SCSG: Real-World Report Augmented Safety-Critical Scenario Generation for Autonomous Vehicles

Nigar Doga Karacik
Technical University of Munich
Munich, Germany
ndoga.karacik@tum.de

Yingjie Xu
Technical University of Munich
Munich, Germany
yingjie.xu@tum.de

Xinyi Li
Technical University of Munich
Munich, Germany
xinyi.li@tum.de

Yingbai Hu
The Chinese University of Hong Kong
Hong Kong, China
yingbai.hu@tum.de

Yinlong Liu *
City University of Macau
Macau, China
ylliu@cityu.edu.mo

Abstract

This study presents an end-to-end safety-critical scenario generation pipeline for autonomous driving vehicles, integrating data-driven, adversarial, and knowledge-based scenario generation techniques. The research utilizes 738 different crash reports, which encompass both traditional crash cases and incidents involving autonomous vehicles. The retrieval augmented generation (RAG) method is employed to convert these reports into structured scenario details, encompassing scenario descriptions, agent behaviors, road geometries, weather conditions, and agent spawn points relative to the ego vehicle. Once the scenario descriptions are established, our code generator module utilizes large language models (LLMs) to transform these structured details into Scenic code. To ensure the generated code functions correctly in the CARLA simulation environment, we have implemented an error correction module that verifies and maintains syntactic correctness. The error-free codes are selected for the risk assessment of the generated scenarios. For the risk assessment, 10 different metrics from SafeBench are used, and our scenarios are compared with the current baseline methods using three distinct ego vehicles pretrained by different reinforcement learning (RL) algorithms. Our generated scenarios demonstrated at least an 8% improvement in the turning obstacle scenario category for all three pretrained reinforcement learning agents. For unprotected left turn, vehicle passing, red light running, and right turn scenarios, our method yielded the best results for at least one out of three RL ego agents. The results indicate that utilizing crash reports, increasing the number of adversarial agents in a scenario, and implementing various weather and road conditions enhance the complexity, variety, and real-world relevance of generated safety-critical scenarios.

*Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Embodied and Safe-Assured Robotic Systems.

1 Introduction

Autonomous vehicles (AVs) aim to reduce traffic fatalities, improve traffic flow, and expand access to mobility. However, their safe and reliable deployment relies on technological advancements and the development of robust testing and validation processes. Current advances in autonomous driving vehicles have created the need for safety-criticality checks, evaluations, and validations.

We can now see SAE Level 2 and Level 3 [1, 2] cars around us. However, it is difficult for humans to fully trust the behaviors of autonomous vehicles to shift to Level 4 or Level 5. One reason for this is that these autonomous vehicles should work properly in any case in the daily traffic with human drivers and other traffic factors such as pedestrians, animals, and other factors affecting the traffic flow. Autonomous vehicles have to be able to perform better than an average human driver in any driving scenario [3, 4]. This creates the need to check how safe the AVs are.

This study proposes a novel safety-critical scenario generation framework that integrates LLMs and real-world crash data to generate Scenic codes. The approach builds on four main pillars:

Real-world crash reports: Using existing crash reports can help us capture critical scenarios while keeping real-world relevance. These reports include weather, lighting, road conditions, number of occupants, injury details, crash case, etc. These help us to generate the scenarios that reflect the challenges faced in real-world traffic. The proposed solution is to use real-world crash reports in textual format and use this as an input to enhance the scenario variety and criticality.

LLMs: They are utilized to create rich, context-aware driving narratives, enabling the exploration of complex and rare edge cases. They serve as a valuable tool for enhancing real-world data, while also summarizing and converting the data into a preferred format.

Probabilistic domain-specific language: Scenic [5] helps us to create agent behaviors, spawn points of the vehicle, select road geometries, and weather conditions in desired distributions, which enhances the creation of a lot of distinct scenarios from a single scenario description.

Risk assessment: It is a crucial aspect of creating safety-critical scenarios. However, there are no standardized metrics to measure whether a generated scenario is considered critical. This study uses Safebench [6] to evaluate criticality across 10 metrics, including collision rates, road completion rates, overall scores, average displacement errors, and other relevant factors.

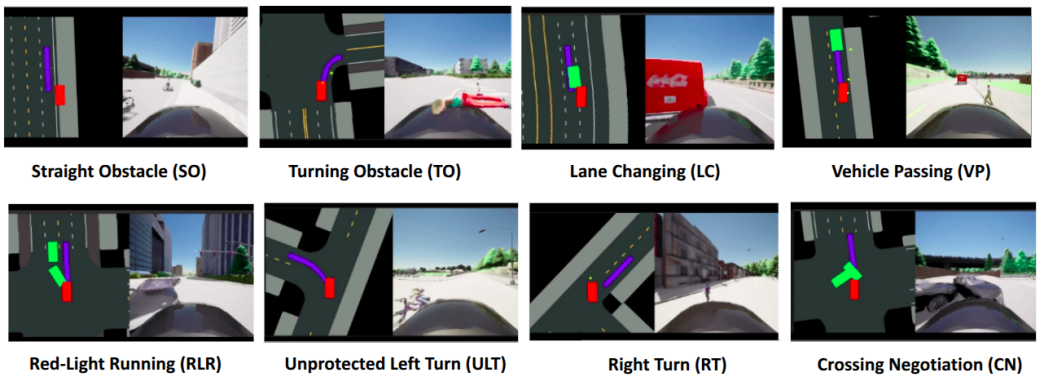

Figure 1: Simulation output examples of the generated Scenic codes for the eight scenario categories.

The proposed solution generates diverse and realistic scenarios for eight different scenario categories, as shown in Figure 1, by integrating the four key elements described. This combination is intended to strengthen the robustness of simulation-based validation, ultimately contributing to the safe and reliable deployment of autonomous driving vehicles.

2 Related Work

To ensure the safety of AVs, we require a large and diverse dataset for testing and validation. As a result, generating driving scenarios for AVs has become a crucial area of research. Numerous studies in this field exist, each focusing on distinct aspects [7, 8, 9].

Recent works such as ChatSumo [10], TTSG [11], and AutoScenario [12] generate traffic scenes from natural language or multimodal inputs. ChatSumo converts textual prompts into simulation-ready networks with summaries of traffic density, travel time, and emissions. TTSG utilizes an LLM-based road retrieval and agent planning pipeline with a pre-built agent database. AutoScenario employs multimodal inputs (text, image, video, GPS) to create safety-critical scenarios. However, none of these approaches evaluate the safety criticality of the generated scenarios.

ScenicNL [13] utilizes publicly available real-world crash reports and applies prompting techniques to create scenarios based on these reports. They do not use the crash reports to enhance the generated scenarios; instead, they replicate the scenarios directly to run them in CARLA using Scenic. In contrast to our approach, they do not include any elements of adversarial behavior in their scenes. This work combines both data-driven and knowledge-based methods for scenario generation.

There are several baselines that use adversarial-based generation and knowledge-based generation techniques. Learning to collide (LC) [14] uses an adaptive scenario generation method using reinforcement learning to train adversarial agents that intentionally cause collisions with the ego vehicle. They manage this by changing the initial conditions of the agents, such as initial velocity, spawn points, and trajectories. AdvSim (AS) [15] generates safety-critical scenarios by utilizing recorded sensor data from self-driving datasets such as lidar point clouds. It applies black box adversarial attacks that perturb actor behaviors and optimize actors' trajectories to increase the risk of collisions, traffic rule violations, and uncomfortable driving behaviors. Adversarial Trajectory Optimization (AT) [16] introduces the first framework for adversarial attacks on trajectory predictions in autonomous vehicles while respecting real-world constraints. This framework features both white-box and black-box attacks, which involve making small, physically feasible adjustments to normal vehicle trajectories. ChatScene [17] uses natural language inputs and generates Scenic code [5]. It takes the input, extracts information, and organizes it in a structured format. Using these descriptions, it retrieves the relevant Scenic code snippets for each code component (behavior, geometry, etc.) to create the complete Scenic script. However, they do not utilize real-world reports, do not consider weather conditions, and have restrictions on the agents involved.

Some studies focus more on evaluating safety-critical scenarios, such as Scenario Runner [18], CommonRoad [19], SUMMIT [20], and DeepDrive [21], From Words to Collisions [22], SafeBench [6]. SafeBench platform offers unique advantages, including knowledge-based and adversary-focused safety-critical scenario generation. It can result in realistic perceptions with customizable scenarios. Additionally, SafeBench is compatible with existing baselines and the CARLA simulation environment.

The main distinctions of our work are as follows:

- We utilize real-world crash reports as an input for LLMs. Not the manually created sentences.
- RAG is implemented to get a structured format for code generation. This structural format includes scenario description, adversarial agent behaviors, road geometry, spawn positions, and weather conditions to enhance the realism.
- Our pipeline includes recursive control for the code generation part, serving as an error corrector to ensure the generation of error-free codes. Generated codes are executable without basic or structural errors after the code correction part.
- Our approach combines data-driven, adversarial-based, and knowledge-based methods for scenario generation. It is data-driven through the use of 738 real-world crash reports that enhance realism and relevance; adversarial-based by defining agent behaviors and interactions that challenge the ego vehicle; and knowledge-based through the Scenic framework, which enforces predefined movement rules for all agents and environmental conditions.

