Dynamic Path Planning for Autonomous Vehicles Using Adaptive Reinforcement Learning

Karim Wahdan[1], Nourhan Ehab[1], Yasmin Mansy[1] and Amr El Mougy[2]
[1]*German University in Cairo, Cairo, Egypt*
[2]*American University in Cairo, Cairo, Egypt*

Keywords: Dynamic Path Planning Autonomous Vehicles, Adaptive Reinforcement Learning.

Abstract: This paper focuses on local dynamic path planning for autonomous vehicles, using an Adaptive Reinforcement Learning Twin Delayed Deep Deterministic Policy Gradient (ARL TD3) model. This model effectively navigates complex and unpredictable scenarios by adapting to changing environments. Testing, using simulations, shows improved path planning over static models, enhancing decision-making, trajectory optimization, and control. Challenges such as vehicle configuration, environmental factors, and top speed require further refinement. The model's adaptability could be enhanced by integrating more data and exploring a fusion between supervised reinforcement learning and adaptive reinforcement learning techniques. This work advances autonomous vehicle path planning by introducing an ARL TD3 model for real-time decision-making in complex environments.

1 INTRODUCTION

Modern transportation systems are an essential cornerstone of contemporary society, revolutionizing urban mobility and the efficient movement of goods. Yet, the increased utilization of these systems has been accompanied by a surge in accident rates, a significant proportion of which can be attributed to human errors (Singh, 2017). In response to this challenge, the pursuit of fully autonomous transportation has emerged as a promising solution. By reducing accidents and mitigating traffic congestion, autonomous vehicles hold the potential to transform transportation systems fundamentally.

However, the realization of autonomous transportation systems is not without its critical challenges. One of the most threatening obstacles faced by current autonomous systems is their ability to execute real-time dynamic path planning in the face of unpredictable environments. In these dynamic and ever-changing scenarios, autonomous systems often struggle to determine the optimal route and trajectory for vehicles to navigate complex urban settings, avoiding potential collisions in real-time. Traditional rule-based systems and even earlier reinforcement learning methods have shown limitations in adapting to the rapidly shifting conditions of the road (Sharma et al., 2021).

This is precisely where Adaptive Reinforcement Learning (ARL) takes center stage. ARL is a specialized subfield of machine learning and reinforcement learning, uniquely designed to create autonomous systems that possess the capacity to adapt their behavior and decision-making processes in response to real-time changes in their environment (Li et al., 2021). In the context of autonomous vehicles, ARL becomes a critical enabler, as it allows these vehicles to continually refine their policies, optimizing their decision-making in complex and uncertain traffic situations. The need for ARL in the development of autonomous vehicles stems from the dynamic and unpredictable nature of real-world traffic environments. Unlike traditional rule-based systems or static reinforcement learning, ARL algorithms can adapt to unexpected occurrences, rapidly shifting traffic patterns, and diverse driving conditions. These models are robust to uncertainties, less likely to fall victim to biases, and can learn efficiently even with limited data, making them well-suited for real-world applications.

The objective of this research is to harness the potential of ARL models for dynamic path planning in autonomous systems, specifically autonomous vehicles. We aim to create a flexible model that can adapt seamlessly to evolving environmental conditions. This adaptability not only enhances the navigational capabilities of autonomous vehicles but also

Wahdan, K., Ehab, N., Mansy, Y. and El Mougy, A.
Dynamic Path Planning for Autonomous Vehicles Using Adaptive Reinforcement Learning.
DOI: 10.5220/0012363300003636

In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART 2024) - Volume 1*, pages 272-279
ISBN: 978-989-758-680-4; ISSN: 2184-433X

minimizes data requirements and reduces the impact of biases. In our evaluation, we place significant emphasis on metrics such as control smoothness and success rate to provide a comprehensive assessment of real-world performance. Our ultimate goal is to contribute to the ongoing evolution of autonomous transportation systems, thereby advancing their capacity for real-time decision-making in ever-changing environments and enhancing the safety and efficiency of our transportation networks, all while highlighting the pivotal role of ARL in this endeavor.

