

GPUDrive: Data-driven, multi-agent driving simulation at 1 million FPS

Saman Kazemkhani^{*†}
New York University

Aarav Pandya^{*†}
New York University

Daphne Cornelisse^{*†}
New York University

Brennan Shacklett
Stanford University

Eugene Vinitzky
New York University

Abstract

Multi-agent learning algorithms have been successful at generating superhuman planning in a wide variety of games but have had little impact on the design of deployed multi-agent planners. A key bottleneck in applying these techniques to multi-agent planning is that they require billions of steps of experience. To enable the study of multi-agent planning at this scale, we present GPUDrive, a GPU-accelerated, multi-agent simulator built on top of the Madrona Game Engine that can generate over a million steps of experience per second. Observation, reward, and dynamics functions are written directly in C++, allowing users to define complex, heterogeneous agent behaviors that are lowered to high-performance CUDA. We show that using GPUDrive we are able to effectively train reinforcement learning agents over many scenes in the Waymo Motion dataset, yielding highly effective goal-reaching agents in minutes for individual scenes and generally capable agents in a few hours. We ship these trained agents as part of the code base at <https://github.com/Emerge-Lab/gpudrive>.

1 Introduction

Multi-agent learning has been impactful across a wide range of fully cooperative and zero-sum games [1, 2, 3, 4, 5, 6]. However, its impact on multi-agent planning for settings that mix humans and robots has been muted. In contrast to the ubiquity of multi-agent learning-based agents in zero-sum games, multi-agent planners for most practical robotic systems are not derived from the output of game-theoretically sound learning algorithms. While it is hard to characterize the space of deployed planners since many of them are proprietary, the majority likely use a mixture of collected data for the prediction of human motion and hand-tuned costs. These are then fed into a robust trajectory optimizer or may be based on imitation learning [7, 8]. This approach has been highly effective in scaling up real-world autonomy but can struggle with reasoning about long-term behavior, contingency planning, and interaction with humans in rare, complex scenarios.

The divergence in preferred technique between these two domains is partially the outcome of two distinct, challenging components of real-world multi-agent planning. First, unlike zero-sum games, it is necessary to play a human-compatible strategy that is difficult to identify without data. Second, generating the billions of samples needed for multi-agent learning algorithms is difficult with existing simulators. The former challenge is difficult for multi-agent learning since there is not a clear equilibrium concept that algorithms should be pursuing. The latter problem is a challenge for simulators since it is difficult to simulate embodied multi-agent environments at appropriately high rates.

^{*}These authors contributed equally to this work

[†]Corresponding authors: skazemkhani@gmail.com, pandya.aarav.97@gmail.com, cornelisse.daphne@nyu.edu.

To address these challenges and unlock multi-agent learning as a tool for generating capable self-driving planners, we introduce GPUDrive. GPUDrive is a simulator intended to mix real-world driving data with simulation speeds that enable the application of sample-inefficient but effective RL algorithms to planner design. GPUDrive runs at over a million steps per second on both consumer-grade and datacenter-class GPUs and has a sufficiently light memory footprint to support hundreds to thousands of simultaneous worlds (environments) with hundreds of agents per world. GPUDrive supports the simulation of a variety of sensor modalities, from LIDAR to a human-like view cone, enabling GPUDrive to be used for studying the effects of different sensor types on resultant agent characteristics. Finally, GPUDrive takes in driving logs and maps from existing self-driving datasets, enabling the mixing of tools from imitation learning with reinforcement learning algorithms. This enables the study of both the development of autonomous vehicles and the learning of models of human driving, cycling, and walking behavior.

Our contributions are:

- We provide a multi-agent, GPU-accelerated, and data-driven simulator that runs at over a million steps per second. Our simulator provides provides a testbed for:
 1. Investigating the capability of learning algorithms to solve challenges related to self-play or autonomous coordination.
 2. Researching the effects of limited or human-like perception on agent behavior.
- We provide gym environments in both `torch` and `jax` that can be easily configured with standard open-source multi-agent RL and imitation learning libraries. Additionally, we have open-sourced a basic policy-gradient training loop that can be readily used to develop agents.
- We release implementations of tuned RL algorithms that can process 20 million steps of experience per hour on consumer-grade GPUs. These can be used to train 95% goal-reaching agents across 100 different scenes in two hours on relatively accessible hardware.
- We open-source strong driving baseline agents that achieve 97% of their goals on a subset of scenes they have been trained to solve. These are integrated into the simulator so that the simulator comes with default, capable, reactive agents.

2 Related work

Frameworks for batched simulators. There are various open-source frameworks available that support hardware-accelerated reinforcement learning environments. These environments are generally written directly in an acceleration framework such as Numpy [9], Jax [10], or Pytorch [11]. In terms of multi-agent accelerated environments, standard benchmarks include JaxMARL [12], Jumanji [13], and VMAS [14] which primarily feature fully cooperative or fully competitive tasks. Each benchmark requires the design of custom accelerated structures per environment. In contrast, GPUDrive focuses on a mixed motive setting and is built atop Madrona, an extensible ECS-based framework in C++, enabling GPU acceleration and parallelization across environments [15]. Madrona comes with vectorization of key components of embodied simulation such as collision checking and sensors such as LIDAR. GPUDrive can support hundreds of controllable agents in more than 100,000 distinct scenarios, offering a distinct generalization challenge and scale relative to existing benchmarks. Moreover, GPUDrive includes a large dataset of human demonstrations, enabling imitation learning, inverse RL, and combined IL-RL approaches.

