A Hybrid Reinforcement Learning and Model Predictive Control Framework for Autonomous Vehicle Lane-Changing

Mayank Deshpande
120387333
msdeshp4@umd.edu

Suraj Kalwaghe
120417634
suraj108@umd.edu

Tanmay Pancholi
120116711
tamy2909@umd.edu

Abstract

Autonomous vehicles (AVs) require precise navigation in complex traffic, especially during lane changes to ensure safety, efficiency, and comfort. Traditional Model Predictive Control (MPC) systems use fixed weighting schemes that may perform suboptimally under varying traffic conditions. This project presents a hybrid framework integrating Reinforcement Learning (RL) with MPC, enabling dynamic, lane-specific risk-aware weight adjustments. Utilizing the Soft Actor-Critic (SAC) algorithm, the RL agent adaptively modifies MPC weights based on factors such as traffic density, speed limits, and collision risks. Implemented within the Simulation of Urban Mobility (SUMO) environment via the TraCI interface, the framework includes a lane-specific risk assessment module categorizing lanes into low, medium, or high-risk. Experimental results demonstrate that the RL-MPC hybrid outperforms traditional MPC in lane-change success rate, collision avoidance, comfort metrics (average jerk and lateral acceleration), and time efficiency. Additionally, the modular design supports future integration of other RL algorithms like Proximal Policy Optimization (PPO) and Twin Delayed Deep Deterministic PolicRL+MPC Hybridy Gradient (TD3). This research advances adaptive control strategies in autonomous driving, enhancing safety and efficiency through intelligent, context-aware decision-making. Code: https://github.com/MayankD409/RL_MPC.git

Keywords: *Autonomous Vehicles, Reinforcement Learning, Model Predictive Control, Lane Changing, Risk Assessment, SUMO, TraCI*

1. Introduction

Autonomous vehicles (AVs) are poised to transform transportation by enhancing safety, reducing traffic congestion, and increasing mobility. A critical capability of AVs is executing lane-changing maneuvers with precision and reliability. These maneuvers are inherently challenging as they require real-time decision-making that accounts for dynamic traffic conditions, ensures passenger comfort, and maintains overall system safety.

Traditional Model Predictive Control (MPC) systems have been widely adopted for controlling AVs due to their ability to handle multi-objective optimization problems. However, MPC systems typically rely on fixed weighting schemes to balance objectives such as safety, efficiency, and comfort. This rigidity often leads to suboptimal performance in varying traffic environments, where the relative importance of these objectives may fluctuate. For instance, overly conservative MPC weights in low-risk scenarios can hinder efficiency, while insufficient safety emphasis in high-risk conditions can compromise safety.

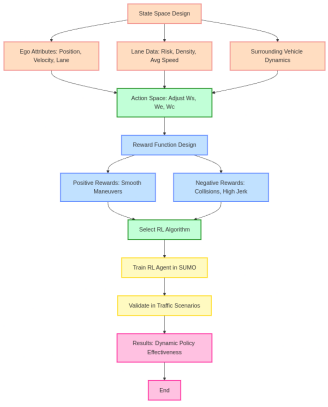

Figure 1. System Architecture of the RL+MPC Hybrid Framework

To address these limitations, this project introduces a hybrid framework that integrates Reinforcement Learning (RL) with MPC, enabling dynamic, lane-specific risk-aware weight adaptation. By leveraging the Soft Actor-Critic (SAC) algorithm, the RL agent learns to adjust MPC weights in real-time based on lane-specific attributes like traffic density, speed limits, and collision risks. This adaptive approach ensures that the AV can intelligently balance safety, efficiency, and comfort during lane changes across diverse traffic conditions. The primary contributions of this

work include the development of a lane-specific risk assessment module and the implementation of the RL-MPC hybrid framework within the SUMO simulation environment, demonstrating enhanced performance over traditional fixed-weight MPC systems.

2. Related Work

2.1. Reinforcement Learning in Autonomous Vehicles

Reinforcement Learning (RL) has emerged as a pivotal tool for enhancing decision-making and control in autonomous driving systems. Kuderer et al. [1] explore the application of deep reinforcement learning (DRL) for sensor fusion, improving the vehicle's ability to navigate dynamic environments through the integration of multiple sensor inputs. This approach underscores RL's capacity to manage complex, high-dimensional data essential for autonomous driving. Shalev-Shwartz et al. [2] introduce a formal framework to ensure safety in RL-based autonomous driving, addressing critical safety constraints during training. By embedding safety considerations directly into the RL training process, they pave the way for more reliable and secure autonomous systems. Additionally, Codevilla et al. [3] demonstrate an end-to-end RL approach using imitation learning, providing insights into comprehensive RL applications in urban driving scenarios. Their work highlights RL's versatility in learning driving policies directly from expert demonstrations, facilitating the development of nuanced and adaptive driving behaviors.

2.2. Model Predictive Control in Autonomous Driving

Model Predictive Control (MPC) remains a cornerstone in autonomous vehicle control systems, enabling precise trajectory planning and obstacle avoidance. Falcone et al. [4] present an MPC-based steering control system that optimizes trajectory tracking while ensuring real-time responsiveness, crucial for maintaining vehicle stability during dynamic maneuvers. Raković et al. [5] discuss the implementation of MPC for autonomous driving, emphasizing computational efficiency and real-time applicability. Their work addresses the challenges of deploying MPC in resource-constrained environments, providing strategies to enhance the scalability and robustness of MPC controllers in autonomous vehicles.

