Zero-Shot Amazing Maze Navigation

Dean Solomon, Benjamin Rosman, Steven James
University of the Witwatersrand

***Abstract*—As robots are expected to navigate autonomously in unfamiliar environments, the need for efficient zero-shot learning approaches in reinforcement learning (RL) becomes critical. This research explores curriculum learning (CL) as a means to improve RL agent adaptability in maze navigation tasks. Agents trained using the Twin Delayed Deep Deterministic Policy Gradient (TD3) and Soft Actor-Critic (SAC) algorithms were exposed to curricula of increasing difficulty levels—progressing from easy to medium to hard environments. We evaluated agents' robustness, speed, and path completeness across test mazes. We observed that agents trained with our easy-to-medium curriculum achieved superior zero-shot performance compared to other curricula. This study demonstrates the promise of CL in RL, suggesting that future research expands on this approach for even more complex navigation scenarios.**

I. Introduction

As technology advances, robotics is expected to play an increasingly significant role in our lives, especially when integrated with machine learning (ML). A key aspect of robotics is navigation—how a robot can autonomously move and interact with its environment. For example, consider autonomous vehicles (Parekh et al., 2022) capable of driving without human intervention. This demonstrates that robotics has the potential to make our lives more convenient and enhance safety in critical situations.

Robotic navigation, however, remains an open problem (Mavrogiannis et al., 2023; Zhu and Zhang, 2021; Adamkiewicz et al., 2022). Before the advent and popularity of Reinforcement Learning (RL), Optimal Control Theory (OCT) was the only solution for addressing challenges in robotic navigation (Kirk, 2004). Yet, in recent years, RL — a sub-field of ML—has shown promise in overcoming certain limitations of OCT (Sutton, 2018). One of the main limitations of OCT is that it often relies on differential equations to model environments, which can be highly non-linear and may lack analytical solutions (Grass et al., 2008). Consequently, OCT typically produces numerical approximations, which can introduce error (Rao, 2009). Another issue that OCT can struggle with is problems where the environment has an underlying stochastic nature to it (Shaikhet, 2022).

Copyright Notice

Reinforcement Learning (RL) addresses this limitation by bypassing the need for explicit environmental modelling; instead, it learns the environment's dynamics implicitly through trial and error (Sutton, 2018). This research specifically focuses on the challenge of robotic navigation through mazes, where high-speed traversal and precision are crucial to avoid collisions with walls. Unlike research aimed at discovering global paths (paths from a starting point to a goal), our objective is to enable an agent to follow a predefined global path and optimize its local trajectory (e.g. improving efficiency in turns) as it progresses.

Navigation in unknown environments remains a fundamental challenge in robotics, demanding solutions that generalize beyond training scenarios. Traditional approaches often require extensive training data or environment-specific fine-tuning, making them impractical for real-world applications where robots encounter novel, unpredictable environments (Aradi, 2022). The ability to navigate unfamiliar spaces efficiently—without additional training—would significantly advance autonomous robotics applications across industries, from search and rescue operations to warehouse automation.

To achieve this, we explored the application of curriculum learning (CL) (Bengio et al., 2009) and Actor-Critic models (Grondman et al., 2012) to train an agent capable of fast, zero-shot maze navigation without fine-tuning. Zero-shot navigation is particularly advantageous, as it is impractical to train an agent on every possible maze configuration. Training a robot to handle unknown mazes is a more efficient use of resources than attempting exhaustive training across all possible scenarios.

We evaluated our models based on three metrics: Completeness, Robustness and Speed. Through these metrics, we demonstrate that CL, combined with Actor-Critic models, can produce an agent capable of navigating mazes at high speeds with minimal crashes. Our results show that not only does CL facilitate the development of a zero-shot maze navigator, but it also leads to better performance than training in a single environment without a curriculum that pre-trains in a simpler environment.

In Section II, we first introduce the research ques-

tion (Subsection II-A), followed by an exploration of Optimal Control Theory and Reinforcement Learning (Subsection II-B). Next, we discuss Actor-Critic models (Subsection II-C), followed by Curriculum Learning (Subsection II-D), and conclude with a review of related works (Subsection II-E).

In Section III, we begin with an overview of Markov Decision Processes (Subsection III-A), followed by a discussion of how Curriculum Learning was applied to our training (Subsection III-B). We then describe the simulator used (Subsection III-C) and cover the criteria used to evaluate our models (Subsection III-D). Lastly, we discuss Model Testing (Subsection III-E).

