Robust Autonomy Emerges from Self-Play

Marco Cusumano-Towner [*1] **David Hafner** [*1] **Alex Hertzberg** [*1] **Brody Huval** [*1] **Aleksei Petrenko** [*1]
Eugene Vinitsky [*1] **Erik Wijmans** [*1] **Taylor Killian** [1] **Stuart Bowers** [1] **Ozan Sener** [1]
Philipp Krähenbühl [1] **Vladlen Koltun** [1]

Abstract

Self-play has powered breakthroughs in two-player and multi-player games. Here we show that self-play is a surprisingly effective strategy in another domain. We show that robust and naturalistic driving emerges entirely from self-play in simulation at unprecedented scale – 1.6 billion km of driving. This is enabled by GIGAFLOW, a batched simulator that can synthesize and train on 42 years of subjective driving experience per hour on a single 8-GPU node. The resulting policy achieves state-of-the-art performance on three independent autonomous driving benchmarks. The policy outperforms the prior state of the art when tested on recorded real-world scenarios, amidst human drivers, without ever seeing human data during training. The policy is realistic when assessed against human references and achieves unprecedented robustness, averaging 17.5 years of continuous driving between incidents in simulation.

1. Introduction

Self-play has been an effective strategy for training policies for board games, card games, 3D multiplayer games, real-time strategy games, robotic manipulation, and even bio-engineering (Silver et al., 2017; 2018; Jaderberg et al., 2019; Berner et al., 2019; Brown & Sandholm, 2019; Vinyals et al., 2019; Plappert et al., 2021; Perolat et al., 2022; Wang et al., 2023a). In this work, we demonstrate the effectiveness of self-play in another domain. We show that simulated self-play yields naturalistic and robust driving policies, while using only a minimalistic reward function and never seeing human data during training.

We demonstrate that qualitatively new levels of realism and robustness emerge when self-play training is taken to unprecedented scale – orders of magnitude beyond prior experiments (Feng et al., 2023; Zhang et al., 2023). This discovery is enabled by GIGAFLOW, a batched simulator architected from the ground up for self-play reinforcement learning on a massive scale. GIGAFLOW is capable of simulating and learning from 4.4 billion state transitions (7.2 million km of driving, or 42 years of continuous driving experience) per hour on a single 8-GPU node. It simulates urban environments with up to 150 densely interacting traffic participants $360\,000$ times faster than real time at a cost of under $5 per million km driven (based on public cloud rates). A full training run simulates over one trillion state transitions, 1.6 billion km driven, or 9500 years of subjective driving experience, and completes in under 10 days on one 8-GPU node.

We use GIGAFLOW to train a parameterized family of driving policies. The parameters specify the type of traffic participant controlled by the policy (passenger vehicle, large truck, bicyclist, or even a pedestrian) and the driving style (e.g., aggressive vs. cautious). These parameters can be modified at test time with no additional training (Dosovitskiy & Koltun, 2017), such that a single trained policy can be used to control a variety of traffic participants, with a variety of behavioral styles. During training, this parameterized policy architecture enables all simulated traffic participants to be collecting experience in parallel, all flowing through a single neural network. This supports self-play simulations where more than a hundred agents are all controlled by a single neural network, which is learning from all of their experiences, yet the agents exhibit diverse outward manifestations (truck vs. bicycle), functional characteristics (turning radius), and behavioral styles (adherence to traffic laws).

The result is a robust and naturalistic driving policy that achieves state-of-the-art performance when tested in recorded real-world scenarios, amidst recorded human drivers, without ever seeing human data during training. We test the GIGAFLOW policy in three leading independent third-party benchmarks: CARLA (Dosovitskiy et al., 2017), nuPlan (Caesar et al., 2022), and the Waymo Open Motion Dataset (Ettinger et al., 2021) (through the Waymax simulator (Gulino et al., 2023)). State-of-the-art performance

[*]Equal contribution, alphabetical order. [1]Apple. Correspondence to: Philipp Krähenbühl <philkr@apple.com>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

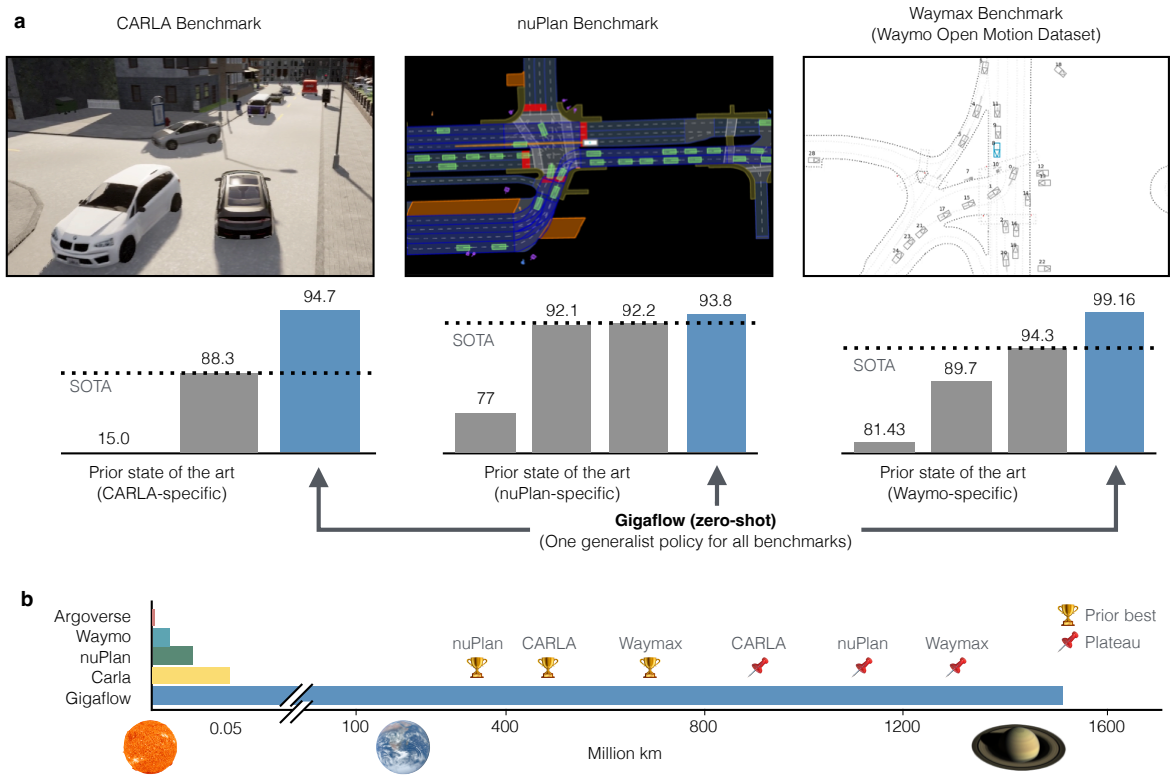

Figure 1: **Self-play reinforcement learning yields a generalist policy**. **a**, A single GIGAFLOW agent outperforms the best dataset-specific specialists across leading benchmarks. GIGAFLOW is evaluated zero-shot without training on a single benchmark, while dataset-specific specialists train on benchmark-specific datasets. Each benchmark includes different maps, scenarios, and evaluation metrics. **b**, GIGAFLOW enables cost-effective training of policies via self-play on a massive scale. Our largest policies drive over 1.6 billion km during training, more than the distance from the Sun to Saturn and orders of magnitude farther than prior datasets or simulations. At this scale, self-play yields a generalist policy. Dashed lines indicate points at which the performance of our single generalist policy passes the prior state of the art on each benchmark ('prior best') and points at which the performance on each benchmark plateaus ('plateau').

on each benchmark was previously achieved by specialist agents that were trained specifically for that benchmark, commonly using benchmark-specific datasets. In contrast, we outperform the prior state of the art on all benchmarks with a single policy (Fig. 1) that was trained entirely via self-play, using none of the provided datasets for training.

The behaviors exhibited by the GIGAFLOW policy are naturalistic despite never seeing human data during training. The trained policy exhibits long-horizon planning without any dedicated planning or search modules, can deal with heavily contentious traffic scenarios, is quantitatively realistic when assessed against human references (Montali et al., 2023), and exhibits unprecedented robustness, averaging over 3 million km (or 17.5 years of continuous driving) between incidents in simulation.

2. GIGAFLOW

The goal of GIGAFLOW is to train a *generalist policy* $\pi(a|W, S, A, C)$ in simulation (Fig. 2d). The policy observes the static world W, its own state S, and other dynamic agents A to produce an action a (Fig. 2c). A conditioning parameter C modulates the policy's behavior. Learning this generalist policy requires careful modeling of two core concepts: uncertainty and other-agent behaviors.

Uncertainty in driving stems from partial or incomplete observations. The driver is generally unaware of the goals and intentions of other agents, or even their exact location, speed, or acceleration. Objects or parts of the static world may be hidden or occluded. Real-world sensors often introduce noise. GIGAFLOW models uncertainty directly through noise on the state S, noise in the state transitions, stochasticity in the dynamic agents, and partial observability on dynamic agents A and the static world W. GIGAFLOW agents observe the positions and speeds of nearby agents

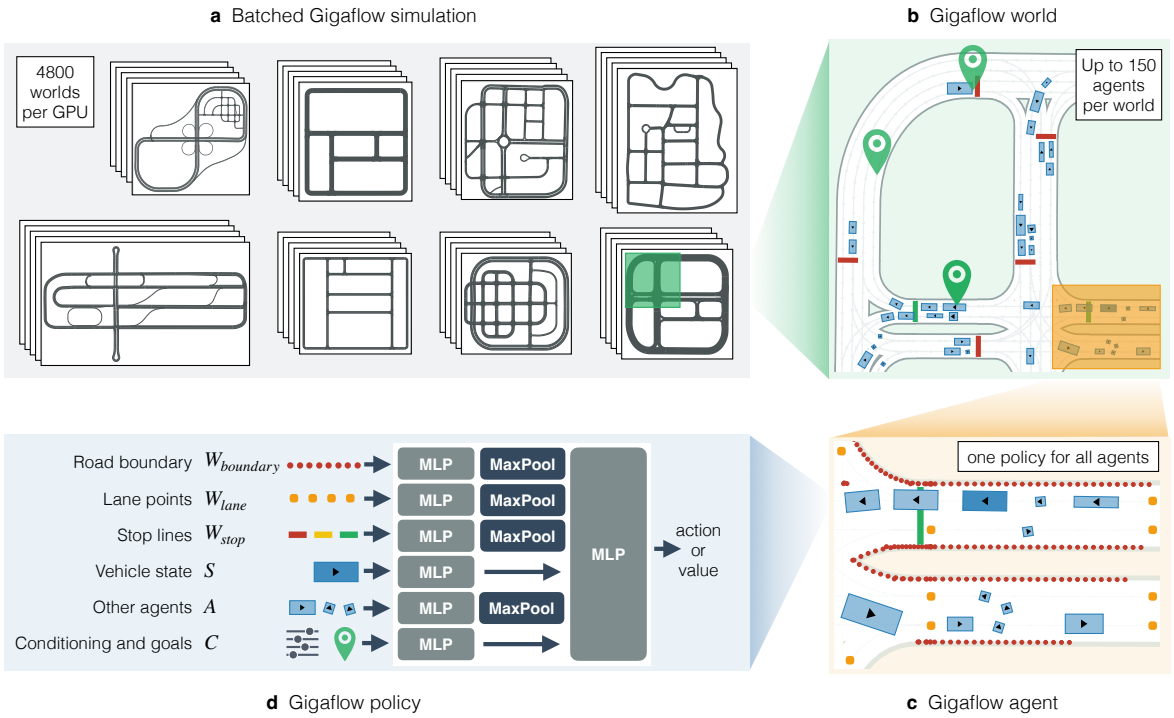

Figure 2: **Overview of GIGAFLOW. a**, GIGAFLOW simulates tens of thousands of worlds with millions of agents in a massively parallel self-play reinforcement learning setup. **b**, Each world tasks agents to navigate to goals on a map without collisions. **c**, Each agent optimizes its own performance, given a set of local observations. **d**, All agents use a compact shared policy network.

but not their acceleration, and crucially, neither their goals nor conditioning.

Modeling agent behaviors is a particularly impactful and complex aspect of driving. Prior work approached behavior models through hand-designed agents (Kesting et al., 2010; 2007; Gulino et al., 2023), recorded and replayed data (Gulino et al., 2023; Li et al., 2023; Vinitsky et al., 2022), or models of driving learned from data (Ivanovic & Pavone, 2019; Nayakanti et al., 2023; Wang et al., 2023b; Suo et al., 2021; Shi et al., 2022; Xu et al., 2023; Zhong et al., 2023). In contrast, in GIGAFLOW, realistic and general driving behaviors for all traffic participants emerge via self-play reinforcement learning on a massive scale.

2.1. GIGAFLOW world

The GIGAFLOW world is simple: we do not script scenarios, use human driving traces, or design delicate reward terms. We show that simulation at massive scale makes up for much of this simplicity (Fig. 2).

Agents train on one of eight maps, randomly perturbed with rescaling, shears, flips and reflections. Total drivable lanes per map range from four to 40 km for a total of 136 km of road across the eight maps (Fig. 2a). In each map, we spawn one to N_a agents at random locations and orientations on the road and ask them to reach goal points sampled uniformly over the map. This creates a world in which agents drive for long distances before reaching their destinations (Fig. 2b). Agents are tasked with visiting a variable number of intermediate waypoints, requiring the ability to follow complex routes (see Appendix B for details).

Dense traffic flows with diverse interactions emerge as agents navigate to their destinations. As training progresses, we can observe agents executing zipper merges and tight maneuvers in traffic jams, managing congested roundabouts and uncontrolled intersections, resolving occasional gridlocks, and performing multi-point turns to reroute around accidents or obstructions.

GIGAFLOW agents train fully in self-play. All dynamic agents – vehicles, pedestrians, and cyclists – use the same single reactive parametric policy π; their behaviors are varied through conditioning C (Fig. 2d). The policy is aware of the dynamics of the agent it controls as part of the conditioning C_{dynamics}. The agent reward is a mixture of incentives to reach its goal, avoid collisions, drive centered and lane aligned, as well as penalties for running red lights or stop signs, and exceeding acceleration and jerk limits. The weights on each of these reward components are randomized per agent and provided as conditioning C_{reward} to the agent (see Appendix B for details). This allows a single reactive policy π to exhibit a wide range of behaviors. The result

is a diverse training world where agents learn a continuum of driving styles: some drive cautiously, others are likely to run traffic lights, while a rare few are willing to drive against the flow of traffic. Because the policy only observes the conditioning C of the agent it controls, it must learn to be robust to the unpredictable behaviors of other drivers.

2.2. GIGAFLOW simulation and training

The GIGAFLOW simulation and training framework is designed to optimize driving data collection and training throughput per unit of computation. We simulate and learn from 4.4 billion state transitions per hour on a single 8-GPU node, rolling out urban commute simulations $360\,000$ times faster than real time at the cost of under $5 per million kilometers driven (based on public cloud rates). These training rates require three core ingredients: a fast batched simulator (Shacklett et al., 2021; Petrenko et al., 2021; Shacklett et al., 2023), a compact and expressive policy for fast inference and backpropagation, and a high-throughput training algorithm.

GIGAFLOW simulation. GIGAFLOW simulates $38\,400$ environments in parallel across 8 GPUs with up to $N_a = 150$ vehicles each (Fig. 2a). Basic operations, such as policy inference and dynamics updates are batched across all agents. Agent localization, collision checking, and observation construction rely on dedicated optimized data structures. Due to the large map sizes, we precompute and cache all map observations in a spatial hash and perform fast, GPU-based runtime lookup and retrieval. Agents perceive the map by observing sets of points sampled sparsely along drivable lanes W_{lane}, and densely along the nearby road edges for precise maneuvering W_{boundary} (Fig. 2c).

