Curriculum Leaning For Autonomous Vehicles

Luc Chen lucchen@g.harvard.edu
Michelle Chang mchang@g.harvard.edu
Theo Lebryk tlebryk@fas.harvard.edu

Abstract

This study investigates how the sequence of training environments affects performance in simple driving tasks for an autonomous driving agent. By training agents solely through interaction with maps of varying difficulty, we demonstrate that transfer learning enhances performance within single-environment driving scenarios. However, we find that agents struggle to master advanced driving capabilities and fail to generalize well to new environments, regardless of the sequence of training data. We conclude by looking at areas to build on this work such by combining imitation learning with curriculum learning and developing curriculum-specific MDP. Our experimental code and preliminary data are available at https://github.com/michellechang77/rl-final-project.

1 Introduction

A crucial task for autonomous vehicles (AV) is to navigate through traffic smoothly while getting to their desired goal location. Many state-of-the-art approaches applying reinforcement learning (RL) to autonomous driving from the last 5 years rely heavily on imitation learning, which requires expensive, often proprietary data [4, 19, 12]. Rather than having expert humans directly instill behaviors in agents, we investigate the use of curriculum learning [3] whereby agents learn merely through interacting with curated training environments. We use transfer learning to refer to the literal process of obtaining knowledge from previous environments, whereas curriculum learning refers to the broader objective of optimizing the order of knowledge acquisition. The promise of curriculum learning is that cleverly ordered training inputs can make algorithms converge faster and at better ultimate performance [14].

2 Related Work

End-to-End RL in Autonomous Driving. A growing body of research has focused on training end-to-end RL policies directly from raw sensory inputs in simulation environments such as CARLA [5], TORCS [22], or MetaDrive [11]. These methods aim to learn driving strategies purely through trial and error, guided by reward signals that encapsulate safety, efficiency, and traffic rule compliance [9]. While RL-based approaches have demonstrated impressive results in navigating simple driving tasks, scaling to more complex, real-world-like scenarios remains challenging due to sparse rewards, high-dimensional state spaces, and the need for robust generalization.

Curriculum Learning for RL Most curriculum learning algorithms function by progressing from simpler tasks to increasingly difficult ones [14]. By structuring the learning process, agents can first acquire basic skills in controlled settings before gradually tackling more complicated scenarios. This approach has seen success in robotics [18], natural language processing [10], and complex strategy games, where progressively introducing difficulty helps the agent discover useful policies that would be intractable to learn from scratch in a fully challenging environment.

Preprint. Under review.

In the context of autonomous driving, curriculum learning has begun to gain traction. For instance, Anzalone et al. [2, 1] explored the use of curricula to refine driving policies learned through RL, incrementally introducing new obstacles and road conditions. Despite showing that agents can improve their capabilities with structured training progressions, these studies also highlighted persistent issues, such as difficulty in reliably recognizing and reacting to obstacles and maintaining smooth acceleration and braking profiles.

3 Dynamic Curriculum Learning

Traditional fixed curriculum approaches specify a predetermined progression of difficulty levels and training durations. While such methods can guide agents toward incrementally more challenging tasks [14], they often lack adaptability. As a result, when the agent learns more slowly or more quickly than anticipated, fixed curricula may not provide optimal conditions for skill acquisition. By contrast, we employ a dynamic curriculum learning approach that adjusts the progression thresholds based on the agent's real-time performance. Specifically, the agent advances to a more difficult level only when it meets or exceeds a performance threshold, such as a rolling average of episode rewards. This threshold is dynamically adapted, taking into account the agent's progress in previous levels [7]. If the agent struggles to improve, the model allocates additional training time at the current difficulty level, ensuring adequate skill acquisition before progressing. Conversely, if the agent demonstrates rapid improvement, it transitions earlier, avoiding redundant training on tasks it has already mastered. This flexibility ensures that the curriculum aligns with the agent's learning curve, fostering more efficient skill acquisition and better generalization across tasks [3, 13]. While the dynamic curriculum setting is not the main focus of this paper, we briefly implemented it to confirm that our findings generalized outside of the more rigid setting of our main experimentation whereby the final model from the previous environment is the model used by the next environment, regardless of performance at previous time steps.