In Section IV, we begin by analyzing Model Robustness (Subsection IV-A), followed by Model Speed (Subsection IV-B). We then provide an overall evaluation (Subsection IV-D), before discussing limitations (Subsection IV-E) and Model Completeness (Subsection IV-C).

Finally, Section V covers our conclusions and outlines future work.

II. Background and Related Works

A. Research Question

How does Curriculum Learning impact the performance of Reinforcement Learning agents in zero-shot maze navigation tasks across different maze environments?

B. Optimal Control Theory and Reinforcement Learning

Navigation in robotics typically requires control systems for environmental interaction (Kirk, 2004). Optimal Control Theory (OCT) approaches this through the tuple (X, U, f, c, T), where X represents state space, U the control space, f the transition function ($x_{t+1} = f(x_t, u_t)$), c the cost function to minimize, and T the time horizon. OCT requires explicit environment and dynamics models (Kirk, 2004; Betz et al., 2022), which presents challenges due to real-world non-linearities and stochasticity (Sutton, 2018; Rao, 2009; Shaikhet, 2022). Reinforcement Learning (RL) addresses these limitations by learning without explicit models (Sutton, 2018). RL uses Markov Decision Processes (MDPs), defined by (S, A, P, R, γ), where S is state space, A action space, $P(s', r|s, a)$ the transition probability, R the reward function, and γ the discount factor. OCT and RL share structural similarities: X maps to S, U to A, f to P, and c to $-R$. Both T and γ consider future outcomes in decision-making. Our research implements Partial End-to-End RL control (Betz et al., 2022), preprocessing sensor data before agent decision-making. Using Deep RL (Li, 2017), neural networks serve as function approximators (Sutton, 2018), combining traditional OCT planning and control components into a single policy network.

C. Actor-Critic Models

Actor-Critic models are a type of RL model where an agent learns both a policy function (referred to as the actor) and a value function (referred to as the critic) (Grondman et al., 2012). In these models, the critic evaluates state-action pairs by learning the Q-value function, which informs the actor in selecting actions that maximize expected rewards in a given state. This model structure is inspired by the Deterministic Policy Gradient (DPG) (Silver et al., 2014) and its deep RL extension, the Deep Deterministic Policy Gradient (DDPG) (Lillicrap, 2015).

In our research, we employed two advanced Actor-Critic algorithms to train our models: Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018) and Soft Actor-Critic (SAC) (Haarnoja et al., 2018). Both of these algorithms build upon DDPG, and they differentiate from each other primarily in their respective loss functions. Each algorithm has two loss functions: one for the actor-network and one for the critic network.

The loss function for the TD3 critic is defined as:

$$L_Q = \frac{1}{N}\sum_i^N (Q_{\theta_1}(s_i, a_i) - y_i)^2 + (Q_{\theta_2}(s_i, a_i) - y_i)^2$$

where $y_i = r_i + \gamma \min(Q_{\theta'_1}(s'_i, a'), Q_{\theta'_2}(s'_i, a'))$. Here, N represents the mini-batch size, and $a' = \pi_{\phi'}(s') + \epsilon$, which is the action selected by the target actor network for the next state s', with added noise ϵ for stability. The parameters θ'_1 and θ'_2 are the parameters for the target Q networks, and ϕ' are the parameters for the target actor-network. γ is the discount factor. (Fujimoto et al., 2018)

The TD3 actor's loss function is expressed as:

$$L_\pi = -\frac{1}{N}\sum_i^N Q_{\theta_1}(s_i, \pi_\phi(s_i))$$

where π_ϕ denotes the actor policy parametrized by ϕ. To enhance stability, TD3 applies a delayed update to the actor network, updating it only after a set number of time steps (Fujimoto et al., 2018).

For the SAC critic, the loss function is given by:

$$L_Q = \frac{1}{N}\sum_i^N (Q_{\theta_1}(s_i, a_i) - y_i)^2 + (Q_{\theta_2}(s_i, a_i) - y_i)^2$$

where $y_i = r_i + \gamma \mathbb{E}_{a' \sim \pi}[\min(Q_{\theta'_1}(s', a'), Q_{\theta'_2}(s', a')) - \alpha \log \pi(a'|s')]$. The hyperparameter α represents the temperature, controlling the balance between reward

maximization and entropy maximization (Haarnoja et al., 2018).