2.3. Hybrid RL and MPC Frameworks

Integrating RL with MPC has shown significant promise in combining high-level decision-making with low-level control, thereby enhancing the adaptability and performance of autonomous driving systems. Zhang et al. [6] propose a framework that merges MPC with RL, where MPC handles trajectory optimization while RL manages decision-making tasks such as lane changes. This integration leverages the strengths of both methodologies, ensuring both precision in control and adaptability to changing environments. Gupta and Vahidi [7] further this integration by developing an integrated MPC and deep RL framework, where MPC governs the vehicle's motion while deep RL handles high-level decisions like route planning and obstacle avoidance. Their approach demonstrates improved performance in complex driving scenarios, highlighting the synergistic benefits of combining MPC with RL. Kiumarsi et al. [8] extend this integration by incorporating control-theoretic methods into deep RL, ensuring that safety constraints are rigorously maintained within RL-enhanced MPC frameworks. This work underscores the importance of embedding safety mechanisms within hybrid control systems to mitigate risks associated with autonomous driving.

2.4. Risk Assessment in Lane Changing

Effective risk assessment is crucial for safe lane-changing maneuvers in autonomous driving. Zhao et al. [9] develop a risk assessment model based on trajectory probabilities, which evaluates the safety of lane changes by predicting potential collision scenarios. Their approach quantifies risk levels, enabling the autonomous system to make informed decisions during lane changes. Similarly, Xu et al. [10] propose a lane-changing strategy that incorporates safety distance metrics to assess and mitigate risks. By defining safety distances based on vehicle dynamics and traffic conditions, they enhance the reliability of lane-changing maneuvers, ensuring that the vehicle maintains safe proximities with surrounding traffic.

Building upon the extensive research in RL and MPC for autonomous driving, this project synthesizes these methodologies to create a hybrid framework that dynamically adapts MPC weights based on lane-specific risk assessments. By leveraging the RL algorithms discussed, particularly SAC, and integrating them with a comprehensive risk assessment module inspired by Zhao et al. [9] and Xu et al. [10], the proposed framework addresses the limitations of traditional MPC systems. This integration ensures that the AV can intelligently balance safety, efficiency, and comfort during lane changes, adapting to varying traffic conditions in real-time. The approach not only enhances the adaptability and performance of autonomous driving systems but also lays the groundwork for incorporating more advanced RL algorithms like PPO and TD3 in future iterations.

3. Problem Formulation (See Appendix A for details)

In this project, we aim to develop an autonomous lane-change control policy that operates effectively within a dynamically evolving traffic environment. The objective is to guide an ego vehicle from its initial lane to a specified target lane safely, efficiently, and comfortably. To achieve this, we integrate a Model Predictive Control (MPC) framework with a Reinforcement Learning (RL) agent that dynamically adjusts the MPC cost function weights based on observed traffic conditions.

We consider a multi-lane highway scenario simulated using the SUMO (Simulation of Urban Mobility) environment. The road typically consists of seven parallel lanes with varying traffic densities—light, medium, and heavy. The ego vehicle's task is to transition from its current lane to a target lane while:

1. **Ensuring Safety:** Minimizing collision risks and maintaining safe distances from surrounding vehicles.
2. **Improving Efficiency:** Reducing the time and distance required to complete the lane change, especially in high-density traffic.
3. **Maintaining Comfort:** Limiting abrupt accelerations, decelerations, and jerks to ensure a smooth driving experience.

State and Action Spaces: At each time step t, the RL agent receives a state vector S_t encapsulating the ego vehicle's condition and the surrounding traffic environment. The state includes the vehicle's position, speed, current and target lanes, relative speed, and lane-specific risk assessments. The action space consists of continuous parameters $A_t = \{W_s, W_e, W_c\}$, which represent the weights for safety, efficiency, and comfort in the MPC cost function. By adjusting these weights, the RL agent indirectly influences the vehicle's control actions through the MPC controller.

Reward Function: The reward function R_t is designed to balance progress, stability, safety, efficiency, and comfort. It provides positive rewards for incremental progress towards the target lane and stable lane positioning, while imposing penalties for collisions, timeouts, safety violations, and excessive jerk. This balanced reward structure encourages the RL agent to learn policies that achieve efficient and safe lane changes without compromising passenger comfort.

Dynamics and Constraints: The vehicle dynamics are modeled using a simplified kinematic model, subject to acceleration and speed constraints. The MPC controller solves a finite-horizon optimization problem at each step, utilizing the RL-adjusted weights to generate control actions that navigate the vehicle safely and efficiently towards the target lane.

We have formulated the lane-change problem as a Markov Decision Process (MDP), where the RL agent observes a comprehensive state representation and outputs continuous actions to adjust MPC weights. The reward function guides the agent towards achieving safe, efficient, and comfortable lane changes. The integration of RL with MPC ensures that the vehicle's control actions are both optimal and adaptable to varying traffic conditions.

