A Simulation-Based Benchmark for Behavioral Anomaly Detection in Autonomous Vehicles

Tarang Shah[*,+], John R. Lepird[+], Andrew T. Hartnett[+], John M. Dolan[*]
[*]*Carnegie Mellon University,* [+]*Argo AI*

***Abstract*— A central challenge in the development of autonomous vehicles (AVs) is identifying and responding to anomalous scenarios. Due to the complexity of urban streets, environmental conditions, and the behavior of other actors, AVs encounter out-of-distribution events at a non-trivial rate (often termed *the long-tail problem*). Anomalous situations may be statically perceivable (e.g. an overturned truck, or a mounted policeman) or require nuanced behavioral inference (e.g. vehicles swerving to avoid broken glass on the roadway). Despite the importance of robustly identifying these situations, there are few resources for developing or benchmarking new methods in the research community. Here we introduce a framework for generating *anomalous* and *nominal* scenarios using a domain-specific scenario description language (Scenic) and a driving simulation environment (CARLA) from multiple viewpoints. We introduce four classes of roadway anomalies and establish a benchmark and baselines for the development of new methods. Crucially, our benchmark supports the development of both sensor-based and semantic approaches.** [1]

I. INTRODUCTION

Building autonomous vehicles (AV) involves writing software to correctly handle the wide range of nominal scenarios that are part of everyday driving. The complexity of these typical scenarios is quite high and has dominated research in perception, prediction, and planning for AVs over the past decade. The field has made significant progress and several pilot programs suggest that core functionality is nearing commercial maturity in restricted domains. However, in order for a fully-autonomous system to safely and ubiquitously operate on public roads, the AV must be capable of identifying when its surroundings have diverged from typical or nominal conditions (requiring the AV to come to a stop, seek assistance, or adopt more conservative margins). There is comparatively little research aimed at identifying such scenarios. Furthermore, while there there are important contributions to anomaly detection in driving scenes [1], [2], [3], [4], we are unaware of any common benchmark for aligning or comparing different methods.

One reason for the lack of a common benchmark is that identifying and generating such anomalous scenarios is in itself a significant challenge. One way to mine for such scenarios is to identify disengagement events while the AV software is being tested. A disengagement event occurs when the human safety driver overrides the AV software and takes control of the car. However, this approach is imperfect: anomalies may only be one of the many reasons for a disengagement—others include a known software limitation, or the safety driver might just be taking a break.

Another approach is to use simulation data and set up scenarios that are designed to be anomalous. Testing the software through simulation is a common and cost-effective approach. Simulators such as CARLA [5] and LGSVL [6] are specifically designed for autonomous vehicle testing. Simulation allows us to test various scenarios, including the normal and anomalous ones, in a safe and reliable way. One can define a common set of scenarios and deterministically replicate them on alternate versions of software—measuring progress and identifying regressions. By definition, it is impossible to exhaustively list all anomalies that a vehicle may face. However, we can define a nominal set of conditions, and define *anomaly* as anything outside of those conditions. Tools such as Scenic and CARLA make such a task relatively straightforward.

Anomalies could either be easily observable by visual systems [7], [8], [2] or could also be a feature of the scenario in general, including the behavior and reactions of the agents. For the purposes of this paper, we focus on scenario-level anomalies. In scenario-level anomalies, the anomaly is not constrained to a singular timestamp or object. It is a sequence of events that happen across multiple timestamps. Examples are a vehicle running a traffic light, or a case where a vehicle is avoiding a particular lane and switching lanes at an unexpected time, possibly due to debris that is not visible to the ego vehicle.

In the following sections we discuss different classes of anomalies, an approach to creating such scenarios in a simulator environment, generating and storing these anomalous scenarios, and finally some baseline results on the collected data.

II. RELATED WORK

The major related work areas include behavior classification approaches, existing datasets for autonomous vehicles research and scenario description approaches.

A. Behavior classification.

There are various approaches to analyzing traffic scenarios to determine behavior classification, which could include anomaly detection. The methods used include supervised approaches, such as classifying scenarios [9], [10], unsupervised methods such as clustering [11], [12], [13] and even anomaly/rare event detection methods [14], [1], [4]. However, there is no way to compare the performance of

[1]The code and related documentation for this work is available at https://github.com/t27/carla-scenic-data-collector

these approaches. Most of the methods use different datasets and also have different evaluation metrics and criteria. This makes it difficult to identify which methods work well for a given application. One of the reasons for this is that all the methods rely on different features to represent the scenarios. Some of them need only the trajectories, whereas some of them need additional information, such as the map elements.

B. Datasets

Real-world autonomous vehicle datasets have multiple data types, including raw sensor (LIDAR and camera) data, manually annotated labels and high-confidence machine-annotated data.

