 35 MPH" will not be observed when the warning is ignored, this failed prediction will provide negative evidence against the statement. On the other hand, when "moving below 35 MPH" is observed it provides positive evidence for the statement.

Many other pieces of background knowledge corresponding to common-sense knowledge are easily learnable by the system from mere observation, which includes knowledge such as

```
<(<(intersection * red) --> [approaching]> &/
   <(<($type * $id) --> [approaching left]>))
     =/> <($type * $id) --> [leaving right]>>.
```

This knowledge encodes that entities (such as cars and trucks) can pass from our left to our right when we are stopped at a red light intersection (since for those cars, the traffic light is green).

The capability to perceive various specific entities is especially useful for Question&Answering, as the system can be asked about the behavior of specific vehicles, which adds to the explainability of the system. When giving a warning, TruePAL lists the conditions relevant to the warning, including the specific vehicles pertinent to the warning and their behaviors.

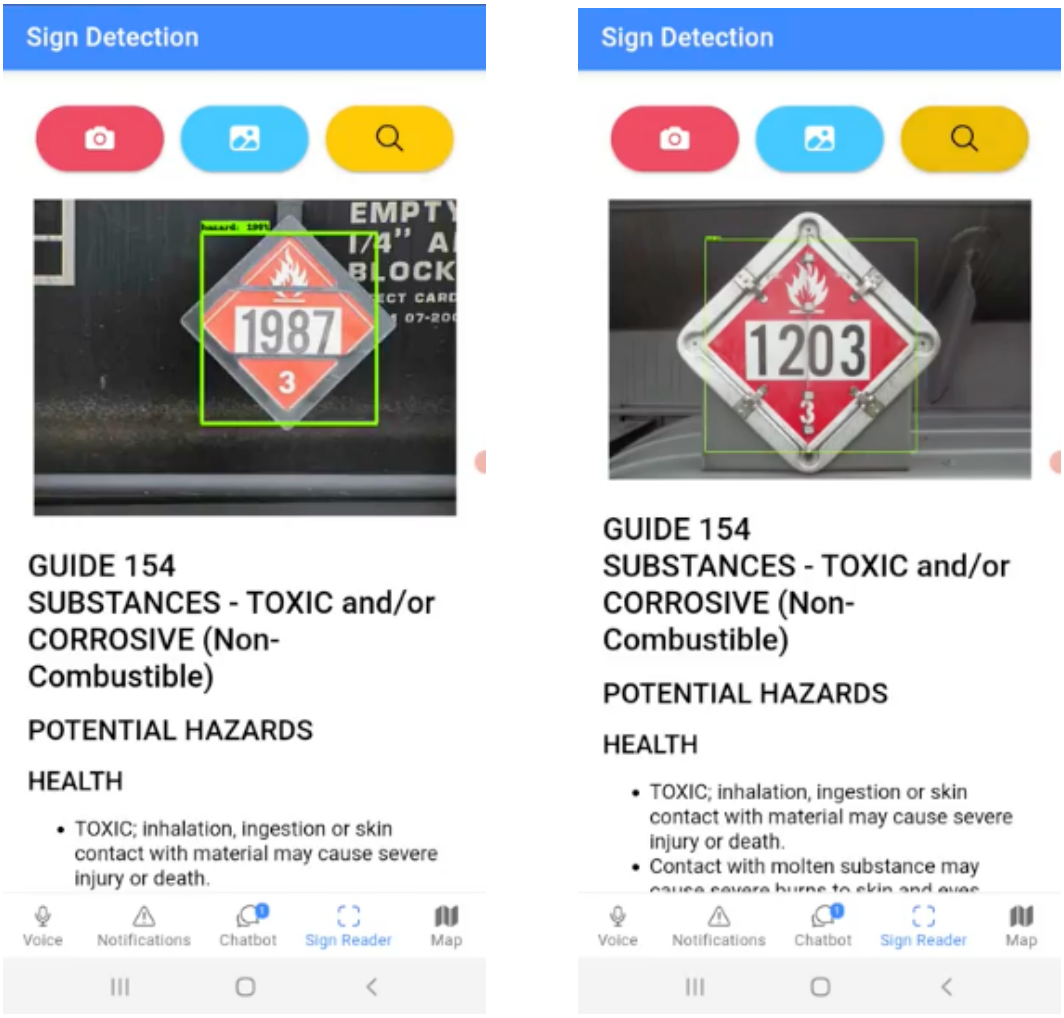
2.5 TruePAL Mobile Application

TruePAL stays with the first responder when they step out of their vehicle, providing different functionalities than the Desktop Application. Each first responder can sign up for their own TruePAL account. A lite version of TruePAL is available as a mobile