4 MDP

In order to investigate the above problem, we set up the following MDP.

State Space

4.1 Vehicle State

1. **Position** v_pos = $(x, y) \in [-\infty, \infty]$: Cartesian coordinates of vehicle on the map.
2. **Heading Angle** $v\_theta \in [-\pi, \pi]$: Orientation of the ego vehicle, represented as an angle in radians.
3. **Linear Velocity** $v\_velocity \in [0, 120]$: Speed of the ego vehicle in km/h.
4. **Angular Velocity** $\omega \in \mathbb{R}$: Rotational speed of the ego vehicle.
5. **Steering Angle** $S \in [-1, 1]$: Indicates the current steering angle of the vehicle, where -1 is full left and 1 is full right.
6. **Acceleration** $a \in [-5, 5]$: The rate of change of velocity of the vehicle, accounting for both throttle and braking.

4.2 Environmental State

7. **Lidar Distances** $L_i \in [0, 50]$: A 1D array of distances to the nearest obstacles (e.g., vehicles, road boundaries), where $L_i = 50$ indicates no obstacle within range.

4.3 Traffic State

8. **Collision Status** $C \in \{0, 1\}$: Binary indicator of whether the ego vehicle has collided with another object or vehicle.

Combined State Space

$$S = [\text{v\_pos}, \text{v\_theta}, \text{v\_velocity}, \omega, S, a, L_i, C]$$

Action Space

The action space in MetaDrive controls the vehicle's movement through steering and throttle/brake inputs. This is a continuous 2D space defined as:

1. **Steering Angle** ($A_{\text{steering}} \in [-1, 1]$): Controls the turning angle of the front wheels of the vehicle.
   - $A_{\text{steering}} = -1$: Maximum left turn.
   - $A_{\text{steering}} = 1$: Maximum right turn.
   - $A_{\text{steering}} = 0$: Driving straight.
2. **Throttle/Brake** ($A_{\text{throttle}} \in [-1, 1]$): Controls the vehicle's acceleration or deceleration.
   - $A_{\text{throttle}} > 0$: Throttle (positive values accelerate vehicle).
   - $A_{\text{throttle}} < 0$: Brake (negative values decelerate vehicle).
   - $A_{\text{throttle}} = 0$: No throttle/brake.

Combined Action Space

$$A = [A_{\text{steering}}, A_{\text{throttle}}], \quad A \in [-1, 1]^2$$

4.4 Reward Function

For our first set of experiments, we decided to explore a reward function that includes a few key parameters that we believe are key in terms of optimizing our agent's trajectory in the state space. This includes aspects for safety, efficiency, comfort and rule compliance.

The total reward is a weighted combination of these components:

$$R = -\alpha \cdot \text{safety} + \beta \cdot \text{efficiency} - \gamma \cdot \text{comfort} + \nu \cdot \text{rule\_compliance}$$

where:

- $\alpha, \beta, \gamma, \nu$ are tunable weight parameters for the reward components.
- R is the total reward at each step.

Initial Reward Function

1. **Safety Penalty**:

   $$\text{safety} = v_{\text{crash\_vehicle}} + v_{\text{crash\_object}} + v_{\text{crash\_sidewalk}}$$

   The safety term penalizes the agent for collisions with other vehicles, objects, or driving onto sidewalks. A higher value indicates worse safety performance.
2. **Efficiency Reward**:

   $$\text{efficiency} = \frac{v_{\text{velocity}}}{v_{\text{max}}}$$

   Efficiency is calculated as the ratio of the vehicle's current speed (v_{velocity}) to its maximum possible speed (v_{max}). This term encourages the agent to maintain higher speeds without compromising safety.
3. **Comfort Penalty**:

   $$\text{comfort} = |A_{\text{steering}}|$$

   Comfort is inversely related to abrupt steering actions (s_{steering}). Minimizing the absolute steering angle ensures smoother driving behavior.
4. **Rule Compliance Reward**:

   $$\text{rule\_compliance} = \begin{cases} 0 & \text{if the vehicle is out of the road,} \\ 1 & \text{otherwise.} \end{cases}$$

   This binary term rewards the agent for staying on the road and penalizes rule violations such as driving off-road.