The SAC actor's loss function is:

$$L_\pi = \frac{1}{N}\sum_{i}^{N} \alpha(\log \pi_\phi(a_i|s_i) - Q_{\theta_1}(s_i, a_i))$$

where π_ϕ is the stochastic policy parametrized by ϕ. This loss encourages actions that maximize the Q-value while also maximizing entropy (Haarnoja et al., 2018). SAC also includes a loss function for the temperature parameter α, represented as:

$$L_\alpha = -\frac{1}{N}\sum_{i}^{N} \alpha(\log \pi_\phi(a_i|s_i) + \mathcal{H}_{\text{target}})$$

Where $\mathcal{H}_{\text{target}}$ is the target entropy, a parameter that determines the extent to which exploration is encouraged (Haarnoja et al., 2018).

These detailed loss functions guide the learning processes of TD3 and SAC, enabling the training of actor and critic networks to achieve stable and optimal policies in complex environments.

D. Curriculum Learning

Curriculum Learning (CL) for robotics, as proposed by Sanger (1994), has been successfully applied in robotics to train networks and agents on increasingly complex tasks. The fundamental idea behind CL is to enable an agent to start learning a simple, easy task, progressing only after achieving a satisfactory level of performance. When the agent performs sufficiently well with the simpler task, it is moved to a more challenging one. This process continues, with the agent advancing to progressively harder tasks once it demonstrates adequate competence in each (Narvekar et al., 2020).

In the context of Markov Decision Processes (MDPs), CL can be conceptualized as a series of MDPs or tasks, with their own state space S_i, action space A_i, and reward function R_i but differ in their transition function P_i (Narvekar et al., 2020). We can then create a directed acyclic graph, where edges vertices are tasks to be learnt, and the directed edges tell us in what order these tasks must be learnt in (Narvekar et al., 2020).

Through this gradual increase in task difficulty, the agent benefits from a structured learning path, facilitating the acquisition of essential skills in earlier, simpler tasks that can then be leveraged when navigating more complex environments.

E. Related Work

Mannucci and van Kampen proposed a hierarchical approach to maze navigation that combines reinforcement learning and mapping to efficiently explore an unknown environment Mannucci and van Kampen (2016). The key aspects of their method are a hierarchical control policy for local navigation and an exploration strategy based on mapping and reinforcement learning. The authors first trained a "trainee" agent in a simplified, abstract environment to develop a low-level control policy, which was then applied iteratively at multiple levels of abstraction to enable local navigation within the maze. To guide global exploration, the agent built an online map of the environment using sensor readings and employed a frontier-based exploration strategy combined with reinforcement learning to select promising areas to explore, preventing repeated mistakes. The authors tested their approach in simulation using the maze environment from Parr and Russell Parr and Russell (1997) and found that the hierarchical control and exploration strategy allowed the agent to achieve a steady rate of exploration from the very first episode, outperforming a baseline policy that relied solely on a semi-empirical exploration strategy. The authors concluded that the combination of abstract, offline training with an adaptive, map-based exploration strategy resulted in an agent with satisfactory initial performance, without the typical "blind search" phase seen in many reinforcement learning approaches.

III. Methodology

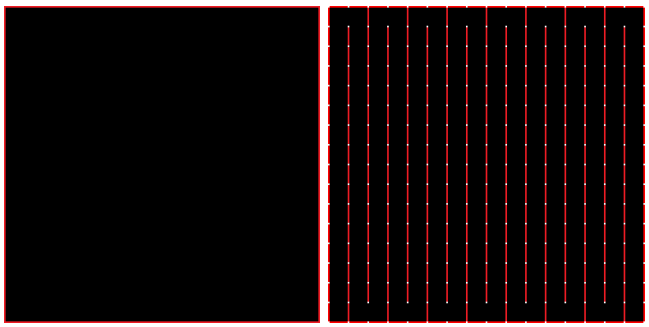
A. Markov Decision Processes

In our approach, we frame the problem as a Markov Decision Process (MDP) to enable reinforcement learning (RL). For our MDP, we define a custom state space, action space, and reward function tailored to our specific navigation task.

The **state space** consists of the following components: Δ (distance to the goal), ϕ (orientation towards the goal), v_x (robot's forward velocity), v_θ (robot's rotational velocity), and λ_{1-10} (ten laser sensor readings located at the front of the robot). Together, these elements form a 14-dimensional state vector representing the robot's status within its environment.