Raw sensor data include images (monocular and stereo), LIDAR, radar and/or thermal data. These data target the perception and tracking systems. Some of these datasets also include labeled trajectory data along with the sensor data for sequences of vehicle position for multiple scenarios. These data address the combined challenge of perception, tracking and trajectory prediction. Datasets of this type include Argoverse [15], NuScenes [16], Waymo Open Dataset [17], Lyft Level 5 Dataset [18], Apolloscapes [19], and BDD [20].

PRECOG[21] proposed a dataset and an approach to generating a dataset using a simulator. It aims to serve as a dataset for the motion prediction task and uses a data collection methodology that involves using multiple "autopilot" agents in CARLA. This dataset specifically focuses on the trajectory of agents and not on the higher level scenarios as such. This dataset was also used for comparing other algorithms on the same task, such as in [22], [23].

The broad availability of such datasets has had an immense impact on autonomous vehicle research, particularly in the perception, tracking and prediction domains. These datasets have provided both extensive training sets and common benchmarks for evaluating and comparing approaches. Despite the utility of these datasets for developing perception and prediction systems, they lack rich semantic scene-level labels. Some sets include basic maneuver classes (e.g., lane changes or unprotected turns). Crucially, none of the datasets includes labeled anomalous or rare scenarios. This makes it difficult to use them for the anomaly detection task without spending a lot of effort in reviewing and labeling sequences.

Our work addresses this gap by providing a collection of scenarios that are generated in a simulator. These scenarios are described using a scenario description language and represent some of anomalies that the authors have personally encountered in developing real-world AVs. We also provide "nominal" data which represent cases without any anomalous behaviors. These labeled scenarios also serve as a dataset to test and evaluate behavior classification algorithms, thereby providing a common benchmark to compare various approaches. Also, since these scenarios are generated in a simulator, they enable algorithms to access any number of features from the simulator state that they need.

C. Scenario Description Approaches

GeoScenario [24] and CommonRoad [25] are a few approaches to scenario descriptions.

GeoScenario is a domain-specific language that allows defining scenarios as XML files. It requires non-ego actors to have fixed paths and it requires a very detailed description of the scenario. This is useful for cases where we need to include OSM maps in scenarios.

CommonRoad aims to provide a benchmark for the general motion planning task. It provides specific scenarios and aims to evaluate different planners using a cost metric.

These approaches work well for replicating or building specific scenarios. However, they need a lot of specific information when describing a scenario, such as a fixed location of the actors and their trajectories. This makes it difficult to randomize and generate multiple scenarios of a the same type. We chose to use another Scenario Description Language called Scenic [26] that addresses this and is also well integrated with Carla, our simulator of choice.

Our dataset generation process in CARLA is similar to the CARLA PRECOG dataset. PRECOG was also used for comparing multiple algorithms on the same task. Our provided dataset and benchmark aim to achieve a similar goal, but specifically for anomalous scenarios.

III. Approach

Anomalous data are scenarios that are rare or uncommon. While it is possible to group certain scenarios and label them as anomalous, exhaustively labeling all possible anomalies would be a sisyphean task. As a result, the problem of detecting anomalies can be addressed in both supervised and unsupervised ways, though the eventual solution might involve a mix of both.

We list here a few broad categories of anomalous scenarios, which will be helpful in describing specific anomalies later.

1) Anomalous Scene: This is when there is something unusual about the scene that the vehicle is currently traversing. For example, if there is some debris on the road that the vehicle is able to observe, it would be part of an anomalous scene.
2) Single Anomalous Agent: This is when there is one anomalous agent in the scene (which is possibly breaking some traffic rule) and one or more other agents as well. One agent traveling in the wrong direction would be a single-agent anomaly.
3) Multiple Anomalous Agents: This is when we have various agents performing anomalous behaviors. This could also include the ego vehicle itself being an anomalous agent. An example of this case might include a street race in which multiple agents violating speed limits by a large amount and/or running traffic lights.

One of the most common anomalies that occur on the road are collisions between vehicles. They represent important anomalies for autonomous vehicles to detect and understand. Rather than introducing a set of contrived collisions, we allow collisions to arise from our existing anomalous variations. We find these collisions to be both more diverse and more realistic than the naive alternative. In the future, a road

accident and the resulting behavior of the agents could be included as an anomaly.

We also need a way to represent the scenario that includes various entities, including active entities like cars, pedestrians and traffic lights and passive entities like the map, road signs, lanes, etc. Along with a good way to represent the scenario data, we need a good way to replicate and generate the scenarios. This is essential for us to be able to test and evaluate AV systems. Using real world anomaly data has its own limitations, since we can only have a finite number of such cases and it also requires a lot of human effort to extract and annotate such sequences.

The solution to this issue is the use of simulation. We use a scenario description language to produce dynamic scenarios that run in a simulator. Many capable open source simulators have been developed. In this paper, we use the CARLA simulator [5] to run the scenarios. We also use a domain-specific language, Scenic [26], for modeling and describing scenarios. Scenic allows us to define consistent, dynamic driving scenarios and also introduce randomness into them.

