DeepRacer: Optimization and Adaption

Kirsten Odendaal
kodendaal3@gatech.edu
College of Computing
Georgia Institute of Technology

1 Introduction

Recent Reinforcement Learning (RL) advancements have accelerated progress in autonomous driving and robotics, enabling agents to navigate complex environments using sensory inputs and learned policies. Autonomous racing is an ideal testbed for RL methods due to its challenging requirements for real-time decision-making, precision control, and generalization across varying conditions and environments. A prominent platform capturing these challenges is AWS DeepRacer, a scaled autonomous racing vehicle for developing and benchmarking RL algorithms in realistic simulated conditions (Balaji et al., 2019). The environment provides a compelling scenario to evaluate how learned control policies transfer across different simulation scenarios, a key step towards bridging gaps between simulated training and real-world applications (Balaji & Etzioni, 2020).

In this study, we utilize the AWS DeepRacer environment to develop and evaluate a robust autonomous racing agent capable of generalizing across multiple track layouts (*reInvent2019 wide, reInvent2019 track, and New York Track*) and diverse racing conditions: time trials, obstacle avoidance scenarios (six static obstacles), and head-to-head competitions (three competing vehicles). We employ Proximal Policy Optimization (PPO), specifically the clipped variant for stable policy updates, combined with Convolutional Neural Networks (CNNs) to process sensory inputs from stereo camera and LIDAR sensors. While the trained agent demonstrates effective generalization across tracks in the time-trial setting, our results reveal significant challenges in transferring this performance to scenarios involving obstacles or dynamic opponents, highlighting a critical generalization gap dependent on task complexity.

2 Technical Background

In this section, we briefly review the core RL methodologies used in our experiments: Proximal Policy Optimization (PPO) for policy learning and Convolutional Neural Networks (CNNs) for processing high-dimensional sensory inputs in autonomous racing.

2.1 Proximal Policy Optimization (Clipped)

Proximal Policy Optimization (PPO) is a widely-adopted policy gradient method designed to improve the stability and reliability of policy updates (Schulman et al., 2017). PPO constrains policy updates to a region near the current policy, preventing excessively large updates that could destabilize learning by optimizing a clipped surrogate objective function:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t,\, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio between the new policy π_θ and the old policy $\pi_{\theta_{old}}$, and $\hat{A}_t$ is the estimated advantage at time-step t. The hyperparameter ϵ clips the ratio $r_t(\theta)$, ensuring the new policy does not deviate drastically from the old, thus enabling more stable training. PPO has proven effective in complex control problems, including robotics and autonomous navigation, due to its balance of sample efficiency, simplicity, and reliable policy improvement.

Compared to value-based methods like Deep Q-Networks (DQN) (Mnih et al., 2015) and its variants such as Double DQN (van Hasselt et al., 2016), PPO is often preferred for complex control tasks, even with discrete action spaces like those typically used in DeepRacer. While DQN performs well in environments with smaller discrete action spaces, PPO can offer greater stability during training

in high-dimensional state spaces and potentially large discrete action spaces. Compared to Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016), which handles continuous actions but can suffer from instability, PPO provides a more robust and empirically stable alternative suitable for optimizing fine-grained control policies where learning stability is critical.

2.2 Convolutional Networks for Image Processing

The AWS DeepRacer environment provides observations including front-facing stereo gray-scale camera images (160×120 pixels each). These images are the primary input for track navigation, encoding visual cues for lane following, curve anticipation, and obstacle detection. To process these high-dimensional inputs, we employ a Convolutional Neural Network (CNN) as a visual feature extractor, transforming raw pixels into a compact latent representation for decision-making.

Input images are normalized to $[0, 1]$ within the network's forward pass. Our CNN architecture uses three convolutional layers (with kernel sizes 8x8, 4x4, 3x3 and strides 4, 2, 1, respectively, each followed by a ReLU activation) to process the raw 160×120 stereo input (2 channels). Batch Normalization is not used in this specific encoder. The resulting feature maps are flattened into a high-dimensional vector. When LIDAR sensor data is available (e.g., in obstacle avoidance and head-to-head modes), the 64-dimensional LIDAR vector (representing radial distances) is processed in parallel by a separate Multi-Layer Perceptron (MLP) consisting of two linear layers (projecting 64 -> 128 -> 128 dimensions) with ReLU activations, producing a LIDAR embedding. The flattened visual features and the LIDAR embedding are concatenated. This combined vector is then passed through a final 'combiner' MLP (a linear layer followed by ReLU) to produce a fixed-size embedding (e.g., 512 dimensions). Layer Normalization is applied to this final embedding vector. This multi-modal fusion allows the agent to integrate both visual-semantic cues and spatial-distance information. The final normalized combined latent representation is fed into two separate heads: an actor network outputting a probability distribution over discrete action tuples (steering, throttle), and a critic network estimating the state-value function $V(s)$. This actor-critic structure facilitates efficient policy learning and stable value estimation.

