Learning to Navigate Intersections with Unsupervised Driver Trait Inference

Shuijing Liu, Peixin Chang, Haonan Chen,
Neeloy Chakraborty, and Katherine Driggs-Campbell

***Abstract*— Navigation through uncontrolled intersections is one of the key challenges for autonomous vehicles. Identifying the subtle differences in hidden traits of other drivers can bring significant benefits when navigating in such environments. We propose an unsupervised method for inferring driver traits such as driving styles from observed vehicle trajectories. We use a variational autoencoder with recurrent neural networks to learn a latent representation of traits without any ground truth trait labels. Then, we use this trait representation to learn a policy for an autonomous vehicle to navigate through a T-intersection with deep reinforcement learning. Our pipeline enables the autonomous vehicle to adjust its actions when dealing with drivers of different traits to ensure safety and efficiency. Our method demonstrates promising performance and outperforms state-of-the-art baselines in the T-intersection scenario. For code implementation and videos, please visit `https://github.com/Shuijing725/VAE_trait_inference`.**

I. Introduction

To successfully navigate through uncontrolled intersections, autonomous vehicles must carefully reason about how to interact with different types of human drivers [1]. It is important for the vehicle to infer the traits of human drivers, such as the propensity for aggression or cooperation, and adjust its strategies accordingly [2], [3]. Inspired by recent advancements of unsupervised learning and deep reinforcement learning, we propose a novel pipeline to learn a representation of traits of other drivers, which is used for autonomous navigation in uncontrolled intersections.

Trait inference is challenging yet essential for the navigation of autonomous vehicles for two reasons. First, the environment is not fully observable to the ego vehicle, since each traffic participant runs its own policy individually and has its own internal states. The ego vehicle needs to interpret the hidden states such as driving styles and intents of other agents to understand future behaviors that may influence planning [2], [3]. Second, uncontrolled intersections are less structured since traffic lights and stop signs are not present to coordinate agent behaviors. The traffic participants implicitly interact and negotiate with each other, making the environment complex and potentially dangerous [4]–[7]. By inferring the traits of other drivers, the ego vehicle can be cautious when other drivers are aggressive or irrational and bold when they are passive or cooperative, improving both the safety and efficiency of interactive navigation.

S. Liu, P. Chang, H. Chen, N. Chakraborty and K. Driggs-Campbell are with the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. emails: {sliu105,pchang17,haonan2,neeloyc2,krdc}@illinois.edu

This material is based upon work supported by the National Science Foundation under Grant No. 2143435.

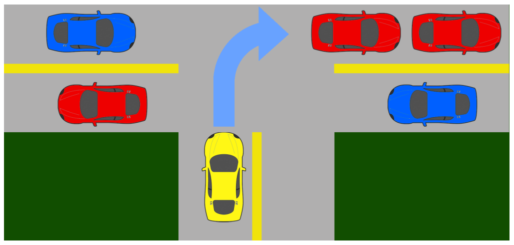


Fig. 1: **The T-intersection scenario in left-handed traffic.** The goal of the ego car (yellow) is to take a right turn and merge into the upper lane without colliding with other cars. The conservative car (blue) yields to the ego car while the aggressive cars (red) do not yield.

To address the above problems, Morton *et al* learns a latent representation of driver traits, which is fed into a feedforward policy to produce multimodal behaviors [8]. However, the feedforward policy only considers current states and actions which are not sufficient to fully express long-term properties of drivers such as traits. As a result, this representation fails to distinguish between different traits. Ma *et al* classifies driver traits with supervised learning to aid navigation in intersections [3] but has the following two problems. First, the trait labels are expensive to obtain and usually do not exist in most real driving datasets [1], [9]. Second, the navigation policy is trained with ground truth trait labels instead of predicted traits. When the trait classifier and the policy are combined in testing, intermediate and cascading errors cause severe performance degradation.

In this paper, we study the same uncontrolled T-intersection navigation problem as in [3], which is shown in Fig. 1. Before entering the T-intersection, the ego vehicle needs to infer the latent driving styles of other drivers to determine whether they are willing to yield to the ego vehicle. Based on the inferred driving style, the ego vehicle must intercept the drivers who will yield to achieve the goal.

We seek to create a pipeline that first learns a representation of driver traits in an unsupervised way, and then uses the trait representation to improve navigation in the T-intersection. In the first stage, we encode the trajectories of other drivers to a latent representation using a variational autoencoder (VAE) with recurrent neural networks (RNN). Since trajectory sequences reveal richer information about driver traits than single states, the VAE+RNN network effectively learns to distinguish between different traits without any explicit trait labels. In the second stage, we use the latent representations of driver traits as inputs to the ego vehicle's navigation policy, which is trained with model-free reinforcement learning (RL). With the inferred traits, the ego vehicle is able to adjust its decisions when dealing with different drivers, which leads to improved performance. Since the RL policy is trained with the learned representations instead of ground truth labels, our pipeline is much less sensitive to cascading errors from the trait inference network.