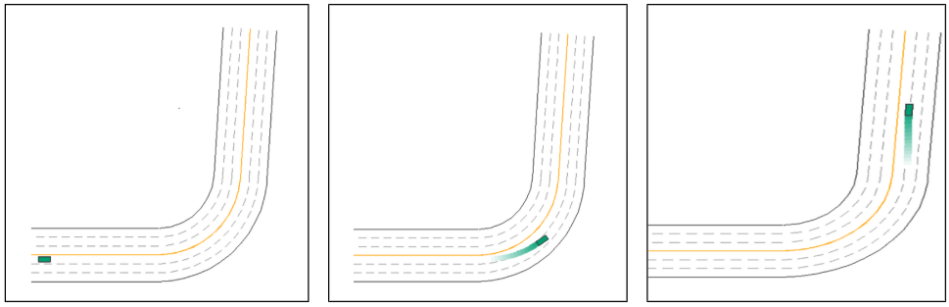


Figure 1: Graphic of our agent's learned trajectory in an easy environment

Given the promising increasing rewards observed after our initial trial run using this reward function, we decided to leverage more parameters in MetaDrive to enhance the reward function and better train our agent.

Enhanced Reward Function

We decided to evaluate the agent's driving behavior based on three main components: driving reward, speed reward, and termination reward to achieve efficient, safe, and rule-compliant driving (refer to Appendix A for graphs).

1. **Driving Reward**: To encourage the agent to move toward the destination by measuring progress along the reference lane.

$$R_{\text{driving}} = \text{driving\_reward} \cdot (\ell_{\text{now}} - \ell_{\text{last}}) \cdot \text{positive\_road}$$

   where:

   - ℓ_{now} and ℓ_{last}: coordinates of vehicle at the current and previous time steps.
   - positive_road $\in \{-1, 1\}$: Indicates whether the vehicle is on a positive or negative road.

2. **Speed Reward**: Incentivizes agent to drive faster, normalized by the maximum speed:

$$R_{\text{speed}} = \text{speed\_reward} \cdot \frac{v_{\text{velocity}}}{v_{\text{max}}}$$

   where:

   - v_{velocity}: Current velocity of vehicle.
   - v_{max}: Maximum velocity of vehicle (80 km/h by default).

3. **Termination Reward**: At the end of an episode, a reward is given based on the termination condition:

$$R_{\text{termination}} = \begin{cases} \text{success\_reward}, & \text{if vehicle reaches the destination,} \\ -\text{out\_of\_road\_penalty}, & \text{if vehicle goes off-road,} \\ -\text{crash\_vehicle\_penalty}, & \text{if vehicle crashes with another vehicle,} \\ -\text{crash\_object\_penalty}, & \text{if vehicle crashes with an object,} \\ -\text{crash\_sidewalk\_penalty}, & \text{if vehicle crashes into a sidewalk.} \end{cases}$$

Full Reward Function

$$R = R_{\text{driving}} + R_{\text{speed}} + R_{\text{termination}}$$

4.5 Defining Required Performance for Each Difficulty

To determine when to advance from difficulty level $d-1$ to difficulty level d, we establish a required performance threshold $R_{\text{req}}^{(d)}$. Inspired by reward thresholding [16] used to determine if a student has

Figure 2: 3-D rendering of the environment

mastered a class, we set this threshold relative to the previously achieved performance $R_{\text{prev}}^{(d-1)}$ plus an improvement margin M:

$$R_{\text{req}}^{(d)} = R_{\text{prev}}^{(d-1)} + M.$$

This ensures that progress within the curriculum is contingent upon genuine improvement, maintaining upward pressure on skill acquisition. The margin M can be tuned to balance stability and the rate at which difficulty increases.

4.6 Computing Recent Performance (Rolling Average)

A key component of our adaptive methodology is a stable measure of recent performance. After each training increment, we compute a rolling average over the last W episodes:

$$R_{\text{recent}} = \begin{cases} \frac{1}{W} \sum_{i=n-W+1}^{n} r_i & \text{if } n \geq W \\ \frac{1}{n} \sum_{i=1}^{n} r_i & \text{if } n < W \end{cases}$$

Here, r_i denotes the reward from the i-th episode, n is the total number of logged episodes, and W is the performance window, defining how many recent episodes are considered. When $n \geq W$, the metric focuses on the latest W episodes, emphasizing current trends, while for $n < W$, it averages over all available episodes to maintain accuracy in early training. If no episodes are logged ($n = 0$), R_{recent} is initialized to $-\infty$, signifying the absence of performance data. This rolling average smooths fluctuations caused by noise or outliers, offering a reliable measure of stable performance and facilitating informed curriculum adjustments.