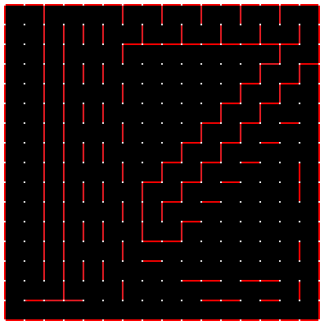
For the **action space**, we define two variables: u_1, representing the left motor voltage, and u_2, representing the right motor voltage. Thus, the action space is a two-dimensional vector that controls the robot's movement. Each element in both the state and action spaces is normalized to fall within the range [0,1].

The **reward function** is designed to encourage the robot to navigate toward the goal efficiently while avoiding collisions. It is defined as follows: Reward Function $= v_x - \phi - \min(\lambda_{1-10}) + 200 \times \mathbb{1}(\Delta < 0.3) - 300 \times \mathbb{1}(\min(\lambda_{1-10}) = 0)$
This function rewards the robot for moving quickly



(a) Easy Training Environment

(b) Medium Training Environment



(c) Hard Training Environment

Figure 1: These are the 3 environments which were used by the curricula to train the different models

toward the goal, penalizes misalignment with the goal, and encourages collision avoidance. Specifically, v_x, the forward velocity, is positively weighted to incentivize faster forward movement. The term ϕ, which represents the robot's orientation relative to the goal, is included as a negative reward—encouraging the robot to align itself with the goal. We also use $\min(\lambda_{1-10})$ to penalize the robot when it is near walls, as this value represents the shortest distance from the robot to a surrounding wall. By using the minimum laser reading, the robot learns to focus on the closest obstacle, which effectively helps it avoid collisions.

To further reinforce goal-oriented behaviour, the indicator function $\mathbb{1}(\Delta < 0.3)$ provides a substantial reward of 200 if the robot comes within a distance of 0.3 from the goal. Conversely, if the minimum laser reading $\min(\lambda_{1-10}) = 0$, indicating a collision, the robot receives a large penalty of -300.

B. Curriculum Learning and Training

Our CL approach involved training models across three progressively difficult environments: Easy, Medium, and Hard, to evaluate effectiveness across six curricula: Easy, Medium, Hard, Easy to Medium, Easy to Hard, and Easy to Medium to Hard. These curricula can be seen in the graph in Figure 2

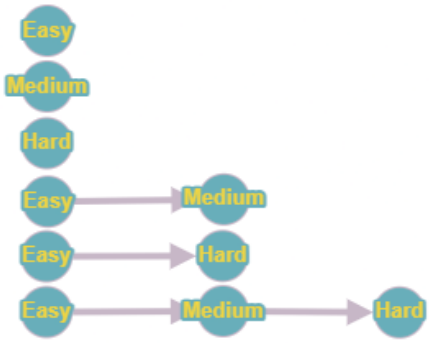


Figure 2: Graph showing the curricula that will be used to train our RL Agents

The **Easy Environment** (Figure 1a) is a basic open maze with perimeter walls, designed for initial navigation skills. The robot starts at a random position, facing a random direction, and aims to reach a random goal cell. Episodes end upon crashing or timeout, helping the agent develop core skills like moving straight, goal alignment, and boundary avoidance.

The **Medium Environment** (Figure 1b) is a zigzag path with narrow straight sections and turns, starting in either bottom corner with a random orientation. The robot's task is to reach the opposite corner, training it in handling narrow passages and precise turns.

The **Hard Environment** (Figure 1c) simulates complex maze-like conditions. Starting from the bottom left, the robot navigates a looping path with various wall orientations, reversing paths for variety. This environment aims to develop generalizable maze-solving skills.

Each environment functions as a task in the curricula, sharing state, action spaces, and reward function but differs in transition function P based on wall and open space layouts. Random starting elements prevent overfitting and enhance robustness. The Easy curriculum trains solely in the Easy Environment, while the Easy to Medium and Easy to Medium to Hard curricula transition through progressively difficult environments, following the principle of CL's gradual task progression (Sanger, 1994).

For each curriculum, we trained three TD3 models to compare across curricula and an additional three SAC models in Easy, Easy to Medium, and Easy to Medium to Hard curricula. This comparison examines whether performance improves through multi-environment exposure versus single-environment training.

The multi-environment CL process involved training agents sequentially in increasingly complex en-

vironments, starting in Easy and selecting the best-performing model for continued training in the next environment. Training timesteps increased with environment difficulty: 950,000 in Easy, 2,500,000 in Medium, and 4,000,000 in Hard, based on preliminary tests showing optimal learning within these limits.