We provide four different classes of anomalous scenarios as Scenic scenarios. We also provide a pipeline to generate any number of random scenarios in CARLA on any compatible map from the Scenic scenario description. This system will produce data that can be further used for any modeling and testing tasks. The scenario data include all the agent states and other information, as well.

Each of the different Scenic files describes one or more broad scenarios. We have a few anomalous scenarios and a scenario which includes "nominal" behavior. The Scenic system allows us to introduce randomization in the roads and locations of the vehicles and relative positions between participating objects in the scenario. We can also enforce constraints to ensure that we can play out the intended behaviors in whatever location is selected for the particular scenario.

By running these scenarios with predefined seeds, we can replicate the same data on any system, as long as base maps and models in the simulator are constant[2]. We can generate multiple scenarios using Scenic and with constant seeds, we can ensure that the initialization and the behaviors depicted in the scenario variants are the same for different users. This consistency is essential for comparing algorithms developed by different researchers.

Using these scenario descriptions and the parameters defined in further sections, we enable a flexible yet consistent way to generate data that can be used to evaluate anomaly detection systems for autonomous driving scenarios.

We also discuss a few baseline approaches to using these data for modeling tasks. These include supervised and unsupervised approaches.

IV. Dataset Description

Each scenario contains static and dynamic entities. In our case the static entities include the map and the road signs. The dynamic entities include the vehicles and traffic lights. Currently, we focus on vehicle-based scenarios and do not include pedestrians.

The static and dynamic aspects of any scenario can be represented as a sequence of states. This sequence contains all agent state information over a given time interval. The state information can include the positions, velocities, accelerations, current light color, etc. Certain attributes might be more relevant for different types of agents. For example, for vehicles the positions would keep changing as the vehicle moves, whereas the light color attribute is not relevant at all. For traffic lights, the light color will keep changing, whereas the position would be constant and might only be useful for identifying if the traffic light is relevant for other nearby agents. This results in a list of agent states at multiple timestamps, and hence gives us a consistent way to represent a scenario. The data at each timestamp are referred to as a "frame", and each frame contains the states of multiple agents. The general hierarchy for this can be seen in Figure 1.

We also describe it symbolically as follows: for K agents, over N timestamps, the scenario is represented as

$$[[\mathbf{A}^1_{t_0}, \mathbf{A}^2_{t_0}, ..., \mathbf{A}^K_{t_0}], ..., [\mathbf{A}^1_{t_N}, \mathbf{A}^2_{t_N}, ..., \mathbf{A}^K_{t_N}]]$$

where $\mathbf{A}^j_{t_m}$ represents the state vector of *Agent* j at timestamp m. N represents the total number of frames (timestamps), and based on the scenario, the interval between two timestamps can be chosen based on the capture frequency. For example, the benchmark data are captured at 5Hz and if the total scenario length is 5 seconds, in this case N would be $5 \times 5 = 25$. Another point of note is that there could be agents that appear later in the scenario or disappear at some time in the scenario. This would depend on the "visibility" radius that we chose. So if an agent enters or leaves the visibility radius at a given timestamp they would not be part of the list of agents before or after that timestamp, respectively. In the above representation, if agent k is only present until the timestamp t_9, the state vector A_k for that agent would effectively be an empty vector for t_{10} and beyond. In our implementation, actors in the scenario can be uniquely identified across frames using their id. Also, their name and object type can be used to differentiate between signs, vehicles, debris, etc.

V. Scenario Description

We use the Scenic [26] scenario description language to describe the anomalous scenarios. Scenic is a domain-specific language that is designed to allow for randomization in a scenario's lower-level parameters, such as initial conditions of positions, orientations and object locations. It also allows enforcement of constraints to ensure the scenario satisfies the requirements, such as a visibility constraint - `car1 can see car2`. The constraint definitions and randomization capability allow a unique definition of a scenario and also add variability to ensure each recording can serve as a separate test case. Additionally, Scenic provides the ability to have dynamic behavior and interactions and change the

[2]For our experiments we used version 0.9.11 of CARLA.