Simulators for autonomous driving research and development. Table 1 shows an overview of current simulators used in autonomous driving research. The purpose of GPUDrive is to facilitate the systematic study of behavioral, coordination, and control aspects of autonomous driving and multi-agent learning more broadly. As such, visual complexity is reduced, which differs from several existing simulators, which (partially) focus on perception challenges in driving [16, 17]. Driving simulators close to GPUDrive in terms of either features or speed include MetaDrive [18], nuPlan [19], Nocturne [20], and Waymax [21] which all utilize real-world data. Unlike MetaDrive and nuPlan, our simulator is GPU-accelerated. Like GPUDrive, Waymax is a JAX-based GPU-accelerated simulator that achieves high throughput through JIT compilation and efficient use of accelerators. With respect to Waymax, our simulator supports a wider range of possible sensor modalities (Section

3.2) including LIDAR and human-like views, and can scale to nearly thirty times more worlds (Section 4.1, and comes with performant reinforcement learning baselines.

Driving agents in simulators and algorithms. Existing simulators often feature baseline agents for interaction, such as *low-dimensional car following models* that describe vehicle dynamics through a limited set of variables or parameters [22, 23, 24]. Rule-based agents exhibit predetermined behaviors, like car-following agents [21, 19, 25, 26] such as the IDM model, or parameterized behavior agents like CARLA’s TrafficManager [16]. Some simulators offer *recorded human driving logs* for interaction through replaying the human driving logs [8, 20, 21, 19]. Additionally, certain simulators provide *learning-based agents*, leveraging reinforcement learning techniques [18]. In our simulator, we provide both human driving logs and high-performing reinforcement learning agents.

Table 1: Comparison of GPUDrive to related driving simulators. Columns represent whether the simulator supports multi-agent simulation, GPU acceleration, simulation of sensors such as LIDAR or human views, is built atop data, comes with existing driver models, and whether the agents are provided explicit goal points or waypoints along the way to the goal.

Simulator	Multi-agent	GPU-Accel	Sensor Sim	Expert Data	Sim-agents	Routes / Goals
TORCS [27]			✓		✓	-
GTA V [28]			✓			-
CARLA [16]			✓		✓	Waypoints
Highway-env [29]						-
Sim4CV [30]			✓			Directions
SUMMIT [17]	✓ (≥ 400)		✓		✓	-
MACAD [31]	✓		✓		✓	Goal point
SMARTS [32]	✓					Waypoints
MADRaS [33]	✓ (≥ 10)		✓			Goal point
DriverGym [34]				✓	✓	-
VISTA [35]	✓		✓	✓		-
nuPlan [19]			✓	✓	✓	Waypoints
Nocturne [20]	✓ (≥ 128)			✓	✓	Goal point
MetaDrive [18]	✓		✓	✓	✓	-
InterSim [36]	✓			✓	✓	Goal point
TorchDriveSim [37]	✓	✓			✓	-
BITS [38]	✓			✓	✓	Goal point
Waymax [21]	✓ (≥ 128)	✓		✓	✓	Waypoints
GPU Drive (ours)	✓ (≥ 128)	✓	✓	✓	✓	Goal point

3 Simulation Design

3.1 Simulation Engine

Learning to safely navigate complex scenarios in a multi-agent setting requires generating many billions of environment samples. To feed sample-hungry learning algorithms, GPUDrive is built on top of Madrona [15], an Entity-Component-State system designed for high-throughput reinforcement learning environments. In the Madrona framework, multiple independent worlds (each containing an independent number of agents[†]) are executed in parallel on accelerators via a shared engine.

However, driving simulation offers a particular set of challenges that require several technical choices. First, road objects, such as road edges and lane lines are frequently represented as polylines (i.e. connected sets of points). These polylines can consist of hundreds of points as they are sampled at every 0.1 meters, leading to even small maps having upwards of tens of thousands of points. This can blow up the memory requirements of each world as well as lead to significant redundancy in agent observations. Second, the large numbers of agents and road objects can make collision checking a throughput bottleneck. Finally, there is immense variability in the number of agents and road objects in a particular scene. Each world allocates memory to data structures that track its state and accelerate simulation code. Though independent, each world incurs a memory footprint proportional to the *maximum* number of agents across all worlds. In this way, the performance of GPUDrive is sensitive to the variation in agent counts across all the worlds in a batch.

[†]In the Waymo Open Motion Dataset, an agent constitutes a vehicle, cyclist, or pedestrian.

These challenges are partially resolved via the following mechanisms. First, a primary acceleration data structure leveraged by GPUDrive is a Bounding Volume Hierarchy (BVH). The BVH keeps track of all physics entities and is used to easily exclude candidate pairs for collisions. This allows us to then run a reduced-size collision check on potential collision candidate pairs. The use of a BVH avoids invoking a collision check that would otherwise always be quadratic in the number of agents in a world. Secondly, we observed that a lot of the lines in the geometry of the roads are straight. This allows us to omit many intermediate points while only suffering a minor hit in the quality of the curves. We apply a polyline decimation algorithm (Viswalingham-Whyatt Algorithm) [39] to approximate straight lines and filter out low-importance points in the polylines. With this modification, we can reduce the number of points by 10-15 times and significantly improve the step times while decreasing memory usage.

