Learning a Decision Module by Imitating Driver's Control Behaviors

Junning Huang[1,*] Sirui Xie[2,*] Jiankai Sun[5,*] Qiurui Ma[4] Chunxiao Liu[3] Dahua Lin[5] Bolei Zhou[5]

[1]Technische Universität Darmstadt [2]University of California, Los Angeles [3]SenseTime Research
[4]Hong Kong University of Science and Technology [5]The Chinese University of Hong Kong

Abstract: Autonomous driving systems have a pipeline of perception, decision, planning, and control. The decision module processes information from the perception module and directs the execution of downstream planning and control modules. On the other hand, the recent success of deep learning suggests that this pipeline could be replaced by end-to-end neural control policies, however, safety cannot be well guaranteed for the data-driven neural networks. In this work, we propose a hybrid framework to learn neural decisions in the classical modular pipeline through end-to-end imitation learning. This hybrid framework can preserve the merits of the classical pipeline such as the strict enforcement of physical and logical constraints while learning complex driving decisions from data. To circumvent the ambiguous annotation of human driving decisions, our method learns high-level driving decisions by imitating low-level control behaviors. We show in the simulation experiments that our modular driving agent can generalize its driving decision and control to various complex scenarios where the rule-based programs fail. It can also generate smoother and safer driving trajectories than end-to-end neural policies. Demo and code are available at `https://decisionforce.github.io/modulardecision/`.

1 Introduction

An autonomous driving system is essentially a decision-making system that takes a stream of on-board sensory data as input, processes it with prior knowledge about driving scenarios, and then make a reasonable *decision*, and outputs *control* signals to steer the vehicle [1]. Such a system is often modularized into perception, decision, planning, and control. While the perception and its auxiliary, map reconstruction, tend to be more open to machine learning methods [2], the downstream modules such as *decision*, *planning* and *control* remain as program-based to ensure the safe interaction with the physical world. In other words, this safety guarantee is built upon a human-inspectable and interruptible basis. However, in complex driving environments, it is extremely difficult to build a complete rule-based decision-making system. So many corner cases are out there such that a significant program refactorization is required when one corner case is inconsistent with the existing rules. Towards a more flexible alternative, we explore a learning-based decision-making module whose downstream modules, i.e. planning and control, remain encapsulated. Essentially, we draw inspiration from a prior work [3], which, in stark contrast to end-to-end control policies, learns a decision module to switch between controllers designed with Lyapunov domain knowledge.

The primary challenge we have for a learning-based decision module is the source of supervision. *Driving decision* is defined as the high-level abstraction about what lane the driver wants the vehicle to be with at which velocity in T seconds. For an autonomous driving system, this decision determines the execution of downstream planning and control modules. For a human driver, however, it is both difficult and ambiguous to describe such a driving decision. Imagine two drivers try to merge into traffic at one roundabout, they may have different habitual behaviors to execute. Even the same individual may make different decisions in almost identical scenarios. To this end, there

[*]indicates equal contribution.

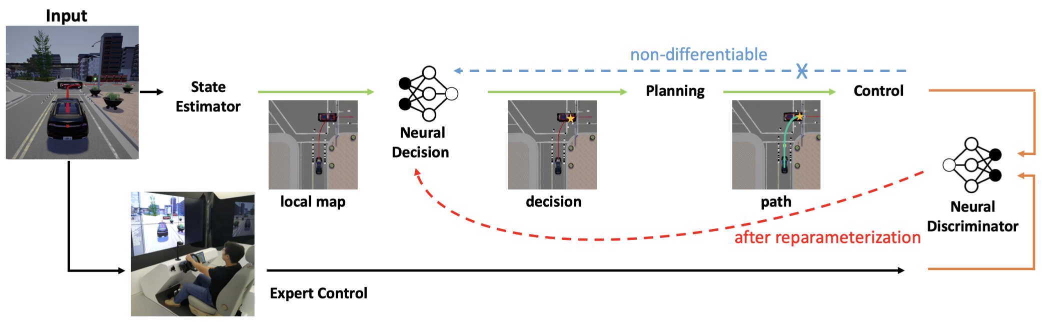


Figure 1: **Modular decision framework.** The green line shows the modular pipeline of the autonomous driving system. Solid orange lines indicate the offline training of the neural discriminator, while the dashed blue line indicates that the planning and control modules are not differentiable. But the decision policy can be trained with reward assigned on control action by the discriminator after reparameterization, as indicated by the dashed red line.