We present the following contributions: (1) We propose a novel unsupervised approach to learn a representation of driver traits with a VAE+RNN network; (2) We use the learned representation to improve the navigation of an autonomous vehicle through an uncontrolled T-intersection; (3) Our trait representation and navigation policy exhibit promising results and outperform previous works.

This paper is organized as follows: We review related works in Section II. We formalize the problem and propose our method in Section III. Experiments and results are discussed in Section IV. We conclude the paper in Section V.

II. Related Works

A. Driver internal state estimation

Driver internal state estimation can be divided into *intent estimation* and *trait estimation* [10]. Intent estimation often uses methods such as probabilistic graph model and unparameterized belief tracker to predict the future maneuvers of other drivers [6], [11], [12], which can then inform downstream planning for the ego driver [6], [12], [13]. Trait estimation infers the properties of drivers such as driving styles, preferences, fatigue, and level of distraction [10]. Some works distinguish between distracted and attentive drivers for behavior prediction and cooperative planning [14], [15]. Driving style recognition has been addressed with both unsupervised and supervised learning methods, which we will discuss in detail below [3], [8], [16], [17].

Morton *et al* propose a method that first encodes driving trajectories with different driving styles to a latent space. Then, the latent encodings and the current driver states are fed into a feedforward policy that produces multimodal actions [8]. The encoder and the policy are optimized jointly. However, since the feedforward policy only considers the relations between current states and actions, the joint optimization encourages the encoder to encode short-term information of the trajectories such as accelerations while ignoring the persistent properties such as traits. In contrast, since our method uses an RNN decoder to reconstruct the trajectories, the encoder must encode the trait information. Ma *et al* propose a graph neural network that classifies the driving styles with supervised learning, which requires large amounts of labeled data and is difficult to scale in reality [3]. In addition, since the definitions of trait properties vary by individuals and cultures, manually labeled trait information will likely be noisy and inconsistent. In contrast, our method is unsupervised and does not need any trait labels.

B. Representation learning for sequential data

Contrastive learning is widely used to learn representations from sequential data such as videos and pedestrian trajectories [18]–[20]. However, the performance of contrastive learning is sensitive to the quality of negative samples, and finding efficient negative sampling strategies remains an open challenge [21].

Another category of representation learning methods is VAE and its variants [22], [23]. Bowman *et al* introduce an RNN-based VAE to model the latent properties of sentences [24], which inspires us to learn the traits of drivers from their trajectories. Conditional VAEs (CVAE) are widely used in pedestrian and vehicle trajectory predictions since discrete latent states can represent different behavior modes such as braking and turning [25]–[28]. While these behavior modes change frequently, the driver traits that we aim to learn are persistent with each driver [10]. Also, the goal of these CVAEs is to generate multimodal trajectory predictions while, to the best of our knowledge, our work is the first to infer driver traits with VAE for autonomous driving.

C. Autonomous driving in uncontrolled intersections

Autonomous navigation through uncontrolled intersections is well studied and has had many successful demonstrations [3], [6], [29], [30]. Some heuristic methods use a time-to-collision (TTC) threshold to decide when to cross [30], [31]. However, the TTC models assume constant velocity for the surrounding vehicles, which ignores the interactions and internal states of drivers. Also, the TTC models can be overly cautious and cause unnecessary delays [29]. Another line of works formulates the problem as a partially observable Markov decision process (POMDP), which accounts for the uncertainties and partial observability in the intersection scenario but is computationally expensive to solve [6], [32], [33]. RL-based methods use neural networks as function approximators to learn driving policies. Isele *et al* learns to navigate in occluded intersections using deep Q-learning [29]. Ma *et al* focuses on navigation in a T-Intersection where the drivers exhibit different traits [3]. However, the navigation policy is trained with ground truth traits and only uses the inferred traits in testing. Thus, the performance of the RL policy is sensitive to the intermediate errors from the trait inference module. In contrast, our method trains the RL policy directly with inferred trait representations, which eliminates the problems caused by the intermediate errors.

III. Methodology

In this section, we first present our unsupervised approach to represent driver traits from unlabeled driving trajectories with a VAE+RNN network. Then, we discuss how we use the inferred traits to improve the navigation policy.