3 Method

The proposed pipeline consists of three components: scenario description generation, code generation, and risk evaluation, as illustrated in Figure 2. Each component is described in detail in the following sections.

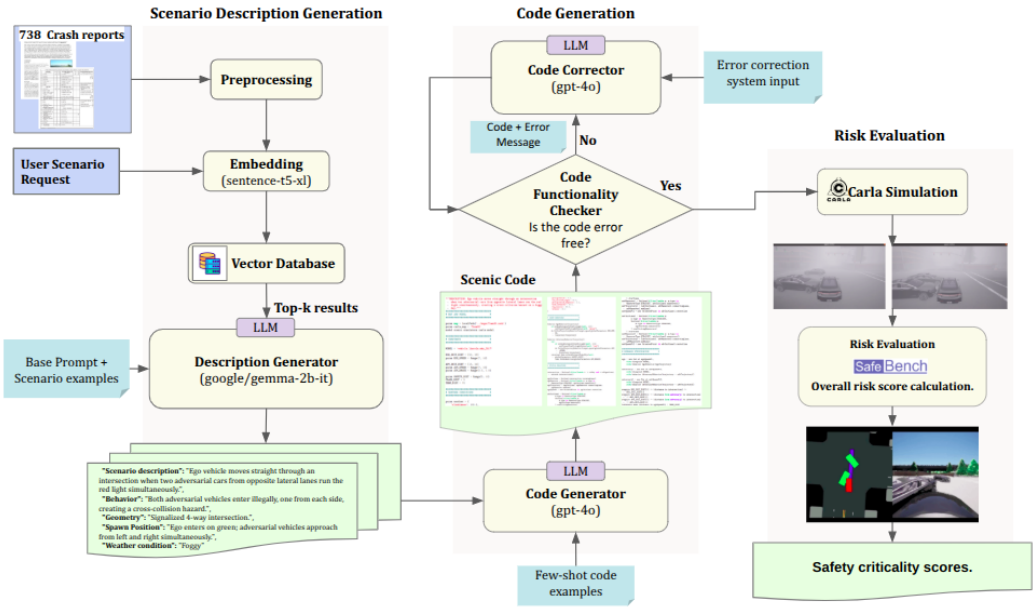

Figure 2: Our scenario generation pipeline.

3.1 Data collection

To generate realistic, safety-critical driving scenarios that capture long-tail events, we leverage real-world crash data rather than synthetic descriptions. Specifically, we retrieve open-access crash reports from the National Highway Traffic Safety Administration (NHTSA) [23] and the California Department of Motor Vehicles (DMV) [24].

The NHTSA website comprises 64 detailed crash reports, each averaging around 50 pages, and containing tables, images, graphs, and detailed textual descriptions. These reports provide comprehensive information about the crash site, pre-crash conditions, the crash event, and post-crash details. They also include information on damages to the vehicle and driver, fatality status, occupant data, and injuries, as well as sensor readings taken before, during, and after the crash.

The 674 reports available on the California DMV website include incidents of autonomous vehicle crashes from 2019 to 2025, involving various manufacturers such as Waymo [25], Zoox [26], GM Cruise [27], WeRide Corp [28], and others. The reports are structured, including tables alongside textual descriptions. They provide information about the manufacturer, the vehicles involved in the accident, the time and location of the crash, information about the occupants, the extent of damage, a detailed description of the accident, the type of collision, roadway conditions, lighting, weather conditions, and more.

3.2 Retrieval augmented generation

The collected data contains extensive information, but we need only the relevant parts for scenario generation. To make the crash reports more systematic and to get the most relevant parts of the document, RAG [29] is used. This approach includes the following steps: data preprocessing, embedding, similarity check, and generation.

3.2.1 Data preprocessing

The NHTSA [23] and DMV [24] crash reports differ significantly in structure, requiring separate preprocessing pipelines. NHTSA reports are lengthy, text-heavy PDFs with embedded tables and figures, whereas DMV reports are concise (4 pages) and primarily tabular, featuring checkboxes.

For NHTSA data, we remove irrelevant sections and extract key textual components describing pre-crash, crash, and post-crash conditions, including weather, lighting, traffic levels, vehicle speeds, and injury details.

DMV reports are parsed using optical character recognition (OCR) based extraction to capture both text fields and checked boxes, addressing the limitations of standard PDF parsers. The processed outputs are merged into unified text representations containing crash descriptions, environmental factors, vehicle information, and collision types. Preprocessed data is used as input for the next steps in retrieval and generation.

3.2.2 Embedding and similarity search

Preprocessed crash data is separated into smaller chunks and is embedded in a vector format. Each scenario has its own query as a user input for similarity checking. In the query, we ask about how the crash happened and request the pre-crash scenarios and conditions for the selected scenario category. For the similarity search between the given query and the selected sections from the document, the sentence-t5-xl sentence transformers model [30] is utilized. It maps sentences and paragraphs to a 768-dimensional dense vector space.

Both the query and the text are transformed into vector-based representations, and the cosine similarity metric is calculated. After conducting this similarity check, the top five most similar results are selected.

3.2.3 Description Generation

Using the top five results and prompts specialized for the given scenario categories, scenario outputs are obtained from the "google/gemma-2b-it" large language model [31]. This model is preferred because it offers a suitable and lightweight option for text generation, reasoning, and summarization tasks on a local machine with 8 GB of VRAM.

In each scenario generation part, two prompts are used that are both specified for the given scenario category. The first one is the base prompt, which describes the desired output structure and provides category-specific examples. An example is shown in the Appendix B.1. The second one is a context prompt derived from the top-k retrieval results to enhance relevance and diversity.

Each category is processed independently, and the resulting outputs are stored separately. The model generates structured scenario descriptions in JSON format containing agents with adversarial behaviors, which are later used for Scenic code generation. A scenario output of the RAG process is shown in Appendix A.2. A complete list of generated scenario descriptions is available in Appendix C.

3.3 Code generation

For code generation, the output of the RAG consists of structured scenarios that serve as user input for the LLM. The system input features two different prompts for each scenario category. The base prompt includes general instructions that outline the basic rules for the scene. The details of this prompt can be seen in the Appendix B.2. The second prompt is tailored for each scenario category and includes several few-shot examples. A representative example of a few-shot system prompt can be found in Appendix B.3.

Each few-shot prompt consists of five unique Scenic scripts, which include adversarial agents, such as pedestrians, cars, trucks, and stationary objects, including trash and debris. Given these structured descriptions and prompts, Scenic code is produced using GPT-4o [32].

3.4 Code functionality check and code correction

To mitigate the risk of producing non-functional code, we have implemented a code functionality checker and a code corrector. The functionality checker monitors the code during execution. If the code runs for more than 30 seconds, the system assumes the code is functional. If an error occurs, the corrector attempts to fix it using GPT-4o along with the error correction system input and relevant error information.

This corrector requires two prompts: the first prompt provides details about the code error information, following the format shown in Appendix B.4. The second prompt outlines the task and valid scenic operations, which include actions, attributes, parameter formats, agent definitions, and scenario definitions. The detailed input on the error correction system is given in Appendix B.5.

3.5 Risk evaluation

As a risk evaluation framework, we utilize SafeBench [6] to compare the safety criticality of our generated scenarios with those of other baseline models. Our method primarily generates dynamic scenarios. After ensuring the generated dynamic Scenic scenarios are error-free, we transform these dynamic codes into a static version suitable for fair comparison. The key difference between static and dynamic scenarios lies in their flexibility. In a static scenario, the routes, velocities, waypoints, and spawn points for the ego vehicle are predetermined. This means that we can only change the spawn points, velocities, and behaviors of surrounding agents and objects. In contrast, a dynamic scenario allows us to create everything in the environment, including the ego vehicle's position, behaviors, velocities, and more. The modified static code maintains the original logic, and events occur around the ego; it specifically updates components related to the ego agent, such as the ego spawn point, trajectory, velocity, and behavior. During the execution of the static scenarios, the movements of the ego agent are controlled by three different pre-trained reinforcement learning agents provided by Safebench. These agents are Soft Actor-Critic (SAC) [33], Proximal Policy Optimization (PPO) [34], and Twin Delayed DDPG(TD3) [35].

After running scenarios with these reinforcement learning agents, the safety criticality of the scenarios is evaluated according to Safebench's metrics. A list of the metrics and their explanations is provided in the appendix E.

4 Experiments and Evaluation

We compare our method with the following baselines: Chatscene [17], Learning to Collide [14], AdvSim [15], Carla Scenario Generator [18], and Adversarial Trajectory Optimization [16].

