

# Learning robust driving policies without online exploration

Daniel Graves<sup>1</sup>, Nhat M. Nguyen<sup>1</sup>, Kimia Hassanzadeh<sup>1</sup>, Jun Jin<sup>1</sup>, Jun Luo<sup>1</sup>

**Abstract**—We propose a multi-time-scale predictive representation learning method to efficiently learn robust driving policies in an offline manner that generalize well to novel road geometries, and damaged and distracting lane conditions which are not covered in the offline training data. We show that our proposed representation learning method can be applied easily in an offline (batch) reinforcement learning setting demonstrating the ability to generalize well and efficiently under novel conditions compared to standard batch RL methods. Our proposed method utilizes training data collected entirely offline in the real-world which removes the need of intensive online explorations that impede applying deep reinforcement learning on real-world robot training. Various experiments were conducted in both simulator and real-world scenarios for the purpose of evaluation and analysis of our proposed claims.

## I. INTRODUCTION

Learning to drive is a challenging problem that is a long-standing goal in robotics and autonomous driving. In the early days of autonomous driving, a popular approach to staying within a lane was based on lane marking detection [1]. However, a significant challenge with this approach is the lack of robustness to missing, occluded or damaged lane markings [2] where most roads in the US are not marked with reliable lane markings on either side of the road [3]. Modern approaches mitigate some of these issues by constructing high definition maps and developing accurate localization techniques [4], [5], [6], [7]. However, scaling both the map and localization approaches globally in a constantly changing world is still very challenging for autonomous driving and robotic navigation [4].

Recently, there have been a growing number of successes in AI applied to robotics and autonomous driving [8], [9], [10], [11], [12], [3]. These data-driven approaches can be divided into two categories: (1) behavior cloning, and (2) reinforcement learning (RL). Behavior cloning suffers from generalization challenges since valuable negative experiences are rarely collected; in addition they cannot offer performance better than the behavior being cloned [8], [13], [12]. RL on the other-hand is a promising direction for vision-based control [14]; however, RL is usually not practical because it requires extensive online exploration in the environment to find the best policy that maximizes the cumulative reward [11], [15], [16]. Moreover, the success in game environments like Go [17] doesn't always transfer well to success in the real-world where an agent is expected to learn policies that generalize well [18], [16]. A key challenge is that RL overfits to the training environment where learned policies tend to perform

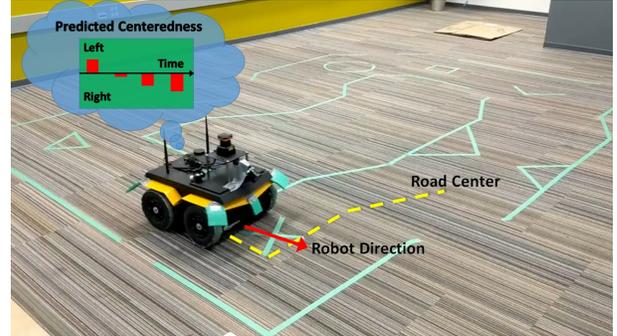


Fig. 1: Lane keeping of a Jackal Robot using vision-based counterfactual predictions of the future lane centeredness over multiple time scales to represent the state of the agent in RL

poorly on novel situations not seen during training [19], [20], [21], [22]. We aim to address the issues of learning both practical and general driving policies in the real-world by combining a novel representation learning approach with offline RL without any online exploration.

Offline RL, or learning RL policies from data without exploration [23], could potentially address many of the practicality issues with applying standard online RL in the real-world. Unfortunately, deep offline RL struggles to generalize to data not in the training set [24], [25]. Our approach applies novel representation learning based on counterfactual predictions [26], [27], [25], shown in Figure 1, to address the generalization issue. We learn predictions of future lane centeredness and road angle from offline data safely collected in the environment using noisy localization sources during training, eliminating the need for expensive, on-vehicle, high accuracy localization sensors during deployment. State-of-the-art offline RL [23] is then applied using these counterfactual predictions as a low dimensional representation of the state to learn a policy to drive the vehicle. These counterfactual predictions are motivated in psychology [28], [29] where we find predictions aid in an agent's understanding of the world, particularly in driving [30]. Similar works in classical control have shown how anticipation of the future is important for driving at the limits of stability through feed-forward [31] and model predictive control [32]. Our work is motivated by the predictive state hypothesis [33], [34] that claims counterfactual predictions help an agent generalize and adapt quickly to new problems [35].

