Safety21
INNOVATING FOR SAFETY

US DOT National
University Transportation Center for Safety

Carnegie Mellon University

Integrating Occupancy Grids with Spatial-Temporal Reinforcement Learning for Enhanced Vehicle Control

Navigating Highly Dynamic and Complex Driving Scenarios

Keith Redmill (PI) (https://orcid.org/0000-0003-1332-1332)
Zhihao Zhang (https://orcid.org/0009-0009-7932-2128)
Ekim Yurtsever (https://orcid.org/0000-0002-3103-6052)

FINAL RESEARCH REPORT - August 18, 2025

Contract # 69A3552344811/69A3552348316

Technical Report Documentation Page

1. Report No. Pending assignment.	**2. Government Accession No** n/a	**3. Recipient's Catalog No.** n/a
4. Title and Subtitle Integrating Occupancy Grids with Spatial-Temporal Reinforcement Learning for Enhanced Vehicle Control Navigating Highly Dynamic and Complex Driving Scenarios		**5. Report Date** August 18, 2025 **6. Performing Organization Code** n/a
7. Author(s) Keith A. Redmill, Ph.D. https://orcid.org/0000-0003-1332-1332 Zhihao Zhang https://orcid.org/0009-0009-7932-2128 Ekim Yurtsever, Ph.D. https://orcid.org/0000-0002-3103-6052		**8. Performing Organization Report No.** n/a
9. Performing Organization Name and Address The Ohio State University Center for Automotive Research 930 Kinnear Road Columbus, OH 43212		**10. Work Unit No. (TRAIS)** n/a **11. Contract or Grant No.** 69A3552344811/69A3552348316
12. Sponsoring Agency Name and Address Safety21 University Transportation Center Carnegie Mellon University 5000 Forbes Avenue Pittsburgh, PA 15213		**13. Type of Report and Period Covered** Final Report. July 1, 2024 to June 30, 2025 **14. Sponsoring Agency Code** USDOT

15. Supplementary Notes

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated in the interest of information exchange. The report is funded, partially or entirely, by a grant from the U.S. Department of Transportation's University Transportation Centers Program. The U.S. Government assumes no liability for the contents or use thereof.

16. Abstract

Autonomous driving in complex traffic environments demands decision-making systems that can interpret diverse sensory inputs, anticipate multi-agent interactions, and execute safe and efficient maneuvers. Roundabouts present a particularly challenging case due to continuous merging, yielding, and exiting maneuvers under heterogeneous and often unpredictable traffic flows. This study addresses these challenges by designing and evaluating a decision-making framework capable of handling high-complexity roundabout scenarios.

We analyze the intrinsic complexity of roundabout navigation by quantifying interaction density, conflict points, and decision latency under varying traffic densities, providing a structured benchmark for evaluating policy robustness. To model the environment, we adopt multi-layer occupancy grids as the primary spatial representation, providing a dense encoding of occupancy, velocities, and road geometry. The backbone combines a CNN-based spatial encoder to capture local spatial patterns with a transformer module for temporal abstraction, enabling the model to reason over both spatial and temporal dependencies in traffic flow.

Building on this foundation, we propose the Uncertainty Weighted Decision Transformer (UWDT) to improve decision making, safety, and efficiency in rare, high-risk situations. Unlike standard Decision Transformers, which optimize a uniform sequence prediction loss, UWDT incorporates an uncertainty-weighted objective that increases the learning signal for states with higher policy or value uncertainty. Experimental simulation results demonstrate that UWDT, combined with the spatial encoder, achieves lower collision rates and shorter traversal times compared to baseline Decision Transformers and Behavior Cloning (BC) Transformers and conventional deep reinforcement learning agents such as Soft Actor Critic (SAC) and Conservative Q-Learning (CQL). In particular, UWDT exhibits greater resilience in rare but safety-critical states, effectively balancing assertiveness with caution.

17. Key Words Occupancy Grids, Decision Transformer, Complex Driving Scenarios, Roundabouts		**18. Distribution Statement** No restrictions.	
19. Security Classif.(of this report) Unclassified	**20. Security Classif.(of this page)** Unclassified	**21. No Pages** 59 pages	**22. Price** n/a

Form DOT F 1700.7 (8-72) **Reproduction of completed page authorized**

Contents

List of Figures

List of Tables

Abstract

Autonomous driving in complex traffic environments demands decision-making systems that can interpret diverse sensory inputs, anticipate multi-agent interactions, and execute safe and efficient maneuvers. Roundabouts present a particularly challenging case due to continuous merging, yielding, and exiting maneuvers under heterogeneous and often unpredictable traffic flows. This study addresses these challenges by designing and evaluating a decision-making framework capable of handling high-complexity roundabout scenarios.

We analyze the intrinsic complexity of roundabout navigation by quantifying interaction density, conflict points, and decision latency under varying traffic densities, providing a structured benchmark for evaluating policy robustness. To model the environment, we adopt multi-layer occupancy grids as the primary spatial representation, providing a dense encoding of occupancy, velocities, and road geometry. The backbone combines a CNN-based spatial encoder to capture local spatial patterns with a transformer module for temporal abstraction, enabling the model to reason over both spatial and temporal dependencies in traffic flow.

Building on this foundation, we propose the Uncertainty Weighted Decision Transformer (UWDT) to improve decision making, safety, and efficiency in rare, high-risk situations. Unlike standard Decision Transformers, which optimize a uniform sequence prediction loss, UWDT incorporates an uncertainty-weighted objective that increases the learning signal for states with higher policy or value uncertainty. Experimental simulation results demonstrate that UWDT, combined with the spatial encoder, achieves lower collision rates and shorter traversal times compared to baseline Decision Transformers and Behavior Cloning (BC) Transformers and conventional deep reinforcement learning agents such as Soft Actor Critic (SAC) and Conservative Q-Learning (CQL). In particular, UWDT exhibits greater resilience in rare but safety-critical states, effectively balancing assertiveness with caution.

The results presented in this report have also been submitted for potential publication in a preferred academic conference.

Chapter 1

Introduction

This study addresses the challenge of enabling autonomous agents to interpret and navigate highly dynamic and complex driving environments using a structured occupancy motion grid as the primary form of state representation. This multi-channel grid encodes semantic and kinematic information, including vehicle presence, road layout, and local velocity components. Traditional reinforcement learning methods often struggle to capture long-term temporal dependencies and exploit such structured state inputs effectively. The purpose of this research is to bridge this gap by introducing a novel transformer-based architecture that jointly models spatial and temporal information, thereby enhancing decision-making performance in driving scenarios.

The proposed method integrates reinforcement learning with transformer-based architectures to process occupancy grid inputs more effectively. Convolutional neural networks (CNNs) are first employed to extract the environment structure such as road boundaries and traffic information. To enhance temporal reasoning, we introduce a decision transformer that models sequential dependencies across time steps. In contrast to conventional CNN-based reinforcement learning (CNN-RL) that only consider immediate transitions, our model incorporates a longer sequence of past observations and actions. This enables the system to make decisions based not only on the current state but also on recent behavioral trends. To further reduce collisions and enhance safety, we propose an Uncertainty Weighted Decision Transformer (UWDT) that explicitly down-weights actions with high epistemic uncertainty. The proposed framework is trained and evaluated in a set of simulated driving environments, with comparisons made against traditional CNN-RL and CNN-Behavior cloning (CNN-BC) baselines to assess the improvements in decision quality.

The key findings of this study are:

- The novel UWDT model we propose shows notable improvements in decision accuracy when compared to standard CNN-based RL, CNN-based BC controllers and DT.
- The inclusion of temporal modeling allows the agent to maintain smoother and more stable behavior over time by considering the sequence of recent observations.
- The proposed model achieves higher reward and lower collision rates across a variety of driving scenarios.

By pairing CNN-based spatial feature extraction with Transformer-based temporal modeling in an offline reinforcement learning setting, this study demonstrates a performance advantage for autonomous driving decision making in complex, dynamic environments. Future work will focus on enhancing the generalization ability of the controller by developing adaptive vehicle control strategies applicable to a broader range of driving scenarios.

1.1 Automated Vehicle Planning and Control

The emergence of autonomous vehicles has introduced a transformative paradigm in the broader landscape of automated driving. Autonomous driving refers to the decision-making and vehicle control processes that enable self-driving vehicles to perceive, navigate, and interact with diverse traffic environments ranging from dense urban streets and intersections to roundabouts and freeways. This discipline unifies advanced artificial intelligence algorithms, robust control methods, multi-modal sensing, sensor fusion, and real-time data processing as illustrated in Figure 1.1. Together, these technologies allow a vehicle to perceive its surroundings, predict the motion of other traffic participants, and plan safe and efficient trajectories. Typical decisions include obstacle avoidance, adaptive cruising, lane changes, merging, overtaking, and negotiation of complex multi-agent scenarios. Autonomous driving technologies promise significant improvements in road safety, reductions in congestion, and enhancements in overall transportation efficiency [3–5]. The ultimate goal is to ensure safe, efficient, and reliable autonomous travel on highways by enabling vehicles to analyze complex scenarios, anticipate potential hazards, and execute actions that align with traffic rules and user preferences [6]. Yet, despite remarkable strides, crafting autonomous vehicles with the capability to make discerning decisions remains a formidable obstacle. This complexity emanates from the amalgamation of diverse disciplines and the intricacies of real-world scenarios, necessitating profound innovation and interdisciplinary collaboration.

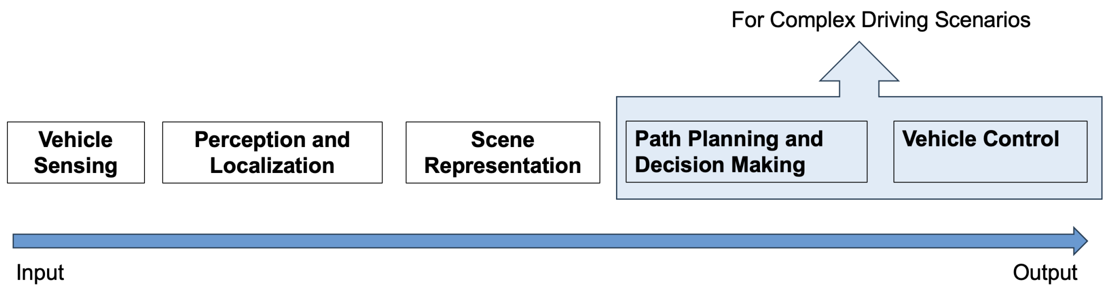

For Complex Driving Scenarios
Vehicle Sensing
Perception and Localization
Scene Representation
Path Planning and Decision Making
Vehicle Control
Input
Output

Figure 1.1: A pipeline of the automated vehicle control process.

In the rapidly evolving field of automated driving systems, several fundamental methodologies have emerged as essential strategies for addressing the complex challenge of decision-making. The first approach, characterized as conventional, embraces a modular framework that governs the decision-making process within driving scenarios [3]. Within this paradigm, distinct modules are meticulously crafted, each responsible for addressing specific scenarios and aspects of decision-making. These modules encompass specific behaviors such as lane changes, merging, overtaking, and other maneuvers crucial for navigation. Ultimately, the amalgamation of these modules, along with a behavior-planning and selection algorithm, culminates in a cohesive decision-making system that interfaces with vehicle-control mechanisms. While promoting transparency and interpretability, this approach demands meticulous calibration of diverse driving factors and a comprehensive understanding of a wide array of driving scenarios while accounting for safety considerations.

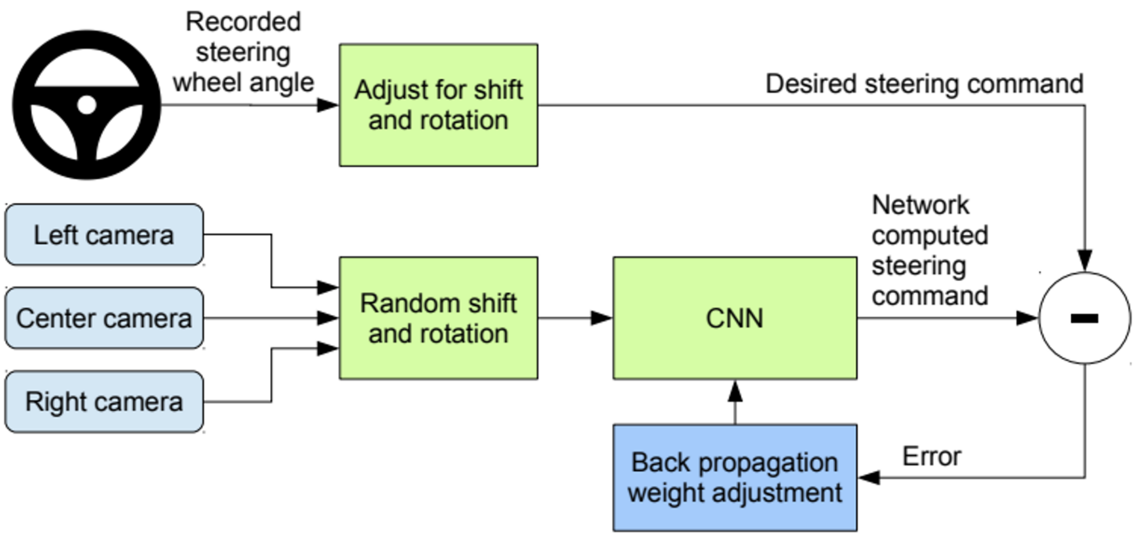

Recorded steering wheel angle
Adjust for shift and rotation
Desired steering command
Left camera
Center camera
Right camera
Random shift and rotation
CNN
Network computed steering command
-
Back propagation weight adjustment
Error

Figure 1.2: A pipeline of end-to-end Behavior Cloning for driving [1, 2].

A second approach, optimization-based planners such as Model Predictive Control and search based planners such as Monte-Carlo Tree Search (MCTS) estimate risk and reward by sampling complete roll-outs of candidate maneuver sequences in a generative traffic model [7–10]. Although these look-ahead optimization approaches deliver interpretable decisions, every simulation step incurs latency, so the overall cost scales sharply with increasing time horizons and, for MCTS, the branching factor that characterizes dense multi-lane traffic [11, 12]. State-of-the-art implementations therefore rely on high-fidelity simulators to generate roll-outs offline [13]. Meeting real-time requirements on board remains difficult, often forcing the use of shallow trees or aggressive pruning strategies that can adversely affect decision quality.

Conversely, behavior cloning methods, exemplified by supervised learning, take a different path towards decision-making refinement [2, 14]. As illustrated in Figure 1.2, this approach simplifies the process by using machine-learning methods to replicate expert driving behavior. Developing such a controller entails concurrently gathering and correlating sensor data with the respective steering, brake, and throttle actuator actions performed by human drivers. However, a notable caveat of this method is its demand for extensive and diverse data collection.

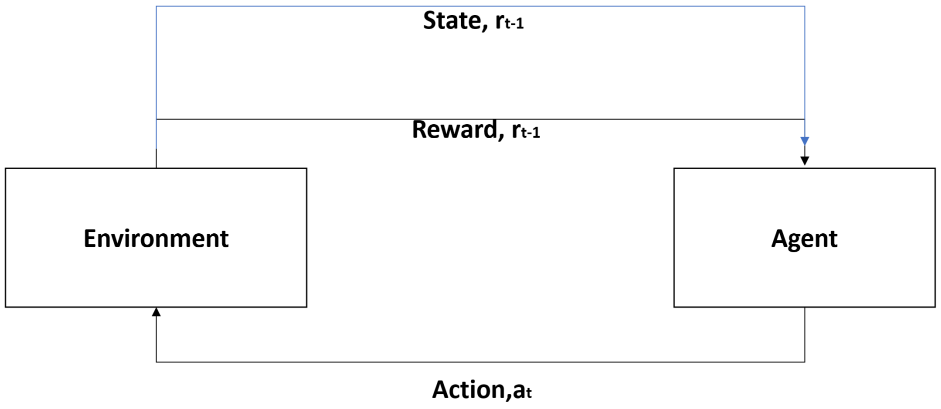


Figure 1.3: An RL agent interacts with the environment and learns from the environment.

Finally, as illustrated in Figure 1.3, RL emerges as a compelling paradigm, aiming to equip autonomous systems with decision-making capabilities through simulated interaction with the environment [4,15–18]. However, online deep-RL controllers often rely on extensive exploration in the early stages of training. Insufficient exploration can hinder the discovery of effective decision-making behaviors, especially for coping with complex traffic conditions, and may expose the system to unsafe states. Moreover, existing studies frequently gravitate towards simplified traffic scenarios, limiting the generalizability of their findings when confronting the unpredictable nature of real-world driving.

1.2 Decision Transformers

Decision Transformers (DT) combine sequence modeling with offline RL [19,20]. The policy is conditioned on a desired return and is trained only on logged trajectories. This design removes unsafe exploration required by online RL and reduces the covariate shift that limits pure behavior cloning. It also supports long-horizon planning. A drawback is that a standard DT can be brittle when a test state differs from the training data, a situation that occurs often in complex highway traffic.

Studies attempt to strengthen DT through auxiliary signals or architectural changes. Methods such as TD-steering and RL-gradient add temporal-difference or policy-gradient targets. These targets align the sequence loss with value consistent features [21]. Episodic importance sampling and retrieval augmented external memory re-weight updates to favor high-return transitions and shorten effective context length, thereby boosting online adaptation and inference efficiency [22]. Bootstrapped Transformer heads form an ensemble whose variance provides principled uncertainty estimates for conservative policy evaluation and safer offline learning [23]. Constrained DT variants impose adaptive cost penalties so that cumulative constraint violations remain within predefined limits without degrading return [24]. Gated Transformer-XL reorders layer norms and inserts gating units to stabilize optimization on deep or long-horizon tasks, yielding more reliable training dynamics [25]. Online DT blends offline pretraining with BC regularization during interaction, while future-conditioned unsupervised pretraining exploits reward-free logs for broader generalization [26, 27]. Across categories, these approaches add auxiliary losses, memory modules, ensemble heads, or constraint terms, yet none directly quantify per-sample reliability at decision time, leaving an open challenge for deployment in safety-critical domains such as autonomous driving.

Uncertainty weighting is a complementary strategy for handling imbalanced or out-of-distribution data. In image classification, predictive entropy from a teacher network guides student optimization, giving larger weights to ambiguous samples [28]. A similar entropy-based scheme improves object detection [29]. Offline RL applies this idea as well. The Uncertainty Weighted Actor-Critic (UWAC) down-weights risky transitions using Monte-Carlo-dropout uncertainty during critic updates [30]. The UNcertainty-awaRE deciSion Transformer (UNREST) identifies high mutual-information segments between returns and states, then retrains those segments with clipped returns to improve robustness in stochastic driving [31].

Building on these ideas, we use a frozen teacher Decision Transformer trained on bird's-eye-view occupancy grids to produce token-level entropies that reflect predictive confidence. During student training, we multiply each token's cross-entropy term by a weight that increases with the teacher's entropy. Confident tokens keep near-original influence, while uncertain tokens receive amplified gradients. This preserves the data efficiency of offline learning, adds explicit uncertainty awareness, and yields safer and more robust policies in complex scenarios such as dense roundabout traffic.

1.3 Occupancy Grids Representing Spatio-Temporal Information

An occupancy grid partitions the bird's-eye-view (BEV) plane surrounding the ego vehicle into a fixed lattice of cells. Each cell m_i stores the probability $p(m_i) \in [0, 1]$ that the corresponding spatial region is occupied. We maintain cell occupancy with a Bayesian filter. Each sensor observation triggers a recursive update. Range hits provide positive evidence and raise the occupancy probability. Free-space returns provide negative evidence and lower it. Updates are performed in log-odds space to prevent numerical saturation and to support consistent multi-frame fusion. This design remains robust under sensor noise and partial occlusion [32–34]. Beyond the binary presence channel, additional layers such as longitudinal velocity v_x, lateral velocity v_y, and an on_road mask can be stacked to form a multi-modal tensor that provide semantic, geometric, and kinematic information for downstream learning [35].

Other frameworks for representing the state of the environment include kinematic tables, raw point clouds, and graphs. Kinematic tables store the ego vehicle and the V closest neighbors in a $V \times F$ matrix where F is the dimension of the kinematic information for each traffic vehicle. They are compact on straight, static roads but rely on fragile data association and cannot represent free space and road information. Raw point clouds offer detail yet their sparsity and irregularity demand costly volatilization or graph networks. Vectorized lane graphs encode HD-map topology for long-range planning but exclude dynamic obstacles and need an additional perception stream [36].

Our four-channel occupancy grid aligns with CNNs and preserves translation equivariant mappings for efficient local reasoning [37]. Per cell probabilities enable principled multi-sensor and temporal fusion without hard thresholds [38]. We stack presence, per-cell velocities, and road semantics into a single tensor. Its size is fixed by the grid rather than the number of vehicles, so it scales to any traffic density. Sparsity in occupancy and velocity keeps memory use low and computation requirements predictable. The tensor is compact, which supports long horizons and large batch sizes. This makes it a practical spatio-temporal state representation for end-to-end autonomous driving.

Our experiments adopt a $4 \times 41 \times 50$ cell grid with 2 m resolution and the channel order {presence, v_x, v_y, on_road}. Figure 1.4 contrasts a conceptual two-lane scene with its presence layer. Cell values of 1 denote occupied space and values of 0 denote free space, illustrating how raw geometry becomes a dense raster amenable to convolution. The longitudinal velocity v_x and lateral velocity v_y are clipped to $[-v_{\max}, v_{\max}]$ and linearly mapped to $[0,1]$. This normalization preserves relative speed information while maintaining the binary occupancy cue. Higher values in the cells indicate faster motion in the corresponding direction. In this study, CNNs are first employed to extract the environmental structure which is then fed into the Decision Transformer, as described in Section 3.3.1.