We conduct experiments to assess the criticality and complexity of the generated scenarios. Each of the eight scenario categories contains five scenarios, which are trained on ten routes for three different RL-based test ego agents. All experiments were conducted on an NVIDIA GeForce RTX 3070 GPU.

Scenic is a probabilistic programming language, so it allows us to execute the same scenario multiple times with varying parameters and conditions, depending on how the code is structured and how the distributions are defined. This property of Scenic allows us to enhance the variety of the scenarios we generate. Specifically, we tried to create a total of $8 \times 5 \times 50 \times 10 \times 3 = 60000$ distinct scenarios. Here, 8 represents the number of scenario categories, 5 is the number of selected scenarios, 50 indicates that each selected scenario is executed 50 times, 10 is the total number of routes, and 3 is the number of different RL-based ego agents. The scenarios are distinct because all parameters are defined within each scenario, including velocities, acceleration, and deceleration values, as well as the positions of the spawn points, which are selected randomly from a given range and distribution.

In our experiments, we compare collision rates (CR) and overall scores (OS). The overall score is a combination of ten scores provided by Safebench. For collision rate results, a higher score indicates better performance, while for the overall score, a lower score is preferable. These two metrics help assess the criticality of the generated scenarios. A high collision rate combined with a low overall score indicates a higher level of safety criticality [6, 17].

The first experiment is done using the pre-trained SAC agent. The comparison between the baselines shows the collision rates for each scenario category separately and their average. For the turning obstacle and vehicle passing scenario categories, our method gives the best results among the

Table 1: Collision rate scores with pretrained SAC ego agent

Model	Algorithm	SO	TO	LC	VP	RLR	ULT	RT	CN	Average
↑ SAC	LC	0.40	0.11	0.60	0.80	0.92	0.86	0.70	0.75	0.642
	AS	0.53	0.40	0.75	0.90	0.62	0.85	0.21	0.53	0.599
	CS	0.53	0.68	0.67	0.90	0.76	0.90	0.69	0.68	0.725
	AT	0.73	0.48	0.77	0.90	**1.00**	**0.97**	0.76	**0.90**	0.814
	Chatscene	**0.94**	0.73	**0.92**	0.81	0.70	0.88	**0.84**	0.78	0.825
	Ours	0.93	**0.84**	0.80	**0.96**	0.90	0.72	0.76	0.71	**0.827**

baselines, as shown in Table 1. One reason for this is that in the baseline scenarios, there is one adversarial agent in the scene, but in our scenarios, our scenario generator is able to generate multiple adversarial agents at the same time, such as a car, a pedestrian, or a cyclist. For the red light running case, our method results in better performance than Chatscene, but it is not the best result among the other baselines. AT and LC result in better performance than our approach. On average, the scenarios we generated outperformed other baselines by achieving the highest collision rate.

Table 2: Collision rate scores with pretrained PPO ego agent

Model	Algorithm	SO	TO	LC	VP	RLR	ULT	RT	CN	Average
↑ PPO	LC	0.09	0.11	**1.00**	**1.00**	0.22	0.20	0.28	0.00	0.363
	AS	0.39	0.19	**1.00**	**1.00**	0.67	0.61	0.62	**0.92**	0.675
	CS	0.22	0.61	**1.00**	**1.00**	0.47	0.37	0.52	0.44	0.579
	AT	0.10	0.11	0.98	0.87	0.13	0.21	0.03	0.05	0.310
	ChatScene	**0.87**	0.61	0.97	0.98	**0.89**	0.52	0.74	0.91	**0.811**
	Ours	0.73	**0.85**	0.63	0.87	0.56	**0.62**	**0.78**	0.51	0.693

The second experiment shown in Table 2 was conducted with the PPO-trained ego agent to compare collision rates. In both the turning obstacle, unprotected left turn, and right turn scenarios, our method achieved the highest collision rates of 0.85, 0.62, and 0.78, respectively. For the lane-changing case, our scenarios are the worst-performing ones compared to the other baseline methods. One reason for this is that the spawn points of the adversarial agents may not align well with the movements of the pretrained ego agents, especially in scenarios on the highway. The second reason for these results is that at the beginning of the scenario, the PPO agent drives very slowly, even though there are no agents nearby. This causes the adversarial agents to begin their moves without encountering the ego vehicle.

Table 3: Collision rate scores with pretrained TD3 ego agent

Model	Algorithm	SO	TO	LC	VP	RLR	ULT	RT	CN	Average
↑ TD3	LC	0.42	0.06	**1.00**	0.70	**1.00**	**1.00**	0.79	**1.00**	0.746
	AS	0.60	0.39	0.83	0.70	0.41	0.65	0.03	0.26	0.484
	CS	0.61	0.53	**1.00**	0.70	0.67	0.80	0.83	0.67	0.726
	AT	0.67	0.35	0.59	0.70	**1.00**	0.85	**0.99**	0.90	0.756
	ChatScene	**0.87**	0.75	0.96	**0.99**	0.77	0.86	0.75	0.90	**0.856**
	Ours	0.63	**0.83**	0.72	0.85	0.63	0.58	0.73	0.45	0.677

The third experiment shown in Table 3 was conducted with the TD3-trained ego agent to compare collision rates. Our method achieved the highest score of 0.83 in the turning obstacle scenario category, but performed relatively poorly for other scenario categories.

In the experiment with the SAC agent, overall scores were compared with the baseline scores. For the overall score, the lower results are better. As shown in Table 4, our method achieved the lowest overall scores among all baselines in three scenario categories: 0.50 for the turning obstacle, 0.45 for the vehicle passing, and 0.44 for the red light running. The average overall score is not the best, but it is very close to ChatScene's result. Although the CR score is not the highest at 0.90 (Table 1), the OS achieved the best score among all baselines in the red light running scenario with a pretrained

Table 4: Overall scores with pretrained SAC ego agent

Model	Algorithm	SO	TO	LC	VP	RLR	ULT	RT	CN	Average
↓ SAC	LC	0.72	0.82	0.62	0.52	0.49	0.52	0.50	0.50	0.59
	AS	0.66	0.67	0.55	0.47	0.65	0.54	0.75	0.62	0.61
	CS	0.66	0.53	0.57	0.47	0.58	0.51	0.50	0.54	0.54
	AT	0.57	0.63	0.55	0.47	0.46	**0.48**	0.46	**0.42**	0.50
	ChatScene	**0.45**	0.51	**0.45**	0.49	0.58	0.49	**0.43**	0.46	**0.48**
	Ours	0.46	**0.50**	0.53	**0.45**	**0.44**	0.57	0.52	0.52	0.50

SAC ego agent. One reason for this is that the other metrics performed better in this case, which influenced our overall score. This demonstrates that even though the generated scenarios for the red light running scenario category do not achieve the highest CR score, the low OS score indicates that driving comfort and safety are still at risk, making the scenario safety-critical.

Table 5: Overall scores with pretrained PPO ego agent

Model	Algorithm	SO	TO	LC	VP	RLR	ULT	RT	CN	Average
↓ PPO	LC	0.86	0.82	0.46	0.45	0.85	0.86	0.70	0.89	0.74
	AS	0.73	0.78	0.46	0.44	0.62	0.65	0.54	**0.41**	0.58
	CS	0.81	0.57	0.47	0.44	0.72	0.77	0.58	0.66	0.63
	AT	0.85	0.82	**0.44**	0.49	0.90	0.85	0.83	0.86	0.75
	ChatScene	**0.50**	0.58	**0.44**	**0.42**	**0.50**	0.71	**0.50**	0.42	**0.51**
	Ours	0.56	**0.49**	0.61	0.48	0.61	**0.62**	0.51	0.62	0.56

In Table 5, we present the overall score results of the pretrained PPO agent. Notably, for the turning obstacle and unprotected left turn cases, our method outperformed all baseline models. Our results for the other scenario categories are also promising; however, they did not achieve better results than the baselines. In the right turn case, our generated scenarios received the highest CR score, although the OS score did not rank the highest. This discrepancy is due to the influence of other metrics on the overall result. In overall performance, our method is the second-best scored among six methods for the PPO pre-trained agent.

Table 6: Overall scores with pretrained TD3 ego agent

Model	Algorithm	SO	TO	LC	VP	RLR	ULT	RT	CN	Average
↓ TD3	LC	0.71	0.84	0.44	0.56	**0.46**	**0.47**	0.45	**0.38**	0.54
	AS	0.63	0.67	0.51	0.56	0.76	0.64	0.83	0.76	0.67
	CS	0.63	0.60	0.43	0.56	0.62	0.56	0.43	0.54	0.55
	AT	0.59	0.69	0.63	0.56	0.46	0.53	**0.35**	0.42	0.53
	ChatScene	**0.46**	**0.48**	**0.41**	**0.41**	0.53	0.48	0.49	0.39	**0.46**
	Ours	0.59	0.50	0.57	0.50	0.57	0.64	0.54	0.65	0.57

Table 6 compares our results with the baseline when the ego agent is pretrained using the TD3 reinforcement learning method. The scenarios we generated did not perform well, possibly because they may not have aligned with the behaviors of the TD3-trained ego agents. Our scenario prompts failed to adequately capture the ego agents' behaviors, resulting in a lack of necessary adversariality in the generated scenarios.