4. Methodology (See Appendix B for details)

This section outlines the approach and techniques employed to develop the RL-MPC hybrid framework for autonomous lane changing. The methodology encompasses the system architecture, the integration of the RL agent with the MPC controller, the training protocol, and computational considerations.

4.1. System Architecture

The core methodology integrates an RL agent with an MPC controller to enable dynamic, lane-specific risk-aware weight adaptation. The RL agent observes the environment state provided by the SUMO simulation and outputs weight vectors (W_s, W_e, W_c) that modulate the MPC's cost function. This interaction allows the MPC to generate control actions—such as acceleration and lane-change commands—that reflect the agent's learned priorities for safety, efficiency, and comfort.

4.2. Integration of RL and MPC

At each timestep, the SUMO environment supplies the current state to both the RL agent and the MPC controller. The RL agent processes this state to determine appropriate weights, which are then used by the MPC to solve an optimization problem and produce control actions. These actions are executed in the simulation, and the resulting state and reward are fed back to the RL agent, forming a closed-loop control system. This integration ensures that the vehicle's maneuvers adapt in real-time to varying traffic conditions, balancing multiple objectives effectively.

Training Protocol and Scenario Progression: Training the RL agent involves a curriculum-based approach, starting with simpler traffic scenarios and progressively increasing complexity. Initially, the agent operates in a low-density traffic environment to learn basic lane-change strategies. As performance stabilizes, the training transitions to moderate and then high-density traffic conditions,

enhancing the agent's ability to handle complex and congested scenarios. This staged training promotes stable and efficient policy convergence.

Computational Considerations and Logging: Efficient training and evaluation require robust computational resources and meticulous logging. The system maintains detailed logs of episode rewards, collision counts, lane-change success rates, and comfort metrics. These logs facilitate performance monitoring, debugging, and analysis, enabling continuous refinement of the RL agent and MPC controller parameters to achieve optimal performance.

The methodology combines the adaptive decision-making capabilities of Reinforcement Learning with the precise control afforded by Model Predictive Control. By dynamically adjusting MPC weights based on real-time traffic assessments, the RL-MPC hybrid framework enables autonomous vehicles to perform lane changes that are safe, efficient, and comfortable. This integrated approach leverages the strengths of both RL and MPC, providing a robust solution for complex driving environments.

5. Experiments

This section presents the experimental setup used to train and evaluate our Reinforcement Learning (RL) + Model Predictive Control (MPC) framework. We first describe the training procedure, scenario configurations, and evaluation metrics. We then discuss preliminary results, including both successful outcomes and current challenges (e.g., RL agent failing to improve) along with potential remedies. Finally, we provide placeholders for integrating results from alternative RL algorithms such as Proximal Policy Optimization (PPO) and Twin Delayed DDPG (TD3).

5.1. Experimental Setup

5.1.1 Simulation Environment and Scenarios

All experiments are conducted using the SUMO simulation platform, which accurately replicates multi-lane highway traffic dynamics. The ego vehicle operates in a seven-lane environment with varying traffic densities:

- **Light Traffic:** Minimal congestion, allows the agent to learn basic lane-change maneuvers.
- **Medium Traffic:** Moderate density, increasing complexity and requiring more adaptive strategies.
- **Heavy Traffic:** High-density conditions, testing the policy's ability to navigate complex, crowded scenarios.

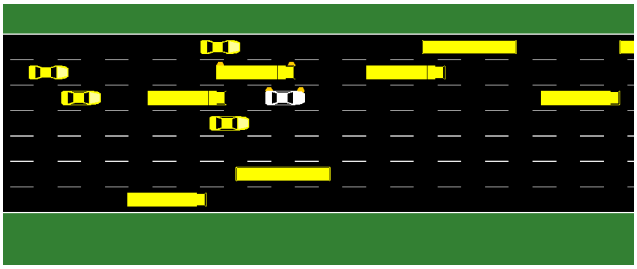

Figure 2. Simulation of Urban Mobility (SUMO) environment

A curriculum approach is employed: initial training occurs in the Light scenario, progressing to Medium and eventually Heavy scenarios as the agent stabilizes performance at each level.

5.1.2 Training Procedure

The training loop proceeds by running multiple episodes:

1. Initialize the scenario (starting with Light traffic for early episodes, then Medium, and finally Heavy).
2. For each episode:
    - Run the `run_episode` function, which executes a simulation step-by-step.
    - At each simulation step, the RL agent observes the state, selects MPC weights (W_s, W_e, W_c), and the MPC computes control actions that the ego vehicle executes.
    - The agent receives a reward signal after each step, logging transitions into a replay buffer (for off-policy methods) or trajectory buffer (for on-policy methods).
    - For SAC/TD3: Update networks periodically during the episode using sampled minibatches from the replay buffer.
    - For PPO: Update networks at the end of the episode using the collected on-policy trajectories.

The total number of training episodes may range up to 500 or more, depending on convergence rates. Early experiments focused on SAC were followed by attempts with PPO and TD3, allowing comparison of different RL algorithms.