Beyond map features, agents get observations of nearby traffic participants A – containing nearby vehicles' sizes, locations, orientations, and velocities – and nearby stop lines and traffic lights W_{stop}. GIGAFLOW models static obstacles as immobile vehicles A_{static}. To reduce memory, we do not store these observations in the rollout buffer, but calculate them on demand from stored world states. See Appendix A for a detailed description of the simulator.

GIGAFLOW policy. GIGAFLOW can simulate diverse actor types, from pedestrians to heavy trucks, by parameterizing a single unified feed-forward policy (Fig. 2d). The decision to use the same underlying neural network policy for all traffic participants significantly impacts the overall throughput: we need only a single (batched) forward pass per simulation step to calculate actions for all agents. The policy resembles a Deep Sets architecture (Zaheer et al., 2017) and is invariant to permutation w.r.t. each observation type. Critically, the entire trainable artifact is relatively compact at six million parameters. On an 8-GPU A100 node, the policy allows inference throughput of 7.4 million decisions per second during experience collection at a batch size of 2.6 million, and eight gradient updates per second in the training phase with a batch size of $256\,000$. See Appendix D for more details.

GIGAFLOW training. We train the GIGAFLOW policy using Proximal Policy Optimization (PPO) (Schulman et al., 2017). One of the main challenges associated with autonomous driving is the inherent imbalance in the data distribution. As training progresses, the on-policy data is dominated by ordinary traffic configurations, such as orderly driving in a straight line between intersections. The critic is often able to accurately predict the returns for such trajectories, resulting in a large portion of samples with near-zero advantage (Greensmith et al., 2004) that consequently yield vanishingly small gradients.

We use a variant of Prioritized Experience Replay (Schaul et al., 2016) that filters samples that have minimal impact on learning. The filtering is based on the absolute value of the estimated advantage. We filter up to 80% of samples with low absolute advantage, which significantly increases learning throughput without sacrificing sample efficiency. Our approach, which we refer to as *advantage filtering*, focuses training on the most informative state transitions, prioritizing learning from the underexplored tails of the data distribution where selected actions are measurably better or worse, and makes more efficient use of the data we generate. See Appendix C for more details.

3. Zero-shot evaluation on driving benchmarks

We evaluate a trained GIGAFLOW policy on the leading closed-loop driving benchmarks: CARLA (Dosovitskiy et al., 2017), nuPlan (Caesar et al., 2022), and the Waymo Open Motion Dataset (Ettinger et al., 2021) through the Waymax simulator (Gulino et al., 2023). These benchmarks encompass a wide range of actor behaviors, driving scenarios, maps, traffic densities, durations, and scoring methodologies. The CARLA benchmark consists of routes with hand-designed scenarios based on the NHTSA pre-crash topology (Najm et al., 2007). It evaluates long distance driving (several minutes per 1–3 km route). nuPlan and Waymax evaluate short distance driving (8–14 seconds per scenario, < 100 m) in scenarios derived from recorded real-world driving with the associated sensor data.

A generalist GIGAFLOW policy outperforms state-of-the-art specialists. For each benchmark, we compare to specialist state-of-the-art policies that are either trained (Gulino et al., 2023) or carefully hand-designed (Chitta et al., 2023; Jaeger et al., 2023; Dauner et al., 2023) to perform well on that specific benchmark. In contrast, we use a single policy across all benchmarks. Our policy is trained purely in self-play and is evaluated zero-shot in each benchmark

environment. Without any fine-tuning, our policy surpasses the state of the art in CARLA, nuPlan, and Waymax (Fig. 1 with details in Tables A5 to A7 and Appendix E). This demonstrates robust driving with strong generalization. Our self-play policy outperforms the state of the art on real driving traces with human traffic participants, without ever seeing human data during training.

GIGAFLOW policy generalizes to diverse actor behaviors. The benchmarks implement a diverse set of environment actors. CARLA uses reactive rules-based vehicles with lane-changing capabilities, combined with events triggered by the driver's behavior (e.g., a pedestrian that darts suddenly in front of the driver). The actors in nuPlan and Waymax are controlled by different variants of the Intelligent Driver Model (Treiber et al., 2000). Vehicles in nuPlan follow the lane center line, whereas vehicles in Waymax follow the paths of logged human drivers. The GIGAFLOW policy exhibits robust driving amongst all of these actor types.

GIGAFLOW policy generalizes to diverse maps and driving situations. GIGAFLOW trains on variants of synthetic maps with closed road networks (Dosovitskiy et al., 2017), but generalizes to the real-world maps in nuPlan (Caesar et al., 2022) and in the Waymo Open Motion Dataset (WOMD) (Ettinger et al., 2021). The WOMD maps are small, with incomplete road networks constructed from logs of instrumented vehicles in several US cities. The nuPlan benchmark is based on driving logs of human drivers in locales with both right-handed and left-handed driving (Caesar et al., 2020); it contains larger maps that encompass the entire testing area of the vehicle. Both nuPlan and WOMD scenarios include merges, unprotected turns, and interactions with pedestrians and cyclists (Ettinger et al., 2021). The GIGAFLOW policy achieves state-of-the-art results in these benchmarks without any training on recorded driving logs or any human-designed scenarios.

GIGAFLOW policy generalizes to real-world observation noise. Both Waymax and nuPlan construct observations, maps, and other actors with auto-labeling tools from real-world perception data. This brings occlusion, incorrect or missing traffic-light states, and obstacles revealed at the last moment. Despite the minimalistic noise modeling in GIGAFLOW, the GIGAFLOW policy generalizes zero-shot to these conditions.

GIGAFLOW policy is state-of-the-art according to multiple scoring methodologies. Each benchmark brings its own definition of 'good driving'. Those definitions are distinct and sometimes contradictory. For example, running a red light in CARLA incurs nearly the same penalty as colliding with another vehicle. Yet the same action can be advantageous in nuPlan, where red light violations are ignored by the scoring criteria, hard braking causes comfort penalties, and forward progress is strongly rewarded. Despite such variations, the single generalist GIGAFLOW driver outperforms specialist policies optimized for individual benchmark scores.

The GIGAFLOW policy approaches the ceiling of benchmark performance. The vast majority of the infractions sustained by the GIGAFLOW policy during testing on the benchmarks can be attributed to limitations of the benchmarks. For instance, 20% of the reported infractions in CARLA are caused by pedestrians or cyclists darting from the sidewalk into the roadway without reacting to the evasive maneuver of the driver or other traffic participants. Preventing such collisions would require drastic overfitting to this type of scenario (Jaeger et al., 2023). Other exemplary limitations are gridlocks caused by CARLA-controlled traffic (33% of all infractions) or fuzzy stop sign and red light checks (16% of all infractions).

In nuPlan our policy sustains 15 collisions in 1118 scenarios. We analyzed each of them. Nine are unavoidable due to invalid initialization or sensor noise (agents appearing inside the vehicle's bounding box). Four are caused by non-reactive pedestrian agents walking into the vehicle while the vehicle was stopped or in an evasive maneuver. Two collisions are due to traffic light violations of other agents.

In Waymax our policy sustains 187 collisions in $44\,097$ scenarios. We again analyzed each of them. 55.6% were caused by unavoidable IDM agent behavior (Treiber et al., 2000) of the traffic participants controlled by the benchmark, such as swerving directly into the ego vehicle. 41.7% were caused by initialization in a state of collision, typically with a pedestrian. 2.7% (i.e. five scenarios) were considered at-fault and avoidable by the GIGAFLOW policy. Of the at-fault collisions, there were additional contributing factors such as perception issues or aggressive and spurious IDM behaviors. One example is when the GIGAFLOW policy seeks to avoid a rear-end collision with an IDM agent approaching from behind at high speed.

We include videos of all reported infractions in the supplementary material.

4. Analysis

GIGAFLOW training employs two neural networks: the policy (actor) that chooses actions and the value function approximator (critic) that estimates the expected cost-to-go from a given state. We examine how the policy's driving behavior changes over the course of training (Fig. 3) and how the policy and value networks respond to targeted changes to their inputs in various scenarios (Fig. 4).

Reinforcement learning at scale yields mastery of complex skills. The scale of GIGAFLOW training enables the policy to handle complex scenarios despite never seeing real-

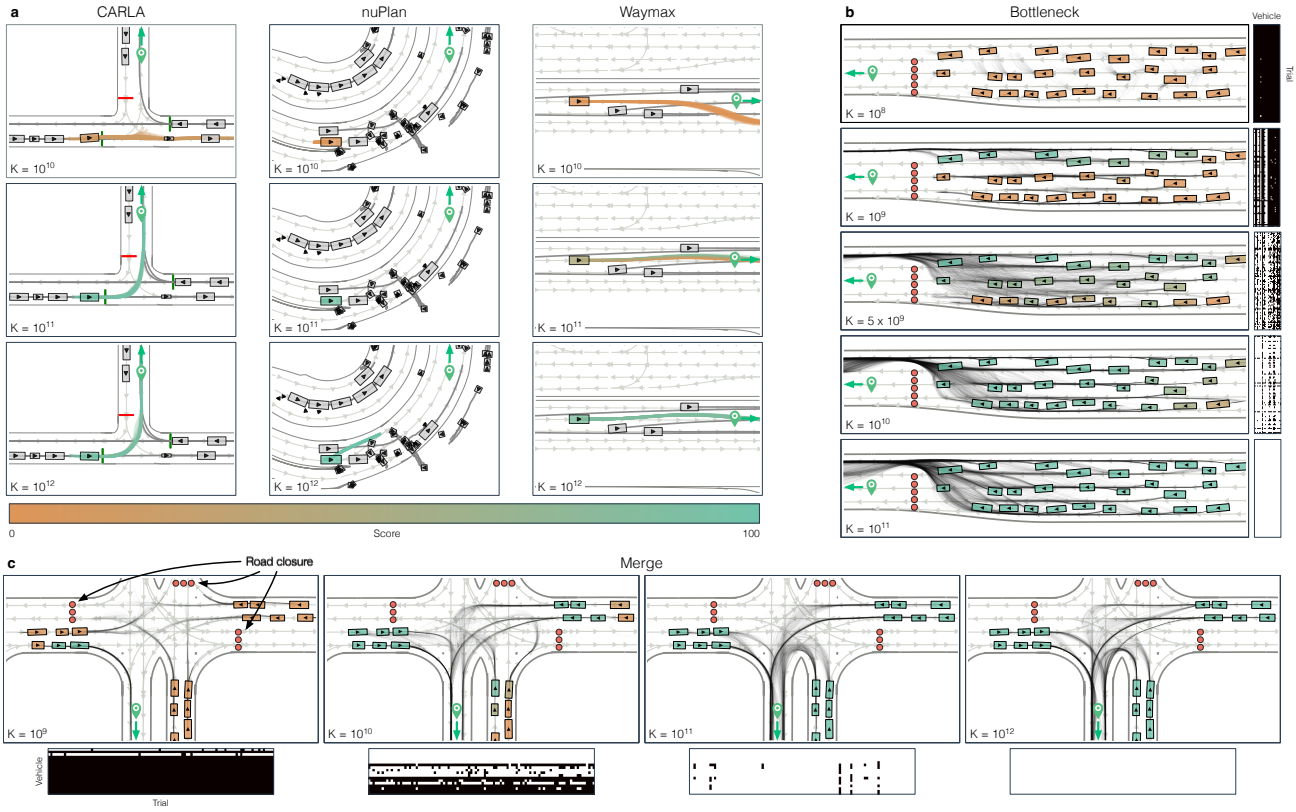

Figure 3: **Evolution of the policy during training.** **a**, Zero-shot performance on example benchmark scenarios with increasing number of state transitions (K) seen during training. From top to bottom: $K = 10^{10}$ (3 million km), $K = 10^{11}$ (90 million km), and $K = 10^{12}$ (1600 million km). The color of the controlled vehicle and its trajectories indicate the benchmark score. **b**, Performance of the GIGAFLOW policy on a diagnostic scenario where a static obstruction (red bounding boxes) forces a merge of several highway lanes into one. All agents are controlled by the same policy. Trajectories from 100 independent rollouts are shown. Agent colors indicate probability of success (moving past the bottleneck). Binary matrices on the right indicate the success of each actor in each rollout (white indicates success). After $K = 10^8$ state transitions, the agents drive out of their lanes and crash. After $K = 10^9$, the agents drive forward, avoid collision, and sometimes successfully change lanes, but do not successfully merge across multiple lanes (only agents on the right succeed). After $K = 5 \times 10^9$, merging ability emerges but agents in the leftmost lane usually fail. After $K = 10^{10}$, agents sometimes merge from the left lane, but not reliably. After $K = 10^{11}$, all agents reliably succeed with no incidents. **c**, Performance of the GIGAFLOW policy in a diagnostic scenario where road closures require three traffic flows to merge into one without the aid of traffic lights. As training progresses, the policy goes from only succeeding at the right turn to successfully completing unprotected left turns and u-turns. After $K = 10^{12}$ steps of training, all agents reliably succeed. Binary matrices show the worst 100 samples from 1000.

world or hand-designed driving scenarios during training. The policy learns to execute unprotected left turns, drive in crowded roads used by both pedestrians and vehicles, and handle vehicles dangerously merging into the driver's lane (Fig. 3a). In diagnostic tests designed for analysis, GIGAFLOW vehicles are able to safely negotiate through a narrow bottleneck into a single lane when the other lanes are blocked by an accident (Fig. 3b) and quickly merge three traffic flows into a single one due to road closures (Fig. 3c). Many of these skills are mastered only after 10^{11} to 10^{12} steps of training experience (90 to 1600 million km driven).

Value network detects dangerous states. To examine the value network's ability to detect dangerous states, we evaluate it on a set of observations generated by densely sampling all possible positions and orientations of the driver on a fixed region of the map. We find that the network appropriately assigns low value to states where the driver is taking a corner too fast, and where collision with another vehicle is imminent due to high relative velocity (Fig. 4a).

