Instant Inverse Modeling of Stochastic Driving Behavior with Deep Reinforcement Learning

Dongsu Lee, *Student Member, IEEE,* Minhae Kwon, *Senior Member, IEEE*

***Abstract*—The rapid advancement in vehicular edge computing technology has enabled the implementation of artificial intelligence computation at the edge of intelligent transportation systems. An essential function of vehicular edge computing is its ability to estimate individual drivers' behavioral preferences for enhancing adaptive decision-making processes. This paper explores the potential of instant inverse modeling for autonomous vehicles, aiming to estimate the behavioral preferences of surrounding vehicles. To represent different behavioral preferences, we introduce a stochastic character in the modeling process of a partially observable reinforcement learning (RL) agent. Our primary purpose is to construct an instant inference network (IIN) capable of promptly estimating character parameters upon receiving behavioral data from a target vehicle. Specifically, the proposed solution first trains a character-aware meta-agent embedding a universal policy generalized across character space. We employ this agent to collect the character-conditioned trajectory dataset, and then train the IIN by leveraging the collected trajectory dataset. To evaluate the performance, we simulate a heterogeneous traffic scenario where vehicles with heterogeneous characters share the road. The simulation results demonstrate that the proposed solution achieves superior inference accuracy compared to benchmarking algorithms even with a shorter inference time. We also provide in-depth analyses regarding interpretability, scalability, robustness, and stability.**

***Index Terms*—Autonomous Driving, Stochastic Reward, Reinforcement Learning, Inverse Modeling, Parameter Inference**

I. Introduction

As vehicular edge computing technology revolutionizes intelligent transport systems, research on autonomous driving systems has gained significant momentum. Autonomous vehicles must be equipped not only with driving control functionality but also with data processing technology that extracts valuable information from observed data [1]–[5]. One critical function at the vehicle edge is the instant inference of surrounding drivers' behavioral preferences [6], [7]. For instance, consider the scenario where autonomous vehicles and human drivers coexist. Human drivers exhibit various driving preferences; some prioritize speed, while others prioritize safety, ensuring a safe distance from nearby vehicles [8], [9]. In such mixed-autonomous traffic, the ability to infer human drivers' preferences accurately and instantaneously becomes paramount for autonomous vehicles to determine their subsequent actions adaptively [10]–[12].

Inverse modeling of drivers' behavioral preferences aims to extract character parameters from their behavioral trajectories, representing the underlying factors influencing their behavior on the road. This task can be approached from two perspectives: forward and inverse directions. In the forward direction, character parameters influence the behavior of an agent at a given observation, constituting a conventional RL problem [13]–[15]. Specifically, we define *character* parameters as a weight vector of the reward function, from which individual behavioral patterns emerge. Conversely, the inverse direction involves an inference task estimating the character parameter from a target agent's behavioral trajectory [16], [17]. Performing the inverse direction inference task requires pairing character parameters with behavioral trajectories, making the forward model a prerequisite.

Although there have been notable efforts in inverse modeling, the process of parameter inference from behavioral trajectories has demonstrated itself to be computationally intensive and time-consuming [18]–[20]. Previous works have employed the hierarchical model [21], Bayesian computation [18], and statistical optimization techniques [19], e.g., least-square and log-likelihood estimations. These models estimate the parameters that replicate the behavioral trajectory, closely mimicking the observed behavioral trajectory through numerous iterations. Furthermore, these approaches lack scalability since they require separate procedures for accommodating new trajectory data from other agents. Next, the above inference approaches have focused on deterministic character parameters, but recent neuroscience studies have revealed that character exhibits *stochasticity* due to epistemic and aleatoric uncertainties [22], [23]. To comprehend the character of humans, the inference model must account for its inherent variability, which originates from the internal reward function. To address these issues, this work proposes a novel framework that builds a practical inference network in real-world applications by leveraging both forward and inverse tasks.

In this study, we train two different types of agents: (1) a *partially observable Markov decision process (POMDP) agent* that embeds a policy with a unique stochastic character, and (2) a *character-aware meta-agent* that comprehends all stochastic characters. During forward simulation, a character-aware meta-agent collects behavioral trajectories from a target vehicle. These datasets contain paring information between characters and trajectories. Subsequently, the inverse modeling process develops an instant inference model using these datasets.

This research was supported in part by the National Research Foundation of Korea (NRF) grant (RS-2023-00278812), and in part by the Institute of Information & communications Technology Planning & Evaluation (IITP) grants (No. 2021-0-00739, 2022-2020-0-01602) funded by the Korea government (MSIT). D. Lee is grateful for financial support from Hyundai Motor Chung Mong-Koo Foundation.

D. Lee and M. Kwon are with the Department of Intelligent Semiconductors, and M. Kwon is also with the School of Electronic Engineering, Soongsil University, Seoul, Republic of Korea (e-mail: movementwater@soongsil.ac.kr, minhae@ssu.ac.kr).(Corresponding author: M. Kwon)

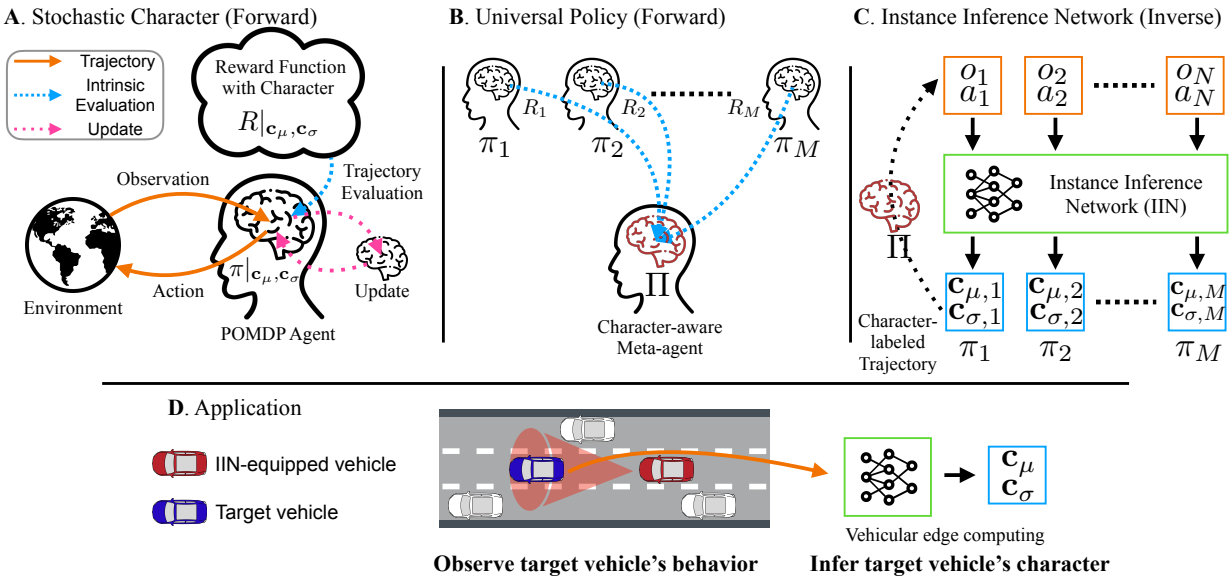


Figure 1: **A**. The policy π of a POMDP agent captures stochastic character $\mathbf{c}_\mu, \mathbf{c}_\sigma$ by using its own reward function R. **B**. The universal policy $\Pi = \{\pi_1, \pi_2, \cdots, \pi_N\}$ comprises all policies based on each character. **C**. The IIN's input and output are observation-action pair and stochastic character, respectively. The universal policy can collect the character-labeled trajectory data. **D**. An application scenario is provided. As the IIN-equipped vehicle collects the target vehicle's behavioral trajectory, the IIN returns the inferred character parameters.

To build the POMDP agent, we formalize a RL problem under partially observable conditions (Figure 1**A**). The character is determined by weights that are drawn from a distribution and assigned to multiple reward components. The trained policy exhibits decision preferences aligned with the given stochastic character. Subsequently, we train the character-aware meta-agent with a universal policy, generalizing over a wide range of character spaces (Figure 1**B**). Next, we build an inference network that instantly infers the stochastic character using an observation-action pair as input (Figure 1**C**). The inference network is based on neural network architecture, and the training dataset can be collected by the character-aware meta-agent with a universal policy. In an application scenario, the IIN-equipped vehicle collects the behavior of a target agent and infers its stochastic character parameters (Figure 1**D**).

The main contributions of this study are summarized as follows.

- We introduce a stochastic agent model to capture stochastic character under a POMDP (Section IV).
- We propose a novel workflow to build an instant inference model by utilizing a character-aware meta-agent with a universal policy, generalizing across character space (Section V).
- The inference task in our framework offers the interpretability of an inferred character regarding the behavioral pattern of the target agent (Section VI-D1).
- The IIN has scalability that can be applied to inference tasks of the stochastic character based on various distribution models and exhibits robustness about environmental noise added in observing the trajectory of the target agent (Section VI-D3 & VI-D2).
- The IIN has stability, which can maintain inference performance over a prolonged operation without significant fluctuations (Secion VI-D4).

In summary, the proposed framework enables the autonomous vehicle to understand the behavioral patterns or preferences of target vehicles with which they interact.[1]

The structure of this paper can be outlined as follows: Section II offers an extensive review of the relevant literature. Section III formulates forward and inverse problems with RL. The subsequent two sections introduce the proposed framework for building an instance inference network. More precisely, Section IV introduces a RL approach for modeling stochastic character agents, while Section V provides the instance inference training approach. Section VI discusses the effectiveness and performance of our algorithm, supported by comprehensive simulation results. The paper is concluded in Section VII, where the findings and contributions are summarized. Notations are summarized in Appendix A.

II. Related Works

A. Parameter Inference for Inverse Modeling

In the realm of autonomous driving research, interactions between autonomous vehicles and human-driven vehicles necessitate the development of artificial intelligence capable of inferring a decision model of a target vehicle. This decision model entails understanding driving characteristics [24]. This ability can be cultivated through inverse modeling, a process that involves grasping parameters relevant to the researcher's

[1]Note that the study for mixed autonomous traffic can be divided into two folds: 1) Understanding neighbor human vehicles' driving characteristics and 2) Making adaptive decisions for autonomous vehicle by integrating additional information regarding surroundings. The scope of this work is limited to the first division.

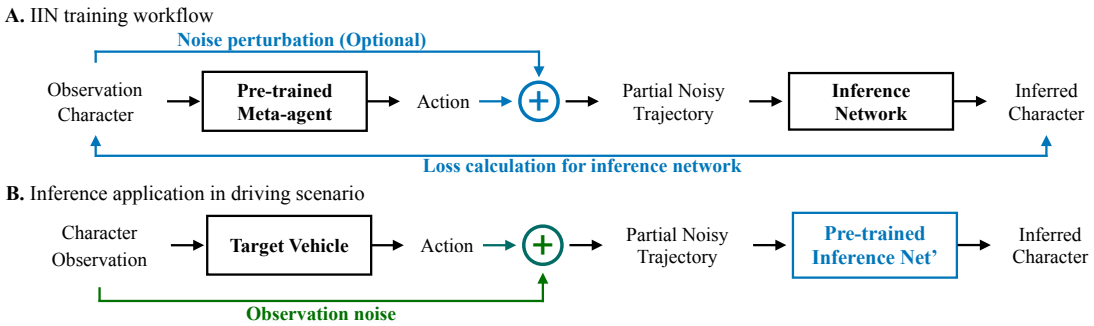


Figure 2: Proposed workflows. Given a character and observation, an agent decides the action. This observation-action trajectory with an observing noise is an input of the IIN, and the character is an output.

interests. These parameters represent latent attributes associated with the target's behavior [11]. By manipulating these parameters, the inference system can gauge the discrepancy between observed and predicted behaviors. Previous studies have framed the inverse modeling problem as optimization problems with maximum likelihood or least square estimation [19], [20].