3.2 Simulator features

We provide an overview of some of the pertinent simulator features as well as sharp edges and limitations of the simulator as a guide to potential users.

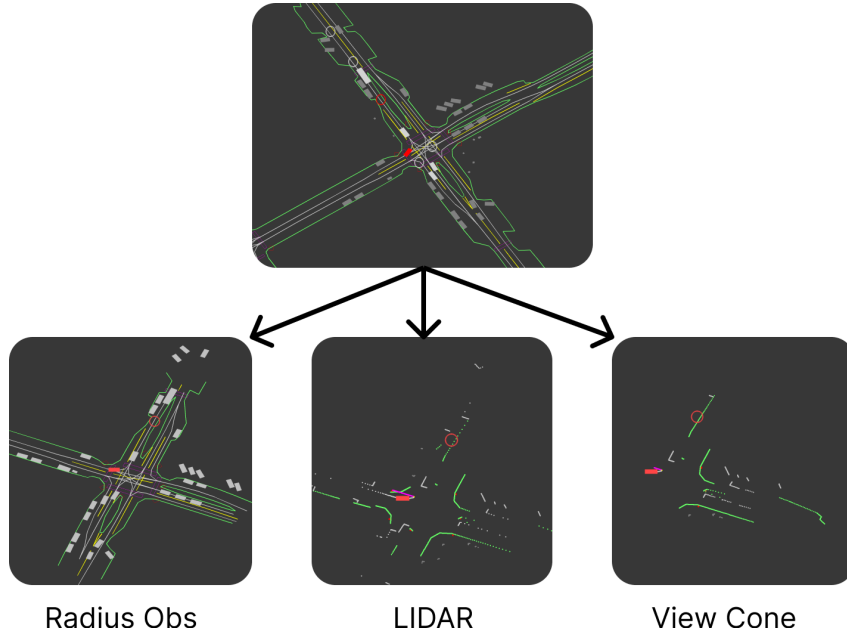


Figure 1: Visualization of different observation spaces available in GPUDrive. The top scene is an example scenario from the dataset, rendered from the ego-centric perspective of the red vehicle. Grey cars are parked cars while white cars are other controlled agents. From left to right: the Radius Obs returns all objects within 100 meters, the LIDAR observation with 3000 rays spread around 360 degrees, and a view cone consisting of 3000 rays emanating in a 120-degree view cone.

Dataset. GPUDrive represents its map as a series of polylines and does not require a connectivity map of the lanes. As such, it can be made compatible with most driving datasets given the pre-processing of the roads into the polyline format. Currently, GPUDrive supports the Waymo Open Motion Dataset (WOMD) [40] which is available under a non-commercial license. The WOMD consists of a set of over 100,000 multi-agent traffic scenarios, each of which contains the following key elements: 1) Road map - the layout and structure of a road, such as a highway or parking garage. 2) Expert human driving demonstrations. 3) Road objects, such as stop signs and crosswalks. Figure 1 depicts an example of an intersection traffic scenario as rendered in GPUDrive.

Sensor modalities. GPUDrive supports a variety of observation spaces intended to enable heterogeneous types of agents. Fig. 1 depicts the three types of supported state spaces. The first mode is somewhat unphysical in which all agents and road objects within a fixed radius are observable

to the agent. This mode is intended primarily for debugging and quick testing, enabling a user to minimize the amount of partial observability in the environment. The other two modes are based on a GPU-accelerated LIDAR scan, representing what an autonomous vehicle would be able to see and what a human would likely be able to see respectively. Both modes are based on casting LIDAR rays; to model human vision we simply restrict the LIDAR rays to emanate in a smaller, controllable-sized cone that can be rotated through an action corresponding to head rotation. Note that since all objects are represented as bounding boxes of fixed height, the LIDAR observations are over-conservative as humans while LIDAR scans in reality are usually able to see over the hoods of cars.

Agent dynamics. Agents are stepped using a standard Ackermann bicycle model with actions corresponding to steering and acceleration. This model enables the dynamics of objects to be affected by their length, creating different dynamics for small cars vs. trucks. However, this model is not fully invertible which can make it challenging to use this model for imitation learning. To enable full invertibility for imitation learning, we also support the simplified bicycle model, taken from Waymax [21], which is a double-integrator in the position and velocity and updates its yaw as:

$$\theta_{t+1} = \theta + s_t(v_t\Delta t + \frac{1}{2}a_t\Delta t^2)$$

where θ is the yaw, s is the steering command, v is the velocity, and a is the acceleration at time t respectively. Δt is the timestep. This model is always invertible given an unbounded set of steering and acceleration actions but is independent of the vehicle length. See the appendix for full details on the models.

Note that this model does not factor in the length of the car, causing both long and short objects to have identical dynamics. However, computing the expert actions and then using them to mimic the expert trajectory under this model leads to lower tracking error than the default bicycle model.