4.7 Mastery Check

Following each increment, we determine whether the agent has mastered the current difficulty level by comparing R_{recent} with $R_{\text{req}}^{(d)}$:

$$R_{\text{recent}} > R_{\text{req}}^{(d)} \implies \text{difficulty } d \text{ mastered.}$$

This mastery check ensures that transitions to higher difficulty levels only occur once the agent has demonstrated sufficient competency at the current level. Such dynamic checks align with progressive curriculum strategies that avoid premature exposure to overly complex tasks [21].

4.8 Incrementing Training Steps per Cycle

When the agent fails to meet $R_{\text{req}}^{(d)}$ at a given increment, we increase the training steps for the next increment by a factor F:

$$T_{\text{next}} = T_{\text{current}} \cdot F.$$

This adaptive time allocation provides the agent with additional opportunities to refine its policy before attempting to pass the mastery check again. Similar concepts have appeared in automated curriculum scheduling, where pacing and patience are key to steady improvement [8].

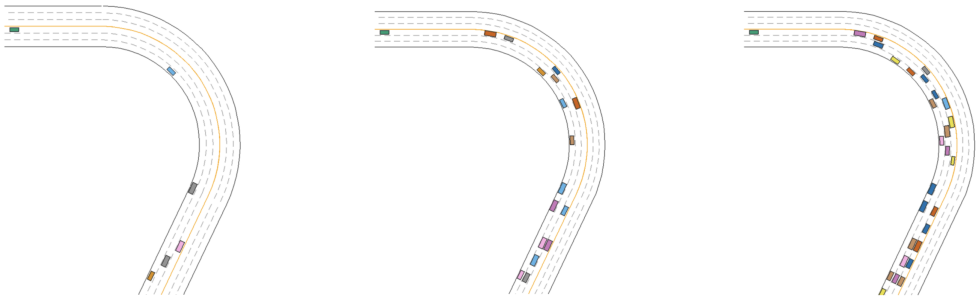
4.9 Stopping Criteria

We define two stopping criteria to maintain efficient use of computational resources and avoid indefinite training plateaus:

- **Global Time step Limit:** If the total time steps T_{total} exceed a predefined maximum T_{max}, training ceases:
$$T_{\text{total}} > T_{\text{max}} \implies \text{stop training.}$$
This ensures that training does not continue indefinitely with diminishing returns [6].
- **Max Increments per Difficulty:** If the agent does not reach $R_{\text{req}}^{(d)}$ after K increments, we proceed to the next difficulty level:
$$\text{If increments used} \geq K, \text{ advance to next difficulty.}$$
This prevents getting stuck on a particular difficulty level and encourages broader coverage of the curriculum, even if not perfectly mastered.

Overall, this dynamic curriculum learning methodology offers a flexible and data-driven framework, better attuned to the agent's performance trajectory than static schedules. It aims to foster continuous improvement, efficient skill acquisition, and stronger policy generalization in increasingly challenging autonomous driving tasks.

5 Methodology



(a) Easy Map (b) Medium Map (c) Hard Map

Figure 3: Comparison of easy, medium, and hard maps.

In order to evaluate the learning progression of autonomous driving agents under a curriculum learning framework, we designed a controlled simulation environment using the MetaDriveEnv platform [11]. We train our agent using PPO [20] using two layer networks for both the actor and critic [17].

We found our agent faced difficulty mastering even a single map, so we concentrated most of our efforts training and evaluating on a single map. We refer to this default map as Map C (curve), The majority of our experiments are trained on a single map with varying degrees of difficulty, as shown in (Figure B).