3 Problem Definition

We frame the DeepRacer racing task as a reinforcement learning problem where an agent must learn to control a simulated autonomous vehicle using high-dimensional sensory input. The objective is to navigate diverse race tracks efficiently and safely under multiple racing conditions.

3.1 Environment Description

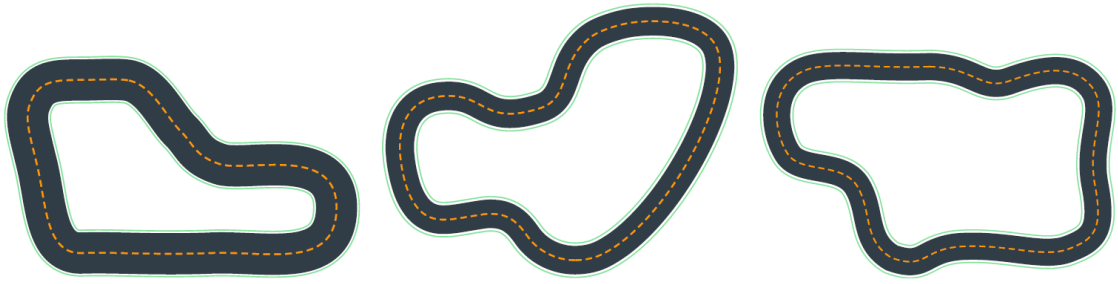
The AWS DeepRacer simulator provides a Gym-compatible (Brockman et al., 2016) environment where an agent interacts with the world via discrete control commands based on camera and LIDAR observations. The task is formulated as a partially observable Markov decision process (POMDP) (Sutton & Barto, 2018) due to limited sensing and the absence of full state observability at inference time. At each timestep, the agent receives stereo grayscale images from front-facing cameras and a 64-dimensional LIDAR vector representing radial distances to nearby obstacles over a 360° range. While internal state variables (e.g., x, y, heading, velocity) are available for reward shaping, they are excluded from the observation space to mimic real-world sensor constraints. The agent selects an action from a discrete set of N steering and throttle combinations. Episodes terminate under conditions like track completion, collision, leaving the track, immobilization, or time expiration. Table 1 summarizes the key components of the DeepRacer environment.

To assess the agent's ability to generalize, we evaluate it across three race tracks of increasing complexity (*A-Z Speedway*, *Smile Speedway*, and *Empire City*, see Figure 1) under three task variants:

- *Time-Trial:* Fast lap completion without obstacles or competitors.
- *Obstacle-Avoidance:* Navigation around six static obstacles.
- *Head-to-Bot:* Racing against three moving opponent vehicles.

Component	DeepRacer Problem	Description
Observation Space (O)	Stereo camera images (160 × 120 each) + 64D LIDAR vector	Partially observable POMDP: Grayscale stereo images provide visual cues; LIDAR gives radial distances to obstacles. Raw state variables (position, heading, velocity) are inaccessible.
Action Space (A)	Discrete: $\{(s_i, v_i)\}_{i=1}^{N}$ steering/throttle tuples	Actions represent joint choices of steering angle ($\pm 30^\circ$) and speed (0.2 to 1.0 m/s), enabling lane following, turning, and evasive maneuvers.
Transition Dynamics	Deterministic simulation with noisy observations	Physics-based simulator provides deterministic transitions; sensor noise and occlusion induce partial observability. Track layout and agent dynamics are fixed.
Reward Function ($R(s, a)$)	$R = R_{\text{center}} + R_{\text{progress}} + R_{\text{alignment}} + R_{\text{obstacle}}$	*Reward Shaping (used in this study):* (+) Centerline tracking (exponential decay from center). (+) Percentage progress increment. (+) Heading alignment to next waypoint. (−) Penalty based on exponential distance to nearest obstacle (via LIDAR).
Episode Termination	Lap complete, collision, off-track, timeout, immobilization	Episodes end if the agent completes the lap, crashes, leaves the track, gets stuck, or exceeds time limit.
Environmental Parameters	Tracks (Figure 1): *reInvent2019 Wide* (L:16.64m, W:1.07m) *reInvent2019 Track* (L:23.12m, W:1.07m) *New York Track* (L:21.88m, W:0.76m)	Race types: Time-trial (clean), Obstacle-Avoidance (6 static), Head-to-Bot (3 opponents). Tracks vary in curvature, width, and complexity.

Table 1: AWS DeepRacer Environment Components



(a) A to Z Speedway `reInvent2019_wide` (b) Smile Speedway `reInvent2019_track` (c) Empire City Training `New_York_Track`

Figure 1: Track layouts used in the study: (a) A-Z Speedway (reInvent2019 Wide), (b) Smile Speedway (reInvent2019 Track), (c) Empire City (New York Track). Track complexity increases from left to right.

