Reinforcement Learning Approach on Autonomous Driving

Kangsheng Qi **Jaime Ren**

Abstract

Autonomous driving refers to the ability of a vehicle to navigate safely and efficiently to a specified destination without direct human intervention. One approach to achieving this is reinforcement learning (RL), which trains an agent to interact with a simulated environment using an objective function. In this project, we aim to improve an RL algorithm for autonomous vehicle parking. We incorporate concepts from residual learning and uncertainty quantification to enhance agent training. Our methods show qualitative improvements over the baseline, suggesting that with more refined design and tuning, performance can be further improved.

1 Introduction

Since their introduction in the early 1900s, automobiles have become an integral part of human society. However, self-driving vehicles remained a fantasy science fiction novels until the past decade. Enabled by recent advancements in computational algorithms and increase in hardware, companies such as Tesla, Waymo, and Cruise have deployed their own versions of autonomous vehicles on public roads. Despite these developments, numerous challenges remain, particularly in areas such as safety guarantees, algorithmic robustness, and societal impact [Bag+16].

The problem of autonomous driving can be formulated into Reinforcement Learning–Markov Decision Process (RL-MDP) framework. RL is a machine learning paradigm in which an agent learns to make decisions by interacting with an environment to maximize a predefined reward function. The Markov Decision Process (MDP) provides a natural way to formulate the said sequential decision-making tasks [Dha23].

In this final project [Baj25], we aim to improve upon an existing robust autonomous driving algorithm as published by Leurent *et al.* [Leu+19]. The baseline MDP-based algorithm can be written as $(\mathcal{S}, \mathcal{A}, T, r)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ is the action space, T represents the transition dynamics, and r is the reward function that encodes the desired learning objective. The agent's goal is to learn a policy $\pi \in \mathcal{M}(\mathcal{A})^{\mathcal{S}}$ that maximizes the expected cumulative discounted reward, defined as

$$R_\pi^T(s) := \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \qquad v_\pi^T := \mathbb{E}[R_\pi^T(s)] \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor, $a_t \sim \pi(s_t)$, $s_{t+1} \sim T(s_{t+1}|s_t, a_t)$, $\mathcal{M}(X)$ is the probability distribution over X, and the expectation is taken over the trajectory distribution induced by the policy π and transition dynamics T.

[Leu+19] proposes a robust control framework that defines the risk (negative outcome when agent performs sub-optimally) of a policy as the worst-case outcome within an ambiguity set of deterministic dynamics models, represented as $s' = T_\theta(s, a)$. This formulation ensures a lower bound on performance in real-world scenarios by optimizing the policy to maximize the minimum expected return across all models in the ambiguity set:

$$\max_{\pi} \min_{T_\theta : \theta \in \Theta} v_\pi^T \tag{2}$$

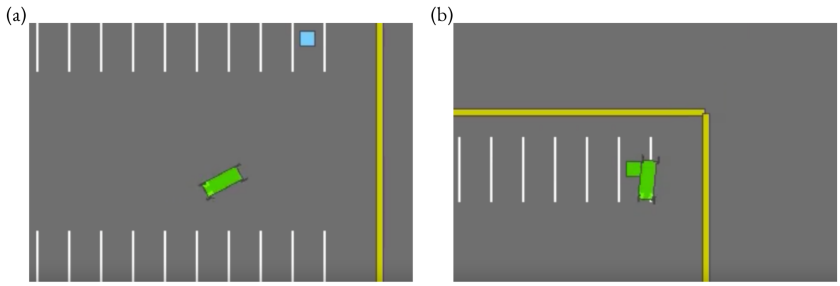

Figure 1: Baseline environment. **(a)** The ego-vehicle (green) is aimed to park at the location indicated by the cyan colored box. **(b)** The ego-vehicle parked at the specified location.

However, this optimization problem is generally intractable. To address this, the authors proposed two tractable approximation methods respectively targeting finite and continuous ambiguity sets and action spaces. Within the finite ambiguity sets case, a infinite look-ahead tree $\mathcal{T}$ of reachable states is built such that each node corresponds to a sequence of actions. During the exploration and expansion of the tree, each node's robust worst case return u^r and optimistic bound b^r are be calculated according to a derivation from [HM08]. At each time step, the $\{u_i^r, b_i^r\}$ for all node leaves i are calculated and the leaf with the largest b^r is selected. For the continuous case, the reachability set is defined with bounds such that

$$S(t, s_0, \pi) := \{s_t : \exists \theta \in \Theta \; s.t. \; s_{k+1} = T_\theta(s_k, a_k), a_k = \pi(s_k), \forall k \in [t]\}$$

which is approximated with its interval hull. This is then optimized with a surrogate lower-bound objective $\hat{v}_\pi := \sum_{t \in [H]} \gamma^t \min_{\text{hull}} r(s, \pi(s))$. We refer interested readers to the original publication for more details.