We can conclude that our generated scenarios exhibit the highest collision rate and the lowest overall score compared to the baselines for the turning obstacle scenario category, across all test cases. This indicates that our generated scenarios for turning obstacles are safety-critical. Additionally, Table 7 presents a comprehensive comparison of the overall scores for the average performance of three pretrained RL ego agents across eight scenario categories, in relation to baseline methods. This comparison shows that our generated scenarios challenge the RL agents, resulting in the lowest route completion score.

Table 7: Detailed results of the averaged overall score performance of three distinct pre-trained ego vehicles across eight base scenarios. The metrics used to evaluate the overall scores include the following: collision rate (CR), red-light running (RR), stop sign running (SS), out of road driving (OR), route following stability (RF), route completion (Comp), completion time (TS), average acceleration (ACC), yaw velocity (YV), lane invasion frequency (LI), and overall score (OS). These metrics provide a comprehensive assessment of the vehicles' performance in various driving scenarios.

Alg.	Safety Level				Functionality Level			Etiquette Level			OS↓
	CR↑	RR↑	SS↑	OR↑	RF↓	Comp↓	TS↑	ACC↑	YV↑	LI↑	
LC	0.58	**0.33**	**0.16**	0.03	0.89	0.73	0.22	0.21	0.24	0.11	0.62
AS	0.59	0.30	**0.16**	0.03	0.89	0.75	0.26	0.20	0.25	0.13	0.62
CS	0.68	0.31	**0.16**	**0.04**	0.89	0.74	0.24	0.22	0.24	0.13	0.57
AT	0.63	0.31	**0.16**	0.03	0.89	0.73	**0.28**	0.22	0.25	0.14	0.60
ChatScene	**0.83**	0.18	0.14	**0.04**	**0.83**	0.54	0.22	**0.71**	**0.53**	**0.24**	**0.48**
Ours	0.73	0.29	0.12	0.02	0.89	**0.52**	0.17	0.33	0.22	0.06	0.54

5 Discussion and Future Work

In this study, an end-to-end pipeline for generating safety-critical scenarios for autonomous driving was developed while assessing the safety-criticality of the generated scenarios. This pipeline utilized 738 real-world crash reports and implemented the RAG framework, along with a few-shot prompting technique to enhance the variety, complexity, and realism of the generated scenarios. LLMs were utilized for code generation and error correction, using few-shot prompts derived from RAG output and prompts specifically designed for error correction. Generated codes are simulated using the CARLA simulation environment. These codes include multiple agents in a scenario, also with different weather, lighting, and road surface conditions.

Although we can gather valuable information from our crash reports, we encountered several issues during the evaluation of our generated scenarios that hindered our ability to use all the retrieved information. We needed to convert our dynamic code into a static version for comparison, which limited our ability to control the spawn points and variables associated with the ego agents. As a result, some static scenarios produced runtime errors because the generated adversarial agents did not fit the predefined map, even though they could be spawned in the dynamic version of the same scenario. We were also unable to incorporate weather conditions into the Safebench evaluation method [6] for static scenarios. As a result, we could only observe the effects of weather conditions on the road surface and lighting in our dynamic cases.

Our method demonstrated promising results when compared to baseline approaches. Specifically, when we evaluated our collision rate results against ChatScene, our generated scenarios outperformed ChatScene when it was trained with the SAC pretrained ego agent. We achieved an 11% improvement in navigating turning obstacles, a 15% improvement in vehicle passing, and a 20% improvement in red-light running categories, as well as an increase in the average score across all categories. For the scenarios that were trained on the PPO agent, we outperformed Chatscene for collision rates with a 24% improvement in the turning obstacle category, a 10% improvement in the unprotected left turn, and a 4% improvement in the right turn category. For the collision rate results of the TD3 ego agent, turning obstacle scenarios again performed the best results with at least 8% improvement among all baseline methods.

When comparing the performance of our generated scenarios for SAC, PPO, and TD3 RL agents, we found that our scenarios performed the best on the SAC pretrained agent and the worst on the TD3 agent. The main reason for this is that the agent's behaviors differ significantly from each other, and these ego behaviors include adversarial behavior, as slow driving on a highway, immediate acceleration after this, without seeing any cars or dangerous situations. In particular, in crossing negotiation scenarios, the criticality of the situations depends on the ego agent's initial speed and starting position. If the ego agent is spawned far from the intersection and drives very slowly, the adversarial agents will likely pass through the intersection before the ego agent arrives. To address this issue, we added waiting distances to our few-shot scenic code prompts. However, generating code with an exact waiting distance that leads to a collision with the non-standard behaving ego agent is unlikely.

Although the current results are promising, there is still considerable room for improvement to address suboptimal performances and further enhance the set of generated scenarios. We have observed that incorporating additional crash reports significantly enhances the performance of scenario generation. A multimodal approach that utilizes various relevant data sources can further improve the effectiveness of the scenarios produced. This study was conducted on CARLA maps, and for comparison, we utilized restricted maps. We can provide more specific information about the roads and include additional maps in the scene, as well as retrieve detailed road structures at the crash site. This enables the generation of granular scenarios and scenic codes with a specific road selection capability. Scenic enables us to record different camera views and also records lidar data. The capability of Scenic allows us to capture sensor data while scenarios are being executed. This significantly enhances the variety of generated code outputs, including Scenic scenario simulations with camera views from preferred perspectives and lidar readings. As a result, it aids in the creation of various types of synthetic data for safety- critical scenarios. Lastly, adding new metrics that can better capture scenario complexity and safety criticality, such as the agent involvement metric and the crash severity metric, can also further improve the risk evaluation.

References

[1] SAE International, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," Apr. 30 2021.

[2] H. Cao, G. Chen, J. Xia, G. Zhuang, and A. Knoll, "Fusion-based feature attention gate component for vehicle detection based on event camera," *IEEE Sensors Journal*, vol. 21, no. 21, pp. 24540–24548, 2021.

[3] P. Liu, R. Yang, and Z. Xu, "How safe is safe enough for self-driving vehicles?," *Risk Analysis*, vol. 39, pp. 315–325, 05 2018.

[4] H. Cao, Z. Zhang, Y. Xia, X. Li, J. Xia, G. Chen, and A. Knoll, "Embracing events and frames with hierarchical feature refinement network for object detection," in *European Conference on Computer Vision*, pp. 161–177, Springer, 2024.

[5] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: a language for scenario specification and scene generation," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '19, p. 63–78, ACM, June 2019.

[6] C. Xu, W. Ding, W. Lyu, Z. Liu, S. Wang, Y. He, H. Hu, D. Zhao, and B. Li, "Safebench: A benchmarking platform for safety evaluation of autonomous vehicles," 2022.

[7] H. Cao, G. Chen, Z. Li, Y. Hu, and A. Knoll, "Neurograsp: Multimodal neural network with euler region regression for neuromorphic vision-based grasp pose estimation," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–11, 2022.

[8] H. Cao, Z. Qu, G. Chen, X. Li, L. Thiele, and A. Knoll, "Ghostvit: Expediting vision transformers via cheap operations," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 6, pp. 2517–2525, 2024.

[9] H. Cao, G. Chen, H. Zhao, D. Jiang, X. Zhang, Q. Tian, and A. Knoll, "Sdpt: Semantic-aware dimension-pooling transformer for image segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 11, pp. 15934–15946, 2024.

[10] S. Li, T. Azfar, and R. Ke, "Chatsumo: Large language model for automating traffic scenario generation in simulation of urban mobility," 2024.

[11] B.-K. Ruan, H.-T. Tsui, Y.-H. Li, and H.-H. Shuai, "Traffic scene generation from natural language description for autonomous vehicles with large language model," 2025.

[12] Q. Lu, M. Ma, X. Dai, X. Wang, and S. Feng, "Realistic corner case generation for autonomous vehicles with multimodal large language model," *ArXiv*, vol. abs/2412.00243, 2024.

[13] K. Elmaaroufi, D. Shanker, A. Cismaru, M. Vazquez-Chanlatte, A. Sangiovanni-Vincentelli, M. Zaharia, and S. A. Seshia, "ScenicNL: Generating probabilistic scenario programs from natural language," in *First Conference on Language Modeling*, 2024.

[14] W. Ding, B. Chen, M. Xu, and D. Zhao, "Learning to collide: An adaptive safety-critical scenarios generating method," 2020.

[15] J. Wang, A. Pun, J. Tu, S. Manivasagam, A. Sadat, S. Casas, M. Ren, and R. Urtasun, "Advsim: Generating safety-critical scenarios for self-driving vehicles," 2023.