The rest of this paper is structured as follows. Section 2 presents the design and implementation of our ARL model for dynamic path planning. Section 3 unveils the specific evaluation metrics and testing scenarios employed, presenting their corresponding results. Section 4 presents the analysis of these results, discusses their implications, and draws comparisons with other related work in the field. Finally, Section 5 presents some concluding remarks.

2 PROPOSED FRAMEWORK DESIGN

In the pursuit of developing advanced dynamic path planning for autonomous vehicles, our paper introduces a comprehensive framework that leverages the power of Adaptive Reinforcement Learning (ARL). This section provides a detailed overview of this framework, which is divided into three main components: the Meta-Drive simulation environment, the TD3 model and its policy, and an adaptive module.

2.1 Framework Overview

The core of our approach lies in the interaction between three central components: the Meta-Drive simulation environment, the TD3 model and its policy, and the adaptive module. Figure 1 is a visualization of those interactions.

The Meta-Drive environment (Li et al., 2022) stands as a robust simulation platform, purpose-built for the training and evaluation of Reinforcement Learning (RL) autonomous vehicles within intricate urban traffic environments. With its high fidelity and realistic features, including an accurate physics engine, diverse vehicle dynamics, and realistic traffic patterns, Meta-Drive provides an exceptional setting for training autonomous vehicles.

At the heart of our framework, the Twin Delayed Deep Deterministic Policy Gradients (TD3) model plays a pivotal role in policy development and action

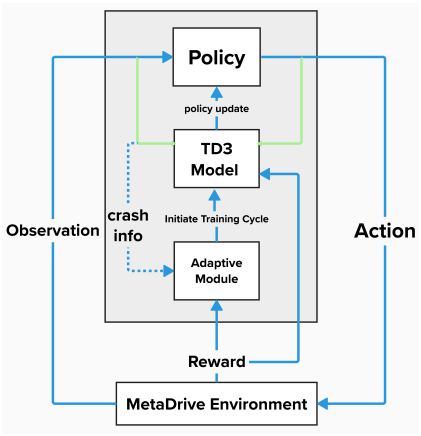

Figure 1: Framework Overview.

generation. Informed by observations from the Meta-Drive environment, the TD3 model formulates actions to be executed by autonomous vehicles. The policy embedded within the TD3 model guides the process of action selection.

Finally, the MetaDrive environment not only supplies observation states and receives actions from the TD3 model but also provides rewards for the actions taken, serving as a feedback signal for learning. Additionally, it shares crash information for specific episodes with the adaptive module, allowing it to initiate new training cycles as necessary. In what follows, we delve into the details of each module.

2.2 Meta-Drive Environment

The Meta-Drive (Li et al., 2022) is a driving simulation platform that provides observations with a wide range of sensory input including low-level sensors LIDAR and depth camera, and high-level scene information such as the road environment, nearby vehicles, and vehicle status. Our observation space can be divided into three main sections: the vehicle state, navigation information, and surrounding data. The vehicle state variables are 19 values including values like steering, heading, and velocity. The navigation information variables are 10 values that represent the relationship between the vehicle and its final destination and following checkpoints The surrounding information is a vector that captures various aspects of the environment, including data from 2 side detectors, and 40 LIDAR points input.

In our context, the action space comprises two essential components: steering and throttle. These components are assigned numerical values that span from -1 to 1. This numerical range facilitates precise and adaptable maneuvering in various driving scenarios.

The reward function in Meta-Drive is designed to

incentivize desirable behaviors and discourage undesirable ones. It is be defined as:

$$R(s,a,s') = c_1 \cdot Driving\_reward + c_2 \cdot Speed\_reward + Termination\_reward$$

The driving reward is calculated as $d(t+1) - d(t)$, where $d(t+1)$ and $d(t)$ denote the longitudinal coordinates of the target vehicle. with $c_1 = 1$ The speed reward is calculated asv_t/v_{max}, which encourages the agent to maintain a high speed while minimizing unnecessary accelerations and decelerations. with $c_2 = 0.1$

The termination reward is given to the autonomous vehicle at the end of the episode, based on the termination state. That can be a success reward (a positive reward), an out-of-road penalty, a crashed-vehicle penalty, or a crashed-object penalty — all of which are negative.