While these methods excel at parameter inference, they come with significant computational costs due to many iterations and the requirement for individual inverse modeling for different candidates. This work aims to build an instant inference model capable of estimating the parameters across different target vehicles, endowing it into autonomous driving systems to enhance its practicality.

B. Reinforcement Learning with Theory of Mind

The theory of mind refers to the ability to understand others by surmising what is happening in their minds. Machine theory of mind, a research topic for transferring these cognitive concepts to artificial machines, is currently being studied [25]–[27]. Its alignment with RL, which focuses on decision-making based on observed interactions, holds promise for enhancing an autonomous driving vehicle's ability to comprehend and interact with human-driving vehicles.

Inverse RL is a representative method for inferring the behavioral pattern of a target agent. The primary objective of the inverse RL is to find a reward function [28]. Previous works use linear and quadratic programming [29] or the maximum entropy [30] to learn mappings from trajectory to a reward function. However, these methods cannot apply to realistic scenarios with partially observable and continuous nonlinear dynamics. To alleviate this problem, [19] and [20] have proposed inverse rational control (IRC) that expands the solution in partially observable settings. Furthermore, [31] combines IRL and RL solutions to extract the policy using an inferred reward function. However, this solution cannot provide an explanation of behavioral patterns since the reward function in this solution is intangible.

Although these efforts successfully develop inverse modeling methods, they are based on iterative optimization, so it is still challenging to achieve real-time inference. This work aims to build a real-time inference model that can promptly provide accurate inference results.

C. Trajectory Prediction with Driving Character

An autonomous driving system should make adaptive decisions in the complicated and dynamic traffic environment where many vehicles coexist. To make adaptive decisions, understanding intentions and predicting trajectories of surrounding vehicles are essential [32], [33]. Trajectory prediction methodology in previous studies can be broadly categorized into three approaches [34]: physics [35], [36], statistics [37], and deep learning [38]–[40]. The physics-based approach predicts future trajectories based on the laws of physics with given data, offering high explainability but lower performance due to uncertainty in real-world scenarios. Next, the statistic-based approach elucidates trajectories by leveraging pre-defined maneuvers, e.g., Hidden Markov model [41], Bayesian model [42], and support vector machine [43]. This outperforms the physics-based one, but it faces challenges in real-time applications due to its high computational cost. Recent extensive research has focused on deep learning-based prediction as a deep data-driven approach. Examples include recurrent neural networks [38], spatial-temporal graph representation [39], [40], and attention-based architecture [44].

Although this approach improves the trajectory prediction accuracy, it often fails because they do not include a comprehension process about the intentions and behavioral preferences of surrounding vehicles. This work proposes an inference approach to driving character that reflects individual behavioral preferences. An inferred driving character can be easily used as additional information to predict the trajectory for downstream tasks.

III. Problem Formulation

This work aims to integrate an instant inference model into autonomous driving systems, enhancing their capability through vehicular edge computing functionality. This section involves formulating both forward and inverse problems in RL-based agent modeling. In the forward process, the focus lies on training a *character-aware meta-agent* with a *universal policy* Π that encompasses all stochastic characters $\mathbf{c}$. We collect character-conditioned behavioral trajectories using the character-aware meta-agent. Subsequently, the inverse task involves constructing an instant inference model using the data obtained during the forward process. Once trained, this inference model can seamlessly integrate into autonomous

driving systems, estimating the driving character of surrounding vehicles.

Figure 2 summarizes the proposed IIN training and inference processes. During training (Figure 2**A**), we first pre-train a character-aware meta-agent. Using the pre-trained character-aware meta-agent, the partial and noisy trajectories are generated and used in IIN training. In the inference process (Figure 2**B**), an autonomous driving system equips the fully trained IIN, and it collects the trajectory of the target vehicle. Then, the IIN promptly returns the inferred character parameters.

A. Partially Observable Markov Decision Process with Character

The forward agent modeling problem can be formulated as a POMDP, which is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, \Omega, R, \gamma \rangle$ that comprises of a state $s_t \in \mathcal{S}$, an observation $o_t \in \mathcal{O}$, an action $a_t \in \mathcal{A}$, a state transition probability $\mathcal{T}(s_{t+1}|s_t, a_t)$, an observation transition probability $\Omega(o_t|s_t)$, a reward function R, and a temporal discounted factor γ. In detail, the reward r_t can be defined as a linear combination of n-th element of character vector c_n and reward component $\mathcal{R}_n(s_t, a_t, s_{t+1})$ as follows[2]:

$$r_t = R(s_t, a_t, s_{t+1}; \mathbf{c}) = \sum_{n=1}^{N} c_n \mathcal{R}_n(s_t, a_t, s_{t+1}), \quad (1)$$

where n-th character component c_n is included in character vector $\mathbf{c} = [c_1, c_2, \cdots, c_N]^\top \in \mathcal{C}$, and $N = |\mathbf{c}|$ denotes the number of components in character vector $\mathbf{c}$.

The goal of the POMDP agent is to find a policy π that can maximize the expected cumulative reward, i.e., $\mathbb{E}_\pi\Big[\sum_t \gamma^t R(s_t, a_t, s_{t+1}; \mathbf{c})\Big]$. The reward maximization strategy can be changed according to the combination of character parameters $\mathbf{c}$, and this difference makes diverse policies. The POMDP agent with a character $\mathbf{c}$ builds the decision-making policy that returns optimal action a_t to the partial observation o_t.

Given this agent modeling problem, we define the POMDP agent and character-aware meta-agent as follows.

- **POMDP agent** has the policy with a single character, i.e., $\pi(a_t|o_t; \mathbf{c})$.
- **Character-aware meta-agent** has a universal policy, reflecting all available characters in character space $\forall \mathbf{c} \in \mathcal{C}$, i.e., $\Pi(a_t|o_t; \mathcal{C})$.

B. Noisy Trajectory Collection

We aim to address the forward and inverse tasks in a noisy environment to consider a realistic scenario. In the deployment phase, an IIN-equipped vehicle infers the driving character of a target agent using the pre-trained IIN. Note that the collected trajectories may include noise since the IIN-equipped vehicle may not be able to get perfect observation toward the target agent.

[2]An example regarding reward component $\mathcal{R}_n(s_t, a_t, s_{t+1})$ are provided in Section VI-A and Appendix D-D.

Specifically, we denote noisy trajectories collected by the IIN-equipped vehicle as $(\tilde{o}_t, \tilde{a}_t)$, whereas the original trajectory of the target agent is (o_t, a_t). Herein, collected observations and actions can be formalized as $\tilde{o}_t \sim \mathcal{N}(\Omega(o_t|s_t), \tilde{\sigma})$ and $\tilde{a}_t \sim \mathcal{N}(a_t, \tilde{\sigma})$, where $\tilde{\sigma}$ denotes the standard deviation of noise that can be included in the IIN-equipped vehicle's observation.

C. Character Inference

The main objective of this work is to inversely find the character parameter $\mathbf{c}$ of the target POMDP agent by observing its trajectory. Since we cannot access the true character distribution of the POMDP agent, we utilize the character-aware meta-agent with the pre-trained universal policy (Section IV-B). The character-aware meta-agent can generate the trajectory data (o_t, a_t) at a given character. Thus, we can obtain character-labeled trajectory data $(o_t, a_t; \mathbf{c})$. After the labeling process, we train a deep neural network to address an optimization problem $\hat{\mathbf{c}} = \arg\min_{\mathbf{c}} \mathcal{L}(\mathbf{c})$, where $\mathcal{L}(\mathbf{c})$ is a loss function regarding the character $\mathbf{c}$.

IV. Forward: Universal Policy with Stochastic Characters

This section aims to build a universal policy for a character-aware meta-agent as a prerequisite for building the proposed IIN. The universal policy encompasses a set of policies over character space $\mathcal{C}$. We first provide a normative definition of the stochastic POMDP agent model with character distribution. Next, we introduce how to capture all the different characters within the character space $\mathcal{C}$ using a single policy.

A. POMDP Agent with Stochastic Character

To define stochastic agent model, we first assume that n-th component of character vector $c_n \in \mathbf{c} = [c_1, c_2, \cdots, c_N]^\top \in \mathcal{C}$ follows character distribution $f(c_n; c_{n,\mu}, c_{n,\sigma}, \mathrm{a}, \mathrm{b})$. Here, $c_{n,\mu} \in \mathbf{c}_\mu = [c_{1,\mu}, c_{2,\mu}, \cdots, c_{N,\mu}]^\top$ denotes the mean of n-th element of character vector and $c_{n,\sigma} \in \mathbf{c}_\sigma = [c_{1,\sigma}, c_{2,\sigma}, \cdots, c_{N,\sigma}]^\top$ denotes standard deviation of n-th element of character vector. Constants a, b finite range of character space. The character distribution is defined as follows.

Definition 1 (Character distribution)**.** *A character distribution is defined as a truncated Gaussian distribution*

$$f(c_n; c_{n,\mu}, c_{n,\sigma}, \mathrm{a}, \mathrm{b}) = \frac{\zeta(\frac{c_n - c_{n,\mu}}{c_{n,\sigma}})}{c_{n,\sigma}(Z(\frac{\mathrm{b} - c_{n,\mu}}{c_{n,\sigma}}) - Z(\frac{\mathrm{a} - c_{n,\mu}}{c_{n,\sigma}}))},$$

where $\zeta(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}x^2)$ *means the probability density function of the Gaussian distribution,* $Z(x) = \frac{1}{2}(1 + \mathrm{erf}(x/\sqrt{2}))$ *is its cumulative distribution function, and character variable* c_n *is constrained to be in the interval* $[\mathrm{a}, \mathrm{b}]$.

We consider the truncated Gaussian distribution as the character distribution to design the character defined in finite character space.[3]

[3]Note that the Gaussian distribution can be set to $a = -\infty$ and $b = +\infty$. It is worth noting that the proposed method is not limited to Gaussian distribution, and it rather works with a broader family of unimodal distributions such as Gamma and Poisson distribution. The empirical results for Gamma and Poisson distribution are provided in Figure 7.

Algorithm 1 Train a Universal Policy

Require: Total episodes K, total timesteps of an episode T

Initialization: Network parameters ϕ, ψ
for episode $k = 1$, K **do**
  Reset s_1 and get $o_1 \sim \Omega(o_1|s_1)$
  Sample character distribution parameters $\mathbf{c}_\mu, \mathbf{c}_\sigma$
  for timestep $t = 1$, T **do**
    Execute $a_t \sim \Pi_\phi(a_t|o_t; \mathbf{c}_\mu, \mathbf{c}_\sigma)$
    Get s_{t+1}, r_t, and o_{t+1}
    Calculate the $Q_\psi^\Pi(\cdot)$
    Update ψ by back-propagating TD-error ϵ
    Update ϕ by back-propagating $-Q_\psi^\Pi$
    $t \leftarrow t + 1$
  end for
end for

As mentioned in Section III-A, the POMDP agent aims to learn an optimal policy at a given character. To capture the stochastic character, we propose a stochastic POMDP agent model as follows.