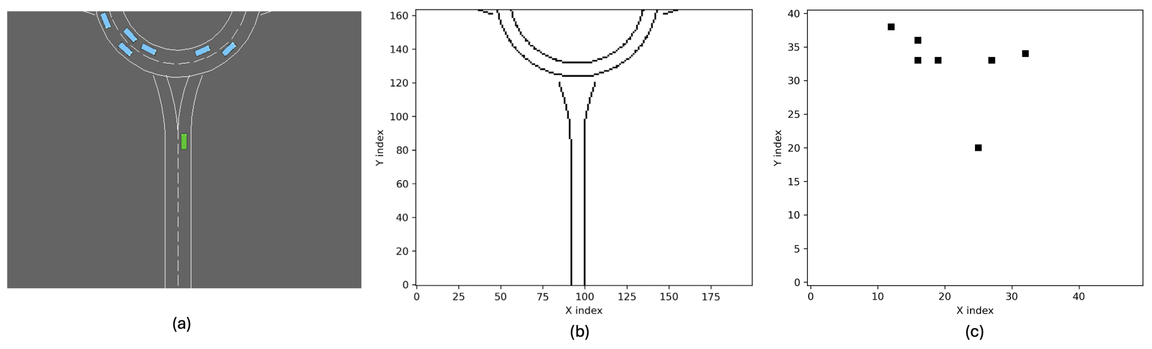


Figure 1.4: (a) Example Birds Eye View (BEV) of traffic scenario and corresponding (b) Road information layer and (c) Presence layer

.

1.4 The Roundabout Complex Traffic Scenario

For this study, we consider the two-lane roundabout configuration shown in Figure 1.5. The goal of the planner is to guide the ego vehicle entering the four arm, two lane roundabout from the south arm safely and efficiently to its fixed exit at the north arm, while negotiating the circulating, interacting, and exiting traffic vehicles. Randomness is introduced into the scenario by varying the number, initial configuration and lane assignments, initial speeds, and the final destination of each background traffic vehicle as well as perturbing the parameters of its Intelligent Driver Model (IDM) to introduce variation in acceleration and time-gap preferences. Further details are provided in Section 2.4 of this report. These stochastic elements ensure that every episode presents the ego vehicle with a unique configuration of circulating, interacting, and exiting traffic.

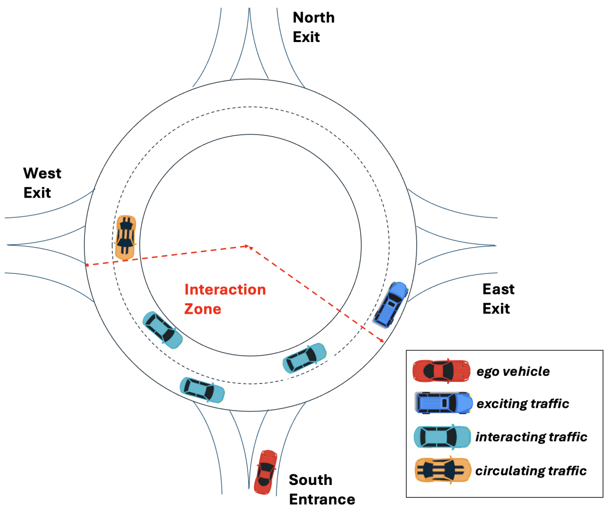


Figure 1.5: Four-arm, two-lane roundabout is used in our experiments. The ego vehicle enters from the south, circulating traffic is already inside the roundabout and travel behind interacting traffic, interacting traffic travels immediately ahead and behind the ego vehicle, and exiting traffic leaves from the east arm. The dashed red polygon denotes the interaction zone, representing the ideal spatial region where traffic vehicles may interfere with or constrain the ego vehicle's entry into the roundabout.

By increasing the number of interacting vehicles, we can systematically escalate planning difficulty by reducing the size and number of available gaps, increasing the uncertainty in gap acceptance, and forcing the planner to negotiate tighter merge windows or defer entry behaviors, all of which dramatically elevate roundabout entry complexity under uncertainty.

1.5 Quantitative Roundabout Scenario Complexity Evaluation

Complex driving scenarios involve multi-agent interactions, high state uncertainty, and long-horizon decision making. To quantify the planning difficulty of such scenarios, we propose using the decision time and simulation budget required by MCTS to successfully solve the planning problem. A higher MCTS budget and longer decision latency indicate increased planning difficulty, reflecting the inherent complexity of the traffic environment. MCTS explores a search tree whose size grows exponentially with the branching factor. The required rollouts and simulation budget escalate rapidly, making real-time deployment infeasible unless strong heuristics or prior guidance are introduced. Figure 1.6 and Figure 1.7 summarize our ablation study over planning budget and traffic density. As the number of incoming vehicles at the roundabout increases, MCTS needs both a larger simulation budget and additional decision time to maintain solution quality. Conversely, holding the budget fixed while increasing traffic density lengthens the decision time and lowers the average return. This design allows us to quantitatively attribute rising MCTS decision latency and predictive entropy to increasing interacting vehicle counts.

These trends confirm that MCTS scales poorly in dense traffic. Moreover, the number of incoming vehicles can offer a monotonic indicator of scenario difficulty and underscores the need for policy learning methods that yield safe, efficient, real-time decisions without exhaustive online search. Detailed information about MCTS will be introduced in Section 3.1.

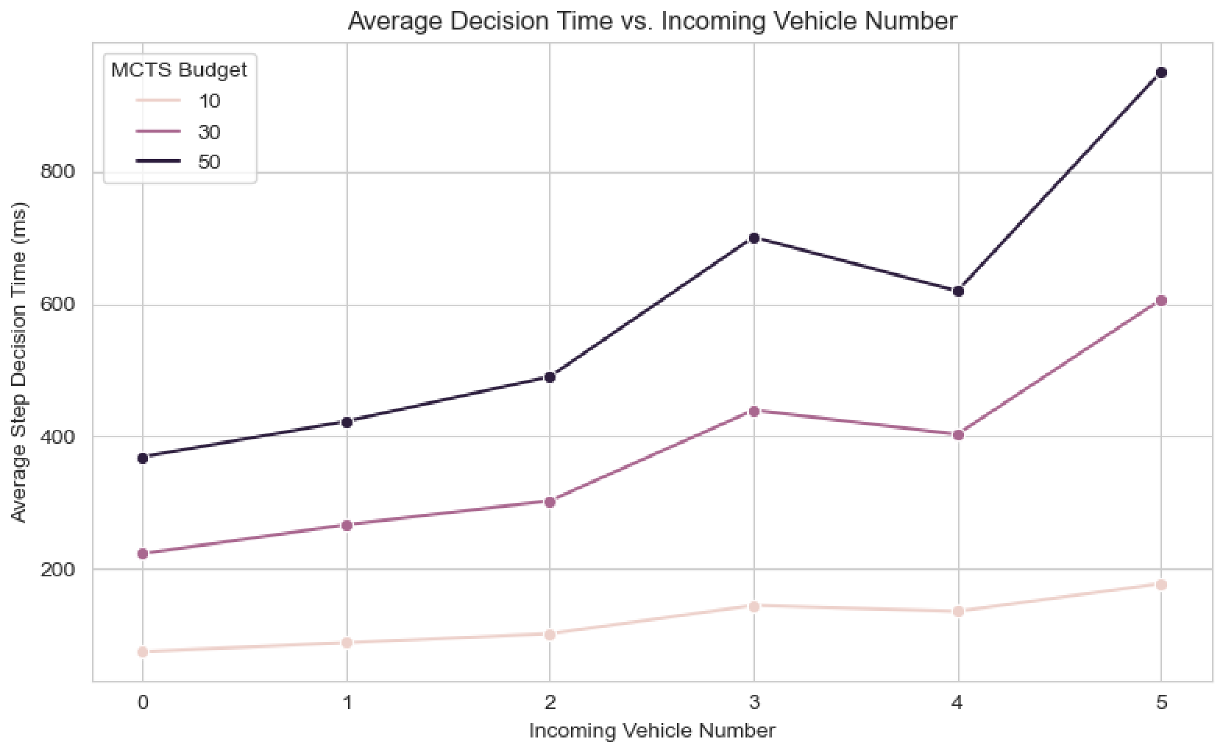


Figure 1.6: As the number of incoming vehicles increases, MCTS requires more decision time.

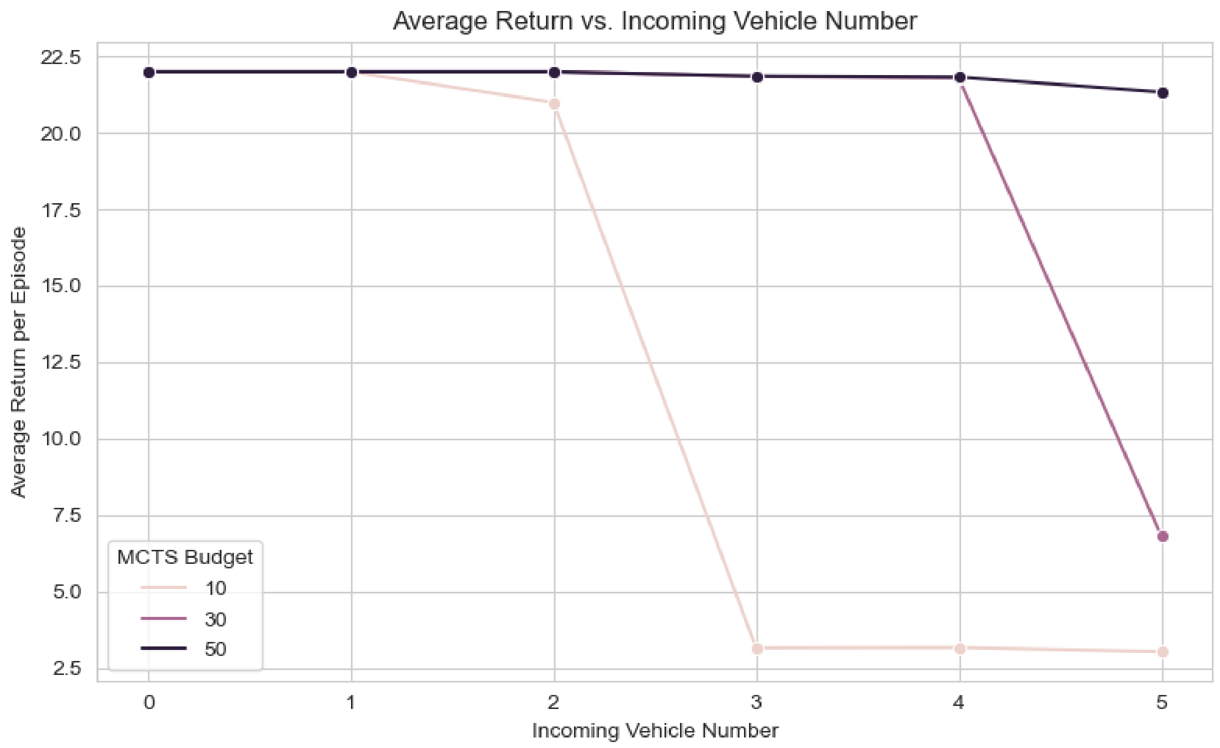


Figure 1.7: As the number of incoming vehicles increases, maintaining successful planning becomes increasingly difficult for MCTS under fixed and limited budgets.

Behavioral uncertainty offers another compact lens through which to evaluate scenario complexity. Information theoretic analyses reveal that the Shannon entropy of a traffic scene increases with the density and interaction richness of dynamic agents [39, 40]. In parallel, empirical studies link larger predictive entropies to shorter time-to-collision intervals and higher collision probabilities, thereby providing a practicable surrogate for safety risk. Modern uncertainty-aware controllers exploit this link. Systems from the UNREST Decision Transformer to MC-Dropout imitation-learning policies use predictive uncertainty to schedule curricula and to trigger cautious maneuvers when risk increases [31, 41]. For each observation o_t, our BC Transformer or DT outputs a prediction distribution $\pi(\cdot \mid o_t)$; the step-wise entropy is

$$H_t \;=\; -\sum_{a\in\mathcal{A}} \pi(a \mid o_t)\, \ln \pi(a \mid o_t), \tag{1.1}$$

and the episode-level score is $\bar{H} = \frac{1}{T}\sum_{t=1}^{T} H_t$. Higher $\bar{H}$ values signify greater behavioral ambiguity and thus more demanding planning problems. Table 1.1 and Figure 1.8 corroborate this claim. As the number of incoming vehicles rises, $\bar{H}$ grows monotonically for both BC Transformer and DT, mirroring the increase in MCTS latency. Because entropy can be computed in constant time per step and is independent of search branching factors, it supplies a lightweight yet informative complement to rollout-based complexity estimates.

Incoming Vehicles	BC Transformer	Decision Transformer
0–2 (random)	0.714 ± 0.072	1.150 ± 0.020
3	0.788 ± 0.019	1.156 ± 0.006
4	0.796 ± 0.020	1.163 ± 0.029

Table 1.1: Average predictive entropy (± standard deviation) for BC Transformer and DT under three traffic-density scenarios.

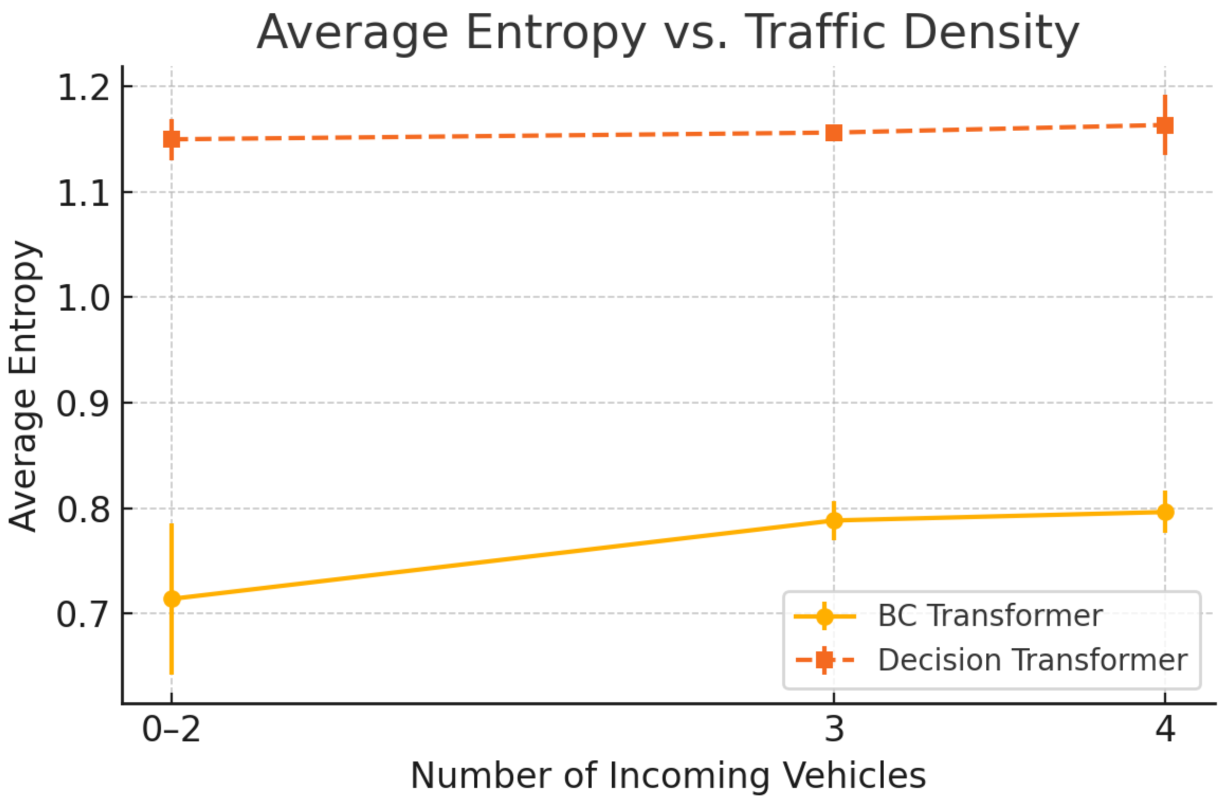


Figure 1.8: Average prediction entropy of BC Transformer and DT is lowest when at most two incoming vehicles are present and rises steadily for both agents when the number of interacting vehicles is increased to three and then four. The upward trend indicates that each model's uncertainty grows with congestion, suggesting that motion planning becomes progressively more difficult as the scene becomes denser and less predictable. Error bars denote per-episode standard deviation.

Chapter 2

Problem Formulation

In this chapter, we introduce the problem formulation and the baseline solution strategies and algorithms used throughout the study. A modern roundabout involves multilane merging, yielding, and exit decisions. The state space is represented as a $4 \times 41 \times 50$ occupancy grid that merges static geometry with dynamic traffic kinematics. The action space comprises high-level, discrete maneuvers that are executed by a separate low-level motion planner. The reward blends collision penalties, speed incentives, and lane-change costs, thereby aligning optimization with both safety and efficiency.

After outlining the general Markov Decision Process (MDP) formulation, we detail each component. Two families of baseline methods are positioned against this backbone. BC, implemented with a lightweight CNN–Transformer, learns a one-step mapping from grid observations to discrete actions but is vulnerable to covariate shift. Reinforcement Learning offers both online and offline variants: Soft Actor Critic (SAC) exploits active exploration, whereas Conservative Q-Learning (CQL) improves a value function directly from logged data. These contrasting baseline methods set the stage for later chapters, which will tackle the twin challenges of rare event imbalance and generalization beyond a narrow demonstration corpus.

2.1 State Space Representation

We use the default observation provided by the *highway-env* [42] simulator with a $4 \times 41 \times 50$ occupancy grid [33,43–46]. This grid consists of four feature channels: *vehicle presence*, longitudinal velocity v_x, lateral velocity v_y, and a binary *on_road* indicator. The grid is centered on the ego vehicle and spans a physical area of $100\,\text{m} \times 82\,\text{m}$, discretized at $2\,\text{m}$ resolution. All velocity features are clipped to predefined ranges and linearly scaled to the interval $[-1, 1]$ to ensure numerical stability and compatibility with the CNN encoder used by our training policy. This representation captures spatial and dynamic context in a fixed-size tensor format, enabling efficient processing of surrounding traffic interactions through convolutional layers.

2.2 Action Space

In this framework, at each timestep the high-level controller processes the state information S and selects a high-level action A_{high}. The available high-level actions for the high-level controller are categorized into lateral and longitudinal decision-making processes. Lateral decisions include behaviors such as left lane change, maintaining the current lane, and right lane change. Longitudinal decisions involve adjustments to the vehicle's speed, either by increasing or decreasing it by a specified increment δ m/s. Lateral decisions do not interfere with longitudinal decisions. The selected high-level action is then interpreted as instructions for setting the target goal for the low-level controller, represented as $(l_{\text{target}}, V_{\text{target}})$. Here, l_{target} specifies a target lane index, and V_{target} designates a target speed.

After selecting a high-level action, we employ a low-level motion planner to translate it into corresponding acceleration and steering commands represented as $(a_{acc}, \theta_{lateral})$. This low-level planner is identical to the one described in Section 2.4.3. The action spaces for acceleration a_{acc} and steering angle $\theta_{lateral}$ are defined as follows:

$$a_{acc} \in [-1,\ 1]\ \text{m/s}^2 \tag{2.1}$$

$$\theta_{lateral} \in \left[-\frac{\pi}{36},\ \frac{\pi}{36}\right]\ \text{rad} \tag{2.2}$$

2.3 Reward Function

Following the reward structures adopted in recent tactical-driving studies for `highway-env` [47, 48], the objective of the roundabout driving task encourages fast and safe driving while discouraging unnecessary lane changes.

Three scalar reward components are derived as follows:

- The collision indicator c_t equals 1 if the ego-vehicle has crashed and 0 otherwise.
- The speed reward indicator v_t equals 1 if the ego-vehicle's speed is within the range $[8, 16]$ m/s and 0 otherwise.
- The lane-change indicator ℓ_t equals 1 when the chosen action commands a lane change, and 0 otherwise.

The raw reward at each time step combines these components linearly with fixed weights from the default configuration:

$$r_t^{\text{raw}} = w_c c_t + w_v v_t + w_\ell \ell_t, \quad \text{with} \quad [w_c = -1,\ w_v = 0.2,\ w_\ell = -0.05], \tag{2.3}$$

where the collision weight w_c imposes a severe penalty upon collisions, the speed weight w_v provides a modest reward for maintaining high velocities (favoring efficient traffic flow without reckless acceleration), and the lane-change weight w_ℓ penalizes unnecessary lateral movements to encourage stable driving patterns. The reward is linearly scaled to $[0, 1]$ as follows:

$$\tilde{r}_t = \frac{r_t^{\text{raw}} - (w_c + w_\ell)}{w_v - (w_c + w_\ell)}, \tag{2.4}$$

such that $\tilde{r}_t = 0$ at collision and $\tilde{r}_t = 1$ when driving at the maximum speed without lane changes. This ensures a consistent reward range for training across episodes.

2.4 Experimental Configuration

2.4.1 Roundabout Traffic Simulation

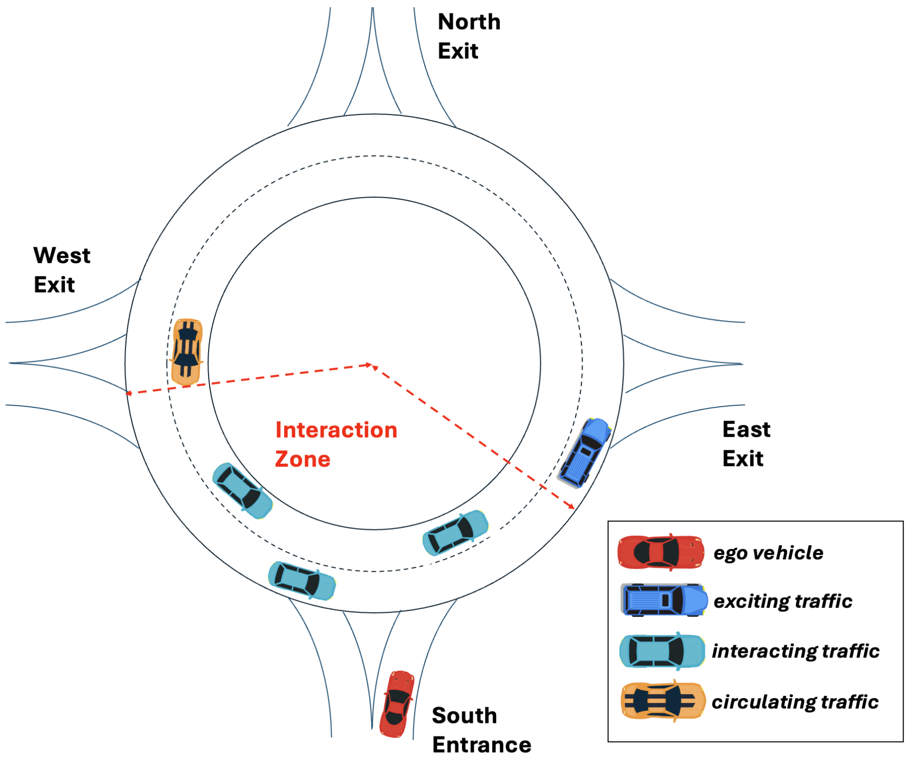


Figure 2.1: Four-arm, two-lane roundabout used in our experiments. The ego vehicle enters from the south, circulating traffic is already inside the roundabout and travels behind interacting traffic, interacting traffic travels immediately ahead and behind the ego vehicle, and exiting traffic leaves from the east arm. The dashed red polygon denotes the interaction zone, representing the ideal spatial region where traffic vehicles may interfere with or constrain the ego vehicle's entry into the roundabout.