The significance of our approach is that it demonstrates practical value in autonomous driving and real-world RL without requiring extensive maps, robust localization techniques or robust lane marking and curb detection. We demonstrate that

<sup>1</sup>Noah's Ark Lab, Huawei Technologies Canada {daniel.graves, minh.nhat.nguyen, kima.hassanzadeh, jun.jin1, jun.luo1}@huawei.com

our approach generalizes to never-before seen roads including those with damaged and distracting lane markings. The novel contributions of this work are summarized as follows: (1) an algorithm for learning counterfactual predictions from real-world driving data with behavior distribution estimation, (2) an investigation into the importance of predictive representations for learning good driving policies that generalize well to new roads and damaged lane markings, and (3) the first demonstration of deep RL applied to autonomous driving with real-world data without any online exploration.

## II. RELATED WORKS

**Deep learning approaches to driving:** There have been many attempts to apply deep learning to driving including deep RL and imitation learning [11]; however generalization is a key challenge. ChaufferNet [36] used a combination of imitation learning and predictive models to synthesize the worst case scenarios but more work is needed to improve the policy to achieve performance competitive with modern motion planners. Another approach trained the agent entirely in the simulator where transfer to the real-world could be challenging to achieve [11]. DeepDriving [37] learned affordance predictions of road angle from an image for multi-lane driving in simulation using offline data collected by human drivers. However, in contrast with our proposed method, DeepDriving used heuristics and rules to control the vehicle instead of learning a policy with RL. Moreover, DeepDriving learned predictions of the current lane centeredness and current road angle rather than long-term counterfactual predictions of the future.

**Offline RL in real-world robot training:** There are many prior arts in offline (batch) RL [38], [24]. However, most prior arts in offline RL have challenges learning good policies in the deep setting [24]. The current state of the art in offline RL is batch constrained Q-learning (BCQ) [23], [24] where success is demonstrated in simulation environments such as Atari but the results still perform badly in comparison to online learning. The greatest challenge with offline RL is the difficulty in covering the state-action space of the environment resulting in holes in the training data where extrapolation is necessary. [39] applied a novel offline RL approach to playing soccer with a real-world robot by exploiting the episodic nature of the problem. Our work overcomes these challenges and is, to the best of our knowledge, the first successful real-world robotic application of batch RL with deep learning.

**Counterfactual prediction learning:** Learning counterfactual predictions as representation of the state of the agent has been proposed before in the real-world [40], [41]. Other approaches demonstrate counterfactual predictions but don't provide a way to use them [26], [42], [27]. While experiments with counterfactual predictions show a lot of promise for improving learning and generalization, most experiments are in simple tabular domains [33], [34], [35]. Auxiliary tasks and similar prediction problems have been applied to deep RL task in simulation but assume the policy is the same

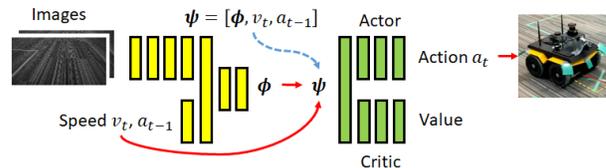


Fig. 2: Overall architecture of the RL system involved learns a predictive representation  $\psi$  to represent the state of the agent. Camera is the only environment sensor at test time.

as the policy being learned and thus are not counterfactual predictions [43], [44], [29], [45].