[16] Q. Zhang, S. Hu, J. Sun, Q. A. Chen, and Z. M. Mao, "On adversarial robustness of trajectory prediction for autonomous vehicles," 2022.

[17] J. Zhang, C. Xu, and B. Li, "Chatscene: Knowledge-enabled safety-critical scenario generation for autonomous vehicles," 2024.

[18] Scenario Runner Contributors, "Carla scenario runner," 2019.

[19] M. Althoff, M. Koschi, and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads," 06 2017.

[20] P. Cai, Y. Lee, Y. Luo, and D. Hsu, "Summit: A simulator for urban driving in massive mixed traffic," 2020.

[21] "Deepdrive simulation," 2018.

[22] Y. Gao, M. Piccinini, K. Moller, A. Alanwar, and J. Betz, "From words to collisions: Llm-guided evaluation and adversarial generation of safety-critical driving scenarios," 2025.

[23] National Center for Statistics and Analysis (NHTSA), "Crashstats: Nhtsa motor vehicle traffic crash data," 2025.

[24] California Department of Motor Vehicles, "Autonomous vehicle collision reports," 2025.

[25] Waymo LLC, "Waymo." `https://waymo.com`, 2025. Accessed: 2025-08-21.

[26] Zoox Inc., "Zoox." `https://zoox.com`, 2025. Accessed: 2025-08-21.

[27] Cruise LLC, "Cruise." `https://www.getcruise.com`, 2025. Accessed: 2025-08-21.

[28] WeRide Corp., "Weride." `https://www.weride.ai`, 2025. Accessed: 2025-08-21.

[29] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2021.

[30] J. Ni, G. H. Ábrego, N. Constant, J. Ma, K. B. Hall, D. Cer, and Y. Yang, "Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models," 2021.

[31] T. M. Gemma Team, C. Hardin, R. Dadashi, S. Bhupatiraju, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Hussenot, and et al., "Gemma," 2024.

[32] OpenAI, "Gpt-4 technical report," 2024.

[33] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018.

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[35] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018.

A Data Integration and Formatting

A.1 Preprocessed crash report example from DMV crash case

```
On December 4, 2021 at 4:10 PM PST a Waymo Autonomous Vehicle (‘“‘Waymo
↪  AV’’) operating in San Francisco, CA was in a collision involving an
↪  SUV on Winson Drive between 19th and 20th Avenues.  While in
↪  autonomous mode, the Waymo AV turned right from southbound 19th
↪  Avenue onto westbound Winston Drive. As the Waymo AV proceeded onto
↪  Winston Drive in the far right lane, an SUV also proceeding westbound
↪  on Winston Drive made a continuous right lane change from the far
↪  left lane, through the middle lane, and entered the far right lane in
↪  front of the Waymo AV. The Waymo AV was decelerating when the test
↪  driver transitioned to manual mode. The Waymo AV then made contact
↪  with the SUV. At the time of the impact, the Waymo AV’s Level 4 ADS
↪  was not engaged and a test driver (in the driver’s seating position)
↪  was operating the Waymo AV in manual mode. The Waymo AV sustained
↪  damage to its front driver side fender and the SUV sustained damage
↪  to its front passenger side fender. The weather is clear. The
↪  lighting condition is daylight. The road is dry. A crash happened
↪  while vehicle 1 was proceeding straight, and vehicle 2 was changing
↪  lanes. The type of collision is broadside.
```

```
"scenario 17": {

"Scenario description": "Ego vehicle moves straight through an intersection when two adversarial
cars from opposite lateral lanes run the red light simultaneously.",

"Behavior": "Both adversarial vehicles enter illegally, one from each side, creating a
cross-collision hazard.",

"Geometry": "Signalized 4-way intersection.",

"Spawn Position": "Ego enters on green; adversarial vehicles approach from left and right
simultaneously.",

"Weather condition": "Foggy"}
```


Figure 3: RAG scenario output for a red-light running scenario tag.

A.2 RAG output

B Prompts for Code Generation

B.1 Base prompt for the scenario description generation

```
Create a safety-critical crash scenario for autonomous driving.
Your task is to decompose full descriptions of safety-critical scenarios
↪  into sub-descriptions for the following distinct components:
Scenario description: Give a short description of the crash scenario.
↪  (Explain how the crash happened.) It must include one ego vehicle,
↪  one or more adversarial agents. The ego must follow the traffic
↪  rules, but other agents must violate the rules.
Behavior: Describe the behavior of the adversarial agent (you should also
↪  indicate the type of object, like pedestrians, cars).
Geometry: Specify the road condition where the scenario occurs (e.g.,
↪  straight road, three-way intersection, etc.).
Spawn Position: Indicate the initial relative position of the adversarial
↪  agent to the ego vehicle.
Weather condition: Indicate the weather condition of the site.

Here is the expected format:
Scenario [Number here]:
Scenario description: [your sentence here]
Behavior: [your sentence here]
Geometry: [your sentence here]
Spawn Position: [your sentence here]
Weather condition: [your sentence here]

Here is an example:
<example>

Ego behavior must be fixed and the same in the example above. Change the
↪  behavior of the adversarial agent and write it in the **behavior:**
↪  part.
Each component (Scenario description, behavior, geometry, spawn position,
↪  weather condition) should include only one sentence for each
↪  scenario. Each section must be a single line with the expected
↪  format: **section_name:** value
Give 5 different scenarios, they must have the same format as the given
↪  example that includes all components outlined above.
```

B.2 Instruction Prompt for Scenic Code Generation

```
You are an expert programming assistant in the Scenic domain-specific
↪  language. The user will give you safety-critical scenarios that
↪  include scenario description, agent behavior, road geometry, and
↪  weather conditions. Now, based on the given scenarios, your task is
↪  to provide Scenic code snippets for the Carla Simulation environment
that simulate adversarial behaviors in traffic scenarios given by the
↪  user. To do so, use the given rules:
1 - Use the given examples.
2 - Do not forget to add scenario descriptions.
3 - Just return the code in a valid format.
4 - Ego and adversary vehicles intersect in space and time.
5 - Random parameters are bounded to encourage collision.
6 - To get adversary behavior for adversary vehicle, you can make the
↪  adversary vehicle stop or slow down **suddenly** in front of the ego
↪  or accelerate **aggressively** when the ego is near.
```

B.3 Few-Shot System Prompt for the Red-Light Running Scenario Category

````
The user will give you safety-critical scenarios that include scenario
↪  description, agent behavior, road geometry, and weather conditions.
Now, based on the given scenarios, your task is to provide Scenic code
↪  snippets for the Carla Simulation environment
that simulate adversarial behaviors in traffic scenarios. To do so, use
↪  the given examples, do not change the valid format.
Do not forget to add scenario descriptions.
Here are five code examples:

Scenario example 1:

```scenic
"""DESCRIPTION: Ego vehicle goes straight at 4-way intersection and must
↪  suddenly stop to avoid collision, when adversary vehicle from
↪  opposite lane makes a left turn aggressively runs the red-light in a
↪  rainy day."""
################################
# MAP AND MODEL                #
################################

param map = localPath('../maps/Town05.xodr')
param carla_map = 'Town05'
model scenic.simulators.carla.model

################################
# CONSTANTS                    #
################################

MODEL = 'vehicle.lincoln.mkz_2017'

EGO_INIT_DIST = [20, 25]
param EGO_SPEED = Range(5, 15)
param EGO_BRAKE = Range(0.3, 0.6)