The environment is implemented with the `roundabout-v0` task in the *highway-env* [42] simulator. It models a single roundabout whose inner and outer circulating lanes have radii of 20 m and 24 m, respectively. Simulation dynamics are integrated at 15 Hz while the agent selects an action at 2 Hz. Each episode lasts 11 s (22 time-steps) or terminates upon collision. In our experimental setup, the planner's mission is to guide the ego vehicle entering the four arm, two lane roundabout from the south arm safely and efficiently to its fixed exit at the north arm, while negotiating with circulating, interacting, and exiting traffic.

Initial states and randomness. For every background vehicle, the initial speed

$$v \sim \mathcal{N}(16,\, 0.1^2)\; m/s$$

is sampled independently and a Gaussian perturbation ($\sigma = 1\,\text{m}$) is added to each longitudinal position. Each vehicle perturbs the parameters of its Intelligent Driver Model (IDM) to introduce variation in acceleration and time-gap preferences. These stochastic elements ensure that every episode presents the ego vehicle with a unique configuration of circulating, interacting, and exiting traffic.

Vehicle categories. Traffic is divided into four functional groups that correspond to Figure 2.1:

- **Ego vehicle** (red). Spawned 125 m before the south entrance on the outer lane with an initial speed of 8 m/s. Its fixed route exits at the north arm.
- **Circulating traffic** (orange). A random number of vehicles

  $$N_{\text{circ}} \sim \mathcal{U}\{0, 1, 2\}$$

  are initialized on the western approach and enter the roundabout, occupying either the inner or outer circulating lane. Each vehicle selects a random destination among the north, east, or inner-west exits, simulating diverse circulation behaviors.
- **Interacting traffic** (teal). A random number of exiting vehicles

  $$N_{\text{interact}} \sim \mathcal{U}\{0, 1, 2, 3, 4\}$$

  are inserted at varying positions along the circulating lanes. These vehicles are likely to interfere with the ego vehicle's entry decision, as they may block or yield at the merging point. Their initial longitudinal offsets and lane assignments are randomized to reflect realistic variability in traffic flow.
- **Exiting traffic** (blue). Two vehicles are placed on the eastern approach, each at a distance of 50 m upstream from the roundabout. They proceed directly toward the eastern exit, simulating through-traffic that does not engage with the ego vehicle.

By increasing the number of interacting vehicles N_{interact}, we systematically escalate planning difficulty: higher N_{interact} reduces available gaps, raises uncertainty in gap acceptance, and forces the planner to negotiate tighter merge windows or defer entry behaviors that are known to dramatically elevate roundabout entry complexity under uncertainty. This design allows us to quantitatively attribute rising MCTS decision latency and predictive entropy to increasing interacting vehicle counts.

2.4.2 Traffic Vehicle Control

To control the traffic vehicles, we implement the Intelligent-Driver Model (IDM) [5] for longitudinal control and the Minimizing Overall Braking Induced by Lane Change (MOBIL) model [49] for lateral decision-making. In our roundabout simulation, vehicles circulating within the roundabout are allowed to perform lane changes, as the roundabout contains two lanes. In contrast, vehicles within or approaching the arms are restricted to longitudinal control. They can only adjust their acceleration, since each entry arm consists of a single lane in each direction.

IDM [5] is a time-continuous vehicle-following model used to simulate the behavior of individual vehicles in a traffic flow. It is a car-following model that considers various factors such as the distance between vehicles, their relative velocity, and the desired speed of the vehicle in order to ensure that vehicles maintain a safe following distance, avoid collisions, and adjust their speed according to the traffic flow. In the IDM model, each vehicle aims to keep a safe distance from the preceding vehicle while maintaining its desired speed. The desired speed is a function of the driver's preference and the posted speed limit. The model also includes the driver's reaction time and the vehicle's acceleration and deceleration capabilities. The IDM model uses these factors to control the acceleration and deceleration of a vehicle and maintain a safe distance from other vehicles. This represents an autonomous vehicle model that is collision-free and has self-adaptive capabilities on highways.

The IDM model [5] calculates the desired acceleration of the vehicle using:

$$acc = a\left[1 - \left(\frac{v}{v_{desired}}\right)^{\delta} - \left(\frac{s_{desired}}{s}\right)^{2}\right] \tag{2.5}$$

where s is the gap to the front vehicle, $s_{desired}$ is the desired distance to the preceding vehicle, v is the vehicle's current velocity, a is a parameter for acceleration, and $v_{desired}$ is the target velocity for the controlled vehicle. The desired distance to the preceding vehicle is calculated using

$$s_{desired} = S_0 + Tv + \frac{v\Delta v}{2\sqrt{ab}} \tag{2.6}$$

where S_0 is the desired distance gap, T the desired time gap to the preceding vehicle, and a and b are acceleration and deceleration parameters. The desired distance is calculated based on the vehicle's speed and the relative speed between the current vehicle and the preceding vehicle.

MOBIL [49] is a model that governs the lane-changing behavior of autonomous vehicles. MOBIL aims to minimize the total amount of braking induced by lane changes while also maximizing the overall traffic flow. It considers factors such as the relative speed and position of surrounding vehicles, as well as the ego-vehicle's acceleration and deceleration capabilities, to determine whether changing lanes would be beneficial.

The MOBIL model incorporates several decision rules to determine the benefit of a lane change. First, the MOBIL model checks if there is a gap in the target lane that the vehicle can occupy without causing any conflict with other vehicles. If there is a gap, the model checks if the vehicle can accelerate to its desired speed before reaching the gap. If the vehicle can safely accelerate to the desired speed, it performs the lane change. If there is no gap in the target lane, MOBIL checks if the lane change would allow the vehicle to travel at a higher speed than the current lane. If so, the vehicle can perform the lane change if it can safely accelerate to the desired speed. If neither of these conditions is met, the vehicle will remain in its current lane. By using MOBIL, autonomous vehicles can make safe and efficient lane-changing decisions. The MOBIL model decides when to perform a lane change based on the impact on other drivers, as defined by the following condition

$$(a'_e - a_e) + p[(a'_b - a_b) + (a'_a - a_a)] > a_{\text{th}} \tag{2.7}$$

where $(a'_e - a_e)$, $(a'_b - a_b)$, $(a'_a - a_a)$ represent the acceleration difference of the driver's vehicle, the following vehicle before the lane change and the following vehicle after the lane change, respectively, p represents the politeness factor which weighs the disadvantages imposed on other drivers due to the lane change, and a_{th} is the acceleration threshold. The control parameters for the traffic vehicles are detailed in Table 2.1.

Symbol	Meaning	Value
l	Vehicle length	5 m
w	Vehicle width	2 m
w_{road}	Lane width	4 m
a_t	Acceleration action range	$[-1,\ 1]\,\text{m/s}^2$
θ_t	Steering angle range	$[-\pi/36,\ \pi/36]\,\text{rad}$
p	Politeness factor (MOBIL)	0.5
a_{th}	Acceleration threshold	$0.2\,\text{m/s}^2$
a	Acceleration (IDM)	$0.5\,\text{m/s}^2$
b	Comfortable deceleration (IDM)	$0.5\,\text{m/s}^2$
δ	Acceleration exponent (IDM)	4
s_{desired}	Desired car-following distance	10 m
v_{desired}	Desired velocity	12.5 m/s
S_0	Minimum spacing (IDM)	10 m
T	Desired time gap	1.5 s

Table 2.1: IDM and MOBIL traffic vehicle parameters used in simulations.

2.4.3 Low-level Motion Planner

As we introduced in Section 2.2, after we obtain the high-level actions A_{high} and set $(l_{\text{target}}, V_{\text{target}})$, we implement a lateral low-level controller to execute lane changes and a longitudinal low-level controller to track the target speed as shown in Figure 2.2. The same lateral low-level controller is also used to execute lane changes for traffic vehicles using the MOBIL controller.

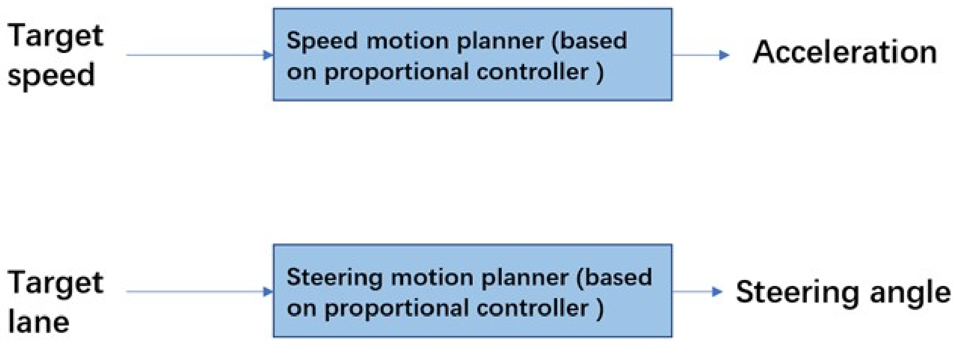

Target speed
Speed motion planner (based on proportional controller)
Acceleration
Target lane
Steering motion planner (based on proportional controller)
Steering angle

Figure 2.2: Speed and steering motion planner

The lateral low-level controller is based on the lateral distance to the target lane center

$$v_{lateral} = -K_{lat}\Delta d_{\text{\_target}} \tag{2.8}$$

where $\Delta d_{\text{\_target}}$ is the lateral distance to target lane center line. K_{lat} is the proportional gain for this speed planner. In the lateral direction, the heading controller deals with the heading of the vehicle with a similar proportional-derivative action

$$\theta_{lateral} = K_{\theta}(\theta_{target} - \theta) \tag{2.9}$$

where θ is the vehicle's current heading angle. θ_{target} is the vehicle's heading angle after lane change behavior. The heading angle after lane change will remain aligned with the direction of the target lane. K_θ is a proportional gain for this heading controller. This lane change motion planner will generate the trajectory of a vehicle from an initial lane to a target lane. Once the lane change decision is made, the motion planner will provide a steering angle command for each time step until the ego-vehicle reaches the target lane center line.

In the longitudinal direction, we interpret the act of accelerating or decelerating at a high-level as setting a target speed. Speed up means to increase the target speed by δ m/s (a numeral factor) and slow down means to decrease the target speed by δ m/s. δ is a parameter than we can select based on the ideal speed of the specific autonomous driving problem. Since we are studying a low-speed lane change problem with an ideal speed of 16m/s, we set δ to 2 m/s. After we set the target speed, a speed motion planner will be implemented to control the acceleration and track the target speed using

$$a_{acc} = K_{speed}\Delta v_{\text{\_target}} \tag{2.10}$$

where K_{speed} is the proportional gain for speed motion planner.

2.5 Reinforcement Learning Foundations

In our methodology, we formulate the task of automated driving on highways as a MDP problem [50]. An MDP can be described by a tuple $\langle S, A, P_{ss'}, R_s\rangle$ where the states S encapsulate the vehicle's environment and its internal status, while the actions A reflect the vehicle's potential maneuvers. The transition between state-actions is denoted as $P_{ss'} = P\left[S_{t+1} = s' | S_t = s\right]$. The rewards $R_s = E\left[R_{t+1} \mid S_t = s\right]$ are designed to guide the vehicle towards an optimal driving behavior. A deep-RL based controller aims to optimize the cumulative reward within a single episode, which is derived from the external reward function.

We adopt two complementary deep-RL algorithms to learn roundabout-driving policies: Soft Actor–Critic (SAC) for online interaction and Conservative Q-Learning (CQL) for offline learning [51, 52]. Below we summarize their objectives and optimization mechanisms.

2.5.1 Online Reinforcement Learning: SAC

SAC optimizes a stochastic actor policy π_ϕ by maximizing the discounted return augmented with an entropy bonus:

$$J_{\text{SAC}}(\pi_\phi) = \mathbb{E}_{\pi_\phi}\Big[\textstyle\sum_{t=0}^{\infty} \gamma^t \big(R(s_t, a_t) + \alpha\, \mathcal{H}\big[\pi_\phi(\cdot \mid s_t)\big]\big)\Big], \tag{2.11}$$

where the temperature $\alpha > 0$ trades off task reward against the differential entropy $\mathcal{H}\big[\pi_\phi(\cdot \mid s_t)\big]$ of the actor [51]. Entropy regularization encourages broad exploration in early stages and guards against premature convergence to deterministic, sub-optimal strategies. Two action-value critic functions Q_{θ_1} and Q_{θ_2} are trained with the soft Bellman backup operator, taking their minimum when the positive bias caused by overestimating the action values is mitigated [53].

Rather than hand-tuning the temperature parameter α, SAC adjusts it online by solving the dual problem

$$\alpha^\star = \arg\min_{\alpha > 0}\ \mathbb{E}_{(s_t,a_t)\sim\mathcal{B}}\big[\alpha\big(-\log \pi_\phi(a_t \mid s_t) - \bar{\mathcal{H}}\big)\big], \tag{2.12}$$

where $\mathcal{B}$ is the replay buffer and $\bar{\mathcal{H}}$ is a negative target entropy usually set to $-\log|A|$ or a scaled variant. This adaptive mechanism is especially useful in discrete action spaces whose utilities can vary sharply across states, as is the case in roundabout navigation.

Because SAC is off-policy, each transition stored in the replay buffer can be reused for many gradient updates. The algorithm therefore achieves high sample efficiency, an essential property when interacting with a computationally intensive, high-fidelity simulator that enables safe, continuous data collection without real-world risk. The pseudocode and training hyperparameters are presented in Algorithm 1 and Table 2.2, respectively.

Symbol	Meaning	Value
γ	Discount factor	0.99
η_π	Actor learning rate	1×10^{-5}
η_Q	Critic learning rate	5×10^{-5}
η_α	Entropy-coefficient learning rate	5×10^{-5}
τ	Target-network soft update	0.005
$\bar{\mathcal{H}}$	Target entropy	-1
N	Batch size	64
$\lvert D \rvert$	Replay buffer capacity	50000

Table 2.2: SAC hyper-parameters used in this study.

Algorithm 1 Soft Actor–Critic for Discrete Action Spaces

```
Require: Environment $\mathcal{E}$, replay buffer $\mathcal{D}$
Require: Learning rates $\eta_Q, \eta_\pi, \eta_\alpha$, discount $\gamma$, target-mix $\tau$
 1: Initialise actor $\pi_\phi$, critics $Q_{\theta_1}, Q_{\theta_2}$
 2: Initialise target critics $\bar{\theta}_i \leftarrow \theta_i$ $(i = 1, 2)$
 3: while training not converged do
 4:     Observe state $s$ and sample $a \sim \pi_\phi(\cdot \mid s)$
 5:     Execute $a$ in $\mathcal{E}$, receive $(r, s', d)$
 6:     Store $(s, a, r, s', d)$ in $\mathcal{D}$
 7:     for all gradient steps do
 8:         Sample mini-batch $\mathcal{B} \subset \mathcal{D}$
 9:         $V_{\bar{\theta}}(s') \leftarrow \sum_{a'} \pi_\phi(a'|s') \big[\min_i Q_{\bar{\theta}_i}(s', a') - \alpha \log \pi_\phi(a'|s')\big]$
10:         $y \leftarrow r + \gamma(1 - d)\, V_{\bar{\theta}}(s')$
11:         $\theta_i \leftarrow \theta_i - \eta_Q \nabla_{\theta_i} \frac{1}{2}\big(Q_{\theta_i}(s, a) - y\big)^2 \quad (i = 1, 2)$
12:         $\phi \leftarrow \phi - \eta_\pi \nabla_\phi \sum_a \pi_\phi(a|s) \big[\alpha \log \pi_\phi(a|s) - \min_i Q_{\theta_i}(s, a)\big]$
13:         $\alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha \sum_a \pi_\phi(a|s) \big[-\alpha(\log \pi_\phi(a|s) + \bar{\mathcal{H}})\big]$
14:         $\bar{\theta}_i \leftarrow \tau\theta_i + (1 - \tau)\bar{\theta}_i \quad (i = 1, 2)$
15:     end for
16: end while
```

2.5.2 Offline Reinforcement Learning: CQL

Offline RL forbids further interaction with the environment and must learn solely from a fixed log $\mathcal{D} = \{(s, a, r, s')\}$ of observation traces collected under unknown behavior policies. Standard Q-learning tends to over-estimate unseen state and action pairs, causing extrapolation error and catastrophic policy degradation when deployed. CQL [52] combats this failure mode by penalizing Q-values that are large for actions not observed in $\mathcal{D}$, thereby inducing a pessimistic critic that stays within the support of the dataset.

Let Q_θ denote the parametric critic and let $\mathcal{L}_{\text{TD}}$ be the usual temporal-difference loss with a target network $\bar{\theta}$. CQL augments this loss with a conservative regularizer that lowers any Q-value which the dataset does not support:

$$\mathcal{L}_{\text{CQL}}(\theta) = \mathbb{E}_{s\sim\mathcal{D}}\Big[\log\sum_{a\in A}\exp Q_\theta(s,a) - \mathbb{E}_{a\sim\mathcal{D}}\big[Q_\theta(s,a)\big] + \lambda\,\mathcal{L}_{\text{TD}}(\theta)\Big], \tag{2.13}$$

where the log sum expectation term upper-bounds the maximum Q over all actions and the second term anchors the estimate to actions actually present in $\mathcal{D}$. The trade-off coefficient $\lambda > 0$ controls conservatism where larger values emphasize Bellman consistency and smaller values enforce stronger pessimism.

Once Q_θ is updated, a stochastic actor π_ϕ is improved by minimizing the Kullback–Leibler divergence between $\pi_\phi(\cdot \mid s)$ and the Boltzmann distribution induced by the conservative Q-values:

$$\min_\phi \mathbb{E}_{s\sim\mathcal{D}}\Big[\text{KL}\big(\pi_\phi(\cdot \mid s) \,\|\, \text{softmax}\big(Q_\theta(s,\cdot)/\tau\big)\big)\Big], \tag{2.14}$$

where τ is a temperature hyper-parameter. The resulting actor is explicitly dataset-compliant because its behavioral support is shaped by the conservative critic.

CQL is well-suited to roundabout navigation logs that contain diverse yet safety-critical maneuvers. By constraining value estimates to lie within the demonstrated action support, CQL avoids optimistic extrapolation for high-risk actions while still allowing policy improvement on frequent controls. Consequently, the learned policy inherits the feasibility and safety characteristics of the dataset, a key requirement when offline logs originate from human drivers or previously validated controllers. The pseudocode and training hyperparameters are presented in Algorithm 2 and Table 2.3, respectively.

Symbol	Meaning	Value
γ	Discount factor	0.99
η_π	Actor learning rate	1×10^{-5}
η_Q	Critic learning rate	5×10^{-5}
η_α	Entropy-coefficient learning rate	5×10^{-5}
τ	Target-network soft update	0.005
$\bar{\mathcal{H}}$	Target entropy	-1
B	Batch size	64
$\|D\|$	Replay buffer capacity	200,000
T	Training episodes	2000

Table 2.3: CQL hyper-parameters used in this study.

Algorithm 2 Discrete CQL built on SAC

```
Require: Environment $\mathcal{E}$, replay buffer $\mathcal{D}$
Require: Actor $\pi_\phi$, critics $Q_{\theta_1}, Q_{\theta_2}$
Require: Hyperparams: $\gamma, \tau, \eta_Q, \eta_\pi, \eta_\alpha$
Require: CQL weight $\lambda$, target entropy $\bar{\mathcal{H}}$
 1: Initialise target critics $\bar{\theta}_i \leftarrow \theta_i$
 2: while training not converged do
 3:     Observe $s$;  sample $a \sim \pi_\phi(\cdot \mid s)$;  step env, store $(s, a, r, s', d)$ in $\mathcal{D}$
 4:     for all gradient steps do
 5:         Sample mini-batch $\mathcal{B} \subset \mathcal{D}$
 6:         $V_{\bar{\theta}}(s') \leftarrow \sum_{a'} \pi_\phi(a'|s') \left[\min_i Q_{\bar{\theta}_i}(s', a') - \alpha \log \pi_\phi(a'|s')\right]$
 7:         $y \leftarrow r + \gamma(1-d)\, V_{\bar{\theta}}(s')$
 8:         MSE $\leftarrow \frac{1}{2}(Q_{\theta_i}(s,a) - y)^2$ for $i$=1, 2
 9:         CQLTerm $\leftarrow \lambda \left[\log\sum_a \exp Q_{\theta_i}(s,a)_{\text{all actions}} - Q_{\theta_i}(s,a)\right]$
10:         $\theta_i \leftarrow \theta_i - \eta_Q \nabla_{\theta_i}[\text{MSE} + \text{CQLTerm}]$
11:         $\phi \leftarrow \phi - \eta_\pi \nabla_\phi \sum_a \pi_\phi(a|s) \left[\alpha \log \pi_\phi(a|s) - \min_i Q_{\theta_i}(s,a)\right]$
12:         $\alpha \leftarrow \alpha - \eta_\alpha \nabla_\alpha \sum_a \pi_\phi(a|s) \left[-\alpha(\log \pi_\phi(a|s) + \bar{\mathcal{H}})\right]$
13:         $\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$
14:     end for
15: end while
```

2.6 Transformer-Based Behavior Cloning

Behavior cloning can be reframed as an auto-regressive sequence model in which a transformer predicts each action token conditioned on the history of past states and actions. Because self-attention provides unbounded receptive fields, the model implicitly captures long-range dependencies that are crucial for roundabout tasks such as merge preparation.