app, the mobile application was developed using a combination of different technologies e.g. Angular, Ionic Framework, and CapacitorJS. Angular Framework is widely used in developing dynamic web applications. The Ionic Framework, an open-source mobile user interface toolkit, enhances Angular effectively, enabling the creation of cross-platform mobile interfaces. We leverage CapacitorJS, a powerful cross-platform native runtime, to implement our vision across Android and iOS. The platform's plugins enable us to leverage native device features such as GPS and camera functionality on mobile devices, much like Ionic. For the backend we employed TensorFlow and Google's machine learning libraries, along with Python, to provide the core machine learning capabilities, and AWS serves as our cloud infrastructure.

The following section summarizes some of the application's functionality:

- **Hazard Sign Detection:** This function captures or uploads an image of a hazard sign, sends it to the server for detection and localization, identifies the associated U.N. number, and displays the corresponding safety guide for the recognized sign. An example can be seen in Figure 5

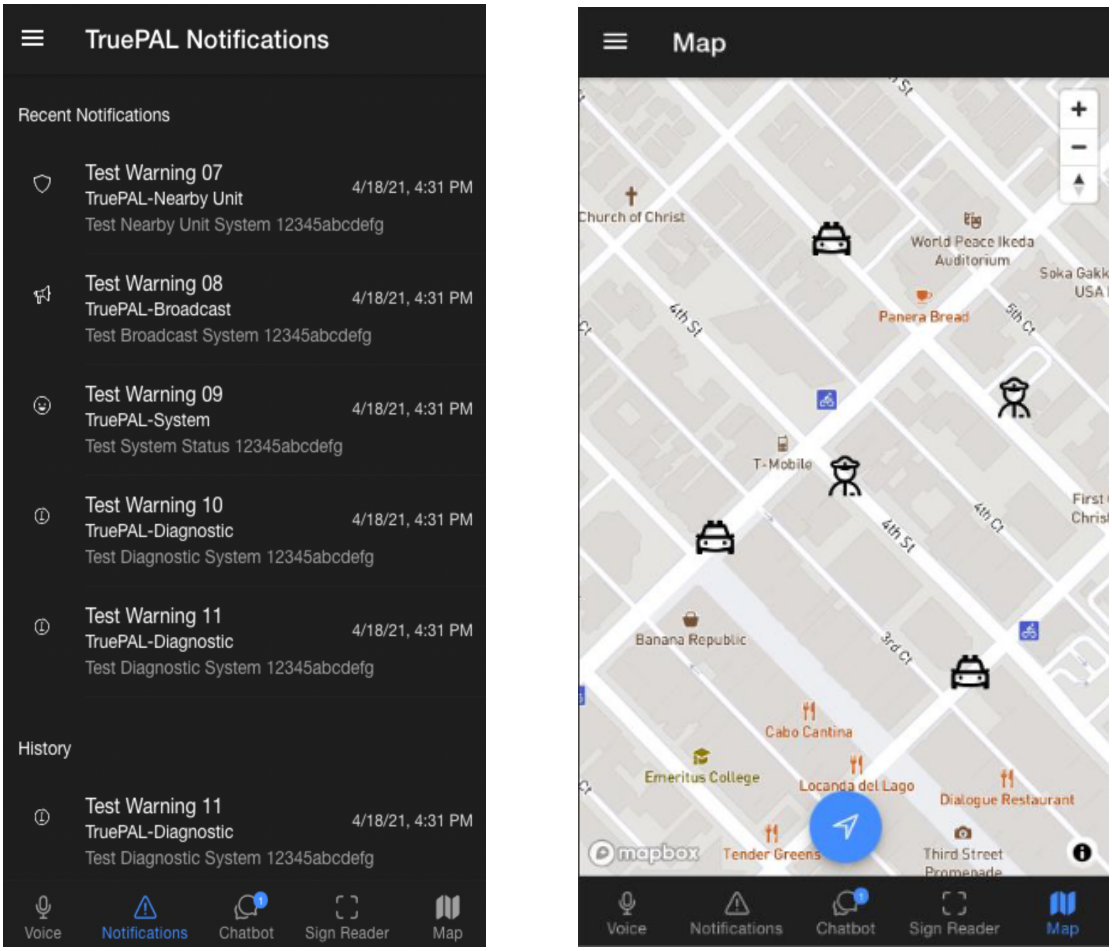
- **Hazardous Materials Handling:** TruePAL employs a Chatbot to facilitate interaction with First Responders (FRs). The Chatbot guides FRs through the necessary steps for safely handling hazardous materials, ensuring proper procedures are followed.