Rewards. All agents are given a target goal to reach; this goal is selected by taking the last point observed in the vehicle’s logged trajectory. A goal is reached when agents are within some configurable distance δ of the goal. By default, agents in GPUDrive receive a reward of 1 for achieving their goal and otherwise receive a reward of 0. There are additional configurable collision penalties or other rewards based on agent-vehicle distances or agent-road distances though these are not used in this work.

Available driving simulation agents. We use reinforcement learning to train a set of agents that reach their goals 95 % of the time on a subset of 500 training scenes. While this number is far below the capability of human drivers, these agents are reactive in a distinct fashion from parametrized driver models in other simulators. In particular, many logged-data simulators construct reactivity by having the driver follow along its logged trajectory but decelerate if an agent passes in front of it. In contrast, these agents can maneuver and negotiate without remaining constrained to a logged trajectory. These trained agents are extremely aggressive about reaching their goals and can be used as an out-of-distribution test for proposed driving agents. The training procedure and more details can be found in Section 4.2.

Simulator sharp-edges. We note the following limitations of the benchmark:

- *Absence of a map.* The current version of the simulator does not have a well-defined notion of lanes or a higher-level road map which makes it challenging for algorithmic approaches that require maps. The absence of this feature also makes it challenging to define rewards such as "stay lane-centered."
- *Convex objects only.* Collision checking relies on the objects being represented as convex objects.
- *Unsolvable goals.* Due to mislabelling in the Waymo dataset, some agent goals (roughly 2%) are unreachable. For these agents, we default them to simply replaying their logged trajectory and do not treat them as agents.
- *Variance in controllable agents per scenario.* In the majority of scenes, there are approximately 8-10 agents and an average of 50 parked cars. Additionally, the dataset is gathered from the sensors of an autonomous vehicle, leading to some agents having their initial states

recorded only after the first time-step of the simulator. These agents are not included, as incorporating them would necessitate "teleporting" them into the scene, potentially leading to unavoidable collisions with agents deviating from their logged trajectories.

4 Simulator performance

The following Sections describe the simulator speed. Section 4.1 first shows the raw simulator speed and peak goodput. Section 4.2 then investigates the impact on reinforcement learning workflows by evaluating the time it takes to train reinforcement learning agents through Independent PPO (IPPO) [41], a widely used multi-agent learning algorithm.

4.1 Simulation speed

Since scenarios contain a variable number of agents, we introduce a metric called Agent Steps Per Second (ASPS) to measure the sample throughput of the simulator. We define the ASPS as the total number of agents across all worlds in a batch that can be fully stepped in a second:

$$\text{ASPS} = \frac{S \times \sum_{k=1}^N |A_k|}{\Delta T} \quad (1)$$

where A_k is the number of agents in the k^{th} world, S is the number of steps taken, and ΔT is the number of seconds elapsed. Figure 2 examines the scaling of the simulator as the number of simulated worlds, which represents the amount of parallelism, increases. To measure performance, we sample random batches of scenarios of size equal to the number of worlds, so that every world is a unique scenario with K agents. On the left-hand side of Figure 2, we compare the performance of GPUDrive to the original Nocturne version [20] (CPU, no parallelism) and a CPU-accelerated version of Nocturne via Pufferlib (16 CPU cores) [42]. Empirically, the maximum achievable AFPS of Nocturne is 15,000 (blue dotted line) though we caution that additional speedups may be possible. In contrast, GPUDrive can reach over a million ASPS on a consumer-grade GPU at 512 worlds (average agents per scenario is 60). This performance also surpasses that of Waymax [21], a JAX-based simulator, where we could not run more than 32 environments in parallel due to Out of Memory (OOM) issues. Note that GPUDrive exhibits near-linear scaling of ASPS between 32 and 128 worlds on a datacenter-grade NVIDIA GPU and between 32 and 256 worlds on a consumer-grade GPU.

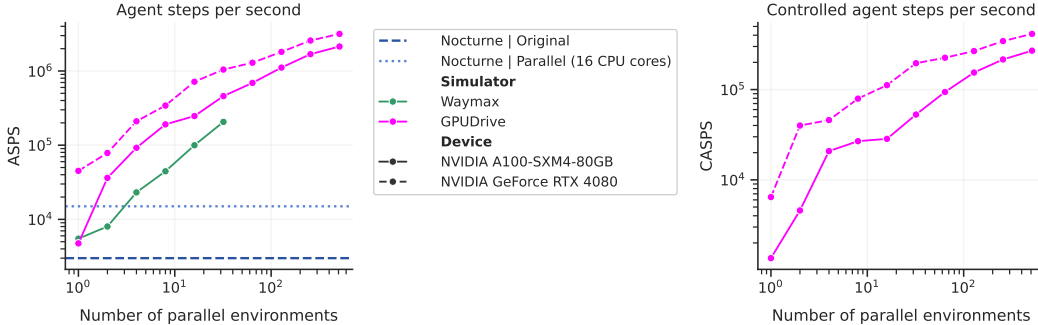


Figure 2: **Peak goodput of GPUDrive on a consumer-grade and datacenter-class GPU compared to original, CPU-based, implementations.** *Left:* The total number of controllable agent steps per second (CASPS) as we increase the number of worlds (parallelism). *Right:* The total number of agent steps per second (ASPS) is the number of objects that our system computes observations for at each time step.