C. Simulator and Environment Setup

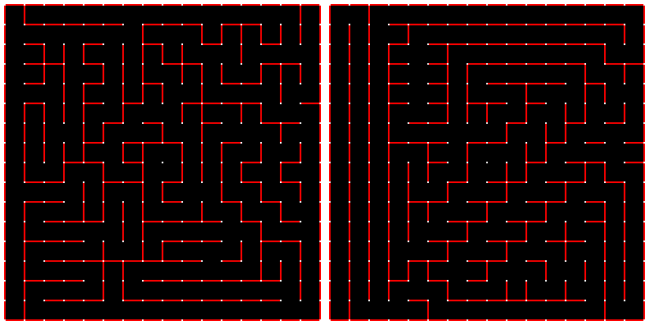
We utilized a modified version of an existing simulator[1] to control an iRobot Create™[2] robot for training purposes. This simulator, originally built using the PyBullet physics engine (Ellenberger, 2019), integrates with Gymnasium (Towers et al., 2024), providing a flexible framework for RL agent training. To facilitate the creation and training of our models, we used the Stable Baselines 3 library (Raffin et al., 2021), which offers reliable implementations of RL algorithms.

D. Performance Criteria

We used three primary criteria to evaluate the performance of our models: Speed, Completeness, and Robustness. Speed was assessed through two metrics: the distribution of the robot's forward velocity and the time taken to complete the environment. These metrics allowed us to gauge the efficiency of each agent's navigation. Robustness was evaluated by examining how the agent ended each episode, specifically comparing the proportions of episodes where the agent successfully completed the path, crashed, or timed out. By analysing these outcomes, we could infer the stability and safety of each model. Completeness was measured by the agent's progress along the designated path, with greater progress indicating closer proximity to the goal. These criteria provided a comprehensive assessment of each model's effectiveness in navigating the environments.

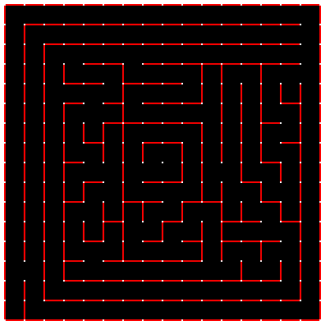
E. Model Testing

We used four distinct environments to evaluate our models. Three of these environments were sourced from previous Micromouse competitions, while the fourth environment, a zigzag maze, was designed to evaluate the models' turning capabilities. Specifically, Testing Environment 1 (Figure 3a) was derived from the 2010 Micromouse Robotic Competition, Testing Environment 2 (Figure 3b) originated from the 2014 Micromouse Japan Competition, and Testing Environment 3 (Figure 3c) was from the 2018 Micromouse APEC Competition. Testing Environment 4 (Figure 3d)

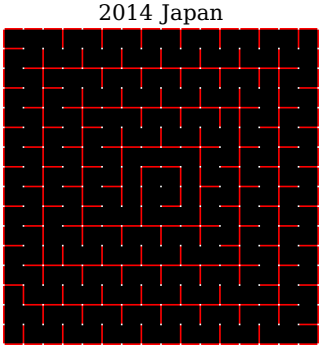


(a) Testing Environment 1: 2010 Robotic

(b) Testing Environment 2: 2014 Japan



(c) Testing Environment 3: 2018 APEC



(d) Testing Environment 4: Zigzag

Figure 3: These are the 4 environments that were used to evaluate the models trained on the different curricula

and the other environments were sourced from an online repository[3].

Each model was run ten times in each environment, and the results were averaged over these ten trials. Each environment featured a unique path for the agent to navigate, with each run terminating when the agent either completed the path, crashed, or timed out.

For the final evaluation, we averaged the results of models trained with different curricula to calculate a mean and variance for each of the performance criteria, allowing us to assess the effectiveness of each curriculum as a whole rather than focusing on individual agents. Results from the TD3 and SAC models were averaged separately to provide a comparative analysis across both model types.

IV. Results and Evaluation

A. Robustness

The results from Table I reveal several key insights regarding the performance of the models trained on different curricula, particularly in terms of the Success Rate, Crash Rate, and Timeout Rate.