ADV_INIT_DIST = [15, 20]
param ADV_SPEED = Range(5, 15)
param ADV_THROTTLE = Range(0.2, 0.7)
````
````

```
param SAFETY_DIST = Range(10, 15)
CRASH_DIST = 5
TERM_DIST = 70

################################
# WEATHER CONDITIONS           #
################################

param weather = {
    'cloudiness': 0.0,
    'precipitation': 80.0,
    'sun_altitude_angle': 40.0,
    'precipitation_deposits': 30.0,
    'wind_intensity': 0.0,
    'fog_density': 0.0,
    'wetness': 40.0
}

################################
# AGENT BEHAVIORS              #
################################

behavior EgoBehavior(trajectory):
    try:
        if withinDistanceToTrafficLight(self, 70):
            setClosestTrafficLightStatus(self, "green")
        do FollowTrajectoryBehavior(target_speed=globalParameters.EGO_SP
        ↪  EED,
        ↪  trajectory=trajectory)
    interrupt when withinDistanceToAnyObjs(self,
    ↪  globalParameters.SAFETY_DIST):
        take SetBrakeAction(globalParameters.EGO_BRAKE)
    interrupt when withinDistanceToAnyObjs(self, CRASH_DIST):
        terminate

behavior AdversaryBehavior(trajectory):
    try:
        if withinDistanceToTrafficLight(self, 100):
            setClosestTrafficLightStatus(self, "red")
        do FollowTrajectoryBehavior(target_speed=globalParameters.ADV_SP
        ↪  EED,
        ↪  trajectory=trajectory)
    interrupt when (distance to self) <= globalParameters.SAFETY_DIST:
        take SetThrottleAction(globalParameters.ADV_THROTTLE)
    interrupt when withinDistanceToAnyObjs(self, CRASH_DIST):
        terminate

################################
# SPATIAL RELATIONS            #
################################

intersection = Uniform(*filter(lambda i: i.is4Way and i.isSignalized,
↪  network.intersections))

egoInitLane = Uniform(*intersection.incomingLanes)
egoManeuver = Uniform(*filter(lambda m: m.type is ManeuverType.STRAIGHT,
↪  egoInitLane.maneuvers))
```

```
egoTrajectory = [egoInitLane, egoManeuver.connectingLane,
↪  egoManeuver.endLane]
egoSpawnPt = new OrientedPoint in egoInitLane.centerline

advInitLane = Uniform(*filter(lambda m:
        m.type is ManeuverType.STRAIGHT,
        egoManeuver.reverseManeuvers)
    ).startLane
advManeuver = Uniform(*filter(lambda m: m.type is ManeuverType.LEFT_TURN,
↪  advInitLane.maneuvers))
advTrajectory = [advInitLane, advManeuver.connectingLane,
↪  advManeuver.endLane]
advSpawnPt = new OrientedPoint in advInitLane.centerline

################################
# SCENARIO SPECIFICATION       #
################################

ego = new Car at egoSpawnPt,
    with blueprint MODEL,
    with behavior EgoBehavior(egoTrajectory)

adversary = new Car at advSpawnPt,
    with blueprint MODEL,
    with behavior AdversaryBehavior(trajectory = advTrajectory)

require EGO_INIT_DIST[0] <= (distance to intersection) <=
↪  EGO_INIT_DIST[1]
require ADV_INIT_DIST[0] <= (distance from adversary to intersection) <=
↪  ADV_INIT_DIST[1]
terminate when (distance to egoSpawnPt) > TERM_DIST
```

B.4 Error information prompt format

```
The following code is producing an error. Please correct the code based
↪  on the error message. Return only the corrected code.
Here is the code:
```scenic
<failing code>
```

Here is the error message:
<error message>
```

B.5 Error Correction System Input Prompt

```
You are an expert programming assistant in the Scenic domain-specific
↪  language. When a user provides buggy code and an error message, your
↪  task is to analyze the code and the error, and return the corrected
↪  code that fixes the problem. Do not explain your changes. Just return
↪  the corrected code in a valid format.

Valid Scenic base behaviors:
- FollowLaneBehavior
    FollowLaneBehavior(target_speed=10, laneToFollow=None,
    ↪  is_oppositeTraffic=False)
```

```
- FollowTrajectoryBehavior
    FollowTrajectoryBehavior(target_speed=10, trajectory=None,
    ↪ turn_speed=None)
- LaneChange
    LaneChangeBehavior(laneSectionToSwitch, is_oppositeTraffic=False,
    ↪ target_speed=10)
- TurnBehavior
    TurnBehavior(trajectory, target_speed=6)
- WalkForwardBehavior -> only for pedestrians
    WalkForwardBehavior()
- ConstantThrottleBehavior
- DriveAvoidingCollisions
- AccelerateForwardBehavior

Valid Scenic Built-in Distributions (syntax and meaning):
- Range(low, high) # uniformly-distributed real number in the interval
- DiscreteRange(low, high) # uniformly-distributed integer in the (fixed)
↪ interval
- Normal(mean, stdDev) # normal distribution with the given mean and
↪ standard deviation
- TruncatedNormal(mean, stdDev, low, high) # normal distribution
↪ truncated to the given window
- Uniform(value, ...) # uniform over a finite set of values
- Discrete({value: weight, ...}) # discrete with given values and weights
- new Point in region # uniformly-distributed Point in a region

Valid Scenic Compound Statements (syntax and meaning):
- class name[(superclass)]: # Defines a Scenic class.
- behavior name(arguments): # Defines a dynamic behavior.
- monitor name(arguments): # Defines a monitor.
- scenario name(arguments): # Defines a modular scenario.
- try: ... interrupt when boolean: # Run code with interrupts inside a
↪ dynamic behavior or modular scenario.

Valid Scenic Simple Statements (syntax and meaning):
- model name # Select the world model.
- import module # Import a Scenic or Python module.
- param name = value, ... # Define global parameters of the scenario.
- require boolean # Define a hard requirement.
- require[number] boolean # Define a soft requirement.
- require monitor monitor # Define a dynamic requirement using a monitor.
- terminate when boolean # Define a termination condition.
- terminate after scalar (seconds | steps) # Set the scenario to
↪ terminate after a given amount of time.
- mutate identifier, ... [by number] # Enable mutation of the given list
↪ of objects.
- record [initial | final] value as name # Save a value at every time
↪ step or only at the start/end of the simulation.

Valid Scenic Dynamic Statements (syntax and meaning):
Note: These statements can only be used inside a dynamic behavior,
↪ monitor, or compose block of a modular scenario.
- take action, ... # Take the action(s) specified.
- wait # Take no actions this time step.
- terminate # Immediately end the scenario.
- terminate simulation # Immediately end the entire simulation.
- do behavior/scenario, ... # Run one or more sub-behaviors/sub-scenarios
↪ until they complete.
```

```
- do behavior/scenario, ... until boolean # Run sub-behaviors/scenarios until they complete or a condition is met.
- do behavior/scenario, ... for scalar (seconds | steps) # Run sub-behaviors/scenarios for (at most) a specified period of time.
- do choose behavior/scenario, ... # Run one choice of sub-behavior/scenario whose preconditions are satisfied.
- do shuffle behavior/scenario, ... # Run several sub-behaviors/scenarios in a random order, satisfying preconditions.
- abort # Break out of the current try-interrupt statement.
- override object specifier, ... # Override properties of an object for the duration of the current scenario.

Valid Scenic Actions:(Actions for dynamic agents in the driving domain.)
- OffsetAction(offset) # Teleports actor forward (in direction of its heading) by some offset. offset (Vector)
- SetSpeedAction(speed) # Set the speed of an agent (keeping its heading fixed). speed (float)
- SetVelocityAction(xVel, yVel, zVel=0) # Set the velocity of an agent. parameters are float type.
- SetPositionAction(pos) # Teleport an agent to the given position. pos (Vector)
- SteeringAction # Abstract class for actions usable by agents that can steer. This is a base class for the following actions;
    - RegulatedControlAction(throttle, steer, past_steer, max_throttle=0.5, max_brake=0.5, max_steer=0.8) # Regulated control of throttle, braking, and steering.
    - SetBrakeAction(brake) # Set the amount of brake. brake (float) Amount of braking between 0 and 1.
    - SetHandBrakeAction(handBrake) # Set or release the hand brake. handBrake (bool)
    - SetReverseAction(reverse) # Engage or release reverse gear. reverse (bool)
    - SetSteerAction(steer) # Set the steering 'angle'. steer (float) Steering ‘angle’ between -1 and 1.
- SetThrottleAction(throttle) # Set the throttle. throttle (float) Throttle value between 0 and 1.
- WalkingAction
This is a base class for the following actions;
    - SetWalkingDirectionAction(heading)
    - SetWalkingSpeedAction(speed)

- Steers # Mixin protocol for agents that can steer.
- Walks # Mixin protocol for agents that can walk in a given direction and speed.

Valid Scenic Attributes for network class:
- elements (Dict[str, NetworkElement]) # All network elements, indexed by unique ID.
- roads (Tuple[Road]) # All ordinary roads in the network (i.e., those not part of an intersection).
- connectingRoads (Tuple[Road]) # All roads connecting one exit of an intersection to another.
- allRoads (Tuple[Road]) # All roads of either type.
- laneGroups (Tuple[LaneGroup]) # All lane groups in the network.
- lanes (Tuple[Lane]) # All lanes in the network.
- intersections (Tuple[Intersection]) # All intersections in the network.
- crossings (Tuple[PedestrianCrossing]) # All pedestrian crossings in the network.
- sidewalks (Tuple[Sidewalk]) # All sidewalks in the network.
```

```
- shoulders (Tuple[Shoulder]) # All shoulders in the network (by default, includes parking lanes).
- roadSections (Tuple[RoadSection]) # All sections of ordinary roads in the network.
- laneSections (Tuple[LaneSection]) # All sections of lanes in the network.
- driveOnLeft (bool) # Whether or not cars drive on the left in this network.
- tolerance (float) # Distance tolerance for testing inclusion in network elements.
- roadDirection (VectorField) # Traffic flow vector field aggregated over all roads (0 elsewhere).
- drivableRegion (PolygonalRegion)
- walkableRegion (PolygonalRegion)
- roadRegion (PolygonalRegion)
- laneRegion (PolygonalRegion)
- intersectionRegion (PolygonalRegion)
- crossingRegion (PolygonalRegion)
- sidewalkRegion (PolygonalRegion)
- curbRegion (PolylineRegion)
- shoulderRegion (PolygonalRegion)

Additional information:
1- Even though you defined the ADV_SPEED like this 'param ADV_SPEED = Range(5, 8)', if you get a name error like this  -> NameError: name 'ADV_SPEED' is not defined,
you can use 'globalParameters.ADV_SPEED' where you use ADV_SPEED or you can configure 'param ADV_SPEED = Range(5, 8)' to 'ADV_SPEED = Range(5, 8)'.

2- If there is an "intersection" term in the scenario description, then you have to define
"intersection = Uniform(*filter(lambda i: i.is4Way, network.intersections))
egoInitLane = Uniform(*intersection.incomingLanes)
egoManeuver = Uniform(*filter(lambda m: m.type is ManeuverType.STRAIGHT, egoInitLane.maneuvers))
egoTrajectory = [egoInitLane, egoManeuver.connectingLane, egoManeuver.endLane]
egoSpawnPt = new OrientedPoint in egoInitLane.centerline" in the spatial relation part.

3- If you want to use a trajectory in some behavior (like 'behavior EgoBehavior(trajectory): '), you have to define the ego trajectory first, like in the previous example (2).

4- If you want to do a lane change behavior, then you have to write it in the following format:
"newLaneSec = self.laneSection.laneToRight
 do LaneChangeBehavior(laneSectionToSwitch=newLaneSec,target_speed=globa」
  lParameters.EGO_SPEED)"

5 - If you get an error similar to this "TypeError: "X relative to Y" with Y a vector but X a <class 'scenic.core.distributions.AttributeDistribution'> (expected Vector, got PolylineRegion)" you can try to delete the line which gives an error, and define the spawn point of a new agent relative to already defined spawn points.
```

```
"adversary2 = new Car behind egoSpawnPt by 20, # no need to create advSpawnPt2, use relative relation with egoSpawnPt, you can change 20 with another number
    with blueprint MODEL,
    with behavior FollowTrajectoryBehavior(target_speed=ADV_SPEED, trajectory=advTrajectory)"

6 - If some agents are stationary, you don't need to add behaviors for them. You can control it by checking the scenario 'DESCRIPTION' and 'SCENARIO SPECIFICATION'.

7 - If you get "AttributeError: 'Lane' object has no attribute 'laneToRight'" or "AttributeError: 'Lane' object has no attribute 'laneToLeft'", you should try "_laneToRight " or "_laneToLeft" in the initlane definitions. Here is a correct example:
"intersection = Uniform(*filter(lambda i: (i.is4Way and i.isSignalized), network.intersections))
egoInitLane = Uniform(*filter(lambda lane: all([sec._laneToRight is not None for sec in lane.sections]), intersection.incomingLanes))"

Here is another correct example:
"initLane = Uniform(*filter(lambda lane: all([sec._laneToLeft is not None for sec in lane.sections]), network.lanes))"

8 - If you get "ScenicSyntaxError: expected 'except' or 'finally' block", that means you used a try interrupt block incorrectly.
Here is a correct usage of a try-interrupt block:
"try:
    do FollowTrajectoryBehavior(target_speed=globalParameters.EGO_SPEED, trajectory=trajectory)
interrupt when withinDistanceToAnyObjs(self, globalParameters.SAFETY_DIST):
    take SetBrakeAction(EGO_BRAKE)"
```