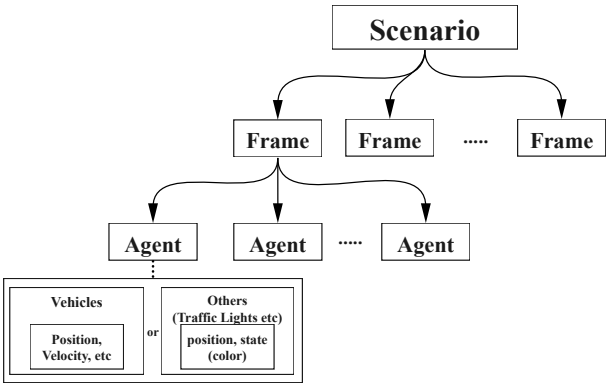

Fig. 1. Hierarchy of the scenario, frames and agents in the data

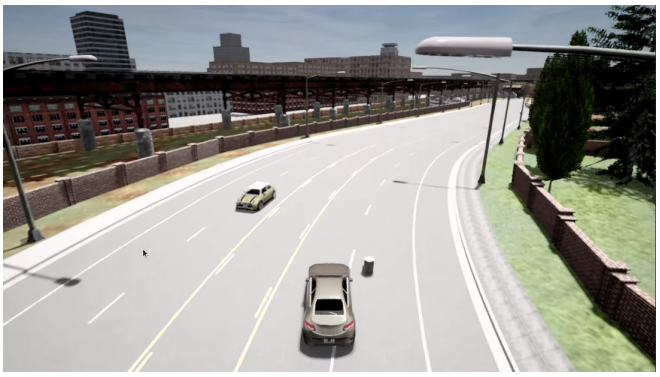

Fig. 3. Snapshot from a debris avoidance scenario in CARLA. This is a Scene anomaly since it is the scene (debris) that is the cause of the anomaly. Here the vehicle facing the debris is the ego vehicle. We also have a spectator vehicle that is driving on the other side. The same scenario data can be transformed such that this spectator vehicle is the ego vehicle, thereby giving us one unique perspective for each vehicle in a given recording.

behavior of a given actor. For example, we can enforce distance constraints between vehicles and then take an action such as stopping the vehicle or changing a lane.

These scenarios are then executed in the CARLA simulator through the Scenic client. We can choose to run the same scenario in multiple randomized configurations using additional parameterization. We record the scenario while it is running in CARLA and then convert these data to a more flexible format (CSV/Parquet) for storing the agent states for each session. Figure 2 shows the higher-level pipeline for this whole process.

A. Types of Scenarios

For the current dataset, we record both anomalous and nominal (non-anomalous) data. The nominal data are based on the default autopilot in CARLA. This is recorded by spawning a random number of agents in the CARLA environment and having them follow all the traffic rules.

For the anomalous data, we consider four specific cases:

1) Road Debris Interaction/Avoidance
2) Oncoming Vehicle Interaction/Avoidance
3) Traffic Light Violations
4) Extreme Speed Limit Violations

These anomalies include multiple vehicles in the scene along with other objects. For each scene, it is possible to consider any agent as the ego vehicle. When selecting an agent as the ego vehicle, we basically transform all the other agents such that the ego vehicle is the observer. This requires translating and rotating the coordinate frames, or it could also mean ensuring that we remove any occluded objects if we are considering a large observation range (e.g., from LIDAR detections). Based on the features chosen, the user needs to ensure that the necessary information is captured from the simulator from the required points of view (PoVs). Also, depending on the agent chosen, the ego vehicle could either be observing or exhibiting anomalous behavior. In cases where a vehicle is facing an external object or vehicle, such as debris or an oncoming vehicle, we also include "spectator" vehicles in our scenario description. Since the ego vehicle can be any vehicle, this allows for a perspective where the ego vehicle is the spectator and must detect the anomaly, even if it isn't specifically a part of it.

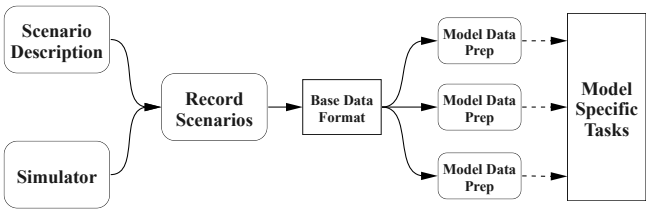

Fig. 2. Pipeline for generating data from the Scenic scenarios and modeling

1) Road Debris Interaction/Avoidance: In this scenario a vehicle is spawned on a road where it encounters and reacts to debris ahead of it. This is an example of an *Anomalous Scene*. The scenario starts at a point when the car is about 15-20 meters away from the debris. The vehicle reacts when close enough to the debris by attempting to turn into an adjacent lane to avoid hitting it. The scenario ends after the car has either successfully avoided the debris, or has hit the debris and stops. For the Scenic description of this scenario, we spawn a trash can in CARLA. This object was chosen for ease of visualization; it is possible to change it to any debris object in the CARLA object library. When storing the information for the scenario, the attributes of the debris are mostly constant. In the current design, the debris object can be accessed using the object name, which starts with `static.*`. Figure 3 shows a debris avoidance scenario in CARLA generated using Scenic.

2) Oncoming Vehicle Interaction/Avoidance: This scenario is quite similar to the previous one, and is an example of a *Single Anomalous Agent*, in which only one vehicle is the root cause of the anomaly and other vehicles are reacting to it. The major difference here is that the primary car is encountering an oncoming car in the same lane instead of a static debris object. The oncoming car is violating lane and road direction rules. Once the primary vehicle is within a

certain distance from the oncoming vehicle, it tries turning into the adjacent lane to avoid the car. The scenario ends when the primary vehicle successfully avoids the oncoming vehicle or collides with it. We can see a snapshot from the simulator during one such case in Figure 4.