To get some insight into the generalizability of our approach, we also run and report on several experiments where we vary the training maps (referred to as maps 2, 3, and 4). We focus on just turns and bends and leave out navigation of ramps, forks, parking lots etc. We evaluate on either a new initialization of Map C or a holdout map (map 5) never seen in training. We evaluate on 10 episodes and report on average reward cumulative reward. In practice, the variance in this reward for a given model is minuscule: almost every agent crashes or reaches the destination within less than a second of one another across the 10 episodes.

We start with a low-difficulty scenario ("easy") characterized by minimal traffic density (0.1) and fixed lane widths. This allows the agent to focus on mastering basic navigation skills without significant environmental variability.

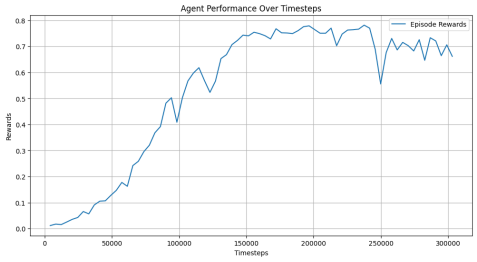
As the difficulty increases to “medium” and “hard,” traffic density rises to 0.3 and 0.5, respectively, simulating increasingly congested driving conditions. Additionally, random lane width variations are enabled for “medium” and “hard” levels, introducing structural diversity across episodes to test the agent’s robustness against geometric variability even within an overarching single map.

We train our models for a cumulative 300,000 steps, except for the dynamic curriculum learning agent. By default, we evenly divide the training steps between easy, medium, and hard environments in our curriculum learning set up. By default, our non-transfer learning set up trains all 300,000 steps on a single environment.

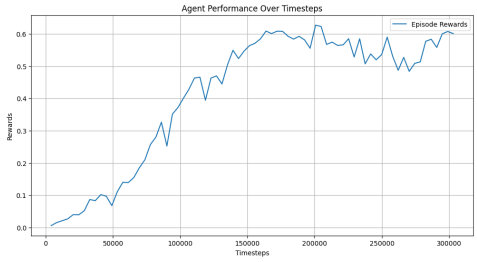
As sanity checks, we ran number of additional experiments, some of which are documented in the Appendices. We experimented with arbitrary sequences (hard-medium-easy, medium-easy-hard) and found results were consistently worse than the logical order (easy-medium-hard). We also ran a 600,000 step run for the transfer and non-transfer learning settings as an ablation measure. We found no substantial improvement over the 300,000 step runs and at double the training time.

6 Result and Analysis

6.1 Training results



(a) Performance for Medium Level (300,000 Time steps)



(b) Performance for Hard Level (300,000 Time steps)

Figure 4: Agent performance across medium and hard difficulty levels with non transfer learning, each trained for 300,000 time steps per level, Reward is averaged per step.

(a) Performance for Medium Level with Curriculum Learning.

(b) Performance for Hard Level with Curriculum Learning.

Figure 5: Agent performance across medium and hard difficulty levels with curriculum learning, each trained for 100,000 time steps per level. The medium agent was previously trained on 100,000 time steps in an easy environment and the hard agent was previously trained for 200,000 time steps in easy and medium environments.

Figure 4 and 5 show the training curves with and without transfer learning respectively. Rather than starting at zero reward as with the no transfer learning agent, the agent with transfer learning "jump-starts" to 0.9 reward, likely having already learned the basic driving skills in the easy and medium environments. The agent with no transfer learning *never* reaches the starting point of the agent with transfer learning. The agent with transfer learning ends up achieving 1.2 average reward, roughly twice that of the non-transfer learning agent. These gaps in jump-start and total reward are less pronounced, but still significant, in the medium setting.

All this comes from the same budget (300,000) of total steps. In terms of pure wall clock time, we found training an easy and medium step are substantially faster than a hard step. An easy step and medium step were 80 percent and 35 percent faster training on a T4 GPU than a hard step. Thus, the transfer learning setting (100,000 easy, 100,000 medium, 100,000 hard) cost 60 percent less than the pure hard setting.