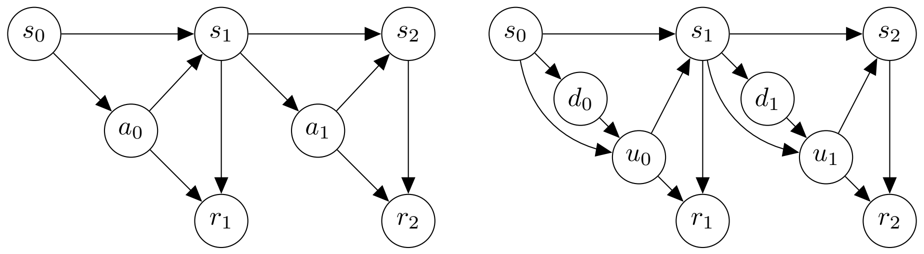
is hardly a set of golden criteria to evaluate the quality of human-annotated driving decisions. In contrast, drivers' physical behaviors, such as stepping on throttle/brake and steering the wheel, implicitly reflect their high-level decisions. Even though it is only a weak supervision in the sense that the decision is not directly supervised, it is easily accessible and technically reliable.

The setup of weak and indirect supervision requires a mechanism to infer human intentions from their behaviors and imitate these cognitive decisions. Intention understanding and behavior imitation have been extensively studied in the field of artificial intelligence and cognitive science, under the banner of Inverse Planning [4] and Inverse Reinforcement Learning [5]. The former one mainly focuses on inferring the unobservable intention from the observation of human behaviors in a third-person view [6]; The latter one adopts end-to-end neural policies and does not distinguish intentions from actions [7, 8]. Henceforth, neither of them discussed the formulation of indirect supervision as in this work. More specifically, our explicit representation of intentions and modular pipeline is appealing to physical systems like autonomous driving since the end-to-end neural counterparts do not guarantee the fulfillment of low-level constraints in states or actions [9]. Nor do they have robustness or stability guarantees [10], which are of great importance to the real-world deployment of classical controllers. Given human driving behaviors, the expected framework should learn a high-level decision module in the hierarchical pipeline where downstream modules inherit the merits of the classical paradigm. Due to the non-differentiablity of these downstream programs, a tailored learning scheme is needed.

In this work, we propose an imitation learning framework for a modular driving pipeline that *learns neural decisions from human behavioral demonstrations*. The learning is conducted in a generative-adversarial manner [11]. The generator (green line in Fig.1) simulates the modularized driving pipeline. At the upstream, a neural decision module generates decisions according to the information from a local map. Decisions are then passed into the programmed downstream modules for planning and control. The generation process ends at the control module, where the interaction with the environment is triggered. For adversarial training (orange lines in Fig.1), the neural discriminator takes in the generated trajectories and compares them with drivers' behavior data. Due to the blockage of gradients at the planning and control modules (blue dashed line in Fig.1), we derive a novel learning objective to propagate credits of control actions back to the corresponding decisions (red line in Fig.1). This learning framework is hence agnostic to the mechanisms of perception, planning, and control modules. We evaluate this framework on various simulated urban driving scenarios in CARLA, including (i) following, (ii) merging at the crossing, (iii) merging at the roundabout, and (iv) overtaking. The learning-based modular system demonstrates the superior performance over rule-based modular systems and end-to-end control policies.

We summarize our contributions as:

- The proposed framework combines a program-based system with a generative learning method, where the high-level decision policy is data-driven while the low-level planning and control modules remain configurable and physically constrained. These low-level constraints also improve sample efficiency in interactive learning.
- The proposed generative adversarial learning method is weakly supervised. It learns high-level driving decisions from low-level control data in an end-to-end manner, circumventing the ambiguous annotation of human driving decisions.



(a) Markov Decision Process (b) Markov Modular Decision Process

Figure 2: Comparison between *Markov Decision Process* and *Markov Modular Decision Process*

- This framework successfully distills behaviors in different scenarios into one decision module. Empirical results demonstrate that the learned decision module forms synergy with the programmed downstream modules, endowing the vehicle with smoother driving trajectories. It also exhibits substantial generalization capability in novel and complex scenarios.