## III. PREDICTIVE CONTROL FOR AUTONOMOUS DRIVING

Let us consider the usual setting of an MDP described by a set of states  $S$ , a set of actions  $A$ , and transition dynamics with probability  $P(s'|s, a)$  of transitioning to next state  $s'$  after taking action  $a$  from state  $s$ , and a reward  $r$ . The objective of an MDP is to learn a policy  $\pi$  that maximizes the future discounted sum of rewards in a given state. Obtaining the state of the agent in an MDP environment is not trivial especially with deep RL where the policy is changing because the target is moving [46]. Our approach is to learn an intermediate representation mapping sensor readings  $s$  to a limited number of counterfactual predictions  $\phi$  as a representation of the state for deep RL. This has the advantage of pushing the heavy burden of deep feature representation learning in RL to the easier problem of prediction learning [47], [48], [49], [43].

The overall architecture of the system is depicted in Figure 2. The proposal is to represent the state of the agent as a vector  $\psi$  which is the concatenation of a limited number of the predictions  $\phi$ , the current speed of the vehicle  $v_t$  and the previous action taken  $a_{t-1}$ . The predictions  $\phi$  are counterfactual predictions, also called general value functions [26]. The previous action taken is needed due to the nature of the predictions which are relative to the last action.

Learning a policy  $\pi(\psi)$  could provide substantial benefits over learning  $\pi$  from image observations: (1) improving learning performance and speed, (2) enabling batch RL from offline data, and (3) improving generalization of the driving policy. Our approach is to learn a value function  $Q(s, a)$  and a deterministic policy  $\pi(\psi)$  that maximizes that value function using batch constrained Q-learning (BCQ) [23]. While the networks can be modelled as one computational graph, the gradients from the policy and value function network are not back-propagated through the prediction network to decouple the representation learning when learning from the offline data. Thus, training happens in two phases: (1) learning the prediction network, (2) learning the policy and value function.

During the first phase of training, a low-accuracy localization algorithm, based on 2D lidar scan matching, produces the lane centeredness  $\alpha$  and relative road angle  $\beta$  of the vehicle, depicted in Figure 3, that are used to train the prediction network. The prediction network is a single network that predicts the lane centeredness and relative road angle over multiple temporal horizons depicted in Figure 4: these

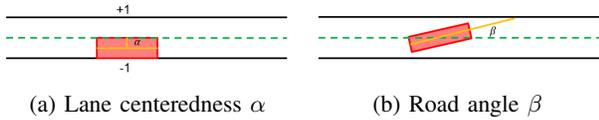


Fig. 3: An illustration of (a) lane centeredness position  $\alpha$ , and (b) the road angle  $\beta$  which is the angle between the direction of the vehicle and the direction of the road.

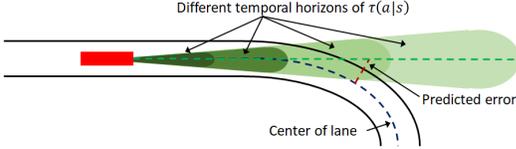


Fig. 4: An illustration of the multiple temporal horizons of the predictions  $\phi$ .

are predictions of the future lane centeredness and relative road angle rather than the current estimates returned by the localization algorithm. They are chosen because they represent both present and future lane centeredness information needed to steer [30]. These predictions are discounted sums of future lane centeredness and relative road angle respectively that are learned with GVF [26]:

$$\phi(s) = \mathbb{E}_\tau \left[ \sum_{i=0}^{\infty} \gamma^i c_{t+i+1} | s_t = s \right] \quad (1)$$