There is one additional optional parameter that we introduced to the reward function called the "lateral reward". This is a multiplier that is added to the driving reward to incentivize the agent to stay in lane. The lateral reward is used on some models in testing

2.3 TD3 Model Development

The TD3 model serves as the fundamental component of our adaptive reinforcement learning (ARL) solution. It is a significant advancement over the traditional policy of the Gradient actor-critic and DDPG model by enhancing the error approximation associated with the Actor-Critic models through the incorporation of twin critics. This effectively mitigates the issue of the overestimation bias commonly observed and a delayed policy update mechanism enabling a more precise estimation of values and target policy smoothing. Consequently, this encourages exploration and prevents the policy from becoming too deterministic by incorporating noise during the training phase.

2.3.1 Replay Buffer

The first component of our model is the replay buffer, which serves as the main memory of the reinforcement learning model and is utilized to store and sample past experiences, allowing the model to draw from its accumulated experiences to generate training data for efficient learning. Our replay buffer is implemented with a fixed capacity, to prevent the model from being influenced by ancient data in new training cycles, allowing it to be more dynamic in new environments. Each transition represents the change of state based on the actions of the autonomous vehicle and is stored in the replay buffer as five components: the current state, the resulting next state, the action taken, the associated reward, and an episode completion indicator.

The initialized replay buffer comprises a data structure of a predetermined size, featuring an indicator pointing to either the subsequent vacant position or the position earmarked for replacement.

Transitions can be added to the replay buffer and they are added to the next vacant position or the oldest transition is replaced with the newest.

Any sample subset of the replay buffer can be generated for training the model on a subset of previous experiences.

2.3.2 Neural Networks Infrastructure

The model consists of six dense deep neural networks each with their functionality, the actor-network serves as the policy network, responsible for generating actions based on the current state of the environment.

The critic-networks play a crucial role in estimating the action-value function, which evaluates the quality of the actions taken by the actor-network, the critic-networks output a scalar value that represents the estimated action-value or Q-value for the given state-action pair.

To stabilize the training process and enhance the overall performance of the TD3 model, the introduction of two sets of target networks is crucial. These networks include the target actor-network and the target-critic networks, which play integral roles in estimating action values and reducing overestimation bias.

The target actor-network and target critic-networks are replicas of their respective actor and critic networks. They do not get updated during training; instead, their parameters change gradually through soft target network updates. This process involves smoothly blending the target network's parameters with those of the actual network, using a small factor to achieve a gradual update. Soft target network updates are crucial for stabilizing the learning process.

2.3.3 TD3 Mechanisms

The TD3 model follows a comprehensive training approach, involving actions like sampling transitions, computing target Q-values, minimizing overestimation bias, and employing soft target network updates. These mechanisms collectively enable precise value estimation, enhanced exploration, and stabilization of the learning process.

The intricacies of the TD3 model and its training process are encapsulated in a single entity, as demonstrated in Algorithm 2. This entity efficiently orchestrates the model's interactions with its environment, facilitating learning, and adaptation.

Data: state_dim, action_dim, max_action
Result: TD3 Agent
Function ***TD3***`(state_dim, action_dim, max_action)`
  Initialize actor, actor_target, critic, and critic_target networks;
  Load actor, critic weights into actor_target, critic_target;
  Initialize actor and critic optimizer;
  Set maximum action value;
Function ***select_action***`(state)`
  return actor(`state`);
Function ***train***`(replay_buffer, iterations, batch_size, discount, tau, policy_noise, noise_clip, policy_freq)`
  for *it* ← 1 **to** `iterations` **do**
    Sample transitions (s, s', a, r, d);
    From the next state s', compute the next action a' using actor target;
    Add Gaussian noise to the next action and clamp it;
    Compute target Q-values using the two critic_targets;
    Keep the minimum of these two Q-values: min(Qt1, Qt2);
    Compute target Q-values with discount factor;
    Compute current Q-values using two critic networks;
    Compute critic loss;
    Backpropagate the Critic loss and update the parameters of the two Critic models using SGD optimizer;
    if *it* % `policy_freq` == *0* **then**
      Compute actor loss;
      Update actor parameters using gradient ascent optimizer;
      Update actor_target and critic_target weights using Polyak averaging every two iterations;
    end
  end
Function ***save***`(filename, directory)`
  Save actor, critic weights to a file;
Function ***load***`(filename, directory)`
  Load actor, critic weights from a file;

Algorithm 1: TD3 Class.