3.2 Main Challenges

Achieving robust policy generalization in the AWS DeepRacer environment involves several non-trivial challenges stemming from environmental variability and RL fundamentals:

1. *Track Variability:* The agent must generalize across three tracks (*A-Z Speedway*, *Smile Speedway*, *Empire City*) varying significantly in curvature, width, and complexity. Overfitting to one track's geometry hinders performance on others, necessitating the learning of abstract driving principles over spatial memorization.
2. *Task Mode Generalization:* Evaluation occurs in three modes: clean time-trials, static obstacle-avoidance, and dynamic multi-agent races. These vary in sensory complexity and decision needs (e.g., reactive maneuvering for obstacles vs. strategic interaction with unpredictable opponents). A general policy must adapt across these contexts.
3. *Partial Observability:* Operating under POMDP constraints, the agent relies solely on noisy, limited-range stereo vision and LIDAR, lacking true global state (e.g., absolute coordinates, opponent headings). This restricts long-term anticipation and complicates generalization, especially on narrow or cluttered tracks.

Figure Placeholder: NN Architecture Diagram

Figure 2: Multi-modal neural network architecture (Conceptual Diagram).

4. *Reward Sparsity and Ambiguity:* Without reward shaping, meaningful rewards (lap completion) are sparse, hindering credit assignment (Sutton & Barto, 2018). Shaped rewards, while denser, risk promoting suboptimal behaviors (e.g., reward hacking) if not carefully designed.
5. *Temporal Credit Assignment:* Associating actions with delayed outcomes (e.g., a crash resulting from an earlier poor line choice) is crucial for non-myopic racing but difficult due to delayed feedback and the lack of explicit planning mechanisms in standard policy architectures.
6. *Dynamic Scene Understanding:* Multi-agent scenarios require interpreting a constantly changing environment. The agent must model opponent vehicle dynamics (speeds, trajectories, collision risks) from partial, low-dimensional inputs, a significant step beyond interpreting static track boundaries.

Together, these challenges demand generalized policies robust across spatial, temporal, and sensory domains. Simple behavioral cloning or track memorization is insufficient; robust solutions require encoding general driving priors effective under varying conditions and realistic perturbations.

4 Methodology

We trained a DeepRacer agent using Proximal Policy Optimization (PPO) with a curriculum learning strategy (Narvekar et al., 2020) and a multi-modal neural network processing visual and LIDAR inputs. To promote generalization and handle increasing complexity, training progressed through stages:

1. *Fixed Start:* Initial training on the easiest track (*reInvent2019_Wide*) with a fixed starting pose to learn basic control.
2. *Randomized Start:* Introducing random starting positions on the track to expose the agent to diverse sections.
3. *Randomized Direction & Harder Tracks:* Applying random start positions and random lap direction (forward/reverse) on progressively harder tracks (*reInvent2019_Track*, *New_York_Track*).

This progression gradually increased task difficulty. Moderate domain randomization (minor variations in start velocity/heading) was also used to enhance robustness, inspired by techniques like (Tobin et al., 2017). The agent trained for a total of 20,000 simulation steps, distributed across the curriculum stages, advancing based on performance convergence.

4.1 Training Configuration and Hyperparameters

We used the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017) with a clipped objective and Generalized Advantage Estimation (GAE) (Schulman et al., 2016). Key hyperparameters are listed in Table 3. These were chosen based on standard practices and limited preliminary tuning due to time constraints, and remained fixed throughout the curriculum. Training involved collecting rollouts of $n_{\text{steps}} = 512$ steps (experience chunks) and performing 5 update epochs per rollout using minibatches of size 64. The total training time for 20,000 steps was approximately two hours on a standard machine using the AWS DeepRacer simulation environment. The final policy obtained after the full curriculum was used for evaluation.

4.2 Multi-Modal Neural Network Architecture

The agent processes sensory inputs using a multi-modal network (summarized in Table 2).

- *Visual Input:* Raw stereo grayscale camera images ($160 \times 120 \times 2$) are normalized and processed by a CNN consisting of three convolutional layers (kernels 8x8, 4x4, 3x3; strides 4, 2, 1; ReLU activations, no BatchNorm). The output is flattened to an 11264-dimensional vector.

Module	Layers (in order)	Output dims
Stereo CNN Encoder	Input: $2 \times 120 \times 160$ stereo images (Normalized in fwd pass) Conv1: 8x8, stride 4, ReLU (32 channels) Conv2: 4x4, stride 2, ReLU (64 channels) Conv3: 3x3, stride 1, ReLU (64 channels) Flatten	11264
LIDAR MLP Encoder	Input: 64-dim distance vector Linear($64 \to 128$), ReLU Linear($128 \to 128$), ReLU	128
Fusion	Concatenate (11264 + 128)	11392
Combiner	Linear($11392 \to 512$), ReLU LayerNorm	512
Actor head	FC($512 \to N_{\text{actions}}$), Softmax	N_{actions}
Critic head	FC($512 \to 1$), Linear	1

Table 2: Multi-modal policy network architecture used with PPO (Updated to match code implementation).

