Scenario Gym: a scenario-centric lightweight simulator

Hamish Scott[1], Lorenzo Niccolini[1] , Chess Stetson[1] , Nils Goldbeck[1], Ioannis Souflas[2], Noyan Songur[3], Alireza Ahrabian[3], Eduardo Candela[4], Panagiotis Angeloudis[4]

***Abstract*— With the rise of simulation in the pursuit of safe autonomous vehicles (AVs) there is a need to be able to work with scenario data in a way that is unrestrictive in its inputs, allows fast reactive simulation of scenarios and enables powerful insight into the interactions and risks present in the data. We propose Scenario Gym[1] as a universal autonomous driving tool to address these needs. Scenario Gym is an open-source, lightweight simulator that allows fast execution of unconfined, complex scenarios containing a range of road users with the ability to gain rich insight via customised metrics and includes a framework for designing intelligent agents for reactive simulation.**

I. INTRODUCTION

Producing autonomous vehicles that are truly safe will rely on training and testing on a wide range of scenarios. A scenario captures a specific interaction of vehicles, pedestrians and other road users on a road configuration, and is the atomic unit for testing an AV, often in simulation [1], [2], before deployment [3]. We identified three main challenges in this domain: high computation and implementation cost of running large batches of scenarios, lack of data input standardisation, and lack of a flexible framework for training and deploying intelligent vehicle and non-vehicle agents in the same scenario. This work addresses these challenges with a new tool, Scenario Gym (Figure 1).

One of the key challenges in simulating realistic driving is the high computation and implementation cost associated with complex road networks, multi-agent interactions and different sensor data representations. This limits the number of scenarios that can be used to train and test AV systems, subsequently limiting the exposure of AV systems to a wide enough range of possible traffic phenomena. Even testing against all the collisions occurring in the US in one year would require running hundreds of thousands of scenarios, and orders of magnitude more to train the AV to avoid them and their plausible variations. This creates a need for a lightweight framework that can evaluate and run scenarios in milliseconds while maintaining the flexibility to input road networks, agent intelligences and data representations.

Having a wealth of datasets and simulators each with different setup costs and input representations means that the integration time of new data sources or using a new tool is often high. Moreover, many have restricted scope for customised input which is especially limiting when the user has their own scenario data that they wish to use. The OpenSCENARIO [4] and OpenDRIVE [5] standards for scenario and road network representation have started to address this problem by allowing users to store scenario data from different sources in a common format. A tool that can make use of these standards and enable users to quickly work with new scenario data from multiple sources can further open the field.

Moreover, managing datasets composed of large numbers of scenarios is a highly complex task. Understanding the range of phenomena present in a given scenario dataset is critical to ensuring that an AV system is trained on the right data. In addition, the use of data augmentation (a technique which is extremely common in other applications of AI [6]) on scenario data is difficult due to the lack of a standard in-memory scenario representation that can be used to define custom forms of augmentation or for other downstream process.

Previous simulators and toolsets designed for leveraging scenario data [1] have a limited scope for intelligent agent modelling, e.g. by requiring the use of particular policies to control vehicles [7], [8] or only allowing simple traffic models [9]. When reactive simulation is available it can often only be applied to vehicle agents, whereas pedestrians, cyclists and other vulnerable road users behavior will be crucial for effective safety tests of AVs.

We introduce Scenario Gym to address these challenges. Scenario Gym is a fully open-source, lightweight and modular framework that enables the rapid simulation of scenarios with customised intelligent agents of all kinds and informative metrics and observers. It defines a flexible in-memory scenario representation allowing users to perform a huge range of tasks with scenario data. It is implemented in Python [10] for ease and speed of use [11] and broad compatibility [12]. It aims to make working with scenario data as easy as possible.

Open Input: Scenario Gym defines a flexible scenario representation that is compatible with the OpenSCENARIO description language and OpenDRIVE road network representation. Scenarios are loaded into memory whereupon they may be analysed, modified or simulated. In simulation entities can adopt predefined trajectories, or be controlled intelligently by an agent. Scenarios are simulated synchronously in discrete time steps within which each agent selects an action and the pose of each entity is updated before moving to the next step.