Given a demonstration buffer $\mathcal{D} = \{(s_{0:T}, a_{0:T-1})\}$, the network is trained under the auto regressive negative log likelihood:

$$\min_\phi \; \mathbb{E}_{\tau\sim\mathcal{D}}\Big[\textstyle\sum_{t=0}^{T-1} -\log\, \pi_\theta\big(a_t \mid s_{0:t}, a_{0:t-1}\big)\Big], \tag{2.15}$$

which treats imitation purely as supervised learning—no value targets or return conditioning are required, distinguishing Transformer BC from DT that conditions on future returns. The pseudocode and training hyperparameters are presented in Algorithm 3 and Table 2.4, respectively.

Symbol	Meaning	Value
K	Sequence length (context window)	20
d	Token/embedding dimension	32
η	Learning rate	5×10^{-5}
B	Batch size (per device)	16
E	Training epochs	20
λ_{wd}	Weight decay	1×10^{-4}
ρ_{wu}	Linear warm-up ratio	0.1
β_1, β_2	AdamW moments	0.9, 0.999
$g_{\max}$	Gradient-norm clip	0.25
n_{layer}	Transformer layers	4
n_{head}	Attention heads	1

Table 2.4: CNN–Transformer BC hyper-parameters used in this study.

Algorithm 3 Transformer-Based Behavior Cloning

```
Require: Demonstration set $\mathcal{D}=\{\tau_i\}_{i=1}^{N}$, each $\tau = (s_1, a_1, \ldots, s_T, a_T)$
Require: Hyperparams: sequence length $K$, embedding dim $d$, lr $\eta$, batch size $B$
Require: Transformer $\pi_\theta$ with causal mask; loss $\mathcal{L}$ is cross-entropy
 1: for iteration = $1, \ldots$ do
 2:     Sample mini-batch $\{\tau_j\}_{j=1}^{B} \subset \mathcal{D}$
 3:     for all $\tau_j$ do
 4:         Slice random window $(s_{t-K+1:t}, a_{t-K+1:t-1})$
 5:         Encode tokens $x_t \leftarrow \text{Embed}(s_t) \parallel \text{Embed}(a_{t-1})$
 6:         $\hat{a}_t \leftarrow \pi_\theta(x_{t-K+1:t})$
 7:         $\mathcal{L} \leftarrow \mathcal{L} - \log P_\theta(a_t = \hat{a}_t | x_{1:t})$
 8:     end for
 9:     $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
10: end for
11: function SelectAction(history, $s_{\text{curr}}$)
12:     $x \leftarrow$ concatenate latest $K-1$ tokens & Embed($s_{\text{curr}}$)
13:     $p(\cdot) \leftarrow \pi_\theta(x)$;    return $\arg\max_a p(a)$
```

Chapter 3

Proposed Method

This chapter presents an offline learning framework that combines expert trajectory synthesis with uncertainty-aware sequence modelling. We begin by using Monte Carlo Tress Search (MCTS) with the Upper Confidence Bounds applied to Trees (UCT) rule to construct a data set of 5400 near-optimal roundabout roll-outs. Because the action space is discrete and the reward mirrors the evaluation metric, the budgeted search delivers trajectories that approximate the true optimum while preserving diversity across traffic seeds and avoiding the sub-optimal biases found in human demonstrations.

Building on this corpus, we introduce the novel Uncertainty Weighted Decision Transformer (UWDT). A frozen, high-capacity teacher provides token-level predictive entropy that acts as an automatic importance weight, increasing gradient focus on ambiguous yet safety-critical frames that are statistically rare in the expert data. The student shares the vanilla DT architecture but is optimized with this entropy-scaled loss, simultaneously addressing class imbalance and covariate shift without architectural change. Compared with value-based offline RL or straightforward behavior cloning, UWDT offers a principled route to risk-aware planning in a purely offline setting.

3.1 Data Collection

To train the Decision Transformer, Transformer-based BC, we first construct an expert dataset using MCTS. MCTS is selected for its ability to generate high-quality trajectories that optimize the task-specific reward function defined in the environment. Unlike purely heuristic rule-based methods, MCTS explicitly balances exploration and exploitation and provably converges toward the optimal open-loop policy under a finite simulation budget through the UCT rule [8]. This property ensures that the collected demonstrations remain close to the reinforcement-learning objective, providing consistent supervision across episodes.

We use Open-Loop Optimistic Planning (OLOP) [54] to split the total simulation budget B into M episodes of length L such that

$$B \;\geq\; M \times L. \tag{3.1}$$

Here B is the total number of calls to the generative model, M is the episode count, and L is the planning horizon in timesteps. This allocation gives each episode the same depth while still guaranteeing that deeper branches are explored when their optimistic value justifies the cost. For each episode, MCTS plans a sequence of $L = 22$ actions and returns the first action of the highest-visitation path. The full search tree is then replayed to generate the complete state-action-return sequence. Because all trajectories have equal length, they map directly to the fixed-width context window of sequence models without padding or masking. During tree expansion, nodes leading to penalties remain in the tree because UCT keeps an optimism term on unvisited or high-variance branches. The pseudocode is presented in Algorithm 4.

In practice we collect $N_{\text{epi}} = 5\,400$ expert episodes, resulting in 118800 state–action pairs. Episodes are shuffled and stored in an offline replay buffer for all downstream learners. We record both discounted return-to-go and per-step rewards so that sequence models can condition on either signal at training time. The roll-out policy π_{roll} used in line 11 of Algorithm 4 is an ε-greedy heuristic that selects the highest-value action with probability $1-\varepsilon$ and samples uniformly otherwise ($\varepsilon = 0.05$). Random seeds are fixed, and all generative-model calls are logged to enable exact regeneration of the dataset.

The MCTS policy in our dataset achieves an average return of 21.81 while remaining collision-free.

Algorithm 4 Monte–Carlo Tree Search with UCT and OLOP Budget Allocation

```
Require: generative model $G$, root state $s_0$
Require: discount $\gamma$, exploration constant $c$, total budget $B$
 1: $(M, L) \leftarrow$ OLOP_allocation$(B, \gamma)$        ▷ split $B$ into $M$ episodes of horizon $L$
 2: Initialize root node $\mathcal{N}_0$; for all edges set $N \leftarrow 0$, $W \leftarrow 0$
 3: for $m = 1$ to $M$ do
 4:     $path \leftarrow \{\}$, $n \leftarrow \mathcal{N}_0$, $d \leftarrow 0$
 5:     while $d < L$ and $n$ is fully expanded and $n$ not terminal do
 6:         $a \leftarrow \arg\max_a \Big[\hat{Q}(n,a) + c\sqrt{\frac{\ln N(n)}{N(n,a)+\varepsilon}}\Big]$
 7:         $path$.append$(n,a)$; $n \leftarrow n$.child$(a)$; $d \leftarrow d+1$
 8:     end while
 9:     if $d < L$ and $n$ not terminal then
10:         Expand $n$: for all $a \in \mathcal{A}$ add child nodes
11:         Choose one unvisited action $a$; $path$.append$(n,a)$; $n \leftarrow n$.child$(a)$
12:     end if
13:     $R \leftarrow 0$, $\Delta \leftarrow 1$, $s \leftarrow n.state$
14:     while $d < L$ and $s$ not terminal do
15:         Sample $a \sim \pi_{\text{roll}}(s)$; $(s,r) \leftarrow G(s,a)$
16:         $R \leftarrow R + \Delta\, r$; $\Delta \leftarrow \Delta \cdot \gamma$; $d \leftarrow d+1$
17:     end while
18:     for each $(\bar{n}, \bar{a})$ in $path$ reverse order do
19:         $N(\bar{n}) \leftarrow N(\bar{n}) + 1$;  $N(\bar{n},\bar{a}) \leftarrow N(\bar{n},\bar{a}) + 1$
20:         $W(\bar{n},\bar{a}) \leftarrow W(\bar{n},\bar{a}) + R$
21:         $R \leftarrow r(\bar{n},\bar{a}) + \gamma\, R$
22:     end for
23: end for
24: return $\arg\max_a N(\mathcal{N}_0, a)$
```

3.2 Reducing Exposure Bias

DTs are trained with teacher forcing where every decoder step receives the ground-truth action as input. At test time, however, the policy must instead feed back its own predictions, a train–test mismatch that accumulates errors and degrades performance exposure bias [55]. We investigate two purely supervised remedies that integrate seamlessly with DT, self-rollout [56] and scheduled sampling [55]. Both are implemented in Algorithm 5 and Algorithm 6.

Self-rollout refines tokens within a frame and supplies rich token-level gradients. Scheduled sampling operates across frames and teaches the network to survive its own history. Both methods are label-efficient and require no modification to the DT. Experiments in Section 4.2 analyze the improvements resulting from these two strategies as compared to plain DT.

3.2.1 Self Rollout Training

Self-rollout refines the action for every frame inside a single forward pass. At time step t we allocate a fixed refinement budget $K = 4$ and iteratively update a previous action sequence $\tilde{a}_{t,0:K-1}$. At iteration $k \in \{0, \dots, K-1\}$ the decoder receives the state encoding $\mathbf{z}_{t,k}$ and all previously chosen actions $\tilde{a}_{t,0:k-1}$. It produces logits $\ell_{t,k} \in \mathbb{R}^{|\mathcal{A}|}$ and an action prediction

$$\hat{a}_{t,k} \sim \mathrm{Softmax}\big(\ell_{t,k}/\tau\big), \tag{3.2}$$

where $\tau > 0$ is a fixed temperature and $|\mathcal{A}|$ is the size of the action space.

Ground-truth tokens are gradually replaced by the model's own predictions

$$\tilde{a}_{t,k} = \begin{cases} a_t^\star, & \text{with probability } \varepsilon(e), \\ \hat{a}_{t,k-1}, & \text{otherwise,} \end{cases} \qquad \varepsilon(e) = 1 - \frac{e}{E}, \tag{3.3}$$

where $a_t^\star$ is the ground truth action, e is the current epoch, E is the total number of epochs and $\hat{a}_{t,k-1}$ denotes the action predicted at refinement iteration $k-1$ within the same frame t. The linear decay $\varepsilon(e)$ follows the curriculum idea introduced in the sequence-modeling literature [55]. In contrast to the across-frame variant presented later in Section 3.2.2, this schedule operates within a single frame. At test time we fix $\varepsilon = 0$ and perform the same K refinements once per frame.

The loss averaged cross-entropy over all refinement steps is

$$\mathcal{L}_{\mathrm{SR}} = \frac{1}{K} \sum_{k=0}^{K-1} \sum_{t=1}^{T} m_{t,k}\ \mathrm{CE}\big(\ell_{t,k}, a_t^\star\big), \tag{3.4}$$

where $m_{t,k} \in \{0, 1\}$ masks padding positions and CE() denotes cross entropy.

Algorithm 5 Self-Rollout Inference for Frame t

```
Require: Encoded state fragments $\mathbf{z}_{t,0:K-1}$, refinement steps $K$, temperature $\tau$
1: Initialise empty list $\mathbf{u} \leftarrow [\,]$
2: for $k = 0$ to $K-1$ do
3:     context $\leftarrow \big(\mathbf{z}_{t,0}, u_0, \dots, \mathbf{z}_{t,k-1}, u_{k-1}, \mathbf{z}_{t,k}\big)$
4:     $\ell_{t,k} \leftarrow \pi_\theta(\text{context})$
5:     $\hat{a}_{t,k} \sim \mathrm{Softmax}(\ell_{t,k}/\tau)$
6:     Append $\hat{a}_{t,k}$ to $\mathbf{u}$
7: end for
8: return $\hat{a}_t = \mathbf{u}[K-1]$
```

3.2.2 Scheduled Sampling Training

Scheduled sampling mitigates exposure bias by feeding the decoder a mixture of ground-truth and self-generated actions drawn from earlier frames. Let $a_t^\star$ denote the ground truth action at frame t and $\hat{a}_{t-1}$ the action predicted at frame $t-1$. At training epoch e the decoder input becomes

$$\tilde{a}_{t-1} = b_t a^{\star}_{t-1} + (1-b_t)\hat{a}_{t-1}, \qquad b_t \sim \text{Bernoulli}\big(\varepsilon(e)\big), \quad \varepsilon(e) = 1 - \frac{e}{E}, \tag{3.5}$$

where $b_t \in \{0,1\}$ selects between ground truth and model predictions. E is the total number of epochs. The linear schedule $\varepsilon(e)$ follows the original formulation of scheduled sampling in [55]. When $\varepsilon(e)$ decays to zero the model trains entirely on its own predictions, improving robustness to cascading errors. During evaluation we fix $\varepsilon = 0$. The loss is the cross-entropy between the logits $\ell_{1:T} = \pi_\theta(s_{1:T}, \tilde{a}_{0:T-1})$ and the target actions $a^{\star}_{1:T}$ as defined by

$$\mathcal{L}_{\text{SS}} = \sum_{t=1}^{T} \text{CE}(\ell_t, a^{\star}_t). \tag{3.6}$$

Algorithm 6 Scheduled Sampling for One Minibatch

```
Require: minibatch $\{\tau_j\}_{j=1}^{B}$, Bernoulli schedule $\varepsilon(e)$
 1: for all $\tau = (s_1, a^\star_1, \dots, s_T, a^\star_T)$ do
 2:     $\hat{a}_{1:T} \leftarrow \arg\max \pi_\theta(s_{1:T}, a^\star_{0:T-1})$
 3:     $\tilde{a}_0 \leftarrow$ null
 4:     for $t = 1$ to $T-1$ do
 5:         sample $b_t \sim \text{Bernoulli}(\varepsilon(e))$
 6:         $\tilde{a}_t \leftarrow b_t\, a^\star_t + (1-b_t)\,\hat{a}_t$
 7:     end for
 8:     $\ell_{1:T} \leftarrow \pi_\theta(s_{1:T}, \tilde{a}_{0:T-1})$
 9:     $\mathcal{L} \mathrel{+}= \sum_{t=1}^{T} \text{CE}(\ell_t, a^\star_t)$
10: end for
11: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
```

3.3 Uncertainty Weighted Decision Transformer

Offline decision-making policies based on the DT formulation achieve state-of-the-art performance by reframing RL as conditional sequence modeling over returns, states, and actions [19]. During training, every transition receives the same loss weight. This uniform weighting causes DTs to memorize abundant, low-risk regimes while leaving rare yet safety-critical states (e.g., roundabout entries, forced merges, and short-gap yields) insufficiently trained, even though these sparse cases dominate safety metrics [57].

Imbalanced data steer gradients toward frequent, low-risk cases. Focal Loss counters this effect by stressing hard samples in dense object detection [58]. Focal Loss is defined as

$$L = -w_i\,(1-p_i)^{\gamma}\,\log p_i, \tag{3.7}$$

where p_i is the predicted probability of class i, w_i is an optional class or sample weight, and $\gamma > 0$ modulates the strength of the focusing term. Unified Focal Loss generalizes the idea by combining Dice and cross-entropy losses [59]. Uncertainty-Aware Focal Loss strengthens it further by linking the factor $(1-p_i)^{\gamma}$ to pixel-wise predictive variance, which improves robustness in safety-critical segmentation [60].

A similar weighting design appears in knowledge distillation. Logit uncertainty distillation converts teacher entropy into a reliability weight

$$w = f\big(H(\mathbf{p}^{(T)})\big), \qquad H(\mathbf{p}^{(T)}) = -\sum_c p_c^{(T)} \log p_c^{(T)}, \tag{3.8}$$

where $\mathbf{p}^{(T)}$ denotes the teacher soft-max vector and $f(\cdot)$ maps low entropy outputs to larger weights [28]. Teaching with Uncertainty adopts Monte-Carlo dropout, passes the resulting variance to the student, and boosts detection accuracy with limited labels [29].

The uncertainty signal that guides vision models also stabilizes offline RL. UWAC reduces the impact of out-of-distribution transitions by scaling the temporal-difference loss with a dropout-based Q-variance estimate [30]. UNREST-DT follows a complementary strategy by measuring the mutual information between returns and state sequence and truncating the return target when mutual information exceeds a threshold, thereby suppressing unreliable high-variance segments [31].

Class-level imbalance calls for a different view. Class Uncertainty replaces raw class counts with the average predictive variance

$$w_c = \frac{1}{N_c} \sum_{i=1}^{N_c} \mathrm{Var}\big[p_c^{(i)}\big], \tag{3.9}$$

where w_c is the weight for class c, N_c is its sample count, and $p_c^{(i)}$ denotes the model probability for class c on sample i [61]. A larger variance indicates a rarer or more ambiguous class and thus receives a larger weight. Skill-level rebalancing extends this concept to sequential decision tasks. The Skill Transformer first enumerates the skill inventory, then estimates each skill's empirical frequency, and finally applies a Focal-Loss-style modulation to emphasize rare skills [62]. Counting skills and their occurrence probabilities, however, becomes infeasible in long-horizon driving where the space of possible maneuvers grows rapidly.

Our approach removes this bottleneck. A high-capacity teacher DT supplies token-level entropy, which we convert into the weight

$$w_t = \big(1 - p_t\big)^{\gamma}, \tag{3.10}$$

where p_t is the teacher confidence for token t and $\gamma > 0$ is a focusing parameter identical to that in Focal Loss. The student Decision Transformer applies w_t directly to its cross-entropy objective. No skill enumeration, frequency counting, or hand-crafted class mapping is required. This design unifies uncertainty-aware knowledge distillation and offline-RL weighting within a single, fully differentiable objective. By operating at sequence level, it corrects both state and action imbalance and preserves temporal coherence, a property that static class-based schemes lack.

To be specific, the Uncertainty Weighted Decision Transformer (UWDT) is trained in three coherent stages:

1. **Teacher fitting.** Train a DT π_T on the offline dataset. The loss is the masked cross-entropy over valid tokens. After convergence, freeze all Teacher parameters.

2. **Entropy calibration.** Run the frozen Teacher DT π_T once over the entire dataset to obtain the per-token predictive entropy

$$H_t = -\sum_a p_{T,t}(a) \log p_{T,t}(a). \tag{3.11}$$

   Estimate robust bounds $H_{\min}$ and $H_{\max}$ and set the exponent

$$\gamma = \ln r \big/ \ln\!\big(H_{\max}/H_{\min}\big), \tag{3.12}$$

   where r is the desired dynamic range of the weights.

3. **Student distillation.** Initialize a Student DT π_S with the same architecture as π_T. For each mini-batch, compute raw weights $\tilde{w}_t = H_t^{\gamma}$, normalize them to unit mean, and clip them to the maximum value $w_{\max}$. Optimize π_S with the weighted cross-entropy loss

$$\mathcal{L}_S = -\frac{1}{|\mathcal{B}_{\text{val}}|} \sum_{t \in \mathcal{B}_{\text{val}}} \bar{w}_t \log p_{S,t}(a_t). \tag{3.13}$$

The result is a Student policy that devotes more gradient effort to tokens where the Teacher is uncertain, improving performance in rare or ambiguous driving scenarios without altering the model architecture.

3.3.1 Teacher Decision Transformer

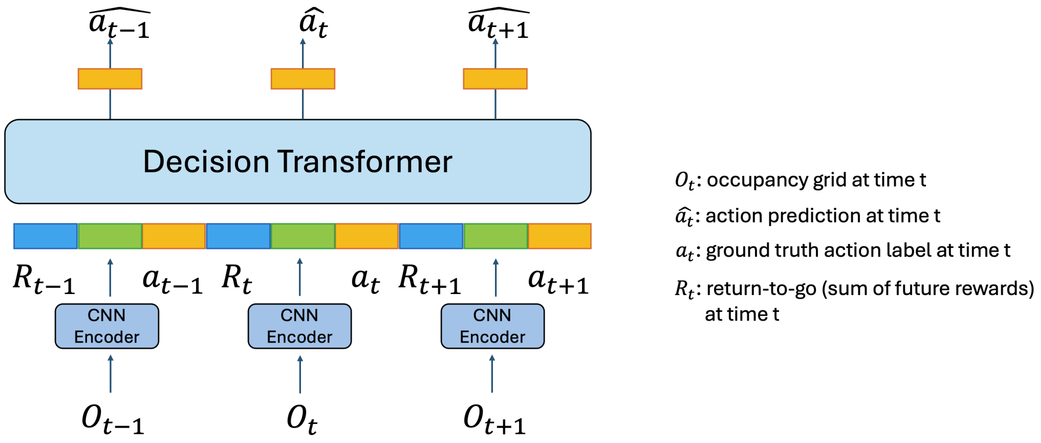


Figure 3.1: Teacher Decision Transformer is a standard DT with CNN Encoders.

As shown in Figure 3.1, the teacher DT uses CNN encoders to capture spatial information and DT for temporal reasoning. The CNN encoder structure is show in Figure 3.2. Each state tensor s_t is encoded by a convolutional network: three 3×3 convolutions with stride 2 (channels $4 \to 32 \to 64 \to 128$), ReLU activations, batch normalization, and spatial dropout. The final feature map is flattened and projected to a fixed d_e–dimensional embedding via a lazy fully connected layer. Denote the resulting embedding as

$$\phi_t = \phi(s_t) \in \mathbb{R}^{d_e}, \qquad d_e = 32. \tag{3.14}$$

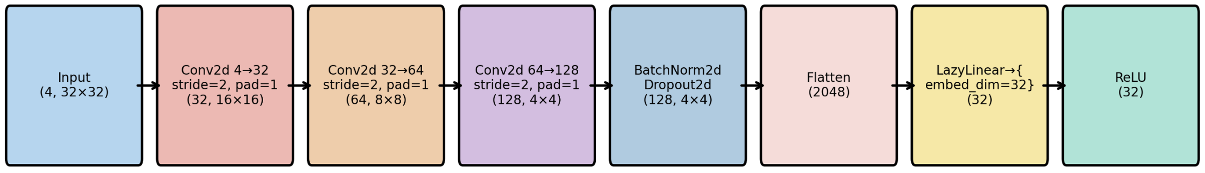

Input (4, 32×32)
Conv2d 4→32 stride=2, pad=1 (32, 16×16)
Conv2d 32→64 stride=2, pad=1 (64, 8×8)
Conv2d 64→128 stride=2, pad=1 (128, 4×4)
BatchNorm2d Dropout2d (128, 4×4)
Flatten (2048)
LazyLinear→{ embed_dim=32} (32)
ReLU (32)

Figure 3.2: The CNN-based encoder structure.