[1] https://github.com/Yurof/WheeledRobotSimulations?tab=readme-ov-file

[2] Copyright and Trademark Notice: iRobot, Roomba and Create are registered trademarks of iRobot Corporation

[3] https://www.tcp4me.com/mmr/mazes/

Model Name	Success Rate	Crash Rate	Timeout Rate
TD3 Easy	0.158 ±0.159	0.3 ±0.053	0.542 ±0.21
TD3 Medium	0.167 ±0.12	0.483 ±0.076	0.35 ±0.138
TD3 Hard	0.042 ±0.055	0.183 ±0.279	**0.775** ±0.262
TD3 E->M	0.533 ±0.125	0.225 ±0.22	0.242 ±0.098
TD3 E->H	0.333 ±0.0	0.308 ±0.014	0.358 ±0.014
TD3 E->M->H	0.158 ±0.159	0.517 ±0.164	0.325 ±0.153
SAC Easy	0.025 ±0.043	**0.667** ±0.226	0.308 ±0.244
SAC E->M	**0.583** ±0.144	0.417 ±0.144	0.0 ±0.0
SAC E->M->H	0.083 ±0.144	0.592 ±0.192	0.325 ±0.06

Table I: Table showing the mean and standard deviation of the robustness of models trained on the different curricula using TD3 and SAC averaged over all models and all testing environments. E means Easy, M means Medium, and H means Hard

Model Name	Average Velocity	Average Time
TD3 Easy	0.203 ±0.201	12934.792 ±3453.339
TD3 Medium	-0.004 ±0.151	10328.142 ±4149.143
TD3 Hard	0.257 ±0.25	**19765.375** ±11634.553
TD3 E->M	0.43 ±0.176	12117.483 ±2415.741
TD3 E->H	0.3 ±0.237	14005.375 ±5989.614
TD3 E->M->H	0.204 ±0.201	10775.358 ±2229.447
SAC Easy	0.167 ±0.212	7202.575 ±3438.892
SAC E->M	**0.464** ±0.136	11060.017 ±4299.264
SAC E->M->H	0.289 ±0.246	8809.533 ±2954.3

Table II: Table showing the mean and standard deviation of the average velocity and average time spent in environments averaged over all models and all testing environments. E means Easy, M means Medium, and H means Hard

1) Success Rate: The success rate, defined as the rate at which agents complete their entire path without crashing or timing out, varies across curricula and algorithms. On average, models trained with multi-environment curricula achieve equal or higher success rates than those trained with uni-environment curricula. Notably, the Easy to Medium curriculum models trained with both TD3 and SAC perform exceptionally well, achieving the highest success rates among all curricula. Interestingly, models trained using the Easy to Medium to Hard curriculum underperform relative to the Easy to Medium models, suggesting a potential loss of learned knowledge rather than further improvement when advancing from the Medium to the Hard environment. Overall, TD3 models tend to exhibit higher success rates than SAC models; however, the SAC Easy to Medium model stands out, achieving the highest success rate of any curriculum configuration tested.

2) Crash Rate: The crash rate, defined as the frequency at which agents trained with a specific curriculum and algorithm collide with a wall in an unseen maze environment, varies significantly across different curricula and algorithms. The SAC Easy curriculum models exhibited the highest crash rate, with an average rate of 0.667 ± 0.226, nearly twice the crash rate of the TD3 Easy models, which recorded a rate of 0.3 ± 0.053. In contrast, the TD3 Hard model achieved the lowest crash rate, with a rate of 0.183 ± 0.279. Consistent with previous findings, the Easy to Medium to Hard curriculum models trained with both SAC and TD3 tended to perform worse in terms of crash rate compared to the Easy to Medium curriculum models. While multi-environment curricula generally have lower crash rates than their uni-environment counterparts, this trend does not hold universally. Overall, models trained with SAC tend to exhibit higher crash rates than those trained with TD3.

3) Timeout Rate: The timeout rate, which reflects the frequency at which agents trained with a specific curriculum and algorithm fail to complete an unseen maze environment within the allotted time, varies across different curricula and algorithms. The curriculum model with the highest average timeout rate is the TD3 Hard model, recording a rate of 0.775 ± 0.262. Generally, multi-environment curricula models show lower timeout rates compared to uni-environment models. Additionally, SAC curricula models tend to have lower timeout rates than their TD3 counterparts, with the SAC Easy to Medium model achieving the lowest average timeout rate of 0.0 ± 0.0. A persistent trend is observed where the Easy to Medium to Hard curriculum models perform worse than the Easy to Medium models.

B. Speed

The results from this section come from Table II. It shows how fast, on average, models trained with different curricula are in unseen maze environments. It does this by showing the average velocity and average time spent in environments by the different curricula models.