Definition 2 (Stochastic POMDP agent model)**.** *A stochastic POMDP agent model is defined* $\mathrm{A} = (R, \pi, Q, f)$*, where* $r_t = R(s_t, a_t, s_{t+1}; \mathbf{c}_\mu, \mathbf{c}_\sigma)$ *means a stochastic character reward,* $\pi(a_t|o_t; \mathbf{c}_\mu, \mathbf{c}_\sigma)$ *and* $Q(o_t, a_t; \mathbf{c}_\mu, \mathbf{c}_\sigma)$ *denote a stochastic character policy and Q-function, and* $f(c_n; c_{n,\mu}, c_{n,\sigma}, \mathrm{a}, \mathrm{b})_{\forall n}$ *represents a character distribution.*

The stochastic agent interacts with the environment to learn a stochastic character policy $\pi^*(a_t|o_t; \mathbf{c}_\mu, \mathbf{c}_\sigma)$. The value function $Q(o_t, a_t; \mathbf{c}_\mu, \mathbf{c}_\sigma)$ follows the objective function $\mathbb{E}_\pi\Big[\sum_t \gamma^t R(s_t, a_t, s_{t+1}; \mathbf{c}_\mu, \mathbf{c}_\sigma)\Big]$. We use the neural network for approximation because calculating and maximizing an estimate of Q-function about all combinations of observation and action requires a high cost in the continuous space.

B. Training a Universal Policy

The universal policy $\Pi(a_t|o_t; \mathcal{C})$ includes a set of policies $\{\pi_1(a_t|o_t; \mathbf{c}_\mu, \mathbf{c}_\sigma), \pi_2(a_t|o_t; \mathbf{c}_\mu, \mathbf{c}_\sigma), \cdots, \pi_M(a_t|o_t; \mathbf{c}_\mu, \mathbf{c}_\sigma)\}$ over the character space $\mathcal{C}$, where M means the number of available character. Note that each policy $\pi_m(\cdot)$ has a different character distribution. Once the universal policy is fully trained, the character-aware meta-agent can emulate a behavioral trajectory tailored to a specific character. Based on such capability of the character-aware meta-agent, we secure the character-labeled trajectory dataset for IIN training.

We construct a universal policy for a stochastic character while taking into account the following considerations.

- How can we approximate the policy and Q-function in continuous observation and action spaces?
- How can we efficiently train the critic network that captures stochastic characters without an extensive amount of samples?

1) Actor-critic networks: : We use the deep RL algorithm based on actor-critic networks as a function approximator to estimate the policy and Q-function in continuous space. The actor network ϕ approximates the universal policy Π over the continuous observation and character parameters. The critic-network ψ approximates the Q-function over the continuous observation, character parameters, and action. The update of the critic network is based on a temporal difference error (TD-error), i.e.,

$$\begin{aligned}\epsilon = &- Q_\psi^\Pi(o_t, a_t; \mathbf{c}_\mu, \mathbf{c}_\sigma) + R(s_t, a_t, s_{t+1}; \mathbf{c}_\mu, \mathbf{c}_\sigma) \\ &+ \gamma Q_\psi^\Pi\Big(o_{t+1}, \Pi_\phi(a_{t+1}|o_{t+1}; \mathbf{c}_\mu, \mathbf{c}_\sigma); \mathbf{c}_\mu, \mathbf{c}_\sigma\Big).\end{aligned} \tag{2}$$

The update of actor-network is based on negative Q-function $-Q_\psi^\Pi(\cdot)$.[4] Both networks are trained using the gradient descent method. We provide the pseudocode in Algorithm 1.

2) Expected reward: : To extract a universal policy accommodating the stochasticity of character, the Q-function needs to be exposed to a sufficient number of character samples. This process can lead to a precise approximation of $Q_\psi^\Pi(o_t, a_t; \mathbf{c}_\mu, \mathbf{c}_\sigma)$, however, it requires a large number of iterations. Additionally, in (2), the stochasticity of the reward function $R(s_t, a_t, s_{t+1}; \mathbf{c}_\mu, \mathbf{c}_\sigma)$ would aggravate its computational cost. The character component c_n changes across samples even though a fixed character distribution $\mathbf{c}_\mu, \mathbf{c}_\sigma$ is given. Consequently, different character c_n results in different values of $R(s_t, a_t, s_{t+1}; \mathbf{c}_\mu, \mathbf{c}_\sigma)$ in (2), posing challenges in training the Q-function in terms of sample efficiency.

To efficiently train the Q-function, we employ an expected character value $\bar{\mathbf{c}} = [\bar{c}_1, \bar{c}_2, \cdots, \bar{c}_N]$ when computing reward $R(s_t, a_t, s_{t+1}; \mathbf{c}_\mu, \mathbf{c}_\sigma)$. In other words, we replace $R(s_t, a_t, s_{t+1}; \mathbf{c}_\mu, \mathbf{c}_\sigma)$ with an expected reward $\mathbb{E}_{c_n \sim f(c_n), n \in \{1,2,\cdots,N\}}\left[R(s_t, a_t, s_{t+1}; \mathbf{c}_\mu, \mathbf{c}_\sigma)\right]$ in (2). The derivation is as follows.

$$\begin{aligned}&\mathbb{E}_{c_n \sim f(c_n), n \in \{1,2,\cdots,N\}}\left[R(s_t, a_t, s_{t+1}; \mathbf{c}_\mu, \mathbf{c}_\sigma)\right] \\ &= \mathbb{E}_{c_n \sim f(c_n), n \in \{1,2,\cdots,N\}}\left[\sum_{n=1}^{N} c_n \mathcal{R}_n(s_t, a_t, s_{t+1})\right] \\ &= \sum_{n=1}^{N} \mathcal{R}_n(s_t, a_t, s_{t+1}) \mathbb{E}_{c_n \sim f(c_n)}\left[c_n\right] \\ &= \sum_{n=1}^{N} \mathcal{R}_n(s_t, a_t, s_{t+1}) \int c_n f(c_n; c_{n,\mu}, c_{n,\sigma}, \mathrm{a}, \mathrm{b}) dc_n \\ &= \sum_{n=1}^{N} \mathcal{R}_n(s_t, a_t, s_{t+1}) \bar{c}_n\end{aligned} \tag{3}$$

This method offers advantages in terms of learning efficiency compared to the sampling-based method. Specifically, our approach does not require extensive character sampling to estimate Q-function at a given stochastic character. We consider the reward based on character expectation, which can theoretically be obtained when assuming the agent experiences enough. Therefore, the policy and Q-function do not need to encounter numerous character samples to form the representation of a stochastic character.

V. Inverse: Instance Inference Network

This section investigates how to build the IIN and how to use it. We first provide the loss function to achieve an objective of

[4]To differentiate the POMDP agent's Q-function, we use the policy notation Π as a superscript for the Q-function of the character-aware meta agent.

IIN. Additionally, we introduce the method that the proposed solution applies to inference on deterministic character. Lastly, we delve into application workflow in the deployment phase.

A. Training Instant Inference Network

The IIN aims to be capable of instant inference on the character of others. To implement this functionality, we configure the IIN Θ as neural network architecture. The input of the IIN is the observation-action pair $(\tilde{o}_t, \tilde{a}_t)$, and the output is the mean $\hat{\mathbf{c}}_\mu$ and standard deviation $\hat{\mathbf{c}}_\sigma$ of the stochastic character. This can be formulated as the optimization problem $\hat{\mathbf{c}}_\mu, \hat{\mathbf{c}}_\sigma = \arg\min_{\mathbf{c}_\mu, \mathbf{c}_\sigma} \mathcal{L}(f(\mathbf{c}; \mathbf{c}_\mu, \mathbf{c}_\sigma, \mathrm{a}, \mathrm{b}))$. As a loss function for the IIN training, we employ cross-entropy between the true $f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b})$ and inferred character distribution $f(\mathbf{c}; \hat{\mathbf{c}}_\mu, \hat{\mathbf{c}}_\sigma, \mathrm{a}, \mathrm{b})$.[5] The trajectory length l_{train} of observation-action pair inputs to the IIN, which then outputs $\hat{\mathbf{c}}_\mu$ and $\hat{\mathbf{c}}_\sigma$.

Algorithm 2 presents the pseudocode for training the IIN. The algorithm begins by initializing the IIN parameters. The training of the IIN occurs within a loop iteratively until either the stopping criterion is met or the pre-defined number of iterations is reached. Within each iteration of the loop, the algorithm samples trajectory data from a dataset. This data consists of an observation-action pair associated with a corresponding character. Additionally, noisy observations along the trajectory can be considered by incorporating noise. Upon receiving this data as input, the IIN outputs an estimated character. Subsequently, the algorithm calculates the loss and updates the IIN parameters using the gradient descent method.

B. Special Case: Deterministic Character

Our inference solution considers stochastic character, which includes a deterministic character as a special case. In the following theorem, we demonstrate that the cross-entropy loss function can be simplified as the mean squared error function in this special case.

Theorem 3. *If the true character distribution is the Dirac delta distribution, i.e., $f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b}) = \delta(\mathbf{c} - \mathbf{c}^*_\mu)$, the inferred character distribution satisfies the Gaussian distribution, and $\mathbf{c}_\sigma$ is a constant, then the optimization problem $\hat{\mathbf{c}}_\mu, \hat{\mathbf{c}}_\sigma = \arg\min_{\mathbf{c}_\mu, \mathbf{c}_\sigma} \mathcal{L}(f(\mathbf{c}; \mathbf{c}_\mu, \mathbf{c}_\sigma, \mathrm{a}, \mathrm{b}))$ can be simplified as $\hat{\mathbf{c}}_\mu = \arg\min_{\mathbf{c}_\mu} \frac{1}{N} \sum_{n=1}^{N} (c^*_{n,\mu} - c_{n,\mu})^2$.*

Proof. See Appendix C. □

Remark. *If a POMDP agent with deterministic character is considered, the loss function of IIN (i.e., cross-entropy) can be replaced by mean squared error.*

C. Character Inference Using Instant Inference Network

Once the IIN is fully trained across character space, the IIN-equipped agent can infer the character of a target agent. The IIN-equipped agent observes the trajectory of the target agent, and infers a character, thereby updating the inferred character over a period. Specifically, the IIN-equipped agent estimates the character of the target agent at every timestep.

We present the pseudocode outlining the inference process of IIN in Algorithm 3. During the deployment phase, the IIN-equipped vehicle with the IIN collects the observation-action trajectory of a target vehicle, which contains observational noise. Utilizing the IIN and collected data, the IIN-equipped vehicle iteratively infers the character of the target vehicle. This iterative process allows for continual refinement of the estimated characteristics over time. Character updates are performed through an aggregation and averaging approach.