3) Traffic Light Violations: This scenario consists of various cars violating traffic lights in the map, and is an example of *Multiple Anomalous Agents*. The primary anomalous behavior here is violating traffic lights (i.e., running a red light). The scenarios also may include scenarios that result from running lights. For example, if two vehicles on perpendicular roads are at the same intersection and one of them runs the red light while the other is traveling normally, they may either collide or slow down to prevent collision with the other vehicle. This scenario contains multiple cars running lights with a very high probability.

4) Speed Limit Violations: This scenario contains vehicles that violate the designated speed in the map by a significant amount. Currently the violating vehicles are configured to have speeds that are 30% higher than the prescribed limit. This scenario also falls into the *Multiple Anomalous Agents* category. Vehicles in the scenario exceed speed limits based on a configurable probability. This even involves violating speed limits at intersections and roundabouts. Due to high speeds, the chance of collisions also increases, especially at intersections and turns. We also combine this scenario with the traffic light violation to produce more anomalous scenarios.

5) Nominal (Non-anomalous) Scenarios: We also have a dynamic scenario description for the nominal, i.e., non-anomalous scenarios. In this scenario, we spawn a fixed number of vehicles in random locations across the map. All these vehicles are assigned simple lane following behaviors. We run this file using Scenic for multiple recordings. This gives us a library of nominal scenario data that can be used for modeling and evaluation purposes.

Fig. 4. Snapshot from CARLA in an oncoming car avoidance scenario. This is a single-agent anomaly case, as we only have one agent that is anomalous (the oncoming vehicle). It is also possible here to transform the scenario into each of the vehicles present in the scene to allow for variability in the scene.

B. Evaluation and Benchmarking

Our benchmark task is straightforward: given a scenario, produce a binary classification that identifies if the scenario is anomalous or not. Because our data generation procedure identifies whether the data came from a nominal or anomalous scenario, we can simply evaluate a proposed anomaly detector using the binary classification performance of the model.

Since we have access to the full simulator, it is possible to use any feature that is available in the simulator for anomaly detection. Some scenarios have been simplified to ensure repeatability and stability during the simulation, such as the case where the "debris" object is a trash can. In such cases, visual approaches become trivial. The goal is to have a generalized approach to anomaly detection. As a result, some of the data in the test set could be of different categories and may even include cases not seen in the training set before.

We provide Scenic files for 2 scenarios and provide Carla Python API-based recorders for the rest. We found some issues with the reliability of the Autopilot while running Scenic scenarios, hence chose to use the CARLA APIs directly for any scenarios that need the Autopilot.

We also provide CARLA log files from the native CARLA recorder which allow us to replay the scenario again in CARLA. Using the log files allows the user to replay and capture features beyond our already provided features. Using more features from the simulator is not necessary, but is encouraged.

The current data capture process involves extracting multiple rounds per scenario type. Each scenario may have a different number of agents. The number of rounds recorded and related data can be found in Table I. Each round involves using a given scenario description, initializing the simulator and running it for a given duration. Scenic ensures the location, distances and the vehicle types are randomized. As a result of this randomization, we get the same core scenario (due to the scenario description) but each round is different. The random seeds in Scenic and in CARLA are set to **27** for the training set. The captured rounds are first stored as raw CARLA logs. The logs are then replayed in CARLA and converted to specific features. The user can choose a set of features to extract and save from CARLA in this step. The anomalous scenarios have a maximum time of 10 seconds but usually last around 7-8 seconds. A debris avoidance scenario ends when the debris-facing vehicle has either avoided the debris or collided with it; depending on the distance of the debris, this may affect the length of a given round. The oncoming vehicle scenario also ends based on similar conditions. The traffic light and speed limit violation scenarios are captured together and usually last indefinitely, as long as there is no collision in the scene, and are capped at 60 seconds. Rounds for the nominal scenario are also capped at around 60 seconds and capture for a given round is stopped if there is a collision (though in nominal scenarios, this is very rare). To adjust for the length, we capture fewer rounds for nominal cases, but each round is generally longer

TABLE I

STATISTICS RELATED TO THE CAPTURED DATA.

Scenario Type	Agents per round	Length(s)	FPS	Frames Per Round	Train Rounds	Test Rounds
Debris Avoidance	2	8-9	6	50	100	30
Oncoming Vehicle	3	8-9	6	50	100	30
Traffic Light and Speed Violations	25	50	5	250	20	30
Nominal Scenarios	25	60	5	300	50	50

in duration than the anomalous scenarios.

For the test dataset, we use a seed of **72** for Scenic and CARLA and we record 30 sequences per anomalous scenario and 10 nominal scenarios. Table I lists the details of the data that were captured for each anomaly type for our experiments for both the train and the test set.

For the benchmark, we use a F1 score metric on the provided test set. The evaluation scripts for the same, along with a detailed description are provided in the accompanying code repository.