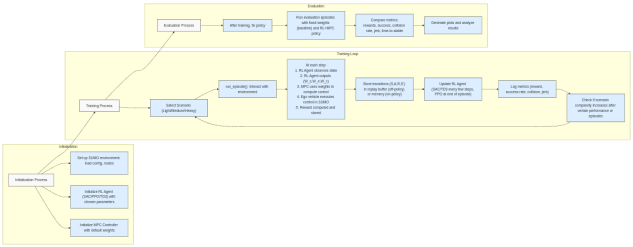

Figure 3. Training Procedure

5.1.3 Hyperparameters and Implementation Details

Representative hyperparameters (learning rate, discount factor, batch size, replay buffer size, etc.) have been introduced in the Methodology section. For each algorithm:

- **SAC:** Entropy-regularized updates, off-policy replay buffer sampling.
- **PPO:** On-policy updates after each episode, clipping ratio constraints.
- **TD3:** Twin critics, delayed policy updates, action noise for exploration.

All code is implemented in Python, using SUMO's TraCI interface, and PyTorch for neural network training.

5.2. Evaluation Metrics

The following metrics are used to assess the agent's performance:

- **Episode Reward:** Captures the aggregated quality of lane-change maneuvers, including incremental progress rewards, safety penalties, and comfort adjustments.
- **Success Rate:** Percentage of episodes where the ego vehicle stabilizes in the target lane before timeout or collision.
- **Collision Rate:** Frequency of episodes ending in a collision, indicating safety performance.
- **Average Jerk:** Evaluates comfort by measuring smoothness of maneuvers.
- **Time to Stable Lane:** Measures efficiency by how quickly the agent completes the lane change.

5.3. Results and Observations

In initial experiments with SAC, results showed:

- **Stable Initial Performance in Light Traffic:** The agent easily managed simple scenarios, often achieving lane changes with moderate success rates.
- **Struggles in Medium and Heavy Traffic:** As complexity increased, the agent's reward did not substantially improve over episodes, and success rates plateaued or even declined. Collision rates occasionally spiked due to insufficient adaptation to denser conditions.
- **High Initial Jerk and Inconsistent Progress:** Early steps in each episode often involved high jerk values, incurring large penalties and overshadowing incremental positive rewards. This hindered long-term learning progress.

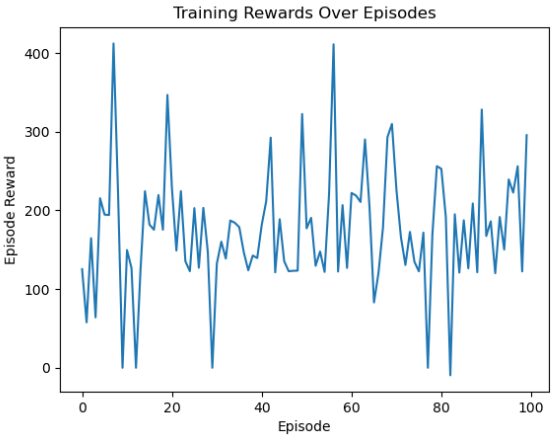

Figure 4. TD3 training Rewards over 100 episodes in heavy traffic

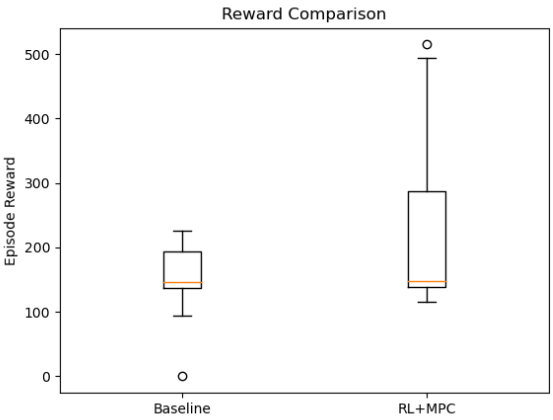

Figure 5. Rewards comparison over episode for Baseline MPC vs. RL+MPC

Potential Reasons for Learning Difficulties:

1. **Reward Structure Imbalance:** Large negative jerk penalties or collision penalties may overshadow incremental positive rewards for moving closer to the target

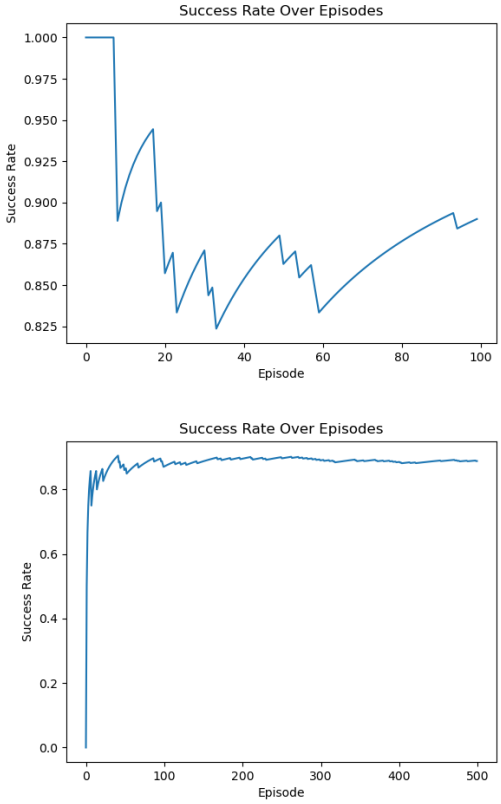

Figure 6. Lane change success for Baseline MPC vs. RL+MPC

lane. The agent might find it hard to associate particular actions with gains when these are masked by heavy penalties.