(a) (b)

Figure 5: The mobile app can detect hazardous materials signs in images and provide handling guidance.

The TruePAL App features like **Voice Control** and an **AI Chatbot Assistant** , allowing users to navigate menus hands-free. The AI Chatbot engages with first responders using voice commands, offering support for hazardous material handling, general first aid, and electric vehicle guidance. Additionally, the app provides notification and alert systems that deliver visual pop-up warnings and audible alerts directly on the mobile application. Furthermore, the app incorporates an Interactive Map powered by MapBox as shown in Figure 6b , enabling the display of first responders' current positions, vehicle locations, hazards, obstructions, and active incidents. It utilizes intelligent routing through QuickRoute, considering vehicle size and weight, as illustrated by the sample screens in Figure 6a. QuickRoute optimizes routes based on the weight and physical dimensions of first responder vehicles, ensuring efficient navigation to desired locations.



(a) (b)

Figure 6: (a) The mobile app stores a complete record of all notifications, allowing users to access and examine specific notifications in-depth. (b) The locations of TruePAL devices show up on a map in the TruePAL mobile app.

TruePAL interacts with FRs through its Chatbot shown in Figure 7, offering guidance in various critical scenarios. In the event of an injury at an accident scene, the Chatbot guides FRs through the steps of providing first aid. Similarly, when faced with an electric vehicle incident, TruePAL's Chatbot aids FRs in handling the situation effectively. This intelligent Chatbot, powered by Google Assistant, also stands ready to address basic inquiries related to first aid and hazardous chemicals. Additionally, can answer questions related to the Hazardous Materials (HazMat) sign detection module - previously mentioned -, employing a convolutional neural network to identify official HazMat signs. FRs can capture images of HazMat signs, such as those on damaged vehicles, and upload them for identification, with the app providing relevant information.