C Generated Scenario Descriptions for Each Scenario Category

Table 8: Generated Scenario Descriptions for Scenario Tags

Scenario Category	Description of Scenarios
Straight Obstacle	1. The ego vehicle goes straight, and a pedestrian runs across the road without looking. 2. The ego vehicle goes straight on a road while two cyclists suddenly stop and then accelerate unpredictably. 3. The ego vehicle goes straight while 5 pedestrians run across the road without looking. 4. The ego vehicle goes straight and is sideswiped by a vehicle merging suddenly from a parking spot. 5. The ego vehicle goes straight and crashes into a pedestrian who passes in front of the suddenly stopped car.
Turning Obstacle	1. The ego vehicle is turning left when two pedestrians suddenly step onto the crosswalk from the sidewalk directly in its path. 2. The ego vehicle is turning left in a 4-way intersection when a vehicle ahead brakes abruptly. 3. The ego vehicle is turning left when a cyclist riding alongside the sidewalk abruptly swerves into the road across the ego's path, and a pedestrian crosses the road. 4. The ego vehicle is turning left in a 4-way intersection when a pedestrian suddenly emerges from the right of a stopped vehicle ahead, crossing directly into the ego vehicle's path. 5. The ego vehicle is turning left in a 4-way intersection when a vehicle ahead of the ego vehicle brakes abruptly before the turn.
Lane Changing	1. The ego vehicle is in the middle of a lane change when an adversarial truck in the adjacent lane abruptly drifts toward the ego's lane without signaling, creating a squeeze hazard on a multi-lane highway with wide lanes in overcast weather. 2. The ego vehicle is changing lanes when a parked vehicle ahead in the lane suddenly pulls out without warning on an urban multi-lane road with roadside parking in light drizzle. 3. The ego vehicle performs a lane change to pass two slow cars, but as it merges, one adversarial car brakes unexpectedly while another changes into the same lane from the opposite side on a busy highway with three lanes of traffic in overcast weather. 4. The ego vehicle is attempting to change lanes to avoid a slow-moving leading vehicle in the left lane; the adversarial car in the target lane suddenly accelerates on a clear day. 5. The ego vehicle makes a lane change on a highway to avoid a leading car, but fails to do so in time, resulting in a collision with a lane-changing adversarial car traveling in the same direction.
Vehicle Passing	1. The ego vehicle is attempting to pass a school bus stopped at a crosswalk while pedestrians are approaching the school bus. The adversarial school bus suddenly starts moving as the ego vehicle passes, narrowing the space and forcing the ego to brake while pedestrians are nearby. The event occurs on a clear day in an urban road with a crosswalk and roadside bus stop. 2. The ego vehicle is approaching road debris blocked by an adversarial car without noticing, while an adversarial car swerves around it without signaling. The adversarial vehicle ahead quickly swerves around an unexpected obstacle without signaling, obstructing the ego vehicle's view and giving it little time to react. The event occurs on a straight highway or arterial road with debris in the lane, under overcast weather conditions.

Scenario Category	Description of Scenarios
	3. The ego vehicle is attempting to pass a slow-moving bus on a crowded day. As the ego begins to pass, the adversarial bus swerves slightly toward the center of the road without signaling, squeezing the ego vehicle between the bus and oncoming traffic. 4. The ego vehicle is approaching an adversarial car from behind when the adversarial car loses steering control and brakes abruptly. The adversarial car ahead experiences a sudden steering fault and sharply decelerates without warning, forcing the ego vehicle to brake hard to avoid collision. The event occurs on a straight road with light traffic during rainy weather conditions. 5. The ego vehicle is driving behind an adversarial van when the van suddenly stops, and passengers begin unloading in the middle of the lane. Passengers exit the van from both sides into the roadway, unexpectedly obstructing the ego vehicle's path. The event occurs on a straight commercial street with moderate traffic during light rain.
Red-light Running	1. The ego vehicle goes straight at a 4-way intersection while an adversarial vehicle from the right lane proceeds through a red light and suddenly slows down in the middle of the intersection on a clear day. 2. The ego vehicle goes straight at a busy intersection when two adversarial vehicles from the opposite lane attempt illegal left turns on a red light, blocking the ego's path. The adversarial vehicles cut across the ego's lane during the turn, forcing the ego to either stop suddenly or risk collision. 3. The ego vehicle proceeds straight through a 4-way intersection while an adversarial truck from the lateral lane runs a red light and stalls mid-intersection on a cloudy day with poor visibility. 4. The ego vehicle moves straight through an intersection when two adversarial cars from opposite lateral lanes run the red light simultaneously, creating a cross-collision hazard on a foggy day. 5. The ego vehicle proceeds straight at a 4-way intersection while two motorcycles from the opposite lane attempt illegal left turns simultaneously, cutting off the ego vehicle on a rainy day.
Unprotected Left-turn	1. The ego vehicle is making an unprotected left turn when an oncoming vehicle suddenly stops, then accelerates through the intersection, ignoring the ego's right-of-way. 2. The ego vehicle is making an unprotected left turn when a vehicle behind it swerves into the intersection to overtake and causes a side collision. 3. The ego vehicle proceeds with an unprotected left turn while a pedestrian crosses the road, and an adversarial vehicle enters the intersection at high speed. 4. The ego vehicle starts an unprotected left turn while yielding to an oncoming car, when a cyclist traveling in the crosswalk area to the right of the oncoming car suddenly accelerates and enters the intersection. 5. The ego vehicle attempts an unprotected left turn, judging the gap between two oncoming vehicles, when the second oncoming vehicle unexpectedly accelerates to tailgate the first, closing the gap too quickly.
Right-turn	1. The ego vehicle is turning right when a pedestrian suddenly steps onto the crosswalk from the sidewalk directly in its path. 2. The ego vehicle is turning right in a 4-way intersection when a vehicle from the left runs a red light and enters the intersection at high speed. 3. The ego vehicle is turning right when a cyclist riding alongside the sidewalk abruptly swerves into the road across the ego's path.