1) Average Velocity: The average forward velocity of agents, measured in meters per second, varies across curricula and algorithms. The models with the highest average velocities are the TD3 and SAC Easy to Medium curriculum models, with velocities of 0.43 $\frac{m}{s}$ ± 0.176 and 0.464 $\frac{m}{s}$ ± 0.136, respectively. In general, multi-environment curriculum models achieve higher average velocities than their uni-environment counterparts, except for the TD3 Easy to Medium to Hard model, which is approximately as fast as the TD3 Easy model. SAC models also tend to be faster than their TD3 counterparts, except for the SAC Easy model. Notably, both TD3 and SAC Easy to Medium to Hard models are slower on average than the Easy to Medium models.

Model Name	Completeness
TD3 Easy	0.221 ±0.184
TD3 Medium	0.225 ±0.076
TD3 Hard	0.378 ±0.287
TD3 E->M	**0.678** ±0.098
TD3 E->H	0.399 ±0.029
TD3 E->M->H	0.2 ±0.141
SAC Easy	0.103 ±0.046
SAC E->M	0.676 ±0.132
SAC E->M->H	0.204 ±0.153

Table III: Table showing the average completeness of the different curricula models averaged over all testing environments. E means Easy, M means Medium, and H means Hard

2) Average Time: The average time, measured in timesteps, that agents spend in an unseen maze environment varies with the curriculum and algorithm used. The TD3 Hard model exhibits the highest average time, spending 19765.375 ± 11634.553 timesteps in the environment. On average, multi-environment curricula models spend approximately the same amount of time in the environment as uni-environment models. SAC models generally have lower average times compared to their TD3 counterparts. Notably, both SAC and TD3 Easy to Medium to Hard curriculum models spend less time in the environment than the Easy to Medium models.

C. Completeness

Table III shows the completeness of models trained with a particular curriculum. Completeness is the metric that shows us how far along the path the agent got while traversing in an unseen maze environment.

The multi-environment curricula models generally outperformed the uni-environment models, except for the Easy to Medium to Hard curriculum models. The best-performing models were the TD3 and SAC Easy to Medium curriculum models, achieving completeness rates of 0.678 ± 0.098 and 0.676 ± 0.132, respectively. Overall, SAC models performed worse than TD3 models in terms of completeness. The poorest-performing model was the SAC Easy curriculum model, with a completeness rate of 0.103 ± 0.046. Additionally, the Easy to Medium to Hard curriculum models for both SAC and TD3 ranked among the worst-performing models overall.

D. Overall Evaluation

The uni-environment curricula, overall, produce models that perform poorly across most metrics, with high variance observed particularly among the Easy and Medium Curriculum models. This variance suggests some potential within these curricula, but ultimately, single-environment training does not equip agents with the robustness required for reliable zero-shot maze navigation.

Among multi-environment curricula, the two-environment approaches generally yield the best results across all metrics. Specifically, the Easy to Medium Curriculum consistently outperforms other curricula in nearly every metric across testing environments, making it the most effective curriculum for training robust navigation agents. In contrast, the Easy to Medium to Hard Curriculum, while offering some advantage over uni-environment curricula, often underperforms relative to other multi-environment curricula. Additionally, we observe a trend where models trained on curricula that incorporate the Hard Environment tend to perform better in the Zigzag Testing Environment, suggesting that exposure to harder environments can improve adaptability in complex scenarios.

Regarding algorithmic performance, there is no consistent pattern indicating that TD3 or SAC outperforms the other on the Easy or Easy to Medium to Hard Curricula, as both perform poorly overall. However, TD3 models trained on the Easy Curriculum perform slightly better than their SAC counterparts. Conversely, SAC models trained on the Easy to Medium Curriculum show superior performance compared to TD3 models and emerge as the best-performing models across all curricula and testing environments.

E. Limitations

One limitation of this study is the limited number of training environments used. Exploring alternative or additional training environments could lead to curricula that better adapt models to the demands of maze navigation. A broader variety of environments could improve the models' generalizability, potentially enhancing their ability to navigate new maze scenarios. Additionally, the models were evaluated and averaged over only 10 testing iterations, which may limit the statistical confidence in the reported results. Furthermore, only three models were trained per curriculum, which constrains the understanding of curriculum effects on model behaviour. Training additional models would yield a more comprehensive perspective on the impact of each curriculum.