To ensure a fair comparison, Figure 2 (LHS) resembles the conditions used in [21], where all cars, bicyclists, and pedestrians are considered as valid experience-generating agents. However, by default, our system only considers something an agent if it is necessary to move to achieve the goal. This means that cars persistently parked throughout the episode are not considered agents, which aligns correctly with what should constitute an agent. This significantly alters the characteristics of the agent

distribution, leading to the right-hand side of Figure 2, where we plot the Controlled Agent Steps Per Second (CASPS). Under this definition of agents, CASPS represents the expected performance of our system when using the Waymo Open Motion Dataset (this mostly excludes parked vehicles).

4.2 End-to-end speed and performance

The purpose of GPUDrive is to facilitate research and development in multi-agent algorithms by 1) reducing the completion time of experiments, and 2) enabling academic research labs to achieve scale on a limited computing budget. Ultimately, we are interested in the rate at which a machine learning researcher or practitioner can iterate on ideas using GPUDrive. This Section highlights what our simulator enables in this regard by studying the end-to-end process of learning policies in our simulator.

Figure 3 contrasts the number of steps (experience) and the corresponding time required to *solve* 10 scenarios from the WOMD between Nocturne and GPUDrive. For benchmarking purposes, we say a scene is solved when 95% of agents across all 10 worlds can navigate to their designated target position without colliding or going off-road across all worlds. Ceteris paribus (Details in Appendix D), GPUDrive achieves a 25-40x training speedup, solving 10 scenarios in less than 15 minutes compared to approximately 10 hours in Nocturne.

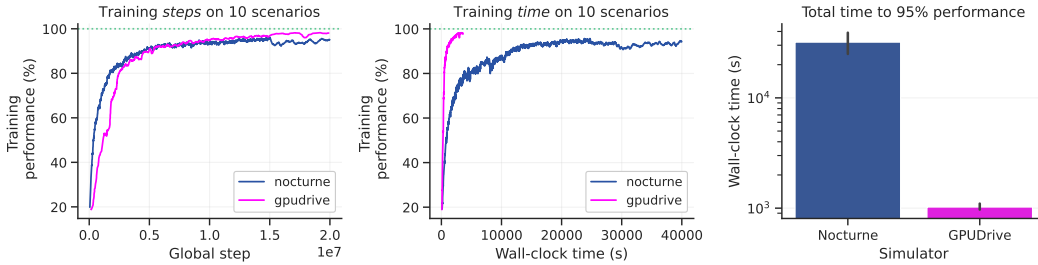


Figure 3: **From hours to minutes.** *Left:* Training performance (goal-reaching rate) as a function of the global step (AFPS). *Center:* Training performance as a function of wall-clock time. *Right:* Comparison of the total time to solve the same 10 scenarios while replicating environmental and experimental conditions as closely as possible. Runs are averaged across three seeds, see Appendix D for the hyperparameters and training details. The green dotted line indicates optimal performance.

As shown above, GPUDrive allows us to solve scenes in minutes. Next, we investigate how the *individual scene completion time*, the time it takes to solve a single scenario, changes as we increase the total number of scenarios we train in. In practice, it may be desirable to train agents on thousands of scenarios. Therefore, we ask whether it is feasible to fully leverage the simulator’s capabilities with a single GPU.

Interestingly, we find that the amortized sample efficiency increases with the size dataset of scenes we train in. Figure 4 shows the average completion time per scenario as we increase the dataset. For instance, using IPPO with 32 scenarios takes 2 minutes per scenario. In contrast, solving 1024 unique scenarios takes about 200 minutes, which amounts to only 15 seconds per scenario. We expect that these scaling benefits will continue as we further increase the size of the training dataset. This suggests that GPUDrive should enable effective utilization of the large WOMD dataset comprising 100,000 diverse traffic scenarios, even with limited computational resources.

5 Conclusion

In this work, we present GPUDrive, a GPU-accelerated, data-driven simulator. GPUDrive is intended to help generate the billions of samples that are likely needed to achieve effective reinforcement learning for training multi-agent driving planners. By building atop the Madrona Engine [15], we can scale GPUDrive to hundreds of worlds containing potentially hundreds of agents leading to throughput of millions of steps per second. This throughput occurs while synthesizing complex observations such as LIDAR. We show that this throughput has consequent implications for training reinforcement learning agents, leading to the ability to train agents to solve any particular scene in