[5]The derivation for the cross-entropy loss function is provided in Property 4 of Appendix B

Algorithm 2 IIN: Training Phase

Require: IIN $\Theta(\cdot)$, character-aware meta-agent's observation noise standard deviation $\tilde{\sigma}_{train}$, trajectory length l_{train}, dataset $\bar{\mathcal{D}}$, learning rate α, and iterations I

Ensure:

Initialize IIN parameters
while Not stop criterion **do**
  Sample trajectory data by the length of l_{train}
  $$(o_t, a_t; \mathbf{c}_\mu, \mathbf{c}_\sigma) \sim \bar{\mathcal{D}}$$
  Add noise on trajectory data
  $\tilde{o}_t \sim \mathcal{N}(\Omega(o_t|s_t), \tilde{\sigma}_{train})$ and $\tilde{a}_t \sim \mathcal{N}(a_t, \tilde{\sigma}_{train})$
  Execute the IIN
  $$(\hat{\mathbf{c}}_\mu, \hat{\mathbf{c}}_\sigma) \leftarrow \Theta(\tilde{o}_t, \tilde{a}_t)$$
  $\hat{\mathbf{c}}_\mu = \frac{1}{l_{train}} \sum_{t=1}^{l_{train}} \hat{\mathbf{c}}_{\mu,t}$ and $\hat{\mathbf{c}}_\sigma = \frac{1}{l_{train}} \sum_{t=1}^{l_{train}} \hat{\mathbf{c}}_{\sigma,t}$
  Calculate the loss
  $$\mathcal{L}(f(\mathbf{c}; \mathbf{c}_\mu, \mathbf{c}_\sigma, \mathrm{a}, \mathrm{b})) = H(f(\mathbf{c}; \mathbf{c}_\mu, \mathbf{c}_\sigma, \mathrm{a}, \mathrm{b}) | f(\mathbf{c}; \hat{\mathbf{c}}_\mu, \hat{\mathbf{c}}_\sigma, \mathrm{a}, \mathrm{b}))$$
  Update the IIN parameters
  $$\Theta \leftarrow \Theta - \alpha \Delta_\Theta \mathcal{L}(\cdot)$$
end while

Algorithm 3 IIN: Inference Phase

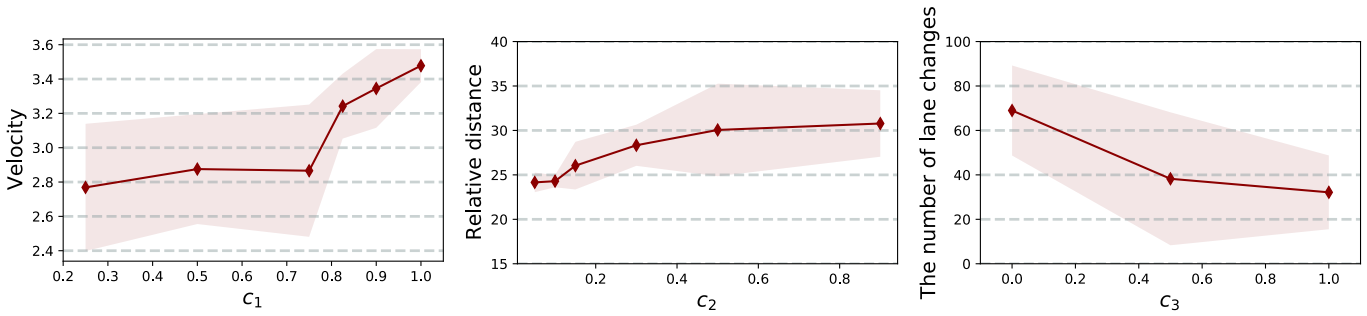
Require: Pre-trained IIN $\Theta(\cdot)$, observation noise standard deviation $\tilde{\sigma}_{test}$, trajectory length l_{test}, and buffer $\mathcal{B}$

Ensure:

for timestep $t = 1, T$ **do**
  Collect noisy observation
  $\tilde{o}_t \sim \mathcal{N}(\Omega(o_t|s_t), \tilde{\sigma}_{test})$
  Collect noisy action $\tilde{a}_t \sim \mathcal{N}(a_t, \tilde{\sigma}_{test})$
  Infer the character distribution
  $$\hat{\mathbf{c}}_{\mu,t}, \hat{\mathbf{c}}_{\sigma,t} = \Theta(\tilde{o}_t, \tilde{a}_t)$$
  Store $(\hat{\mathbf{c}}_{\mu,t}, \hat{\mathbf{c}}_{\sigma,t})$ in $\mathcal{B}$
  Calculate the estimated character
  $$\hat{\mathbf{c}}_\mu = \frac{1}{|\mathcal{B}|} \sum_{t=1}^{|\mathcal{B}|} \hat{\mathbf{c}}_{\mu,t}$$
  $$\hat{\mathbf{c}}_\sigma = \frac{1}{|\mathcal{B}|} \sum_{t=1}^{|\mathcal{B}|} \hat{\mathbf{c}}_{\sigma,t}$$
end for

VI. Simulation Resutls

This section outlines the experimental setups and empirical results. We first provide an environmental setup and an intuitive explanation of the driving character we consider. The empirical results are assessed on the FLOW traffic simulator [45].



A. Behavioral pattern related to c_1 **B**. Behavioral pattern related to c_2 **C**. Behavioral pattern related to c_3

Figure 3: The behavioral differences according to the change of each character element. **A**. Behavioral pattern as c_1 changes related to velocity. **B**. Behavioral pattern as c_2 changes related to relative distance. **C**. Behavioral pattern as c_3 changes related to lateral maneuvering preference.

Resource specifications and detailed hyperparameter settings are provided in Appendix G and Appendix F, respectively.

A. Simulation Setup

This experiment considers a three-lane highway environment with 22 vehicles driving. This road environment is designed as a closed loop to control the number of total vehicles. In the forward process, we deploy a single RL agent and 21 human-driven vehicles. The RL agent (e.g., character-aware meta-agent or POMDP agent) learns a character-conditioned policy, and human-driven vehicles are simulated by a noisy version of the intelligence driving model (IDM) [46].

To train the IIN, we use a million samples. As discussed in Section III-B, we incorporate observation noise to simulate a noisy environment. The performance of the IIN is comprehensively influenced by the properties of the environment. In this experiment, we distinguish between the training and test phases in terms of the notation used for the variance of observation noise and trajectory length: 1) $\tilde{\sigma}_{train}$ and $\tilde{\sigma}_{test}$, 2) l_{train} and l_{test}.[6]

To express characters $[c_1, c_2, c_3]$ of each agent, we parameterize the three terms of the reward function as follows.

$$R_t = c_1\mathcal{R}_1 + c_2\mathcal{R}_2 + c_3\mathcal{R}_3 + r_{fail}$$

Each reward term represents behavioral preferences in terms of desired velocity, relative distance, and lateral maneuvering preference. Next, r_{fail} is based on the safety technique and punishes the agent if the decision made by the agent is unfeasible [47].[7] We provide more details about experiments and the POMDP model in Appendix D.

All experimental results show the average performance and confidence interval with three standard deviations over 10^3 inference samples as the marker and dimmed area, respectively. We define the inference accuracy as $1 - \frac{1}{N}\|\mathbf{c}_\mu - \hat{\mathbf{c}}_\mu\|_2^2$.

[6]If the subscript $train$ or $test$ is not explicitly specified in l or $\tilde{\sigma}$, it indicates that the setups are consistent between the test and training phases.

[7]The unfeasible action of the agent is restrained by a shield. This technique effectively works from the viewpoint of learning because it rapidly narrows the range of feasible actions and avoids the stopping of learning by accidents.

B. Empirical Results of Driving Character

We validate observed behavioral differences linked to character variations, thereby providing intuition about how the character affects the generated trajectory. We expect that a higher character value will result in stronger regulation of vehicle behavior. Figure 3 illustrates the experimental results, which isolate the effect of each character, allowing us to understand their influence.

Figure 3**A** focuses on the first character c_1, which governs the vehicle's ability to maintain a desired velocity. In this simulation setting, the desired velocity is set as $3.5m/s$. As c_1 increases, the velocity of a trained RL vehicle is close to the desired velocity of $3.5m/s$ more consistently. Notably, lower character values correlate with greater dispersion in driving velocity.

Figure 3**B** explores the role of the second character c_2, responsible for keeping the relative distance of a trained RL vehicle to surrounding vehicles in the same lane. The result substantiates that the relative distance converges as the second character c_2 grows, indicating a focus on maintaining proper relative distance for safer driving.

Lastly, Figure 3**C** investigates the effect of the third character c_3 on lane-changing behavior. This character is designed to control the vehicle's preferences for lateral maneuvers. This result demonstrates a decrease in the number of lane-changing with an increase in c_3, suggesting that a higher value discourages lateral maneuvers.

In summary, these results demonstrate the explicit influence of the considered characters on the agent's behavior.

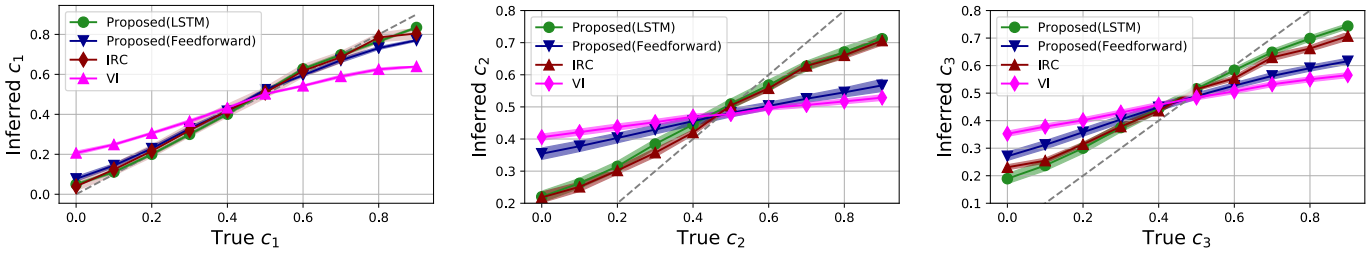
C. Performance Comparisons

We present extensive simulation results to demonstrate that the proposed solution exhibits competitive performance compared to existing methods. We evaluate performance based on inference accuracy and time complexity. We benchmark the following methods.

- Proposed (LSTM): The IIN includes the long short-term memory (LSTM) [48] layers. It allows the inference process to consider time series property in observation-action trajectories.

Table I: Inference accuracy (mean $\pm$ three standard deviations) and time complexity for four algorithms. In this table, l represents the train and test length of the trajectory (i.e., $l_{train} = l_{test}$), d denotes the dimension of the input feature, and k means the number of gradient ascent iterations in IRC. Note that we record the wall-time of the trajectory-based as a wall-time based on a single sample. Note that we provide the wall-time by dividing the total inference time for the trajectory by a single sample. For instance, with 200 samples, the proposed solution takes $1.153s$, whereas IRC takes $91.47s$.

Category	Algorithm	Inference Accuracy l=10	l=100	l=200	Time Complexity	Wall-time
Trajectory-based	Proposed (LSTM)	0.9564 $\pm$0.0017	0.9621 $\pm$0.0007	0.9633 $\pm$0.0006	$O(ld^2)$	$0.006s$
	IRC	0.7874 $\pm$0.0883	0.9441 $\pm$0.0451	0.9604 $\pm$0.0344	$O(kld^2)$	$0.457s$
Data point-based	Proposed (Feedforward)	0.9533 $\pm$0.0019			$O(d^2)$	$0.003s$
	VI	0.8900 $\pm$0.0022			$O(d^2)$	$0.004s$



A. Inference on c_1 **B**. Inference on c_2 **C**. Inference on c_3

Figure 4: The inferred character parameters of the agent. The grey dotted line represents a diagonal line, where the estimated result is the same as the true one. **A**: Inference on c_1, **B**: Inference on c_2, **C**: Inference on c_3.

- Proposed (Feedforward): The IIN includes only feedforward layers. It uses a single data point for the inference, not the trajectory sequence.
- IRC [19]: This inference method is based on Monte Carlo maximization estimation. It is to find the stochastic character distribution that best explains the given trajectories by maximizing their log-likelihood. This method includes iterations to perform the gradient ascent process.
- Variational Inference (VI) [49]: This inference method is based on Bayesian theorem. It is to approximate the posterior $p(\mathbf{c}|o,a)$ as a different probability $q(\mathbf{c})$ regardless of distribution type.

Table I presents the inference accuracy and time complexity over algorithms. The trajectory-based approaches utilize trajectory data, while the data point-based approach uses a single data point ($l = 1$). Therefore, we report simulation results for multiple trajectory lengths ($l \in 10, 100, 200$) for trajectory-based approaches.

1) Inference Accuracy: The result confirms that the LSTM-based solution with $l = 200$ performs best. As expected, trajectory-based approaches typically exhibit improved inference accuracy with longer observed trajectory sequences because a longer sequence provides a clearer interpretation of the target vehicle's behavior. Interestingly, the proposed feedforward-based solution achieves comparable performance to trajectory-based approaches. This suggests that feedforward-based IIN could serve as a valuable alternative if the IIN-equipped vehicle is not able to obtain a long sequence of observations for the target vehicle. Next, the IRC with $l = 200$ demonstrates the second-best inference performance, but its reliance on the Monte Carlo approach introduces higher variance. Lastly, the accuracy of VI is the lowest among the methods, despite its similarities to the proposed feedforward-based solution.