A. Preliminaries

Consider a T-intersection environment with an ego vehicle and n surrounding vehicles. The number of surrounding vehicles n may change at any timestep t. Suppose that all vehicles move in a 2D Euclidean space. Let o_0^t be the state of the ego vehicle and o_i^t be the observable state of the i-th surrounding vehicle at time t, where $i \in \{1, ..., n\}$. The state of the ego vehicle o_0^t consists of the position (p_x, p_y) and

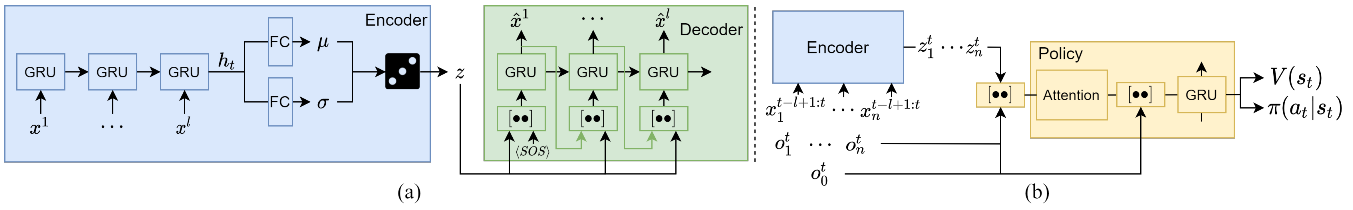


Fig. 2: **The network architectures.** (a) The VAE+RNN network for trait representation learning. The fully connected layers before and after GRUs are eliminated for clarity. The dice represents the reparameterization trick. The start-of-sequence state is denoted by $\langle SOS \rangle$. (b) The navigation policy network. The weights of the encoder (blue) is fixed and only the yellow part is trained with RL. We use $[\bullet\bullet]$ to denote concatenation. For illustration purposes, we assume that the inferred trait $z_1, ..., z_n$ is updated at time t.

the velocity (v_x, v_y) of the vehicle. The observable state of each surrounding vehicle o_i^t consists of its position (p_x^i, p_y^i). In contrast to the previous work [3], o_i^t does not include other vehicles' velocities because they are hard to accurately measure without special facilities and algorithms in the real world [34], [35]. In addition, each surrounding vehicle has a latent state z_i that indicates the aggressiveness (i.e., the trait) of the i-th driver. We assume that the driving style of each driver z_i does not change throughout an episode. The positions of surrounding vehicles $o_1^t, ..., o_n^t$ are observable to the ego vehicle, while the latent states $z_1, ..., z_n$ are not.

B. Trait representation learning

1) Data collection: For each vehicle in the T-Intersection, the vehicle in front of it has the most direct influence on its behaviors. For this reason, in the trait representation learning stage, we define the observable state of each vehicle to be $x = (\Delta p_x, \Delta p_{x,f})$, where Δp_x is the horizontal offset from the vehicle's starting position in the trajectory, and $\Delta p_{x,f}$ is the horizontal displacement of the vehicle from its front vehicle. Then, the trajectory of each driver is a sequence of states $\mathbf{x} = [x^1, ..., x^l]$, where l is the length of the trajectory.

We only keep the longitudinal state information because all vehicles move in horizontal lanes and the lateral states are not insightful in this setting, except indicating which lane the vehicle is in. We rotate the trajectories so that all trajectories in the dataset are aligned in the same direction. Thus, the lane and the directional information is indistinguishable within the trajectory data, allowing the network to focus on learning the trait difference instead of other differences between vehicles.

To collect the trajectory data, we run a simulation of the T-intersection scenario without the presence of the ego car and record the trajectories of all surrounding vehicles, which are controlled by the Intelligent Driver Model (IDM) [36]. Learning trait representations from this dataset allows the ego car to infer the traits from the trajectories of other drivers before deciding to intercept or wait. The dataset is denoted as $\{\mathbf{x}_j\}_{j=1}^N$, where N is the total number of trajectories.

2) Network architecture: We use the collected dataset to train the VAE+RNN network to learn a representation of traits, as shown in Fig. 2a. The VAE network consists of an encoder, which compresses a trajectory $\mathbf{x}$ to a distribution of a latent trait vector z, and a decoder, which reconstructs the trajectory from the latent vector z. Both the encoder and the decoder are gated recurrent unit (GRU) networks since GRUs are more computationally efficient than long short-term memory networks (LSTM).

Given a trajectory $\mathbf{x} = [x^1, ..., x^l]$, the encoder GRU first applies a non-linear embedding layer f_{encoder} to each state x^t and then feeds the embedded features to the GRU cell:

$$h_e^t = \text{GRU}\left(h_e^{t-1}, f_{\text{encoder}}(x^t)\right) \tag{1}$$

where h_e^t is the hidden state of the encoder GRU at time $t \in \{1, ..., l\}$. After the entire trajectory is fed through the encoder GRU, we take the last hidden state h_e^l as the encoded latent feature of the trajectory $\mathbf{x}$ and apply fully connected layers f_μ and f_σ to get the Gaussian parameters of the latent driving style $z \in \mathbb{R}^2$ in a two-dimensional latent space:

$$\mu = f_\mu(h_e^t), \quad \sigma_i = f_\sigma(h_e^t). \tag{2}$$

Finally, we use the reparameterization trick to sample z from $\mathcal{N}(\mu, \sigma)$ for efficient learning: $z = \mu + \epsilon\sigma, \epsilon \sim \mathcal{N}(0, I)$.