Consider a finite-horizon MDP $\big(\mathcal{S}, \mathcal{A}, P, r, \gamma\big)$ with discount $\gamma \in [0, 1)$. Given an offline dataset $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$ where each trajectory $\tau = (s_1, a_1, r_1, \ldots, s_T, a_T, r_T)$, we first compute the *return-to-go* (RTG) for every time-step:

$$R_t \;=\; \sum_{k=t}^{T} \gamma^{\,k-t}\, r_k, \quad t = 1, \ldots, T. \tag{3.15}$$

We then interleave the scalar return, state, and action tokens to obtain the length-$3K$ sequence

$$\Big(R_{t-K+1},\, s_{t-K+1},\, a_{t-K},\; \ldots,\; R_t,\, s_t\Big), \tag{3.16}$$

which is then fed into a causal Transformer π_T that autoregressively predicts the next action distribution $\pi_T(a_t \,|\, \cdot)$. DT is trained purely by supervised learning to maximize the log-likelihood of expert actions conditioned on the context:

$$\begin{aligned} \mathcal{L}(\theta) &= -\mathbb{E}_{\tau\sim\mathcal{D}} \sum_{t=1}^{T} \log \pi_T\big(a_t \quad R_t, s_t, a_{t-1}, \ldots, R_{t-K+1}, s_{t-K+1}, a_{t-K}\big) \\ &= -\mathbb{E}_{\tau\sim\mathcal{D}} \sum_{t=1}^{T} \log \pi_T\big(a_t \quad X_{1:t}\big), \end{aligned} \tag{3.17}$$

where $X_{1:t}$ denotes the length-$3K$ token context prior to a_t. Gradient updates are performed with AdamW and running-mean-square layer-norm, following [19].

At inference time the Decision Transformer operates in a discrete action space. At each timestep t it first forms the policy distribution $\pi_T(\cdot \mid X_{1:t})$. It then generates a_t using one of two polices

$$a_t \;=\; \begin{cases} \text{sample}\big(\pi_T(\cdot \mid X_{1:t})\big), & \text{stochastic rollout,} \\ \arg\max\limits_{a} \; \pi_T\big(a \mid X_{1:t}\big), & \text{greedy rollout.} \end{cases} \tag{3.18}$$

The selected action is then converted to a one-hot vector. The triplet $(\tilde{R}_{t+1},\, s_{t+1},\, a_t)$ is then appended to the autoregressive input sequence, yielding $X_{1:t+1}$. In practice, we adopt the greedy rollout because it consistently delivers higher driving performance and lower variance. After convergence, Teacher parameters are frozen and reused for entropy estimation during Student DT training. The pseudocode and training hyperparameters are presented in Algorithm 7 and Table 3.1, respectively.

Symbol	Meaning	Value
K	Context length (seq. window)	20
d	Embedding dimension	32
γ	Discount factor for returns-to-go	0.99
η	Learning rate	1×10^{-5}
B	Batch size	16
E	Training epochs	20
λ_{wd}	Weight decay (AdamW)	5×10^{-5}
ρ_{wu}	Linear warm-up ratio	0.1
$g_{\max}$	Gradient-norm clip	0.25
n_{layer}	Transformer decoder layers	4
n_{head}	Attention heads	1

Table 3.1: UWDT hyper-parameters used in this study.

Algorithm 7 Decision Transformer: Return-Conditioned Sequence Modeling

```
Require: Offline dataset $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$ with trajectories $\tau = (s_1, a_1, r_1, \ldots, s_T, a_T, r_T)$
Require: Context length $K$, discount $\gamma$, learning rate $\eta$, batch size $B$
Require: Causal Transformer $\pi_T$ with three token embeddings (return, state, action)
 1: for iteration $= 1, \ldots$ do
 2:     Sample $B$ trajectories $\{\tau_j\}_{j=1}^{B}$ from $\mathcal{D}$
 3:     for all $\tau_j$ do
 4:         Pick random index $t \in [K, T]$
 5:         Compute returns-to-go $R_k \leftarrow \sum_{u=k}^{T} \gamma^{u-k} r_u$ for $k = t - K + 1 \ldots, t$
 6:         Form token sequence
                $$X = \Big(R_{t-K+1}, s_{t-K+1}, a_{t-K},\ \ldots,\ R_t, s_t\Big)$$
 7:         $\hat{p}(\cdot) \leftarrow \pi_T(X)$
 8:         $\mathcal{L} \mathrel{+}= -\log \hat{p}(a_t)$
 9:     end for
10:     $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
11: end for
12: function SelectAction($s_t$, $R^{\text{target}}$, history)
13:     Update remaining return $R_t \leftarrow R^{\text{target}} - \sum_{u=1}^{t-1} r_u$
14:     Construct latest context of length $K$ from history + $(R_t, s_t)$
15:     $p(\cdot) \leftarrow \pi_T(\text{context})$
16:     return $\arg\max_a p(a)$
17: end function
```

3.3.2 Student Decision Transformer

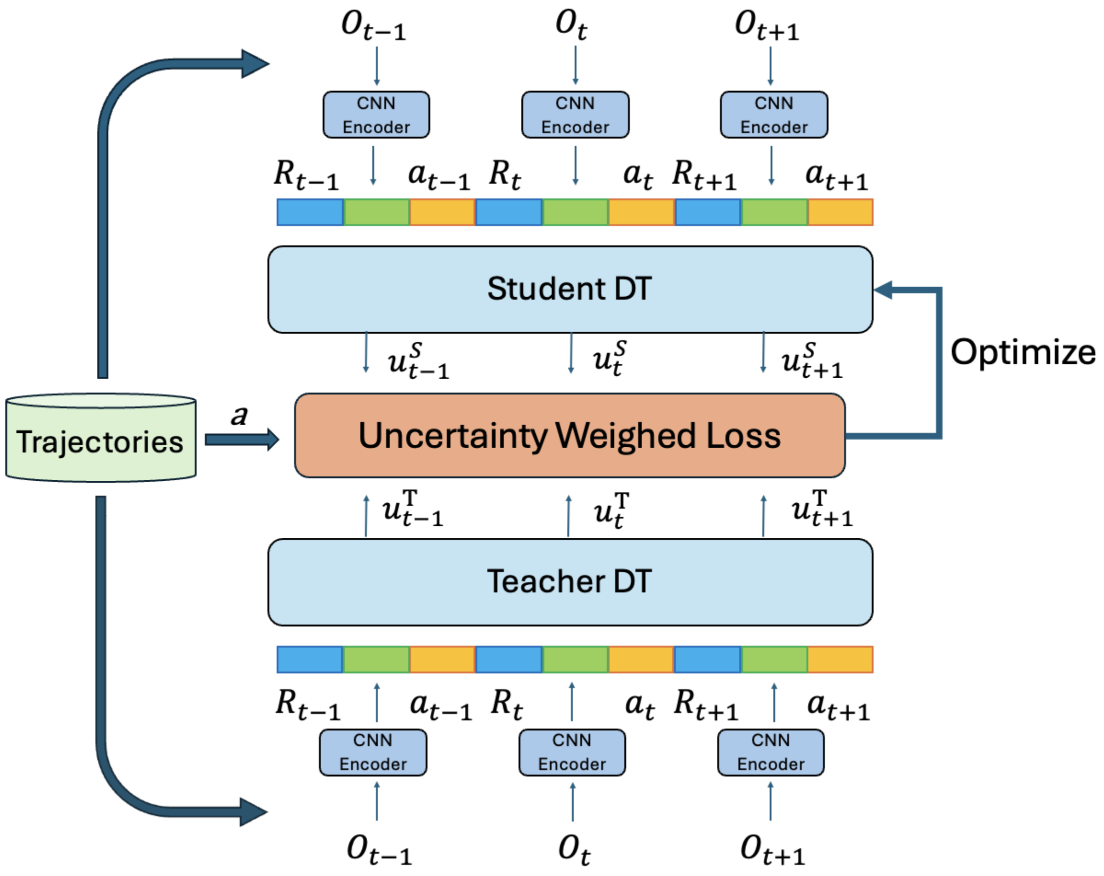


Figure 3.3: Teacher DT predictive uncertainty is estimated via the entropy of the action distribution. This entropy then acts as a weighting signal during student DT training, modulating the impact of each action prediction on the loss.

The student policy π_T shares the architecture with the teacher DT. After obtaining the trained Teacher DT, we estimate predictive uncertainty by computing the entropy of its action distribution. During Student training, this entropy serves as an uncertainty-based weight, modulating the loss contribution of each training token. Given frozen Teacher logits u_t^T for a batch token t, we form probabilities $p_{T,t}(a) = \mathrm{softmax}(u_t^T)_a$. The predictive entropy of the action distribution is

$$H_t = -\sum_{a=0}^{A-1} p_{T,t}(a) \log p_{T,t}(a), \qquad 0 \le H_t \le \log A. \tag{3.19}$$

Higher H_t indicates the Teacher is uncertain about which action matches the dataset at that token. We wish to emphasize samples where the Teacher exhibits high uncertainty. Let $H_{\min}$ and $H_{\max}$ denote the minimum and maximum Teacher entropies observed across the training set. In practice, after training the Teacher DT, we deploy it in the simulation environment for 400 episodes. During these rollouts, we record the entropy of the Teacher's action distribution at each timestep. We then compute the average, minimum, and maximum entropy values across all collected tokens to quantify the Teacher's confidence range in deployment scenarios. Collecting entropy during rollout rather than from the static training set has two advantages. First, it reflects the Teacher's behavior in realistic, sequentially generated trajectories, where compounding errors and distribution shift may arise. Second, it provides a more accurate estimate of uncertainty under deployment dynamics, which may include rare or out-of-distribution states not well represented in the training set.

A standard DT or the Teacher DT is trained by supervised behavior cloning on the offline dataset using the standard masked cross-entropy over valid tokens:

$$\mathcal{L}_T = -\frac{1}{|\mathcal{B}_{\text{val}}|} \sum_{t \in \mathcal{B}_{\text{val}}} \log p_T\big(a_t \mid \mathbf{z}_{\le t}\big), \tag{3.20}$$

where the sum ranges over all non-padded positions in the mini-batch $\mathcal{B}_{\text{val}}$.

Compares to the above Teacher DT, the Student policy π_S is architecturally identical to the Teacher but trained with entropy-weighted supervision. Let u_t^S be Student logits and $p_{S,t}(a) = \text{softmax}(u_t^S)_a$. For each valid token ($m_t = 1$) we compute the masked cross-entropy scaled by $\bar{w}_t$:

$$\mathcal{L}_S = -\frac{1}{|\mathcal{B}_{\text{val}}|} \sum_{t \in \mathcal{B}_{\text{val}}} \bar{w}_t \, \log p_{S,t}(a_t). \tag{3.21}$$

Eq. (3.21) reduces to standard behavior cloning when all $\bar{w}_t$=1. When $\gamma > 0$, tokens with higher Teacher entropy carry larger gradient weight, encouraging the Student to allocate additional capacity to states the Teacher found ambiguous.

Let $r > 1$ denote the desired ratio between the largest and smallest per-token weights. We adopt a power mapping

$$\tilde{w}_t = H_t^{\gamma}, \tag{3.22}$$

with exponent $\gamma > 0$ chosen so that the dynamic range matches r:

$$\gamma = \frac{\ln r}{\ln\big(H_{\max}/H_{\min}\big)}. \tag{3.23}$$

Because the scale of $\tilde{w}_t$ influences optimization, we normalize in each mini-batch:

$$w_t = \frac{\tilde{w}_t}{\frac{1}{M}\sum_{j=1}^{M} \tilde{w}_j}, \tag{3.24}$$

where M is the number of valid (non-padded) tokens in the batch. Finally we clip to a ceiling $w_{\max}$:

$$\bar{w}_t = \min\{w_t, w_{\max}\}. \tag{3.25}$$

We set $w_{\max} = 1.5$ and $r = 1.3$. The Student DT hyperparameters are the same with Teacher DT, shown in Table 3.1. The pseudocode of UWDT are presented in Algorithm 8.

Algorithm 8 Uncertainty Weighted Decision Transformer (UWDT)

```
Require: Offline dataset $\mathcal{D}$, horizon $L$, batch size $B$
Require: Target range $r$, max weight $w_{\max}$, learning rates $\eta_T, \eta_S$
Require: Causal Transformers $\pi_T, \pi_S$
    Stage 1: Teacher training
 1: for epoch $= 1, \ldots, E_T$ do
 2:     Sample mini-batch $\mathcal{B} \subset \mathcal{D}$
 3:     Mask padded tokens $\mathcal{B}_{\text{val}}$
 4:     $\mathcal{L}_T \leftarrow -\frac{1}{|\mathcal{B}_{\text{val}}|} \sum_{t \in \mathcal{B}_{\text{val}}} \log p_T(a_t \mid \mathbf{z}_{\leq t})$
 5:     $\theta_T \leftarrow \theta_T - \eta_T \nabla_{\theta_T} \mathcal{L}_T$
 6: end for
 7: Freeze $\theta_T$
    Stage 2: Entropy statistics
 8: Run $\pi_T$ over all tokens in $\mathcal{D}$ to get entropies $H_t$
 9: Compute $H_{\min}, H_{\max}$
10: $\gamma \leftarrow \frac{\ln r}{\ln(H_{\max}/H_{\min})}$
    Stage 3: Student training
11: for epoch $= 1, \ldots, E_S$ do
12:     Sample mini-batch $\mathcal{B} \subset \mathcal{D}$
13:     for all $t \in \mathcal{B}_{\text{val}}$ do
14:         $\tilde{w}_t \leftarrow H_t^{\gamma}$
15:     end for
16:     Normalize $\tilde{w}_t$ and clip to $w_{\max}$, giving $\bar{w}_t$
17:     $\mathcal{L}_S \leftarrow -\frac{1}{|\mathcal{B}_{\text{val}}|} \sum_{t \in \mathcal{B}_{\text{val}}} \bar{w}_t \log p_{S,t}(a_t)$
18:     $\theta_S \leftarrow \theta_S - \eta_S \nabla_{\theta_S} \mathcal{L}_S$
19: end for
20: return $\pi_S$
21: function SelectAction($s_t$, $R^{\text{target}}$, history)
22:     $R_t \leftarrow R^{\text{target}} - \sum_{u=1}^{t-1} r_u$
23:     Construct latest context of length $K$ from history $+ (R_t, s_t)$
24:     $p(\cdot) \leftarrow \pi_S(\text{context})$
25:     return $\arg\max_a p(a)$
26: end function
```

Chapter 4

Experimental Results

This chapter empirically evaluates the proposed approach under three levels of roundabout congestion and compares it with strong online, offline, and imitation-learning baselines. Each controller is trained and tested under identical hyper-parameters and five random seeds; performance is measured by accumulated reward, collision rate, average speed, halt duration, and time-to-exit, covering both efficiency and safety. Three findings emerge:

1. UWDT sustains near-optimal reward even when four interacting vehicles occupy the circulating lanes, whereas DT and SAC degrade gracefully and BC or CQL collapse.
2. Entropy histograms show that UWDT selectively broadens its action distribution during merging conflicts yet remains deterministic in routine flow, indicating that the weighting strategy activates precisely where uncertainty peaks.
3. Outcome variance across seeds is minimal, confirming that the observed gains stem from the objective rather than favorable randomness.

Together, these results demonstrate the practicality of entropy-modulated sequence learning for safety-critical autonomous driving.

4.1 Online and Offline RL Baseline

Our baselines comprise an online RL agent trained with SAC and an offline RL agent trained with CQL. Each algorithm was run with five independent seeds, each seed executing 1 000 episodes. Figure 4.1 reports the mean episode reward over all seeds. CQL saturates at a reward value of approximately 12.5 and shows little improvement, whereas SAC exhibits a steady rise and eventually outperforms CQL.

The reward function balances two terms: keeping the ego-vehicle within 8 – 16 m/s and avoiding collisions. Figures 4.2 and 4.3 reveal that CQL maintains a non-negligible collision rate while cruising below the desired speed band. Both shortcomings depress its cumulative reward. SAC, in contrast, progressively lowers its collision rate, boosting the safety component of the reward. It achieves this by adopting an ever more conservative speed profile; by the end of training, the average speed approaches 0. Stopping before entering the roundabout maximizes safety under the current weighting scheme but sacrifices driving efficiency.

Figures 4.4 and 4.5 corroborate this interpretation. The episode horizon is capped at 22 decision steps. SAC approaches this upper bound as collisions disappear, yet the traveled distance per episode shrinks because the vehicle moves slowly. CQL shows the opposite trend: shorter, riskier episodes cover slightly longer distances but incur more crashes and therefore lower reward. These results highlight a limitation of vanilla SAC under our safety-oriented reward: the agent learns to drive slowly regardless of traffic situations rather than navigate efficiently.

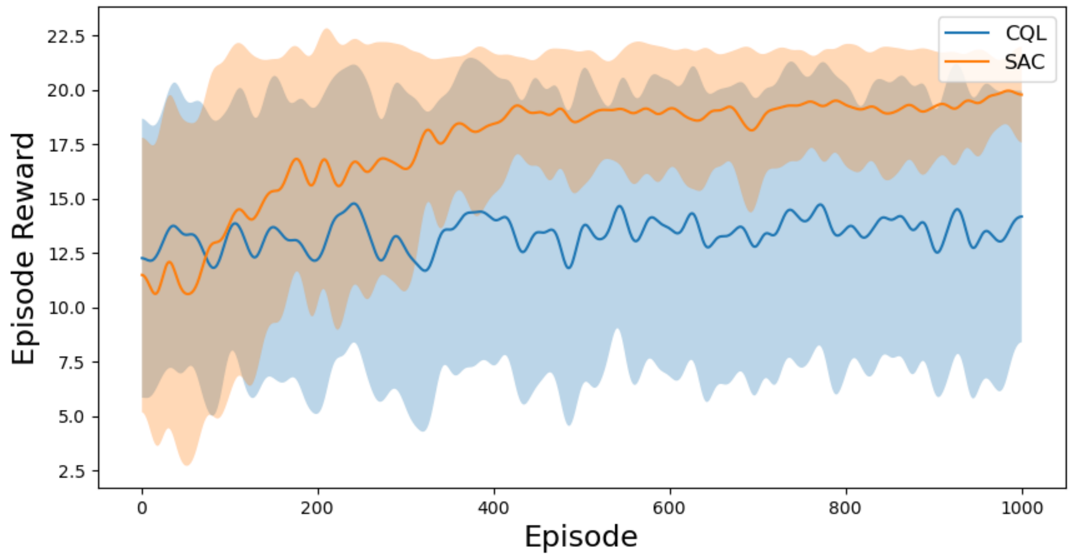


Figure 4.1: Episode reward during training.

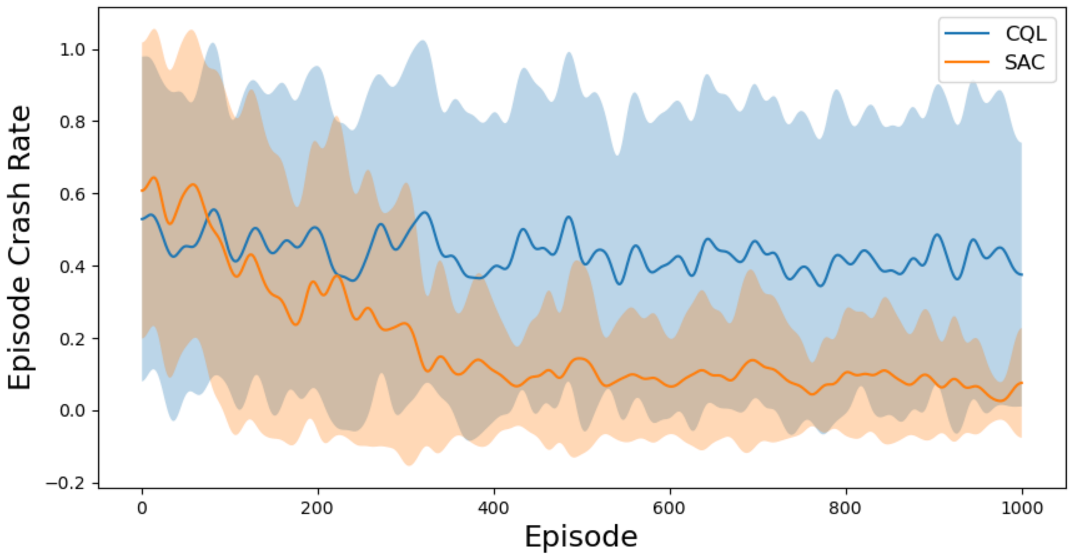


Figure 4.2: Episode collision rate during training.

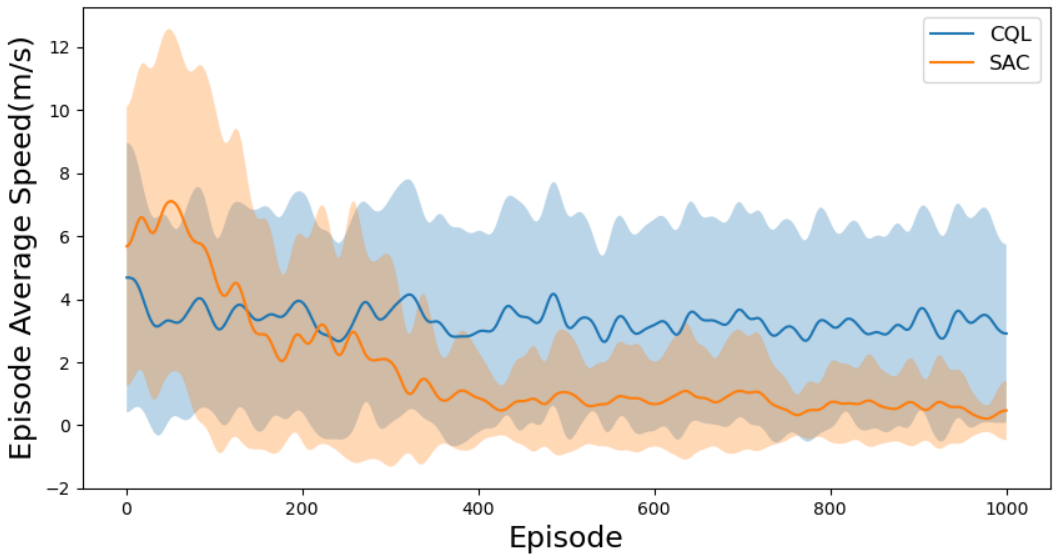


Figure 4.3: Episode average speed during training.