We chose to work within the parking environment, which is outlined in Figure 1 as the baseline. With the goal of enhancing performance, we explored ideas of residual networks and residual learning to improve model optimization and better capture complex vehicle dynamics. To more closely approximate real world driving environment, we also explored uncertainty quantification through the inclusion of Gaussian noise and corresponding updates to the loss function. Our methods do offer a slight improvement over the baseline algorithm, and we believe that with further tuning and more precise design choices, these enhancements could become more e concrete.

The remainder of this report is organized as follows. In Section 2, we discuss related works in RL based autonomous driving. In Section 3, we describe our experimental setup. Section 4 presents the results of our study. Finally, we conclude the report in Section 5. The code written for this project are as follows: the baseline method can be accessed within here and the code for our improved method can be found here.

2 Related Works

There exist a rich set of literature that explores the problem of autonomous driving through RL. We note that most of the related works we explored implements a model-free framework as compared to the model-based framework taken by the baseline algorithm. We believe that most of the ideas/approaches offered by these works can be readily applied within a model-based setting. Other robust control theory based works have sought to introduce defense-aware robust RL framework that includes a adversarial attacker [HHL24]. Here, the adversarial attacker, integrating Bayesian optimization with fast gradient sign attacks, seeks to maximize

$$\mathbb{E}[C_a^\pi(\Delta^m s + \Delta^a), \tilde{a}]$$

which is the expected value of the cost of the action under the constrained multiplicative and additive perturbation Δ^m and Δ^a. The goal of the defender, on the other hand, is to maximize the objective as outlined in Equation 2 in a constrained action cost optimization. A separate work [AMG20] developed a robust RL-based agent trained using Deep Q-Networks (DQN) [Mni+13] with a convolutional neural network (CNN) processing pipeline. The CNN processes image inputs to capture navigation

dynamics effectively, enabling the agent to perform lane-following tasks reliably in both simulated and real-world settings.

Since real-world driving constantly involves the interaction between multiple vehicles, a growing body of work has explored the applications of applied multi-agent reinforcement learning (MARL) to model cooperative behaviors among different vehicles. [Shi+20] integrates MARL algorithm with graph neural networks by representing each vehicle (agent) as a node and edges as links to neighboring vehicles. At each timestep t, each agent learns about its current state and the state of its neighbors before performing an action $a_t \in \mathcal{A} = [a_{dec}, a_{acc}]$. The reward function is designed under a shared-policy framework that promotes shared high average speeds. Each agent is able to capture the spatial and temporal interaction of its neighbors through a Gaussian process regression model defined as

$$K(x_{ego}, x_{nbhd}) = A \cdot \exp\left(-\frac{(x_{ego} - x_{nbhd})^2}{2\sigma^2}\right)$$

where A is an amplitude constant, and σ is a decaying length scale constant with respect to the distance. The network information is then extracted through a graph convolution layers with attention mechanisms [Vas+17].

A major challenge in multi-agent systems is the problem of fair credit assignment: how to distribute a global reward such that each agent is appropriately incentivized to cooperate. [Han+22] formalizes this as a Transferable Utility (TU) game: $(\mathcal{N}, f)$, where $\mathcal{N}$ is the optimal cooperation between all agents and f measures the total reward of any cooperation $\mathcal{C} \subseteq \mathcal{N}$. Within a TU game, the Shapley value for a single agent i is defined as

$$\phi_i(f) := \sum_{\mathcal{C} \subseteq \mathcal{N} \setminus \{i\}} \frac{|\mathcal{C}|!(|\mathcal{N}| - |\mathcal{C}| - 1)!}{|\mathcal{N}|!}(f(\mathcal{C} \cup \{i\}) - f(\mathcal{C})) \tag{3}$$