In the decoding stage, since the driving style of each driver does not change over time, we treat the latent state z as part of the vehicle state instead of the initial hidden state of decoder GRU. To this end, at each timestep t, we concatenate the reconstructed state $\hat{x}^{t-1}$ from the last timestep and the latent state z from the encoder. Then, we embed the joint states using f_{decoder}, feed the embeddings into the next decoder GRU cell, and apply another embedding g_{decoder} to the next hidden state h_d^t, which outputs the next reconstructed state $\hat{x}^t$:

$$\begin{aligned} h_d^t &= \text{GRU}\left(h_d^{t-1}, f_{\text{decoder}}([\hat{x}^{t-1}, z])\right) \\ \hat{x}^t &= g_{\text{decoder}}(h_d^t). \end{aligned} \tag{3}$$

In the first timestep, we use a special start-of-sequence (SOS) state, which is similar to the start-of-sequence symbol in natural language processing to reconstruct $\hat{x}^1$ [37]. The process in Eq. 3 repeats until we reconstruct the whole trajectory $\hat{\mathbf{x}} = [\hat{x}^1, ..., \hat{x}^l]$.

The objective for training our VAE+RNN network is

$$\mathcal{L} = \beta D_{KL}\left(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, I)\right) + ||\mathbf{x} - \hat{\mathbf{x}}||_2 \tag{4}$$

where D_{KL} is the Kullback–Leibler (KL) divergence. The first term regularizes the distribution of the latent trait z to be close to a prior with a standard normal distribution. The second term is the reconstruction loss and measures the $L2$ error of the reconstructed trajectories from the original trajectories. The two terms are summed with a weight β.

By optimizing Eq. 4, our network learns latent encodings that represent the trait of each trajectory without any ground truth trait labels. Note that we also make no assumptions on the number of trait classes or the semantic meanings of the trait classes. Thus, our network has the potential to generalize to real trajectory datasets with more complex traits.

C. Navigation policy learning

We model the T-intersection scenario as a POMDP, defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O}, \mathcal{P}, \gamma \rangle$. Suppose that there are n surrounding vehicles at timestep t. We use $o_t = [o_0^t, o_1^t, ..., o_n^t] \in \mathcal{O}$ to denote the observations of the ego vehicle, where $\mathcal{O}$ is the observation space. Let $u_i^t = [o_i^t, z_i] \in \mathbb{R}^4$ be the state of the i-th surrounding vehicle. Since the ego vehicle is influenced by all surrounding vehicles, we use $s_t = [o_0^t, u_1^t, ..., u_n^t] \in \mathcal{S}$ to denote the state of the POMDP, where $\mathcal{S}$ is the state space. And $\mathcal{P} : \mathcal{S} \rightarrow \mathcal{O}$ is the set of conditional observation probabilities.

At each timestep t, the ego vehicle chooses the desired speed for the low-level controller $a_t \in \mathcal{A}$ according to its policy $\pi(a_t|s_t)$, where $\mathcal{A} = \{0, 0.5, 3\}$m/s is the action space. In return, the agent receives a reward r_t and transits to the next state s_{t+1} according to an unknown state transition $\mathcal{T}(\cdot|s_t, a_t)$. Meanwhile, all other vehicles also take actions according to their policies and move to the next states with unknown state transition probabilities. The episode continues until t exceeds the maximum episode length T, the ego vehicle reaches its goal, or the ego vehicle collides with any other vehicle. The goal of the agent is to maximize the expected return, $R_t = \mathbb{E}[\sum_{i=t}^{T} \gamma^{i-t} r_i]$, where γ is a discount factor. The value function $V^\pi(s)$ is defined as the expected return starting from s, and successively following policy π.

To handle the unknown transitions and environment complexity, we train our policy network illustrated in Fig. 2b using model-free deep RL. During RL training, we fix the trainable weights of the encoder. For every l timesteps, we feed the trajectories of the surrounding vehicles over the past l timesteps to the encoder and infer the driving style z_i^t of each driver i. To improve efficiency, we only update the latent states of the drivers in the lanes that the ego car has not passed yet, as shown in Fig. 5. Since the ego car is not allowed to move backward, the latent states of the drivers that the ego car has already passed or the drivers that have already yielded to the ego car have no effect on the ego car's decisions and thus are not updated anymore.

Our policy network is a GRU with an attention module. We concatenate the state of each driver u_i^t with the state of the ego vehicle o_0 to obtain $q_i^t = [u_i^t, o_0^t]$, where $q_i^t \in \mathbb{R}^8$ and $i \in \{1, ..., n\}$. We feed each concatenated state q_i^t into an attention module which assigns attention weights to each surrounding vehicle. Specifically, we embed q_i^t using a multi-layer perceptron (MLP) denoted as f_{emb} and calculate the mean of the embeddings of each surrounding vehicle:

$$m^t = \frac{1}{n} \sum_{i=1}^{n} e_i^t, \quad e_i^t = f_{emb}(q_i^t) \tag{5}$$

where m^t is the resulting mean. The weighted feature of each surrounding vehicle c_i^t is calculated by

$$c_i^t = \alpha_i^t \cdot e_i^t, \quad \alpha_i = f_{attn}([e_i^t, m^t]) \tag{6}$$

where $\alpha_i^t \in \mathbb{R}$ is the attention score for the i-th vehicle, and f_{attn} is another MLP. We then concatenate the sum of $c_1^t, ..., c_n^t$ with the state of the ego vehicle o_0^t and feed the result to a GRU:

$$h_\pi^t = \text{GRU}\left(h_\pi^{t-1}, \left[\sum_{i=1}^{n} c_i^t, o_0^t \right] \right) \tag{7}$$