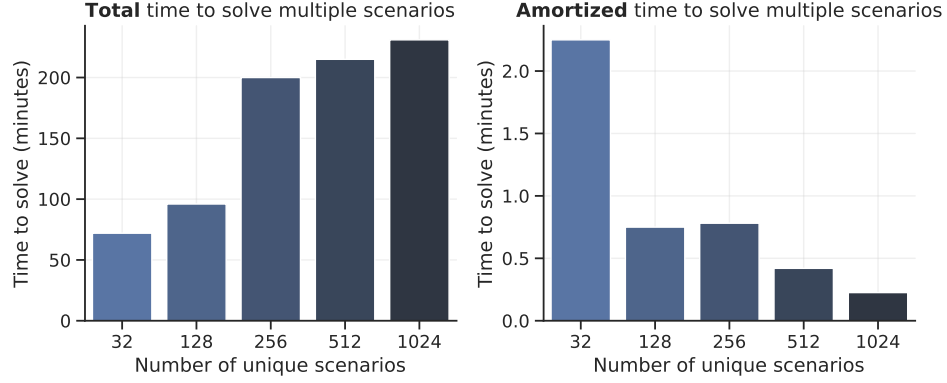


Figure 4: **Scale reduces individual scene completion time.** *Left:* Total time required to solve a fixed number of scenarios to a goal-reaching rate of 95%. Note that time-to-completion is sub-linear concerning the number of scenes. *Right:* Each additional scenario costs less to solve than the previous scenario. At 1024 scenes, the per-scene cost of solving an additional scene is on the order of 15 seconds.

minutes and in seconds when amortized across many scenes. We release the simulator and integrated trained agents to enable further research.

This paper is a first step in scaling up reinforcement learning for multi-agent planning in safety-critical, mixed human-autonomous settings. However, several important challenges remain for future work. Firstly, we have not yet identified the optimal hyperparameter settings to effectively utilize the collected data, resulting in training being the bottleneck instead of data collection time. Secondly, while the simulator is fast, collecting data for reinforcement learning leads to frequent reset calls that significantly impact throughput. Lastly, training fully human-level drivers in the simulator to navigate without crashing in any scenario remains an ongoing challenge.

Acknowledgments and Disclosure of Funding

This work is funded by the C2SMARTER Center through a grant from the U.S. DOT’s University Transportation Center Program. The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. The U.S. Government assumes no liability for the contents or use thereof. This work was also supported in part through the NYU IT High-Performance Computing resources, services, and staff expertise.

References

- [1] Brandon Cui et al. “Adversarial Diversity in Hanabi”. In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. 2023. URL: https://openreview.net/pdf?id=uLE3WF3-H_5.
- [2] Peter R. Wurman et al. “Outracing champion Gran Turismo drivers with deep reinforcement learning”. In: *Nat.* 602.7896 (2022), pp. 223–228. DOI: [10.1038/S41586-021-04357-7](https://doi.org/10.1038/S41586-021-04357-7). URL: <https://doi.org/10.1038/s41586-021-04357-7>.
- [3] Julien Pérolat et al. “Mastering the Game of Stratego with Model-Free Multiagent Reinforcement Learning”. In: *CoRR* abs/2206.15378 (2022). DOI: [10.48550/ARXIV.2206.15378](https://doi.org/10.48550/ARXIV.2206.15378). arXiv: [2206.15378](https://arxiv.org/abs/2206.15378). URL: <https://doi.org/10.48550/arXiv.2206.15378>.
- [4] David Silver et al. “Mastering the game of Go without human knowledge”. In: *Nat.* 550.7676 (2017), pp. 354–359. DOI: [10.1038/NATURE24270](https://doi.org/10.1038/NATURE24270). URL: <https://doi.org/10.1038/nature24270>.
- [5] Max Jaderberg et al. “Human-level performance in first-person multiplayer games with population-based deep reinforcement learning”. In: *CoRR* abs/1807.01281 (2018). arXiv: [1807.01281](https://arxiv.org/abs/1807.01281). URL: <http://arxiv.org/abs/1807.01281>.
- [6] Anton Bakhtin et al. “Mastering the Game of No-Press Diplomacy via Human-Regularized Reinforcement Learning and Planning”. In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. 2023.
- [7] Eli Bronstein et al. “Hierarchical Model-Based Imitation Learning for Planning in Autonomous Driving”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Kyoto, Japan, October 23-27, 2022*. IEEE, 2022, pp. 8652–8659. DOI: [10.1109/IROS47612.2022.9981695](https://doi.org/10.1109/IROS47612.2022.9981695). URL: <https://doi.org/10.1109/IROS47612.2022.9981695>.
- [8] Yiren Lu et al. “Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios”. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 7553–7560.
- [9] Charles R. Harris et al. “Array programming with NumPy”. In: *Nat.* 585 (2020), pp. 357–362. DOI: [10.1038/S41586-020-2649-2](https://doi.org/10.1038/S41586-020-2649-2). URL: <https://doi.org/10.1038/s41586-020-2649-2>.
- [10] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: <http://github.com/google/jax>.
- [11] Jason Ansel et al. “PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation”. In: *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, ASPLOS 2024, La Jolla, CA, USA, 27 April 2024- 1 May 2024*. ACM, 2024, pp. 929–947. DOI: [10.1145/3620665.3640366](https://doi.org/10.1145/3620665.3640366). URL: <https://doi.org/10.1145/3620665.3640366>.
- [12] Alexander Rutherford et al. “Jaxmarl: Multi-agent rl environments in jax”. In: *arXiv preprint arXiv:2311.10090* (2023).
- [13] Clément Bonnet et al. *Jumanji: a Diverse Suite of Scalable Reinforcement Learning Environments in JAX*. 2024. arXiv: [2306.09884](https://arxiv.org/abs/2306.09884) [cs.LG]. URL: <https://arxiv.org/abs/2306.09884>.
- [14] Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. “VMAS: A Vectorized Multi-agent Simulator for Collective Robot Learning”. In: *Distributed Autonomous Robotic Systems - 16th International Symposium, DARS 2022, Montbéliard, France, 28-30 November 2022*. Vol. 28. Springer Proceedings in Advanced Robotics. Springer, 2022, pp. 42–56. DOI: [10.1007/978-3-031-51497-5_4](https://doi.org/10.1007/978-3-031-51497-5_4). URL: https://doi.org/10.1007/978-3-031-51497-5_4.
- [15] Brennan Shacklett et al. “An extensible, data-oriented architecture for high-performance, many-world simulation”. In: *ACM Transactions on Graphics (TOG)* 42.4 (2023), pp. 1–13.
- [16] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. “CARLA: An open urban driving simulator”. In: *Conference on robot learning*. PMLR. 2017, pp. 1–16.

