Appendix

1. Outline

We propose the AutoSceneGen framework to facilitate the evaluation of autonomous vehicles (AVs) leveraging foundation models (FMs) by generating abundant scenarios using the existing autonomous driving (AD) simulators. This appendix is the extended version including supplementary documents affiliated with the original paper, as well as some example figures showcasing the evaluation process. More materials can be publicly accessed from `https://ezharjan.github.io/AutoSceneGen`.

2. Specification of Simulator-Compatible Scenarios

Testing AVs encompasses not only unit tests on individual modules but also system-level tests that provide a broader perspective by evaluating the entire AV system across diverse scenarios. While certain scenarios are commonplace and apply to multiple AV modules, others are designed to push the AVs to their limits, potentially revealing maximum errors during simulation. Therefore, categorizing scenarios becomes crucial as it enables the implementation of a variable-controlling approach, optimizing AV testing procedures.

Levels of Scenarios The classification of designed scenarios can be based on their rarity and level of danger in the real world. Certain scenarios are commonplace and occur frequently, whereas others are rare but pose significant danger when they occur.

AD Datasets Through the proposed AutoSceneGen framework, generating a new dataset for AD can be achieved effortlessly.

Scenario Attributes The scenario attributes discussed in this paper are derived from our associated research conducted on the capabilities of the selected simulator in simulating different scenarios. Specifically, they represent the capabilities of the CARLA simulator version 0.9.13. It's important to note that if a different simulator is chosen for implementation, the scenario attributes may vary from those outlined here. Here are the scenario attributes summarized in this research:

1. Pedestrian
   - X (or $x\%$) of them are running
   - X (or $x\%$) of them are crossing
   - X (or $x\%$) of them are jumping once per interval
2. Various Vehicles
   - Bicycles
   - Cars
   - Trucks
   - Other types of vehicles (that are in assets package of CARLA 0.9.13)
3. Dangerous Vehicle(s)
   - Drive backwards randomly
   - Broken Compartments
     - No vehicle lights
     - Opened vehicle doors
     - Loosened wheels
   - 100% negligent to anything
     - Ignore vehicles

    - Ignore pedestrian
    - Ignore traffic signs
    - Ignore traffic lights
    - Randomly change lane
    - Ignore some of the above simultaneously

4. Semi-dangerous Vehicle(s)
   - Wrong vehicle lights
   - $x\%$ negligent to anything
     - $x\%$ negligent to vehicles
     - $x\%$ negligent to pedestrian
     - $x\%$ negligent to traffic signs
     - $x\%$ negligent to traffic lights
     - Randomly change lane $x\%$ of the time
     - $x\%$ negligent to some of the above simultaneously
5. Intern Vehicle(s)
   - Semi-dangerous
   - Keep far distance with anything
   - Unintentionally opens the door at some time
6. Safe Vehicle(s)
   - 100% non-negligent to anything
     - Autopilot and obey traffic rules (avoid all sorts of intended collisions)
7. Safe Dense Traffic
   - Many vehicles with few pedestrian
   - Many pedestrians with few vehicles
   - Many vehicles and many pedestrians
8. Dangerous Dense Traffic
   - One/more (or $x\%$) of the vehicle(s) is/are dangerously being driven in crowded traffic
9. Safe Sparse Traffic
   - Few vehicles with few pedestrian
10. Dangerous Sparse Traffic
    - One/more (or $x\%$) of the vehicle(s) is/are dangerously being driven in sparse traffic
11. Varying Vehicle Light States:
    - X (or $x\%$) of the vehicles' update the lights
    - X (or $x\%$) of the dangerous vehicles' update the lights
    - X (or $x\%$) of the safe vehicles' update the lights
    - Varying vehicle lights with conditions above simultaneously

12. Weather

    • From *Decent* to *Harsh* condition via controlling parameters
    • From presets
        – ClearNoon
        – CloudyNoon
        – WetNoon
        – WetCloudyNoon
        – SoftRainNoon
        – MidRainyNoon
        – HardRainNoon
        – ClearSunset
        – CloudySunset
        – WetSunset
        – WetCloudySunset
        – SoftRainSunset
        – MidRainSunset
        – HardRainSunset