Finally, the hidden state of the GRU h_π^t is fed to a fully connected layer to obtain the value $V(s_t)$ and the policy $\pi(a_t|s_t)$. We use Proximal Policy Optimization (PPO), a model-free policy gradient algorithm, for policy and value function learning [38].

IV. Experiments and Results

In this section, we first describe the simulation environment for training. We then present our experiments and results for trait representation and navigation policy.

A. Simulation environment

Fig. 5 shows our simulation environment adapted from [3]. At the beginning of an episode, the surrounding vehicles are randomly placed in a two-way street with opposite lanes and we assume that they never take turns or change lanes. The number of surrounding vehicles varies as vehicles enter into or exit from the T-intersection. We assume that all cars can always be detected and tracked. The surrounding vehicles are controlled by IDM [36]. Conservative drivers vary their front gaps from the preceding vehicles between $0.5m$ and $0.7m$ and have the desired speed of $2.4m/s$. Aggressive drivers vary their front gaps between $0.3m$ and $0.5m$ and have the desired speed of $3m/s$. If the ego car moves in front of other cars, conservative drivers will yield to the ego car, while aggressive drivers will ignore and collide with the ego car.

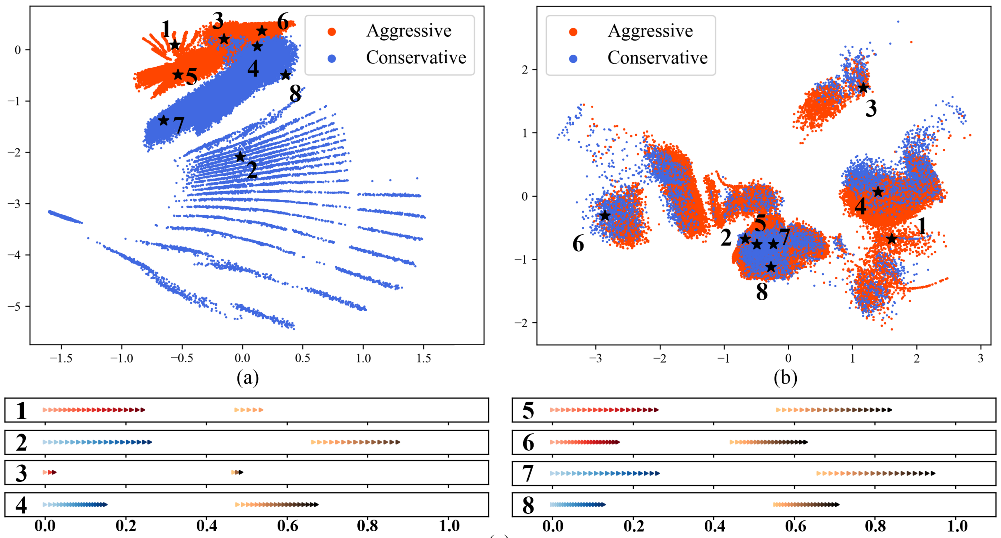
The ego car starts at the bottom of the T-intersection. The goal of the ego car is to take a right turn to merge into the upper lane without colliding with other cars. The ego car is controlled by a longitudinal PD controller whose desired speed is set by the RL policy network. The right-turn path of the ego car is fixed to follow the traffic rule. The ego car also has a safety checker that clips the magnitude of its acceleration if it gets dangerously close to other cars.

Let S_{goal} be the set of goal states, where the ego vehicle successfully makes a full right-turn, and S_{fail} be the set of failure states, where the ego vehicle collides with other vehicles. Let $r_{speed}(s) = 0.05 \times \|v_{ego}\|_2$ be a small reward on the speed of the ego vehicle and $r_{step} = -0.0013$ be a constant penalty that encourages the ego vehicle to reach the goal as soon as possible. The reward function is defined as

$$r(s,a) = \begin{cases} 2.5, & \text{if } s \in S_{goal} \\ -2, & \text{if } s \in S_{fail} \\ r_{speed}(s) + r_{step}, & \text{otherwise.} \end{cases} \tag{8}$$

B. Trait inference

1) Baseline: We compare the latent representation of our method with the method proposed by Morton *et al* [8]. The encoder of the baseline is the same as our encoder



(c)

Fig. 3: **Visualizations of the latent representations of two driver traits.** (a) Our method. (b) The method by Morton *et al.* (c) The original processed trajectories corresponding to the labeled latent vectors in (a) and (b). The x-axis is the horizontal displacement in meters. Each triangle marker indicates the position of a car at each timestep and the markers become darker over time. Denser triangles indicate smaller velocities. The blue and red trajectories indicate conservative and aggressive cars respectively. The brown trajectories indicate the front cars. If the front car goes out of the boundary before the trajectory ends, the brown trajectories will be shorter than the red or blue trajectories, such as #1 and #2.