2 Related Work

Classical autonomous driving system. The design of the autonomous driving pipeline could be traced back to DARPA Urban Challenge 2007 [12]. The survey from [1] provides an overview of the hierarchical structure of perception, decision, planning, and control. Specifically, finite state machine (FSM) was used in the decision level, or equivalently upper planning level, in [13]. Classical autonomous driving systems are organized in this way such that it is accessible to testing and diagnosis. Based on this modular pipeline, our work aims at developing a data-driven decision module to replace the FSM.

Imitation learning-based autonomous driving system. With the popularity of deep generative learning, methods have been proposed to learn end-to-end neural control policies from driver's data. Previously, Ziebart *et al.* [8] and Ross *et al.* [7] proposed general methods in Inverse Reinforcement Learning and Interactive Learning from Demonstration, with an empirical study on a driving game. More recently, Kuefler *et al.* [14] and Behbahani *et al.* [15] learn an end-to-end policy in a GAIL[16]-like manner. Codevilla *et al.* [17] and Liang *et al.* [18] share similar hierarchical perspective as us, but their control policies are completely neural. One thing worth mentioning is Codevilla *et al.* [17] proposed to learn a downstream policy through the imitation of human's high-level command, whose motivation is similar to ours. Rhinehart *et al.* [19] propose to learn a generative predictor for vehicles' coordinate sequence. Li *et al.* [20] imitate the mapping from predicted waypoints to control commands via neural network policy. Our work differentiates from all of them as we imitate the driver's low-level behaviors to learn high-level decisions, whose execution is conducted by a transparent and configurable program.

Learning human intention. Baker *et al.* [4] formalized humans' mental model of planning with Markov Decision Process (MDP) and proposed to learn human's mental states with Bayesian inverse planning. A similar probabilistic model of decisions is adopted by us but the learning is done with a novel generative adversarial method. Wang *et al.* [6] proposed a Hidden Markov Model (HMM) for human behaviors and materialized it with Gaussian Process (GP), where the intent is the hidden variable. Jain *et al.* [21] model the interaction between human intentional maneuvers and their behaviors in a driving scenario, both in-car and outside, as an Auto-Regressive Input-Output HMM and learn it with EM algorithm. However, they didn't explicitly distinguish humans from the environment. Besides, rather than inferring the human's mental state, we focus more on how a robot, *i.e.* can make good decisions.

3 Modular Driving System

3.1 System Overview

Fig.1 illustrates the proposed learning framework. We focus on the decision-making module and its connection to the downstream planning and control modules. The *state estimator* (usually a perception module) constructs a local map by processing the sensory data and combining them with prior knowledge from the map and rules. The *decision* module then receives *observation* and decides a legal local driving task that steers the car towards the destination. To complete this task, the *planning* module searches for an optimal trajectory under enforced physical constraints. The *control* module then reactively corrects any errors in the execution of the planned motion.

In our proposed framework, the *decision* module is a conditional probability distribution parameterized by neural networks. The downstream *planning* and *control* module are encapsulated. Because they are not the main contribution of this work, we adopt minimal settings to stay focused on our learning frameworks. These capsulated modules could be replaced by arbitrary alternatives.

3.2 Decision Making Module

We assume that a local map centered at the ego vehicle could be constructed from either the state estimator or an offline global HD map. This local map contains critical driving information such as the routing guidance to the destination, legal lanes, lane lines, other vehicle coordinates and velocity in the ego vehicle's surroundings, as well as ego vehicle's speed and acceleration.

The interaction between the decision module and the environment through its downstream is modeled as Markov Decision Process (MDP). MDP is normally represented as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{D}, \mathcal{P}, r, \rho_0, \gamma)$, with a state set $\mathcal{S}$, a decision set $\mathcal{D}$, a transitional probability distribution $\mathcal{P} : \mathcal{S} \times \mathcal{D} \times \mathcal{S} \to \mathbf{R}$, a bounded reward function $r : \mathcal{S} \times \mathcal{D} \to \mathbf{R}$, an initial state distribution $\rho_0 : \mathcal{S} \to \mathbf{R}$, a discount factor $\gamma \in [0, 1]$ for infinite horizon. Decision policy $\pi_\theta : \mathcal{S} \times \mathcal{D} \to \mathbf{R}$ takes current state $s_t \in \mathcal{S}$ from local map and generates a high-level decision d_t. Note that the notation of $\mathcal{D}$ and d are unconventional, we explicitly denote *decision* with d_t to differentiate it from *control action* u_t. This decision directs the execution of planning and control module, and makes the autonomous driving system proceed in the environment to acquire next state s_{t+1}. Therefore there is a modification of the decision process, termed as Markov Modular Decision Process shown in Fig.2. The optimization objective of this policy is to maximize the expected discounted return $\mathbf{E}_\tau[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, u_t)]$, where $\tau = (s_0, d_0, u_0, ...)$ denotes the whole trajectory, $s_0 \sim \rho_0(s_0)$, $d_t \sim \pi_\theta(d_t|s_t)$, $u_t \sim P(u_t|s_t, d_t)$ and $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, u_t)$.