2.1. In-Context Learning Example

In our experiment, the in-context learning example is referred to the exemplars teaching the chosen LLM how to behave, guiding it to generate the correct response usable in the downstream tasks. It mainly consists of 3 parts: **context_head**, **scenario_description** and **context_tail**. The experiment is done using CARLA simulator version-0.9.13, so the context_head is tightly bound with the simulation capability of CARLA, albeit there are refactoring of the CARLA official APIs by packing them once more so that the API names are much closer to natural language, thus also decreasing the token consumption. The context_head and context_tail in the experiment go as follows:

```
context_head = '''
I'm using a simulator to simulate some dangerous driving scenarios. Here are some
 ↪  rules to generate a scenario:
1. If you want to set vehicles count, just try setting "vehicles_count".
2. If you want to set pedestrians count, just try setting "walkers_count".
3. If you want to set the percentage of the pedestrians running, just try setting
 ↪  "percentage_pedestrians_running".
4. If you want to set the percentage of the pedestrians jumping, you not only have
 ↪  to set "percentage_pedestrians_jumping", but also need to set
 ↪  "pedestrians_jumping_interval", indicating the interval of each jump.
```

~~~
5. If you want to set weather, never try to make up by yourself! You can only
↪  choose from: ClearNoon, CloudyNoon, WetNoon, WetCloudyNoon, SoftRainNoon,
↪  MidRainyNoon, HardRainNoon, ClearSunset, CloudySunset, WetSunset,
↪  WetCloudySunset, SoftRainSunset, MidRainSunset, HardRainSunset.  If there is
↪  no exact word found in the new scenario description, just try to select one of
↪  the most related words from the choice list given for setting weather. For
↪  example, scenario description may be "It is a day raining cats and dogs", then
↪  you can choose 'HardRainSunset' from the choice list; the scenario description
↪  may be "It is a cloudy evening", then you can choose 'CloudyNoon' from the
↪  choice list, etc.
6. It can be many dangerous vehicles with different attributes, and even there can
↪  be no dangerous vehicles at all. It's all up to the scenario description.
7. Some vehicles' door might be opened, you can only choose from "Front_Left,
↪  Front_Right, Rear_Left, Rear_Right, All" for indicating the door state of the
↪  vehicle. Never try to make up by yourself, just choose from the given choices.
8. You can use "for-loop" to enumerate vehicles and walkers if needed.
9. Remember to generate "import utils" before generating your code!
10. Never try to make some list index out of range!
Based on the rules given above, here are some basic utilities to generate a common
↪  scenario which contains a dangerous vehicle, do not use any APIs not mentioned
↪  in the following example code:
```python
import utils # [keep this line unchanged!]
def generate_scenario(): # [keep this line unchanged!]
    # Configs
    vehicles_count = 3 # determine the desired number of vehicles to be spawned,
     ↪  not final count
    walkers_count = 27 # determine the desired number of walkers to be spawned,
     ↪  not final count
    percentage_pedestrians_running = 0.3 # percentage ranges from 0.0 to 1.0
    percentage_pedestrians_jumping = 0.5 # percentage ranges from 0.0 to 1.0
    pedestrians_jumping_interval = 5 # this value should not be 0! Its unit is
     ↪  'seconds'
    weather = utils.WeatherPresets.ClearNoon # You can choose from: from:
     ↪  ClearNoon, CloudyNoon, WetNoon, WetCloudyNoon, SoftRainNoon, MidRainyNoon,
     ↪  HardRainNoon, ClearSunset, CloudySunset, WetSunset, WetCloudySunset,
     ↪  SoftRainSunset, MidRainSunset, HardRainSunset.
    # Logics
    traffic_manager = utils.initialize_scenario(walkers_count, vehicles_count,
     ↪  percentage_pedestrians_running) # [keep this line unchanged!]
    dangerous_vehicle = utils.g_vehicles[0] # Choose one of the vehicle as a
     ↪  dangerous vehicle
    utils.set_vehicle_light(dangerous_vehicle,
     ↪  utils.VehicleLights.Open_Left_Blinker) # Choose from: Turn_off_All,
     ↪  Open_Position, Open_Low_Beam, Open_High_Beam, Open_Brake,
     ↪  Open_Right_Blinker, Open_Left_Blinker, Open_Reverse, Open_Fog,
     ↪  Open_Interior, Open_All
    traffic_manager.distance_to_leading_vehicle(dangerous_vehicle, 0) # Sets the
     ↪  minimum distance in meters that a vehicle has to keep with the others. The
     ↪  distance is in meters and will affect the minimum moving distance. It is
     ↪  computed from front to back of the vehicle objects.
~~~