Table 1: Holdout Rewards for 300,000 Training Steps On Maps C

Holdout Map	Difficulty Level	Cumulative No Transfer Reward	Cumulative Transfer Reward	Transfer Delta
C	Easy	118.18	**146.75**	28.58
	Medium	**48.12**	47.88	-0.23
	Hard	**55.07**	53.51	-1.56
5	Easy	122.40	**201.88**	79.48
	Medium	**44.98**	36.94	-8.04
	Hard	**38.07**	37.86	-0.21

Table 1 shows holdout rewards when training on different maps at different difficulties, not including Map 5.

Table 2: Average Reward Per Time Step for Medium and Hard Difficulty Levels at Different Time steps

Difficulty Level	Time steps	Non-Transfer Learning	Curriculum Learning	Dynamic Curriculum
Medium	100,001	0.4546	**0.4712**	0.3726
	150,000	**0.7375**	0.5325	N/A
	200,000	0.7553	**0.9456**	N/A
Hard	200,001	0.6186	0.9359	**0.9764**
	250,000	0.5541	1.0700	**1.1123**
	300,000	0.6012	**1.2200**	1.1245

Note: The time steps shown for curriculum learning include training in the easy mode. Results for medium difficulty levels in the dynamic curriculum column are not available (N/A) for time steps 150,000 and 200,000 due to transitions to hard difficulty mode. Bolded values indicate the highest performance for each row.

Despite the promising training reward of the transfer learning agent, we find that our agent is never able to completely master the hard settings and reach the target destination. The agent makes it far closer to the target by the end of training, but each time fails to brake before hitting another vehicle or accidentally veers off the road to avoid a another car, findings in line with prior research [2]. What's more, when evaluated on a holdout environment never seen before, performance at the medium and hard difficulties are equally poor regardless of whether transfer learning is used.

Encouragingly, however, performance does diverge when evaluating on held-out easy environments. This trend may seem obvious: a model which has never seen an easy environment will perform worse on easy environments at test time. However, there is no guarantee that over the course of the curriculum, the agent doesn't get so confused by the difficult environments that it forgets skills it previously learned. In this case, the agent never fully forgets how to navigate a simple traffic bend with no surrounding traffic. When training exclusively on the hard environment, the agent never even learns this basic skill. Visualizations of the non transfer learning models trained exclusively on the harder difficulty show that it simply drives off road in the easy environment even when there is no traffic.

7 Conclusion and Future Work

We found that curriculum learning can improve performance on progressively complex training environments. By offering progressively difficult single task environments in an autonomous driving setting, we were able to nearly double performance on a hard environment in less than half the wall-clock training time.

However, we find that these encouraging training results fail to generalize to new environments. Despite the increase in total training reward, agents trained purely using curriculum learning in a simulator fail to achieve mastery of essential driving techniques. Even with using curriculum learning

to speed up training times, training agents to execute basic driving techniques takes hours of training time and likely overfits to the environment based on our findings.

There is still plenty of room to sprinkle in other ways to improve performance. Our agent faced known challenges such as failing to brake for traffic. We briefly attempted to use custom reward functions to combat these known challenges, but found the model to be highly sensitive to even small changes and often reacted in unintended ways. In short, optimizing custom reward functions for individual curriculum to learn these types of behavior would have required a good deal of trial and error. One interesting area of research known as curriculum MDP involves treating the curriculum itself as an MDP, allowing for an agent to metalearn the state, reward, and action space of a curriculum [15]. Given the robust state space of potential AV curricula, we think this would be a particularly interesting way to address the challenges faced in this research. Finally, while this paper set out to train AV agents relying only on task sequencing and not on expert labels, a hybrid approach whereby experts are more strategically used at different points in training could combine our encouraging findings around curriculum learning with the known power of imitation learning.

References

[1] L. Anzalone, P. Barra, S. Barra, A. Castiglione, and M. Nappi. An end-to-end curriculum learning approach for autonomous driving scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):19817–19826, 2022.

[2] L. Anzalone, S. Barra, and M. Nappi. Reinforced curriculum learning for autonomous driving in carla. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3318–3322. IEEE, 2021.

[3] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[4] F. Codevilla, E. Santana, A. M. López, and A. Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9329–9338, 2019.

[5] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[6] G. Farquhar, S. V. Albrecht, and S. Whiteson. Tiered reward-driven exploration for reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 1966–1976. PMLR, 2019.

[7] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pages 1311–1320. Pmlr, 2017.