Policy and value networks attend to salient scene features. Driving requires attending to the most consequential traffic participants at any given time, among hundreds of

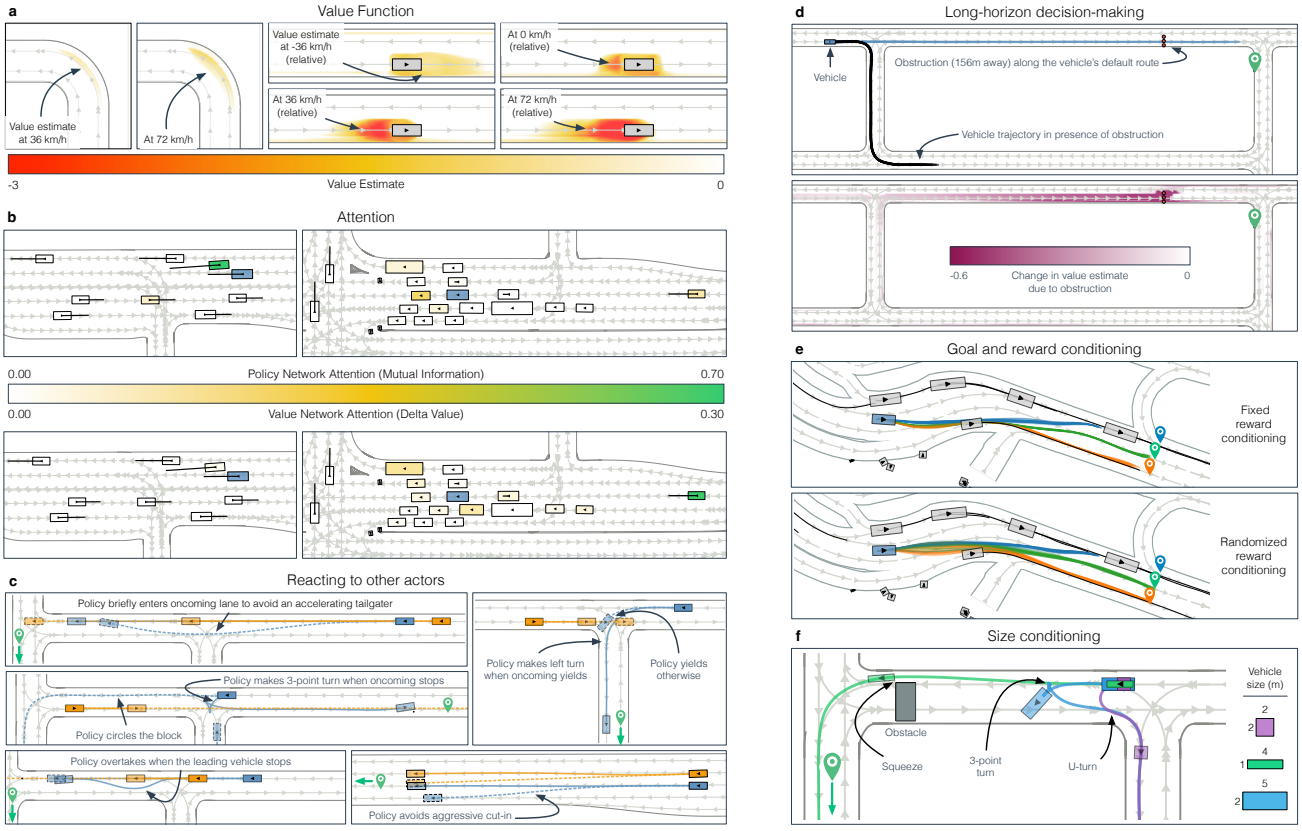

Figure 4: **Analysis of the learned policy and value networks. a**, Value network estimates danger at fine granularity over the surface of the road. Heatmaps show state-value estimates for a driver placed at every possible location, oriented in parallel to the nearest lane. Left: The network assigns lower value to rounding a sharp corner at higher speed. Right: Value estimates for driving at various speeds in the vicinity of a vehicle (gray) moving 36 km/h to the right. Being placed at rest (0 km/h) in front of the moving vehicle is likely to result in collision. As velocity increases, the danger zone shifts behind the other vehicle. The estimated danger becomes acute as the relative speed increases. **b**, Attention of the policy and value networks of the controlled vehicle (blue) to surrounding actors in two scenarios from the Waymo Open Motion Dataset. Top row: Mutual information between the policy network's action distribution and actor presence. Bottom row: Decrease in the value network's state-value estimate due to actor presence. **c**, Scenarios where a scripted nearby vehicle (orange) either continues moving forward at constant velocity (solid orange line) or executes a different behavior (dashed orange line). The policy (blue vehicle) executes maneuvers (solid blue line, dashed blue line) contingent on the other vehicle's behavior. Final poses are translucent. **d**, Policy and value networks perform long-horizon prediction and decision-making. Top: Policy drives to its goal (20 second trajectory in blue) when the road is clear. When an obstruction (black) is placed 156 m away, the policy re-routes (black trajectory). Bottom: Heatmap shows change in value estimate over the surface of the road due to introduction of the obstruction. **e**, Top: Diversity of trajectories due to goal conditioning (three goals shown with associated trajectories in matching color). Bottom: Additional variability due to randomized reward conditioning parameters. **f**, Changing vehicle dimensions with all else fixed yields different behaviors: A slender vehicle can squeeze around an obstacle; a compact vehicle with a tight turning radius can make a u-turn; a larger vehicle resorts to a three-point turn.

actors who may be present in the environment. We assess the attention of our policy and value networks by analyzing the change in action distribution (via mutual information) and change in value estimate when each actor is individually removed (Fig. 4b). As expected, the networks sometimes attend to different actors: For example, the value estimate is affected by all actors that make the scene more dangerous over the long term (for example a speeding car approaching a line of vehicles queued at a red traffic light), whereas the policy's action might not change due to such an actor if there is no way for the policy to mitigate the danger (Fig. 4b).

Policy executes maneuvers contingent on nearby traffic behavior. We evaluate the policy in scenarios where a nearby vehicle either behaves predictably (continuing to move at constant velocity) or unpredictably, with all else

fixed. We find that the policy executes appropriate discrete maneuvers contingent on the nearby vehicle behavior, like changing lanes to avoid collision with a vehicle that is cutting into its lane and passing a vehicle that unexpectedly stops in the road. The policy executes contingent longer-term routing maneuvers, like turning around by circling the block instead of making a three-point turn, depending on traffic (Fig. 4c).

Policy reacts to potential events far in the future. Both networks are also able to react to salient scene features even when they are distant, like an obstruction $150\,\mathrm{m}$ down the road (Fig. 4d), and can ignore actors that are near but irrelevant (e.g., a car parked few meters from the driver in a parking lot). While considering potential events beyond the planning horizon is often a challenge for trajectory-based planners (Casas et al., 2021), the GIGAFLOW policy optimizes long-term return directly without the limitations of a short time horizon.

One policy learns a continuum of driving styles. Reward function coefficients C_{reward}, vehicle dimensions, and vehicle goals are all randomized during training. As a result, the policy learns a parameterized family of driving styles (Fig. 4e,f; Fig. A1); different styles can be elicited from a single trained policy by setting the parameters accordingly, without any retraining or fine-tuning. For example, the policy squeezes through narrow passages or performs tight turns if and only if the vehicle dimensions permit this (Fig. 4f). Likewise, reducing the conditioning parameter that controls sensitivity to red lights makes the policy more willing to run red lights in order to accomplish other goals.

GIGAFLOW yields a highly efficient, capable, and realistic simulation environment. The GIGAFLOW training configuration features substantial dynamics noise and diverse reward conditioning parameters C_{reward} (see Section 2). We can configure the same simulation infrastructure for long-form evaluation of trained policies. In this configuration, we reduce the injected dynamics noise, increase control frequency, and set conditioning parameters C_{reward} that prioritize safety for all actors. This yields a fast, cost-effective, and highly robust traffic simulator. In this regime, a fully trained GIGAFLOW agent experiences on average 17.5 years of driving and travels over 3 million km before encountering an incident. (For reference, human drivers average approximately $829\,000$ vehicle kilometers traveled per police-reported traffic crash in the United States (Stewart, 2023), or as much as 1 crash per $24\,800\ \mathrm{km}$ in narrower domains such as San Francisco (Flannagan et al., 2023).) To our knowledge this is the first demonstration of long-term robust traffic simulation based on independent agents traversing a diverse urban road network.

We evaluate the realism of the driving behaviors learned via GIGAFLOW on the Waymo Open Sim Agents Challenge (WOSAC), which measures the ability to reproduce real-world driving behaviors for simulation purposes (Montali et al., 2023). Despite not using any human data for training, the GIGAFLOW policy exhibits many characteristics of human driving, achieving a score of 0.62 in zero-shot evaluation on the realism meta-metric, outperforming several approaches based on supervised autoregressive prediction. (Details in Section E.4.)

5. Discussion

Many questions remain to fully understand the long-term role of self-play in delivering broad-competence robust autonomy. First, our work has been conducted entirely in simulation. Techniques for transferring policies from simulation to reality will have to be brought to bear before claims can be made regarding the efficacy of self-play policies in the physical world (Müller et al., 2018; Lee et al., 2020; Kaufmann et al., 2023).

Second, our work has focused on planning and decision-making, largely abstracting the perception stack. To integrate the presented findings into an operational system, sensing and perception will have to be modeled much more closely. An exciting possibility is to combine large-scale self-play training with data-driven simulation of the associated perceptual inputs (e.g., camera images) (Ost et al., 2021; Yang et al., 2023; Hu et al., 2023). It is likely feasible in the coming years due to ongoing improvements in simulation methodology (Shacklett et al., 2021; Petrenko et al., 2021; Shacklett et al., 2023), computing hardware, and system architectures. Combining self-play with photorealistic sensor simulation would substantially increase the computational footprint of each experience, but the wall-clock training time can be maintained by scaling out over a commensurate number of compute nodes (Dubey et al., 2024).

Third, our work has demonstrated that training without real-world driving traces can yield policies that are surprisingly human-like (Montali et al., 2023) and highly robust when tested in recorded real-world scenarios with human participants (Caesar et al., 2022; Gulino et al., 2023). By contrast, common perspectives on learning-based autonomous driving hold that recorded datasets will play a key role in training driving policies (Jain et al., 2021; Hawke et al., 2021; Chen et al., 2023). How do we reconcile our findings with these views? One possibility is to combine large-scale self-play training with training on recorded scenarios, perhaps via a combination of reinforcement learning and imitation learning (Lu et al., 2023; Zhang et al., 2023). This can further increase robustness and help bridge simulation and reality.

Impact Statement

Our findings may inspire broader application of self-play in training agents that act in the presence of (and in close coordination with) humans in physical and digital environments. Such coordinated action may be called for in mobile robotics, in both consumer and industrial settings, and in digital domains such as online games. We have shown that policies that function effectively in the presence of human actors in complex dynamic environments can be trained without utilizing human data. Broader application of this methodology may substantially reduce the cost and complexity of training autonomous policies by meaningfully reducing the need for human data collection.

References

Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., et al. PyTorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2024.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., de Oliveira Pinto, H. P., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., and Zhang, S. Dota 2 with large scale deep reinforcement learning. *arXiv:1912.06680*, 2019.

Brown, N. and Sandholm, T. Superhuman AI for multiplayer poker. *Science*, 365(6456):885–890, 2019. doi: 10.1126/science.aay2400.

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. nuScenes: A multimodal dataset for autonomous driving. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Caesar, H., Kabzan, J., Tan, K. S., Fong, W. K., Wolff, E., Lang, A., Fletcher, L., Beijbom, O., and Omari, S. nuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles, 2022. URL https://arxiv.org/abs/2106.11810.

Casas, S., Sadat, A., and Urtasun, R. MP3: A unified model to map, perceive, predict and plan. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Chen, D. and Krähenbühl, P. Learning from all vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Chen, L., Wu, P., Chitta, K., Jaeger, B., Geiger, A., and Li, H. End-to-end autonomous driving: Challenges and frontiers, 2023. URL http://arxiv.org/abs/2306.16927.

Chitta, K., Prakash, A., Jaeger, B., Yu, Z., Renz, K., and Geiger, A. TransFuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):12878–12895, 2023.

Dauner, D., Hallgarten, M., Geiger, A., and Chitta, K. Parting with misconceptions about learning-based vehicle motion planning. In *Conference on Robot Learning (CoRL)*, 2023.

Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

Dosovitskiy, A. and Koltun, V. Learning to act by predicting the future. *International Conference on Learning Representations (ICLR)*, 2017.

Dosovitskiy, A., Ros, G., Codevilla, F., López, A. M., and Koltun, V. CARLA: An open urban driving simulator. *Conference on Robot Learning (CoRL)*, 78:1–16, 2017.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Ettinger, S., Cheng, S., Caine, B., Liu, C., Zhao, H., Pradhan, S., Chai, Y., Sapp, B., Qi, C. R., Zhou, Y., Yang, Z., Chouard, A., Sun, P., Ngiam, J., Vasudevan, V., McCauley, A., Shlens, J., and Anguelov, D. Large scale interactive motion forecasting for autonomous driving: The Waymo open motion dataset. In *International Conference on Computer Vision (ICCV)*, 2021.

Feng, S., Sun, H., Yan, X., Zhu, H., Zou, Z., Shen, S., and Liu, H. X. Dense reinforcement learning for safety validation of autonomous vehicles. *Nature*, 615(7953):620–627, 2023. doi: https://doi.org/10.1038/s41586-023-05732-2.

Flannagan, C., Leslie, A., Kiefer, R., Bogard, S., Chi-Johnston, G., Freeman, L., Huang, R., Walsh, D., and Anthony, J. Establishing a crash rate benchmark using large-scale naturalistic human ridehail data. Technical Report UMTRI-2023-18, University of Michigan Transportation Research Institute, 2023. URL https://dx.doi.org/10.7302/8636.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. Brax - A differentiable physics engine for large scale rigid body simulation. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

Greensmith, E., Bartlett, P. L., and Baxter, J. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 2004.

Gulino, C., Fu, J., Luo, W., Tucker, G., Bronstein, E., Lu, Y., Harb, J., Pan, X., Wang, Y., Chen, X., Co-Reyes, J. D., Agarwal, R., Roelofs, R., Lu, Y., Montali, N., Mougin, P., Yang, Z., White, B., Faust, A., McAllister, R., Anguelov, D., and Sapp, B. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. In *Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Guo, Z., Gao, X., Zhou, J., Cai, X., and Shi, B. SceneDM: Scene-level multi-agent trajectory generation with consistent diffusion models, 2023. URL https://arxiv.org/abs/2311.15736.

Hallgarten, M., Stoll, M., and Zell, A. From prediction to planning with goal conditioned lane graph traversals. In *International Conference on Intelligent Transportation Systems*, 2023. doi: 10.1109/ITSC57777.2023.10421854.

Hansen, E. A., Bernstein, D. S., and Zilberstein, S. Dynamic programming for partially observable stochastic games. In *Conference on Artificial Intelligence*. AAAI Press / The MIT Press, 2004. URL http://www.aaai.org/Library/AAAI/2004/aaai04-112.php.

Hawke, J., E, H., Badrinarayanan, V., and Kendall, A. Reimagining an autonomous vehicle, 2021. URL https://arxiv.org/abs/2108.05805.

Hu, A., Russell, L., Yeo, H., Murez, Z., Fedoseev, G., Kendall, A., Shotton, J., and Corrado, G. GAIA-1: A generative world model for autonomous driving, 2023. URL https://arxiv.org/abs/2309.17080.

Huang, Z., Zhang, Z., Vaidya, A., Chen, Y., Lv, C., and Fisac, J. F. Versatile scene-consistent traffic scenario generation as optimization with diffusion, 2024. URL https://arxiv.org/abs/2404.02524.

Ivanovic, B. and Pavone, M. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. *International Conference on Computer Vision (ICCV)*, pp. 2375–2384, 2019. doi: 10.1109/ICCV.2019.00246.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., Fernando, C., and Kavukcuoglu, K. Population based training of neural networks, 2017. URL http://arxiv.org/abs/1711.09846.

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castañeda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., Sonnerat, N., Green, T., Deason, L., Leibo, J. Z., Silver, D., Hassabis, D., Kavukcuoglu, K., and Graepel, T. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019. doi: 10.1126/science.aau6249.

Jaeger, B., Chitta, K., and Geiger, A. Hidden biases of end-to-end driving models. In *International Conference on Computer Vision (ICCV)*, 2023.

Jain, A., Pero, L. D., Grimmett, H., and Ondruska, P. Autonomy 2.0: Why is self-driving always 5 years away?, 2021. URL https://arxiv.org/abs/2107.08142.

Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., and Scaramuzza, D. Champion-level drone racing using deep reinforcement learning. *Nature*, 620 (7976):982–987, 2023.

Kesting, A., Treiber, M., and Helbing, D. General lane-changing model mobil for car-following models. *Transportation Research Record*, 1999(1):86–94, 2007.

Kesting, A., Treiber, M., and Helbing, D. Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity. *Philosophical Transactions of the Royal Society A*, 2010.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., and Hutter, M. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):5986, 2020.

Li, Q., Peng, Z., Feng, L., Zhang, Q., Xue, Z., and Zhou, B. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3461–3475, 2023. doi: 10.1109/TPAMI.2022.3190471.

Lu, Y., Fu, J., Tucker, G., Pan, X., Bronstein, E., Roelofs, R., Sapp, B., White, B., Faust, A., Whiteson, S., Anguelov, D., and Levine, S. Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios. *International Conference on Intelligent Robots and Systems (IROS)*, pp. 7553–7560, 2023. doi: 10.1109/IROS55552.2023.10342038.

Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., and State, G. Isaac gym: High performance GPU based physics simulation for robot learning. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

Montali, N., Lambert, J., Mougin, P., Kuefler, A., Rhinehart, N., Li, M., Gulino, C., Emrich, T., Yang, Z. Z., Whiteson, S., White, B., and Anguelov, D. The Waymo open sim agents challenge. *Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Müller, M., Dosovitskiy, A., Ghanem, B., and Koltun, V. Driving policy transfer via modularity and abstraction. In *Conference on Robot Learning (CoRL)*, pp. 1–15. PMLR, 2018.