Hyper-parameter	Value	Description
Total timesteps	20 000	Environment steps used for training (across curriculum)
PPO clip range ϵ	0.2	Ratio clipping threshold
Discount factor γ	0.99	Long-term reward discount
GAE λ	0.95	Generalized Advantage Estimation parameter
Learning rate	8×10^{-4}	Adam optimizer step size
Batch size	64	Minibatch size for SGD per epoch
Rollout length n_{steps}	512	Steps collected before each parameter update
Epochs / update	5	Number of gradient passes over rollout buffer per update
Value-loss coef.	0.5	Weight of the critic (value function) loss term
Entropy coef.	0.01	Weight of the policy entropy bonus (encourages exploration)
Seed	42	Random seed for reproducibility

Table 3: Training hyper-parameters for PPO.

- *LIDAR Input:* A 64-dimensional LIDAR vector (radial distances) is processed by a two-layer MLP (Linear 64->128->ReLU, Linear 128->128->ReLU), generating a 128-dim LIDAR embedding.
- *Fusion, Combination, and Policy Heads:* The flattened visual features (11264-dim) and the LIDAR embedding (128-dim) are concatenated (11392-dim total). This combined vector is passed through a 'combiner' MLP (Linear layer projecting to 512 dimensions, followed by ReLU) and then Layer Normalization. This final 512-dimensional normalized embedding is fed into separate fully-connected heads for the *actor* (outputting a policy over N_{actions} discrete steering/throttle tuples via Softmax) and the *critic* (outputting the value estimate $V(s)$ via a linear layer), following the standard PPO actor-critic structure.

This architecture allows the agent to integrate visual cues with spatial proximity information from LIDAR, combining them into a fixed-size representation suitable for policy and value function approximation.

4.3 Detailed Reward Shaping

The AWS DeepRacer environment requires a shaped reward signal to encourage efficient, safe driving and to mitigate the sparsity of natural rewards (e.g., only receiving a positive signal at lap completion). Our implementation employs a modular reward function composed of several interpretable components, aligned with specific driving objectives such as maintaining central alignment, progressing steadily, and avoiding collisions. The total reward is computed as a weighted sum of individual reward terms:

$$R_t = \lambda_{\text{center}} R_{\text{center}} + \lambda_{\text{speed}} R_{\text{progress}} - \lambda_{\text{obstacle}} R_{\text{obstacle}} - \lambda_{\text{crash}} \cdot \mathbb{I}_{\text{crash}}$$

where λ_i denote weighting coefficients, and $\mathbb{I}_{\text{crash}}$ is an indicator function for terminal collision events.

1. *Centerline Following:* To promote lane-keeping, we apply an exponential decay based on the vehicle's normalized lateral offset d_t from the centerline:

$$R_{\text{center}} = e^{-0.8 \cdot \left(\frac{d_t}{w/2}\right)^2}$$

where w is the track width and d_t is the distance from center. This term ensures smoother, center-aligned trajectories.

2. *Progress Incentive:* The agent receives a linear reward proportional to the track progress:

$$R_{\text{progress}} = \frac{\text{progress}}{100}$$

where progress is expressed as a percentage. This encourages continuous forward motion.

3. *Lap Completion Bonus:* Upon nearing the end of the lap (i.e., progress $\geq 98\%$), a completion bonus is awarded, scaled by a factor inversely proportional to the number of steps taken:

$$R_{\text{completion}} = \begin{cases} 10 \cdot \max(0.5, \min(1.5, \frac{S_{\text{target}}}{S_t})) & \text{if progress} \geq 98 \\ 0 & \text{otherwise} \end{cases},$$

where S_t is the number of steps taken and $S_{\text{target}} = 250$ is the target for an optimal lap.

4. *Obstacle Penalty:* When static obstacles are present, the agent is penalized based on the minimum Euclidean distance d_o to nearby obstacles:

$$R_{\text{obstacle}} = \begin{cases} \min\left(1.0, 1.5 \cdot \exp\left(-\frac{d_{\text{safe}}}{d_o + \varepsilon}\right)\right) & d_o < d_{\text{safe}} \\ 0 & \text{otherwise} \end{cases},$$

with $d_{\text{safe}} = 1.0$ meters and $\varepsilon = 10^{-5}$ to avoid division by zero. This exponential penalty discourages risky proximity to obstacles.