Figure 4 visualizes the inferred character value compared to the true value. The gray diagonal line represents perfect inference. Therefore, the closer the plotted points are to this line, the higher the inference accuracy. The proposed LSTM-based solution consistently outperforms other methods by exhibiting points that are closer to the gray reference line. Likewise, trajectory-based approaches are generally better than data point-based approaches for all characters c_1, c_2, c_3. Notably, inferring character c_1 is more accurate than c_2 and c_3. The most prominent difference in behavioral patterns across varying characters suggests a stronger influence of the reward component linked to c_1 on the RL-based vehicle's actions. This aligns with Figure 3, where c_1 exhibits the most distinct change in behavior. In simpler terms, c_1 plays a more significant role in shaping the target vehicle's decision-making process, leading to clearer behavioral patterns for the inference solution to pick up on.

2) Computational complexity: In Table I, k, l, and d denote the number of iterations, the trajectory length, and the input dimension, respectively. It demonstrates how quickly each solution can perform inference. All algorithms exhibit quadratic time complexity with respect to the input dimension d. Trajectory-based approaches require additional processing for each data point in the trajectory l. Furthermore, the IRC requires multiple iterations k using the gradient ascent process to find the best solution. This divergence in time complexity

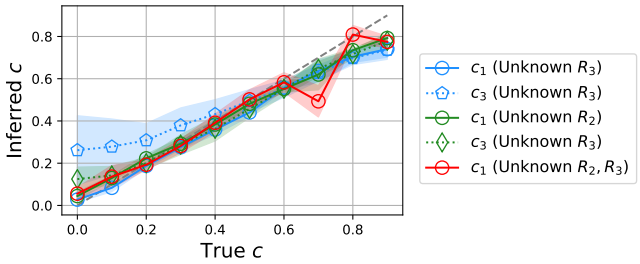
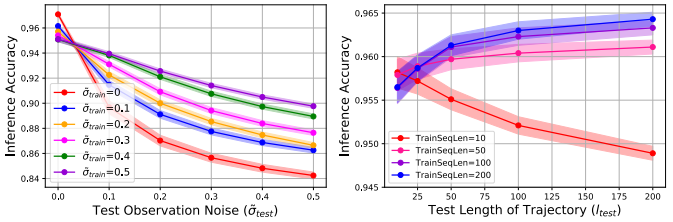


Figure 5: The inferred character by the IIN-equipped vehicle with an incomplete reward function. A target POMDP agent is based on a complete reward function. In legend, unknown reward components for the IIN-equipped vehicle are stated.



A. Accuracy over observation noise **B**. Accuracy over sequence length

Figure 6: **A**. The inference accuracy of IIN according to the magnitude of observation noise in the training and test phases when $l_{train} = l_{test} = 100$. **B**. The inference accuracy of IIN over the trajectory length in the training and test phases when $\tilde{\sigma}_{train} = \tilde{\sigma}_{test} = 0$.

explains why the IRC might not be suitable for real-time applications where quick inference is crucial. Conversely, the proposed solution offers superior computational efficiency. Detailed technical analysis of time complexity is provided in Appendix E.

In conclusion, the proposed solution outperforms others comprehensively in terms of both inference accuracy and time complexity.

D. In-depth Analyses for Practical Insights

Beyond simple performance comparison, this subsection enlightens a deeper understanding of the properties of the proposed solution. We aim to answer the following questions to provide practical insights.

- (**Interpretability**) Can we use the IIN to interpret the behavioral pattern of a target agent even if its reward function is inaccessible?
- (**Robustness**) Can the IIN perform well under high noise levels and short trajectory lengths?
- (**Scalability**) Does the IIN work beyond the truncated Gaussian of character distribution?
- (**Stability**) Can the IIN maintain the inference performance over the prolonged operation?

1) Interpretability: Complete knowledge about the reward components of the target vehicle may not always be possible. Our primary purpose is to interpret the behavioral pattern of the target vehicle by inferring character coefficients about the sketched reward function. The reward function contains several reward components interesting to scientists. In this part, we show that the proposed solution can be applied in scenarios where we do not know the exact reward function $r_t = c_1\mathcal{R}_1(s_t, a_t, s_{t+1}) + c_2\mathcal{R}_2(s_t, a_t, s_{t+1}) + c_3\mathcal{R}_3(s_t, a_t, s_{t+1})$.

To demonstrate our claim, we first build the IIN by considering part of the reward function and, after that, inferring the character of a target vehicle trained by the complete reward function. For example, if $\mathcal{R}_2$ is unknown, the IIN-equipped vehicle infers the characters $[c_1, c_3]$ of the target vehicle, which is trained by the complete three reward components. Inference results can be shown in Figure 5, and it demonstrates that the inferred character is close to the true character in all considered cases. This result confirms that the proposed solution can explain behavioral patterns even without knowing the exact reward function of the target vehicle.

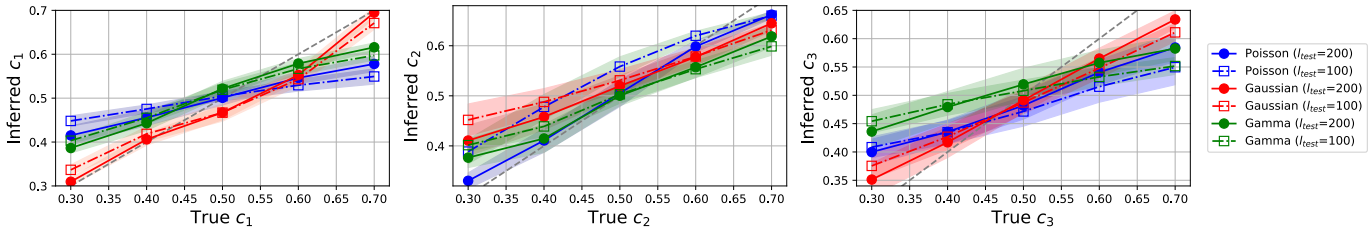
2) Robustness: We study the robustness of the IIN performance as the observation noise increases and the test sequence length decreases. Figure 6**A** shows the inference accuracy of the proposed solution as the observation noise considered in the inference phase increases. This figure confirms that the inference accuracy declines as the standard deviation of the observation noise increases in the inference phase. Interestingly, the higher the standard deviation of the observation noise in the training phase, the more robust the trained network is. We conclude that considering the observation noise in the training phase improves the inference accuracy when there is observation noise in the inference phase.

Next, Figure 6**B** depicts how the inference accuracy of the proposed model varies with the sequence length l_{test} of the test. When the sequence length in the training phase is more than 50, the inference accuracy increases as the sequence length in the inference phase grows. On the other hand, when the sequence length in the training phase is 10, the inference accuracy decreases as the sequence length in the test phase increases. This result suggests that the trained IIN with a short trajectory length might not have learned the patterns necessary to make accurate predictions for the longer sequences.

3) Scalability: We explore the possibility of IIN operating with unimodal distributions from the exponential family other than the truncated Gaussian distribution. Specifically, we consider the Gaussian, Poisson, and Gamma distributions, each defined by a characteristic parameter.

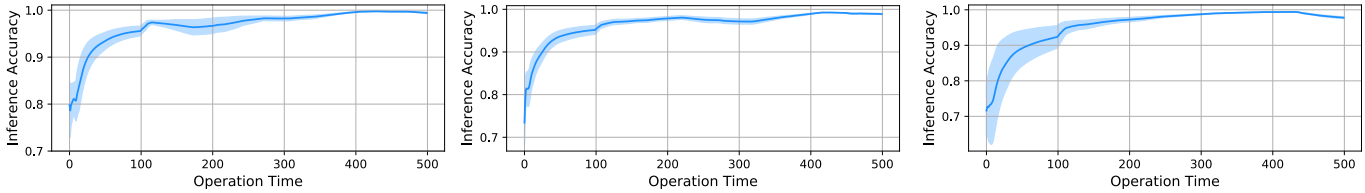
Figure 7 shows the inferred results of the LSTM-based proposed solution for each distribution. The proposed solution demonstrates effectiveness with these unimodal distributions from the exponential family, achieving decent performance. Interestingly, the trend of inference performance for each distribution is slightly different, For example, the Gaussian distribution performs better than others for c_1 and c_3 while the Poisson distribution performs better for c_2. Consequently, this result demonstrates that the proposed solution can operate effectively with various exponential family distributions beyond the truncated Gaussian.

4) Stability: We evaluate the performance change of the



A. Inference on c_1 **B**. Inference on c_2 **C**. Inference on c_3

Figure 7: The inferred character parameters for each distribution, belong to the exponential family. A square with a dotted line and a circle with a solid line denotes the inferred performance of LSTM-based proposed with $l_{test} = 200$ and $l_{test} = 100$, respectively. Blue, red, and green mean the Poisson, the Gaussian, and the Gamma, respectively. **A**: Inference on c_1, **B**: Inference on c_2, **C**: Inference on c_3.



A. Inference on c_1 **B**. Inference on c_2 **C**. Inference on c_3

Figure 8: The inference accuracy of IIN per character over operation time. We visually plot the mean and three standard deviations of 20 trajectories, which are generated by different characters. **A**: Inference on c_1, **B**: Inference on c_2, **C**: Inference on c_3.

proposed solution over a prolonged operation to study its inference stability. Figure 8 shows the inferred results of the LSTM-based proposed solution as operation time goes on. In the initial step, it can be seen that the inference accuracy is low, and the variance is relatively large due to the limited number of trajectory data about a target vehicle. The accuracy of the inferred character increases with low variance by updating it as the trajectory data accumulates. Consequently, this empirical result confirms that the proposed solution can continue stable inference without severe performance degradation over the prolonged operation.

VII. Conclusion

This study investigated the estimation approach as an essential function of vehicular edge computing to enhance decision-making robustness. Specifically, we proposed a novel workflow for an inference model to determine the driving character of a target vehicle based on its behavioral trajectories. The proposed solution first constructs a POMDP agent and a character-aware meta-agent using a stochastic agent model in RL, which captures the stochastic nature of character variations. The IIN-equipped vehicle collects observation-action pairs by observing a target POMDP agent. Subsequently, the IIN is trained using a universal policy and collected trajectory data. Once fully trained, the IIN is integrated into a vehicular edge computing module to infer the character of a target vehicle. Experimental results confirm that our proposed solution outperforms existing inference approaches comprehensively. This work further provides practical insights into enhancing the interpretability, robustness, and scalability of the proposed solution.

References

[1] D. Vitas, M. Tomic, and M. Burul, "Traffic light detection in autonomous driving systems," *IEEE Consumer Electronics Magazine*, vol. 9, no. 4, pp. 90–96, 2020.

[2] C. Tang, G. Yan, H. Wu, and C. Zhu, "Computation offloading and resource allocation in failure-aware vehicular edge computing," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 1877–1888, 2023.

[3] S. Yun, T. Teshima, and H. Nishimura, "Human–machine interface design and verification for an automated driving system using system model and driving simulator," *IEEE Consumer Electronics Magazine*, vol. 8, no. 5, pp. 92–98, 2019.

[4] D. Mijić, M. Vranješ, R. Grbić, and B. Jelić, "Autonomous driving solution based on traffic sign detection," *IEEE Consumer Electronics Magazine*, vol. 12, no. 5, pp. 39–44, 2023.

[5] C. Eom, D. Lee, and M. Kwon, "Selective imitation for efficient online reinforcement learning with pre-collected data," *ICT Express*, vol. 11, no. 2, 2025.