2.3.4 Model Training

The training process unfolds over millions of timesteps, driven by a robust training loop. The key steps in the training loop encompass episode monitoring, policy evaluation, reward calculation, and storage of evaluation results. These iterative steps ensure that the agent continuously refines its policy through interactions, training, and evaluations within the environment.

At each timestep, the procedure checks if the episode has ended or if the maximum steps per episode have been reached. If so, training begins. Using experiences from the replay buffer, the policy is learned with the `train()` function, but only if there are enough experiences and it's not the first timestep. After training, the current policy is evaluated with the `evaluate_policy()` function, based on the `eval_freq` argument. The evaluation results are stored, and the policy is saved.

The `evaluate_policy()` function takes a given model and "n" episodes to test it on each episode consisting of "m" steps and then returns the average reward over testing and the average number of steps per episode before termination. In our train the models were tested on 10 episodes of $10e4$ steps each. again these values were obtained through multiple iterations of tuning of trying to ensure a steady reward output of the same model.

Episode-related variables are updated, timesteps increase, and the environment state resets. The agent decides whether to explore (before start timesteps) or choose an action based on the learned policy. If there's exploration noise, it adds noise to the action within action space boundaries. The chosen action is executed in the environment, and the code retrieves the next observation, reward, and episode status. Episode and overall rewards are updated, and the transition is stored in the replay buffer.

Once the main loop ends, the final policy evaluation is added to the evaluations list, and if the save models flag is set, the policy is stored. The final evaluation result is saved as well. The environment is closed and restarted. The average reward across episodes is computed by dividing the total reward by the number of episodes (episode num). This loop iteratively improves the agent's policy through interactions, training, and periodic evaluations with the environment.

2.4 Adaptive Module

The final piece of our framework is the adaptive module, a hybrid approach that combines batch learn-

ing and online learning to maximize adaptability. It begins with an offline training phase in which the model calculates an average reward across all training episodes. This average reward serves as a threshold for subsequent episodes.

During new episodes, if the model's performance falls below 20% of the average reward, it triggers an online learning session, indicating a significant environmental change. If, during three consecutive episodes, the model scores below 70% of the average reward, it initiates another online training session, addressing more subtle changes in the environment.

To ensure continual adaptability, the model engages in batch learning cycles every 10^4 timesteps, enabling it to remain up-to-date with evolving trends and environmental shifts. This hybrid incremental/batch adaptive learning approach enhances the model's capacity to adapt to both significant and subtle changes in the environment, making it well-equipped to handle dynamic, complex traffic scenarios.

The parameters of 20%, 70%, and 10^4 steps were derived through multiple sessions of hyperparameter tuning by trying multiple permutations of values for each parameter.

3 RESULTS

To test the effectiveness of our proposed model, rigorous tests on 5 different challenging real-world urban scenarios were performed. Each is designed to test an aspect of the model's performance. Four distinct TD3 models, each with variations in training type, training duration, and the presence of lateral reward were tested.

3.1 Scenarios and Model Variations

Figure 2 Represents a top overview of each of the 5 distinct maps that models were tested on, tests include high traffic density, complex road layouts, and diverse lane configurations. This diversity ensures a comprehensive assessment of the models' adaptability to various urban settings. a detailed description of each scenario will be provided in the corresponding scenario result's sub-section

Figure 2: Scenarios Maps Overview.

Table 1 shows a comparison between the attributes and distinguishing features of the four models subjected to testing which can be elaborated upon as follows:

Table 1: Model Performance Summary.

Model	Training Type	Duration	Lateral Reward
Model 1	Static	10^6	No
Model 2	Static	$2 * 10^6$	No
Model 3	Adaptive	10^6	Yes
Model 4	Adaptive	$2 * 10^6$	Yes

3.2 Scenario 1

In the first scenario, the models face a map with six blocks. The environment is configured with a traffic density of 0.1, representing a moderate level of traffic. The lanes on the roads are randomly assigned, and the vehicle spawn lane is randomized.