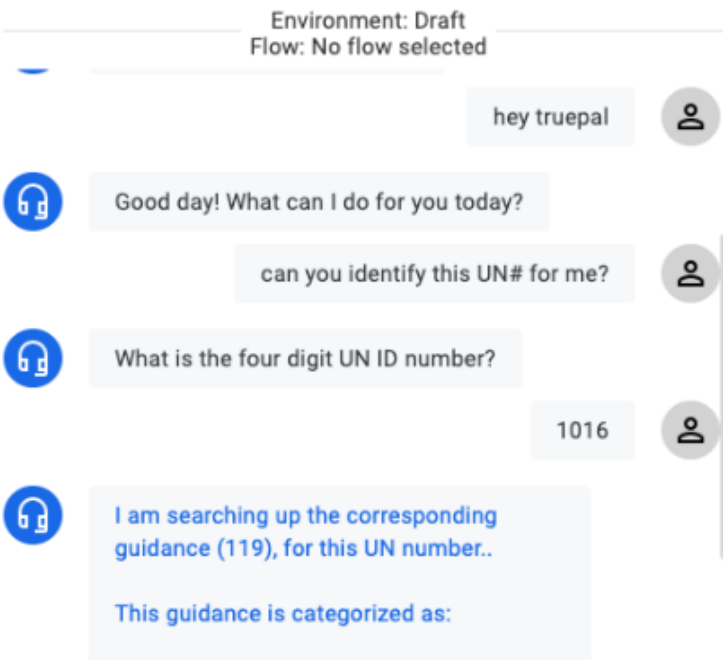


Figure 7: TruePAL Chatbot.

Furthermore, the app incorporates a live map feature, displaying the real-time locations of TruePAL units, thanks to the continuous GPS uploads from each unit to the server, offering enhanced situational awareness.

3 Test Cases

The primary goal of TruePAL was to reliably provide timely warnings in risky situations.

3.1 List of Test Cases

A human user was able to pass 16 test cases at least once when paired with TruePAL. The test cases are partly described here.

There were 13 TruePAL Desktop Application test cases, built in the CARLA simulator:

- **Scenario 1: Intersection**
  - **Scenario 1.1:** Warning is provided to the FR if approaching a high-risk intersection. (Green Light, Siren On)
  - **Scenario 1.10:** Warning is provided to the FR if driving around and approaches multiple high-risk intersections. (Green Light, Siren On)
  - **Scenario 1.2:** Warning is provided to the FR if approaching a red light intersection at a high speed. (Red Light, Siren On)
  - **Scenario 1.3:** Warning is provided if 2 FRs approach the same intersection. Provide priority to one FR, though warn both. (Green Light, Siren On)
  - **Scenario 1.4:** Warning is provided if a civilian car does not slow down and speeds quickly, on a trajectory to t-bone the FR. The warning must be provided with enough time to brake and prevent the collision. (Green Light, Siren On/Off)
  - **Scenario 1.5:** No warning is provided if opposing cars are stopped properly at the intersection, there is minimal vehicle movement, and FRs can move through safely. (Green Light, Siren On)
  - **Scenario 1.6:** Warning is provided to the FR if siren is on and there is a non-yielding nearby vehicle. (Red Light, Siren On)

- **Scenario 1.7:** FR turns left, cutting between two lanes of cars. Warning is provided if a vehicle going straight continues moving on a trajectory to T-bone the turning FR. (T-shaped intersection without lights, Siren On)
- **Scenario 1.8:** FR drives forward slowly into the intersection, which is visually blocked by dense traffic. A warning is provided if a truck is incoming to the intersection at a high speed; the warning is provided despite the truck being previously hidden by a stationary car. (Green Light, Siren Off)
- **Scenario 1.9:** A civilian vehicle drives forward slowly into an intersection preparing to turn left as FR enters the intersection at a very high speed. A warning is provided to the FR to stop. The warning must be provided with enough time to brake and prevent the collision. (Siren On)

- **Scenario 2: Highway Roadside**
  - **Scenario 2.1:** When FR pulls to roadside, TruePAL checks the database on the section of the road. If the vehicle's GPS position is unsafe, TruePAL advises FR to move to a safer section.
  - **Scenario 2.2:** FR is parked on the roadside. TruePAL provides a warning if it detects a vehicle behind, on a straight collision course with TruePAL vehicle.
  - **Scenario 2.3:** FR is parked on the roadside. TruePAL provides a warning if it detects a vehicle behind, displaying dangerous behavior (swerving left and right).

There were 3 TruePAL Mobile App test cases:

- **Scenario 3: Hazardous Materials Truck**
  - **Scenario 3.1:** FR inputs HazMat UN ID number into the chatbot, which responds with the proper information.
  - **Scenario 3.2:** FR takes a picture of a HazMat Placard. The TruePAL app identifies it automatically, and pulls up the proper information.