[6] L. Li, W. Zhao, C. Xu, C. Wang, Q. Chen, and S. Dai, "Lane-change intention inference based on RNN for autonomous driving on highways," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 5499–5510, 2021.

[7] D. González, V. Cano, S. Dibangoye, and C. Laugier, "Interaction-aware driver maneuver inference in highways using realistic driver models," in *IEEE International Conference on Intelligent Transportation Systems*, pp. 1–8, 2017.

[8] S. Park, D. Moore, and D. Sirkin, "What a driver wants: User preferences in semi-autonomous vehicle decision-making," in *ACM Conference on Human Factors in Computing Systems*, 2020.

[9] S. Griesche, E. Nicolay, D. Assmann, M. Dotzauer, and D. Käthner, "Should my car drive as I do? what kind of driving style do drivers prefer for the design of automated driving functions," in *Braunschweiger Symposium*, 2016.

[10] T. Chakraborti, S. Sreedharan, and S. Kambhampati, "Explicability versus explanations in human-aware planning," in *International Conference on Autonomous Agents and Multiagent Systems*, 2018.

[11] H. Moon, A. Oulasvirta, and B. Lee, "Amortized inference with user simulations," in *ACM Conference on Human Factors in Computing Systems*, pp. 1–20, 2023.

[12] D. Lee, C. Eom, and M. Kwon, "AD4RL: Autonomous driving benchmarks for offline reinforcement learning with value-based dataset," in *IEEE International Conference on Robotics and Automation*, 2024.
[13] O. Osoba, R. Vardavas, J. Grana, R. Zutshi, and A. Jaycocks, "Policy-focused agent-based modeling using RL behavioral models," in *International Conference on Machine Learning and Applications*, 2020.
[14] C. Kim, J. Park, J. Shin, H. Lee, P. Abbeel, and K. Lee, "Preference transformer: Modeling human preferences using transformers for RL," in *International Conference on Learning Representations*, 2023.
[15] G. Qu, H. Wu, R. Li, and P. Jiao, "DMRO: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3448–3459, 2021.
[16] M. Glöckler, M. Deistler, and J. Macke, "Variational methods for simulation-based inference," in *International Conference on Learning Representations*, 2022.
[17] P. Humphreys, D. Raposo, T. Pohlen, G. Thornton, R. Chhaparia, A. Muldal, J. Abramson, P. Georgiev, A. Santoro, and T. Lillicrap, "A data-driven approach for learning to control computers," in *International Conference on Machine Learning*, 2022.
[18] A. Kangasrääsiö, K. Athukorala, A. Howes, J. Corander, S. Kaski, and A. Oulasvirta, "Inferring cognitive models from data using approximate Bayesian computation," in *ACM Conference on Human Factors in Computing Systems*, pp. 1295–1306, 2017.
[19] M. Kwon, S. Daptardar, P. Schrater, and X. Pitkow, "Inverse rational control with partially observable continuous nonlinear dynamics," *Advances in Neural Information Processing Systems*, 2020.
[20] Z. Wu, M. Kwon, S. Daptardar, P. Schrater, and X. Pitkow, "Rational thoughts in neural codes," *Proceedings of the National Academy of Sciences*, vol. 117, no. 47, pp. 29311–29320, 2020.
[21] C. Gebhardt, A. Oulasvirta, and O. Hilliges, "Hierarchical reinforcement learning explains task interleaving behavior," *Computational Brain & Behavior*, vol. 4, no. 3, pp. 284–304, 2021.
[22] N. Chater, J. Zhu, J. Spicer, J. Sundh, P. Villagrá, and A. Sanborn, "Probabilistic biases meet the Bayesian brain," *Current Directions in Psychological Science*, vol. 29, no. 5, pp. 506–512, 2020.
[23] D. McNamee and D. Wolpert, "Internal models in biological control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, p. 339, 2019.
[24] T. Chakraborti, S. Sreedharan, and S. Kambhampati, "Human-aware planning revisited: A tale of three models," in *International Joint Conference on Artificial Intelligence and European Conference on Artificial Intelligence*, 2018.
[25] N. Rabinowitz, F. Perbet, F. Song, C. Zhang, A. Eslami, and M. Botvinick, "Machine theory of mind," in *International Conference on Machine Learning*, 2018.
[26] J. Lee, F. Sha, and C. Breazeal, "A Bayesian theory of mind approach to nonverbal communication," in *ACM/IEEE International Conference on Human–Robot Interaction*, 2019.
[27] M. Sclar, G. Neubig, and Y. Bisk, "Symmetric machine theory of mind," in *International Conference on Machine Learning*, 2022.
[28] A. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *International Conference on Machine Learning*, 2000.
[29] N. Ratliff, A. Bagnell, and M. Zinkevich, "Maximum margin planning," in *International Conference on Machine Learning*, pp. 729–736, 2006.
[30] B. Ziebart, A. Maas, A. Bagnell, and A. Dey, "Maximum entropy inverse reinforcement learning.," in *AAAI Conference on Artificial Intelligence*, vol. 8, pp. 1433–1438, 2008.
[31] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in Neural Information Processing Systems*, vol. 29, 2016.
[32] V. Kukkala, J. Tunnell, S. Pasricha, and T. Bradley, "Advanced driver-assistance systems: A path toward autonomous vehicles," *IEEE Consumer Electronics Magazine*, vol. 7, no. 5, pp. 18–25, 2018.
[33] P. Du, Y. Shi, H. Cao, S. Garg, M. Alrashoud, and P. Shukla, "AI-enabled trajectory optimization of logistics UAVs with wind impacts in smart cities," *IEEE Transactions on Consumer Electronics*, 2024.
[34] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen, "A survey on trajectory-prediction methods for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 652–674, 2022.
[35] C. Lin, G. Ulsoy, and D. LeBlanc, "Vehicle dynamics and external disturbance estimation for vehicle path prediction," *IEEE Transactions on Control Systems Technology*, vol. 8, no. 3, pp. 508–518, 2000.
[36] T. Batz, K. Watson, and J. Beyerer, "Recognition of dangerous situations within a cooperative group of vehicles," in *IEEE Intellgent Vehicles Symposium*, pp. 907–912, 2009.
[37] P. Pecher, M. Hunter, and R. Fujimoto, "Data-driven vehicle trajectory prediction," in *ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pp. 13–22, 2016.
[38] L. Lin, W. Li, H. Bi, and L. Qin, "Vehicle trajectory prediction using LSTMs with spatial–temporal attention mechanisms," *IEEE Intelligent Transportation Systems Magazine*, vol. 14, no. 2, pp. 197–208, 2021.
[39] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang, "STGAT: Modeling spatial-temporal interactions for human trajectory prediction," in *IEEE/CVF International Conference on Computer Vision*, pp. 6272–6281, 2019.
[40] Z. Zhong, Y. Luo, and W. Liang, "STGM: Vehicle trajectory prediction based on generative model for spatial-temporal features," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 18785–18793, 2022.
[41] A. Akabane, R. Pazzi, E. Madeira, and L. Villas, "Modeling and prediction of vehicle routes based on hidden Markov model," in *IEEE Vehicular Technology Conference*, pp. 1–5, 2017.
[42] X. Zhang and S. Mahadevan, "Bayesian neural networks for flight trajectory prediction and safety assessment," *Decision Support Systems*, vol. 131, 2020.
[43] Y. Feng and X. Yan, "Support vector machine based lane-changing behavior recognition and lateral trajectory prediction," *Computational Intelligence and Neuroscience*, 2022.
[44] K. Messaoud, I. Yahiaoui, A. Blondet, and F. Nashashibi, "Attention based vehicle trajectory prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 1, pp. 175–185, 2020.
[45] E. Vinitsky, A. Kreidieh, L. Flem, N. Kheterpal, K. Jang, C. Wu, F. Wu, R. Liaw, E. Liang, and A. Bayen, "Benchmarks for reinforcement learning in mixed-autonomy traffic," in *Conference on Robot Learning*, 2018.
[46] D. Lee and M. Kwon, "Stability analysis in mixed-autonomous traffic with deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 2848–2862, 2023.
[47] D. Lee and M. Kwon, "ADAS-RL: Safety learning approach for stable autonomous driving," *ICT Express*, vol. 8, no. 3, pp. 479–483, 2022.
[48] T. Blom, D. Feuerriegel, P. Johnson, S. Bode, and H. Hogendoorn, "Predictions drive neural representations of visual events ahead of incoming sensory information," *Proceedings of the National Academy of Sciences*, vol. 117, no. 13, pp. 7510–7515, 2020.
[49] D. Blei, A. Kucukelbir, and J. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

Dongsu Lee received the B.S. degree from the School of Systems Biomedical Science, Soongsil University, Seoul, South Korea. He is currently pursuing a Ph.D. degree in the Department of Intelligent Semiconductors at Soongsil University, Seoul, South Korea. His research interest includes development of reinforcement learning and understanding its mathematical properties.

Minhae Kwon received the B.S., M.S., and Ph.D. degrees from the Department of Electronic and Electrical Engineering, Ewha Womans University, Seoul, South Korea. She was a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, Rice University, and the Center for Neuroscience and Artificial Intelligence, Baylor College of Medicine, Houston, TX, USA. She is currently an Assistant Professor at Soongsil University, Seoul, Republic of Korea. Her research focuses on decision-making in intelligent systems, particularly in communication and networking systems and autonomous driving systems. Prof. Kwon won many prestigious awards, including the Minister's Award and the Excellence Award for Young Scientists from the Korean government, the Best Poster Video Award from the IEEE Consumer Electronics Society, the Qualcomm Innovation Award from Qualcomm Inc., and the Google Anita Borg Fellowship from Google Inc.

Instant Inverse Modeling of Stochastic Driving Behavior with Deep Reinforcement Learning

Dongsu Lee and Minhae Kwon

APPENDIX A
SUMMARY OF NOTATIONS

Notation	Description Notation	Description	
$\mathcal{S}$	a state space	s_t	a state at time t
$\mathcal{O}$	an observation space	o_t	an observation at time t
$\mathcal{A}$	an action space	a_t	an action at time t
$\mathcal{T}$	a state transition probabilities	Ω	an observation transition probabilities
R_t	a reward function	γ	a temporal discounted factor
$\mathcal{C}$	a character space	$\mathbf{c}$	a discrete character vector
$f(\cdot)$	a character distribution	$[a, b]$	bounded finite space of $f(\cdot)$
$\mathbf{c}_\mu$	a mean of $f(\cdot)$	$\mathbf{c}_\sigma$	a standard deviation of $f(\cdot)$
$\mathbf{c}'$	a character variable	$Q_\psi(\cdot)$	a Q-function with network parameter ψ
$\pi_\phi(\cdot)$	a policy with network parameter ϕ	$\Pi_\phi(\cdot)$	a universal policy with ϕ
$\tilde{\Omega}_o(\cdot)$	an observation collection function of meta-agent	$\tilde{\Omega}_a(\cdot)$	an action collection function of meta-agent
$\tilde{\sigma}$	a standard deviation of the collection function of meta-agent		

APPENDIX B
LOSS FUNCTION DESIGN OF THE IIN TRAINING

This section provides an objective function to address the inference problem. The main objective of the IIN is to minimize the distance between the true $f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b})$ and inferred character distribution $f(\mathbf{c}; \hat{\mathbf{c}}_\mu, \hat{\mathbf{c}}_\sigma, \mathrm{a}, \mathrm{b})$, where $\hat{\mathbf{c}}_\mu, \hat{\mathbf{c}}_\sigma$ represent the output of the IIN $\Theta(o_t, a_t)$. Therefore, we adopt the Kullback-Leibler (KL) divergence as distance metric. Finding arguments $\hat{\mathbf{c}}_\mu, \hat{\mathbf{c}}_\sigma$ that minimize the KL-divergence between the true and inferred character distribution is identical to finding arguments that minimize the cross-entropy between the true and inferred one:

$$H(f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b}) | f(\mathbf{c}; \hat{\mathbf{c}}_\mu, \hat{\mathbf{c}}_\sigma, \mathrm{a}, \mathrm{b})).$$

This is summarized in the following property.