Below we introduce the *observation* space and *decision* space, which are the interfaces in our modular system. Note that we assume full access to necessary information on the local map.

To make the decision-making policy more generalizable to different driving scenarios, the output of the state estimator, also as the input to the neural decision module, is the local map and traffic state. As shown in Fig.3, a part of the observation is the 3D coordinates of sample points of lane lines. We select two sets of lanes for the decision policy. One for the ego vehicle's current lane and the other for the edge of *legal* regions. Note that here *legal* means driving in that region abides by both the traffic rules and a global routing, which would have a crucial effect at the crossing. To reduce the dimension of input, these lane points are sampled with exponentially increasing slots towards the further end. In some sense, it mimics the effect of Lidar, with the observation more accurate in the near end. The observation also includes the coordinates and velocities of the 6 nearest vehicles in traffic within $70m$ range of the ego vehicle.

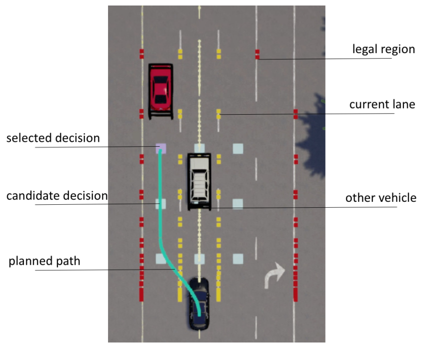


Figure 3: **Illustration of input and output of the decision module when the driving system conducts *overtake*.** The trained policy decides to pick the left lane, in terms of the target point and the target speed. The planner generates a trajectory, which is tracked by the controller. Observations are shown on the right. Yellow lines are the ego vehicle's current lane lines and the red lines are the edge of *legal* regions.

We define the *decision* as three independent categorical variables, one for lateral, one for longitudinal and one for velocity. Combining with the local map, each of them is assigned a specific semantic meaning. The lateral decision variable has three classes as *changing to the left lane*, *keeping current lane* and *changing to the right lane*. Note that at a crossing, global routing information has been implanted into the local map as introduced in the last paragraph. The lateral decision set is complete since these three are the only possible lateral decisions for a vehicle. The longitudinal decision variable has four different classes, with each indicating *the distance along with waypoints in time interval T*. The exact coordinate of the endpoint could be extracted from the local map. Combining with the predicted target speed at the endpoint from the velocity decision variable, which discretizes the allowed speed range equally into four baskets, this decision module provides a quantitative configuration of goal states for the downstream trajectory planner.

3.3 Planning and Control Modules

We introduce a minimal design of the planning module and the control module for completeness, which is not the main contribution of this work. In classical autonomous driving system, the planning module calculates the trajectories under constraints such as collision avoidance. Its optimization objective is to reach the goal state specified by the decision module with minimal cost. In our minimal setting, the planning module processes path and velocity separately. Paths are planned with cubic Bezier curves, while velocity is planned with Quadratic Programming (QP) to minimize the acceleration, jerk, and discontinuity in curvature. The control module drives the vehicle to trace the planned trajectories. And we implement it minimally with PID. Details are provided in Appx. A.

It is flexible to further extend each module of the proposed framework, as long as the alternatives are deterministic or stationary. Note that while the interface between decision and planning is the location of a goal point and the speed ego vehicle is expected to maintain when reaching it, the planning module could be replaced by more sophisticated search-based or optimization-based trajectory planner, to fulfill more practical requirements in execution time and constraint enforcement. Similarly, the model-free controller could be replaced with alternatives like Model Predictive Control (MPC). Other practical constraints and concerns over the controller such as robustness and stability could also be taken into consideration if necessary.