VI. EXPERIMENTS AND RESULTS

Modeling

In order to kickstart this benchmark, we implement two simple baseline models. For each approach, we use the scenario data we recorded using the Scenic scenario descriptions. We transform these data into modeling-specific formats and run the models. Since a "scenario" contains multiple frames, and the processing and modeling using the data can be at either the frame level or the scenario level as a whole. We show examples of both methods, although the final evaluation is done for the scenario as a whole.

A. Random Forest-Based Frame Classifier

In this model, we use the scenario data and the scenario-level labels. For each scenario we have a sequence of frames. Each frame contains the positions, velocities, accelerations and angular velocities and accelerations of all vehicles at a given timestamp.

We use a supervised approach and generate a feature vector, which serves as an embedding for each frame. This embedding is a list of parameters that describe the current frame. This includes values such as the maximum velocity, minimum velocity, and maximum/minimum acceleration (See Table II). Each frame is assigned a label based on the label assigned to its parent scenario. We aim to train a supervised binary classifier to predict whether a frame is anomalous or not. We use a collection of simple/non-anomalous scenarios to serve as the "nominal" class. Since this method predicts at the frame level, we need to aggregate all the frame-level information to produce the final scenario-level decision. The simplest could be a threshold on the percentage of anomalous frames in the scenario; for example, if a scenario has more than 20% anomalous frames, it could be labeled anomalous. Currently, we use a threshold of 50%, which corresponds to a majority vote on the results of the frame classification on each frame of a scenario.

TABLE II

FIELDS USED IN THE RANDOM FOREST-BASED FRAME CLASSIFIER. THE X, Y, Z DIRECTIONS ARE IN THE GLOBAL FRAME

{Min, Max} Velocity in {X, Y, Z} direction
{Min, Max} Angular Velocity about {X, Y, Z} Axis
{Min, Max} Acceleration in {X, Y, Z} direction
Number of vehicles

For the training set, we use a distribution as mentioned in Table I. Based on the frame rate (FPS) and the length, on an average, we have around 15,000 frames each for the nominal and anomalous classes.

With a frame classifier, we consider the full scene, without any constraints on the visibility radius. We also don't consider any specific ego vehicles. Given these relaxations, it is expected that the model is very likely to perform very well on this simplified task.

On a total of 30,000 frames, we first ran a random forest classifier and extracted the frame-level accuracy on the validation set. This results in a validation score of around 97%. As described above, this was expected.

For the *test* set, we aim to get scenario-level metrics. We run this frame-level model on the whole scenario and run an aggregation on the results for each frame in our scenario. We use a simple, arbitrary threshold of 50%, which implies at least half of the frames must be true for an anomalous scenario. This gives us an accuracy of 100% for classifying nominal and anomalous scenarios. The aggregation step adds a layer of simplification in our case since we use a lenient score of 50%.

The main goal of this baseline approach is to show an example of a frame-level evaluation approach. We can improve realism by incorporating ego vehicle transformations of the scenario (rotation and translation of the whole scenario to keep the ego at the center). We can also filter the vehicles in a given frame to only include vehicles in the vicinity of the chosen ego vehicle. Some aspects of this are used in the Autoencoder-based approach we describe in the next section.

We are also working on extending the test dataset to include more complex scenarios which would allow more realistic challenges, even for a simplified model such as the one described above.

B. Dynamic Time Warping-Based Autoencoder

This approach is an unsupervised approach. We first generate a Dynamic Time Warping (DTW) [27] map (also known as DTW cost map or cost matrix) for a given scenario that

we describe below. We use the Python version of the cited package. We use this DTW map object and pass it through a convolutional autoencoder. We train this autoencoder on the "nominal" data to reconstruct the input DTW maps. Once the model is trained, we use just the encoder. Using the encoder, we get a single vector for each scenario represented as a DTW map. This embedding is in *N*-dimensional space. Since the original autoencoder was only trained on "nominal" data, the idea is that anomalous data embeddings will be far from the "nominal" data embeddings in the *N*-dimensional space. To produce classification results, we can also use these embeddings as features for a classifier and analyze the results.

DTW map generation process: This is a multi-agent implementation of the idea originally proposed in [12] for two agents. For each ego vehicle in the scene, take the N nearest vehicles in the range of, say, 50 meters from the ego vehicle. If we have more than N vehicles in the vicinity, take the N closest vehicles. For each of the surrounding vehicles, we make a DTW cost map using the ego trajectory and the trajectory of the surrounding vehicle. The length of the trajectory is the width (W) of the DTW cost map. The selected trajectory length is fixed for all vehicles. The total length of the trajectory is chosen to be 25 frames in this implementation. Since the data we have are captured at 5-6 Hz, the length of the scenario is about 4-5 seconds. We then stack these DTW maps to produce a 3D tensor. This 3D tensor will be WxWxN, where W is the dimension of the DTW map and N is the maximum number of vehicles considered (25 and 10, respectively, in our implementation). If we have fewer than N vehicles, we pad the tensor with zeros, i.e., the last few maps will be zeros of size WxW. This tensor is then used as the input to the autoencoder. For this method, we need more than 2 actors in the scenario, as DTW requires atleast 2 objects to build a cost map. All the scenarios we provide include more than one actor.