Crash Penalty. A large negative penalty is applied if the agent crashes:

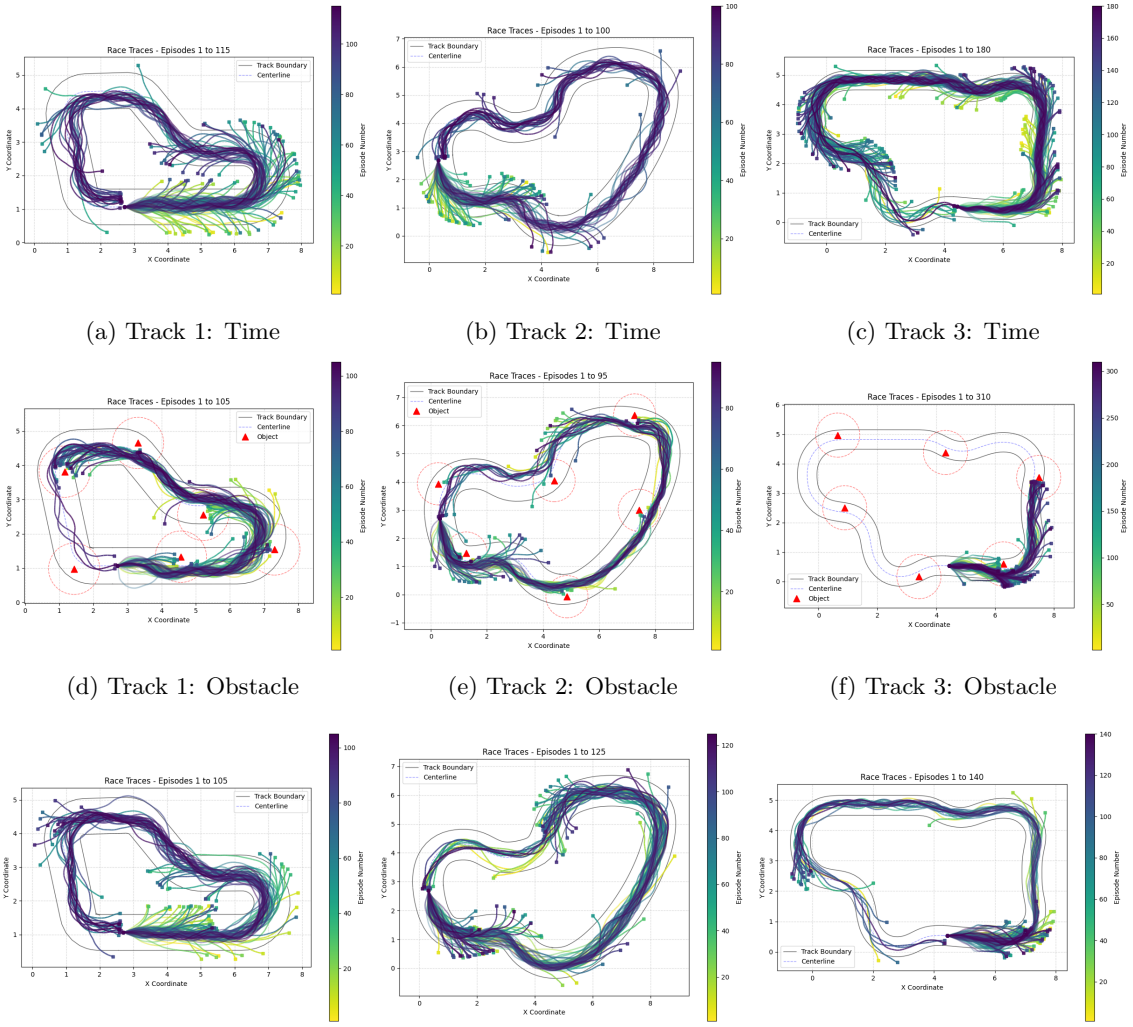
$$R_{\text{crash}} = \begin{cases} -10.0 & \text{if crash} \\ 0 & \text{otherwise} \end{cases}.$$

The reward weights λ_i were set heuristically based on preliminary tests to prioritize steady progress and safety: $\lambda_{\text{center}} = 1.0, \lambda_{\text{progress}} = 1.0, \lambda_{\text{obstacle}} = 1.0$.

5 Results

After training, we evaluated the learned agent across all nine scenarios (3 tracks $\times$ 3 modes) measuring performance and generalization via *lap completion rate*, *average lap time* (successful runs), and *average progress* (percent track completed, including failures). Agent trajectories (Figure **??**) illustrate driving behavior, while quantitative performance metrics are summarized in Figure 4. Clear trends emerge:

- *Time-Trial Performance:* The agent excelled in time-trials (no obstacles/opponents), achieving nearly **100%** lap completion and progress on all tracks (Figure 4a). Average lap times demonstrated adaptation to track complexity: **9.2s** (A-Z Speedway, wide), **14-15s** (reInvent2019 Track, medium), and **17.8s** (Empire City, narrow). These efficient times and high completion rates indicate a robust lane-following and cornering policy learned via curriculum training, successfully generalizing across track geometries seen during training (including the narrow New York track) in this simplified setting. Trajectories in Figure **??** (top row) show consistent lines.
- *Obstacle Avoidance:* Introducing static obstacles significantly degraded performance, despite LIDAR input and reward penalties. Having never explicitly trained with obstacles, the agent struggled to generalize. Completion rates dropped: **75%** (A-Z), **50-60%** (reInvent2019), and below **40%** (Empire City). Successful lap times increased substantially (**20s** on A-Z, **30+s** estimated on Empire City) due to cautious navigation, halts, or sometimes collisions after evading initial obstacles (see trajectories in Figure **??**, middle row). Average progress per episode dropped significantly compared to time-trials (Figure 4b), highlighting a major generalization gap when encountering static hazards.