TABLE I: Testing results of simple supervised classifiers with learned latent representations

Method	Accuracy
Ours	**98.08%**
Morton *et al.*	60.22%

except that the baseline takes the longitudinal acceleration at each timestep as an extra input. The policy network of the baseline is a 4-layer multilayer perceptron (MLP) network that imitates the IDM policy from the trajectory dataset.

2) Training and evaluation: Our dataset contains approximately $696,000$ trajectories from both classes, and the train/test split ratio is 2:1. We train both methods for 1000 epochs with a decaying learning rate 5×10^{-4}. The weight of the KL divergence loss β is 5×10^{-8} for both methods.

To better understand the learned latent representation and how it provides trait information to the navigation policy, we fix the trainable weights of the encoders, train linear support vector classifiers using the inferred latent states as inputs, and record the testing accuracy of the classifiers. To visualize the learned representations, we feed a set of testing trajectories into the encoder and visualize their latent representations.

3) Results and interpretability: In Table I, the supervised classifier with our latent representation achieves much higher classification accuracy than that of Morton *et al*. Together with Fig. 3a, we show that our representation successfully separates the vehicle trajectories with different traits in most cases. For example, in Fig. 3c, #1, #5, and #6 are in the aggressive cluster, while #2, #7, and #8 are in the conservative cluster. The overlapped region encodes the trajectories with ambiguous traits, such as very short trajectories (#3) and the trajectories with ambiguous front gaps. For example, the average front gap of #4 is between the aggressive #6 and the conservative #8. Besides the binary traits, our latent representation also captures other meaningful information. First, the trajectories of the cars whose front cars go out of the border, such as #1 and #2, are mapped into the fan-shaped peripherals of the two clusters respectively. Second, in the central regions of the two clusters, the trajectories with larger average speeds are mapped in the lower left part (#5 and #7), while those with smaller average speeds are in the upper right part (#6 and #8).

From Table I and Fig. 3b, the baseline suffers from severe model collapse and the representation fails to separate the two traits. The reason is that the MLP policy only considers current state-action pairs, which encourages the encoder to only encode features within short time windows such as accelerations while ignoring the properties of the trajectories such as traits. Despite the model collapse, the baseline still learns some meaningful representations. For example, the trajectories with uniform speeds together such as #5 and #7 and forms separate clusters for the decelerating trajectories such as #6. Therefore, we conclude that compared with single states, trajectories exhibit richer information about traits and are better suited for trait representation learning.

C. Navigation with inferred traits

1) Baselines and ablations: We use the following two baselines: (1) The pipeline proposed by Ma *et al*, which trains a supervised trait predictor and an RL policy with binary ground truth trait labels separately and combines them at test time [3]; (2) We use the latent representation by Morton *et al* to train a policy network using the same method described in Sec. III-C. In addition, we use an RL policy trained with ground truth labels as an oracle, and another RL policy trained without any trait information as a naïve model. To examine the effectiveness of the attention mechanism, we train an ablated model of our method without the attention module. For a fair comparison, the architectures of all policy networks are kept the same.

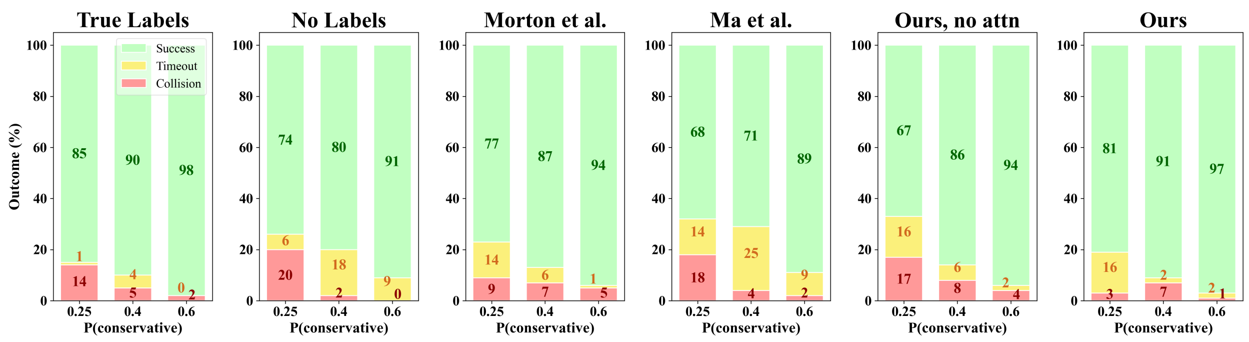


Fig. 4: **Success, timeout, and collision rates w.r.t. different driver trait distributions.** $P(\text{conservative})$ is the probability for each surrounding driver to be conservative. The task difficulty increases as $P(\text{conservative})$ decreases. The numbers on the bars indicate the percentages of the corresponding bars.