V. Conclusions and Future Work

The research presented here confirms that progressive training through curriculum learning, particularly with easy-to-medium curricula, leads to superior performance in zero-shot maze navigation. Our experiments identified that multi-environment training, especially with TD3 and SAC algorithms, leads to

greater robustness, speed and completion than single-environment training. This study's primary contribution is in establishing an effective curriculum-based approach to prepare RL agents for navigation in previously unseen mazes. Future work could explore additional curriculum structures such as adding curricula where models get sent back to easier environments if they fail in harder ones or by adding additional training environments which can be added to curricula. Adding more diverse testing environments to further enhance agent adaptability in complex, real-world maze navigation tasks could also be advantageous. Finally, future work could also examine using hierarchical approaches for maze navigation.

Acknowledgement

Simulator:
https://github.com/dean-south/HonoursResearchSim

References

Adamkiewicz, M., Chen, T., Caccavale, A., Gardner, R., Culbertson, P., Bohg, J., and Schwager, M. (2022). Vision-only robot navigation in a neural radiance world. *IEEE Robotics and Automation Letters*, 7(2):4606–4613.

Aradi, S. (2022). Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):740–759.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

Betz, J., Zheng, H., Liniger, A., Rosolia, U., Karle, P., Behl, M., Krovi, V., and Mangharam, R. (2022). Autonomous vehicles on the edge: A survey on autonomous vehicle racing. *IEEE Open Journal of Intelligent Transportation Systems*, 3:458–488.

Ellenberger, B. (2018–2019). Pybullet gymperium. https://github.com/benelot/pybullet-gym.

Feng, S., Sun, H., Yan, X., Zhu, H., Zou, Z., Shen, S., and Liu, H. X. (2023). Dense reinforcement learning for safety validation of autonomous vehicles. *Nature*, 615(7953):620–627.

Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR.

Grass, D., Caulkins, J. P., Feichtinger, G., Tragler, G., Behrens, D. A., et al. (2008). Optimal control of nonlinear processes. *Berlino: Springer*.

Grondman, I., Busoniu, L., Lopes, G. A. D., and Babuska, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR.

Kirk, D. E. (2004). *Optimal control theory: an introduction*. Courier Corporation.

Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

Lillicrap, T. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Mannucci, T. and van Kampen, E.-J. (2016). A hierarchical maze navigation algorithm with reinforcement learning and mapping. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE.

Mavrogiannis, C., Baldini, F., Wang, A., Zhao, D., Trautman, P., Steinfeld, A., and Oh, J. (2023). Core challenges of social robot navigation: A survey. *J. Hum.-Robot Interact.*, 12(3).

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

Parekh, D., Poddar, N., Rajpurkar, A., Chahal, M., Kumar, N., Joshi, G. P., and Cho, W. (2022). A review on autonomous vehicles: Progress, methods and challenges. *Electronics*, 11(14):2162.

Parr, R. and Russell, S. (1997). Reinforcement learning with hierarchies of machines. *Advances in neural information processing systems*, 10.

Peng, X. B., Coumans, E., Zhang, T., Lee, T.-W., Tan, J., and Levine, S. (2020). Learning agile robotic locomotion skills by imitating animals.

Puterman, M. L. (1990). Markov decision processes. *Handbooks in operations research and management science*, 2:331–434.

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

Rao, A. V. (2009). A survey of numerical methods for optimal control. *Advances in the astronautical*

Sciences, 135(1):497–528.

Remonda, A., Krebs, S., Veas, E., Luzhnica, G., and Kern, R. (2022). Formula rl: Deep reinforcement learning for autonomous racing using telemetry data.

Sanger, T. (1994). Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE Transactions on Robotics and Automation*, 10(3):323–333.

Shaikhet, L. (2022). Some unsolved problems in stability and optimal control theory of stochastic systems. *Mathematics*, 10(3):474.

Shamshad, F., Khan, S., Zamir, S. W., Khan, M. H., Hayat, M., Khan, F. S., and Fu, H. (2023). Transformers in medical imaging: A survey. *Medical Image Analysis*, 88:102802.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 387–395, Bejing, China. PMLR.

Song, Y., Romero, A., Müller, M., Koltun, V., and Scaramuzza, D. (2023). Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. *Science Robotics*, 8(82):eadg1462.

Sutton, R. S. (2018). Reinforcement learning: An introduction. *A Bradford Book*.

Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., et al. (2024). Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.(nips), 2017. *arXiv preprint arXiv:1706.03762*, 10:S0140525X16001837.

Zhu, K. and Zhang, T. (2021). Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Science and Technology*, 26(5):674–691.