As Illustrated by figure 3 the performance of the four models on this map. Model 1 achieved a 60% success rate, indicating a moderate level of competence in completing driving tasks. However, it had a higher collision rate of 25% and a 15% out-of-road rate. Model 2 showed improvement with a 70% success rate, a lower collision rate of 20%, and a reduced out-of-road rate of 10%. Model 3 displayed further improvement, achieving a 75% success rate, matching Model 2's collision rate while significantly reducing out-of-road incidents to 5%. The top-performing model in this scenario was Model 4, with an impressive 90% success rate, excellent collision avoidance (10% collision rate), and perfect adherence to the designated road path (0% out-of-road rat

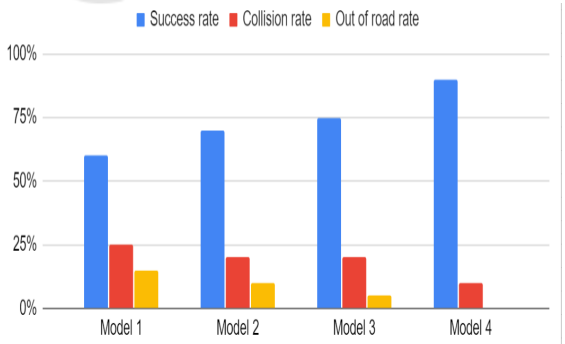

Figure 3: Scenario 1 Results.

3.3 Scenario 2

In the second scenario, the models are tested on a map with six blocks, featuring a higher traffic density of 0.4, representing heavy traffic. The lanes on the roads are randomly assigned, and the vehicle spawn lane is randomized.

As Illustrated by figure 4 All Models had slightly worse performance than scenario 1 due to it being a challenging scenario with a more complex configuration of blocks and higher traffic density Model 4 had the best performance by quite a large margin while model 1 had the worst performance. while model 3 slightly outperformed model 2.

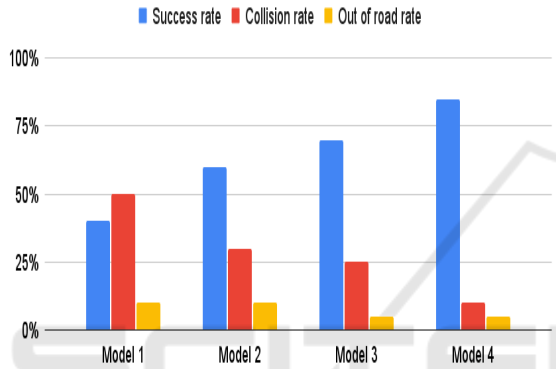

Figure 4: Scenario 2 Results.

3.4 Scenario 3

In the third scenario, the models face a map with six blocks and a traffic density of 0.1, similar to Scenario 1. However, the number of lanes on the roads is assigned to three lanes, and the vehicle spawn lane is also randomized.

As Illustrated by figure 5 All Models had significantly better performance than scenario 1,2 due to it being a simple scenario with a less complex configuration of blocks, lower traffic density, and a known number of lanes The highest-performing model remained Model 4 with its success rate at 90% while model 1 had the worst performance. while model 3 slightly outperformed model 2.

3.5 Scenario 4

Scenario 4 presents a map with six blocks and a traffic density of 0.1, mirroring Scenario 3. However, in this scenario, the lanes on the roads are assigned to three lanes, and the vehicle spawn lane is placed in the middle lane.

As Illustrated by figure 6 All Models had the best

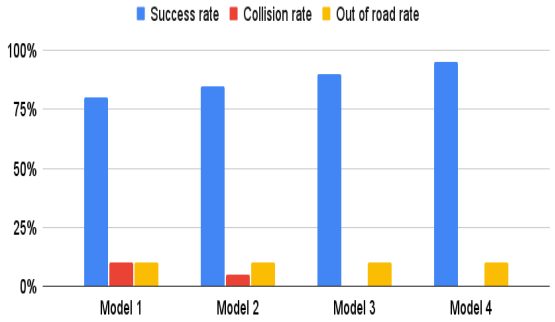

Figure 5: Scenario 3 Results.

performance due to it being a simple scenario with a less complex configuration of blocks lower traffic density and a known number of lanes and spawning point Model 4 had the best performance by quite a large margin with a success rate of 95% while model 1 had the worst performance with a success rate of about 80%. while model 3 slightly outperformed model 2.