Property 4. *Finding arguments $\hat{\mathbf{c}}_\mu, \hat{\mathbf{c}}_\sigma$ that minimize the KL-divergence between the true character $f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b})$ and inferred character distribution $f(\mathbf{c}; \hat{\mathbf{c}}_\mu, \hat{\mathbf{c}}_\sigma, \mathrm{a}, \mathrm{b})$ is identical with finding arguments that minimize the cross-entropy between the true character and inferred character distribution $H(f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b}) | f(\mathbf{c}; \hat{\mathbf{c}}_\mu, \hat{\mathbf{c}}_\sigma, \mathrm{a}, \mathrm{b}))$, where $H(\cdot|\cdot)$ denotes the cross entropy between two distribution.*

Proof. We aim to find arguments $\hat{\mathbf{c}}_\mu, \hat{\mathbf{c}}_\sigma$ that minimize the KL-divergence between the true character $f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b})$ and inferred character distribution $f(\mathbf{c}; \hat{\mathbf{c}}_\mu, \hat{\mathbf{c}}_\sigma, \mathrm{a}, \mathrm{b})$ as followings.

$$\begin{aligned}
&\hat{\mathbf{c}}_\mu, \hat{\mathbf{c}}_\sigma \\
&= \arg\min_{\mathbf{c}_\mu, \mathbf{c}_\sigma} D_{KL}(f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b}) || f(\mathbf{c}; \mathbf{c}_\mu, \mathbf{c}_\sigma, \mathrm{a}, \mathrm{b})) \\
&= \arg\min_{\mathbf{c}_\mu, \mathbf{c}_\sigma} \left[- \int f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b}) \ln \frac{f(\mathbf{c}; \mathbf{c}_\mu, \mathbf{c}_\sigma, \mathrm{a}, \mathrm{b})}{f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b})} \right] d\mathbf{c} \\
&= \arg\min_{\mathbf{c}_\mu, \mathbf{c}_\sigma} \Big[\int f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b}) \ln f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b}) d\mathbf{c} \\
&\quad - \int f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b}) \ln f(\mathbf{c}; \mathbf{c}_\mu, \mathbf{c}_\sigma, \mathrm{a}, \mathrm{b}) d\mathbf{c} \Big]
\end{aligned} \tag{4}$$

Hereforth, since $\int f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b}) \ln f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b}) d\mathbf{c}$ is not dependent on $\mathbf{c}_\mu, \mathbf{c}_\sigma$, then (4) can be written as follows.

$$\begin{aligned}
&\hat{\mathbf{c}}_\mu, \hat{\mathbf{c}}_\sigma \\
&= \arg\min_{\mathbf{c}_\mu, \mathbf{c}_\sigma} \left[- \int f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b}) \ln f(\mathbf{c}; \mathbf{c}_\mu, \mathbf{c}_\sigma, \mathrm{a}, \mathrm{b}) d\mathbf{c} \right] \\
&= \arg\min_{\mathbf{c}_\mu, \mathbf{c}_\sigma} H\left(f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b}) | f(\mathbf{c}; \mathbf{c}_\mu, \mathbf{c}_\sigma, \mathrm{a}, \mathrm{b}) \right)
\end{aligned}$$

Therefore, finding arguments $\mathbf{c}_\mu, \mathbf{c}_\sigma$ that minimize $D_{KL}(f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b})||f(\mathbf{c}; \mathbf{c}_\mu, \mathbf{c}_\sigma, \mathrm{a}, \mathrm{b}))$ can be rewritten as those that minimize $H(f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b})|f(\mathbf{c}; \mathbf{c}_\mu, \mathbf{c}_\sigma, \mathrm{a}, \mathrm{b}))$. □

APPENDIX C
PROOF FOR THEOREM 3

Suppose that the true character distribution follows the Dirac delta distribution, expressed as $f(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma, \mathrm{a}, \mathrm{b}) = \delta(\mathbf{c} - \mathbf{c}^*_\mu)$, and that the inferred character distribution $\hat{f}(\mathbf{c}; \mathbf{c}^*_\mu, \mathbf{c}^*_\sigma)$ satisfies the Gaussian distribution, with $\mathbf{c}_\sigma$ is a constant, the objective function can be simplified as follows.

$$\begin{aligned}\hat{\mathbf{c}}_\mu &= \arg\min_{\mathbf{c}_\mu} \mathcal{L}(f(\mathbf{c}; \mathbf{c}_\mu, \mathbf{c}_\sigma, \mathrm{a}, \mathrm{b})) \\ &= \arg\min_{\mathbf{c}_\mu} \left[-\int \delta(\mathbf{c} - \mathbf{c}^*_\mu) \ln \hat{f}(\mathbf{c}; \mathbf{c}_\mu, \mathbf{c}_\sigma) d\mathbf{c} \right]\end{aligned}$$

Given that we are treating each character $c_1, c_2, \cdots, c_N$ within the character vector $\mathbf{c}$ independently, we can decompose the vector as each character.

$$\hat{\mathbf{c}}_\mu = \arg \min_{c_{1,\mu}, c_{2,\mu}, \cdots, c_{N,\mu}} \frac{1}{N} \sum_{n=1}^{N} \Big[-\int \delta(c_n - c^*_{n,\mu}) \ln \hat{f}(c_n; c_{n,\mu}, c_{n,\sigma}) dc_n \Big] \tag{5}$$

Considering $\ln \hat{f}(c_n; c_{n,\mu}, c_{n,\sigma})$ as function $g(x)$, we can rewrite (5) using the property of the Dirac delta function, $\int \delta(x - k) g(x) dx = g(k)$, as follows.

$$\hat{\mathbf{c}}_\mu = \arg \min_{c_{1,\mu}, c_{2,\mu}, \cdots, c_{N,\mu}} -\frac{1}{N} \sum_{n=1}^{N} \ln \hat{f}(c^*_{n,\mu}; c_{n,\mu}, c_{n,\sigma})$$

Substituting $\hat{f}(c^*_{n,\mu}; c_{n,\mu}, c_{n,\sigma})$ with $\frac{1}{c_{n,\sigma}\sqrt{2\pi}} \exp(-\frac{(c_{n,\mu} - c^*_{n,\mu})^2}{2c^2_{n,\sigma}})$ due to $\hat{f}(\cdot)$ following a Gaussian distribution, we obtain:

$$\begin{aligned}\hat{\mathbf{c}}_\mu &= \arg \min_{c_{1,\mu}, c_{2,\mu}, \cdots, c_{N,\mu}} -\frac{1}{N} \sum_{n=1}^{N} \ln \left[\frac{1}{c_{n,\sigma}\sqrt{2\pi}} \exp\left(-\frac{(c_{n,\mu} - c^*_{n,\mu})^2}{2c^2_{n,\sigma}} \right) \right] \\ &= \arg \min_{c_{1,\mu}, c_{2,\mu}, \cdots, c_{N,\mu}} \frac{1}{N} \sum_{n=1}^{N} \left[\ln(c_{n,\sigma}\sqrt{2\pi}) + \frac{1}{2c^2_{n,\sigma}} (c_{n,\mu} - c^*_{n,\mu})^2 \right]. \end{aligned} \tag{6}$$

Disregarding terms related to $c_{n,\sigma}$, which are constant, we rewrite (6) as:

$$\hat{\mathbf{c}}_\mu = \arg \min_{c_{1,\mu}, c_{2,\mu}, \cdots, c_{N,\mu}} \frac{1}{N} \sum_{n=1}^{N} (c^*_{n,\mu} - c_{n,\mu})^2.$$

Therefore, the optimization task can be satisfied as $\hat{\mathbf{c}}_\mu = \arg\min_{\mathbf{c}_\mu} \frac{1}{N} \sum_{n=1}^{N} (c^*_{n,\mu} - c_{n,\mu})^2$, if the character is deterministic.

APPENDIX D
DEMONSTRATION TASK: AUTOMATED DRIVING

To demonstrate our proposed solution, we adopt the automated driving task that diverse vehicles with heterogeneous characters coexist on the road. In this task, the agents are the automated vehicles that drive on the L-lane roundabout road. In a policy training phase, the agents are placed at random along the road in every episode. Each agent aims to learn the driving policy that maximizes the reward according to a given character. Specifically, all agents can partially observe the state information and learn about the acceleration and lane-changing controls. We formalize this task as the POMDP, which comprises a state, an observation, an action, a reward, and so on. Here, the state and observation include information about full roads and local roads, respectively. As earlier said, the action involves acceleration and lane-changing controls. Finally, the reward function comprises four terms with respect to a target velocity, safety distance, lane change, and unfeasible action. In the following subsection, we provide the details of the POMDP setting.

A. State

The state $s_t \in \mathcal{S}$ includes the information on all vehicles that drive on the road, as follows:

$$s_t = [\mathbf{v}_t^T, \mathbf{p}_t^T, \mathbf{k}_t^T]^T,$$

where $\mathbf{v}_t = [v_{t,1}, v_{t,2}, \cdots, v_{t,E}]$ denotes a vector on the velocity of all vehicles, $\mathbf{p}_t = [p_{t,1}, p_{t,2}, \cdots, p_{t,E}]$ represents a vector on the positions of the vehicles, and $\mathbf{k}_t = [k_{t,1}, k_{t,2}, \cdots, k_{t,E}]$ indicates a vector on the lane position of all vehicle at a timestep t. Here, the subscripts $\{1, 2, \cdots, E\}$ represent the vehicle number.

B. Observation

The observation $o_t \in \mathcal{O}$ includes the information on neighboring vehicles that drive on the surrounding road of the agent e. Specifically, the agent e can observe the leading and following vehicles located in the same and adjacent lanes. Therefore, o_t is defined as follows:

$$o_t = [v_{t,e}, \Delta\mathbf{v}_{t,e}, \Delta\mathbf{p}_{t,e}, k_{t,e}]^T,$$

where $v_{t,e}$ indicates the velocity of the agent e, $\Delta\mathbf{v}_{t,e}$ is relative velocity between the agent e and observable vehicles, $\Delta\mathbf{p}_{t,e}$ is relative position between the agent e and observable vehicles, and $k_{t,e}$ represents the lane number at a timestep t. Here, $\Delta\mathbf{v}_{t,e} = [\Delta v_{t,fL}, \Delta v_{t,fS}, \Delta v_{t,fR}, \Delta v_{t,lL}, \Delta v_{t,lS}, \Delta v_{t,lR}]$, and $\Delta\mathbf{p}_{t,e} = [\Delta p_{t,fL}, \Delta p_{t,fS}, \Delta p_{t,fR}, \Delta p_{t,lL}, \Delta p_{t,lS}, \Delta p_{t,lR}]$, where subscripts f and l signify following and leading vehicles, and subscripts L, S, and R indicate located left, same, and right lane, respectively.

C. Action

The action $a_t \in \mathcal{A}$ is defined as follows:

$$a_t = [a_t^c, a_t^d]^T, \tag{7}$$

where a_t^c indicates an acceleration control, and a_t^d represents a lane-changing control. Specifically, the acceleration control $a_t^c \in \mathcal{A}^c$ is a continuous action, where $\mathcal{A}^c$ means a continuous action space; the lane-changing control $a_t^d \in \mathcal{A}^d$ is a discrete action, where $\mathcal{A}^d$ means a discrete action space. In this task, we set the continuous action space as $[-1m/s^2, 1m/s^2]$ and the discrete action space as $\{-1, 0, 1\}$. Here, $a_t^d = -1$ and $a_t^d = 1$ represent lane-changing to the right and left lanes, respectively. $a_t^d = 0$ indicates keeping to the same lane.