Modular Intelligence: Reactive agents interact with the environment through a simple sensor-agent-controller architecture. This streamlines the agent design by splitting it into

Corresponding author email: hamish@drisk.ai. Author affiliations: [1] dRISK, [2] former Hitachi Europe Ltd., [3] Hitachi Europe Ltd., and [4] Imperial College London.

[1]https://github.com/driskai/scenario_gym

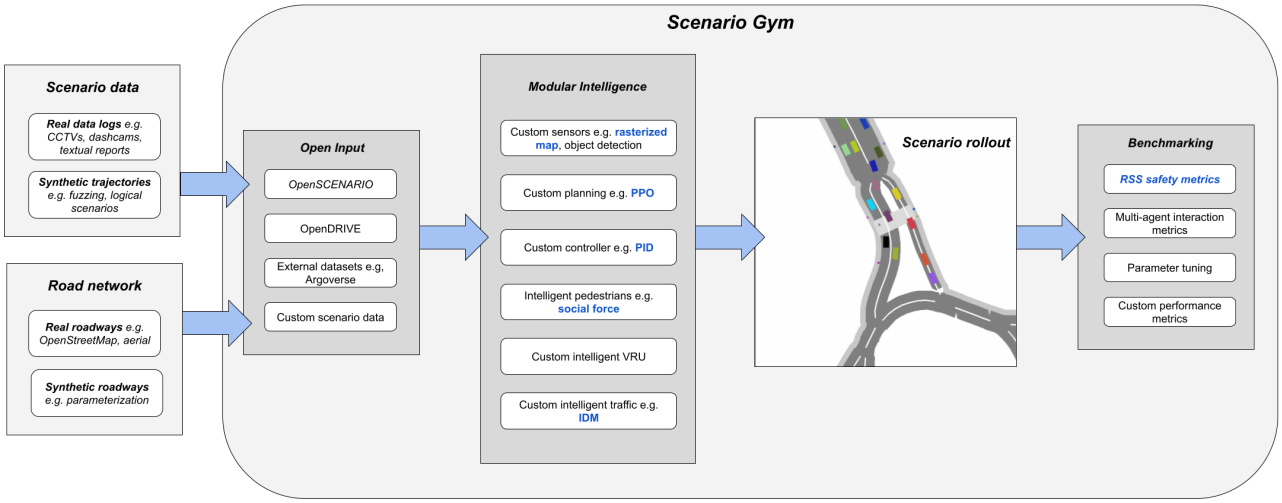


Fig. 1: Scenario Gym overview.

three modules that emulate the design of autonomous agent systems. The sensor module produces a local observation for the agent from the current global state of the environment. The agent then selects an action and passes it to the controller. The controller manages the physical model of the agent e.g. converting steering and acceleration commands into a new pose. This modular architecture provides reusability and quick iteration of agent designs, not only for vehicular agents but also pedestrians, bicycles and other vulnerable road users.

Benchmarking: Custom metrics can be implemented to allow quick and specific yet comprehensive insights. Through the scenario representation these can be constructed to efficiently track statistics such as speeds and distances, to record events such as collisions and near misses or to capture more compound measures such as safe distances and risk measures. In this paper, we use metrics for manoeuvre classification and hazard detection, as well as an implementation of the Responsibility-Sensitive-Safety metrics [13].

Scenario Gym allows fast development and insight into agent performance through its API and modular architecture and rapid access to results via lightweight simulation. Moreover, when sensor-realistic 3D simulation is required Scenario Gym can be configured as a bridge on top of a powerful 3D simulator. Scenario Gym communicates agent positions and/or action commands to the underlying simulator and receives updated poses in return. This way all agents and metrics are run from Scenario Gym enabling a huge range of agents models and metrics to be re-used in the simulator of choice.