Najm, W., Smith, J. D., and Yanagisawa, M. Pre-crash scenario typology for crash avoidance research. Technical report, John A. Volpe National Transportation Systems Center (U.S.), 2007. URL https://rosap.ntl.bts.gov/view/dot/6281.

Nayakanti, N., Al-Rfou, R., Zhou, A., Goel, K., Refaat, K. S., and Sapp, B. Wayformer: Motion forecasting via simple & efficient attention networks. In *International Conference on Robotics and Automation (ICRA)*, pp. 2980–2987, 2023.

Ost, J., Mannan, F., Thuerey, N., Knodt, J., and Heide, F. Neural scene graphs for dynamic scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2856–2865, June 2021.

Perolat, J., Vylder, B. D., Hennes, D., Tarassov, E., Strub, F., de Boer, V., Muller, P., Connor, J. T., Burch, N., Anthony, T., McAleer, S., Elie, R., Cen, S. H., Wang, Z., Gruslys, A., Malysheva, A., Khan, M., Ozair, S., Timbers, F., Pohlen, T., Eccles, T., Rowland, M., Lanctot, M., Lespiau, J.-B., Piot, B., Omidshafiei, S., Lockhart, E., Sifre, L., Beauguerlange, N., Munos, R., Silver, D., Singh, S., Hassabis, D., and Tuyls, K. Mastering the game of Stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022. doi: 10.1126/science.add4679.

Petrenko, A., Wijmans, E., Shacklett, B., and Koltun, V. Megaverse: Simulating embodied agents at one million experiences per second. In *International Conference on Machine Learning*, 2021.

Petrenko, A., Allshire, A., State, G., Handa, A., and Makoviychuk, V. PDexPBT: Scaling up dexterous manipulation for hand-arm systems with population based training. In *Robotics: Science and Systems*, 2023.

Philion, J., Peng, X. B., and Fidler, S. Trajeglish: Learning the language of driving scenarios. In *International Conference on Learning Representations (ICLR)*, 2023.

Plappert, M., Sampedro, R., Xu, T., Akkaya, I., Kosaraju, V., Welinder, P., D'Sa, R., Petron, A., de Oliveira Pinto, H. P., Paino, A., Noh, H., Weng, L., Yuan, Q., Chu, C., and Zaremba, W. Asymmetric self-play for automatic goal discovery in robotic manipulation. *arXiv:2101.04882*, 2021.

Qian, C., Xiu, D., and Tian, M. A simple yet effective method for simulating realistic multi-agent behaviors. Technical report, 2023. URL https://storage.googleapis.com/waymo-uploads/files/research/2023%20Technical%20Reports/SA_2nd_MTR%2B%2B%2B%20(1).pdf.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 2021.

Renz, K., Chitta, K., Mercea, O.-B., Koepke, A. S., Akata, Z., and Geiger, A. PlanT: Explainable planning transformers via object-level representations. In *Conference on Robot Learning (CoRL)*, 2022.

Rudin, N., Hoeller, D., Reist, P., and Hutter, M. Learning to walk in minutes using massively parallel deep reinforcement learning. In Faust, A., Hsu, D., and Neumann, G. (eds.), *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pp. 91–100. PMLR, 2021. URL https://proceedings.mlr.press/v164/rudin22a.html.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

Scheel, O., Bergamini, L., Wolczyk, M., Osinski, B., and Ondruska, P. Urban driver: Learning to drive from real-world demonstrations using policy gradients. In *Conference on Robot Learning (CoRL)*, 2021.

Schulman, J., Moritz, P., Levine, S., Jordan, M. I., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017. URL http://arxiv.org/abs/1707.06347.

Shacklett, B., Wijmans, E., Petrenko, A., Savva, M., Batra, D., Koltun, V., and Fatahalian, K. Large batch simulation for deep reinforcement learning. In *International Conference on Learning Representations*, 2021.

Shacklett, B., Rosenzweig, L. G., Xie, Z., Sarkar, B., Szot, A., Wijmans, E., Koltun, V., Batra, D., and Fatahalian, K. An extensible, data-oriented architecture for high-performance, many-world simulation. *ACM Trans. Graph.*, 42(4), 2023.

Shi, S., Jiang, L., Dai, D., and Schiele, B. Motion transformer with global intention localization and local movement refinement. *Neural Information Processing Systems*, pp. 6531–6543, 2022.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T. P., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017. doi: 10.1038/NATURE24270.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362 (6419):1140–1144, 2018. doi: 10.1126/science.aar6404.

Stewart, T. Overview of motor vehicle traffic crashes in 2021. Report No. DOT HS 813 435, 2023. URL https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813435. National Highway Traffic Safety Administration.

Suo, S., Regalado, S., Casas, S., and Urtasun, R. TrafficSim: Learning to simulate realistic multi-agent behaviors. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10400–10409, 2021.

Tao, Y., Genc, S., Chung, J., Sun, T., and Mallya, S. Repaint: Knowledge transfer in deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 10141–10152, 2021.

Treiber, M., Hennecke, A., and Helbing, D. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805–1824, August 2000. ISSN 1095-3787. doi: 10.1103/physreve.62.1805.

Vinitsky, E., Lichtlé, N., Yang, X., Amos, B., and Foerster, J. Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world. *Neural Information Processing Systems*, 35, 2022.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gülçehre, Ç., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T. P., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354, 2019. doi: 10.1038/S41586-019-1724-Z.

Wang, W. and Zhen, H. Joint-multipath++ for simulation agents. Technical report, 2023. URL https://storage.googleapis.com/waymo-uploads/files/research/2023%20Technical%20Reports/SA_hm_jointMP.pdf.

Wang, Y., Tang, H., Huang, L., Pan, L., Yang, L., Yang, H., Mu, F., and Yang, M. Self-play reinforcement learning guides protein engineering. *Nature Machine Intelligence*, (5):845–860, 2023a.

Wang, Y., Zhao, T., and Yi, F. Multiverse transformer: 1st place solution for Waymo open sim agents challenge 2023, 2023b. URL https://arxiv.org/abs/2306.11868.

Wijmans, E., Kadian, A., Morcos, A., Lee, S., Essa, I., Parikh, D., Savva, M., and Batra, D. DD-PPO: learning near-perfect pointgoal navigators from 2.5 billion frames. In *International Conference on Learning Representations ICLR*, 2020.

Xu, D., Chen, Y., Ivanovic, B., and Pavone, M. Bits: Bi-level imitation for traffic simulation. In *International Conference on Robotics and Automation (ICRA)*, pp. 2929–2936, 2023.

Yang, B., Su, H., Gkanatsios, N., Ke, T.-W., Jain, A., Schneider, J., and Fragkiadaki, K. Diffusion-ES: Gradient-free planning with diffusion for autonomous driving and zero-shot instruction following, 2024. URL https://arxiv.org/abs/2402.06559.

Yang, Z., Chen, Y., Wang, J., Manivasagam, S., Ma, W.-C., Yang, A. J., and Urtasun, R. UniSim: A neural closed-loop sensor simulator. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1389–1399, June 2023.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. J. Deep sets. In *Neural Information Processing Systems*, 2017.

Zhang, C., Tu, J., Zhang, L., Wong, K., Suo, S., and Urtasun, R. Learning realistic traffic agents in closed-loop. In *Conference on Robot Learning (CoRL)*, 2023.

Zhong, Z., Rempe, D., Xu, D., Chen, Y., Veer, S., Che, T., Ray, B., and Pavone, M. Guided conditional diffusion for controllable traffic simulation. In *International Conference on Robotics and Automation (ICRA)*, pp. 3560–3566, 2023.

A. Simulator Design

GIGAFLOW is a *batched* simulator (Makoviychuk et al., 2021; Freeman et al., 2021; Petrenko et al., 2021; Shacklett et al., 2021), implemented in PyTorch (Ansel et al., 2024) and designed for GPU acceleration. A single instance of GIGAFLOW simulates thousands of worlds, enabling it to leverage the parallelism of modern GPUs. Implementing GIGAFLOW efficiently required the development of custom batched operators for the kinematics, collision checking, initialization of urban driving environments, and many other features. Below we outline the overall flow of GIGAFLOW and describe the acceleration techniques used.

At every timestep, GIGAFLOW takes the current state of the i-th world $s_i^{(t)}$ and vehicle controls $a_i^{(t)}$ to produce the state at the next timestep $s_i^{(t+1)}$. Instead of running multiple copies of the environment to simulate $N = 38\,400$ worlds, a single instance of GIGAFLOW simulates N worlds in parallel. The state and vehicle controls of *all* N worlds are denoted $\mathbf{s}^{(t)}$ and $\mathbf{a}^{(t)}$.

A.1. World initialization

The simulation process begins with the initialization of urban driving environments by placing up to $N_a = 150$ vehicles at random positions on the map. We first draw a random sample over map locations, vehicle headings, and vehicle bounding box dimensions and reject states that are off-road (off-road checking is detailed below). A naive application of this process results in a marginal distribution of vehicle locations that is biased towards wider road sections. To correct for this, we estimate this marginal distribution from an initial sample and use it to adjust the proposal distribution used in subsequent rejection sampling. Given the set of valid vehicle states, we then select a collision-free subset with the desired number of agents (collision detection is detailed below). This subset becomes the initial traffic configuration at $t = 0$.

For each retained vehicle, we select a sequence of its waypoints (goals). The first waypoint is sampled uniformly over the map and additional waypoints are sampled such that given the jth waypoint, the $(j + 1)$th waypoint is at least $20\,\mathrm{m}$ away and no more than $200\,\mathrm{m}$ away, and has lane heading that is within 60 degrees of the jth waypoint's lane heading. There are cases where the jth waypoint is in a location where these constraints cannot be met (e.g., a dead end). In these cases, we gradually relax the constraints as we try to sample points that fit them. The intent of this sampling procedure is to generate waypoint sequences that resemble realistic driving routes where intermediate destinations are reached in a natural succession.

We use two acceleration techniques specific to initialization. First, we draw a large buffer of vehicle states and goals all at once and then consume that buffer until it is empty as new initializations are requested, e.g., when episodes reset. This allows for better cost amortization when generating initial states. Second, we use sequential rejection sampling to produce collision-free initializations. We add one agent to the world, then add a second agent that is not in collision with existing agents (via rejection sampling), and so on until we reach the desired number of agents. This approach was necessary because the probability that an independent sample of over one hundred vehicles contains no collisions is extremely low for small maps.

A.2. Dynamics model update

The dynamics model (described below) produces $\mathbf{s}^{(t+1)}$ given $\mathbf{s}^{(t)}$ and $\mathbf{a}^{(t)}$. This is a set of element wise operations (trigonometric functions, multiplications, divisions, *etc.*) that are parallelized on a GPU using their respective PyTorch implementations.

A.3. Road localization

The next step is to localize $\mathbf{s}^{(t+1)}$ on the road surface.

Road representation. GIGAFLOW represents the road surface as a set of potentially overlapping polygons, exclusively using convex quadrilaterals. We find quadrilaterals to offer a good balance between expressivity and the simplicity of operations. A given lane on the road surface is approximated by quadrilaterals that have the same width as the lane and are $1\,\mathrm{m}$ in length. We find this resolution of polygons to be a good trade-off between the accuracy of road geometry approximation and the total number of primitives.

It should be noted that the number of geometric primitives could be minimized further by merging polygons in the regions with simple geometry, such as straight road segments. However, we retain the uniform polygon density irrespective of the geometric complexity because this polygonal subdivision additionally serves as a spatial hash map for certain types of queries (e.g., map observations are pre-computed for all polygon center points).

Frenet coordinates. Let q represent the distance along a lane, d be the distance from lane center, and `polyId` be a unique identifier of the polygon approximating the road geometry at the current location. The Frenet coordinate for position (x, y) is then $(q, d, \texttt{polyId})$.

We convert world-frame state $\mathbf{s}^{(t)}$ to Frenet-frame state $\mathbf{f}^{(t)}$ as this information is useful for constructing actor observations and covers off-road checking in the majority of cases. We construct the Frenet-frame state by first finding the polygon that contains (x, y). Given this polygon, we can compute (q, d) by transforming (x, y) into the coordinate frame defined by the polygon's heading and center point then adding the distance from the start of the lane to the

polygon's center to q. If multiple polygons contain (x, y), we use the polygon where the agent is most lane centered (where d is the smallest).

Naively implementing this would require performing $O(N \times A \times P)$ point-in-polygon checks, where N is the number of worlds, A is the number of agents, and P is the number of polygons. This is prohibitively computationally expensive.

Spatial hashing. We accelerate this via spatial hashing on the GPU. Specifically, we first construct a 2-D grid of non-overlapping axis-aligned boxes with a fixed width and height. This scheme allows us to trivially map a point (x, y) to its given bucket in the hash map. We then assign each polygon in the road representation to the (potentially multiple) buckets that it overlaps. Note that this step only needs to be performed once at startup.

We localize (x, y) by first retrieving the polygons that are in its bucket and then performing a point-in-polygon check for each matching quadrilateral. One challenge of implementing this efficiently is that each hash bucket has a different number of corresponding polygons. We maintain efficiency by designing APIs to expect these "ragged" variable-size tensors and only perform padding on operations that require it (e.g., finding the polygon that minimizes d).

This spatial hash additionally allows for the support of multiple maps via augmentation of vehicle coordinates (x, y) with the map Id: $(x, y, \texttt{mapId})$. Essentially, the map an agent is driving on functions as a "third" dimension, allowing us to filter out polygons from other maps at the hash bucket retrieval step.

A.4. Off-road checking

Given $\mathbf{s}^{(t)}$, we localize the vehicle's bounding box center and corners on the road surface using the spatial hash procedure described above. In the majority of cases, a vehicle is off-road if any of these 5 points could not be localized onto the road surface. However, there are two edge cases that are important to handle.

Curved roads and islands. These are various situations where all 5 of these points can be on the road surface but the vehicle should still be considered off-road. Two such examples are curved roads (where part of the bounding box overhangs the edge of the road) and pedestrian safety islands (where the vehicle can straddle the islands). To handle both these cases, we generate a set of out-of-bounds (OOB) points by taking the mid-point of each polygon edge, nudging it slightly in the outwards direction, and keeping all points that do not lie within any other polygon. A vehicle is classified as off-road if any of these OOB points are found to be within its bounding box.

We again use our spatial hashing data structure to accelerate the check between vehicle bounding box and the OOB points. Specifically, each OOB point is first associated with its hash bucket. Then, we lookup which buckets the vehicle overlaps with and perform a point-in-bounding-box check with that set of points.

We associate the vehicle bounding box with appropriate hash buckets as follows: first, we set the bucket size to be $2\times$ the maximum possible vehicle length. Then, we approximate the vehicle's oriented bounding box by its axis-aligned bounding box (AABB). Due to our choice of spatial hash bucket size, the AABB overlaps with at most 4 buckets, each containing (at least) one of the AABB's corners. We retrieve the OOB points that correspond to ≤ 4 buckets overlapped by the vehicle and perform a point-in-bounding-box check with each. We find that this fairly simple approximation is appropriate since it enables very fast lookup which is crucial as it is performed at every simulation step.

Gaps in the road surface. Due to the extremely low off-road rates achieved by GIGAFLOW policies, we found that most remaining off-road events are characterized by thin gaps in the road surface (sometimes under $1\,\mathrm{mm}$), often located between neighboring lanes. These thin gaps are not actual geometrical features but numerical inaccuracies in the underlying map files.

During training, we allow any of the 5 points used for off-road checking to be in one of the spurious gaps by measuring the distance from the point to the closest road polygon and considering points within $\delta = 15\,\mathrm{cm}$ to be still on the road surface. We again use spatial hashing to accelerate this lookup. We take edges of all road polygons and assign them to any bucket that they overlap with or pass within the distance δ of, so we can quickly obtain a list of candidate edges.