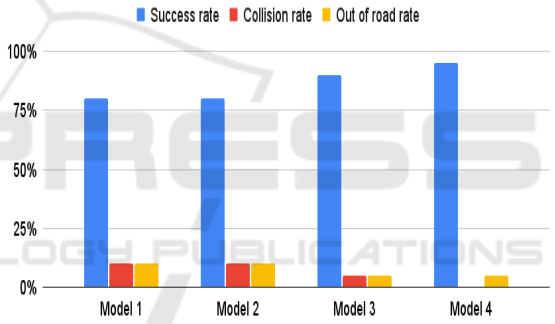

Figure 6: Scenario 4 Results.

3.6 Scenario 5

In this scenario, the four models are tested on the fifth map, which consists of 7 blocks. The environment is configured with a traffic density of 0.7, meaning that there is a moderate amount of traffic on the roads. The lanes in the environment roads are randomly assigned and the vehicle spawn lane is also randomized.

As Illustrated by figure 7 All Models had the worst performance due to it being a simple scenario with a more complex configuration of ”7” blocks, and higher traffic density. Model 4 had the best performance by quite a large margin with a success rate of 85% while model 1 had the worst performance with a success rate of about 50%. while model 3 slightly outperformed model 2.

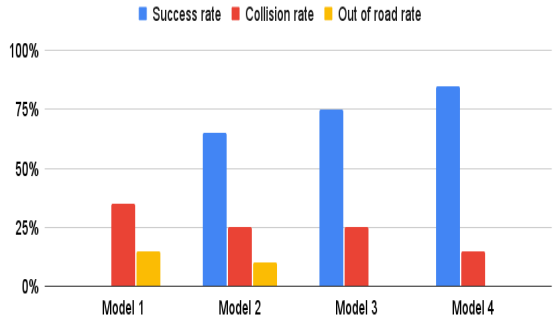

Figure 7: Scenario 5 Results.

4 DISCUSSION

The evaluation of the four models in realistic urban environments provided valuable insights into their performance, accuracy, and adaptability. The comparison of metrics across different scenarios sheds light on the strengths and weaknesses of each model configuration.

4.1 Model Performance and Accuracy

The adaptive TD3 models (Models 3 and 4) consistently outperformed the static TD3 models (Models 1 and 2) in all scenarios. They showed higher success rates, lower collision rates, and better adherence to traffic rules, indicating their effectiveness in handling dynamic urban scenarios.

Model 4 outperformed all other models across various scenarios, achieving the highest success rate in urban driving. Its success can be attributed to adaptive training, longer training duration, and the inclusion of a lateral reward component. This model excelled in collision avoidance and adherence to traffic rules, making it the preferred choice for real-world urban driving.

Model 3 also performed well, with success rates comparable to Model 4. This highlights the significance of adaptive elements in reinforcement learning models for urban driving.

In contrast, Models 1 and 2, which employed static training, exhibited lower success rates and higher collision rates due to their inability to adapt to changing conditions. While Model 2 performed better than Model 1, both lagged behind adaptive models, underscoring the limitations of static training for handling complex urban driving scenarios.

Furthermore, Models 3 and 4 demonstrated improved steering control smoothness, thanks to their adaptability and continuous learning during testing. This improvement in steering control contributed to their higher success rates and better adherence to traffic rules.

Table 2: Model Performance Summary.

Model	Success Rate	Collision Rate	Steering Control
Model 1	Low	High	Not Smooth
Model 2	Moderate	Moderate	Not Smooth
Model 3	Above Model 2	Moderate	Improved
Model 4	Highest	Low	Improved

4.2 Training Time and Average Reward

The training time and average reward achieved during the training phase were analyzed, revealing some noteworthy findings. Models 2 and 4, benefiting from longer training durations of $2 * 10^6$ steps, outperformed Models 1 and 3 in terms of average reward. This highlights the positive correlation between extended training time and improved training performance, resulting in higher rewards.