We release Scenario Gym open source under the MIT license [14]. Our code follows many production standards to ensure ease of use and reliability and uses a minimal number of requirements. We provide integrations for using Scenario Gym with OpenAI Gym [15] for experiments in reinforcement learning and interfaces for using data sources such as the Argoverse [16] and nuScenes [17] datasets.

To demonstrate these functionalities, our experiments show how Scenario Gym facilitates a number of useful tasks relevant to AV development and research:

- Analyse the feature distributions of large public AV datasets using metrics to track high risk behaviours such as safe distances, collisions and speeds.
- Implement intelligent agents for reactive simulation (demonstrated via a social force model to govern the dynamics of pedestrians in simulation).
- Train a vehicle agent with deep reinforcement to navigate scenarios with crowds spilling into the road.

II. RELATED WORK

A great variety of simulators and learning environments have been proposed to facilitate training and testing of AV systems on scenario data [1]. These include physical simulators [2], learning environments for deep reinforcement learning [7], [8], [18], [9] and traffic simulators [19]. We examine these and compare their capabilities for scenario representation and reactive simulation with intelligent agents and metric benchmarking to Scenario Gym.

3D simulators for sensor-physical simulation [2] accurately recreate physical effects, such as collisions, weather conditions and scene illumination. However, due to the high computation required, it is impossible to run large numbers of scenarios faster than real time in serial, and prohibitively expensive to do so in parallel. Moreover, even for those compatible with OpenSCENARIO and OpenDRIVE, a limited selection of simulation assets (such as types of vehicles or props) and road networks limits the benefits of the sensor realism.

While full 3D sensor realism may be necessary for some parts of AV development, lightweight 2D simulators offer an efficient solution for training and testing planning, motion control and much of decision making under uncertainty. Scenario Gym allows fast iteration and learning for modules that are not dependent on 3D rendering (e.g. path planning) but can also be configured to run on top of another simulator if high-fidelity physical models are required. Moreover

Scenario Gym's in-memory scenario representation can be used to analyse scenario data with ease and perform augmentations before moving to a 3D simulator for high fidelity simulation.

Environments that are most similar to Scenario Gym are DriverGym [7], higway-env [8], SMARTS [18] and FLOW [9]. While these environments facilitate deep reinforcement learning using traffic assumptions and limited intelligent agents, Table I shows the advantages of Scenario Gym for a holistic solution to training, testing and validating on the full diversity of scenario data with a highly flexible open environment.

While not focused on AV operations, FLOW [9] is a computational framework to combine RL environments and traffic simulators such as SUMO [19]. Scenarios are defined by the road network upon which traffic is simulated. By using traffic simulators this allows a reactive microscopic simulation of traffic agents. The main difference with Scenario Gym is that in FLOW scenarios are created only using traffic simulators on a given road network. Scenario Gym supports scenario data from a variety of sources, including real world collected data which can then be then interrogated in memory, replayed in simulation with intelligent agents and understood through rich metrics.

DriverGym [7] is an environment designed for training AV agents with deep reinforcement learning following the Open AI gym framework [15]. It allows some customised metrics and the possibility to simulate with reactive agents controlled by neural network policies [20], but only for vehicles. Other key differences to Scenario Gym are that DriverGym is designed only for use with the 1001 Hours dataset [21] and that it has a fixed observation representation (a rasterised map), which is both computationally expensive and limits possible sensor representations.

Similar but of a more limited scope is HighwayEnv [8], another gym-style environment for training models with RL. Scenarios are defined programatically with a limited set included. Each scenario is treated as an individual learning environment.

SMARTS [18] is designed for multi-agent learning and also uses the Open AI gym framework. Reactive agents parameterised by neural networks can be used via the social agent zoo. When outside of certain zones on the road network agents are controlled by the SUMO traffic simulator. A selection of relatively simple scenarios can be defined with a domain specific language. While HighwayEnv and SMARTS both allow learning, their data input interfaces lack important elements, like support for pedestrians. Furthermore, they cannot be scaled to large scenario sets from diverse data sources because every scenario must be defined programmatically in a domain-specific language.