The autoencoder architecture is a simple convolutional neural network. The bottleneck embedding vector size is 1024. A detailed architecture can be seen in Table III.

We train an autoencoder on a sample of 2000 nominal scenarios from the dataset. The autoencoder was trained for around 400 epochs. The encoder was then used to generate embeddings on a test dataset that included both the nominal and the anomalous data.

The test data distribution for this included a sample of 600 nominal and 600 anomalous scenarios. For each of the 1200 scenarios, we calculated the 1024-dimensional embedding vector. These 1200 scenarios were used to train a classification model which incorporated a t-SNE [28] projection that converts the 1024 dimensional vector to a 2-dimensional vector. The t-SNE plot can be seen in Figure 5. We can see that the anomalous and nominal data are somewhat clustered and separated in the plots, though non-linearly. Hence, we then use a small neural network to classify this 2-dimensional projection as anomalous or nominal. We also trained a similar neural network to classify the raw 1024-dimension embedding. When trained on a 70:30 split of the 1200 embeddings, the t-SNE based neural network model gave a higher accuracy score of 83.43%, whereas using the 1024-dimension embedding instead of the t-SNE projections in a similarly trained neural network performed poorly, giving an accuracy of around 59.67%. One drawback of t-SNE is that it depends a lot on the initialization. With random initialization the results vary from 70 to 90% accuracy. Instead of a random initialization, we use a PCA initialization for t-SNE, which gives more consistent, repeatable results.

Based on the above results we are further working to run experiments on larger samples. Also, variation in the test set is something that we are working on, as that is an essential aspect of anomalous scenarios.

TABLE III
THE AUTOENCODER ARCHITECTURE USED TO ENCODE THE DTW MAPS

Layer	Input Shape	Output Shape	Kernel Shape
Encoder			
Conv2d	[1, 10, 25, 25]	[1, 16, 25, 25]	[10, 16, 3, 3]
LeakyReLU	[1, 16, 25, 25]	[1, 16, 25, 25]	–
Conv2d	[1, 16, 25, 25]	[1, 32, 13, 13]	[16, 32, 3, 3]
LeakyReLU	[1, 32, 13, 13]	[1, 32, 13, 13]	–
Conv2d	[1, 32, 13, 13]	[1, 64, 13, 13]	[32, 64, 3, 3]
LeakyReLU	[1, 64, 13, 13]	[1, 64, 13, 13]	–
Conv2d	[1, 64, 13, 13]	[1, 128, 7, 7]	[64, 128, 3, 3]
LeakyReLU	[1, 128, 7, 7]	[1, 128, 7, 7]	–
Conv2d	[1, 128, 7, 7]	[1, 1024, 1, 1]	[128, 1024, 7, 7]
LeakyReLU	[1, 1024, 1, 1]	[1, 1024, 1, 1]	–
Decoder			
ConvTranspose2d	[1, 1024, 1, 1]	[1, 128, 7, 7]	[128, 1024, 7, 7]
LeakyReLU	[1, 128, 7, 7]	[1, 128, 7, 7]	–
ConvTranspose2d	[1, 128, 7, 7]	[1, 64, 13, 13]	[64, 128, 3, 3]
LeakyReLU	[1, 64, 13, 13]	[1, 64, 13, 13]	–
ConvTranspose2d	[1, 64, 13, 13]	[1, 32, 13, 13]	[32, 64, 3, 3]
LeakyReLU	[1, 32, 13, 13]	[1, 32, 13, 13]	–
ConvTranspose2d	[1, 32, 13, 13]	[1, 16, 25, 25]	[16, 32, 3, 3]
LeakyReLU	[1, 16, 25, 25]	[1, 16, 25, 25]	–
ConvTranspose2d	[1, 16, 25, 25]	[1, 10, 25, 25]	[10, 16, 3, 3]

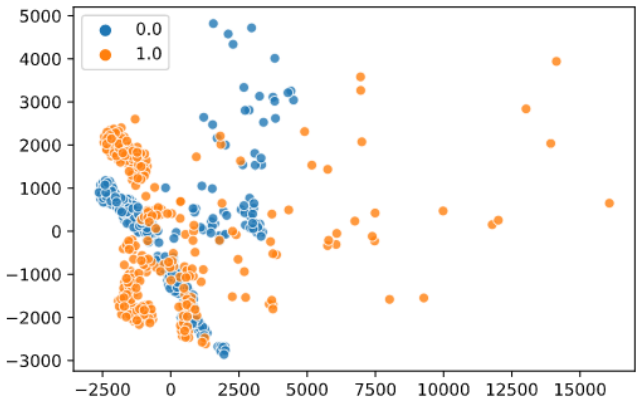

Fig. 5. t-SNE Projection of the embeddings of the sampled subset. The orange dots signify anomalous scenarios and the blue ones signify nominal scenarios.