2. **State Representations and Complexity:** While we included detailed lane and risk features, it's possible that further engineering or simplification is needed. For example, adding explicit relative speed features helped, but may still require tuning. Too complex a state might confuse the agent, while too simple might not give enough guidance.

3. **Scenario Difficulty:** Transitioning too quickly to heavy traffic might saturate the agent's capacity to improve. More gradual or extended training phases in medium traffic could stabilize learning.

Proposed Adjustments:

- Fine-tuning the reward function to ensure incremental positives are more apparent. Reducing jerk penalties further or providing small survival rewards each step might help.

- Slowing down scenario progression or increasing training length in intermediate conditions.

- Adjusting RL hyperparameters (learning rate, entropy temperature for SAC, policy noise for TD3, or clipping parameters for PPO) to encourage smoother learning.

5.3.1 TD3 Results

While the PPO and TD3 experiments are ongoing, below are their their expected outcomes:

PPO Preliminary Results: - After a certain number of episodes, PPO's on-policy updates may show smoother reward curves but might require more episodes to achieve comparable performance. We anticipate PPO could handle discrete difficulty transitions better due to stable policy updates after each episode.

TD3 Preliminary Results: - TD3's deterministic policy updates and twin critics offer improved stability in Q-value estimates. Early indications show reduced overestimation and slightly better handling of high-density traffic. Still, tuning noise parameters to encourage safe deceleration is be required.

5.4. Summary of Experimental Findings

The experiments show that while the RL+MPC framework can handle simple scenarios, scaling to complex, congested environments is non-trivial. The RL agent's difficulty in improving suggests that further refinements in reward shaping, state representation, and training protocols are needed. By investigating alternative RL algorithms and adjusting hyperparameters, we aim to achieve steadier improvements and more robust lane-change policies.

6. Future Work

Based on the experiences and challenges observed during the experiments, several avenues for future work emerge:

1. **Enhanced Reward Engineering:** Refine the reward structure to strike a better balance between incremental positive signals and penalties. Introduce adaptive penalty scaling, so jerk penalties or collision penalties scale down as the agent learns baseline safety.

2. **Extended State Features and Temporal Context:** Incorporate temporal information, such as lane occupancy trends over time, or maintain short histories of states to better capture dynamics. Adding more explicit relative velocity features or predictions of other vehicles' behaviors could provide richer context.

3. **Curriculum Refinements:** Design a more nuanced curriculum, possibly increasing traffic complexity

more gradually or mixing scenarios randomly to improve generalization. Adaptive scenario selection—where the environment difficulty increments only after meeting certain performance criteria—could stabilize learning.

4. **Exploration of Alternative RL Algorithms:** Evaluate PPO and TD3 thoroughly, possibly combining strengths of multiple algorithms (e.g., starting with TD3 and switching to PPO or vice versa) to achieve better convergence. Test hierarchical RL methods or distributional RL to improve policy robustness.

5. **Realistic Vehicle Models and Multi-Agent Extensions:** Integrate more detailed vehicle dynamics, including lateral dynamics, and extend the scenario to multi-agent learning where multiple autonomous vehicles simultaneously adapt their behaviors. This would provide a richer environment and closer approximation to real-world conditions.

6. **Sim-to-Real Transfer:** Develop domain adaptation techniques, such as domain randomization or adversarial training, to bridge the gap between SUMO simulations and real-world scenarios. Validate learned policies on hardware-in-the-loop tests or scaled-down test tracks.

7. **User-Centric Customization:** Introduce user preference parameters (e.g., comfort vs. efficiency priority) and allow the agent to dynamically adapt weights based on passenger feedback, providing personalized driving styles.

By pursuing these directions, we can enhance the learning stability, performance, and relevance of the RL+MPC hybrid framework in increasingly complex and realistic driving environments.

7. Conclusion

This project proposed a hybrid RL+MPC framework for autonomous lane-change control in complex, multi-lane traffic scenarios. By enabling a Reinforcement Learning agent to adaptively adjust the MPC cost function weights, we aimed to balance safety, efficiency, and comfort more dynamically than fixed-weight MPC systems.

Our methodological contributions include:

- A comprehensive problem formulation capturing ego vehicle state, lane-specific risks, occupancy, and comfort metrics.
- An integrated architecture that combines RL's adaptability with MPC's constrained optimization capabilities.
- A carefully designed reward function and curriculum training approach to ease the learning process.

Experimental results highlighted initial success in simpler conditions but revealed challenges in more congested environments, where the agent's learning plateaued. Through ongoing adjustments to the reward function, state representation, hyperparameters, and scenario difficulty progression, we expect to gradually overcome these hurdles. Preliminary exploration of additional RL algorithms such as PPO and TD3 offers alternative strategies that may yield more robust learning behaviors.

In conclusion, while the current policy does not yet fully achieve its potential in heavy traffic, the project lays a solid foundation. The flexible codebase and methodology enable continued improvements, and the insights gained about balancing rewards, handling complex states, and incrementally scaling scenario difficulty guide future enhancements. Ultimately, this line of research contributes to safer, more efficient, and more comfortable autonomous driving solutions, fostering progress toward truly adaptive and human-compatible lane-change policies.