We defer to [Cae22] for an explanation of the intuition behind Equation 3. Given this, we can adjust (1) into

$$R_\pi^T(s) := \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, \mathcal{N}) \qquad v_\pi^T := \mathbb{E}[R_\pi^T(s)] \tag{4}$$

to take into account the desired total cooperation. Here, the value of any cooperation $\mathcal{C}$ can be written similarly as Equation 4 by replacing $\mathcal{N}$ with $\mathcal{C}$, and the Shapley value in (3) can also be extended to

$$\phi_i(s, a) := \sum_{\mathcal{C} \subseteq \mathcal{N} \setminus \{i\}} \frac{|\mathcal{C}|!(|\mathcal{N}| - |\mathcal{C}| - 1)!}{|\mathcal{N}|!}(f(\mathcal{C} \cup \{i\} \,|s, a) - f(\mathcal{C}|s, a)) \tag{5}$$

to incorporate the state and action information into the marginal contributions. This formulation results in a convex game with a stable and efficient payoff structure.

The actions required for navigating a vehicle can sometimes be decomposed into levels of increasingly difficult sub-actions. [Dua+20] took this hierarchical reinforcement learning approach to train the agent to first learn the sub-policy of individual lane switching before aggregated together to form the master policy to determine which action to take at each time step.

An alternative approach seeks to integrate deep learning into RL framework to enable agents to learn optimal policies in high-dimensional environments. An example of this approach involves using a deep neural network to train the ego-vehicle's policy on whether to overtake another vehicle [Che+21]. While the problem is framed within a hierarchical framework that captures cooperative and non-cooperative interactions through modeling diverse driving behaviors among non-ego vehicles, the RL algorithm, through deep learning, nevertheless seeks to maximize the expected return through gradient descent:

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau \left[\sum_{t=0}^{T} \nabla_\theta \log(\pi_\theta(s_t)) \sum_{t=0}^{T} r(s_t, a_t)\right] \tag{6}$$

where π_θ is the explicit form of policy π, τ is a trajectory with $\pi(\tau)$ being its likelihood, and the reward function is composed of multiple components to encourage safe and efficient driving behaviors.

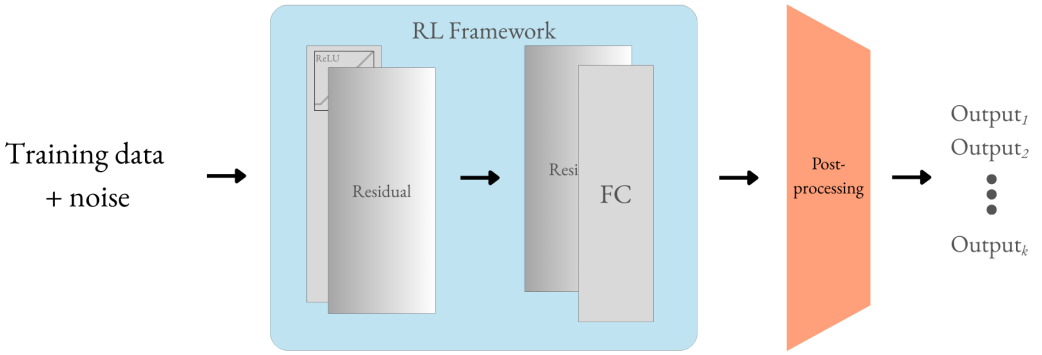

Figure 2: Model architecture. Training data is first augmented with Gaussian noise before being passed into a reinforcement learning (RL) framework with residual connections.

3 Methods

The overall architecture of our approach is illustrated in Figure 2, which we will refer to as "residual model". We train the model using the ADAM optimizer [KB14] with a learning rate of 0.01. Below, we detail the key modifications introduced to improve model performance.

3.1 Residual network

Residual networks [He+16] enable the training of deep neural networks by improving gradient flow, achieving faster convergence, enhancing robustness, and presenting a more stable optimization. Inspired by these benefits, we incorporated residual connections into our architecture.

We reason this integration with deep learning ideas would enable the model to better capture complex dynamics inherent to autonomous driving and improves its ability to differentiate between diverse driving states.

3.2 Bounded noise perturbation

Real-world driving occurs in highly dynamic environments where noise and uncertainty are constant. Human drivers implicitly process noisy inputs into "clear" signals, which are then used to make decisions. Therefore, we reason that training an agent solely on idealized, noise-free data may hinder its performance in practical applications.