4 Imitation Learning

4.1 From IRL to GAIL

Under the MDP modeling, we choose imitation learning over reinforcement learning. In principle, imitation learning could save practitioners from *ad hoc* reward engineering and focus their effort on reusable infrastructure development. Among all the imitation learning methods, behavior cloning (BC) [22] is the most straightforward one. The policy is trained as a regressor in a purely supervised manner, ignoring the temporal effect of each action in an MDP trajectory. Thus it suffers from covariate shift when presented with states which are not covered by the training data. By contrast, imitation learning such as Inverse Reinforcement Learning (IRL) explicitly models the interaction between policy and environment and approximates for exact value iteration [8] or Monte Carlo approximation [16]. Thus they can generalize better in states not covered in the demonstration.

Fig.2 illustrates the difference in the graphical model of a trajectory from an MDP when the modular system is adopted for the agent. In this modified probabilistic model, a trajectory is a sequence $\tau = (s_0, d_0, u_0, s_1, d_1, u_1 ...)$. However, what is observable from demonstration is $\hat{\tau} = (s_0, u_0, s_1, u_1 ...)$. According to the formulation of MaxEnt IRL [8], when maximum entropy principle is applied for the tie breaking between cost functions that can generate identical optimal trajectories, the probability of a trajectory is

$$p(\hat{\tau}|R) = \frac{1}{Z}\exp(R(\hat{\tau})), \tag{1}$$

where Z is the partition function, R is the reward function, which is normally Markovian $R(\hat{\tau}) = \sum r(s_i, u_i)$. Given a set of demonstration $\mathcal{D}_E$, we can infer the reward function by Maximum Likelihood (MLE):

$$\mathcal{L}_{IRL} = \mathbf{E}_{\mathcal{D}_E}[\log p(\hat{\tau}|R)] = \mathbf{E}_{\mathcal{D}_E}[\sum_{\hat{\tau}} r(s_i, u_i) - \log Z]. \tag{2}$$

In the standard IRL, with the learned reward function, we can solve for the policy with any reinforcement learning methods. However, as the partition function is always intractable if the state space is continuous, methods have been proposed to approximate it [8, 23]. Basically, there is a background distribution $q(\hat{\tau})$ for the estimation of Z. In [24], it is further illustrated how a delicately designed importance sampling scheme can connect this IRL loss to a specific type of discriminator over trajectories. Ho *et al.* [16] propose Generative Adversarial Imitation Learning (GAIL) to further approximate this discriminator with a step-wise one, remedy the trajectory-wise information with advantage accumulation and learn a policy with off-the-shelf reinforcement learning methods. Intuitively, it learns a policy whose actions at given states are indistinguishable from demonstration data. More formally, with expert demonstration $\mathcal{D}_E$, a neural discriminator $D_\phi : \mathcal{S} \times \mathcal{U} \rightarrow (0,1)$ is introduced and the reward function is defined as $r(s,u) = -\log(D_\phi(s,u))$. The training objective is:

$$\mathcal{L}_{GAIL} = \min_\theta \max_\phi \{\mathbf{E}_{p_\theta(u|s)}[\log(D_\phi(s,u))] + \mathbf{E}_{\mathcal{D}_E}[\log(1 - D_\phi(s^E, u^E))]\}. \tag{3}$$

D_ϕ is optimized with cross entropy loss, while π_θ is optimized with policy gradient [25].

4.2 Handling Non-differentiable Downstream Modules

Different from GAIL, in our framework the generation process is modularized, following the structure of classical autonomous driving systems. Because the downstream planning and controls modules are not necessarily differentiable, a separation occurs between decisions from neural policy and the control data from drivers, as illustrated by the blue dotted line in Fig 1: policy π_θ generates *decisions* d_t while discriminator only distinguishes *actions* u_t. And as shown in Fig 2, $p_\theta(u_t|s_t) = \sum_{d_t} p(u_t|d_t)\pi_\theta(d_t|s_t)$. Since planning and control module are both deterministic, at every state, once the decision is chosen, the transformation from decision to control is a deterministic and correspondence mapping. Hence, one important insight of our work is that the transformation at each state s

$$g_s : \{d_t\} \xrightarrow[\text{1-to-1}]{\text{onto}} \{u_t\}, \tag{4}$$

which is the abstraction of the planning and control module, is a deterministic and correspondence mapping. Therefore, it is a candidate for *reparametrization* [26] or *push-forward* [27]. Reparametrize u in the first expectation in Eq.3 with d:

$$\begin{aligned} \mathbf{E}_{p_\theta(u|s)}[\log(D_\phi(s,u))] &= \sum p_\theta(u|s)\log(D_\phi(s,u)) \\ = \sum \pi_\theta(d|s)\log(D_\phi(s, g_s(d))) &= \mathbf{E}_{\pi_\theta(d|s)}[\log(D_\phi(s, g_s(d)))]. \end{aligned} \tag{5}$$