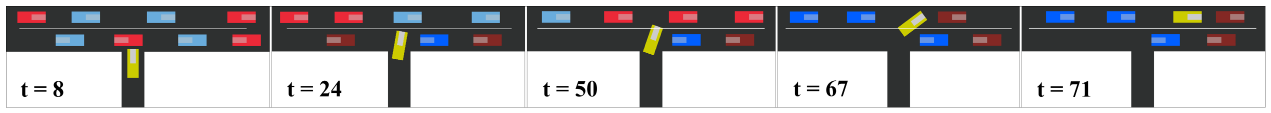


Fig. 5: **Qualitative result of our method.** The ego car is in yellow, the conservative cars are in blue, and the aggressive cars are in red. As mentioned in Sec. III-C, the latent traits of the light blue and light red cars are updated periodically, while those of the dark blue and dark red cars are not updated and stay the same as before.

2) Training: We run three experiments with different proportions of two types of drivers, as shown in Fig. 4. We train the policy networks for all methods and ablations for 1×10^7 timesteps with a decaying learning rate 1×10^{-4}. To accelerate and stabilize training, we run twelve instances of the simulation environment in parallel for collecting the ego car's experiences. At each policy update, 30 steps of 6 episodes are used. For Ma *et al*, we pretrain a trait classification network with a 96% testing accuracy and use the classifier to infer traits for the RL policy.

3) Evaluation: We test all models with 500 random unseen test cases. We measure the percentage of success, collision, and timeout episodes as the evaluation criteria.

4) Results: As shown in Fig. 4, the success rates of our method are closest to the oracle who has access to true trait labels in all experiments, with an average difference of 2%. We believe the main reason is that our trait representation effectively captures the trait differences of the surrounding drivers and aids the decision-making in RL. The performance gap between the oracle and our method is caused by the drivers with ambiguous traits, as well as the fact that the learned representation is noisier than the true labels. Although the model with no labels can implicitly infer traits to some extent in RL training, our policy is able to utilize the existing trait representation and focuses more on the decision-making of the ego vehicle, which leads to better navigation performance. Fig. 5 shows a successful episode of our method, where the ego car waits until a conservative car appears, cuts in the front of the conservative cars when passing both lanes, and completes the right turn.

Compared with our model, the model by Morton *et al* has a lower success rate especially in more challenging experiment settings when $P(\text{conservative})$ is smaller. The reason is that the latent representation does not distinguish between different traits and only provides very limited useful information to RL. This implies that the policy still needs to implicitly infer the traits while being distracted by the latent representation, which negatively affects the performance.

For Ma *et al*, the trait classifier and the RL policy both have good performance when tested separately. However, when the two modules are combined together, intermediate and cascading errors significantly lower the success rates. The performance drop is due to the distribution shift between the true traits and inferred traits. Since the policy is trained with true traits, it fails easily whenever the trait classifier makes a small mistake. Moreover, Ma *et al* requires trait labels, but our trait representation is learned without any labels and still outperforms Ma *et al* in navigation.

D. Attention vs. no attention

The second from the rightmost graph in Fig. 4 shows that the removal of the attention module results in $3\% \sim 14\%$ lower success rates. Since the attention module assigns different weights to the cars with different traits and in different positions, the policy is able to focus on the cars which have a bigger influence on the ego car, which validates the necessity of the attention mechanism.

V. Conclusions and Future Work

We propose a novel pipeline that encodes the trajectories of drivers to a latent trait representation with a VAE+RNN network. Then, we use the trait representation to improve the navigation of an autonomous vehicle through an uncontrolled T-intersection. The trait representation is learned without any explicit supervision. Our method outperforms baselines in the navigation task and the trait representation shows interpretability. Possible directions to explore in future work include (1) validating our method with real driving trajectory data, (2) generalizing our model to more sophisticated driver internal states and more navigation tasks, and (3) incorporating occlusions and limited sensor range of the ego vehicle to close the gap between the simulation and the real world.

Acknowledgements

We thank Xiaobai Ma for thoughtful discussions and for providing the baseline code. We thank Zhe Huang and Tianchen Ji for feedback on paper drafts.

References

[1] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kümmerle, H. Königshof, C. Stiller, A. de La Fortelle, and M. Tomizuka, "INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps," *arXiv preprint arXiv: 1910.03088*, 2019.

[2] Z. N. Sunberg, C. J. Ho, and M. J. Kochenderfer, "The value of inferring the internal state of traffic participants for autonomous freeway driving," in *American Control Conference (ACC)*, 2017, pp. 3004–3010.