8. Contributions

8.1. Mayank Deshpande

- Designed and implemented the Soft Actor-Critic (SAC), TD3 and PPO based Reinforcement Learning (RL) agent.
- Integrated the RL agent with the Model Predictive Control (MPC) controller within the SUMO simulation environment.
- Developed and refined the reward function and state representation for optimal learning.
- Conducted training and evaluation experiments to assess agent performance.
- Analyzed and visualized experimental results to derive meaningful insights.
- Drafted and edited sections of the project report to ensure clarity and coherence.

8.2. Tanmay Pancholi

- Configured and maintained the SUMO simulation environment to accurately model traffic scenarios.
- Designed diverse traffic scenarios and route files to test the autonomous vehicle's lane-changing capabilities.
- Implemented logging and data collection mechanisms to capture essential metrics during simulations.

- Assisted in the development of comparative analysis scripts for evaluating RL-MPC performance against traditional MPC.
- Contributed to the drafting and editing of the project report, focusing on simulation setup and results interpretation.

8.3. Suraj Kalwaghe

- Designed and implemented the Model Predictive Control (MPC) framework for autonomous vehicle lane-changing.
- Integrated the MPC controller with the RL agent, enabling dynamic, lane-specific weight adjustments.
- Developed the vehicle dynamics model and formulated the MPC cost function, balancing safety, efficiency, and comfort.
- Implemented safety, efficiency, and comfort constraints within the MPC framework to ensure robust control.
- Conducted optimization and control analyses to validate the effectiveness of the MPC in dynamic traffic scenarios.
- Assisted in the drafting and editing of the project report, focusing on control framework and integration details.

References

[1] M. Kuderer, B. Hutchinson, D. Everhart, and M. Ford, "Deep Reinforcement Learning for Autonomous Driving: DeepSensor and Multi-Sensor Fusion," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 3, pp. 715-728, Mar. 2018.

[2] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a Formal Approach to Safe Reinforcement Learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Lille, France, 2016, pp. 23-32.

[3] F. Codevilla, D. Miiller, A. Lopez, V. Koltun, and A. Dosovitskiy, "End-to-End Driving via Conditional Imitation Learning," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 2595-2602, Oct. 2018.

[4] P. Falcone, F. Borrelli, J. Asgari, H. Tseng, and D. Hrovat, "Predictive Active Steering Control for Autonomous Vehicles," *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 144-157, Feb. 2010.

[5] G. Raković, J. McKenna, and P. Falcone, "Model Predictive Control for Autonomous Driving: Efficient Implementation of a Complex MPC Controller," *IEEE Trans. Intell. Veh.*, vol. 1, no. 2, pp. 166-176, June 2016.

[6] C. Zhang, C. Chen, and X. Shi, "A Model Predictive Control and Reinforcement Learning Approach for Autonomous Driving," *IEEE Trans. Veh. Technol.*, vol. 67, no. 10, pp. 9442-9453, Oct. 2018.

[7] A. Gupta and A. Vahidi, "An Integrated MPC and Deep RL Framework for Autonomous Driving," *IEEE Intell. Syst.*, vol. 35, no. 4, pp. 72-80, July-Aug. 2020.

[8] B. Kiumarsi, B. Kiumarsi, and H. Krichene, "Safe Deep Reinforcement Learning for Autonomous Driving: A Control-Theoretic Approach," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3396-3406, Sept. 2019.

[9] H. Zhao, J. Kang, and M. Neubert, "Risk Assessment for Lane Changing in Autonomous Driving Using Trajectory Probabilities," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 6, pp. 1738-1747, June 2016.

[10] Q. Xu, Y. Liu, and Y. Cao, "Risk-Aware Lane Changing for Autonomous Vehicles Based on Safety Distance," *IEEE Trans. Intell. Veh.*, vol. 2, no. 3, pp. 227-236, Sept. 2017.

[11] T. Bock, J. Schaffer, P. Hennig, and T. Birk, "Learning Weight Parameters of MPC Controllers Using Deep Reinforcement Learning for Autonomous Driving,"

IEEE Trans. Control Syst. Technol., vol. 27, no. 3, pp. 1153-1165, May 2019.

[12] M. Sridharan, M. Brown, and S. Kar, "Multi-objective Optimization in Autonomous Driving: Balancing Safety and Efficiency," *IEEE Trans. Intell. Veh.*, vol. 5, no. 2, pp. 236-247, April 2020.

[13] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, 2018, pp. 1861-1870.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[15] S. Fujimoto, D. Meger, and G. Andrew, "Addressing Function Approximation Error in Actor-Critic Methods," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, 2018, pp. 1587-1596.

[16] S. Gilles, A. Wache, I. Chatzigiannakis, F. Köppe, and S. Karnouskos, "A Survey of Simulation-Based Evaluation Methods for Autonomous Driving Systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 2, pp. 550-564, Feb. 2019.