- [17] Panpan Cai, Yiyuan Lee, Yuanfu Luo, and David Hsu. “Summit: A simulator for urban driving in massive mixed traffic”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 4023–4029.
- [18] Quanyi Li et al. “Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning”. In: *IEEE transactions on pattern analysis and machine intelligence* 45.3 (2022), pp. 3461–3475.
- [19] Holger Caesar et al. “nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles”. In: *arXiv preprint arXiv:2106.11810* (2021).
- [20] Eugene Vinitsky, Nathan Lichtlé, Xiaomeng Yang, Brandon Amos, and Jakob Foerster. “Noc-turne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 3962–3974.
- [21] Cole Gulino et al. “Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research”. In: *Advances in Neural Information Processing Systems* 36 (2024).
- [22] Karsten Kreutz and Julian Eggert. “Analysis of the generalized intelligent driver model (GIDM) for uncontrolled intersections”. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2021, pp. 3223–3230.
- [23] Arne Kesting, Martin Treiber, and Dirk Helbing. “General lane-changing model MOBIL for car-following models”. In: *Transportation Research Record* 1999.1 (2007), pp. 86–94.
- [24] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. “Congested traffic states in empirical observations and microscopic simulations”. In: *Physical review E* 62.2 (2000), p. 1805.
- [25] Pablo Alvarez Lopez et al. “Microscopic Traffic Simulation using SUMO”. In: *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. URL: <https://elib.dlr.de/124092/>.
- [26] Jordi Casas, Jaime L Ferrer, David Garcia, Josep Perarnau, and Alex Torday. “Traffic simulation with aimsun”. In: *Fundamentals of traffic simulation* (2010), pp. 173–232.
- [27] Bernhard Wymann et al. “Torcs, the open racing car simulator”. In: *Software available at http://torcs.sourceforge.net* 4.6 (2000), p. 2.
- [28] Mark Martinez et al. “Beyond grand theft auto V for training, testing and enhancing deep learning in self driving cars”. In: *arXiv preprint arXiv:1712.01397* (2017).
- [29] Edouard Leurent et al. *An environment for autonomous driving decision-making*. 2018.
- [30] Matthias Müller, Vincent Casser, Jean Lahoud, Neil Smith, and Bernard Ghanem. “Sim4cv: A photo-realistic simulator for computer vision applications”. In: *International Journal of Computer Vision* 126 (2018), pp. 902–919.
- [31] Praveen Palanisamy. “Multi-agent connected autonomous driving using deep reinforcement learning”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–7.
- [32] Ming Zhou et al. “Smarts: An open-source scalable multi-agent rl training school for autonomous driving”. In: *Conference on Robot Learning*. PMLR. 2021, pp. 264–285.
- [33] Anirban Santara et al. “Madras: Multi agent driving simulator”. In: *Journal of Artificial Intelligence Research* 70 (2021), pp. 1517–1555.
- [34] Parth Kothari, Christian Perone, Luca Bergamini, Alexandre Alahi, and Peter Ondruska. “Drivergym: Democratising reinforcement learning for autonomous driving”. In: *arXiv preprint arXiv:2111.06889* (2021).
- [35] Alexander Amini et al. “Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 2419–2426.
- [36] Qiao Sun, Xin Huang, Brian C Williams, and Hang Zhao. “InterSim: Interactive traffic simulation via explicit relation modeling”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 11416–11423.
- [37] Adam Ścibior, Vasileios Lioutas, Daniele Reda, Peyman Bateni, and Frank Wood. “Imagining the road ahead: Multi-agent trajectory prediction via differentiable simulation”. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2021, pp. 720–725.
- [38] Danfei Xu, Yuxiao Chen, Boris Ivanovic, and Marco Pavone. “Bits: Bi-level imitation for traffic simulation”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 2929–2936.