where  $c_{t+i+1}$  is the cumulant vector consisting of the current lane centeredness  $\alpha$  and current relative road angle  $\beta$ . It is important to understand that  $\phi(s)$  predicts the sum of all future lane centeredness and road angle values collected under some policy  $\tau$ . The policy  $\tau$  is counterfactual in the sense that it is different from the behavior policy  $\mu$  used to collect the data and the learned policy  $\pi$ . Formally, the policy  $\tau(a_t | s_t, a_{t-1}) = \mathcal{N}(a_{t-1}, \Sigma)$  where  $\Sigma = 0.0025I$  is a diagonal covariance matrix. The meaning of this policy is to “keep doing what you are doing”, similar to the one used in [32] for making counterfactual predictions. Therefore,  $\phi(s)$  predicts the discounted sum of future lane centeredness and road angle if the vehicle takes similar actions to its last action. Moreover,  $\phi(s)$  can be interpreted as predictions of the deviation from the desired lane centeredness and road angle. Counterfactual predictions can be thought of as anticipated future “errors” that allow controllers to take corrective actions before the errors occur. The discount factor  $\gamma$  controls the temporal horizon of the prediction. It is critical to learn  $\phi(s)$  for different values of  $\gamma$  in order to control both steering and speed. The details for learning  $\phi(s)$  are provided in the next section.

During the second stage of training, the localization algorithm is no longer needed; it was used to provide the labels for training the predictive representation in the first stage. Instead, the counterfactual predictions  $\phi$  are concatenated with the vehicle speed  $v_t$  and last action  $a_{t-1}$  to form a predictive representation  $\psi$ . The RL agent receives  $\psi$  as the state of the agent in the environment which is used to

predict the value and produce the next action  $a_t$  as depicted in Figure 2. In our offline learning approach, we used the state of the art batch RL BCQ [23][24] to train the policy.

Note that the same architecture can also be applied online where the counterfactual prediction, policy and value networks are all learned online simultaneously with deep deterministic policy gradient (DDPG) [50] but the details are left in the appendix.

#### IV. PREDICTIVE LEARNING

The counterfactual predictions given in Equation (1) are general value functions (GVFs) [26] that are learned with a novel combination of different approaches including (1) off-policy, or counterfactual, prediction learning with importance resampling [47], and (2) behavior estimation with the density ratio trick [51].

##### A. Counterfactual Predictions

To ask a counterfactual predictive question, we use the GVF framework, where one must define a cumulant  $c_t = c(s_t, a_t, s_{t+1})$ , a.k.a. pseudo-reward, a policy distribution  $\tau(a|s)$  and continuation function  $\gamma_t = \gamma(s_t, a_t, s_{t+1})$ . The answer to the predictive question is the expectation of the return  $\phi_t$ , when following policy  $\tau$ , defined by

$$\phi^\tau(s) = \mathbb{E}_\tau \left[ \sum_{k=0}^{\infty} \left( \prod_{j=0}^{k-1} \gamma_{t+j+1} \right) c_{t+k+1} | s_t = s, a_t = a \right] \quad (2)$$

where the cumulant is  $c_t$  and  $0 \leq \gamma_t \leq 1$  [26]. This is the more general form for learning a prediction than the one given in Equation (1) where the only difference is that  $\gamma$  is replaced by a continuation function which allows for predictions that predict the sum of cumulants until an episodic event occurs such as going out of lane. The agent usually collects experience under a different behavior policy  $\mu(a|s)$ . When  $\tau$  is different from both the behavior policy  $\mu$  and the policy being learned  $\pi$ , the predictive question is a counterfactual prediction<sup>1</sup>. Cumulants are often scaled by a factor of  $1 - \gamma$  when  $\gamma$  is a constant in non-episodic predictions. The counterfactual prediction  $\phi^\tau(s)$  is a general value function (GVF) is approximated by a deep neural network parameterized by  $\theta$  to learn (2). The parameters  $\theta$  are optimized with gradient descent minimizing the following loss function

$$L(\theta) = \mathbb{E}_\mu [\rho \delta^2] \quad (3)$$

where  $\delta = \phi^\tau(s; \theta) - y$  is the TD error and  $\rho = \frac{\tau(a|s)}{\mu(a|s)}$  is the importance sampling ratio to correct for the difference between the policy distribution  $\tau$  and behavior distribution  $\mu$ . Note that only the behavior policy distribution is corrected; but the expectation is still over the state visitation distribution under the policy  $\mu$ . In practice, this is usually not an issue