Intuitively, we only use this black-box function g_s for Monte Carlo sampling, thus there is no need to know its analytical form. In the training stage, we do not actually need to calculate u_t with the push-forward function as in Eq.4. Instead, we calculate the true $g_s(d_t)$ according to Sec. 3 during sampling, and save both d_t and u_t for training. During training, d_t is fed to π_θ and the corresponding u_t is fed to D_ϕ. Since $\mathcal{D}$ is discrete, it is differentiated with policy gradient. We then have this modified learning objective:

$$\mathcal{L} = \min_\theta \max_\phi \{\mathbf{E}_{\pi_\theta(d|s)}[\log(D_\phi(s,u))] + \mathbf{E}_{\mathcal{D}_E}[\log(1 - D_\phi(s^E, u^E))]\}. \tag{6}$$

The whole algorithm is described in Algorithm 1.

Algorithm 1 Learning executable decisions from human behavioral data

```
Require: Expert demonstration $\mathcal{D}_E$, batch size $B$
  Initialize policy $\pi_\theta$, discriminator $D_\phi$, (optionally) value function $V_w$
  while policy iteration not converged do
    Initialize data buffer $\mathcal{B}$ with length $B$
    while $\neg\mathcal{B}.full()$ do
      Sample expert trajectory: $\tau_E \sim \mathcal{D}_E$
      Initialize $sim$ with $s_0^E$
      $traj$ = GENERATE-TRAJ
      Append $traj$ to $\mathcal{B}$
    end while
    Compute reward for samples in $\mathcal{B}$ with $D_\phi$
    Update $D_\phi$ with data in $\mathcal{B}$ and $\mathcal{D}_E$
    Policy iteration on $\pi_\theta$ with tuples in $\mathcal{B}$, (optionally with value iteration on $V_w$)
  end while
```

Algorithm 2 GENERATE-TRAJ

```
Require: Simulator (or ROS offline replayer) with local map $sim$, planning module $planner$, control module $controller$, maximum trajectory horizon $\mathcal{H}$
  Initialize placeholder $traj$, $t = 0$
  while $t < \mathcal{H}$ or $\neg sim.done$ do
    Get $s_t$ from $sim$
    Sample $d_t$ from $\pi_\theta(o_t)$
    Get $pln_t$ from $planner(sim, d_t)$
    Get $u_t$ from $controller(sim, pln_t)$
    Append $(s_t, d_t, u_t)$ to $traj$
    Send $u_t$ to $sim$
    $t++$
  end while
  Yield $traj$
```

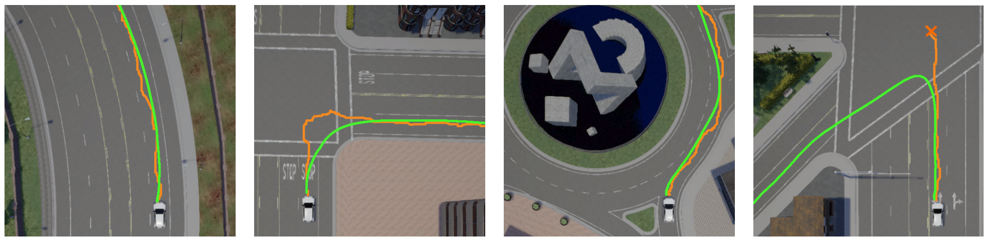


Figure 4: Trajectories from our proposed framework (green lines) are smoother than the ones from the end-to-end neural policy (orange lines), because of the explicit enforcement of geometrical constraints in the downstream planning module. ($\times$ indicates the car violates routing rules.)

5 Experiments

5.1 Rudimentary Driving in Empty Town

Experiments in *Empty Town* show the difference between a modular driving system and an end-to-end neural control policy. The training time on an 8-GPU computation server for the end-to-end control policy is 31 hours versus 15 hours for our modular driving system. Different training time in the same platform implies different numbers of interactions these two methods need to converge, showing the advantage of using the modular pipeline.