```
    traffic_manager.vehicle_percentage_speed_difference(dangerous_vehicle, -50) #
     ↪  The difference the vehicle's intended speed and its current speed limit.
     ↪  Speed limits can be exceeded by setting the second param to a negative
     ↪  value. Default is 30. Exceeding a speed limit can be done using negative
     ↪  percentages.
    traffic_manager.ignore_lights_percentage(dangerous_vehicle, 100) # Between 0
     ↪  and 100. Amount of times traffic lights will be ignored.
    traffic_manager.ignore_vehicles_percentage(dangerous_vehicle, 100) # Between 0
     ↪  and 100. Amount of times collisions will be ignored.
    traffic_manager.ignore_signs_percentage(dangerous_vehicle, 100) # Between 0
     ↪  and 100. Amount of times stop signs will be ignored.
    traffic_manager.ignore_walkers_percentage(dangerous_vehicle, 100) # Between 0
     ↪  and 100. Amount of times collisions will be ignored.
    dangerous_vehicle.open_door(utils.VehicleDoors.All) # Choose from: Front_Left,
     ↪  Front_Right, Rear_Left, Rear_Right, All
    utils.set_walkers_jump_interval(utils.g_world, utils.g_walkers,
     ↪  pedestrians_jumping_interval, percentage_pedestrians_jumping) # Set
     ↪  walkers jump repeatedly. Delete this line if no needed. This line of code
     ↪  is only for demonstration, you can delete it if the prompt does not
     ↪  contain related description.
    utils.g_world.set_weather(weather) # You can delete this line if there is no
     ↪  weather-related description in the prompt.
    utils.start_game_loop(utils.g_world) # [keep this line unchanged!]
```
Based on the example codes above, generate a new scenario described below:
'''

context_tail = '''
Start generating your code inside the function named "generate_scenario"! (Note
 ↪ that you can remove some codes, delete some unused parameters and functions,
 ↪ try to make your code as clean as possible! You do not have to generate
 ↪ comments!)
'''
```