- [39] M. Visvalingam and J. D. Whyatt. “Line generalisation by repeated elimination of points”. In: *The Cartographic Journal* 30 (1 1993), pp. 46–51. DOI: [10.1179/000870493786962263](https://doi.org/10.1179/000870493786962263).
- [40] Scott Ettinger et al. “Large Scale Interactive Motion Forecasting for Autonomous Driving : The Waymo Open Motion Dataset”. In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 9690–9699. DOI: [10.1109/ICCV48922.2021.00957](https://doi.org/10.1109/ICCV48922.2021.00957). URL: <https://doi.org/10.1109/ICCV48922.2021.00957>.
- [41] Chao Yu et al. “The surprising effectiveness of ppo in cooperative multi-agent games”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 24611–24624.
- [42] Joseph Suarez. “PufferLib: Making Reinforcement Learning Libraries and Environments Play Nice”. In: *arXiv preprint arXiv:2406.12905* (2024).
- [43] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

A Reproducibility

A.1 Code Reproducibility

All code required to reproduce the paper is open-sourced at <https://github.com/Emerge-Lab/gpudrive> under release number v0.1

A.2 Computational Resources

All RL experiments in this paper were run on an NVIDIA RTX 8000 or A100. Total resources for the paper correspond to less than 24 GPU-days.

B Vehicle Model

Our vehicles are driven by a kinematic bicycle model [43] which uses the center of gravity as reference point. The dynamics are as follows. Here (x_t, y_t) stands for the coordinate of the vehicle's position at time t , θ_t stands for the vehicle's heading at time t , v_t stands for the vehicle's speed at time t , a stands for the vehicle's acceleration and δ stands for the vehicle's steering angle. L is the distance from the front axle to the rear axle (in this case, just the length of the car) and l_r is the distance from the center of gravity to the rear axle. Here we assume $l_r = 0.5L$.

$$\begin{aligned}\dot{v} &= a \\ \bar{v} &= \text{clip}(v_t + 0.5 \dot{v} \Delta t, -v_{\max}, v_{\max}) \\ \beta &= \tan^{-1} \left(\frac{l_r \tan(\delta)}{L} \right) \\ &= \tan^{-1}(0.5 \tan(\delta)) \\ \dot{x} &= \bar{v} \cos(\theta + \beta) \\ \dot{y} &= \bar{v} \sin(\theta + \beta) \\ \dot{\theta} &= \frac{\bar{v} \cos(\beta) \tan(\delta)}{L}\end{aligned}$$

We then step the dynamics as follows:

$$\begin{aligned}x_{t+1} &= x_t + \dot{x} \Delta t \\ y_{t+1} &= y_t + \dot{y} \Delta t \\ \theta_{t+1} &= \theta_t + \dot{\theta} \Delta t \\ v_{t+1} &= \text{clip}(v_t + \dot{v} \Delta t, -v_{\max}, v_{\max})\end{aligned}$$

C License Details and Accessibility

Our code is released under an MIT License. The Waymo Motion dataset is released under a Apache License 2.0. The code is available at <https://github.com/Emerge-Lab/gpudrive> and at release commit TODO.

D Training details

D.1 End-to-end performance

The Table below depicts the hyperparameters used to produce the results in Section 4.2.

Table 2: Experiment hyperparameters used for comparing the training runs between Nocturne and GPUDrive in Figure 3. The environment configurations are aligned as closely as possible, using the same observations and field of view. The dataset includes the same 10 scenarios. It’s important to note that the length of the GPUDrive rollout is approximately equal to the number of worlds multiplied by the rollout length and then multiplied by the number of controllable agents. We have set this value to be $92 \times 50 \approx 4600$ to approximately match the rollout length in Nocturne.

Parameter	IPPO GPUDrive	IPPO Nocturne
γ	0.99	0.99
λ_{GAE}	0.95	0.95
PPO rollout length	92	4096
PPO epochs	5	5
PPO mini-batch size	2048	2048
PPO clip range	0.2	0.2
Adam learning rate	3e-4	3e-4
Adam ϵ	1e-5	1e-5
normalize advantage	yes	yes
entropy bonus coefficient	0.001	0.001
value loss coefficient	0.5	0.5
seeds	42, 12, 67	42, 12, 67
number of worlds	50	1

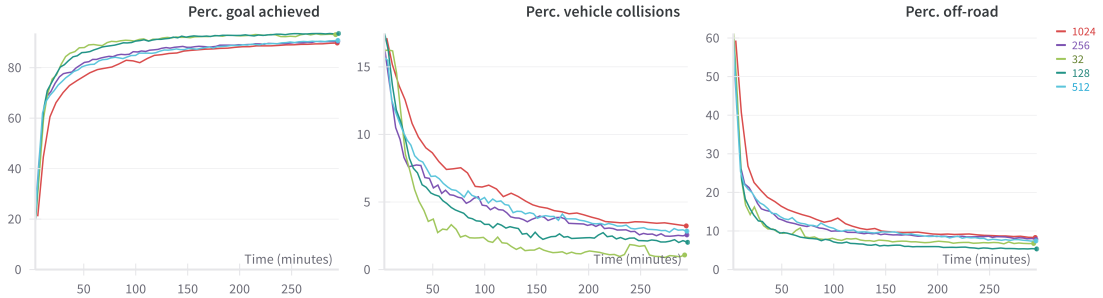


Figure 5: **Key performance metrics as a function of training time grouped by the number of unique scenes in a batch reported in Figure 4.** *Left:* The aggregate percentage of agents that achieved their goal. *Center:* The aggregate percentage of agents that collided with another vehicle. *Right:* The aggregate number of vehicles that crossed a road edge.