Scenario Category	Description of Scenarios
	4. The ego vehicle is turning right in a 4-way intersection when a pedestrian suddenly emerges from in front of a stopped vehicle ahead, crossing directly into the ego vehicle's path. 5. The ego vehicle is turning right in a 4-way intersection when a vehicle behind aggressively overtakes on the left and cuts in front, braking abruptly.
Crossing Negotiation	1. The ego vehicle goes straight at a 4-way intersection while a vehicle from the right performs a rolling stop, followed by a second vehicle that tailgates and tries to squeeze through, creating a conflict zone in front of the ego. 2. The ego vehicle goes straight at a 4-way intersection as two adversary vehicles, one from the right and one from the opposite lane, attempt to cross simultaneously, forcing the ego into an evasive maneuver. 3. The ego vehicle goes straight at a 4-way intersection while an adversary vehicle from the left attempts an unprotected left turn, occluded by a large truck in the adjacent lane, requiring the ego to brake late. 4. The ego vehicle goes straight at a 4-way intersection and must suddenly stop to avoid a collision when an adversary vehicle from the opposite lane makes a left turn on a rainy day. 5. The ego vehicle goes straight at a 4-way intersection and must suddenly stop to avoid a collision when an adversary vehicle from the opposite lane makes a left turn. At the same time, another adversarial vehicle turns left from the right lane.

D Generated Scenic Codes

D.1 Generated Scenic Code for Red-Light Running Scenario Category

```
"""DESCRIPTION: Ego vehicle moves straight through an intersection when
↪  two adversarial cars from opposite lateral lanes run the red light
↪  simultaneously, creating a cross-collision hazard on a foggy day."""
################################
# MAP AND MODEL                #
################################

param map = localPath('../maps/Town05.xodr')
param carla_map = 'Town05'
model scenic.simulators.carla.model

################################
# CONSTANTS                    #
################################

MODEL = 'vehicle.lincoln.mkz_2017'

EGO_INIT_DIST = [20, 25]
param EGO_SPEED = Range(7, 10)

ADV_INIT_DIST = [15, 20]
param ADV_SPEED = Range(7, 10)
param ADV_BRAKE = Range(0.5, 1.0)

param SAFETY_DIST = Range(5, 10)
CRASH_DIST = 5
TERM_DIST = 70
```

```
param weather = {
    'cloudiness': 100.0,
    'precipitation': 0.0,
    'sun_altitude_angle': 40.0,
    'precipitation_deposits': 0.0,
    'wind_intensity': 20.0,
    'fog_density': 90.0,
    'wetness': 30.0
}

################################
# AGENT BEHAVIORS              #
################################

behavior EgoBehavior(trajectory):
    if withinDistanceToTrafficLight(self, 100):
        setClosestTrafficLightStatus(self, "green")
    do FollowTrajectoryBehavior(target_speed=globalParameters.EGO_SPEED,
    ↪  trajectory=trajectory)

behavior AdversaryBehavior(trajectory):
    try:
        if withinDistanceToTrafficLight(self, 100):
            setClosestTrafficLightStatus(self, "red")
        do FollowTrajectoryBehavior(target_speed=globalParameters.ADV_SP
        ↪  EED,
        ↪  trajectory=trajectory)
    interrupt when withinDistanceToAnyObjs(self,
    ↪  globalParameters.SAFETY_DIST):
        take SetBrakeAction(globalParameters.ADV_BRAKE)

################################
# SPATIAL RELATIONS            #
################################

intersection = Uniform(*filter(lambda i: i.is4Way and i.isSignalized,
↪  network.intersections))

egoInitLane = Uniform(*intersection.incomingLanes)
egoManeuver = Uniform(*filter(lambda m: m.type is ManeuverType.STRAIGHT,
↪  egoInitLane.maneuvers))
egoTrajectory = [egoInitLane, egoManeuver.connectingLane,
↪  egoManeuver.endLane]
egoSpawnPt = new OrientedPoint in egoInitLane.centerline

advInitLane1 = Uniform(*filter(lambda m:
        m.type is ManeuverType.STRAIGHT,
        Uniform(*filter(lambda m:
            m.type is ManeuverType.STRAIGHT,
            egoInitLane.maneuvers)
        ).conflictingManeuvers)
    ).startLane
advManeuver1 = Uniform(*filter(lambda m: m.type is ManeuverType.STRAIGHT,
↪  advInitLane1.maneuvers))
```

```
advTrajectory1 = [advInitLane1, advManeuver1.connectingLane, advManeuver1.endLane]
advSpawnPt1 = new OrientedPoint in advInitLane1.centerline

advInitLane2 = Uniform(*filter(lambda m:
        m.type is ManeuverType.STRAIGHT,
        Uniform(*filter(lambda m:
            m.type is ManeuverType.STRAIGHT,
            egoInitLane.maneuvers)
        ).conflictingManeuvers)
    ).startLane
advManeuver2 = Uniform(*filter(lambda m: m.type is ManeuverType.STRAIGHT, advInitLane2.maneuvers))
advTrajectory2 = [advInitLane2, advManeuver2.connectingLane, advManeuver2.endLane]
advSpawnPt2 = new OrientedPoint in advInitLane2.centerline

################################
# SCENARIO SPECIFICATION       #
################################

ego = new Car at egoSpawnPt,
    with blueprint MODEL,
    with behavior EgoBehavior(egoTrajectory)

adversary1 = new Car at advSpawnPt1,
    with blueprint MODEL,
    with behavior AdversaryBehavior(trajectory = advTrajectory1)

adversary2 = new Car at advSpawnPt2,
    with blueprint MODEL,
    with behavior AdversaryBehavior(trajectory = advTrajectory2)

require EGO_INIT_DIST[0] <= (distance to intersection) <= EGO_INIT_DIST[1]
require ADV_INIT_DIST[0] <= (distance from adversary1 to intersection) <= ADV_INIT_DIST[1]
require ADV_INIT_DIST[0] <= (distance from adversary2 to intersection) <= ADV_INIT_DIST[1]
terminate when (distance to egoSpawnPt) > TERM_DIST
```

E Risk evaluation metrics

The metrics used for comparison are provided by SafeBench [6] as follows:

1. Collision Rate (CR): The metric that shows how frequently collision occurs. High collision can be an indicator of complex scenarios when combined with other metrics.
2. Red Light Running (RR): Frequency of cases in which the red light running occurs.
3. Stop Sign Running (SS): Frequency of the vehicle missing the stop sign and continuing to drive.
4. Out of Road Driving (OR): It is the average measure of driving in the non-drivable area.
5. Route Following Stability (RF): The measure of how stable the ego agent is driving a road. It is the measure of the lane following capabilities of the ego.
6. Route Completion (Comp): The measure of route completion.
7. Completion Time (TS): Average time spent to complete the route.

8. Average Acceleration (ACC): This score measures the average acceleration throughout the route, indicating the level of driving comfort.
9. Yaw Velocity (YV): Average yaw velocity, indicates turning characteristics of the vehicle.
10. Lane Invasion Frequency (LI): It measures how often lane invasions occur, which indicates the accuracy of lane-keeping of the vehicle.
11. Overall Score (OS): Overall score is $OS = \sum_{i=1}^{10} w^i \times g(m^i)$ a weighted average of all scores described here. For each ith metric m^i, $g(m^i)$ is calculated as:

$$g(m^i) = \begin{cases} \dfrac{m^i}{m^i_{\max}}, & \text{if } m^i \text{ is "the higher the better",} \\ 1 - \dfrac{m^i}{m^i_{\max}}, & \text{if } m^i \text{ is "the lower the better".} \end{cases} \tag{1}$$

Lower OS indicates how hard/risky the scenario is. The weights and constants associated with the metrics are provided in Table 9.

Table 9: SafeBench evaluation metrics along with their respective weights and constants [6]

Sym.	**Safety Level**				**Functionality Level**			**Etiquette Level**		
	CR ↑	RR ↑	SS ↑	OR ↑	RF ↓	Comp ↓	TS ↑	ACC ↑	YV ↑	LI ↑
$m^i_{\max}$	1	1	1	50	1	1	60	8	3	20
w^i	0.495	0.099	0.099	0.099	0.050	0.050	0.050	0.020	0.020	0.020

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper's contributions and the scope are given in the abstract and section 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are described section 4 and section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: It can be reproduced by following the steps described in the GitHub repository. The codes will be released upon acceptance.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The codes will be released upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Test details can be found in the section 3 and section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification:

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: It can be found in the section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper utilizes only publicly available crash reports, focusing solely on the textual descriptions within these reports.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Utilized codes and assets are cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification:

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.