- **Scenario 4: Chatbot Information**
  - **Scenario 4.1:** FR asks TruePAL chatbot to describe a specific first aid, crashed electric vehicle, or hazmat, protocol.

3.2 Discussion

1. Ability to predict an incoming collision as TruePAL cruises through an intersection. This demands the detection and tracking of the instances based on visual input (via YOLOv4) and RADAR input. The predicted trajectory, together with additional contextual cues (red lights, FR vehicle reducing its speed, etc.), is key to decide whether a collision could happen. Since this trajectory is in 3D space, associating depth information from RADAR to the object detection bounding boxes is crucial. The user successfully passed the test case if they could brake so as to prevent the vehicle collision from occurring.



Figure 8: TruePAL collision detection

2. Ability to warn when another first responder approaches the same intersection, based on GPS location and vehicle-to-vehicle communication. For this to work, ONA queried a database where the location of each first responder vehicle is streamed to in real-time. The querying is efficient, based on a certain distance and direction (via GPS), and occurs when the system perceives an approaching intersection (i.e, traffic lights). The user successfully passed the test case if they could brake so as to prevent the vehicle collision from occurring.



Figure 9: By comparing GPS locations and velocities of nearby first responder vehicles, TruePAL warns that two first responder vehicles are on course to collide with each other in the intersection. This allows the first responder to brake, preventing the collision.

3. Ability to warn of historically risky streets based on a database of past accidents. Such databases are based on police records, and are generally available. Here, we mocked up historical data for car accidents in our simulated town. TruePAL occasionally queries the current road based on the current GPS location. From there, the evidence of riskiness is calculated from the amount of recent incidents and their severity. We check recent incidents only to ensure that also potential changes made to the intersection or city to enhance its safety will be taken into account.



Figure 10: TruePAL warns that the street is historically risky.

4. A speeding warning when approaching a red-light intersection with too high speed. The speed of TruePAL's vehicle is known directly from the speedometer, or could also be computed from GPS coordinates if necessary. The traffic light is detected via YOLOv4, and its color by determining if a mostly (with a small threshold) red/green color is among the K dominant colors, which are determined by applying K-Means Clustering to the pixel values within the detection bounding box. This ensures that the traffic light color is correctly detected even when the bounding box is imperfect (e.g. containing a piece of blue sky or gray metal). Additionally, since multiple traffic light bounding boxes can exist, the evidence for multiple red/green detections are merged into one statement, leaving NARS to pick the traffic light color with the highest amount of positive evidence.