For a comprehensive survey of publicly available datasets for AV development we refer the reader to [1]. Some of these datasets can be accessed via APIs that provide similar functionality to Scenario Gym, for example, 1001 Hours [21], nuScenes [17] and Argoverse [16]. However, implementing functionality in a data-source-agnostic framework like Scenario Gym has the advantage that code for custom metrics and intelligent agents can be used with scenario data from different sources.

III. SCENARIO GYM

A. Scenario representation

Each scenario is represented as a set of entities and a road network. Entities have a catalog entries, containing information such as the type of vehicle and its bounding box, and a trajectory. An entity with a reactive agent can use the trajectory to represent target positions, or not at all. The scenario may also include a set of actions for example route directions, traffic light commands or triggers for hazard entities.

The road network is represented as a collection of geometries such as roads and lanes and their connectivity information as well as any other road objects. By default the road network includes roads and intersections which are made up of lanes as well as pavements, pedestrian crossings and buildings. Each lane contains references to its successor and predecessor lanes which allows routes to be found across the network. Scenario Gym also supports the definition of custom road objects and geometries or the additional road network information e.g. traffic elements such as traffic lights, supplying speed limits of different roads or restrictions for the vehicles that can use each lane.

Scenario Gym's atomic scenario representation makes managing large datasets of scenarios easy and simplifies processes such as domain randomisation, wherein scenarios can be cloned and modified by varying the timing, speeds, numerosities of entities, further augmenting the dataset.

Both the scenario and road network representations are compatible with the OpenSCENARIO and OpenDRIVE standards respectively meaning any scenarios stored in these formats can be used in Scenario Gym with minimal setup time.

B. Intelligent agents

Simulating scenarios with reactive intelligent agents is a core feature. An agent can be defined to control any entity by taking observations from the global state, selecting actions and having those actions converted into future poses via physical models. To streamline the process of designing and implementing these agents Scenario Gym splits the process into these three steps as different submodules: the sensor, agent and controller.

The sensor module produces the observation at any point in time from the global state. It is responsible for passing the information that would be available to the agent at the current time in the correct format. This module can be used to model the sensory information available to AVs on the road (e.g. by providing the output of an object detection system) or to provide information in a simpler way to aid learning (e.g. a rasterised map around the entity rendered as an image).

The observation provided by the sensor is then passed to the agent which selects an action. The action can take a variety of formats e.g. vehicle controls such as steering and acceleration, specific target poses or pedestrian commands

	SMARTS	highway-env	DriverGym	Flow	Scenario Gym
Input flexibility and dataset extensibility	×	×	×	×	✓
In-memory scenario representation	×	×	×	×	✓
Customisable vehicle intelligence	✓	✓	✓	✓	✓
Customisable non-vehicle intelligence	×	×	×	×	✓
Customisable physical model	×	×	×	×	✓
Safety and evaluation metric extensibility	×	×	✓	×	✓

TABLE I: Comparison of similar environments to Scenario Gym.

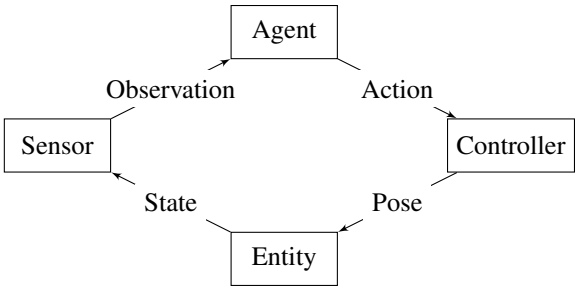


Fig. 2: The Scenario Gym agent framework.

such as walk or stand still. Here the agent's intelligence can be configured by the user. For example, the agent could use a mathematical model such as the Intelligent Driver Model [22] or a trained neural network to produce actions.

Once the action is selected it is passed to the controller module. This is the physical model that controls the agent's dynamics. This allows modelling different entities with varying levels of detail. For example, vehicle agents can be controlled with detailed physical models while pedestrians could use a much more simple controller. The controller takes the action and outputs a new pose for the agent.