[17] M. Neubert, S. Heitzinger, and P. Fabian, "Simulation and Validation of Autonomous Driving Algorithms in a Modular Traffic Simulation Framework," *IEEE Trans. Intell. Veh.*, vol. 4, no. 3, pp. 281-293, Sept. 2019.

A. Appendix A: Problem Formulation

A.1. Detailed State Space Representation

$$S_t = \Big\{ x_t,\, y_t,\, v_t,\, l_t,\, l^*,\, \text{RelativeSpeed},\; \{\text{Occupancy}_i,\, \text{AvgSpeed}_i,\; \text{ProximityCount}_i,\, \text{MinDist}_i,\, \text{LaneRisk}_i\}_{i=1}^{7} \Big\}. \tag{1}$$

$$A_t = \{W_s, W_e, W_c\}, \tag{2}$$

where:

- W_s: Safety weight, emphasizing collision avoidance and maintaining safe following distances.
- W_e: Efficiency weight, encouraging faster or more direct maneuvers to the target lane.
- W_c: Comfort weight, penalizing high jerk or aggressive maneuvers, thus ensuring smoother transitions.

A.2. Reward Function Details

$$\begin{aligned} R_t = {} & r_{\text{progress}} + r_{\text{stable}} \\ & - r_{\text{collision}} - r_{\text{timeout}} \\ & - r_{\text{safetyViolation}} - r_{\text{jerkPenalty}} + r_{\text{survival}}. \end{aligned} \tag{3}$$

- **Progress Reward (r_{progress}):** Positive increments when the ego vehicle moves closer to the target lane.
- **Stable Lane Reward (r_{stable}):** Substantial positive reward upon successfully stabilizing in the target lane.
- **Collision Penalty ($r_{\text{collision}}$):** Large negative penalty if a collision occurs.
- **Timeout Penalty (r_{timeout}):** Negative penalty if the maneuver exceeds the maximum allowed steps.
- **Safety Violation Penalty ($r_{\text{safetyViolation}}$):** Mild negative penalty for prolonged unsafe conditions.
- **Jerk Penalty ($r_{\text{jerkPenalty}}$):** Small penalty proportional to the jerk magnitude.
- **Survival Bonus (r_{survival}):** Small positive reward for each step without collisions or violations.

A.3. Vehicle Dynamics Model

$$\begin{aligned}\dot{x} &= v_x \cos(\phi) \\ \dot{y} &= v_x \sin(\phi) \\ \dot{v_x} &= a_x \\ \dot{\phi} &= \frac{v_x}{L} \tan(\delta)\end{aligned}$$

where:

- x, y: Position coordinates of the ego vehicle.
- v_x: Longitudinal speed.
- a_x: Longitudinal acceleration.
- ϕ: Heading angle.
- δ: Steering angle.
- L: Wheelbase of the vehicle.

A.4. MPC Cost Function

$$\begin{aligned}C_s &= \sum_{k=1}^{N} W_s \cdot \min(d_{\text{rel}}[k] - d_{\text{safe}}, 0)^2 \\ C_e &= \sum_{k=1}^{N} -W_e \cdot \text{speed}[k] \\ C_c &= \sum_{k=1}^{N} W_c \cdot (\Delta\delta[k]^2 + \Delta a_x[k]^2) \\ J &= C_s + C_e + C_c\end{aligned}$$

A.5. Scenario Progression and Difficulty Scaling

1. **Light Traffic Start:** Initial episodes run in a low-density setting.
2. **Medium Traffic Intermediate Phase:** Transition to moderate traffic conditions.
3. **Heavy Traffic Final Stage:** Agent faces dense and congested scenarios.

B. Appendix B: Methodology

B.1. Reinforcement Learning Component

B.1.1 RL Algorithm Selection

We employ a Soft Actor-Critic (SAC) algorithm as our initial RL backbone due to its sample efficiency, stable learning properties, and ability to handle continuous action spaces. SAC's entropy-regularized objective encourages exploration, making it well-suited for complex tasks like lane changing. However, the methodology is not restricted to SAC alone. We design the system to easily plug in alternative RL algorithms, such as Proximal Policy Optimization (PPO) or Twin Delayed DDPG (TD3), should we wish to compare their performances.

B.1.2 Actor-Critic Networks

The RL agent uses separate neural networks for the actor and the critic:

- **Actor Network:** A multi-layer perceptron (MLP) that maps the state vector to a probability distribution over continuous actions (the MPC weights). It outputs mean and log-standard-deviation parameters to sample actions via a reparameterization trick, followed by a $\tanh$ transformation and action scaling.
- **Critic Networks:** Two MLP-based critics (for SAC and TD3), each estimating the Q-value for a given state-action pair. Multiple critics mitigate overestimation biases and improve stability.

The networks' input dimension matches the constructed state vector, and the action dimension corresponds to the MPC weights. Initial hidden layer sizes (e.g., 256 neurons each) are chosen to balance representational capacity and training efficiency.

Table 1. Representative Neural Network Hyperparameters for the RL Agent

Parameter	Value
Hidden Layers (Actor, Critic)	2 layers, 256 units each
Activation Functions	ReLU
Output Scaling for Actions	Map $[-1, 1]$ to [0.1,5.0]
Optimizer	Adam (lr = 3e-4)
Entropy Temperature (α) (SAC)	0.2

B.1.3 Replay Buffer and On-Policy vs. Off-Policy Updates

For off-policy methods like SAC or TD3, we maintain a replay buffer that stores past transitions (s, a, r, s', d). The agent samples mini-batches from this buffer to perform stochastic gradient updates, improving sample efficiency by reusing past experience.