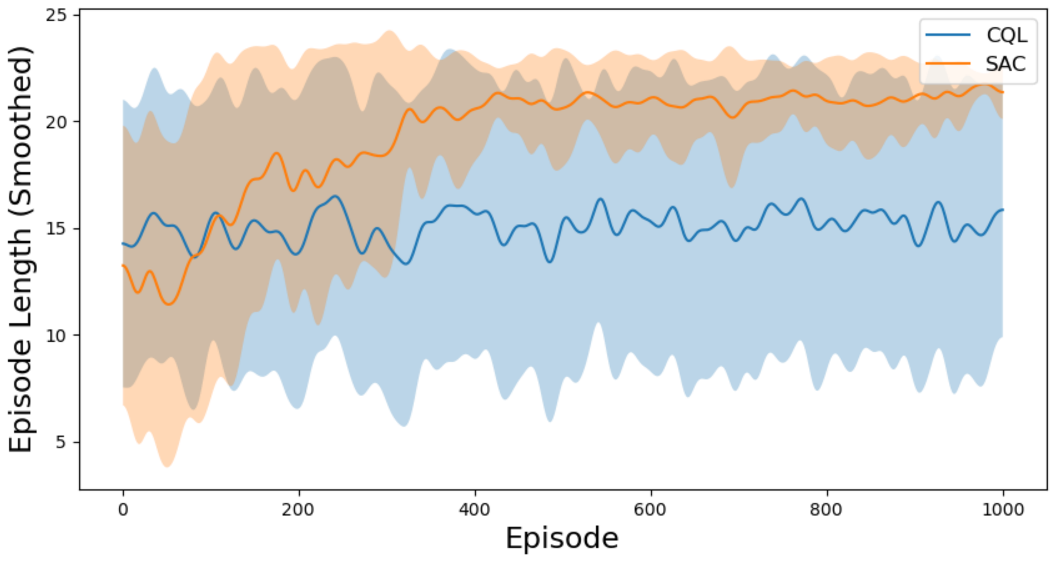


Figure 4.4: Episode length during training.

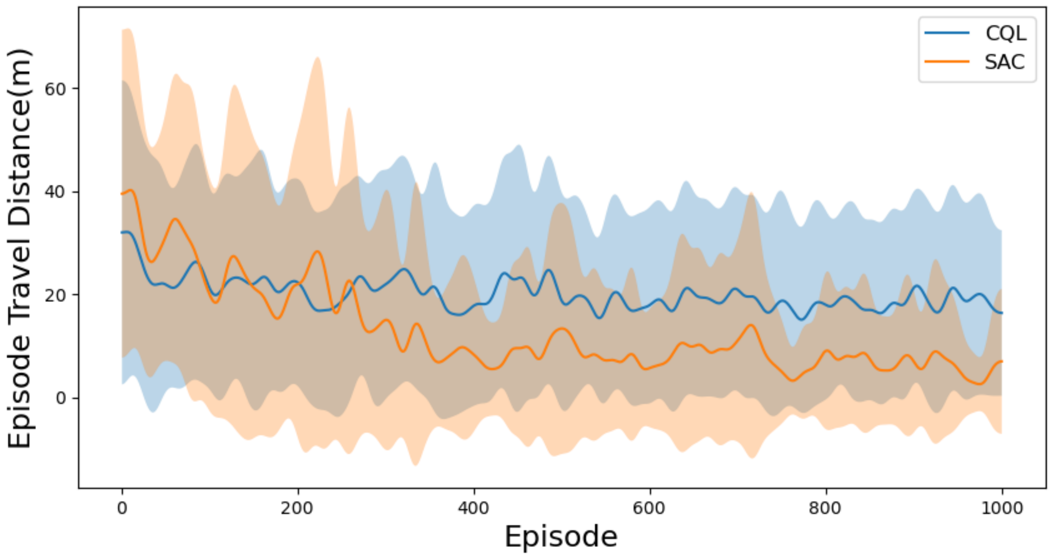


Figure 4.5: Episode distance during training.

4.2 Reducing Exposure Bias

To benchmark alternative strategies for improving robustness in rare or high-uncertainty states, we test two teacher-forcing variants of the Decision Transformer: self rollout and scheduled sampling.

As shown in Table 4.1, DT already attains near-optimal performance, with a 94% exit rate and 6% collisions. Self rollout was introduced to mitigate exposure bias by augmenting the training set with model-generated trajectories, but in our purely offline setting this procedure injects low-quality samples whose return annotations are inconsistent with the original reward scale. The additional data therefore acts as noise. As a result, reward falls by almost 29%, collision rate rises, and episode length shortens. Because the Decision Transformer conditions on desired return, mislabeled rollouts distort that conditioning return value leading to an overall degradation. Scheduled sampling alternates between teacher-forced and model-generated tokens during training. However, scheduled sampling perturbs the supervised signal without providing extra reward feedback, so the model's return-conditioning objective remains unchanged.

When return-conditioning is accurate and the offline dataset covers the relevant state distribution, neither self-rollout nor scheduled sampling yields additional benefit. In fact, the former can poison training with low-fidelity data, while the latter merely injects noise with no upside in this control domain.

Method	Accumulated Reward	Average Speed [m/s]	Episode Length [s]	Travel Distance [m]	Reach Exit Rate [%]	Collision Rate [%]	Time–to–Exit [s]	Halt Duration ($v < 1\,\mathrm{m/s}$) [s]
DT	20.884 ± 4.166	15.027 ± 1.730	21.020 ± 3.888	315.863 ± 68.813	94.000 ± 23.780	6.000 ± 23.780	15.765 ± 1.650	0.000 ± 0.000
DT + self-rollout	14.912 ± 9.214	14.818 ± 1.191	15.223 ± 8.811	225.569 ± 131.813	62.750 ± 48.410	37.250 ± 48.410	17.608 ± 3.389	0.000 ± 0.000
DT + scheduled sampling	20.839 ± 4.258	15.056 ± 1.690	20.970 ± 3.980	315.724 ± 69.620	93.750 ± 24.240	6.250 ± 24.240	15.758 ± 1.681	0.000 ± 0.000

Table 4.1: Effect of self rollout and scheduled sampling on DT performance. Metrics are reported as mean ± standard deviation over 400 evaluation episodes on the roundabout task.

4.3 Uncertainty Weighted Decision Transformer

Offline RL methods aim to learn effective policies from previously collected data without further interaction with them environment. Distributional shift between the training data and the learned policy can cause standard off-policy algorithms to overestimate returns. CQL addresses this by learning a conservative Q-function whose expected value lower-bounds the true value of the policy and adds a simple Q-value regularizer to the Bellman error. SAC is an off-policy actor–critic algorithm based on the maximum entropy framework which optimizes both the expected return and an entropy bonus, yielding improved exploration and stability across tasks. BC treats policy learning as supervised regression from states to actions. It can be implemented with various network architectures including transformers but it typically imitates the behavior policy and may suffer from compounding errors on long horizons. The DT casts RL as a sequence modeling problem: a transformer is trained on sequences of returns, states and actions with an auto regressive loss, avoiding unstable dynamic programming and producing diverse behaviors by conditioning on desired returns. UWDT proposes to estimate epistemic uncertainty via conditional action prediction entropy and to condition on truncated returns, thereby mitigating over-optimism in stochastic environments.

In this section we evaluate our approach under varying traffic densities and compare it with several offline and imitation–learning baselines. We consider three RL algorithms: CQL, SAC, and our proposed UWDT. For imitation learning we include BC Transformer. Finally we evaluate the original DT. Throughout this section rewards are normalized in $[0, 1]$ and velocities are expressed in m/s.

4.3.1 Performance and Entropy Statistics Results

Table 4.2 summarizes key performance metrics for the compared methods. The Episode Length measures the average duration until episode termination, capped at 22 timesteps unless a collision occurs earlier. Time-to-Exit denotes the average time taken to reach the exit. If a collision occurs first, the maximum duration value of 22 timesteps is recorded. Halt Duration indicates the amount of time spent at speeds below 1 m/s.

Method	Accumulated Reward	Average Speed [m/s]	Episode Length [s]	Travel Distance [m]	Reach Exit Rate [%]	Collision Rate [%]	Time–to–Exit [s]	Halt Duration ($v < 1\,\mathrm{m/s}$) [s]
CQL	12.82 ± 7.55	7.67 ± 3.27	14.60 ± 7.82	111.96 ± 76.64	14.00 ± 34.74	49.00 ± 50.05	21.44 ± 1.62	2.48 ± 5.11
SAC	20.66 ± 1.99	10.42 ± 6.61	21.71 ± 1.27	228.24 ± 145.33	60.00 ± 49.05	1.50 ± 7.06	17.80 ± 3.43	6.01 ± 7.76
BC Transformer	13.63 ± 7.82	7.89 ± 0.16	15.26 ± 8.06	120.47 ± 63.64	0.00 ± 0.00	41.25 ± 49.29	22.00 ± 0.00	0.00 ± 0.00
DT	20.88 ± 4.17	15.03 ± 1.73	21.02 ± 3.89	315.86 ± 68.81	94.00 ± 23.78	6.00 ± 23.78	15.77 ± 1.65	0.00 ± 0.00
UWDT	$\mathbf{21.69 \pm 2.01}$	$\mathbf{15.31 \pm 0.79}$	$\mathbf{21.79 \pm 1.90}$	$\mathbf{333.55 \pm 33.73}$	$\mathbf{98.75 \pm 11.12}$	$\mathbf{1.25 \pm 11.12}$	$\mathbf{15.52 \pm 0.98}$	$\mathbf{0.00 \pm 0.00}$

Table 4.2: Performance comparison on the roundabout task. Eight metrics are reported including accumulated reward, average speed, episode length, travel distance, reach exit rate, collision rate, time to exit, and halt duration. Values are expressed as mean ± standard deviation over 400 evaluation episodes.

Method	Min entropy	Max entropy	Average entropy
BC Transformer	0.60	1.27	0.68 ± 0.07
DT	1.14	1.47	1.15 ± 0.02
UWDT	**1.14**	**1.34**	$\mathbf{1.15 \pm 0.01}$

Table 4.3: Action-distribution entropy statistics (minimum, maximum, and mean $\pm$ standard deviation over the evaluation set).

CQL and BC rely purely on offline data. CQL penalizes uncertain, out-of-distribution actions, obtaining the lowest reward of 12.82, the shortest trajectories of 14.60 s, and an exit rate of 14 %. BC Transformer also struggles due to compounding errors and the lack of uncertainty handling, resulting in a low reward of 13.63 and never reaching the exit. Its performance gap versus DT highlights poor generalization when the policy is purely imitative.

SAC, an online RL method, yields an average reward of 20.66, comparable to DT and UWDT due to reward normalization during training. However, higher rewards mainly indicate fewer collisions rather than optimal driving. With an episode length of 21.71 s and an exit success rate of 60 %, SAC often behaves overly cautiously, halting for 6.01 s. Observations show that SAC waits excessively at the roundabout entrance, causing potential congestion despite clear merging opportunities.

DT, which conditions actions on target returns, achieves notable improvements: a speed of 15.03 m/s, a travel distance of 315.86 m, and an exit rate of 94 %. Yet the lack of explicit uncertainty awareness sometimes causes premature terminations due to risky actions. Its maximum entropy of 1.47 reflects uncertainty without a mechanism to exploit it for safer decisions.

The proposed UWDT explicitly integrates epistemic uncertainty, outperforming all baselines. It attains the highest reward of 21.69, the fastest speed of 15.31 m/s, the greatest travel distance of 333.55 m, and a near-perfect exit rate of 98.75 %. Table 4.3 corroborates these findings. UWDT maintains a more tightly controlled entropy range of 1.14–1.34, confirming better-calibrated uncertainty than DT.

4.3.2 Entropy Distribution Results for DT and UTDT

As previously discussed, the computational demand of MCTS grows with traffic density in the roundabout. Specifically, as the number of incoming vehicles increases, MCTS requires more simulation rollouts and longer decision times to generate feasible actions. We quantify scenario complexity by the number of incoming vehicles present at the entrance of the roundabout.

During training, the number of incoming vehicles n is uniformly sampled from $\{0, 1, 2, 3, 4\}$ to expose the agent to a diverse set of driving situations. At test time, we evaluate the learned policy under three separate complexity levels—low, medium, and high—corresponding to $n \in \{0, 1, 2\}$, $n = 3$, and $n = 4$, respectively.

Figures 4.6, 4.7, 4.8, and 4.9 depict the per-episode entropy distributions for DT in orange and UWDT in green. In all scenarios the two curves peak near $z \approx -0.3$, showing that both controllers choose nearly deterministic actions in routine states. In the simplest scenes, the entropy tails of DT and UWDT nearly overlap (Figure 4.7). The uncertainty weight rarely triggers, so UWDT behaves like DT. As scenario complexity increases with more interacting vehicles (Figures 4.7, 4.8, and 4.9), UWDT's entropy distribution shifts toward higher values relative to DT. This reflects lower confidence induced by penalizing rarely observed state–action pairs, rather than ignoring them and overfitting to common patterns. The reweighting surfaces safety-critical but rare actions more often and provides a stronger training signal.

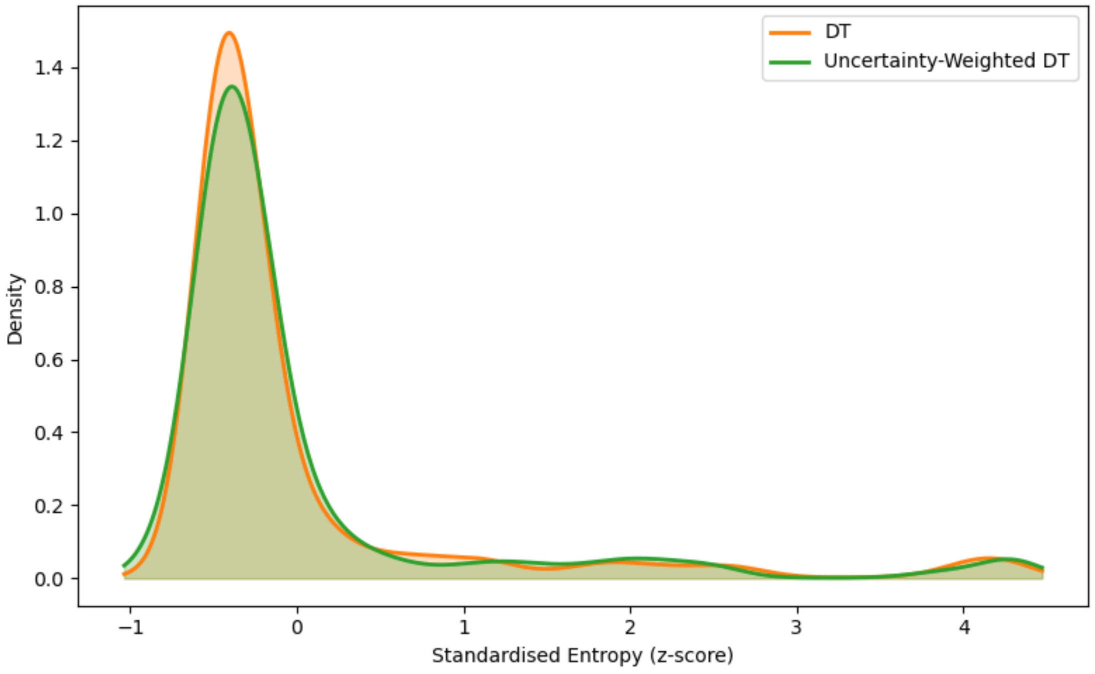


Figure 4.6: The Entropy distribution comparison of DT and UWDT. The number of interacting vehicles is uniformly sampled from the range $[0, 4]$, and the results are aggregated over 400 episodes.

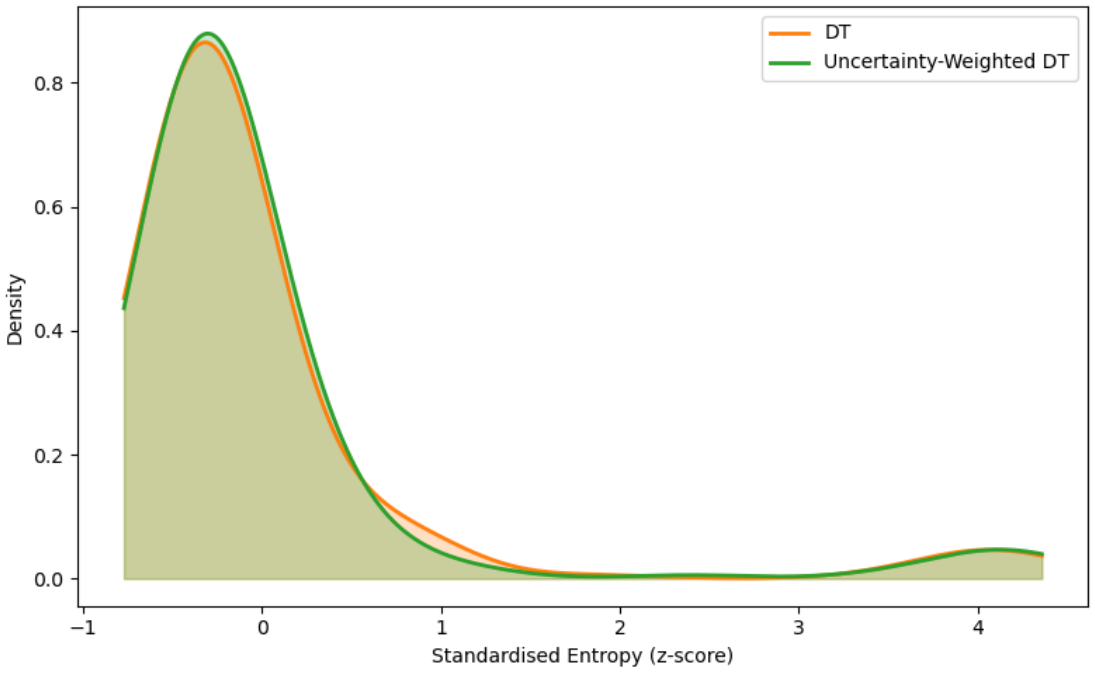


Figure 4.7: The Entropy distribution comparison of DT and UWDT. The number of interacting vehicles is uniformly sampled from the range $[0, 2]$, and the results are aggregated over 400 episodes.

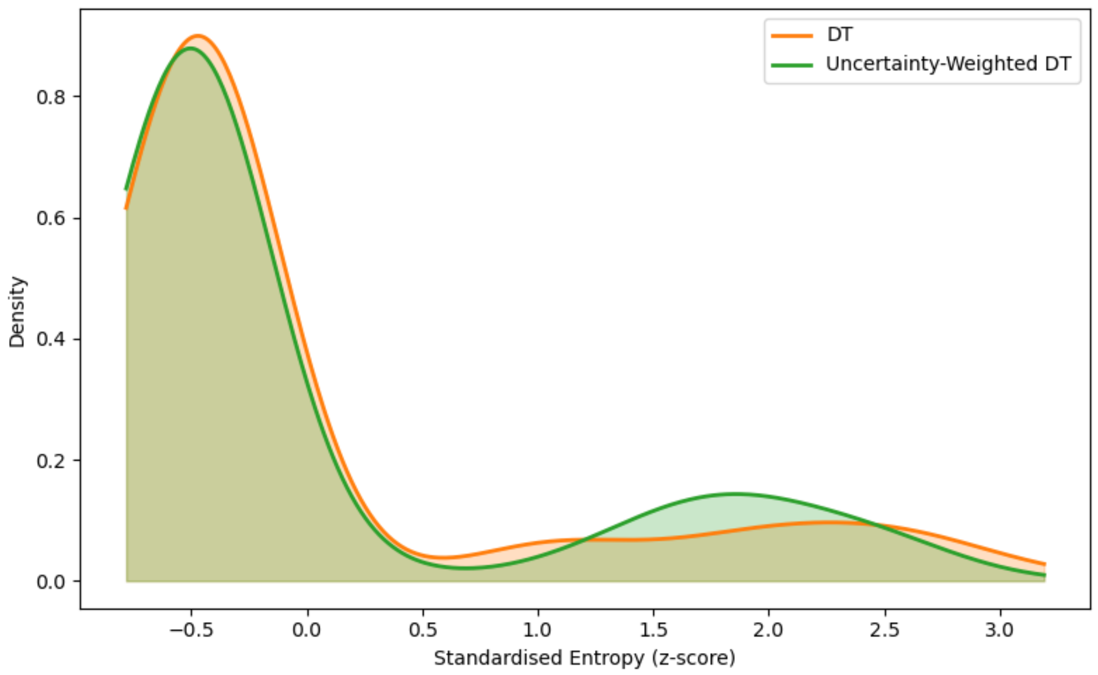


Figure 4.8: The Entropy distribution comparison of DT and UWDT. The number of interacting vehicles is 3, and the results are aggregated over 100 episodes.

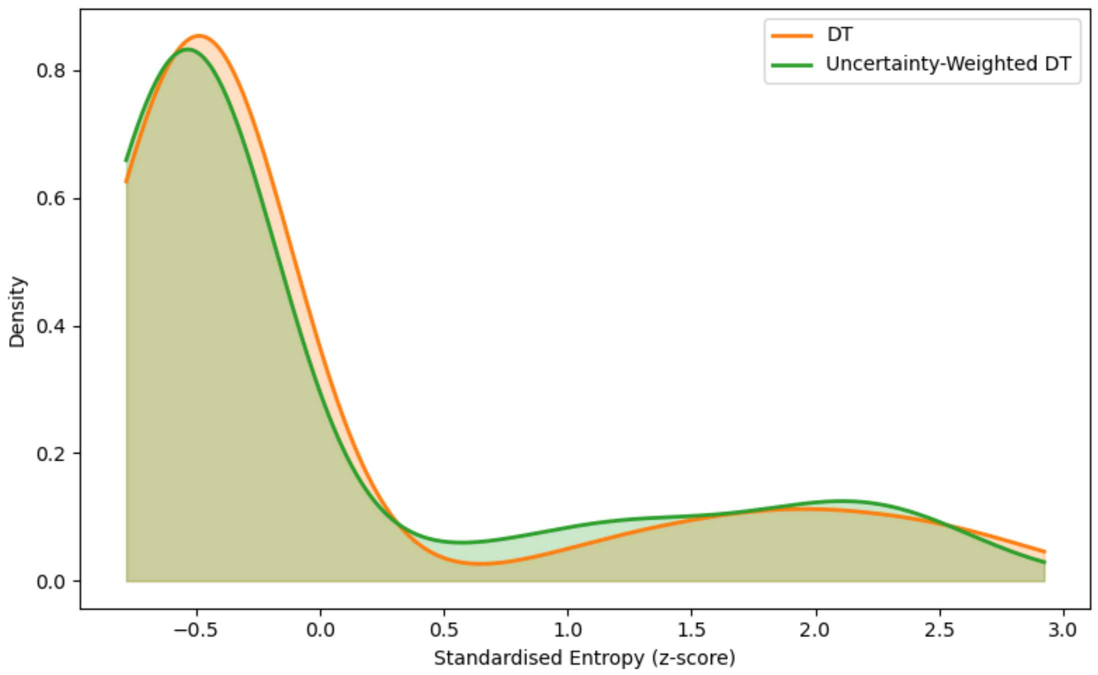


Figure 4.9: The Entropy distribution comparison of DT and UWDT. The number of interacting vehicles is 4, and the results are aggregated over 100 episodes.