A.5. Collision detection

Given $\mathbf{s}^{(t)}$ and $\mathbf{s}^{(t+1)}$, we detect collisions as follows. Let $s_{a_i}^{(t)}$ and $s_{a_i}^{(t+1)}$ be successive states of agent i, and $s_{a_j}^{(t)}$ and $s_{a_j}^{(t+1)}$ be the successive states of agent j. We perform two checks to see if a collision occurred. First we transform $s_{a_j}^{(t)}$ into the coordinate frame of $s_{a_i}^{(t)}$ and $s_{a_j}^{(t+1)}$ into the coordinate frame of $s_{a_i}^{(t+1)}$. Then we check to see if any of the lines defined by the movement of agent j's corners intersect with agent i's bounding box (the bounding box is centered at the origin). We then swap the roles of agents i and j and perform this check again. A collision occurred between agents i and j if either check is positive. Note that we only perform collision detection, not collision simulation.

We accelerate this using our spatial hash by constructing, for all agents, the axis-aligned bounding box (AABB) that

contains the vehicle's bounding box at *both* states $s^{(t)}$ and $s^{(t+1)}$. We assign the vehicle to all buckets that overlap with this bounding box and perform the collision detection procedure described above for all pairs of vehicles that share at least one bucket. We use the world ID of each agent as an additional dimension in the spatial hash so that vehicles can only collide with other vehicles in the same world. This hashing method is fast to compute and greatly reduces the number of candidate collision pairs. Therefore hashing represents a large improvement over a naive pairwise check.

The check described above only works when (at least) one vehicle is in motion. This is acceptable during simulation, as at least some movement is required to transition from collision-free to a colliding state. For detecting collisions at $t = 0$, we simply check to see if the two bounding boxes overlap at state $s^{(0)}$. We use the spatial hash in a similar manner as before, except the AABB corresponds to a single (initial) state.

A.6. 2.5-D simulation

Simulating driving can be largely approximated without errors as a two-dimensional problem. This approximation enables performance improvements and reduces code complexity (thereby limiting the surface area for coding errors). However, certain cases cannot be accurately approximated in two dimensions, like overpasses. GIGAFLOW handles this with "2.5-D" simulation. We simulate the world as if it were 2-D and then correct for these errors. For example, we perform collision detection in two dimensions and then remove collisions that would not have occurred in 3-D. We maintain the vehicle's z-coordinate by applying the dynamics model purely in 2-D and then looking up new $\mathbf{z}^{(t+1)}$ for all vehicles from the map that correspond to new locations $\mathbf{s}^{(t+1)}$. We use $\mathbf{z}^{(t)}$, $\mathbf{s}^{(t)}$, and the maximum slope of a given map to filter out conflicting z values in cases like overpasses.

A.7. Hardening

Given the extremely low collision and off-road rates seen in GIGAFLOW training we found that extremely rare bugs would dominate the collisions and off-road events when present. We iterated extensively; training numerous policies to very high fidelity, watching videos of collisions and off-road events, and sieving those caused by coding errors and numerical inaccuracies rather than the agent's poor decision making. Each iteration would yield new bug fixes, improving both training and downstream benchmark performance.

To address these bugs, we built numerous visualization tools, as merely knowing that a rare bug existed was not enough to fix it — we had to find the exact steps to reproduce it as well. The visualization, diagnostic, and recording tools we developed were instrumental to the overall success of the project.

We additionally developed multiple simplified versions of GIGAFLOW to speed up diagnostics and troubleshooting. Using the minimal single-agent configuration we could train policies to convergence in under 20 minutes, allowing us to get immediate feedback on the core functionality. We established infrastructure for continuous testing, repeatedly retraining policies from scratch using these streamlined simulator configurations. This system has significantly accelerated the identification and isolation of regressions, thereby conserving numerous development cycles.

B. Defining the partially observed stochastic game

We model the environment that our agents learn in as a *partially observed stochastic game* (POSG) (Hansen et al., 2004); an extension of POMDPs to the multi-agent setting in which there are multiple agents with conflicting goals. We define each of the components of the POSG below.

B.1. Observations

We render the world state $s^{(t)}$ into a relatively low dimensional vector representation. In order to drive safely, a GIGAFLOW agent utilizes information about vehicle's dynamics and its position w.r.t. the lane $S^{(t)}$, approximate map of the surrounding area, including roads and traffic lights $\left(W^{(t)}_{\text{lane}}, W^{(t)}_{\text{boundary}}, W^{(t)}_{\text{stop}}\right)$, observations of other traffic participants $A^{(t)}$, desired destination and intermediate waypoints $G^{(t)}$, as well as the agent's conditioning C_{reward} (Table A2) .

Local observation $S^{(t)}$ can be further broken down as follows (time indices omitted for clarity):

- c, θ: the distance from the current lane center and the angle relative to lane heading.
- κ: local road curvature.
- v: current speed of the vehicle.
- v_{lim}: maximum allowed speed.
- ϕ: current steering angle.
- $a_{\text{long}}, a_{\text{lat}}$: current longitudinal and lateral acceleration.
- Driver's acceleration limits C_{acc}.
- C_{throttle}, C_{steer}: randomized coefficients determining the vehicle's responsiveness to throttle and steering inputs.
- l, w: driver's vehicle's length and width.

Note that all of these observations are normalized to $[-1, 1]$ range and provided in egocentric frame.

Map observations outline the surrounding area at two levels of resolution, including the high-level overview of the nearby road network $W^{(t)}_{\text{lane}}$ and a more detailed set of features $W^{(t)}_{\text{boundary}}$ that represent the precise shape of the local driveable area.

The coarse map view $W^{(t)}_{\text{lane}}$ provides a set of positional features sampled along driveable lanes at $40\,\text{m}$ intervals, containing information about lane widths and heading directions. These features also convey higher-level routing information similar to that provided by GPS-based navigation applications. Each observation within $W^{(t)}_{\text{lane}}$ is augmented with relative (w.r.t. other observations) and absolute normalized distances to the next goal, which allows GIGAFLOW agents to make informed routing decisions at complex junctions. We pre-calculate all routing information by running the Dijsktra's algorithm (Dijkstra, 1959) offline, computing pairwise distances between all locations of interest.

For a more fine-grained road geometry representation $W^{(t)}_{\text{boundary}}$ we provide midpoints of nearby polygon edges spaced roughly $1\,\text{m}$ apart which delineate the out-of-bounds areas closest to the driver (Fig. 2c). For both $W^{(t)}_{\text{lane}}$ and $W^{(t)}_{\text{boundary}}$ we provide the 80 features closest to the driver, additionally limited to $200\,\text{m}$ viewing horizon for the coarse map. We pre-calculate W_{lane} and W_{boundary} for each map polygon which enables extremely fast retrieval at runtime, only requiring a simple conversion to driver's egocentric frame to obtain $W^{(t)}_{\text{lane}}$ and $W^{(t)}_{\text{boundary}}$.

For observations of other agents we return the N_o nearest agents within $\delta_{\text{max}} = 200\,\text{m}$ of the driver. We pad the array of observations in case there are fewer than N_o nearby actors. For each agent, we observe its position, orientation, velocity, dimensions (i.e. width and length), and z-coordinate, all transformed into driver's egocentric frame. The goals, momentary accelerations, dynamics properties, or conditioning parameters of other agents are not observed.

While we do not explicitly model occlusions, we keep the maximum number of observed agents during training relatively low: $N_o = 20 \ll N_a$. Together with the distance-based visibility threshold δ_{max} this allows us to train the agent well adjusted to limited observability. Our permutation-invariant model architecture allows the value N_o to be trivially increased at test time to use all available information (see the model architecture details below).

B.2. Actions and dynamics

Our agents use a discrete set of actions to control the vehicle's change in acceleration using a jerk-actuated bicycle dynamics model. The action space includes 12 total actions, the Cartesian product of the sets of available values of longitudinal $\dot{a}_{\text{long}} \in \{-15, -4, 0, 4\}\,\text{m}\,\text{s}^{-3}$ and lateral jerk $\dot{a}_{\text{lat}} \in \{-4, 0, 4\}\,\text{m}\,\text{s}^{-3}$.

We compute longitudinal and lateral accelerations using numerical integration ($\Delta t = 0.3\,\text{s}$ during training):

$$a^{(t)}_{\text{long}} = a^{(t-1)}_{\text{long}} + C_{\text{throttle}}\,\dot{a}_{\text{long}}\,\Delta t \tag{1}$$

$$a^{(t)}_{\text{lat}} = a^{(t-1)}_{\text{lat}} + C_{\text{steer}}\,\dot{a}_{\text{lat}}\,\Delta t \tag{2}$$

where coefficients $C_{\text{throttle}}, C_{\text{steer}} \in C_{\text{dynamics}}$ are sampled from a mixed uniform distribution $X(1.25)$, defined as:

$$X(a) = 0.5\,U(a^{-1}, 1) + 0.5\,U(1, a), \quad a > 1$$

This distribution generates an equal number of samples smaller and greater than one, thereby allowing for a balanced randomization of dynamics properties.

We apply a small modification to the Eqs. (1) and (2), setting values $a^{(t)}_{\text{long}}, a^{(t)}_{\text{lat}}$ to exactly 0 when acceleration changes sign (i.e. when $a^{(t-1)}_{\text{long}}\,a^{(t)}_{\text{long}} < 0$). We found that this modification makes it easier for the agent to wait in place or drive at a constant velocity, producing smoother trajectories.

The acceleration components are then clipped ensuring the g-forces stay within the specified limits (here $C_{\text{acc}} \sim X(1.5)$):

$$a^{(t)}_{\text{long}} \leftarrow \texttt{clip}\left(a^{(t)}_{\text{long}},\, -5,\, 2.5\,C_{\text{acc}}\right)$$

$$a^{(t)}_{\text{lat}} \leftarrow \texttt{clip}\left(a^{(t)}_{\text{lat}},\, -4,\, 4\right)$$

We update the velocity magnitude using the trapezoidal rule (averaging previous and current accelerations):

$$v^{(t)} = v^{(t-1)} + 0.5\,\left(a^{(t)}_{\text{long}} + a^{(t-1)}_{\text{long}}\right)\Delta t$$

Just as for the accelerations, we set $v^{(t)}$ to exactly 0 when its value changes sign. We then clip $v^{(t)}$ to stay within the randomized speed limit ($C_{\text{vel}} \sim X(1.5)$):

$$v^{(t)} \leftarrow \texttt{clip}\left(v^{(t)},\, -2,\, 20\,C_{\text{vel}}\right)$$

To reach the acceleration $a^{(t)}_{\text{lat}}$, the vehicle would have to follow the arc with radius $|\rho|$ and signed curvature ρ^{-1} applying the steering angle ϕ (here l_{wb} is the vehicle's wheelbase and $\epsilon = 10^{-5}$ ensures numerical stability):

$$\rho^{-1} = \frac{a_{\text{lat}}}{\max(v^2, \epsilon)}$$

$$\rho^{-1} \leftarrow \text{sign}\left(\rho^{-1}\right)\,\max\left(\left|\rho^{-1}\right|, \epsilon\right)$$

$$\phi = \arctan\left(\rho^{-1}\,l_{\text{wb}}\right)$$

We calculate the change in the steering angle, δ_ϕ, and update the effective steering angle, $\phi^{(t)}$, ensuring that both values remain within conservatively defined limits ($\delta_{\text{max}} = 0.6\,\text{rad}\,\text{s}^{-1}$, $\phi_{\text{max}} = 0.55\,\text{rad}$):

$$\delta_\phi = \texttt{clip}(\phi - \phi^{(t-1)},\, -\delta_{\text{max}}\,\Delta t,\, \delta_{\text{max}}\,\Delta t) \quad (3)$$

$$\phi^{(t)} = \texttt{clip}(\phi^{(t-1)} + \delta_\phi,\, -\phi_{\text{max}},\, \phi_{\text{max}}) \quad (4)$$

The clipping in Eqs. (3) and (4) prevent excessive changes in the steering angle, limiting the maximum lateral acceleration at low speed. To account for the limited steering actuation, we update the effective signed curvature ρ^{-1} and acceleration $a_{\text{lat}}^{(t)}$ accordingly:

$$\rho^{-1} \leftarrow \frac{\tan\left(\phi^{(t)}\right)}{l_{\text{wb}}}$$
$$a_{\text{lat}}^{(t)} \leftarrow \left(v^{(t)}\right)^2 \rho^{-1}$$

Finally, the resulting movement of the vehicle is updated using the bicycle dynamics model:

$$d = 0.5\,\left(v^{(t)} + v^{(t-1)}\right)\Delta t$$
$$\theta = d\,\rho^{-1}$$
$$\Delta x = \rho \sin(\theta)$$
$$\Delta y = \rho \cos(\theta)$$

where d is the displacement along the arc and θ is the angular displacement.

B.3. Reward

Our agents optimize a scalar reward function consisting of multiple terms weighted by their respective coefficients $\alpha_{(\cdot)}$ which are randomly sampled at the beginning of each episode (see Table A2). This reward function R is defined by:

$$R = R_{\text{goal}} + R_{\text{collision}} + R_{\text{off-road}} + R_{\text{comfort}} + R_{\text{lane}} + R_{\text{velocity}} + R_{\text{reverse}} + R_{\text{stop-line}} + R_{\text{timestep}}$$

where the individual terms can be described as follows:

- R_{goal} rewards the agent for reaching its intermediate goals (waypoints) and the final goal.
- $R_{\text{collision}}$ penalizes agents for colliding with vehicles or pedestrians with an additional penalty for colliding at high speed. Randomizing $\alpha_{\text{collision}}$ allows us to populate the training environments with agents of varying risk tolerance, from very aggressive to very conservative.
- $R_{\text{off-road}}$ penalizes agents for leaving the road.
- R_{comfort} is a penalty for exceeding the comfortable limits of acceleration and jerk. Randomized α_{comfort} creates a distribution of driving styles, from relatively smooth to practically unconcerned with comfort.
- $R_{\text{l-align}}$ incorporates the preferences for driving in the designated driving direction and staying **parallel** to road lanes. We randomize this term in a wide range, which occasionally exposes our agent to erratic actors driving against traffic.
- $R_{\text{l-center}}$ rewards the agent for staying **centered** in the lane. We randomize $\alpha_{\text{center-bias}}$ for additional behavioral diversity.
- R_{velocity} encourages forward progress and motivates the agent to prefer routes with consistent traffic flow over traffic jams. We found that this term accelerates convergence in the early stages of training and helps us train policies that are better at avoiding gridlocks in self-play.
- R_{reverse} penalizes the agents for reversing. Due to the randomization of α_{reverse} some agents perform multi-point turns to reach the goals behind them, while others prefer to drive forward and wrap around the block or perform a U-turn instead.
- $R_{\text{stop-line}}$ penalizes the agent for crossing a stop line at a red light.
- R_{timestep} is a small penalty applied at each simulation step. This serves a function similar to the discount γ, except we disable R_{timestep} when ego is stationary which creates agents more willing to patiently wait at traffic lights and intersections when necessary.

B.4. Additional randomized components

In addition to the reward function coefficients we randomize a number of simulator's components to generate diverse embodiments and behavioral patterns. Agents have access to the randomized parameters as a part of their conditioning.

Vehicle size. We randomize vehicle's dimensions as follows (measured in meters):

- Vehicle length and wheelbase: $l \sim U(0.8, 7)$, $l_{\text{wb}} = 0.6\,l$.
- Vehicle width: $w \sim U(0.8, 3)$.
- The vehicle width is clipped: $w \leftarrow \min(w, l)$.

Size randomization allows us to generate a variety of road users, from very maneuverable agents with a footprint of a pedestrian to relatively big vehicles. We represent occasional bigger vehicles at test time using a set of smaller

bounding boxes (e.g., a semi-truck with a trailer can be split into two bounding boxes).

Goals. A final goal is sampled for each agent. Additionally we sample $N_{\text{wp}} \sim U\{0, 3\}$ intermediate waypoints (see "World Initialization" for details). We randomize the goal collection radius δ_{goal} to control how precisely our driver is required to adhere to the designated route (see Table A2).

Vehicle dynamics. For each agent in each episode we randomly sample maximum permitted acceleration, velocity, throttle and steering coefficients (refer to "Actions and dynamics" for specific randomization ranges).