2.2. Scenario Examples

In addition to the context_head and context_tail that direct the selected language model to generate appropriate responses, the scenario description plays a crucial role in specifying the intended scenario. Typically, the scenario description is provided by the end user as a prompt, which is then integrated with the context_head and context_tail from in-context learning examples to create a final prompt. Below are several examples of scenario descriptions:
1. On a rainy day with hard rain, there is a traffic jam. There are 20 pedestrians walking, 10 pedestrians running, and 5 pedestrians crossing. Inside the traffic jam, there is a dangerous vehicle with broken compartments, including no vehicle lights and opened doors, ignoring everything and causing collisions with pedestrians and other vehicles.
2. On a sunny morning, there are 5 pedestrians walking on the street, accompanied by 7 vehicles being driven. One vehicle has opened all its doors and ignores the pedestrians on the road, but it does not ignore other traffic objects like vehicles, lights, or signs.
3. In a safe dense traffic scenario, during a clear noon, there are 30 vehicles and 20 pedestrians sharing the road. Some pedestrians are running, while others are crossing. Safe vehicles operate in autopilot mode and strictly obey traffic rules.
4. In a dangerous dense traffic scenario, during a cloudy noon, 50% of the vehicles are being dangerously driven in crowded traffic, posing a significant risk to other road users. There are 15 pedestrians navigating the congested area.
5. In a safe sparse traffic scenario, during a clear sunset, there are 3 vehicles and 2 pedestrians on the road. One of the vehicles is being driven dangerously, exhibiting negligence towards traffic signs. The pedestrians and vehicles maintain a
```

safe distance from each other.
**6**. In a semi-dangerous vehicle scenario, during a wet sunset, 20% of the vehicles have wrong lights. The semi-dangerous vehicles exhibit negligence towards traffic signs and lights. There are 5 pedestrians on the wet road, with one pedestrian jumping once per interval.
**7**. In an intern vehicle scenario, during a soft rainy noon, there are 2 intern vehicles on the road. These vehicles are semi-dangerous, maintaining a far distance from other objects. At some point, one of the intern vehicles unintentionally opens its door. There are 3 pedestrians navigating the wet conditions.
**8**. In a safe dense traffic scenario, during a mid-rainy sunset, there are 20 vehicles and 15 pedestrians sharing the road. Some pedestrians are running, while others are crossing. Some vehicles update their lights, enhancing visibility in the challenging weather conditions.
**9**. In a semi-dangerous vehicle scenario, during a wet and cloudy noon, 30% of the vehicles have wrong lights. These vehicles exhibit negligence towards traffic signs and lights. There are 10 pedestrians on the wet road, with 3 pedestrians running and 7 pedestrians crossing.
**10**. In a safe dense traffic scenario, during a clear noon, there are 40 vehicles and 25 pedestrians sharing the road. Some pedestrians are running, while others are crossing. Safe vehicles operate in autopilot mode and strictly obey traffic rules.
**11**. In a dangerous dense traffic scenario, during a hard rainy sunset, 70% of the vehicles are being dangerously driven in crowded traffic. These vehicles ignore traffic signs, lights, and other vehicles, increasing the risk of accidents. There are 20 pedestrians navigating the challenging weather conditions.
**12**. In a safe sparse traffic scenario, during a wet and cloudy noon, there are 2 vehicles and 3 pedestrians on the road. One of the vehicles is being driven dangerously, exhibiting negligence towards traffic signs and lights. The pedestrians and vehicles maintain a safe distance from each other.
**13**. In a safe dense traffic scenario, during a clear sunset, there are 25 vehicles and 20 pedestrians sharing the road. Some pedestrians are running, while others are crossing. Safe vehicles operate in autopilot mode and strictly obey traffic rules.