4.3.3 Single Epoch and Average Performance at Varying Interacting Vehicle Densities

We first consider cases involving 0-2 interacting vehicles. Figure 4.10 shows that BC Transformer maintains a average reward of of 0.9, whereas CQL collapses to 0 within the first few steps. UWDT, DT, and SAC saturate at the maximum reward throughout the horizon. The velocity profiles in Figure 4.11 reveal the underlying behavior: BC Transformer cruises steadily at roughly 8 m/s; UWDT, DT, and SAC accelerate smoothly from 13 to the 16 m/s speed limit; CQL stalls after an early collision. Because no vehicle must yield, all three high-performing agents merge without braking and achieve near-optimal performance.

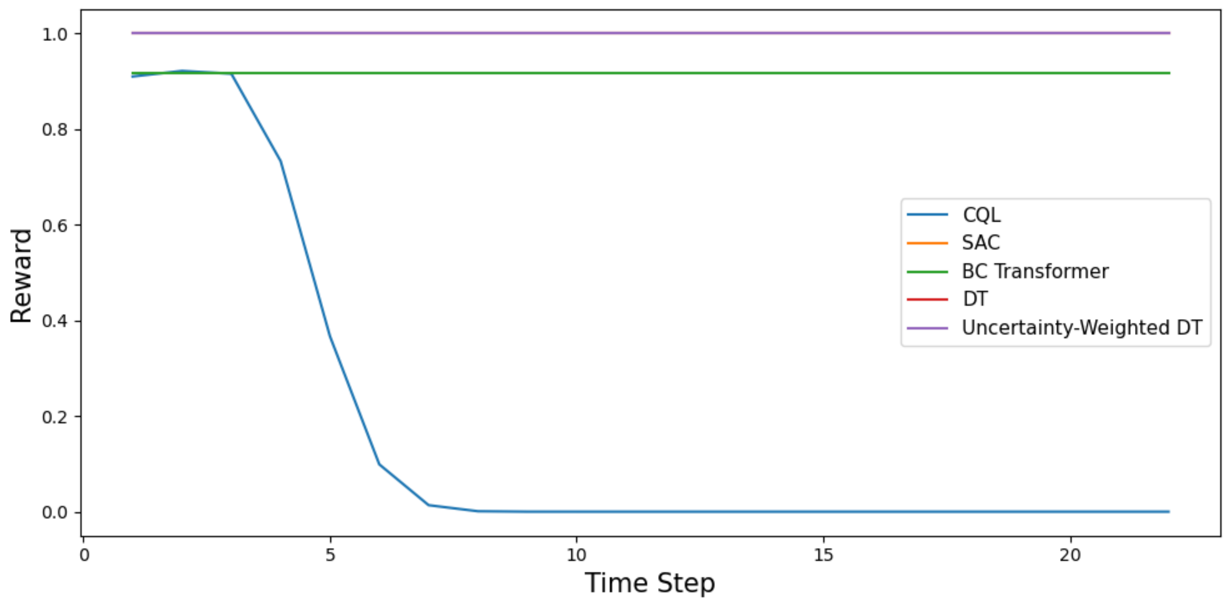
With one additional interacting vehicle, the task becomes substantially harder. In Figure 4.12 the rewards of BC Transformer and CQL drop sharply, indicating frequent collisions. SAC stabilises around 0.8, while UWDT and DT recover to near-optimal reward after a brief transient. Figure 4.13 confirms that UWDT and DT reach the speed limit almost immediately and maintain it. SAC exhibits a full stop midway, waiting for a conservative gap before merging; once merged, it accelerates and finishes strongly. CQL decelerates to around 2.5 m/s and seldom recovers, matching its poor reward.

At the highest density of interacting vehicles, shown in Figure 4.14, BC Transformer deteriorates fastest, losing all reward within three steps. CQL starts near 0.85 but slips to 0.2 by the end of the horizon as collisions accumulate. SAC falls to zero reward during a prolonged wait, then merges safely and finishes with a reward value around 0.9. This conservative policy avoids crashes but is highly inefficient: the ego vehicle remains stationary even when sizable gaps emerge, degrading traffic throughput and increasing travel time. UWDT and DT again dominate, sustaining nearly perfect rewards despite the crowded roundabout. Velocity traces in Figure 4.15 corroborate these findings. BC Transformer decelerates to rest almost instantly. CQL's speed declines from 8 to 4 m/s as its reward vanishes. SAC's speed profile shows a complete stop followed by a late acceleration to 4.5 m/s. UWDT and DT accelerate briskly to the 16 m/s limit and reamins at that speed. Across all densities, UWDT outperforms DT by incurring fewer minor contacts, yielding slightly higher cumulative rewards.

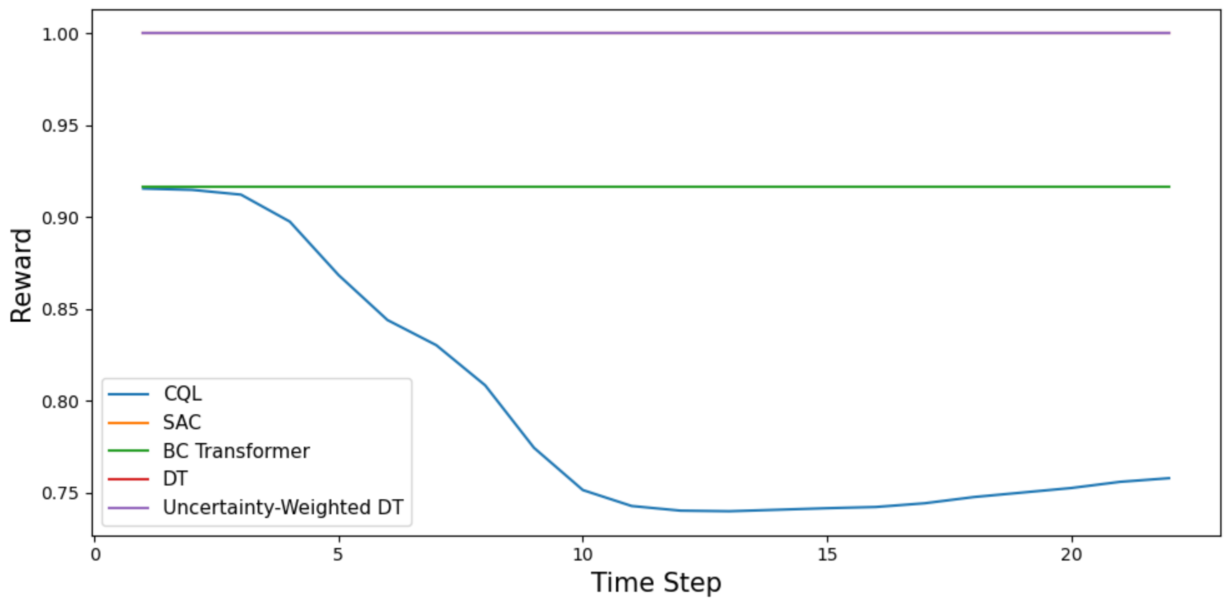
For every density we report single-episode traces and the mean over five independent episodes.

4.3.4 Summary of Results

Overall, these results demonstrate that our UWDT consistently outperforms conventional RL algorithms and imitation learning baselines across varying traffic densities. By leveraging a transformer architecture with uncertainty-weighted return conditioning, the UWDT maintains near-optimal rewards and accelerates to the speed limit even in congested scenarios.

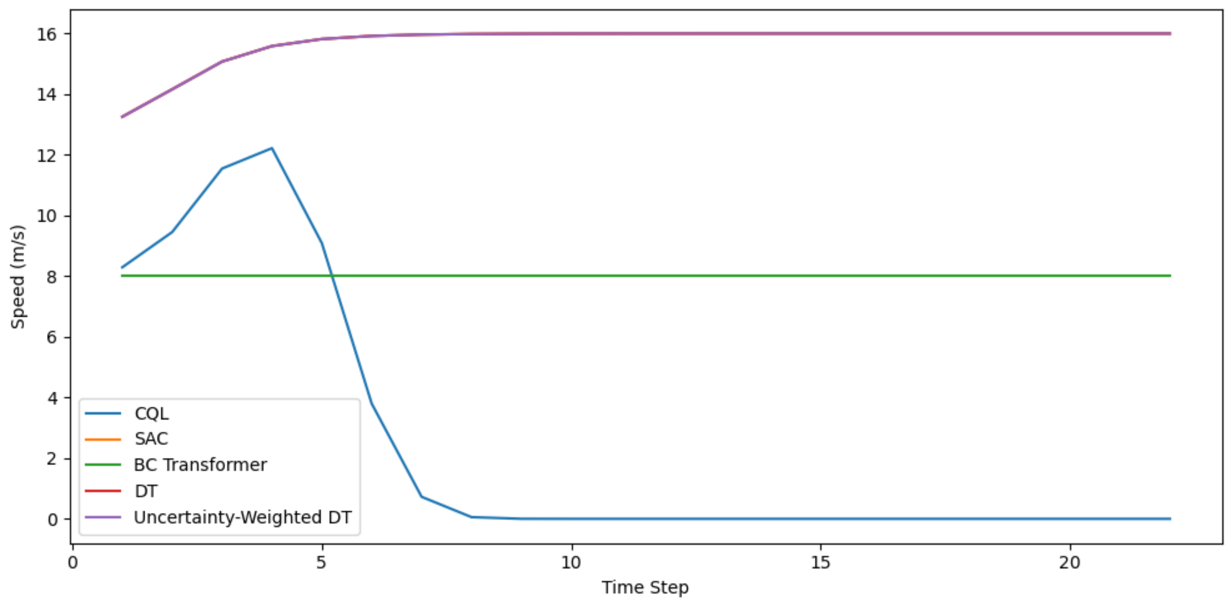


(a) Single episode reward profile with interacting vehicle count uniformly drawn from $[0, 2]$.

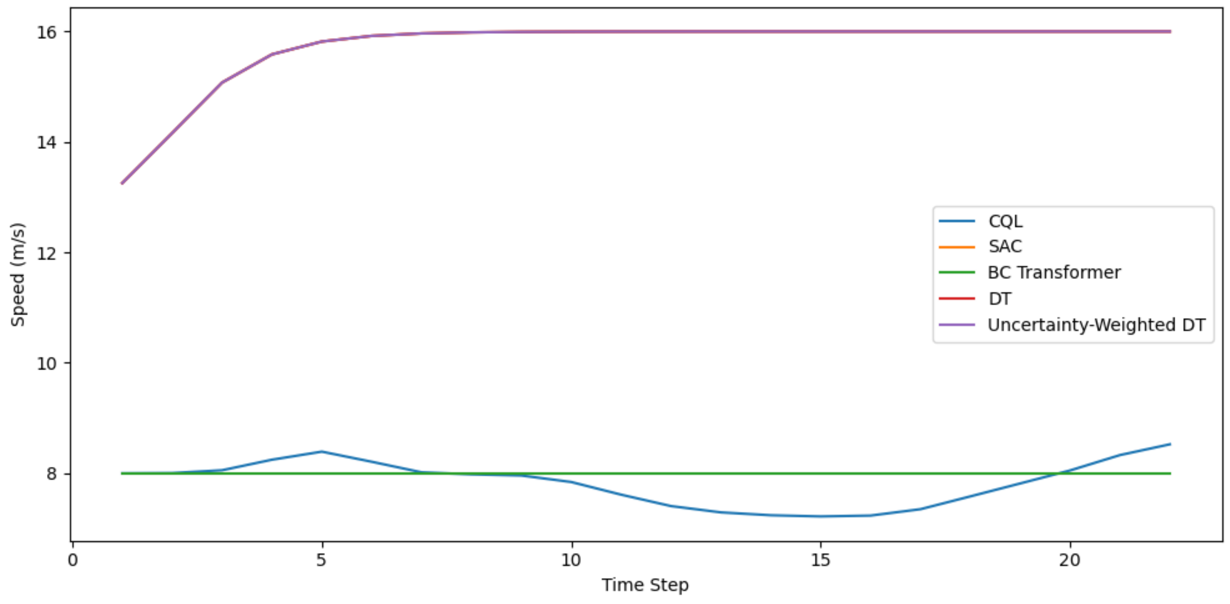


(b) Average reward profile over 20 episodes with interacting-vehicle count uniformly drawn from $[0, 2]$.

Figure 4.10: Normalized reward with at most two incoming vehicles. UWDT consistently attains the highest return while CQL suffers severe degradation.

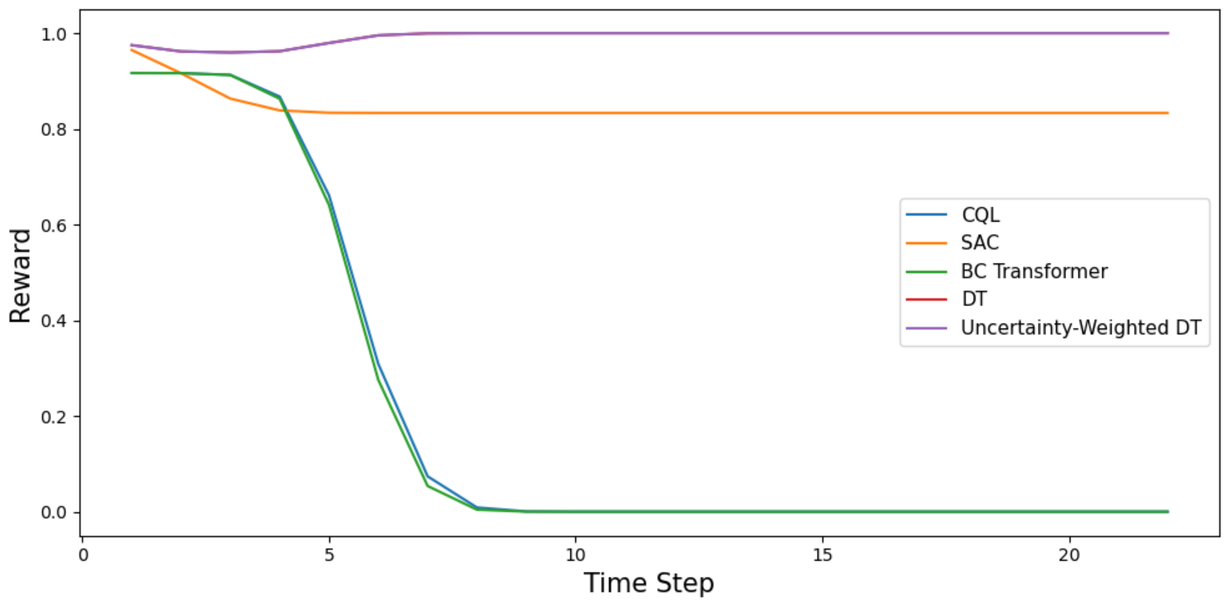


(a) Single episode velocity profile with interacting vehicle count uniformly drawn from $[0, 2]$.

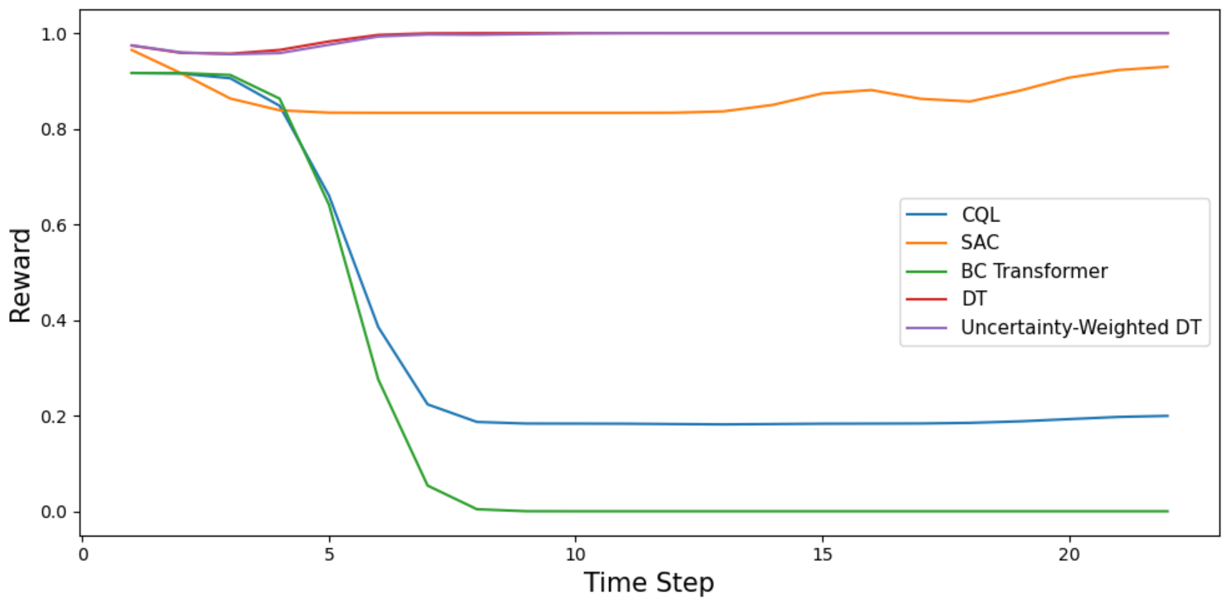


(b) Average velocity profile over 20 episodes with interacting vehicle count uniformly drawn from $[0, 2]$.

Figure 4.11: Velocity profiles with at most two incoming vehicles. UWDT and DT quickly reach the speed limit, whereas CQL stalls and SAC shows moderate recovery when averaging across episodes.

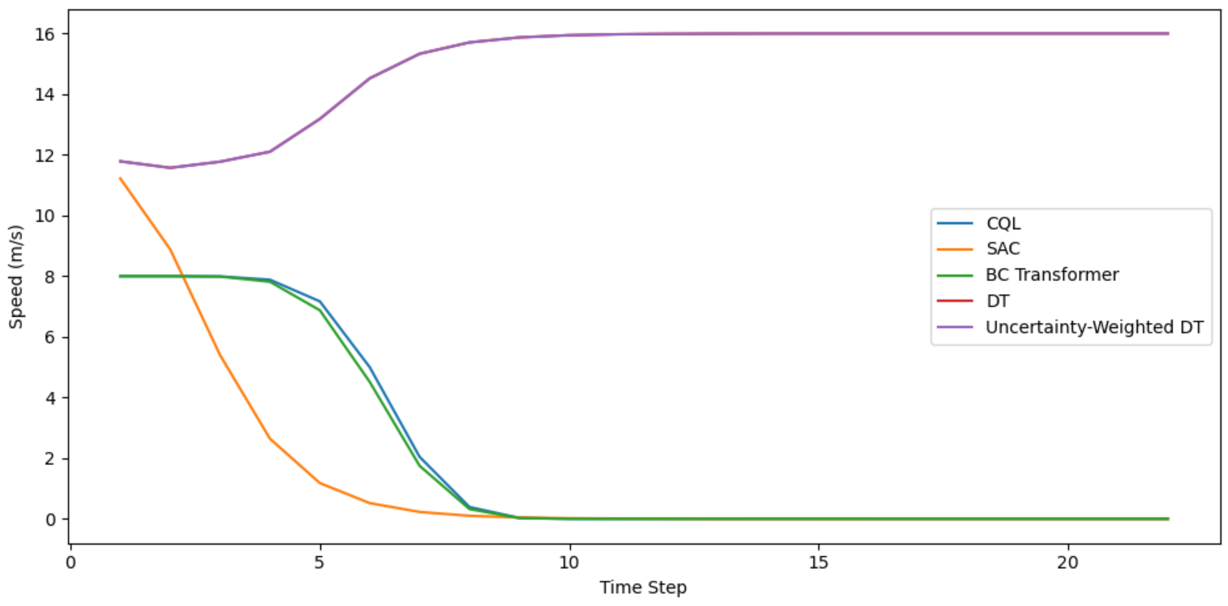


(a) Single episode reward profile with three interacting vehiclesz

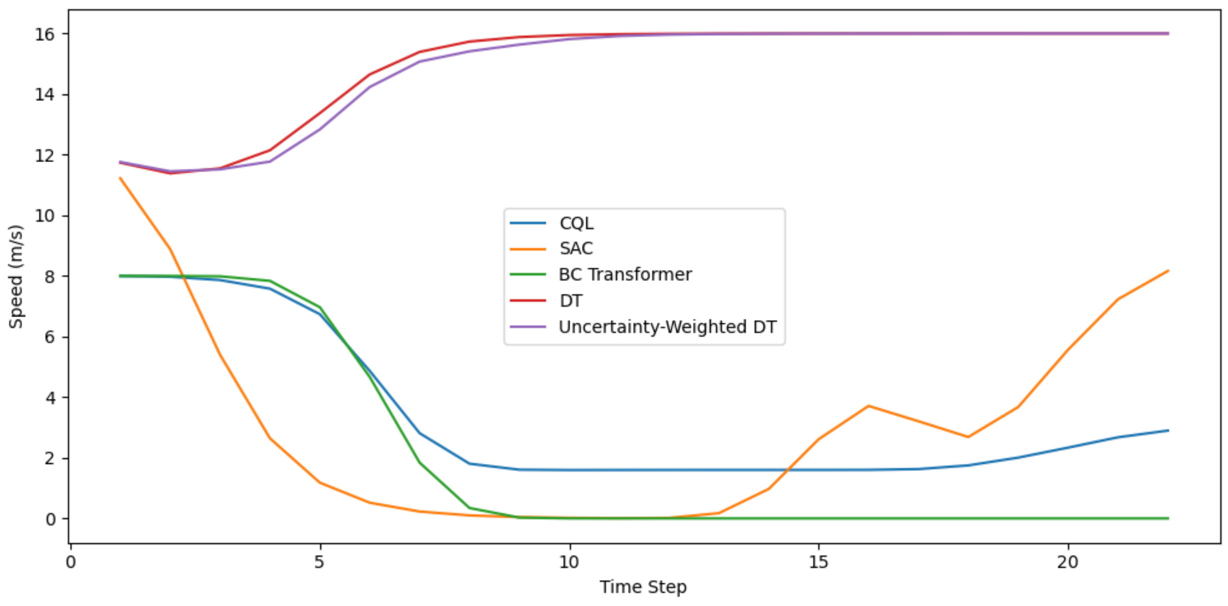


(b) Average reward profile over 20 episodes with three interacting vehicles.