These three steps are executed for each agent at each timestep (Figure 2). By splitting the process like this we control the assumptions built into our agent models. For example through the sensor we have control over exactly what information is available to the agent at each point in time. By modularising the process we are also able to reuse common modules. For example, we can share a vehicle controller and sensor across agents but use different decision making systems.

This architecture allows easy and precise control over the simulation. For example, different entities may follow predefined trajectories, obey trigger actions, be controlled by intelligent traffic agents or data driven models and use a range of physical models.

In the base package we include implementations of several agent modules. For vehicles these include a rasterised map sensor to provide semantic information around the agent and controllers such as a physical vehicle model and a PID controller. For pedestrians we provide implementations of a random walk model and a social force model [23].

C. Metrics

Customised metrics are called every timestep to update their internal state. The package includes pre-implemented metrics to measure the speeds and distances of the ego vehicle, a collision classifier (e.g. as head on, rear end, etc) as well as more complex metrics such as the Responsibility-Sensitive Safety metric [13]. These can easily be attached to scenarios to obtain detailed insight in seconds (see section IV-A).

D. Simulation

When scenarios have been loaded and agents and metrics defined Scenario Gym simulates the scenario synchronously through discrete time steps where the submodules for each agent are called to produce the new pose for each entity in the scenario. After the new pose has been calculated for each entity the next timestep can begin. The metrics are called after each step to update their internal state. The scenario ends either when its total length (derived from the trajectories of its entities) has elapsed or under other conditions defined by the user (e.g. collisions between entities or goals achieved by the ego).

A simulated scenario can be rendered as a video with a range of options available to control the layers and colouring used e.g. entities by can be coloured by a measure of risk to the ego. Scenario Gym can also output the resulting simulation as a new scenario.

Since Scenario Gym is lightweight it can be trivially parallelised to run over as many CPUs as are available which can significantly speed-up computation.

IV. EXPERIMENTS

In this section we show how Scenario Gym can be used to accomplish a variety of tasks along the AV development process: understanding scenario data, intelligent agent modelling and learning from scenario data with deep reinforcement learning.

A. Interrogating scenario data

Scenario Gym aims to make working with scenario data easy and allow users to quickly access the features that matter. We explore this by simulating two large publicly available AV datasets and feeding the outputs into informative metrics. Specifically we take the motion forecasting sections of the Argoverse [16] and nuScenes [17] datasets and simulate each scenario using metrics to record specific features of each. For nuScenes we simulate each sample using a different tracked entity as the ego vehicle. Scenario Gym simulates around 43 scenarios per second at around 400x realtime. The metrics identify particular manoeuvres by the ego vehicle (e.g. left and right turns, going straight, etc.), potential hazardous entities (e.g. pedestrians in the

road, nearby cyclists) and whether the ego is at an unsafe distance according to RSS [13].

We compare the feature distributions over the training and validation sets in Figure 3 between the two datasets. We find that a large proportion of the scenarios involve either the ego travelling straight or stationary in traffic, typical of urban driving. A small proportion of scenarios are detected as having hazardous pedestrians or cyclists which typically comes from pedestrians crossing the road or cyclists changing lanes. We believe Scenario Gym demonstrates value not only by making datasets like these easily tractable, but by providing a quick way to observe the bias in the input data. For example, Argoverse appears to be biased towards easy driving (i.e. the overwhelming prevalence of straight driving) due to its scenarios being uniformly sampled from many hours of driving while nuScenes is manually curated to oversample certain features such as the presence of cyclists. Visibility over statistical biases such as these will be crucial for training AVs with methods such as deep reinforcement learning.