(a) Track 1: Time (b) Track 2: Time (c) Track 3: Time

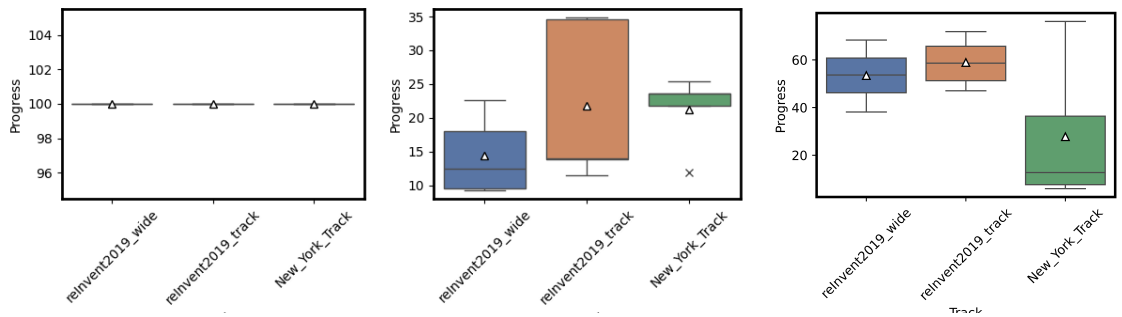
(d) Track 1: Obstacle (e) Track 2: Obstacle (f) Track 3: Obstacle

(g) Track 1: Moving (h) Track 2: Moving (i) Track 3: Moving

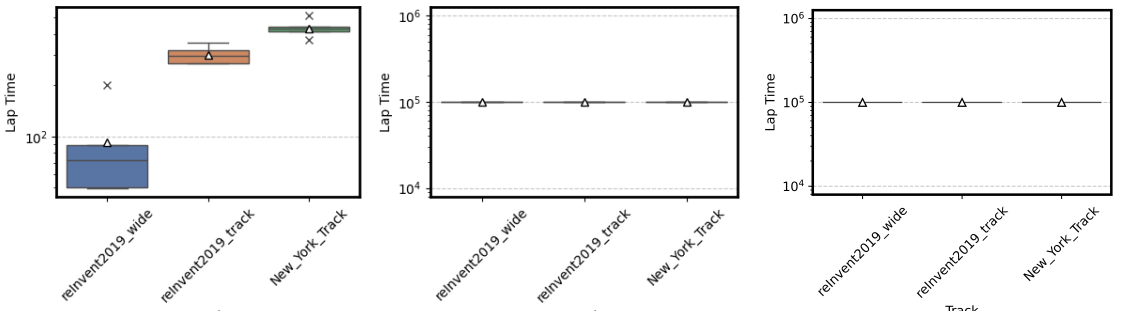
Figure 3: Training time traces for (top) Time Trial, (middle) Obstacle, and (bottom) Moving Bot configurations across the three circuits

- *Head-to-Head (Dynamic Opponents):* Racing against three moving bots proved the most challenging, revealing a severe lack of generalization to dynamic, interactive scenarios. Completion rates plummeted further: **50%** (A-Z), **20-30%** (reInvent2019), and nearly **0%** (Empire City). Collisions were frequent, especially in narrow sections. Even successful laps were much slower (**22s** on A-Z) as the agent often trailed opponents cautiously (using LIDAR) but failed to initiate safe overtakes, sometimes timing out. Average progress was low, especially on the hardest track (**40-50%**, Figure 4c). The agent, trained only in single-agent mode, lacked strategies for multi-agent dynamics. Trajectories (Figure **??**, bottom row) often show aborted laps or hesitant following. This outcome underscores the failure to generalize to dynamic multi-agent environments.