Traffic lights. Real driving situations include a diverse set of traffic light systems, from synchronized lights at 3-way and 4-way intersections to occasional faulty, disabled, or miscalibrated signals. Instead of explicitly modeling this distribution in GIGAFLOW we simply pick random durations of green, yellow, and red signals for each traffic light independently. We start with default signal durations used in CARLA (typically $\hat{\tau}_{\text{red}} = 2$, $\hat{\tau}_{\text{yellow}} = 3$, and $\hat{\tau}_{\text{green}} = 10$ seconds) and then randomize them for each episode within the following ranges:

- $\tau_{\text{red}} \sim U(0.15\,\hat{\tau}_{\text{red}}, 5.0\,\hat{\tau}_{\text{red}})$
- $\tau_{\text{yellow}} \sim U(0.5\,\hat{\tau}_{\text{yellow}}, 0.75\,\hat{\tau}_{\text{yellow}})$
- $\tau_{\text{green}} \sim U(0.1\,\hat{\tau}_{\text{green}}, \hat{\tau}_{\text{green}})$

We additionally remove 20% of individual lights, 20% of traffic light groups (e.g., all lights at an intersection), and in 20% of all episodes we disable traffic lights entirely, making all intersections unregulated. We also set 5% of all remaining traffic lights to be constantly green.

Even though we train only on 128 variants of CARLA maps (8 maps modified with affine transformations), this aggressive traffic light randomization allows us to combinatorially increase the total number of unique training environments and traffic patterns, preventing overfitting. Our policy learns to handle arbitrary traffic light configurations which helps significantly in benchmarks like nuPlan where traffic signal states are derived from the sense stack and often have errors.

The maximum red light duration in our simulator is deliberately limited ($\tau_{\text{red}} \leq 10\,\text{s}$). This allows us to mitigate the impatient nature of discounted reward maximization. When durations of required stops are short during training, agents with $\alpha_{\text{stop-line}} \gg 0$ are never incentivized to run red lights to maximize their return by getting to the goal faster. We find that our agents, due to their reactive nature, generalize to much longer red light durations occasionally encountered in benchmarks.

Modeling erratic drivers. We aim to train an attentive, rational, and defensive driver which does not require cooperation of other traffic participants to ensure safety. Ideally, even clearly erratic maneuvers by other vehicles should not lead to collisions or comfort violations on behalf of GIGAFLOW agent. In order to achieve this, we degrade a fraction of agents during training by introducing two types of modifications:

1. Up to 5% of agents occasionally do not see other vehicles. This models inattentive drivers and drivers with blind spots.
2. Up to 10% of agents sharply apply their brakes at arbitrary moments for a short duration before resuming normal driving. This models drivers that stop without warning or remain stationary when a traffic light turns green.

Between random applications of these modifications, agents are still controlled by GIGAFLOW policy and are completely oblivious to their modification, which makes it impossible for regular drivers to predict their erratic behavior. We exclude trajectories from these agents from the rollouts used for training.

C. Training algorithm

Agents are trained using a version of Proximal Policy Optimization (PPO) (Schulman et al., 2017) derived from the Stable Baselines codebase (Raffin et al., 2021). PPO trains a policy that outputs actions, and a critic that estimates discounted returns. In our implementation, we do not share any parameters between the policy and the critic, which empirically produces comparatively more robust low entropy policies at convergence. Additionally, we add the terminal value estimate to reward at the end of all truncated episodes, thus emulating infinite-horizon learning, similar to "Reset Handling" in Rudin et al. (2021). We use Adam optimizer (Kingma & Ba, 2015) with annealed cosine-rate learning schedule:

$$\alpha^{(k)} = \frac{\alpha^{(0)}}{2}\left[1 - \cos\left(\pi - \frac{\pi\,k}{K}\right)\right]$$

where k is the current training iteration and K is the maximum number of iterations. We use in DD-PPO (Wijmans et al., 2020) to train with mutliple GPUs. Specifically, each GPU collects experience independently and then gradients are synchronized before the model update.

During development, we optimized a subset of hyperparameters of PPO and GIGAFLOW simulator using Population Based Training (PBT) (Jaderberg et al., 2017; Petrenko et al., 2023). We conducted this optimization using a simplified version of GIGAFLOW training setup (single map, reduced randomization) which enabled faster iteration. Our PBT experiments informed a number of non-trivial hyperparameter

Algorithm 1 Advantage filtering

Require: Initial model parameters Θ, RL environment $\mathcal{E}$, EWMA decay $\beta = 0.25$.

for $k = 0$ to $K - 1$ **do** ▷ Training iteration k

  $B_{\text{exp}} \leftarrow \text{CollectRollouts}(\Theta, \mathcal{E})$ ▷ Experience buffer

  $\hat{A}^{(\cdot)}_{\text{GAE}} \leftarrow \textbf{GAE}(B_{\text{exp}}, \Theta, \gamma, \lambda)$ ▷ GAE advantages

  $A_{\text{max}} \leftarrow \max_{t \in B_{\text{exp}}} \left|\hat{A}^{(t)}_{\text{GAE}}\right|$ ▷ Max abs. advantage

  $\bar{A}_{\text{max}} \leftarrow \mathbb{1}_{k=0} A_{\text{max}} + \mathbb{1}_{k>0} \left(\beta A_{\text{max}} + (1-\beta)\bar{A}_{\text{max}}\right)$

  Filtering threshold: $\eta \leftarrow 0.01\, \bar{A}_{\text{max}}$

  $B_{\text{filtered}} \leftarrow \text{Filter}(B_{\text{exp}}, \left|\hat{A}^{(\cdot)}_{\text{GAE}}\right| < \eta)$

  $\Theta \leftarrow \textbf{PPO}(B_{\text{filtered}}, \Theta)$

end for

choices, from the learning rate schedule to comparatively large integration step $\Delta t = 0.3\,\text{s}$. Table A3 provides our final list of training hyperparameters.

Advantage filtering. To make use of the vast scale of synthesized data, we implement a technique termed *advantage filtering* which can be viewed as a variant of Prioritized Experience Replay (Schaul et al., 2016) where the majority of transitions are sampled exactly **zero times**.

Specifically, for each transition we calculate the advantage estimate $\hat{A}^{(t)}_{\text{GAE}}$ (Schulman et al., 2016) and then simply discard all transitions that satisfy $\left|\hat{A}^{(t)}_{\text{GAE}}\right| < \eta$. The adaptive filtering threshold η is set to 1% of the moving average estimate of the maximum advantage magnitude at the current stage of training, thus rendering the method insensitive to the absolute scale of rewards (see Algorithm 1).

Empirically, we filter out on average $\sim 80\%$ of all samples throughout training (over 90% in the early epochs). This significantly increases learning throughput as we avoid computing gradients that contribute minimally to the overall parameter update. With the proposed adaptive threshold η we observe a 2.3-fold increase in training throughput, from 0.53 to 1.2 million steps per second. Our experiments suggest that application of this technique not only accelerates learning, but also yields more robust policies (e.g., see Carla LAV results in Fig. A2). We hypothesize that advantage filtering could be beneficial across various RL setups, especially where data collection is cheaper than gradient calculation (e.g., training LLMs on synthetic data).

Previously Tao et al. (2021) explored an experience filtering approach based on the fixed advantage threshold, selecting a subset of maximally informative samples from the teacher policy in the context of transfer learning. We further develop this idea and propose a general adaptive method compatible with a broad class of reinforcement learning applications.

D. Neural network architecture

Our actor and critic are parameterized by virtually identical compact neural networks with 3 million parameters each (6 million trainable parameters combined). At the high level, the architecture consists of a fully-connected backbone ($[1024 \times 1024 \times 1024]$ MLP) which receives concatenated feature embeddings and outputs either a distribution over actions or a scalar value estimate (see Fig. 2d).

Observations represented by simple feature vectors ($S^{(t)}$, $G^{(t)}$, C_{reward}, *etc.*) are trivially encoded by smaller fully-connected networks, e.g.,: $f_S^{(t)} = \texttt{MLP}\left(S^{(t)}\right)$.

Observations containing multiple features per agent $\left(W^{(t)}_{\text{lane}}, W^{(t)}_{\text{boundary}}, W^{(t)}_{\text{stop}}, A^{(t)}\right)$ are represented by sets of feature vectors and require permutation-invariant encoders (Zaheer et al., 2017). For each feature type we employ a small fully-connected network to encode each element in a set (e.g., an individual map feature in $W^{(t)}_{\text{lane}}$) followed by channel-wise maxpooling layer. Outputs of all maxpooling encoders are then concatenated.

Observation sets $W^{(t)}$ and $A^{(t)}$ represent some of the largest data structures in our computation graph containing over 10^8 individual feature vectors per inference step across all agents. Storing all of these observations in the PPO's rollout buffer would be prohibitively expensive. Instead, we store only the world states $\mathbf{s}^{(t)}$ and reconstruct most of the observations as needed for each training minibatch.

In addition to that, for the largest feature sets W_{lane} and W_{boundary} we omit a fraction of input features (akin to the dropout regularization technique). We randomly drop 40% of W_{boundary} and 50% of W_{lane} features during training for each agent. This *feature dropout* approach allows us to fit the full GIGAFLOW training system on 40 GB A100 GPUs, while additionally modelling potential sensor noise, increasing robustness of the policy and preventing overfitting.

E. Benchmark evaluation

For each benchmark, we use the benchmark-provided simulation infrastructure, translate observations into GIGAFLOW, and run GIGAFLOW as a co-simulator. For Waymax and CARLA, we co-simulate a single GIGAFLOW step for each evaluation step. nuPlan additionally requires trajectory predictions at each step. At each time-step, we roll out an entire scenario using GIGAFLOW policies for all agents for 0.3 seconds, and use the subsequent observed trajectory from the driver as its trajectory.

E.1. nuPlan benchmark evaluation

The nuPlan benchmark consists of a training, validation and held out test set built from thousands of hours of data col-

lected from Las Vegas, Boston, Pittsburgh, and Singapore. However, in our work we *never use data from the training set*. The benchmark provides three different challenges, an open-loop (OL) Challenge 1, closed-loop non-reactive (CL-NR) Challenge 2, and closed-loop reactive (CL-R) Challenge 3. We focus only on challenge 3 because it represents the most realistic setting with both the driver and the traffic vehicles being closed loop. The reactive traffic vehicle agents are controlled by the Intelligent Driver Model (IDM) (Kesting et al., 2010) with their initial placement taken from real traffic distributions. However pedestrians are non-reactive and follow their logged trajectory. After initialization, the IDM agent will select a series of lanes (independent of the log vehicles trajectory) and follow a centerline lane path provide by nuPlan's map. Longitudinal acceleration control comes from the IDM equation.

$$\frac{dv}{dt} = a\left(1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*}{s}\right)^2\right)$$

With acceleration limit a, desired speed v_0, current speed v, distance to lead agent s, safety margin s^*, and exponent δ is usually set to 4. The agent defaults to acceleration a unless it's close to desired v_0 or close to an object in its path. In this challenge, there is less diversity in behavior, but it has more realistic vehicle interactions when the driver deviates from the logged vehicle trajectory.

Due to the nuPlan online leaderboard evaluation servers no longer being accessible, we evaluate GIGAFLOW on the Val14 benchmark, which has been shown to be a good proxy for the leaderboard evaluation and has other publicly reported results (Dauner et al., 2023).

The GIGAFLOW policy achieves state of the art results zero-shot on the Challenge 3 closed loop score. The closed-loop score is measured from 0 to 100 and is a weighted evaluation composed of at-fault collisions, drive-able area compliance, driving direction compliance, progress towards goal, time-to-collision bounds, speed limit compliance, and comfort. Full results are shown in Table A5. We are able to compute the component scores for the PDM-Hybrid entrant and simply present the total score for the other entrants. Finally, we categorized all videos of nuPlan's collisions and we provide them in the supplementary material.

E.2. CARLA benchmark evaluation

Jaeger et al. (2023) implemented a CARLA expert driver that serves as a teacher for a trained perception-based model. To our best knowledge, the presented expert manifests the best scores on CARLA benchmarks ever reported in the literature. This expert makes use of privileged information such as exact positions of all traffic participants and perfect map information. Since the default CARLA leaderboard tracks do not provide such information, Jaeger et al. adapt the leaderboard code accordingly. We adopt this procedure and extract this information in a similar way. Moreover, we identified a bug in CARLA that can lead to faulty pedestrian states when a pedestrian collides with other traffic participants. This bug can result in erroneous collision infractions; see video in the supplementary material. We obviate this bug by removing a pedestrian actor from the scene in case of an event. To ensure the same test bed for all competing approaches, we run all of them in our setup and report the resulting scores along with the scores reported by the authors. In addition to the expert in Jaeger et al. (2023), we compare to the CARLA autopilot and another expert from Chitta et al. (2023). Since simulations in CARLA contain a large amount of randomness, Table A6 presents the mean and standard deviations of the evaluation results for 3 different runs.

E.3. Waymax benchmark evaluation

To run the Waymax simulator, we wrote a distributed scenario runner to evaluate on the full Waymo Open Motion Dataset (WOMD) 1.2.0 validation set consisting of $44\,097$ scenarios each $8\,\mathrm{s}$ long running at $10\,\mathrm{Hz}$. To match the results from Table 3 in Gulino et al. (2023), for our `DatasetConfig` we use `max_num_objects=128`, and use the default `IDMRoutePolicy` settings. We also applied this patch, `https://github.com/waymo-research/waymax/pull/54`, to fix IDM agent issues on top of commit `720f9214a` in the public Waymax Github repo. For goals used for the GIGAFLOW policy we use the final location of the expert logged trajectory as an intermediate goal, as well as two more goals projected out at $50\,\mathrm{m}$ and $100\,\mathrm{m}$ and snapped to the nearest lane center.

We use the provided agent overlap, off road rate, log divergence, and SDC kinematic infeasibility metrics provided by Waymax to calculate our results. However, due to the Waymax route information not being released at the time of this writing, we report a modified metric for `Route Progress Ratio`. We calculate it by dividing the distance the GIGAFLOW policy drove by the expert logged trajectory *per scenario*, and clamp to 100% if the expert drove less than $1\,\mathrm{m}$. We also report an additional progress metric, `Total Distance Driven Ratio`, which is simply the total distance the GIGAFLOW policy drove over all scenarios divided by the total the expert log drove.

Unlike Carla and nuPlan, Waymax does not have an aggregate score which allows for easier comparison and visualization of results. Because of this, we propose a simple aggregate score composed of a progress score (i.e. a clamped `Total Distance Driven Ratio`), collision rates, and off-road rates show in Eq. (5) below. This

metric is what is reported in our figures when referring to a single Waymax score.

$$\text{progress} = \min\left(\frac{\sum_{s\in\mathcal{S}} d_s}{\sum_{s\in\mathcal{S}} d_s^{\text{expert}}},\ 1\right)$$

$$\text{success} = 1 - \frac{\sum_{s\in\mathcal{S}} \mathbb{1}_s^{\text{collision}} \times \mathbb{1}_s^{\text{off-road}}}{|\mathcal{S}|}$$

$$\text{score} = \text{progress} \times \text{success} \tag{5}$$

Where $\mathcal{S}$ is the set of scenarios, d_s is the distance the policy under test drove in scenario s, d_s^{expert} is the distance the expert policy drove in scenario s, $\mathbb{1}_s^{\text{collision}}$ is the indicator function for if scenario s contained a collision, and $\mathbb{1}_s^{\text{off-road}}$ is the indicator function for if scenario s contained an off-road event.

Table A7 presents Waymax results.

E.4. Waymo Open Sim-Agents Challenge

Waymo introduced their Open Sim Agents Challenge (WOSAC) in 2023 to promote the development of intelligent, interactive simulation agents capable of exhibiting a diversity of behaviors in response to decisions made by the autonomous vehicle (Montali et al., 2023). In WOSAC, driving scenarios are characterized by 9.1 second sequences of recorded tracks of road users derived from the Waymo Open Motion Dataset (WOMD) (Ettinger et al., 2021), stored at 10 Hz, where the first 1.1 s of history is used to form an initial context. Models are evaluated by their ability to accurately reproduce the remaining 8 s of the scenario for up to 128 agents.