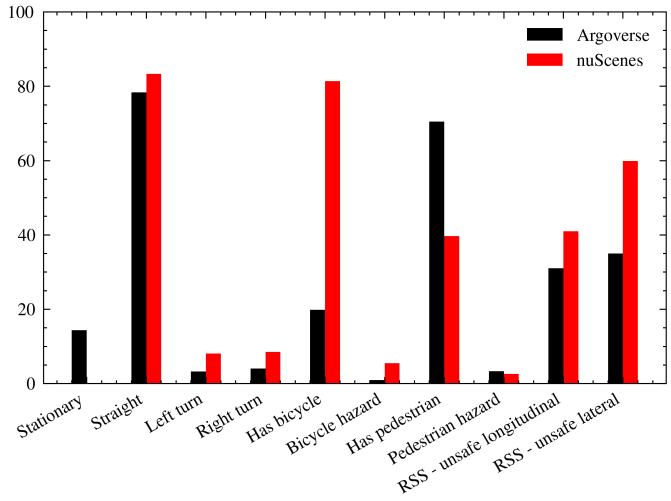


Fig. 3: Proportion of scenarios in the Argoverse and nuScenes datasets with each feature.

B. Intelligent agent modelling

We demonstrate how Scenario Gym can be used to simulate the behaviour of intelligent agents to which an AV may need to react. We implement a Social Force Model [23] to control the dynamics of pedestrians in simulation. This models the interactions of a pedestrian with its environment as a collection of attractive and repulsive forces. For example, its proximity to the goal point, the boundaries of nearby roads and buildings and any nearby pedestrians. The net force is then used to determine the pedestrian's acceleration and direction. In our model each pedestrian receives an attractive force from the route to its goal point and to nearby pedestrians moving in the same direction. The pedestrian receives a repulsion force to the boundary of buildings, roads and other entities as well as a repulsion to other pedestrians within a certain distance. We also model the angle of sight of each pedestrian which affects the strength of forces from other pedestrians if they are in view.

We implement this model in the Scenario Gym framework. A sensor provides the pedestrian's observation which includes the pedestrians that are nearby and in view as well as relevant road network features. Then the social force model produces an action in the form of a desired speed and heading by computing the attractive and repulsive forces. Finally a controller applies the physical model for the pedestrian and determines the new pose from the action subject to constraints on speed and acceleration.

To show that the implemented agent can effectively model pedestrian dynamics we simulate the motion of a crowd walking along a road (Figure 4 (a-d)). A number of pedestrians are initialised on the pavement alongside the start of a straight road. They aim to pass along the pavement to the end of the road however halfway along a building restricts the width of the pavement. We simulate the scenario with a social force model controlling each pedestrian. The pedestrians initially spread out and spill into the road due to their mutual repulsive forces overcoming the repulsive force from the road boundary. Then they move towards the goal before a bottleneck appears at the boundary of the building which causes the crowd to spread out along the path. Pedestrians rejoin the pavement where the crowd thins as they move towards the goal.

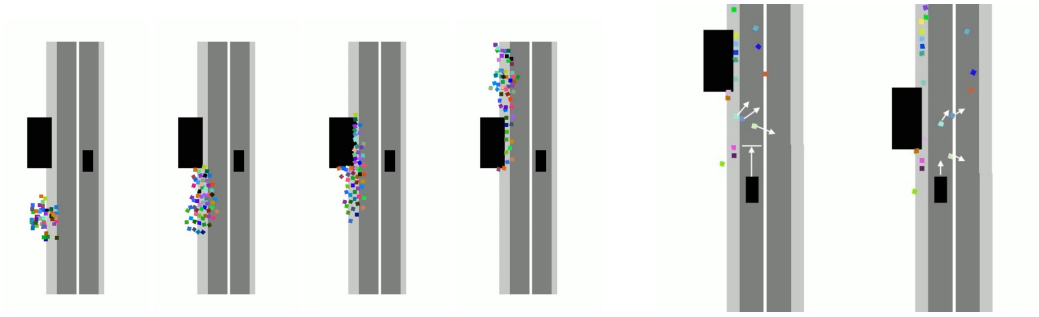
C. Learning from scenario data

A key application for a framework such as Scenario Gym is training AV systems with deep reinforcement learning (Section II). Scenario Gym is fully compatible with the increasingly popular OpenAI Gym [15] framework, meaning Scenario Gym can be used with many reinforcement learning libraries such as TensorFlow Agents [24] and Stable Baselines3 [25].