The results highlight a stark contrast: while the PPO agent with curriculum training mastered solo time-trials across varied tracks, its performance drastically dropped when faced with static obstacles or dynamic opponents—conditions absent during training. The high completion rates and fast times in time-trials confirm successful learning of basic driving skills and generalization to track geometry. However, the significantly lower completion rates, slower times, and reduced progress in obstacle/multi-agent modes reveal a critical limitation in generalizing to increased task complexity involving hazard avoidance and interaction. The narrowest track (*New York*) consistently posed the



(a) Time Trial: Progress (%) (b) Obstacle: Progress (%) (c) Moving: Progress (%)



(d) Time Trial: Lap Time (s) (e) Obstacle: Lap Time (s) (f) Moving: Lap Time (s)

Figure 4: Performance summary across scenarios and tracks. Box plots show distribution of average progress per episode (top row) and average lap time for completed laps (bottom row). Tracks are ordered (left to right within each plot): A-Z Speedway, reInvent2019 Track, Empire City Track.

greatest difficulty, particularly with opponents. This demonstrates that while the training regime produced a competent solo racer, it failed to equip the agent with the robustness needed for more complex, realistic racing scenarios.

6 Discussion

The results demonstrate successful learning for the time-trial task across multiple tracks but reveal significant limitations when generalizing to scenarios with obstacles or dynamic opponents. This performance degradation highlights critical challenges related to the training methodology and the inherent complexities of reinforcement learning generalization.

1. *Training-Evaluation Mismatch and Lack of Task Generalization:* The primary reason for poor performance in obstacle avoidance and head-to-head modes is a fundamental mismatch between training and evaluation conditions. Our curriculum, while varying tracks and start conditions, *exclusively featured single-agent, obstacle-free driving.* Consequently, the learned policy likely overfit to this narrow task distribution (Zhang et al., 2018). Introducing obstacles or other agents at evaluation time constituted a significant domain shift for which the agent was unprepared (Cobbe et al., 2019).

   This gap manifested clearly: the agent often failed to identify or react appropriately to obstacles despite LIDAR input and penalty terms, suggesting learned visual features did not prioritize hazard identification. Similarly, in multi-agent scenarios, the agent operated purely reactively, lacking the ability to anticipate opponent actions or plan strategically (e.g., for overtaking). This stems from framing the problem as a single-agent POMDP, which is insufficient for the non-stationary dynamics introduced by other agents (Albrecht et al., 2024; Lowe et al., 2017). The absence of opponent modeling capabilities (Carroll et al., 2019; Albrecht et al., 2024) or training exposure to interactive scenarios meant the agent lacked policies for negotiation or complex avoidance maneuvers. Effectively, the training lacked the task diversity necessary for robust

generalization; achieving robustness requires training curricula that encompass the variations expected during deployment (Tobin et al., 2017; Cobbe et al., 2019).

2. *Potential Conflicts and Credit Assignment Issues in Reward Design:* While providing dense feedback, the multi-term shaped reward function may have inadvertently hindered adaptation. Conflicting incentives could arise—for example, between maintaining centerline proximity (highly rewarded) and deviating to avoid an obstacle (incurring a temporary penalty). Heuristic tuning of reward weights might not have optimally balanced competing objectives, especially given the temporal credit assignment problem (Sutton & Barto, 2018), where linking delayed consequences (collisions) to earlier actions is difficult. Overly complex shaping can potentially lead to suboptimal local optima (Sutton & Barto, 2018), where maximizing parts of the reward hinders the overall goal of safe completion. Simpler or adaptive reward structures might be necessary for robustly learning complex behaviors like obstacle avoidance or strategic overtaking.
3. *Limitations from Training Budget and Hyperparameter Tuning:* Practical constraints likely impacted the agent's capabilities. The relatively short training duration (20k timesteps, compared to millions often used in complex deep RL tasks (Mnih et al., 2015; Schulman et al., 2017)) might mean the policy was under-trained, particularly for adapting to the most challenging curriculum stages. Furthermore, using baseline PPO hyperparameters and a fixed network architecture without extensive tuning (cf. (Akiba et al., 2019)) might have limited performance. Suboptimal learning rates, entropy weights, or network capacity could affect learning speed, stability, and the ability to capture behaviors required for challenging tasks. While our approach yielded strong time-trial results, further optimization of training time and hyperparameters could potentially improve performance and generalization.

While combining PPO, curriculum learning, and multi-modal inputs produced a competent solo racer generalizable across track geometries, the agent's capabilities proved brittle. Failures in complex scenarios primarily stem from insufficient task diversity during training, potentially compounded by reward conflicts and resource limitations. This underscores the critical need for training regimes that mirror the target deployment's complexity and variability, especially when transitioning from single-agent to interactive or hazard-rich settings (Cobbe et al., 2019). The positive generalization in time-trials suggests, however, that extending the curriculum's task diversity remains a promising direction for future work.