Despite being designed for imitative traffic modeling, we use WOSAC to evaluate GIGAFLOW in real-world multi-agent settings and demonstrate the policy's ability to control a diversity of road users while avoiding undesired behavior. Following the evaluation of other benchmarks, we evaluate a fully trained GIGAFLOW policy zero-shot on WOSAC without fine-tuning. We use the provided 1.1 s history to conditionally sample from the map distribution of possible goal locations, filtering out positions that are either unreachable or are trivially reachable. Proposed goal locations are retained proportional to the GIGAFLOW agent's value estimate from the provided initial position of each actor.

Behavior of all vehicles and bicycles is modeled using a GIGAFLOW policy, while pedestrians are controlled with an IDM-like policy based on the provided initial velocity and heading. If a collision with a vehicle is forecast within a 0.5 s horizon, the pedestrian actor stops moving forward until a collision is no longer imminent. To account for sensor noise in recorded data, we add random Gaussian noise to all generated actor positions. We achieved a 2-point improvement in the overall metametric using $\sigma = 0.0125$ when compared to rollouts without the added noise.

Results of our zero-shot evaluation of GIGAFLOW to WOSAC are reported in Table A8, using the Waymo-defined validation set to allow for detailed analysis in Table A9. In addition to GIGAFLOW we include a comparison to a policy where all actors are fixed to be stationary, establishing a lower bound on generated behaviors by the GIGAFLOW policy. We also include the reported performance for the top methods included on the public 2023 WOSAC leaderboard as well as a set of baseline approaches reported in Montali et al. (2023), nearly all based around supervised autoregressive modeling.

We see that the GIGAFLOW policy approaches the performance of expert models, despite having never been exposed to any map or human data recorded in WOMD during training. Notably, GIGAFLOW offers strong performance for collision and off-road metrics, out-performing most approaches. GIGAFLOW also maintains respectable performance among acceleration metrics, a consequence of producing kinematically feasbile and smooth rollouts following the dynamics model the policy was trained with. With GIGAFLOW, we are able to control multiple actors within real-world scenarios without colliding or driving off-road, producing naturalistic driving. The scoring metrics used for WOSAC are likelihood measures of the ground truth behavior within the distribution induced by generated actor trajectories. To achieve idealized performance for the likelihood of collision and off-road behavior, it is expected that the generated trajectories collide or drive off-road whenever ground truth actors are assessed to do so. Because GIGAFLOW explicitly penalizes these outcomes during training we cannot expect our policy to substantively improve over the reported scores in Table A8 for these metrics. We characterize the collision and off-road behavior of GIGAFLOW within WOSAC in Table A9, conditioned on actor type and whether the ground truth trajectories collide or go off-road.

F. Analysis

F.1. Reward conditioning analysis

To capture a continuum of driving styles in GIGAFLOW we vary reward parameters for our agents and then allow the policy to condition on them. In total we condition on 12 different parameters used in our reward shown in Table A2.

To better understand how these reward parameter conditions affect GIGAFLOW policy behavior, we attempt to find which parameters can explain the most variation in trajectories. We do this by sampling 200 random reward parameters

within the training distribution range, roll out each of them on various scenarios in nuPlan, then cluster trajectories to find which conditioned reward parameter provides the most mutual information on the clusters. Because single reward parameter will typically dominate the variation seen in the trajectories, we repeat the analysis again but with the previous dominant parameter fixed. We repeat this procedure for the top 6 parameters with the results shown in Fig. A1.

The order of parameters with highest mutual information with the clusters is $\alpha_{\text{center-bias}}$ followed by δ_{goal}, $\alpha_{\text{l-center}}$, α_{comfort}, $\alpha_{\text{l-align}}$, and $\alpha_{\text{vel-align}}$. As seen in Fig. A1, the variation is interpretable and relatable to a human driver. For $\alpha_{\text{center-bias}}$, the policy will pick the left lane if it's biased towards the right (so that it can be further from the boundary) and vice versa for the right lane. For δ_{goal}, the policy will change lanes to get closer to the goal when the max distance is small enough. For $\alpha_{\text{l-center}}$, if this value is high enough, the policy prefers to stay centered in its lane to not incur the extra cost of a lane change, despite the additional progress it could make. For α_{comfort}, the policy cares less about high accelerations and jerks when this penalty is closer to zero, so will take more aggressive turns to stay within its lane and receive centering rewards. For $\alpha_{\text{l-align}}$, the policy incurs less reward for staying oriented with its lane if this value is low, and therefore if this value is low enough the driver is willing to U-turn to get to its goal behind it. Finally for $\alpha_{\text{vel-align}}$, we are rewarded for staying centered in our lane at higher speeds, and therefore when this value is low the policy takes a wider turn to maximize comfort.

By allowing our policy to condition on a range of reward parameters, we solve a few problems at once while keeping the simplicity of a single policy. First, it introduces a wide variety of agent behaviors that makes our policy more robust. One reason is because the policy now must handle both the hidden goals and hidden behaviors of other agents. Due to this, the GIGAFLOW policy learns to handle vehicles that favor different positioning within the lane, drive aggressively, lane change more often, plus combinations of the above. The second benefit is that the optimal driving style could be captured in our conditioning range. So as an additional alignment step, the reward conditions can be tuned to reduce the error between the GIGAFLOW policy and an ideal logged driver. This method would then allow us to create both a chaotic self-play training environment to increase robustness, and a safe and comfortable driving controller, all with the same policy.

F.2. Ablation studies

We perform a series of experiments to understand how different algorithmic choices influence the final performance of the GIGAFLOW agent using a reduced compute budget of 660 GPU-hours ($\sim 30\%$ of the final training run's budget).

Fig. A2 demonstrates the effect of the *advantage filtering* technique described earlier. Using this method we filter out samples that have near zero contribution to PPO's policy and critic losses, thus focusing on rare events at the tails of data distribution. While we expect this technique to improve the wall time performance of the algorithm, to our surprise we also see qualitative difference at convergence. The version of GIGAFLOW without filtering plateaus substantially lower on the Carla LAV benchmark and is unable to reach up to 5% of goals in time leading to timed out episodes in the self-play evaluation mode.

Fig. A3 demonstrates the influence of various algorithmic features on zero-shot benchmark performance. Here, we quantify the error rate across multiple benchmarks; specifically, we measure the additional percentage points needed to achieve a perfect score on each benchmark.

F.3. Long-form evaluation in self-play

To evaluate robustness of our agent in self-play, we devise a version of GIGAFLOW training environment tuned for long-form evaluation. We remove dynamics noise, set the number of drivers to $N_a = 50$, and slightly modify the initialization procedure to ensure minimal clearance between all vehicles at $t = 0$. We remove all traffic light randomization and simply use CARLA's default traffic light system to promote conventional interactions at intersections. We additionally modify drivers' observations to increase their perceived vehicle size by $10\,\text{cm}$ on each side.

We observe that at decision-making frequency of $15\,\text{Hz}$, GIGAFLOW agent conditioned for safe and conservative driving (see Table A4) exhibits nearly accident-free driving, covering on average 3 million km before encountering any collision or going off-road (Fig. A4).

Parameter	Value
Total num. of maps w/ affine transformations	128
Number of agents per sim.	$U\{1, 150\}$
Number of sims per GPU	4800
Max number of agents per GPU	720 000
GPUs used	$8\times$ Nvidia A100 (40 GB)
Total GPU hours used in training	1900
Total number of sims	38 400
Total number of agents	5 760 000
Throughput: steps per second	$\sim$ 1 200 000
Throughput: km driven per second	$\sim$ 1850
Training cost per million km driven	$\sim$ \$5 USD
Total simulated distance driven	1 600 000 000 km
Total number of simulated state transitions	1 000 000 000 000

Table A1: Setup and scale of GIGAFLOW training.

Reward	Training distribution
$R_{\text{goal}} = \mathbb{1}_{(\lvert\lvert x-g\rvert\rvert<\delta_{\text{goal}} \wedge (\mathbb{1}_{\text{waypoint}} \vee \lvert v\rvert<v_{\text{goal}}))}$	$\delta_{\text{goal}} \sim U(2, 12)$ $v_{\text{goal}} = 3$
$R_{\text{collision}} = -(\alpha_{\text{collision}} + 0.1\lvert v\rvert)\,\mathbb{1}_{\text{collision}}$	$\alpha_{\text{collision}} \sim U(0, 3)$
$R_{\text{off-road}} = -\alpha_{\text{boundary}}\,\mathbb{1}_{\text{boundary}}$	$\alpha_{\text{boundary}} \sim U(0, 3)$
$R_{\text{comfort}} = -\alpha_{\text{comfort}}\left(\mathbb{1}_{\lvert a_{\text{long}}\rvert>3} + \mathbb{1}_{\lvert a_{\text{lat}}\rvert>3} + \mathbb{1}_{\lvert \dot{a}_{\text{long}}\rvert>5 \vee \lvert \dot{a}_{\text{lat}}\rvert>5}\right)$	$\alpha_{\text{comfort}} \sim U(0.0, 0.1)$
$R_{\text{l-align}} = \alpha_{\text{l-align}}\Delta t\left(\min(\cos(\theta_f), 0) + \alpha_{\text{vel-align}}\min(\cos(\theta_f) * v, 0) + 0.0025\left(1 - \frac{\lvert\theta_f\rvert}{\pi/2}\right)\right)$	$\alpha_{\text{l-align}} \sim U(2.5\times10^{-4}, 2.5\times10^{-2})$ $\alpha_{\text{vel-align}} \sim U(0, 1)$
$R_{\text{l-center}} = -\alpha_{\text{l-center}}\Delta t\left(\mathbb{1}_{\cos(\theta_f)>0.5} * \lvert x_f - \alpha_{\text{center-bias}}\rvert - \frac{0.05}{\exp(\lvert x_f - \alpha_{\text{center-bias}}\rvert - 0.5)}\right)$	$\alpha_{\text{l-center}} \sim U(2.5\times10^{-4}, 7.5\times10^{-3})$ $\alpha_{\text{center-bias}} \sim U(-0.5, 0.5)$
$R_{\text{velocity}} = \alpha_{\text{velocity}}\,\Delta t\,\max(\cos(\theta_f), 0.0)\,\mathbb{1}_{\lvert v\rvert>2.5}$	$\alpha_{\text{velocity}} = 2.5\times10^{-3}$
$R_{\text{reverse}} = -\alpha_{\text{reverse}}\,\Delta t\,\mathbb{1}_{v<0}$	$\alpha_{\text{reverse}} \sim U(2.5\times10^{-4}, 7.5\times10^{-3})$
$R_{\text{stop-line}} = -\alpha_{\text{stop-line}}\,\mathbb{1}_{\text{stop-line-violation}}$	$\alpha_{\text{stop-line}} \sim U(0, 1)$
$R_{\text{timestep}} = -(\alpha_{\text{timestep}}\Delta t)\,\mathbb{1}_{\lvert v\rvert>0 \vee \lvert a\rvert>0}$	$\alpha_{\text{timestep}} = 2.5\times10^{-5}$

Table A2: Training distribution for each of the reward components.

Parameter	Value
Training batch size	256 000
Batch size per GPU	32 000
Rollout length	128
Num. PPO epochs	3
Discount factor γ	0.999
λ_{GAE}	0.95
Max. episode length	1200 steps (360 s)
PPO clipping ratio	0.2
Value function clipping	None
Initial LR $\alpha^{(0)}$	5×10^{-4}
LR schedule	Cosine
Entropy coefficient	0.01
Value loss coefficient	0.5
Max grad. norm	0.5
Advantage normalization	Enabled
Adv. filtering threshold η	0.01 $\bar{A}_{\text{max}}$ (Alg. 1)
Inference & training precision	16-bit AMP
Model weights initialization	Orthogonal, zero bias

Table A3: RL algorithm settings and hyperparameters used during training.

Setting/Parameter	Value
Vehicle length	$l \sim U(2, 5.5)$
Vehicle width	$w \sim U(1.5, 2.5)$
Maximum speed	$20\,\text{m}\,\text{s}^{-1}$
Timestep Δt	0.066 s
Episode length	9000 steps (600 s)
Agents per sim N_a	50
Observed agents N_o	up to 40 closest
δ_{goal}	10 m
v_{goal}	$3\,\text{m}\,\text{s}^{-1}$
$\alpha_{\text{collision}}$	3.0
α_{boundary}	3.0
α_{comfort}	0.05
$\alpha_{\text{l-align}}$	2.5×10^{-2}
$\alpha_{\text{vel-align}}$	1.0
$\alpha_{\text{l-center}}$	3.8×10^{-3}
$\alpha_{\text{center-bias}}$	0.0
α_{velocity}	2.5×10^{-3}
α_{reverse}	5.0×10^{-3}
$\alpha_{\text{stop-line}}$	1.0
α_{timestep}	2.5×10^{-5}

Table A4: GIGAFLOW settings for long-form evaluation in self-play.

Method	Score ↑	Ego Progress↑	No AF-Collisions ↑	Comfort ↑	TTC in bounds ↑	Driving Direction ↑	Speed Limit ↑	Drivable Area ↑	Ego Making Progress ↑
Val14 Benchmark									
Urban Driver (Scheel et al., 2021)	50	-	-	-	-	-	-	-	-
GC-GPP (Hallgarten et al., 2023)	55	-	-	-	-	-	-	-	-
PlanCNN (Renz et al., 2022)	72	-	-	-	-	-	-	-	-
IDM (Treiber et al., 2000)	77	-	-	-	-	-	-	-	-
PDM-Hybrid (Dauner et al., 2023)	92.1	90.2	98.1	94.8	93.5	99.9	99.8	99.5	99.1
Diffion-ES (Yang et al., 2024)	92.2	91.2	97.7	93.4	93.8	100.0	99.7	99.6	99.2
GIGAFLOW (ours)	93.8±0.11	93.6±0.06	98.4±0.09	96.4±0.27	93.8±0.23	99.6±0.05	99.9±0.00	99.7±0.04	99.0±0.07

Table A5: Results on Val14 (Dauner et al., 2023) nuPlan benchmark.

Method	DS↑	RC↑	IP↑	Ped↓	Veh↓	Lay↓	Red↓	Stop↓	Off↓	Dev↓	TO↓	Block↓
CL1 Testing Routes (`leaderboard.carla.org/get_started_v1`)												
CARLA Agent (our run)	29 ±0	41 ±0	0.73 ±0.03	0.03 ±0.00	0.36 ±0.11	0.00 ±0.00	0.28 ±0.02	0.03 ±0.02	0.00 ±0.00	0.44 ±0.02	0.01 ±0.01	0.15 ±0.00
Expert from Jaeger et al. (2023) (our run)	90 ±0	96 ±1	0.94 ±0.01	0.00 ±0.00	0.06 ±0.01	0.00 ±0.00	0.06 ±0.03	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.05 ±0.02	0.05 ±0.00
GIGAFLOW (ours)	93 ±1	97 ±2	0.95 ±0.01	0.00 ±0.00	0.07 ±0.01	0.00 ±0.00	0.01 ±0.01	0.00 ±0.00	0.64 ±0.44	0.01 ±0.01	0.07 ±0.02	0.00 ±0.00
LAV Benchmark (Chen & Krähenbühl, 2022) (with adapted scenarios from Jaeger et al. (2023))												
CARLA Agent (our run)	9 ±2	58 ±0	0.18 ±0.04	0.22 ±0.00	0.74 ±0.43	0.00 ±0.00	3.99 ±0.10	0.58 ±0.21	0.00 ±0.00	0.87 ±0.00	0.00 ±0.00	0.00 ±0.00
Expert from Jaeger et al. (2023)	94	95	0.99	0.00	0.02	0.00	0.02	0.00	-	0.00	0.00	0.08
Expert from Jaeger et al. (2023) (our run)	92 ±9	95 ±7	0.98 ±0.02	0.00 ±0.00	0.04 ±0.05	0.00 ±0.00	0.02 ±0.03	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.09 ±0.07	0.07 ±0.07
GIGAFLOW (ours)	99 ±1	99 ±1	1.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00	0.02 ±0.03	0.00 ±0.00
Longest6 Benchmark (Chitta et al., 2023)												
CARLA Agent (our run)	7 ±1	54 ±0	0.25 ±0.01	0.31 ±0.00	1.50 ±0.21	0.00 ±0.00	1.84 ±0.06	0.10 ±0.02	0.00 ±0.00	0.43 ±0.00	0.02 ±0.01	0.04 ±0.00
Expert from Chitta et al. (2023)	77 ±2	89 ±1	0.86 ±0.03	0.02	0.28	0.01	0.03	0.00	0.00	0.00	0.08	0.13
Expert from Jaeger et al. (2023)	81 ±3	90 ±1	0.91 ±0.04	0.01	0.21	0.00	0.01	-	-	0.00	0.07	0.09
Expert from Jaeger et al. (2023) (our run)	83 ±1	94 ±2	0.88 ±0.01	0.00 ±0.00	0.20 ±0.03	0.00 ±0.00	0.04 ±0.01	0.02 ±0.01	0.00 ±0.00	0.00 ±0.00	0.09 ±0.00	0.06 ±0.02
GIGAFLOW (ours)	92 ±2	99 ±1	0.93 ±0.01	0.02 ±0.00	0.08 ±0.01	0.00 ±0.00	0.04 ±0.02	0.03 ±0.01	0.24 ±0.08	0.03 ±0.02	0.05 ±0.03	0.00 ±0.00

Table A6: Results on CARLA benchmarks.