To address this, we explored uncertainty estimation in order to improve multi-object tracking. To that end, we introduced a bounded perturbation within both the state and action at each time step. Specifically,

$$s_{t+1} \sim T(s_{t+1}|s_t, a_t) + \delta_t^s; a_t \sim \pi_\theta(a|s_t) + \delta_t^a; \text{ where } ||\delta|| \leq \epsilon \tag{7}$$

Here, the noise terms δ_t^s and δ_t^a are sampled from Gaussian distributions. This perturbation aims to approximate the unpredictability of real-world driving, thereby improving the model's robustness and generalization.

3.3 Modified loss function

To account for the introduced noise, we revised the original loss function. The baseline mean squared error (MSE) loss is

$$\mathcal{L}(T_\theta) = \mathbb{E}[||T_\theta(s_t, a_t) - s_{t+1}||^2]$$

In our framework, we applied the model to perturbed inputs, where $\tilde{s}_t = s_t + \delta_t^s$ and $\tilde{a}_t = a_t + \delta_t^a$, leading to the updated loss:

$$\mathcal{L}(T_\theta) = \mathbb{E}[||T_\theta(\tilde{s}_t, \tilde{a}_t) - s_{t+1}||^2]$$

We hope this adjustment would simulate the uncertainties within real-world situation and produce a more generalizable model.

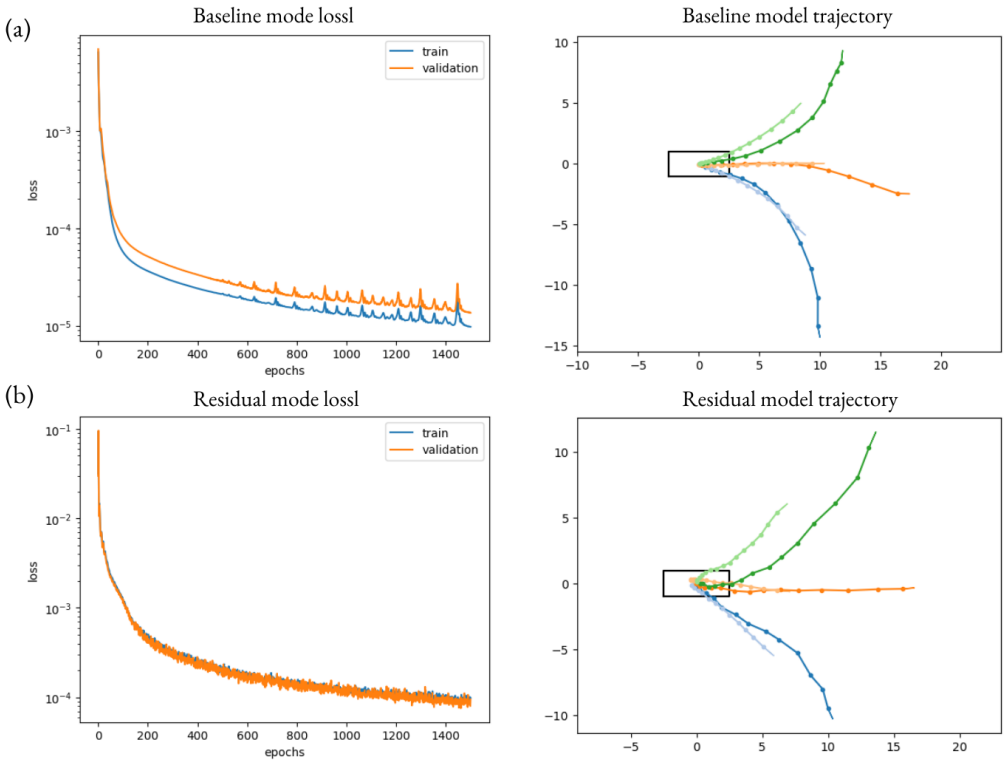

Figure 3: Results. **(a)** Baseline results. **(b)** Residual model results.

4 Results

When first attempting to use the code base provided by [Leu+19] and [Baj25], we encountered significant difficulties due to software and system updates, as well as logic issues that prevented the baseline algorithm from running smoothly in the parking scenario. Consequently, much of our early work was devoted to maintaining and fixing the baseline code to ensure proper functionality.

Once the baseline was running correctly, we proceeded to implement our model. The results are presented in Figure 3. At first glance, the original model achieved a lower loss and exhibited a smoother loss curve, which was characterized by a relatively thin line and only a few small spikes near the end. In contrast, the residual model displayed greater variance in the loss across epochs as well as more fluctuation within each window of training.