[3] X. Ma, J. Li, M. J. Kochenderfer, D. Isele, and K. Fujimura, "Reinforcement learning for autonomous driving with latent state inference and spatial-temporal relationships," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[4] "Traffic safety facts 2019," National Highway Traffic Safety Administration, Tech. Rep. DOT HS 813 141, 2021.

[5] B. Qian, H. Zhou, F. Lyu, J. Li, T. Ma, and F. Hou, "Toward collision-free and efficient coordination for automated vehicles at unsignalized intersection," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 408–10 420, 2019.

[6] W. Song, G. Xiong, and H. Chen, "Intention-aware autonomous driving decision-making in an uncontrolled intersection," *Mathematical Problems in Engineering*, vol. 2016, 2016.

[7] S. Liu, P. Chang, W. Liang, N. Chakraborty, and K. Driggs-Campbell, "Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 3517–3524.

[8] J. Morton and M. J. Kochenderfer, "Simultaneous policy learning and latent state inference for imitating driver behavior," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–6.

[9] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2118–2125.

[10] K. Brown, K. Driggs-Campbell, and M. J. Kochenderfer, "A taxonomy and review of algorithms for modeling and predicting human driver behavior," *arXiv preprint arXiv:2006.08832*, 2020.

[11] C. Dong, J. M. Dolan, and B. Litkouhi, "Intention estimation for ramp merging control in autonomous driving," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1584–1589.

[12] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 454–460.

[13] K. Driggs-Campbell, V. Govindarajan, and R. Bajcsy, "Integrating intuitive driver models in autonomous planning for interactive maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 12, pp. 3461–3472, 2017.

[14] K. Driggs-Campbell, V. Shia, and R. Bajcsy, "Improved driver modeling for human-in-the-loop vehicular control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1654–1661.

[15] D. Sadigh, N. Landolfi, S. S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state," *Autonomous Robots*, vol. 42, no. 7, pp. 1405–1426, 2018.

[16] C. M. Martinez, M. Heucke, F.-Y. Wang, B. Gao, and D. Cao, "Driving style recognition for intelligent vehicle control and advanced driver assistance: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 666–676, 2017.

[17] W. Dong, T. Yuan, K. Yang, C. Li, and S. Zhang, "Autoencoder regularized network for driving style representation learning," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017, p. 1603–1609.

[18] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *International Conference on Computer Vision (ICCV)*, 2015, pp. 2794–2802.

[19] I. Misra, C. L. Zitnick, and M. Hebert, "Shuffle and learn: unsupervised learning using temporal order verification," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 527–544.

[20] Y. Liu, Q. Yan, and A. Alahi, "Social nce: Contrastive learning of socially-aware motion representations," in *International Conference on Computer Vision (ICCV)*, 2021.

[21] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent - a new approach to self-supervised learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 21 271–21 284.

[22] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations(ICLR)*, 2014.

[23] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015, pp. 3483–3491.

[24] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *arXiv preprint arXiv:1511.06349*, 2015.

[25] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *European Conference on Computer Vision (ECCV)*, 2020, pp. 683–700.

[26] B. Ivanovic, K. Leung, E. Schmerling, and M. Pavone, "Multimodal deep generative models for trajectory prediction: A conditional variational autoencoder approach," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 295–302, 2020.

[27] X. Feng, Z. Cen, J. Hu, and Y. Zhang, "Vehicle trajectory prediction using intention-based conditional variational autoencoder," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2019, pp. 3514–3519.

[28] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, "Multimodal probabilistic model-based planning for human-robot interaction," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3399–3406.

[29] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2034–2039.

[30] A. Cosgun, L. Ma, J. Chiu, J. Huang, M. Demir, A. M. Anon, T. Lian, H. Tafish, and S. Al-Stouhi, "Towards full automated drive in urban environments: A demonstration in gomentum station, california," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1811–1818.

[31] M. M. Minderhoud and P. H. Bovy, "Extended time-to-collision measures for road traffic safety assessment," *Accident Analysis & Prevention*, vol. 33, no. 1, pp. 89–97, 2001.

[32] M. Bouton, A. Cosgun, and M. J. Kochenderfer, "Belief state planning for autonomously navigating urban intersections," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 825–830.

[33] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2014, pp. 392–399.

[34] K.-H. Lee, "A study on distance measurement module for driving vehicle velocity estimation in multi-lanes using drones," *Applied Sciences*, vol. 11, no. 9, p. 3884, 2021.

[35] K. Han, E. Lee, M. Choi, and S. B. Choi, "Adaptive scheme for the real-time estimation of tire-road friction coefficient and vehicle velocity," *IEEE/ASME Transactions on mechatronics*, vol. 22, no. 4, pp. 1508–1518, 2017.

[36] A. Kesting, M. Treiber, and D. Helbing, "Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1928, pp. 4585–4605, 2010.

[37] E. Grefenstette, K. M. Hermann, M. Suleyman, and P. Blunsom, "Learning to transduce with unbounded memory," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, pp. 1828–1836, 2015.

[38] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.