D. Reward

The reward function $R_t(\cdot)$ comprises four terms as follows:

$$R_t = c_1\mathcal{R}_1 + c_2\mathcal{R}_2 + c_3\mathcal{R}_3 + r_{fail}.$$

First reward term $\mathcal{R}_1$ is defined as follows:

$$\mathcal{R}_1 = 1 - \left|\frac{v_{t+1,e} - v_e^*}{v_e^*}\right|,$$

where v_e^* represents the desired velocity of the agent e. This term induces that the agent can drive close the v_e^*. Specifically, if $v_{t,e} = v_e^*$, $\mathcal{R}_1$ is maximized as the highest value 1; if $v_{t,e} \neq v_e^*$, $\mathcal{R}_1$ decreases as the difference between $v_{t,e}$ and v_e^* increases.

The second reward term $\mathcal{R}_2$ induces the agent to change lanes while keeping in mind the following vehicle. Therefore, we define $\mathcal{R}_2$ as follows:

$$\mathcal{R}_2(\Delta p_{t+1,fS}) = \min\left[0, 1 - \left(\frac{s^*}{\Delta p_{t+1,fS}}\right)^2\right],$$

where s^* denotes the safety distance between the vehicles, and it is defined as follows.

$$s^* = s_0 + \max\left[0, v_{t+1,fS}\left(t^* + \frac{\Delta v_{t+1,fS}}{2\sqrt{|A_{min} \times A_{max}|}}\right)\right],$$

where s_0 represents a minimum gap between vehicles, t^* indicates a minimum time headway, which is a minimum time gap between two sequential vehicles required to arrive at the same location. If $s^* \leq \Delta p_{t+1,fS}$, $\mathcal{R}_2$ becomes the 0; on the other hand, if $s^* > \Delta p_{t+1,fS}$, $\mathcal{R}_2$ becomes the negative value. In other words, the agent is punished when the agent does not keep a safe distance from a following vehicle when moving the lane. We set s^* based on the Intelligent Driving Model (IDM) controller, which is the representative adaptive cruise controller (Treiber et al., 2000).

Third reward term $\mathcal{R}_3$ is defined as follows:

$$\mathcal{R}_3 = |a_t^d|\Delta p_{t,lS} \times \min[0, \Delta p_{t+1,lS} - \Delta p_{t,lS}].$$

This reward term encourages the agent to avoid unnecessary lane changes, which are movements to lanes with less driving space than the present lane. When $|a^d| = 1$ and $\Delta p_{t,lS} < \Delta p_{t+1,lS}$ (i.e., the agent is moving to lanes with more driving space than the present lane) or $|a_t^d| = 0$ (i.e, the agent does not move the lane), $\mathcal{R}_3$ becomes 0. On the other hand, when $|a_t^d| = 1$ and $\Delta p_{t,lS} \geq \Delta p_{t+1,lS}$, $\mathcal{R}_3$ becomes the negative value.

Finally, r_fail is based on the safety RL technique and punishes the agent if the decision made by the agent is unfeasible. Specifically, the unfeasible action of the agent is restrained by *shield*. Therefore, it is possible to indirectly punish unfeasible actions without experiencing accidents during training. This technique effectively works from the viewpoint of learning because it rapidly narrows the range of feasible actions and avoids the stopping of learning by accidents.

APPENDIX E
COMPUTATIONAL COMPLEXITY ANALYSIS

Proposed	Computation	Equation	Matrix Size	Complexity
1^{st} LSTM	input gate	$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi})$	$W_{ii} \in \mathbb{R}^{2d\times d}$, $W_{hi} \in \mathbb{R}^{2d\times 2d}$, $x_t \in \mathbb{R}^d, h_t \in \mathbb{R}^{2d}$	$6d^2 + 4d$
	forget gate	$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf})$	$W_{if} \in \mathbb{R}^{2d\times d}$, $W_{hf} \in \mathbb{R}^{2d\times 2d}$, $x_t \in \mathbb{R}^d, h_t \in \mathbb{R}^{2d}$	$6d^2 + 4d$
	cell gate	$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg})$	$W_{ig} \in \mathbb{R}^{2d\times d}$, $W_{hg} \in \mathbb{R}^{2d\times 2d}$, $x_t \in \mathbb{R}^d, h_t \in \mathbb{R}^{2d}$	$6d^2 + 4d$
	output gate	$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho})$	$W_{io} \in \mathbb{R}^{2d\times d}$, $W_{ho} \in \mathbb{R}^{2d\times 2d}$, $x_t \in \mathbb{R}^d, h_t \in \mathbb{R}^{2d}$	$6d^2 + 4d$
	cell state update	$c_t = f_t \odot c_{t-1} + i_t \odot g_t$	$c_{t-1} \in \mathbb{R}^{2d}, f_t \in \mathbb{R}^{2d}, i_t \in \mathbb{R}^{2d}, o_t \in \mathbb{R}^{2d}$	$4d$
	hidden state	$h_t = o_t \odot \tanh(c_t)$	$o_t \in \mathbb{R}^{2d}, c_t \in \mathbb{R}^{2d}$	$4d$
2^{nd} LSTM	input gate	$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi})$	$W_{ii} \in \mathbb{R}^{d\times 2d}$, $W_{hi} \in \mathbb{R}^{d\times d}$, $x_t \in \mathbb{R}^{2d}, h_t \in \mathbb{R}^d$	$3d^2 + 2d$
	forget gate	$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf})$	$W_{if} \in \mathbb{R}^{d\times 2d}$, $W_{hf} \in \mathbb{R}^{d\times d}$, $x_t \in \mathbb{R}^{2d}, h_t \in \mathbb{R}^d$	$3d^2 + 2d$
	cell gate	$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg})$	$W_{ig} \in \mathbb{R}^{d\times 2d}$, $W_{hg} \in \mathbb{R}^{d\times d}$, $x_t \in \mathbb{R}^{2d}, h_t \in \mathbb{R}^d$	$3d^2 + 2d$
	output gate	$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho})$	$W_{io} \in \mathbb{R}^{d\times 2d}$, $W_{ho} \in \mathbb{R}^{d\times d}$, $x_t \in \mathbb{R}^{2d}, h_t \in \mathbb{R}^d$	$3d^2 + 2d$
	cell state update	$c_t = f_t \odot c_{t-1} + i_t \odot g_t$	$c_{t-1} \in \mathbb{R}^d, f_t \in \mathbb{R}^d, i_t \in \mathbb{R}^d, o_t \in \mathbb{R}^d$	$2d$
	hidden state	$h_t = o_t \odot \tanh(c_t)$	$o_t \in \mathbb{R}^d, c_t \in \mathbb{R}^d$	$2d$
Dense layer		$\mathbf{c}_t = \sigma(W_D \cdot h_t + b)$	$W_D \in \mathbb{R}^{3\times d}, h_t \in \mathbb{R}^d$	$3d$

IRC	Computation	Equation	Matrix Size	Complexity
The 1^{st} policy layer		$\text{out}_1 = \sigma_1(W_1 \cdot x_t + b_1)$	$W_1 \in \mathbb{R}^{2d\times d}, x_t \in \mathbb{R}^d$	$2d^2$
The 2^{nd} policy layer		$\text{out}_2 = \sigma_2(W_2 \cdot \text{out}_1 + b_2)$	$W_2 \in \mathbb{R}^{4d\times 2d}, \text{out}_1 \in \mathbb{R}^{2d}$	$8d^2$
The output layer		$a_t = \tanh(W_3 \cdot \text{out}_2 + b_3)$	$W_3 \in \mathbb{R}^{2\times 4d}, \text{out}_2 \in \mathbb{R}^{4d}$	$8d$
Loss calculation		$(a_t^* - \hat{a}_t)^2$	$a_t^* \in \mathbb{R}^{\lvert a_t\rvert}, \hat{a}_t \in \mathbb{R}^{\lvert a_t\rvert}$	$\lvert a_t\rvert$
Gradient Ascent		$\mathbf{c}_{t+1} \leftarrow \mathbf{c}_t + \alpha\nabla_{\mathbf{c}}\mathcal{L}(\mathbf{c})$	$\mathbf{c}_t \in \mathbb{R}^N, \mathcal{L}(\cdot) \in \mathbb{R}^N$	N

VI & MLP	Computation	Equation	Matrix Size	Complexity
The 1^{st} network layer		$\text{out}_1 = \sigma_1(W_1 \cdot x_t + b_1)$	$W_1 \in \mathbb{R}^{2d\times d}, x_t \in \mathbb{R}^d$	$2d^2$
The 2^{nd} network layer		$\text{out}_2 = \sigma_2(W_2 \cdot \text{out}_1 + b_2)$	$W_2 \in \mathbb{R}^{4d\times 2d}, \text{out}_1 \in \mathbb{R}^{2d}$	$8d^2$
The output layer		$\mathbf{c}_t = \tanh(W_3 \cdot \text{out}_2 + b_3)$	$W_3 \in \mathbb{R}^{N\times 4d}, \text{out}_2 \in \mathbb{R}^{4d}$	$4Nd$

We calculate the time complexity of algorithms in the inference phase (i.e., execution phase). We denote the dimension of input as d, the trajectory as l, and the iteration as k. As can be seen from the table, the complexity for one operation of each algorithm is as follows: the proposed has $36d^2 + 39d$, IRC has $10d^2 + 8d + N|a_t|$, and VI & MLP have $10d^2 + 4Nd$, where N means the size of the character vector, and $|a_t|$ denotes the action size. Here, we can rewrite $\mathcal{O}(36d^2 + 39d) = \mathcal{O}(d^2)$, $\mathcal{O}(10d^2 + 8d + N|a_t|) = \mathcal{O}(d^2)$, and $\mathcal{O}(10d^2 + 4Nd) = \mathcal{O}(d^2)$, where all constants can be ignored by $\mathcal{O}$, and $d \gg N, |a_t|$. Subsequently, the proposed and IRC consider the sequence length l, thus their complexity is $\mathcal{O}(ld^2)$. Finally, the IRC requires k-times iterations because the optimization of IRC is based on the gradient ascent method. Therefore, the complexity of IRC is $\mathcal{O}(kld^2)$.

APPENDIX F
HYPERPARAMETERS

A. Universal Policy Network

Hyperparameter	**Value**	**Hyperparameter**	**Value**
total episodes	4000	total timesteps	3000
target noise variance	0.2	target noise clip	0.2
replay buffer size	3×10^6	train batch size	128
discount factor (γ)	0.99	soft update rate	1×10^{-3}
actor learning rate	5×10^{-4}	actor hidden node	$[32, 64, 32]$
critic learning rate	5×10^{-4}	critic hidden node	$[32, 64, 32]$
activation function of actor hidden layer	ReLU	activation function of actor output layer	tanh
activation function of critic hidden layer	ReLU	activation function of critic output layer	linear

B. Instane Inference Network

Hyperparameter	**Value**	**Hyperparameter**	**Value**
epochs	100	batch size	4096
optimizer	Adam	learning rate	0.005
LSTM layer	2	bidirectional LSTM	False
LSTM network size	$[32, 16]$	LSTM sequence length size	100
activation function of hidden layer	ReLU	activation function of output layer	linear

APPENDIX G
SYSTEM SPECIFICATION

CPU	AMD Ryzen 9 3950X 16-core
GPU	GeForece RTX 2080 Ti
RAM	128 GB
SSD	1T
OS	Linux 18.04