However, a closer inspection revealed that the baseline model's loss curve consisted of several significant spikes, which could reflect late instability in the learned policy. Meanwhile, despite its higher loss variance, the residual model consistently produces smoother and more symmetrical trajectories. This suggests that the residual model explored a wider range of policy behaviors and learns to generalize more effectively under uncertainty. This broader search space could be inferred from its loss variability, which indicates that the agent avoids committing to a narrow policy and instead adapts more robustly to noise.

Therefore, while the baseline model shows a cleaner loss curve, the residual model demonstrates stronger and more stable decision-making, particularly evident in the improved trajectory outcomes.

The baseline model's trajectories tend to be less symmetrical and more inconsistent across trials, suggesting that the agent is not converging to an optimal or robust policy. This could be a result of incomplete updates to the action distribution or inaccurate state estimation, which would require more experiments to verify. In contrast, the residual model generates more regular and symmetrical trajectories. This symmetry is desirable because it reflects more consistent decision-making. The improvement is likely due to precomputing the action distributions and explicitly accounting for uncertainty in state transitions, both of which help the policy generalize more effectively.

Overall, these differences support the idea that our residual model leads to qualitatively better behavior as compared to the baseline.

5 Conclusion

Building on the baseline algorithm from [Leu+19] and [Baj25], we explored techniques from residual learning and uncertainty quantification with the goal of obtaining better learning and more accurate modeling of the environment. Our methods showed qualitative improvements over the baseline, suggesting that with better design and careful turning, performance could be further enhanced.

While the proposed methods offer potential improvements, autonomous driving still remains a challenging task with many problems needing to be addressed, thus motivating further research in this area.

Acknowledgment

A large language model [Ope25] was used in the writing of this work.

References

[Bag+16] Saeed Asadi Bagloee et al. "Autonomous vehicles: challenges, opportunities, and future implications for transportation policies". In: *Journal of modern transportation* 24 (2016), pp. 284–303.

[Dha23] Binay Dhakal. *MDP (Markov Decision Process) — RL (Reinforcement Learning)*. 2023. URL: https://medium.com/@binaydhakal35/mdp-markov-decision-process-rl-reinforcement-learning-bc85e5d25031#:~:text=Reinforcement%20Learning%20is%20the%20science,are%20solved%20with%20reinforcement%20learning..

[Baj25] Chandrajit Bajaj. *CS 378 Final Project*. 2025. URL: https://utexas.instructure.com/courses/1414435/files/folder/Assignments2025?preview=84146117.

[Leu+19] Edouard Leurent et al. "Approximate robust control of uncertain dynamical systems". In: *arXiv preprint arXiv:1903.00220* (2019).

[HM08] Jean-Francois Hren and Rémi Munos. "Optimistic planning of deterministic systems". In: *European Workshop on Reinforcement Learning*. Springer. 2008, pp. 151–164.

[HHL24] Xiangkun He, Wenhui Huang, and Chen Lv. "Trustworthy autonomous driving via defense-aware robust reinforcement learning against worst-case observational perturbations". In: *Transportation Research Part C: Emerging Technologies* 163 (2024), p. 104632.

[AMG20] Péter Almási, Róbert Moni, and Bálint Gyires-Tóth. "Robust reinforcement learning-based autonomous driving agent for simulation and real world". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8.

[Mni+13] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[Shi+20] Tianyu Shi et al. "Efficient connected and automated driving system with multi-agent graph reinforcement learning". In: *arXiv preprint arXiv:2007.02794* (2020).

[Vas+17] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[Han+22] Songyang Han et al. "Stable and efficient Shapley value-based reward reallocation for multi-agent reinforcement learning of autonomous vehicles". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 8765–8771.

[Cae22] Olivier Caelen. *What is the Shapley value?* 2022. URL: https://medium.com/the-modern-scientist/what-is-the-shapley-value-8ca624274d5a.

[Dua+20] Jingliang Duan et al. "Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data". In: *IET Intelligent Transport Systems* 14.5 (2020), pp. 297–305.

[Che+21] Xiaochang Chen et al. “Automatic overtaking on two-way roads with vehicle interactions based on proximal policy optimization”. In: *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2021, pp. 1057–1064.

[KB14] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[He+16] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[Ope25] OpenAI. *ChatGPT*. 2025. URL: https://chatgpt.com/.