Figure 4.12: Reward profiles with three incoming vehicles. BC and CQL collapse under heavy traffic, while UWDT and DT maintain near-optimal rewards. SAC shows gradual improvement over multiple episodes.

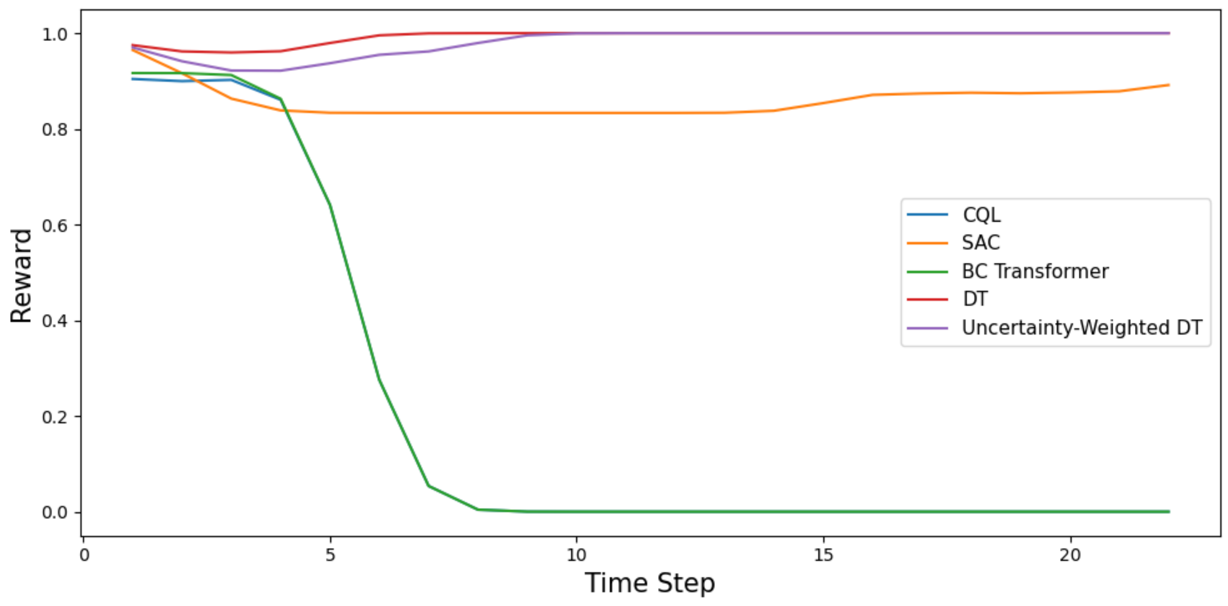


(a) Single episode velocity profile with three interacting vehicles.

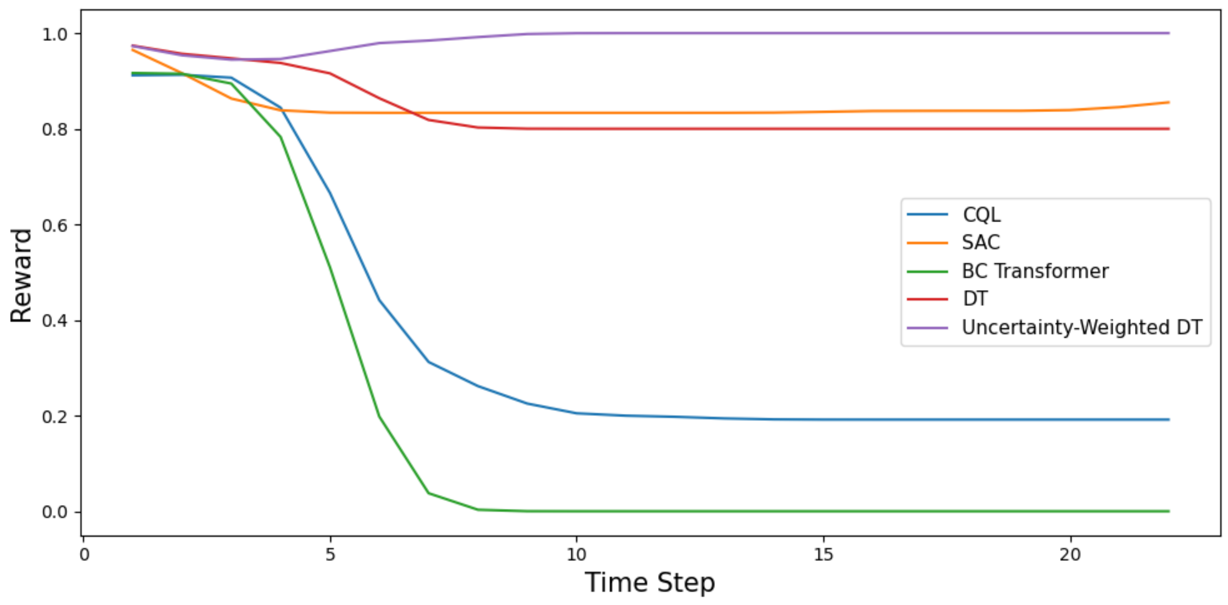


(b) Average velocity profile over 20 episodes with three interacting vehicles.

Figure 4.13: Velocity profiles with three incoming vehicles. UWDT and DT rapidly achieve maximum speed, CQL and BC decelerate to rest, and SAC recovers after an initial decline.

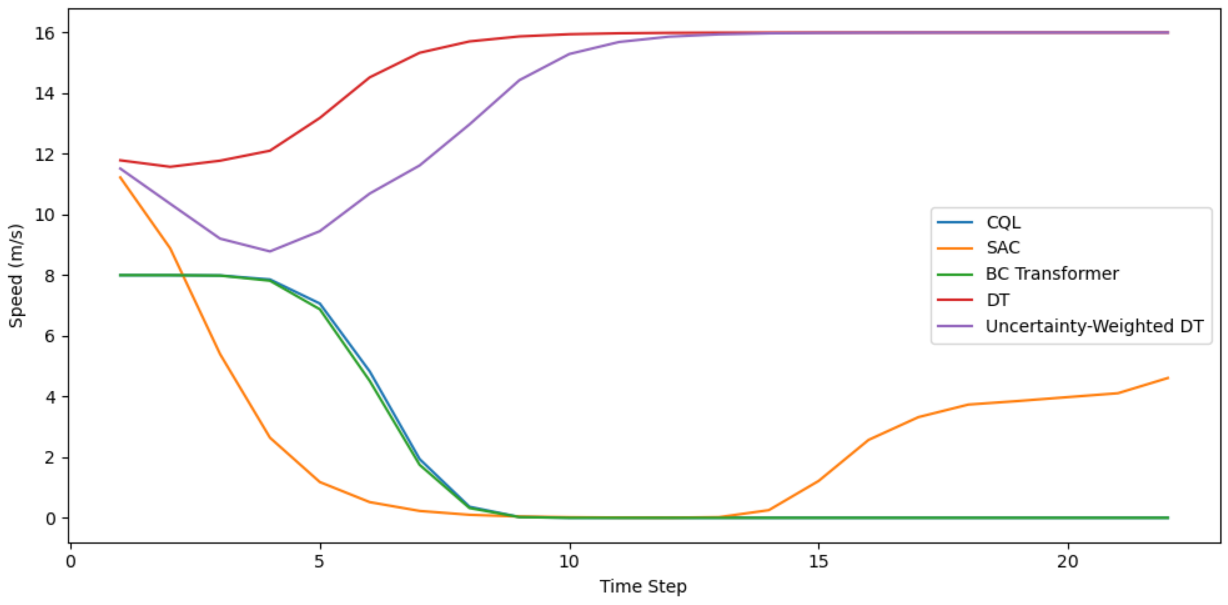


(a) Single episode reward profile with four interacting vehicles.

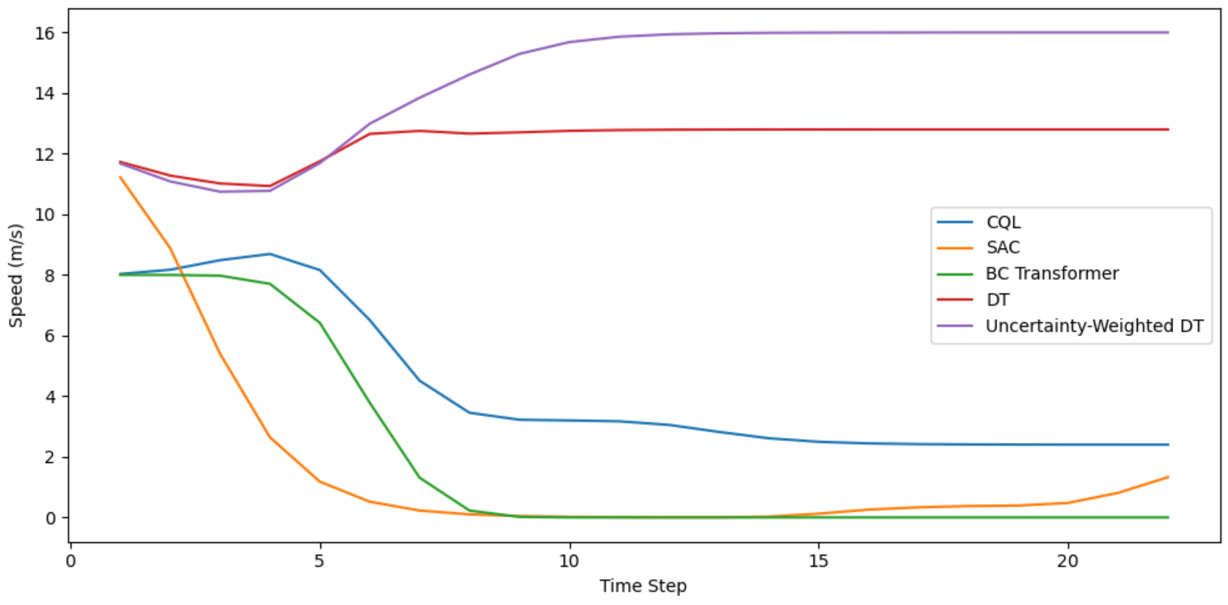


(b) Average reward profile over 20 episodes with four interacting vehicles.

Figure 4.14: Reward profiles with four incoming vehicles. BC and CQL collapse under heavy traffic, while UWDT and DT maintain near-optimal rewards. SAC shows gradual improvement over multiple episodes.



(a) Single episode velocity profile with four interacting vehicles.



(b) Average velocity profile over 20 episodes with four interacting vehicles.

Figure 4.15: Velocity profiles with four incoming vehicles. UWDT and DT rapidly achieve maximum speed, CQL and BC decelerate to rest, and SAC recovers after an initial decline.

Chapter 5

Conclusion

Our study focuses on autonomous driving in complex and dense traffic scenarios. We introduce a quantitative complexity assessment for roundabout navigation that combines interaction density, required MCTS budget, decision latency, and model uncertainty from both behavior cloning and Decision Transformers. We also show that occupancy grids are an effective state representation in dense, heterogeneous traffic. A CNN-based spatial encoder paired with a temporal transformer backbone preserves local geometry and long-horizon dependencies.

We developed and presented the Uncertainty Weighted Decision Transformer for autonomous navigation, which augments the standard transformer architecture with a principled mechanism for quantifying and exploiting epistemic uncertainty at inference time. Unlike purely BC baselines or conservative offline RL methods, UWDT conditions its action selection on both desired return and an uncertainty measure, allowing it to choose high-reward trajectories when confident and adopt safer maneuvers when its predictions are less certain. Through extensive simulation experiments on a complex roundabout driving task, we demonstrated that UWDT achieves higher accumulated rewards, longer travel distances and shorter traversal times, and near-optimal exit success rates while simultaneously reducing collision rates to the lowest value when compared to other tested methods including baseline Decision Transformers and Behavior Cloning (BC) Transformers and conventional deep reinforcement learning agents such as Soft Actor Critic (SAC) and Conservative Q-Learning (CQL).

These results underscore the importance of incorporating uncertainty into sequential decision making, especially when critical situations are rare in the training set but have significant impact during deployment. UWDT improves safety and efficiency, making it a promising approach for safety critical driving applications.

Bibliography

[1] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, "Explaining how a deep neural network trained with end-to-end learning steers a car," *arXiv preprint arXiv:1704.07911*, 2017.

[2] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[3] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58443–58469, 2020.

[4] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

[5] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[6] P. G. Gipps, "A behavioural car-following model for computer simulation," *Transportation research part B: methodological*, vol. 15, no. 2, pp. 105–111, 1981.

[7] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *International conference on computers and games*, pp. 72–83, Springer, 2006.

[8] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*, pp. 282–293, Springer, 2006.

[9] P.-A. Coquelin and R. Munos, "Bandit algorithms for tree search," *arXiv preprint cs/0703062*, 2007.

[10] J.-F. Hren and R. Munos, "Optimistic planning of deterministic systems," in *European Workshop on Reinforcement Learning*, pp. 151–164, Springer, 2008.

[11] Z. Cao, D. Yang, S. Xu, H. Peng, B. Li, S. Feng, and D. Zhao, "Highway exiting planner for automated vehicles using reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 990–1000, 2020.

[12] C.-J. Hoel, K. Driggs-Campbell, K. Wolff, L. Laine, and M. J. Kochenderfer, "Combining planning and deep reinforcement learning in tactical decision making for autonomous driving," *IEEE transactions on intelligent vehicles*, vol. 5, no. 2, pp. 294–305, 2019.

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

[14] J. Hawke, R. Shen, C. Gurau, S. Sharma, D. Reda, N. Nikolov, P. Mazur, S. Micklethwaite, N. Griffiths, A. Shah, *et al.*, "Urban driving with conditional imitation learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 251–257, IEEE, 2020.

[15] Z. Zhang, E. Yurtsever, and K. A. Redmill, "Extensive exploration in complex traffic scenarios using hierarchical reinforcement learning," *arXiv preprint arXiv:2501.14992*, 2025.

[16] K. Redmill, Z. Zhang, E. Yurtsever, *et al.*, "Reinforcement learning," 2024.

[17] K. A. Redmill, Z. Zhang, and E. Yurtsever, "Hierarchical decision making and control in rl-based autonomous driving for improved safety in complex traffic scenarios: Extensive exploration in complex traffic scenarios using hierarchical reinforcement learning," 2024.

[18] C. Peng, Z. Zhang, S. Gong, S. Agrawal, K. A. Redmill, and A. Hereid, "Reinforcement learning with data bootstrapping for dynamic subgoal pursuit in humanoid robot navigation," *arXiv preprint arXiv:2506.02206*, 2025.

[19] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in neural information processing systems*, vol. 34, pp. 15084–15097, 2021.

[20] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," *Advances in neural information processing systems*, vol. 34, pp. 1273–1286, 2021.

[21] H.-L. Hsu, A. K. Bozkurt, J. Dong, Q. Gao, V. Tarokh, and M. Pajic, "Steering decision transformers via temporal difference learning," *arXiv preprint*, 2023.

[22] X. Ma and W.-J. Li, "Weighting online decision transformer with episodic memory for offline-to-online reinforcement learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024.

[23] K. Wang, H. Zhao, X. Luo, K. Ren, W. Zhang, and D. Li, "Bootstrapped transformer for offline reinforcement learning," *arXiv preprint arXiv:2206.08569*, 2022.

[24] Z. Liu, Z. Guo, Y. Yao, Z. Cen, W. Yu, T. Zhang, and D. Zhao, "Constrained decision transformer for offline safe reinforcement learning," in *International Conference on Machine Learning*, pp. 21611–21630, PMLR, 2023.

[25] E. Parisotto, F. Song, J. Rae, R. Pascanu, C. Gulcehre, S. Jayakumar, M. Jaderberg, R. L. Kaufman, A. Clark, S. Noury, *et al.*, "Stabilizing transformers for reinforcement learning," in *International conference on machine learning*, pp. 7487–7498, PMLR, 2020.

[26] Q. Zheng, A. Zhang, and A. Grover, "Online decision transformer," in *international conference on machine learning*, pp. 27042–27059, PMLR, 2022.

[27] Z. Xie, Z. Lin, D. Ye, Q. Fu, W. Yang, and S. Li, "Future-conditioned unsupervised pretraining for decision transformer," *arXiv preprint arXiv:2305.16683*, 2023. ICML 2023.

[28] Z. Guo, D. Wang, Q. He, and P. Zhang, "Leveraging logit uncertainty for better knowledge distillation," *Scientific Reports*, vol. 14, no. 1, p. 31249, 2024.

[29] J. Yi, J. Mao, T. Liu, M. Li, H. Gu, H. Zhang, X. Chang, and Y. Wang, "Teaching with uncertainty: Unleashing the potential of knowledge distillation in object detection," *arXiv preprint arXiv:2406.06999*, 2024.

[30] Y. Wu, S. Zhai, N. Srivastava, J. Susskind, J. Zhang, R. Salakhutdinov, and H. Goh, "Uncertainty weighted actor-critic for offline reinforcement learning," *arXiv preprint arXiv:2105.08140*, 2021.

[31] Z. Li, F. Nie, Q. Sun, F. Da, and H. Zhao, "Uncertainty-aware decision transformer for stochastic driving environments," *arXiv preprint arXiv:2309.16397*, 2023.

[32] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 2002.

[33] C. Coué, C. Pradalier, C. Laugier, T. Fraichard, and P. Bessière, "Bayesian occupancy filtering for multitarget tracking: an automotive application," *The International Journal of Robotics Research*, vol. 25, no. 1, pp. 19–30, 2006.

[34] S. Hoermann, P. Henzler, M. Bach, and K. Dietmayer, "Object detection on dynamic occupancy grid maps using deep learning and automatic label generation," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 826–833, IEEE, 2018.

[35] M. Schreiber, S. Hoermann, and K. Dietmayer, "Long-term occupancy grid prediction using recurrent neural networks," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9299–9305, IEEE, 2019.

[36] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11525–11533, 2020.

[37] Y. Ma, T. Wang, X. Bai, H. Yang, Y. Hou, Y. Wang, Y. Qiao, R. Yang, and X. Zhu, "Vision-centric bev perception: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[38] J. Zhang, Y. Ding, and Z. Liu, "Occfusion: Depth estimation free multi-sensor fusion for 3d occupancy prediction," in *Proceedings of the Asian Conference on Computer Vision*, pp. 3587–3604, 2024.

[39] T. Liu, C. Wang, Z. Yin, Z. Mi, X. Xiong, and B. Guo, "Complexity quantification of driving scenarios with dynamic evolution characteristics," *Entropy*, vol. 26, no. 12, p. 1033, 2024.

[40] P. Huang, H. Ding, and H. Chen, "An entropy-based model for quantifying multi-dimensional traffic scenario complexity," *IET Intelligent Transport Systems*, vol. 18, no. 7, pp. 1289–1305, 2024.

[41] Y. Cui, D. Isele, S. Niekum, and K. Fujimura, "Uncertainty-aware data aggregation for deep imitation learning," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 761–767, IEEE, 2019.

[42] E. Leurent, "An environment for autonomous driving decision-making." `https://github.com/eleurent/highway-env`, 2018.

[43] S. Hoermann, M. Bach, and K. Dietmayer, "Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2056–2063, IEEE, 2018.

[44] R. Danescu, F. Oniga, and S. Nedevschi, "Modeling and tracking the driving environment with a particle-based occupancy grid," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1331–1342, 2011.

[45] A. Nègre, L. Rummelhard, and C. Laugier, "Hybrid sampling bayesian occupancy filter," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 1307–1312, IEEE, 2014.

[46] L. Rummelhard, A. Nègre, and C. Laugier, "Conditional monte carlo dense occupancy tracker," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pp. 2485–2490, IEEE, 2015.

[47] E. Leurent and J. Mercat, "Social attention for autonomous decision-making in dense traffic," *arXiv preprint arXiv:1911.12250*, 2019.

[48] X. Wang, Z. Li, J. Alonso-Mora, and M. Wang, "Reachability-based confidence-aware probabilistic collision detection in highway driving," *Engineering*, vol. 33, pp. 90–107, 2024.

[49] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.

[50] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[51] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, Pmlr, 2018.

[52] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in neural information processing systems*, vol. 33, pp. 1179–1191, 2020.

[53] P. Christodoulou, "Soft actor-critic for discrete action settings," *arXiv preprint arXiv:1910.07207*, 2019.

[54] S. Bubeck and R. Munos, "Open loop optimistic planning," in *COLT 2010-The 23rd Conference on Learning Theory*, 2010.

[55] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," *Advances in neural information processing systems*, vol. 28, 2015.

[56] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[57] L. A. Rodegerdts, *Roundabouts: An informational guide*, vol. 672. Transportation Research Board, 2010.

[58] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[59] M. Yeung, E. Sala, C.-B. Schönlieb, and L. Rundo, "Unified focal loss: Generalising dice and cross entropy-based losses to handle class imbalanced medical image segmentation," *Computerized Medical Imaging and Graphics*, vol. 95, p. 102026, 2022.

[60] L. Chen, Y. Wang, J. Yang, Y. Zheng, T. Han, B. Zhang, and T. Cao, “Uncertainty-aware focal loss for object segmentation,” *Engineering Applications of Artificial Intelligence*, vol. 149, p. 110599, 2025.

[61] Z. S. Baltaci, K. Oksuz, S. Kuzucu, K. Tezoren, B. K. Konar, A. Ozkan, E. Akbas, and S. Kalkan, “Class uncertainty: A measure to mitigate class imbalance,” *arXiv preprint arXiv:2311.14090*, 2023.

[62] X. Huang, D. Batra, A. Rai, and A. Szot, “Skill transformer: A monolithic policy for mobile manipulation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10852–10862, 2023.