We show how Scenario Gym can be used to train agents in reactive simulation with deep reinforcement learning. We sample scenarios similar to the above with an added complexity: we add a random number of the pedestrians that will cross the road during the scenario. We then place the ego vehicle at the start of the lane passing alongside the crowd and train it to safely travel along the lane (Figure 4 (e-f)).

The agent controls its speed by selecting between discrete acceleration and braking levels which are be decoded via a kinematic model to produce an updated pose. The agent's sensor module detects the top n pedestrians that are nearby and returns their velocity and relative position to the ego. The top n are selected by computing the time taken at their current velocity to intersect the ego's path and selecting the n smallest. The reward signal is given by the speed of the vehicle with a multiplicative constant which varies depending on whether the ego has collided with a pedestrian: $r = \alpha v$, where $\alpha = \alpha_- < 0$ if collision, and $\alpha_+ > 0$ otherwise.

In our experiments with set $\alpha_- = -10.$ and $\alpha_+ = 0.1$. We train our agent with Proximal Policy Optimisation [26] for 500k timesteps with a learning rate of 0.0004, a discount factor of 0.99 and an advantage factor of 0.9. The model is



(a) Starting positions (b) Crowd spills into road (c) Bottleneck forms (d) Crowd thins as pedestrians pass through bottleneck (e) Agent brakes as pedestrians cross the road. (f) Agent accelerates as road begins to clear.

Fig. 4: (a-d) Crowd dynamics simulated with a Social Force Model. (e-f) Ego agent trained to navigate through the crowd. Pedestrians are shown as small coloured squares. The large and small black rectangles are a building and a vehicle respectively.

a multi-layer perceptron which outputs action probabilities and value estimates. Figure 5 shows the reward throughout training.

To evaluate the agent we make use of some of the metrics available in Scenario Gym. We track the mean and maximum speeds achieved by the ego as well as its distance travelled and the number of collisions that occur in each scenario. Over the course of training we find a 49.5% reduction in the number of collisions. We also perform a qualitative comparison of the learned policies via rendered videos of the agent's performance. Halfway through training the agent has learned a policy of heavy initial acceleration to gain reward and then coming to a full stop when nearing pedestrians to avoid collisions. This is successful initially however the agent is overly cautious and too slow to react when the path is clear. By the end of training the agent has learned to control its speed more effectively. It moves forward and slows when a pedestrian gets near or starts to move into the road but then accelerates when the lane becomes clear again. The agent is still sometimes slow to react suggesting that with further training or hyperparameter tuning its performance could be improved.

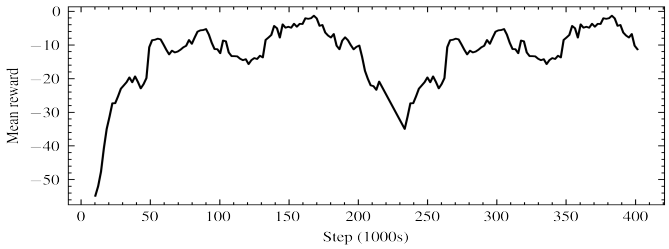


Fig. 5: Mean reward throughout training.

V. CONCLUSION

We introduced Scenario Gym as a framework that allows users to efficiently understand and work with scenario data. We outlined its scenario representation and intelligent agent framework. Through our experiments we demonstrated the ability of Scenario Gym to facilitate a variety of relevant tasks for AV development.

REFERENCES

[1] P. Ji, L. Ruan, Y. Xue, L. Xiao, and Q. Dong, "Perspective, survey and trends: Public driving datasets and toolsets for autonomous driving virtual test," 2021. [Online]. Available: https://arxiv.org/abs/2104.00273

[2] P. Kaur, S. Taghavi, Z. Tian, and W. Shi, "A survey on simulators for testing self-driving cars," 01 2021.