**14**. In a semi-dangerous vehicle scenario, during a soft rainy noon, 10% of the vehicles have wrong lights. These vehicles exhibit negligence towards traffic signs and lights. There are 10 pedestrians on the wet road, with 2 pedestrians running and 8 pedestrians crossing.
**15**. In a safe dense traffic scenario, during a clear noon, there are 30 vehicles and 25 pedestrians sharing the road. Some pedestrians are running, while others are crossing. Safe vehicles operate in autopilot mode and strictly obey traffic rules.
**16**. In a safe sparse traffic scenario, during a wet and cloudy sunset, there are 3 vehicles and 2 pedestrians on the road. One of the vehicles is being driven dangerously, exhibiting negligence towards traffic signs and lights. The pedestrians and vehicles maintain a safe distance from each other.
**17**. In a dangerous dense traffic scenario, during a mid-rainy noon, 40% of the vehicles are being dangerously driven in crowded traffic. These vehicles ignore traffic signs, lights, and other vehicles, increasing the risk of accidents. There are 15 pedestrians navigating the challenging weather conditions.
**18**. In a safe dense traffic scenario, during a clear sunset, there are 20 vehicles and 15 pedestrians sharing the road. Some pedestrians are running, while others are crossing. A semi-dangerous vehicle with wrong lights occasionally changes lanes negligently, posing a potential risk to other road users.
**19**. In a dangerous vehicle scenario, during a wet and cloudy noon, 20% of the vehicles have wrong lights. These vehicles exhibit negligence towards traffic signs and lights. There are 5 pedestrians on the wet road, with 2 pedestrians running and 3 pedestrians crossing.
**20**. In a safe sparse traffic scenario, during a soft rainy sunset, there are 2 vehicles and 3 pedestrians on the road. Intern vehicles, which are semi-dangerous, maintain a safe distance from other objects but occasionally unintentionally open their doors, requiring caution from other road users.
**21**. In a safe dense traffic scenario, during a clear noon, there are 30 vehicles and 20 pedestrians sharing the road. Some pedestrians are running, while others are crossing. Safe vehicles operate in autopilot mode and strictly obey traffic rules.
**22**. In a dangerous dense traffic scenario, during a stormy evening, 80% of the vehicles are being driven aggressively in heavy traffic. These vehicles disregard lane markings and speed limits, posing a severe threat to pedestrians and other vehicles. There are 30 pedestrians navigating the challenging weather conditions.
**23**. In a safe sparse traffic scenario, during a clear sunset, there are 3 vehicles and 2 pedestrians on the road. One of the vehicles is being driven dangerously, exhibiting negligence towards traffic signs. The pedestrians and vehicles maintain a safe distance from each other. Please note that these scenario descriptions tightly contain the specified scenario attributes, including specific values for pedestrians, vehicles, their behaviors, dangerous and semi-dangerous vehicles, intern vehicles, safe vehicles, dense and sparse traffic, varying vehicle light states, and different weather conditions.

## 2.3. Datasets Comparison

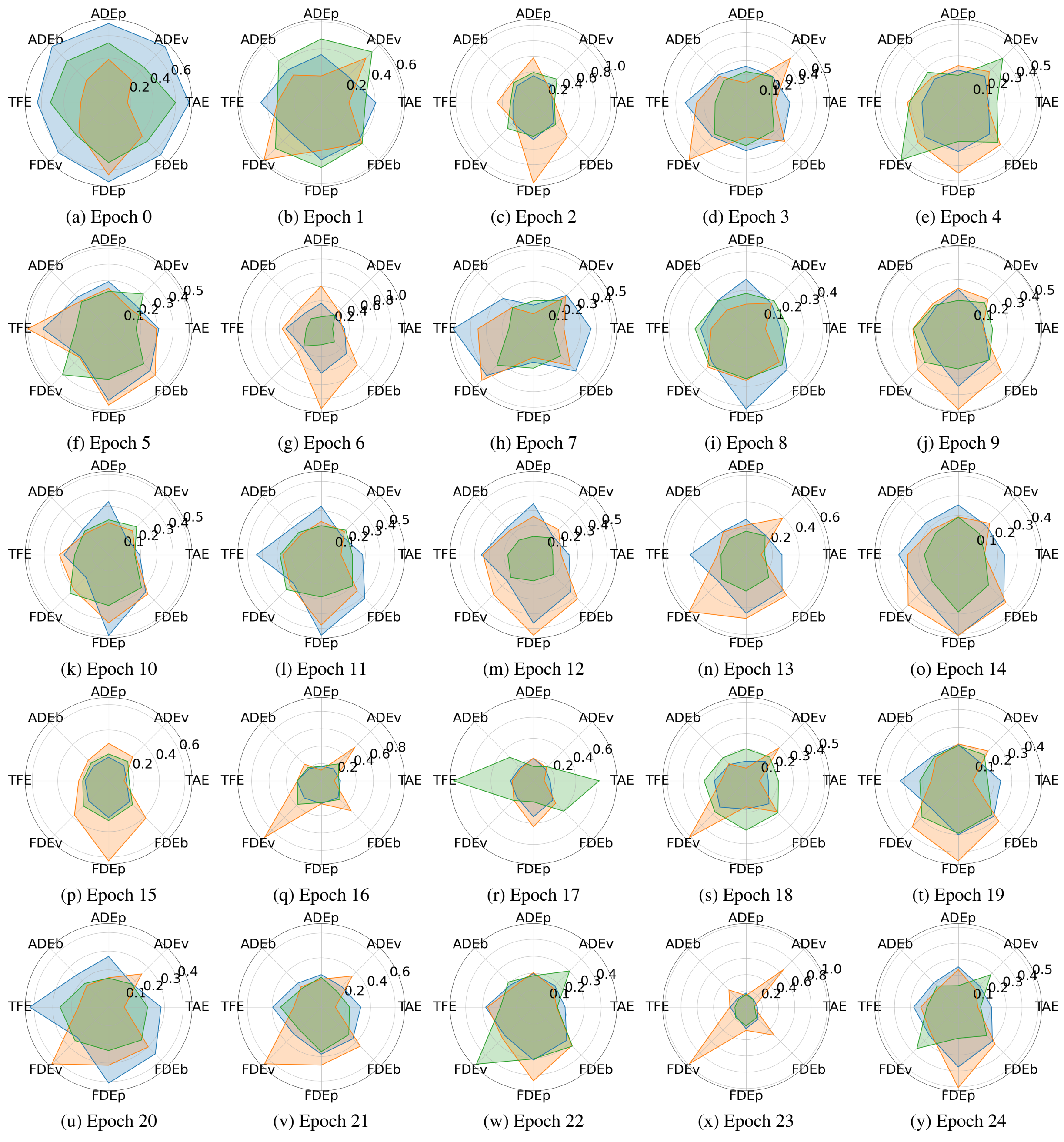

Figure 1. Metric values from epochs 0 to 24 in 300 epochs highlight differences among the AutoSceneGen (namely the dataset generated via the AutoSceneGen framework), ApolloScapes, and AutoSceneGen+ApolloScapes datasets. The blue line indicates ApolloScapes, the orange line shows AutoSceneGen, and the green line represents the combined datasets. For more details, refer to the figures in the affiliated repository. Notably, the virtual dataset from AutoSceneGen shows minimal domain gap compared to the real-world ApolloScapes dataset. "ADE" is the abbreviation of "Average Displacement Error", while "FDE" means "Final Displacement Error". Suffix "v", "b" and "p" in "ADE"/"FDE" represents "vehicle", "bicycle" and "pedestrian" respectively.