Method	Off-Road Rate (%) ↓	Collision Rate (%) ↓	Kinematic Infeasibility (%) ↓	Log ADE (m)	Route Progress Ratio (%) ↑	Total Distance Driven Ratio (%) ↑	Score (%) ↑
Expert Demonstration (Gulino et al., 2023)	0.32	0.61	4.33	0.00	100.00	100.00	≤ 99.07
Wayformer (Gulino et al., 2023)	7.89	10.68	5.40	2.38	123.58	-	≤ 81.43
DQN (Gulino et al., 2023)	3.74±0.90	6.50±0.31	0.00±0.0	9.83±0.48	177.91±5.67	-	≤ 89.76±0.95
BC (Gulino et al., 2023)	1.11±0.2	4.59±0.06	0.00±0.0	2.26±0.2	129.84±0.98	-	≤ 94.3±0.21
GIGAFLOW (ours)	0.43±0.008	0.43±0.005	0.14±0.008	5.87±0.01	146.27±0.08	106.79±0.05	99.16±0.009

Table A7: Results on Waymax benchmarks when evaluated with IDM agents. For the `Score` of other agents, we assumed 100% progress and mutually exclusive collision and offroad events.

Method	Linear Speed	Linear Accel.	Ang. Speed	Ang. Accel.	Dist. to Obj.	Collision	TTC	Dist. to Road Edge	Off-road	**Composite Metric**
Expert Demo.* (Montali et al., 2023)	0.5610	0.3300	0.5630	0.4890	0.4850	1.0000	0.8810	0.7130	1.0000	0.7220
Random Agent*	0.0020	0.0440	0.0740	0.1200	0.0000	0.0000	0.7340	0.1780	0.2870	0.1550
Linear Extrapolation*	0.1570	0.1190	0.0190	0.0350	0.2470	0.4110	0.7750	0.5020	0.4630	0.3240
Stationary Policy	0.0408	0.0612	0.4984	0.3195	0.0553	0.9466	0.7363	0.2408	0.8575	0.5007
Joint-Multipath++* (Wang & Zhen, 2023)	0.4340	0.2300	0.5150	0.4520	0.3450	0.5670	0.8120	0.6390	0.6820	0.5330
PredSim*	0.4051	0.2208	0.4958	0.4653	0.3441	0.7193	0.8027	0.6167	0.7519	0.5663
Wayformer* (Nayakanti et al., 2023)	0.3310	0.0980	0.4130	0.4060	0.2970	0.8700	0.7820	0.5920	0.8660	0.5750
SceneDMF* (Guo et al., 2023)	0.4315	0.2767	0.5230	0.4666	0.3678	0.7447	0.8128	0.6215	0.7392	0.5821
MTR+++* (Qian et al., 2023)	0.4119	0.1066	0.4838	0.4365	0.3457	0.8630	0.7969	0.6545	0.8954	0.6077
GIGAFLOW (ours)	0.2613	0.2534	0.5060	0.4756	0.3161	0.9470	0.8077	0.5445	0.9095	0.6190
VPD-BP* (Huang et al., 2024)	0.4751	0.2161	0.5358	0.4775	0.3908	0.8162	0.8234	0.6628	0.9010	0.6315
MTR-E* (Qian et al., 2023)	0.4278	0.2353	0.5335	0.4753	0.3455	0.8774	0.7983	0.6541	0.9143	0.6348
MVTE* (Wang et al., 2023b)	0.4426	0.2218	0.5353	0.4810	0.3819	0.8943	0.8321	0.6641	0.9086	0.6448
Trajeglish* (Philion et al., 2023)	0.4504	0.1929	0.5382	0.4850	0.3869	0.9226	0.8369	0.6596	0.8864	0.6451
InteractionFormer*	0.4294	0.2394	0.5291	0.4780	0.3776	0.9591	0.8311	0.6464	0.9347	0.6587

Table A8: Per-component metrics, as defined and computed within WOSAC using the WOMD validation split. *-scores publicly reported on WOMD test set at `https://waymo.com/open/challenges/2023/sim-agents/` and from Montali et al. (2023). All scores are derived using the 2023 metric definitions. Shaded rows correspond to approaches that do not use training data. We see that GIGAFLOW, in zero-shot evaluation, is capable of producing effective driving performance that approaches expert policies specifically developed for imitative traffic modeling using the provided training data, even without having been shown WOMD data or maps previously.

Collisions

	Overall			Ego			Vehicles			Pedestrians			Bicycles		
Method	*Total*	$\neg C$	*C*	*Total*	$\neg C$	*C*	*Total*	$\neg C$	*C*	*Total*	$\neg C$	*C*	*Total*	$\neg C$	*C*
Expert Demo.	0.0337	0.0000	1.0000	0.0048	0.0000	1.0000	0.0076	0.0000	1.0000	0.2986	0.0000	1.0000	0.0547	0.0000	1.0000
GIGAFLOW (ours)	0.0322	0.0085	0.7082	0.0021	0.0006	0.3223	0.0080	0.0051	0.3700	0.2391	0.0029	0.7938	0.0291	0.0003	0.5268
Stationary	0.0220	0.0017	0.6015	0.0021	0.0007	0.2832	0.0035	0.0012	0.3513	0.2012	0.0003	0.6731	0.0242	0.0001	0.3556

Off-road

	Overall			Ego			Vehicles			Pedestrians			Bicycles		
	Total	$\neg O$	*O*	*Total*	$\neg O$	*O*	*Total*	$\neg O$	*O*	*Total*	$\neg O$	*O*	*Total*	$\neg O$	*O*
Expert Demo.	0.1263	0.0000	1.0000	0.0127	0.0000	1.0000	0.0599	0.0000	1.0000	0.8513	0.0000	1.0000	0.3611	0.0000	1.0000
GIGAFLOW (ours)	0.1105	0.0072	0.8252	0.0071	0.0003	0.5367	0.0409	0.0051	0.6033	0.8385	0.0014	0.9847	0.2397	0.0013	0.6635
Stationary	0.0959	0.0060	0.7053	0.0041	0.0001	0.3220	0.0298	0.0052	0.5117	0.6999	0.0002	0.8201	0.2346	0.0002	0.6493

Table A9: Calculated rates by which vehicles controlled by either the GIGAFLOW policy or with the baseline stationary policy are assessed to be in collision (condition C) or drive off-road (condition O), grouped by actor type and whether these outcomes were assessed to occur in the ground truth. We anchor this analysis by the provided expert demonstrations recorded in WOMD. GIGAFLOW produces the expected vehicle behavior when the ground truth trajectories do not collide or drive off-road, achieving incidence rates well below one percent. While not directly rewarded in WOSAC scoring, the aversion of GIGAFLOW to colliding and off-road driving is evident among circumstances where the ground truth trajectories are assessed to do so. Aside from pedestrians, all other actors controlled by GIGAFLOW have incidence rates far lower than anticipated (where generated rollouts are expected to collide or drive off-road if the underlying ground truth trajectory does so). Notably, there is a proportion of trajectories that are initialized to be in collision or off-road (unsurprising given the presence of pedestrians) since the incidence rates of the stationary policy are not zero for these events in this conditional analysis. From this, we conclude that GIGAFLOW avoids an excess amount of collisions or off-road behavior as the incidence rates do not significantly increase when applying the self-play policy.

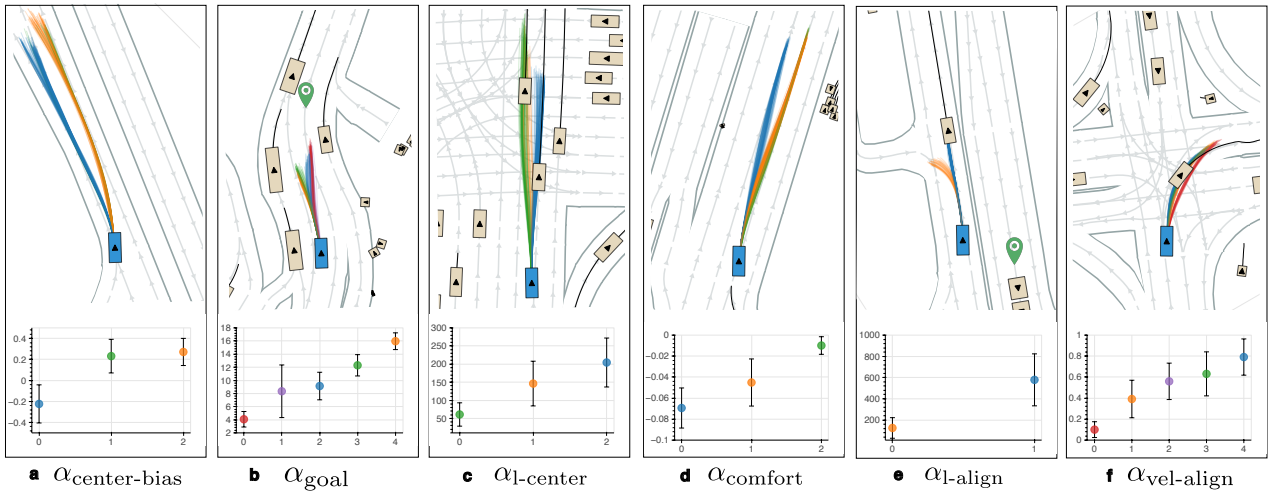

Figure A1: Conditional analysis.

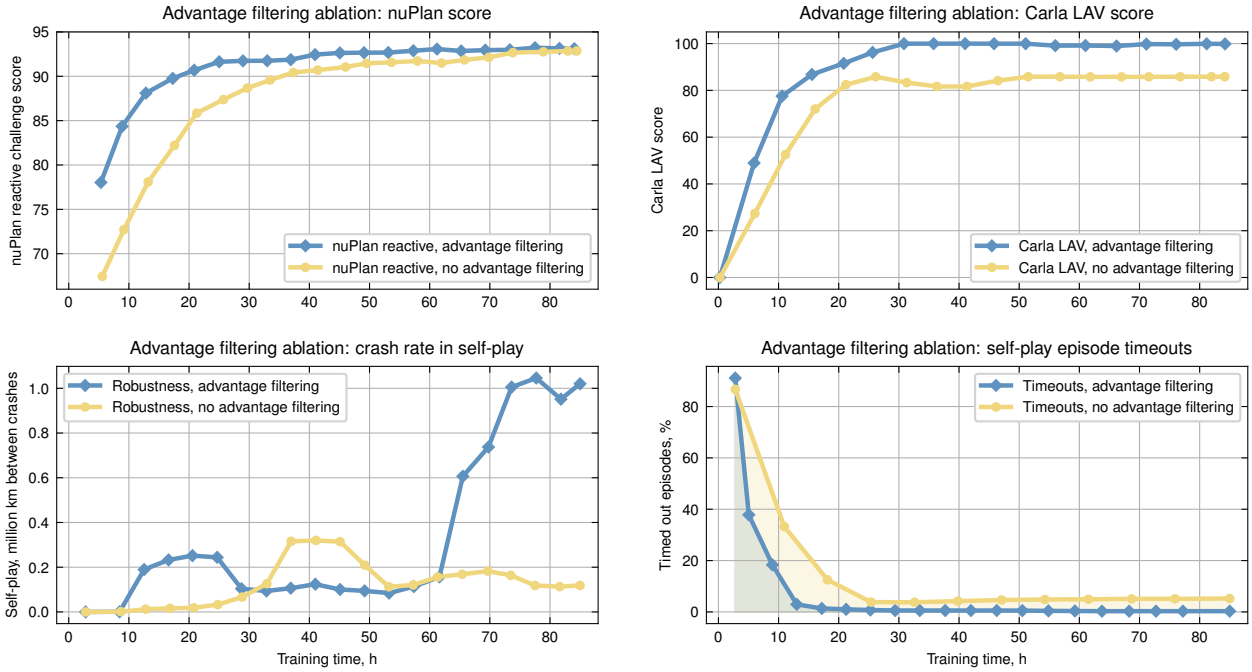

Figure A2: Training with and without *advantage filtering* (see Algorithm 1). Top row: nuPlan reactive challenge and Carla LAV Benchmark results. Bottom row: accident rate and task completion rate in self-play.

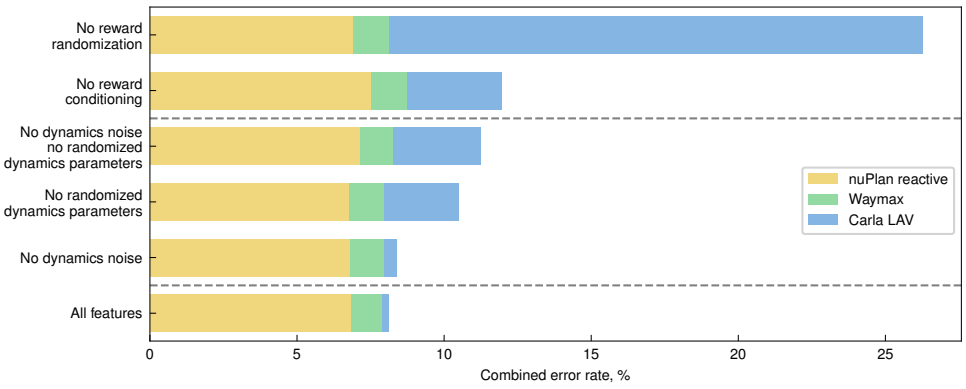

Figure A3: Aggregate impact of algorithmic features used in GIGAFLOW. The length of each bar indicates the cumulative percentage points required to attain a perfect score across three benchmarks.

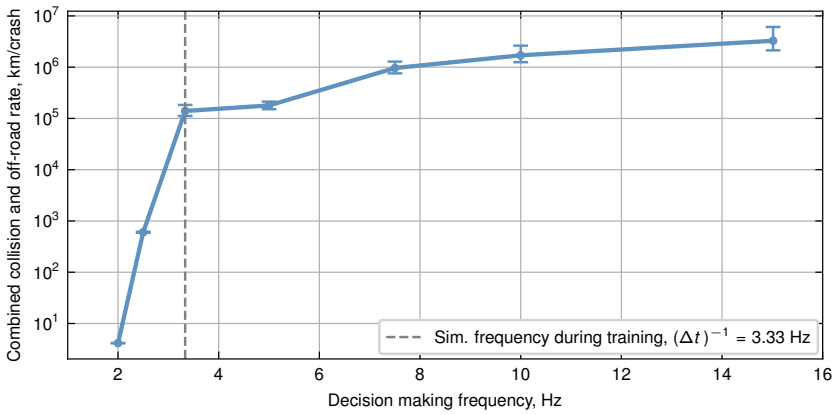

Figure A4: Accident rate in GIGAFLOW self-play at different values of simulation/decision-making frequency.