As shown in Fig.4, trajectories from our proposed framework are smoother than the ones from the end-to-end policy. This demonstrates the effect of explicit enforcement of geometrical constraints in the downstream planning module. Interestingly, there are some road structures where the learning-based modular system can pass while the end-to-end control policy cannot. For example, Fig.4(d) shows a narrow sharp turn where end-to-end control policy drives across the legal lane line. We believe this is a difficult temporally-consistent exploration problem. An agent needs some successful trials of consecutive right-turning action samples to learn, which is challenging for generative interactive learning. The modular system, in contrast, only explores behaviors that are temporally plausible like human drivers. With the planning module that searches for a geometrically constrained path, the agent can effortlessly make this sharp turn.

Table 1 shows the statistics of behaviors from the learning-based modular system, end-to-end control policy, and rule-based method. The comparison shows that our method can drive more safely (0% collision rate), much faster (less time to finish), and with higher comfort (less acceleration). This exhibits the advantage of having both the learned decision policy and the classic planning and control module. Besides, the decision module trained in Town 3 works similarly well in Town 2, showing its generalization ability.

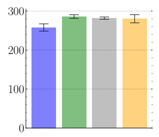
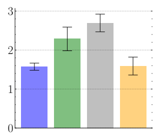
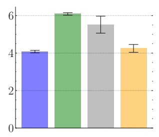
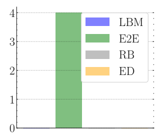
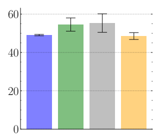
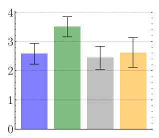
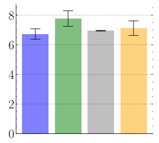
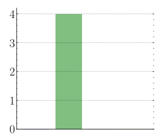
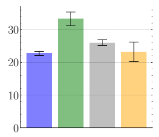
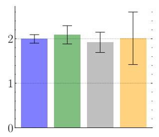
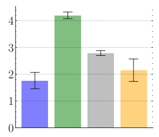
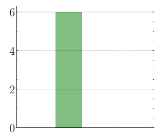
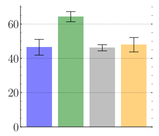
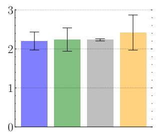
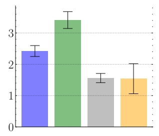
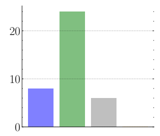
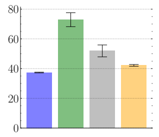
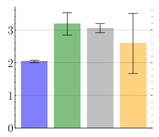
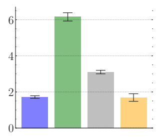
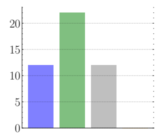
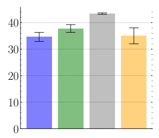
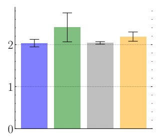
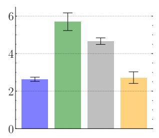
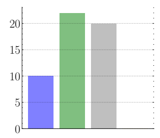
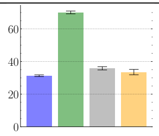
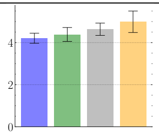
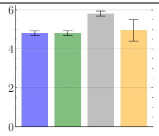
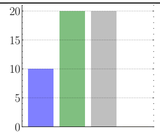
5.2 Socially Interactive Driving in Traffic Scenarios

We further test our model's performance in each traffic scenario. The statistics of Traffic Scenarios in Table 1 are the average of 100 evaluations. Traffic scenarios include basic scenarios such as *Car Following* and *Crossroad Merge* where vehicle's interactions with zombies are relatively simple and monotonic; and complex ones such as *Crossroad Turn Left*, *Roundabout Merge* and *Overtake*. In complex traffic scenarios, the dynamics around the ego vehicle is of higher variation, due to either more traffic users or more complex relations between them.

As shown in Table 1, in basic traffic scenarios, rule-based and learning-based modular systems are somehow on par in terms of time taken to finish, while rule-based agent seems to drive a little bit more comfortably. End-to-end agent drives more rudely in most of the scenarios, expect in *Crossroad Merge*, where it takes much longer time to finish. Since the modular pipeline enforces smoother planning and control, the learning-based modular system offers more driving comfort than the end-to-end system. In basic traffic scenarios, none of them drives as well as experts.