However, it's crucial to consider that longer training durations come with increased computational resource requirements and time investments. Therefore, there exists a trade-off between training time and performance, as shown in Model 3 performance, despite having a shorter training duration of 10^6 steps, still delivered impressive performance and accuracy. This indicates that adaptive training techniques can effectively optimize training efficiency, allowing competitive results to be achieved within a reasonable time frame.

4.3 Related Work

In the realm of local path planning for autonomous vehicles in dynamic environments, research has identified limitations in classical algorithms for real-time adaptability, as discussed by (Gonzalez Bautista et al., 2015). In contrast, our Adaptive Reinforcement Learning (ARL) model excels in swift decision-making in rapidly changing situations.

Another study by (Panda et al., 2020) found existing algorithms lacking in reliability for Automated Underwater Vehicles (AUVs) in unmapped environments. Our model addresses this challenge with an adaptive module for effective adaptation in unmapped territories over time.

In a different study, (Hebaish et al., 2022) introduced a fusion method combining Twin Delayed

Deep Deterministic (TD3) and Supervised Learning (SL) to reduce training time and improve success rates in specific scenarios.

Our model surpasses this approach and results in even more complex scenarios with multiple lanes and variable spawning positions, thanks to fine-tuning and its adaptive module.

Additionally, (Zhou et al., 2022) extended the work of (Gonzalez Bautista et al., 2015) by exploring various machine learning techniques and highlighting the time-consuming nature of training RL policies. Our model builds upon the TD3 model by continuously updating its policy and adapting to new data without extensive training or concept drift concerns.

5 CONCLUSION

The Adaptive TD3 model introduced shows a promising solution to the complex problem of local dynamic path planning in autonomous driving. It builds upon the foundation of the classical TD3 model and aims to address the real-world challenges faced by autonomous vehicles. The results of our evaluation demonstrate the model's superior performance in various scenarios, emphasizing its adaptability and effectiveness.

However, it is important to acknowledge several key limitations. The model's maximum velocity constraint of 40 km/h, while necessary for stability within the limited training time, is a significant drawback. Future research should focus on developing methods to raise the speed limit without compromising performance, allowing the model to operate effectively at higher speeds.

Another limitation is the lengthy training period, especially at low speeds. This highlights the need for more efficient training techniques that can expedite the learning process. One potential solution is the integration of Adaptive Reinforcement Learning (ARL) with Supervised Reinforcement Learning (SRL) to create an ASRL TD3 model, offering faster and more effective training.

Environmental factors, such as weather conditions, lighting variations, and interactions with pedestrians, were not considered in the evaluation. These factors play a crucial role in real-world driving scenarios, and future research should focus on incorporating them into the model's input and training process. Addressing these aspects is vital for enhancing the real-world applicability of the Adaptive TD3 model.

REFERENCES

Gonzalez Bautista, D., Pérez, J., Milanes, V., and Nashashibi, F. (2015). A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17:1–11.

Hebaish, M. A., Hussein, A., and El-Mougy, A. (2022). Supervised-reinforcement learning (srl) approach for efficient modular path planning. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 3537–3542.

Li, Q., Peng, Z., Feng, L., Zhang, Q., Xue, Z., and Zhou, B. (2022). Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Li, X., Xu, H., Zhang, J., and Chang, H.-h. (2021). Optimal hierarchical learning path design with reinforcement learning. *Applied psychological measurement*, 45(1):54–70.

Panda, M., Das, B., Subudhi, B., and Pati, B. B. (2020). A comprehensive review of path planning algorithms for autonomous underwater vehicles. *International Journal of Automation and Computing*.

Sharma, O., Sahoo, N. C., and Puhan, N. B. (2021). Recent advances in motion and behavior planning techniques for software architecture of autonomous vehicles: A state-of-the-art survey. *Engineering applications of artificial intelligence*, 101:104211.

Singh, S. K. (2017). Road traffic accidents in india: issues and challenges. *Transportation research procedia*, 25:4708–4719.

Zhou, C., Huang, B., and Fränti, P. (2022). A review of motion planning algorithms for intelligent robots. *Journal of Intelligent Manufacturing*, 33(2):387–424.