VII. CONCLUSION

In this work, we address the challenge of anomalous scenarios in autonomous driving systems and the lack of a common benchmark to compare different algorithms. To do

so, we propose a method of generating various anomalous scenario data in a simulator. This is achieved by using a dynamic scenario description language and integrating this with the simulator. We provide a pipeline to take the scenario description, run it in the simulator and capture a set of features from the simulator state information. These scenarios can be used for evaluating and benchmarking various algorithms using the described rules.

We also suggest a few baseline modeling approaches which use different features to model the scenario and predict anomalous scenarios using supervised and unsupervised approaches. While our approaches are rudimentary, we invite others to accept our challenge and develop novel and creative ways for identifying out-of-distribution events in an AV.

REFERENCES

[1] M. O'Kelly, A. Sinha, H. Namkoong, J. Duchi, and R. Tedrake, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," 2019.

[2] S. Bai, Z. He, Y. Lei, W. Wu, C. Zhu, M. Sun, and J. Yan, "Traffic anomaly detection via perspective map based on spatial-temporal information matrix," in *CVPR Workshops*, 2019.

[3] D. J. Fremont, E. Kim, Y. V. Pant, S. A. Seshia, A. Acharya, X. Bruso, P. Wells, S. Lemke, Q. Lu, and S. Mehta, "Formal scenario-based testing of autonomous vehicles: From simulation to the real world," in *23rd IEEE International Conference on Intelligent Transportation Systems (ITSC)*, Sept. 2020.

[4] J. Norden, M. O'Kelly, and A. Sinha, "Efficient black-box assessment of autonomous vehicle safety," 2020.

[5] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

[6] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta, E. Agafonov, T. H. Kim, E. Sterner, K. Ushiroda, M. Reyes, D. Zelenkovsky, and S. Kim, "Lgsvl simulator: A high fidelity simulator for autonomous driving," 2020.

[7] A. Ben Mabrouk and E. Zagrouba, "Abnormal behavior recognition for intelligent video surveillance systems: A review," *Expert Systems with Applications*, vol. 91, pp. 480–491, 2018.

[8] A. A. Sodemann, M. P. Ross, and B. J. Borghetti, "A review of anomaly detection in automated surveillance," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1257–1272, 2012.

[9] R. Gruner, P. Henzler, G. Hinz, C. Eckstein, and A. Knoll, "Spatiotemporal representation of driving scenarios and classification using neural networks," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1782–1788, 2017.

[10] L. Hartjen, R. Philipp, F. Schuldt, B. Friedrich, and F. Howar, "Classification of driving maneuvers in urban traffic for parametrization of test scenarios," in *9. Tagung Automatisiertes Fahren*, Lehrstuhl für Fahrzeugtechnik mit TÜV SÜD Akademie, 2019.

[11] S. Li, W. Wang, Z. Mo, and D. Zhao, "Cluster naturalistic driving encounters using deep unsupervised learning," 2018.

[12] W. Wang, A. Ramesh, J. Zhu, J. Li, and D. Zhao, "Clustering of driving encounter scenarios using connected vehicle trajectories," *IEEE Transactions on Intelligent Vehicles*, vol. 5, p. 485–496, Sep 2020.

[13] N. Harmening, M. Biloš, and S. Günnemann, "Deep representation learning and clustering of traffic scenarios," 2020.

[14] J. A. Barria and S. Thajchayapong, "Detection and classification of traffic anomalies using microscopic traffic variables," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 695–704, 2011.

[15] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," 2019.

[16] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," 2020.

[17] "Waymo open dataset: An autonomous driving dataset," 2019.

[18] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet, "Lyft level 5 perception dataset 2020." https://level5.lyft.com/dataset/, 2019.

[19] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The apolloscape open dataset for autonomous driving and its application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, p. 2702–2719, Oct 2020.

[20] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," 2020.

[21] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, "Precog: Prediction conditioned on goals in visual multi-agent settings," 2019.

[22] Y. C. Tang and R. Salakhutdinov, "Multiple futures prediction," 2019.

[23] W. Zeng, S. Wang, R. Liao, Y. Chen, B. Yang, and R. Urtasun, "Dsdnet: Deep structured self-driving network," 2020.

[24] R. Queiroz, T. Berger, and K. Czarnecki, "Geoscenario: An open dsl for autonomous driving scenario representation," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 287–294, 2019.

[25] M. Althoff, M. Koschi, and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 719–726, 2017.

[26] D. J. Fremont, E. Kim, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: A language for scenario specification and data generation," 2020.

[27] T. Giorgino, "Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package," *Journal of Statistical Software*, vol. 31, no. i07, 2009.

[28] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.