[8] M. Jiang, E. Grefenstette, and T. Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pages 4940–4950. PMLR, 2021.

[9] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2021.

[10] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

[11] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, and B. Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, 45(3):3461–3475, 2022.

[12] X. Liang, T. Wang, L. Yang, and E. Xing. Cirl: Controllable imitative reinforcement learning for vision-based self-driving. In *Proceedings of the European conference on computer vision (ECCV)*, pages 584–599, 2018.

[13] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

[14] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

[15] S. Narvekar, J. Sinapov, and P. Stone. Autonomous task sequencing for customized curriculum design in reinforcement learning. In *IJCAI*, pages 2536–2542, 2017.

[16] R. Portelas, C. Colas, K. Hofmann, and P.-Y. Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *Conference on Robot Learning*, pages 835–853. PMLR, 2020.

[17] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[18] K. Ryu, Q. Liao, Z. Li, K. Sreenath, and N. Mehr. Curricullm: Automatic task curricula design for learning complex robot skills using large language models. *arXiv preprint arXiv:2409.18382*, 2024.

[19] A. Sauer, N. Savinov, and A. Geiger. Conditional affordance learning for driving in urban environments. In *Conference on robot learning*, pages 237–252. PMLR, 2018.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[21] D. Weinshall, G. Cohen, and D. Amir. Curriculum learning by transfer learning: Theory and experiments with deep networks. In *International conference on machine learning*, pages 5238–5246. PMLR, 2018.

[22] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner. Torcs, the open racing car simulator. *Software available at http://torcs. sourceforge. net*, 4(6):2, 2000.

A Appendix: Transfer learning with initial reward function on 10k steps

Figure 6: Easy map 10,000 total time steps

Figure 7: Medium map 10,000 total time steps

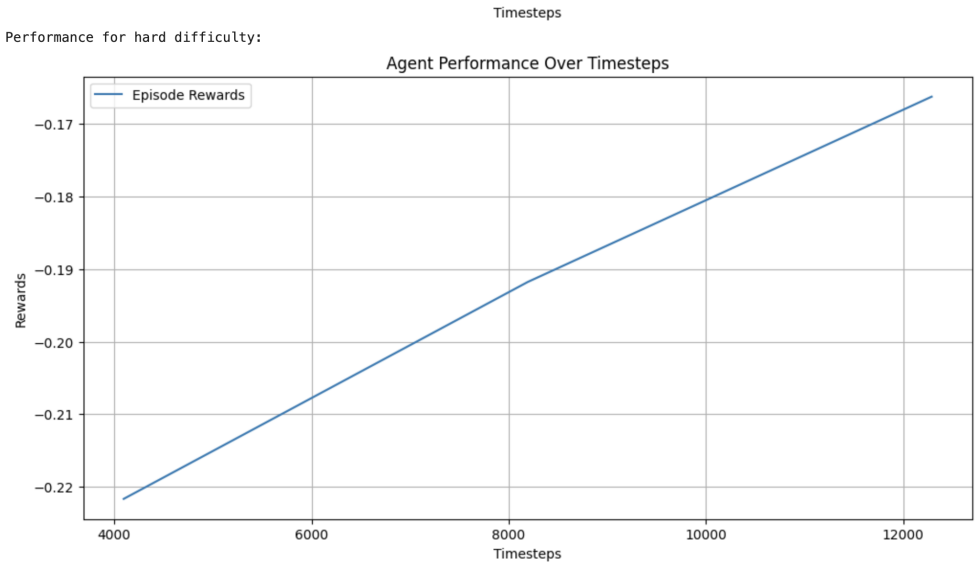


Figure 8: Hard map 10,000 total time steps

B Appendix: 600,000 Time Step Transfer Learning Experiment

We trained an additional model using 200,000 time steps per difficulty level across different maps (maps 2,3, and 4). We found that the rewards did increase for the easy and medium agents after the 100,000 step, but did not overtly help out the transfer learning portion of the experiment. The hard map started to crater after 100,000 time steps, This finding suggests the agents might overfit at 600,000 total times steps per environment. Due to that preliminary finding and the long compute times, we stuck with 300,000 time steps per trial.