In complex traffic scenarios, rule-based agent configured in some scenarios fails to drive safely in others while learning-based modular agent is safer and smarter. This is because rule-based decision module is based on FSM, which is composed of a huge number of rules. In order to drive safely in all environments, the rules of safety have priority over other factors such as driving comfort. In complex environments, the rules in FSM would be more conservative to handle corner cases and uncertainty(*e.g.*, more hard brakes or accelerations). Unlike the rule-based system, the learning-based system can minimize the acceleration and jerk by imitating human experiences. Furthermore, since the rule-based system is program-based, its configurations might vary in different scenarios, which

Table 1: Performance Comparision (LBM: Learning-based Module, E2E: End-to-end neural policy, RB: Rule-based method, ED: Expert Data)

Scenario	**Time Taken** (s)	**Acceleration** (m/s^2)	**Jerk** (m/s^3)	**Collision Rate** $(\%)$
Empty Town				
Two Lanes Car Following				
Single Lane Car Following				
Crossroad Merge				
Roundabout Merge				
Crossroad Turn Left				
Overtake				

makes it difficult to generalize across scenarios. When compared with end-to-end control policy, a similar conclusion could be drawn as in the previous subsection that learning-based modular agent offers more comfort because the modular pipeline enforces smoother planning and control. Interestingly, in complex social scenarios, the learning-based modular system achieves higher comfort (lower acceleration) than experts. This may be attributed to human errors, and a learning-based modular agent somehow fixes it. Experiments of generalization capability can be found in Appendix.

6 Conclusion

This work introduces a flexible framework for learning modular decision-making for autonomous driving. To learn the driver's high-level driving decision, the proposed framework imitates driver's control behavior with a modular generation pipeline. Being agnostic to the design of the non-differentiable downstream planning and control modules, this framework can train the neural decision module through reparametrized generative adversarial learning. We evaluate its effectiveness in simulation environments with human driving demonstrations. This work can be extended to more complex environments where other vehicles are also learning agents, *i.e.* multi-agent learning system. Alternatively, if we can collect data in the real-world and replay them in the simulator, this framework could be tested in real driving environments.

References

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[2] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press, 2005.

[3] T. J. Perkins and A. G. Barto. Lyapunov design for safe reinforcement learning. *Journal of Machine Learning Research*, 3(Dec):803–832, 2002.

[4] C. L. Baker, R. Saxe, and J. B. Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.

[5] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[6] Z. Wang, K. Mülling, M. P. Deisenroth, H. Ben Amor, D. Vogt, B. Schölkopf, and J. Peters. Probabilistic movement modeling for intention inference in human–robot interaction. *The International Journal of Robotics Research*, 32(7):841–858, 2013.

[7] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[8] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. 2008.

[9] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 2018.

[10] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[12] M. Buehler, K. Iagnemma, and S. Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*, volume 56. springer, 2009.

[13] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[14] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer. Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 204–211. IEEE, 2017.

[15] F. Behbahani, K. Shiarlis, X. Chen, V. Kurin, S. Kasewa, C. Stirbu, J. Gomes, S. Paul, F. A. Oliehoek, J. Messias, et al. Learning from demonstration in the wild. *arXiv preprint arXiv:1811.03516*, 2018.

[16] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[17] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[18] X. Liang, T. Wang, L. Yang, and E. Xing. Cirl: Controllable imitative reinforcement learning for vision-based self-driving. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 584–599, 2018.

[19] N. Rhinehart, R. McAllister, and S. Levine. Deep imitative models for flexible inference, planning, and control. *arXiv preprint arXiv:1810.06544*, 2018.

[20] G. Li, M. Mueller, V. Casser, N. Smith, D. L. Michels, and B. Ghanem. Oil: Observational imitation learning. *arXiv preprint arXiv:1803.01129*, 2018.

[21] A. Jain, H. S. Koppula, B. Raghavan, S. Soh, and A. Saxena. Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3182–3190, 2015.

[22] D. A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

[23] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58, 2016.

[24] C. Finn, P. Christiano, P. Abbeel, and S. Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

[25] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[26] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[27] N. Rhinehart, K. M. Kitani, and P. Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018.