[3] T. Dawkins, "Safe drive initiative: Safedi scenario-based av policy framework – technical implementation guidance." [Online]. Available: https://www3.weforum.org/docs/WEF_Safe_DI_AV_policy_framework_2020.pdf

[4] ASAM, "Asam openscenario," accessed: 2022-08-31. [Online]. Available: \https://www.asam.net/standards/detail/openscenario/

[5] ——, "Asam opendrive," accessed: 2022-08-31. [Online]. Available: https://www.asam.net/standards/detail/opendrive/

[6] A. Mumuni and F. Mumuni, "Data augmentation: A comprehensive survey of modern approaches," *Array*, vol. 16, p. 100258, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590005622000911

[7] P. Kothari, C. Perone, L. Bergamini, A. Alahi, and P. Ondruska, "Drivergym: Democratising reinforcement learning for autonomous driving," 2021. [Online]. Available: https://arxiv.org/abs/2111.06889

[8] E. Leurent, "An environment for autonomous driving decision-making," https://github.com/eleurent/highway-env, 2018.

[9] C. Wu, A. R. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, "Flow: A modular learning framework for mixed autonomy traffic," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 1270–1286, apr 2022. [Online]. Available: https://doi.org/10.1109\%2Ftro.2021.3087314

[10] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[11] kostya, "benchmarks," accessed: 2022-08-31. [Online]. Available: https://github.com/kostya/benchmarks

[12] E. Adetiba, T. John, A. Akinrinmade, F. Moninuola, O. Akintade, and J. Badejo, "Evolution of artificial intelligence languages, a systematic literature review," 2021. [Online]. Available: https://arxiv.org/abs/2101.11501

[13] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," 2017. [Online]. Available: https://arxiv.org/abs/1708.06374

[14] "Mit license." [Online]. Available: https://opensource.org/licenses/MIT

[15] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016. [Online]. Available: https://arxiv.org/abs/1606.01540

[16] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[17] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," 2019. [Online]. Available: https://arxiv.org/abs/1903.11027

[18] M. Zhou, J. Luo, J. Villella, Y. Yang, D. Rusu, J. Miao, W. Zhang, M. Alban, I. Fadakar, Z. Chen, A. C. Huang, Y. Wen, K. Hassanzadeh, D. Graves, D. Chen, Z. Zhu, N. Nguyen, M. Elsayed, K. Shao, S. Ahilan, B. Zhang, J. Wu, Z. Fu, K. Rezaee, P. Yadmellat, M. Rohani, N. P. Nieves, Y. Ni, S. Banijamali, A. C. Rivers, Z. Tian, D. Palenicek, H. b. Ammar, H. Zhang, W. Liu, J. Hao, and J. Wang, "Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving," 2020. [Online]. Available: https://arxiv.org/abs/2010.09776

[19] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker-Walz, "Recent development and applications of sumo - simulation of urban mobility," *International Journal On Advances in Systems and Measurements*, vol. 3&4, 12 2012.

[20] L. Bergamini, Y. Ye, O. Scheel, L. Chen, C. Hu, L. Del Pero, B. Osinski, H. Grimmett, and P. Ondruska, "Simnet: Learning reactive self-driving simulations from real-world observations," 2021. [Online]. Available: https://arxiv.org/abs/2105.12332

[21] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, L. Chen, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, "One thousand and one hours: Self-driving motion prediction dataset," 2020. [Online]. Available: https://arxiv.org/abs/2006.14480

[22] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, pp. 1805–1824, aug 2000. [Online]. Available: https://doi.org/10.1103\%2Fphysreve.62.1805

[23] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, no. 5, pp. 4282–4286, may 1995. [Online]. Available: https://doi.org/10.1103\%2Fphysreve.51.4282

[24] S. Guadarrama, A. Korattikara, O. Ramirez, P. Castro, E. Holly, S. Fishman, K. Wang, E. Gonina, N. Wu, E. Kokiopoulou, L. Sbaiz, J. Smith, G. Bartók, J. Berent, C. Harris, V. Vanhoucke, and E. Brevdo, "TF-Agents: A library for reinforcement learning in tensorflow," https://github.com/tensorflow/agents, 2018, [Online; accessed 25-June-2019]. [Online]. Available: https://github.com/tensorflow/agents

[25] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347