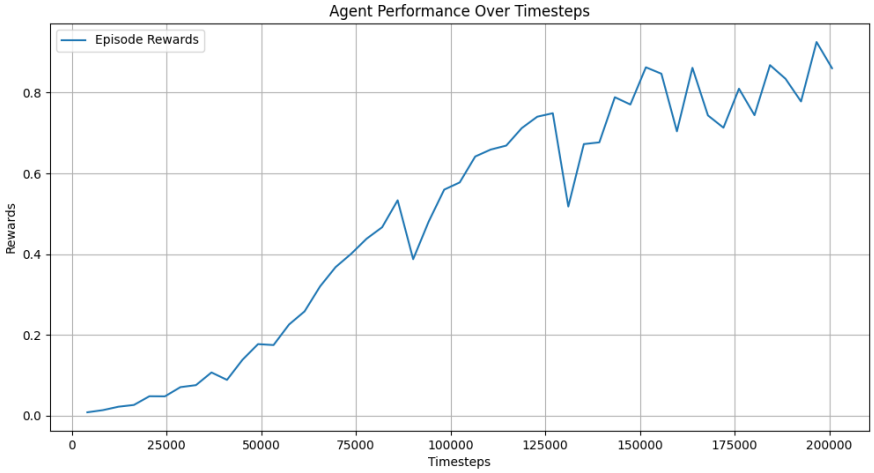


Figure 9: Easy map 600,000 total time steps

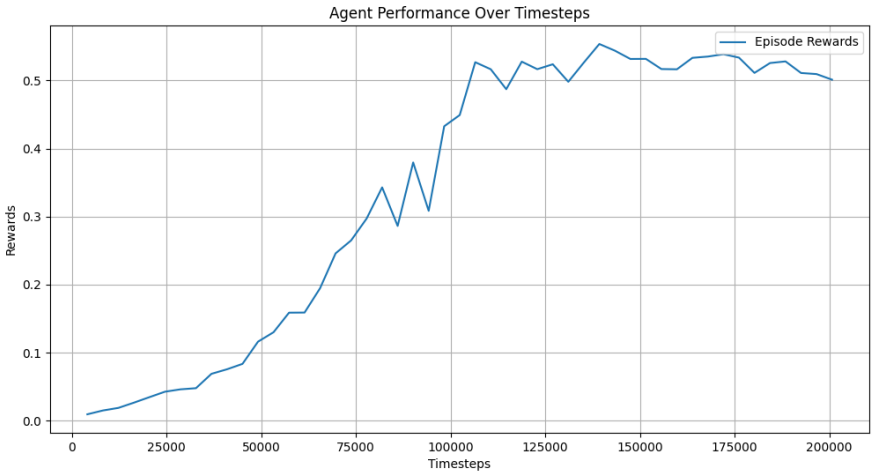


Figure 10: Medium map 600,000 total time steps

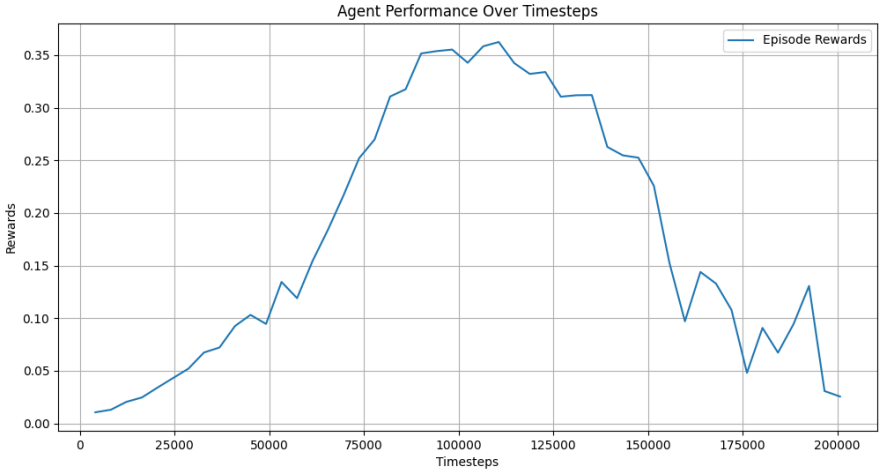


Figure 11: Hard map 600,000 total time steps