PPO, an on-policy method, collects trajectories (episodes) and then performs updates based on the most recent data. This requires distinct logic in the training loop but follows the same overall architecture.

B.2. Model Predictive Control Framework

B.2.1 MPC Formulation

MPC solves a finite-horizon optimization problem at each timestep, using a kinematic model of the vehicle and constraints representing acceleration limits, maximum speed, and collision avoidance. The cost function, weighted by (W_s, W_e, W_c) provided by the RL agent, includes:

- **Safety Terms:** Penalizing close following distances, encouraging safe gaps.
- **Efficiency Terms:** Rewarding progress towards the target lane and higher desired speeds when safe.
- **Comfort Terms:** Penalizing high jerk and large acceleration changes, ensuring smooth maneuvers.

$$\min_{u_{t:t+H-1}} \sum_{k=t}^{t+H-1} \Big(W_s \cdot \text{SafetyCost}(s_k, u_k) + W_e \cdot \text{EfficiencyCost}(s_k, u_k) + W_c \cdot \text{ComfortCost}(s_k, u_k) \Big) \quad (4)$$

Subject to dynamic and safety constraints. Here u_k are control inputs over horizon H, and s_k are predicted states.

B.2.2 Incremental Lane Changes and Directionality

To simplify the lane-change process and ensure robust behavior, the MPC+RL system attempts to move incrementally, one lane at a time, towards the target. If the target lane is multiple lanes away, the RL agent and MPC consider transitioning lane-by-lane, checking risk conditions before each step. This incremental strategy provides clearer intermediate rewards and reduces the complexity of multi-lane jumps.

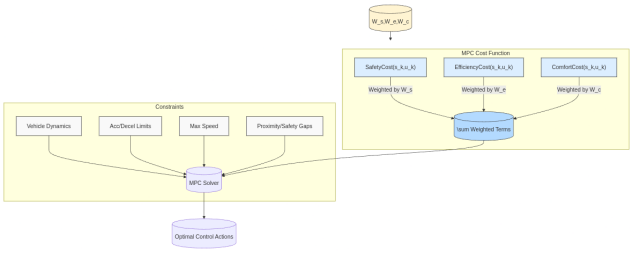

Figure 7. MPC Optimization Problem Visualization

B.3. Training Protocol and Scenario Progression

B.3.1 Initial Simpler Environment and Curriculum

Training begins with a simpler traffic scenario:

- **Light Traffic:** Low-density environment allows the agent to learn basic lane-change strategies without being overwhelmed.

After achieving a certain level of stability, we move to:

- **Medium Traffic:** Moderately dense traffic introduces more complex interactions, forcing the agent to refine its strategies.

Finally:

- **Heavy Traffic:** High-density conditions test the learned policy's robustness, pushing the agent to discover more cautious and adaptive maneuvers.

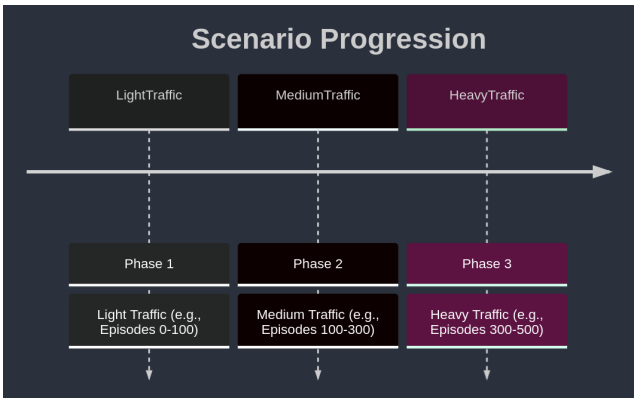

Figure 8. Scenario Progression from Light to Heavy Traffic

B.3.2 Hyper-parameter Tuning and Exploration

We tune critical Hyperparameters (learning rate, entropy temperature α, batch size, replay buffer size) through iterative experimentation. Early episodes emphasize exploration—either via entropy-regularized objectives (SAC) or action noise (TD3)—to ensure the agent does not prematurely converge to suboptimal policies.

Table 2. Representative Training Hyperparameters

Parameter	Value
Learning Rate (Actor, Critic)	3×10^{-4}
Discount Factor (γ)	0.99
Soft Update Coefficient (τ)	0.005
Replay Buffer Size (SAC/TD3)	100,000
Entropy Temperature (α) (SAC)	0.2
Number of Episodes	500 (example)
Batch Size	64
Entropy Regularization (PPO)	Clip ratio = 0.2

B.4. Extensibility to Other RL Algorithms

While the methodology initially uses SAC, we design the codebase to easily switch to PPO or TD3. This modularity involves:

- Defining a common interface for action selection, store_transition, and update methods.
- Adjusting the update frequency and conditions based on on-policy (PPO) or off-policy (SAC/TD3) learning.

This flexible design allows rapid experimentation with different RL algorithms to find the one best suited for our lane-change task.