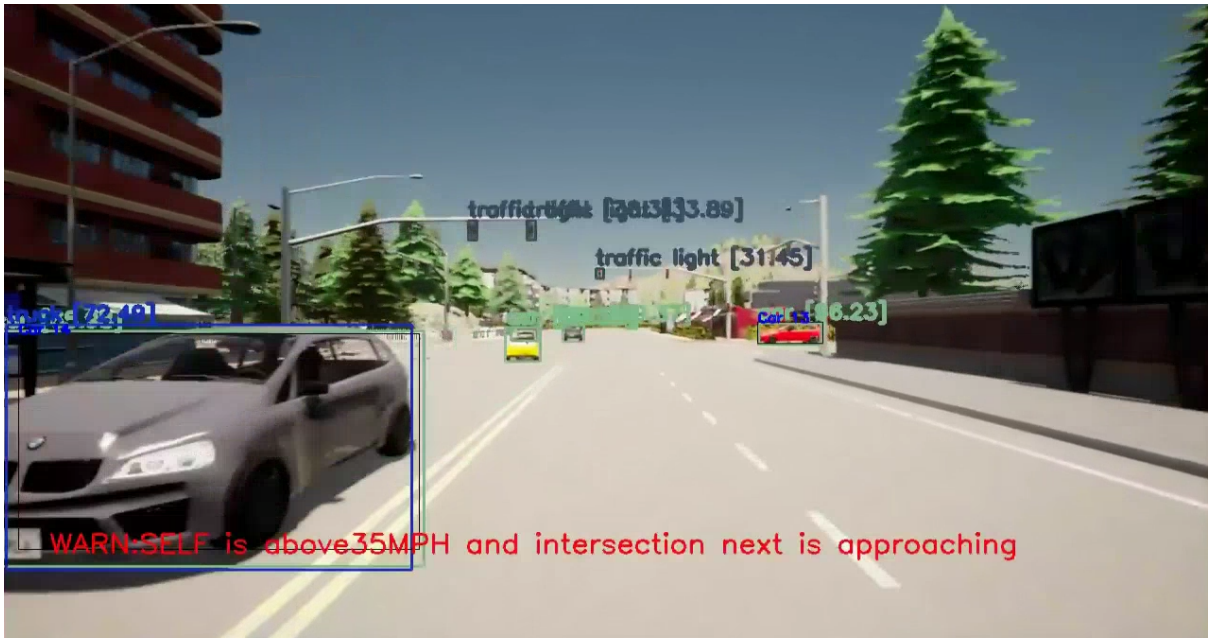


Figure 11: TruePAL speed warning

5. Alert dangerous vehicles behind first responders parked at the side of the road. This includes warnings from speeding vehicles in same lane and collision warning when they drive outside of their lane (especially at the side of the road the FR car is parked at), utilizing both RADAR and visual detection of cars and lanes. If the vehicle switches to the inner lane in time (within acceptable distance safety-margins) then no warning was generated. Otherwise, if the vehicle is too close and over the lane line, a warning is generated. The detection of lanes currently happens via canny edge detection and best-match fitting of two lines via OpenCV. This could be supplemented with a semantic segmentation approach in the future. The line-fitting quality can be taken as truth value's confidence for the line being present.