7 Future Work

Several promising directions exist to address the limitations identified and enhance the agent's capabilities:

1. *Enhance Training Curriculum and Multi-Task Learning:* The most critical step is to broaden the training distribution. Incorporating obstacle avoidance and multi-agent interaction scenarios (potentially via self-play, training against scripted bots, or using techniques from multi-agent RL (Albrecht et al., 2024)) directly into the curriculum is essential for developing robust policies (Cobbe et al., 2019; Narvekar et al., 2020). This includes varying obstacle placements and opponent behaviors.
2. *Simplify or Adapt Reward Function:* Conduct ablation studies to assess the impact of individual reward terms. Exploring simpler reward structures (e.g., focusing primarily on progress and safety penalties) or adaptive reward weighting mechanisms could potentially mitigate conflicting incentives and improve learning in complex scenarios.
3. *Incorporate Opponent Modeling / MARL Techniques:* Explicitly address the multi-agent challenge by adopting techniques like centralized training with decentralized execution (CTDE) (Lowe et al., 2017; **?**; Sunehag et al., 2017) or training separate opponent prediction models (potentially using recurrent networks like LSTMs (Sutton & Barto, 2018)) to inform the agent's policy, enabling more strategic interactions.
4. *Optimize Training Process:* Extend training durations and perform systematic hyperparameter tuning (learning rates, entropy coefficients, network architecture), potentially using automated

methods (Akiba et al., 2019), to ensure the agent reaches its full potential within the PPO framework.

5. *Investigate Sim-to-Real Transfer:* For physical deployment, focus on sim-to-real transfer, likely requiring enhanced domain randomization (visuals, physics) during simulation (Tobin et al., 2017) and potentially simpler reward functions robust to real-world noise and sensor limitations.

By pursuing these avenues, particularly enhancing curriculum task diversity and adopting multi-agent learning strategies, future work can build upon our findings to develop DeepRacer agents proficient across a wider spectrum of competitive racing challenges.

8 Conclusion

This study demonstrated the application of deep reinforcement learning, specifically Proximal Policy Optimization (PPO) combined with curriculum learning, to train an autonomous agent for the AWS DeepRacer platform using multi-modal (stereo vision, LIDAR) sensory inputs. Our approach successfully produced an agent capable of achieving fast and reliable lap times in solo time-trial scenarios across multiple tracks, confirming the effectiveness of PPO and track-geometry-focused curriculum learning for acquiring basic autonomous racing skills. The shaped reward function proved viable for guiding efficient navigation in this context.

However, evaluation across diverse task modes revealed significant generalization limitations. The agent, trained exclusively in a single-agent, obstacle-free setting, struggled markedly when faced with static obstacles or dynamic opponents. This highlights a critical gap between the training regime and the requirements of more complex, interactive scenarios. Our key finding underscores that effective curriculum learning must encompass *task diversity* (including hazards and interactions), not just environmental variation (like tracks and start points). Omitting crucial complexities from training resulted in a policy ill-equipped for them. This work provides an empirical analysis quantifying this generalization drop, emphasizing the need to align training complexity with deployment expectations in RL for autonomous systems. While the shaped reward aided the primary task, its complexity might warrant investigation regarding potential suboptimal behavior in avoidance tasks. Ultimately, our results emphasize that robust autonomous agents require training experience reflecting the full spectrum of anticipated operational challenges.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024. URL https://www.marl-book.com.

Janakiram Balaji and Oren Etzioni. Bridging simulation and reality: Lessons from the DeepRacer platform. In *Proceedings of the 2020 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pp. 45–52, 2020.

Janakiram Balaji, Shane Ross, and Sachin Tangirala. AWS deepracer: Accelerating reinforcement learning through autonomous-scale racing. In *NeurIPS 2019 Workshop on Machine Learning for Autonomous Driving*, 2019. URL https://drive.google.com/file/d/1Yc76l5h7y1dFazw6VvVIBd90ZNt27XJL. Extended Abstract.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL https://arxiv.org/abs/1606.01540.

Micah Carroll, Rohin Shah, Mark K. Ho, Thomas L. Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination, 2019. URL https://arxiv.org/abs/1910.05789.

Karl Cobbe, Oleg Klimov, Christopher Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 1282–1289, 2019.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, and et al. Continuous control with deep reinforcement learning. *arXiv preprint*, arXiv:1509.02971, 2016. URL https://arxiv.org/abs/1509.02971.

Ryan Lowe, Yi Wu, Aviv Tamar, and et al. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pp. 6379–6390, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, and et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. doi: 10.1038/nature14236.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey, 2020. URL https://arxiv.org/abs/2003.04960.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, 2016. URL https://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint*, arXiv:1707.06347, 2017. URL https://arxiv.org/abs/1707.06347.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning, 2017. URL https://arxiv.org/abs/1706.05296.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 2nd edition, 2018.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017. doi: 10.1109/IROS.2017.8202133.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2094–2100, 2016.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, and Benjamin Recht. A study on overfitting in deep reinforcement learning. *arXiv preprint*, arXiv:1804.06893, 2018. URL https://arxiv.org/abs/1804.06893.

A Available Repository

All code and supplementary materials used in this research are available in a GitHub repository. The repository can be accessed at: https://github.gatech.edu/gt-omscs-rldm/7642RLDMSpring2025kodendaal3 where the latest commit hash is: *504c10665d66b4abb92f30c06eb3459f987f3173*