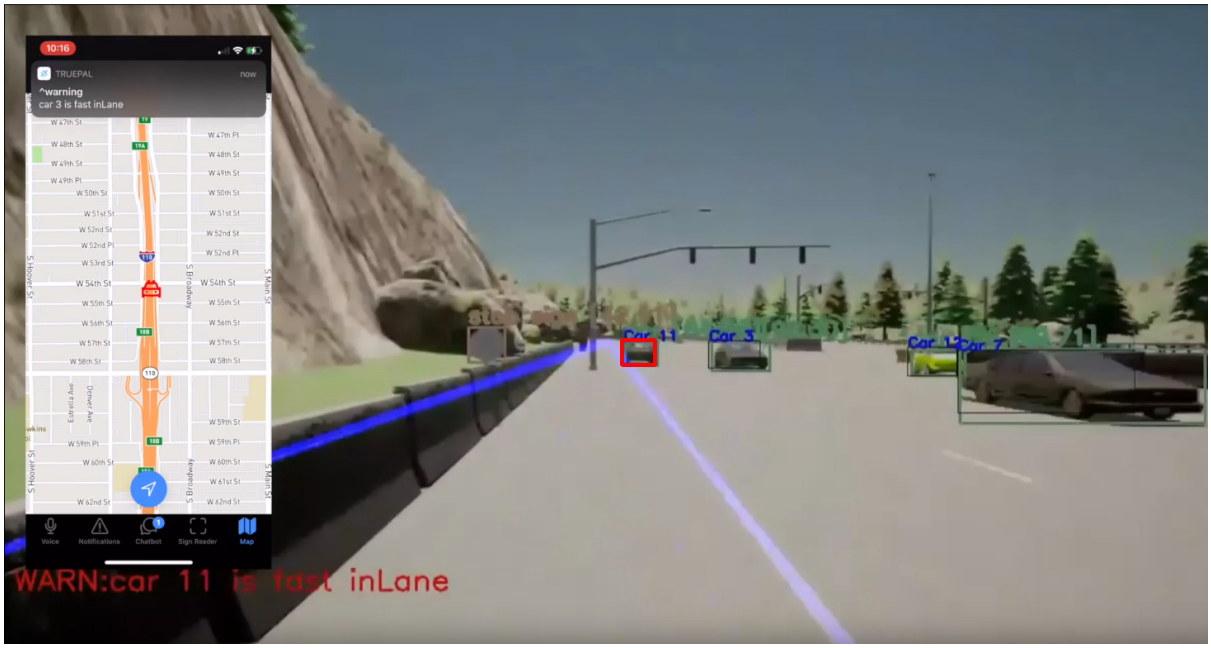


Figure 12: TruePAL roadside safety

4 Conclusion

TruePAL was successfully able to pass a full suite of test cases. In the time-critical cases, TruePAL was able to provide alerts in a timely enough fashion, that the user could brake to prevent the collision, and thus prevent death or serious injury.

In these scenarios, TruePAL has convincingly demonstrated the desired capabilities. The system rarely misses warnings, failing only occasionally in the roadside safety and collision detection case which are more tricky. In these more complex cases, the warning rates were however at least above 80 percent in the preliminary tests, which seems promising, and the explanations given by NARS for the warning consistently made sense for the situation. However, detailed performance metrics have yet to be developed and evaluated, which will be crucial if the project is to migrate to real hardware. Most importantly, the system not only generates warnings but also explains why they were given, which makes the system more trustable to a human user. Also, due to its evidence tracking capabilities, NARS can compensate for certain types of input errors (such as an object incorrectly identified by YOLO during a couple frames) and can give information about how certain it is about a warning it has generated (i.e., by providing the user with confidence values, describing the evidence used to derive the conclusion, etc.).

We see TruePAL as a successful first step to more powerful applications in which NARS plays a major role.

Acknowledgement

This research was carried out under a subcontract titled “Trusted and ExPlainable Artificial Intelligence for Saving Lives (TruePAL) Technology for First Responder Safety” (TP # 82-106589) from NASA Jet Propulsion Lab.

References

[Bewley et al., 2016] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE.

[Bochkovskiy et al., 2020] Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

[Dosovitskiy et al., 2017] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR.

[Fetto Law Group, 2018] Fetto Law Group (2018). Fetto law group: Statistics on emergency vehicle accidents in the united states. https://www.fettolawgroup.com/Personal-Injury-Blog/2018/March/Statistics-on-Emergency-Vehicle-Accidents-in-the.aspx.

[Gray, 2020] Gray, J. (2020). Analysis of crashes involving first responder vehicles. Master's thesis, Embry-Riddle Aeronautical University.

[Hammer and Lofthouse, 2020] Hammer, P. and Lofthouse, T. (2020). 'opennars for applications': Architecture and control. In *International Conference on Artificial General Intelligence*, pages 193–204. Springer.

[Hammer et al., 2021] Hammer, P., Lofthouse, T., Fenoglio, E., Latapie, H., and Wang, P. (2021). A reasoning based model for anomaly detection in the smart city domain. In *Intelligent Systems and Applications: Proceedings of the 2020 Intelligent Systems Conference (IntelliSys) Volume 2*, pages 144–159. Springer.

[Hammer et al., 2016] Hammer, P., Lofthouse, T., and Wang, P. (2016). The opennars implementation of the non-axiomatic reasoning system. In *International conference on artificial general intelligence*, pages 160–170. Springer.

[Lin et al., 2014] Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Doll'a r, P., and Zitnick, C. L. (2014). Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

[Vaishnav, 2020] Vaishnav, D. (2020). https://www.geeksforgeeks.org/opencv-real-time-road-lane-detection.

[Wang, 1995] Wang, P. (1995). *Non-Axiomatic Reasoning System: Exploring the Essence of Intelligence*. PhD thesis, Indiana University.

[Wang, 2013] Wang, P. (2013). *Non-Axiomatic Logic: A Model of Intelligent Reasoning*. World Scientific, Singapore.

[Wang et al., 2020] Wang, P., Hammer, P., Isaev, P., and Li, X. (2020). The conceptual design of opennars 3.1.0. Technical report, Temple University.