<sup>1</sup>Some literature call this an off-policy prediction

[47]. The target  $y$  is produced by bootstrapping a prediction of the value of the next state [52] under policy  $\tau$

$$y = \mathbb{E}_{s_{t+1} \sim P}[c_{t+1} + \gamma \phi^\tau(s_{t+1}; \hat{\theta}) | s_t = s, a_t = a] \quad (4)$$

where  $y$  is a bootstrapped prediction using the most recent parameters  $\hat{\theta}$  that are assumed constant in the gradient computation. Learning a counterfactual prediction with a fixed policy  $\tau$  tends to be very stable when minimizing  $L(\theta)$  using gradient descent approaches and therefore doesn't require target networks originally used in [46] to stabilize DQN.

The gradient of the loss function (3) is given by

$$\nabla_\theta L(\theta) = \mathbb{E}_\mu[\rho \delta \nabla_\theta \phi^\tau(s; \theta)] \quad (5)$$

However, updates with importance sampling ratios are known to have high variance which may negatively impact learning; instead we use the importance resampling technique to reduce the variance of the updates [47]. With importance resampling, a replay buffer  $D$  of size  $N$  is required and the gradient is estimated from a mini-batch and multiplied with the average importance sampling ratio of the samples in the buffer  $\bar{\rho} = \frac{\sum_{i=1}^N \rho_i}{N}$ .

The gradient with importance resampling is given by

$$\nabla_\theta L(\theta) = \mathbb{E}_{s, a \sim D_\rho}[\bar{\rho} \delta \nabla_\theta \hat{v}^\tau(s; \theta)] \quad (6)$$

where  $D_\rho$  is a distribution of the transitions in the replay buffer proportional to the importance sampling ratio. The probability for transition  $i = 1 \dots N$  is given by  $D_i = \frac{\rho_i}{\sum_{j=1}^N \rho_j}$  where the importance sampling ratio is  $\rho_i = \frac{\tau(a_i | s_i)}{\mu(a_i | s_i)}$ . An efficient data structure for the replay buffer is the SumTree used in prioritized experience replay [53].

### B. Behavior Estimation

When learning predictions from real-world driving data, one needs to know the behavior policy distribution  $\mu(a|s)$ ; however, in practice this is rarely known. Instead we estimate it using the density ratio trick [51] where the ratio of two probability densities can be expressed as a ratio of discriminator class probabilities that distinguish samples from the two distributions. Let us define an intermediate probability density function  $\eta(a|s)$  such as the uniform distribution; this will be compared to the behavior distribution  $\mu(a|s)$  which we desire to estimate. The class labels  $y = +1$  and  $y = -1$  are labels given to samples from  $\mu(a|s)$  and  $\eta(a|s)$ . A discriminator  $g(a, s)$  is learned that distinguishes state action pairs from the two distributions using the cross-entropy loss. The ratio of the densities can be computed using only the discriminator  $g(a, s)$ .

$$\begin{aligned} \frac{\mu(a|s)}{\eta(a|s)} &= \frac{p(a|s, y = +1)}{p(a|s, y = -1)} = \frac{p(y = +1|a, s)/p(y = +1)}{p(y = -1|a, s)/p(y = -1)} \\ &= \frac{p(y = +1|a, s)}{p(y = -1|a, s)} = \frac{g(a, s)}{1 - g(a, s)} \end{aligned} \quad (7)$$

Here we assume that  $p(y = +1) = p(y = -1)$ . From this result, we can estimate  $\mu(a|s)$  with  $\hat{\mu}(a|s)$  as follows

$$\hat{\mu}(a|s) = \frac{g(a, s)}{1 - g(a, s)} \eta(a|s) \quad (8)$$

where  $\eta(a|s)$  is a known distribution over action conditioned on state. Choosing  $\eta(a|s)$  to be the uniform distribution ensures that the discriminator is well trained against all possible actions in a given state; thus good performance is achieved with sufficient coverage of the state space rather than the state-action space. Alternatively, one can estimate the importance sampling ratio without defining an additional distribution  $\eta$  by replacing the distribution  $\eta$  with  $\tau$ ; however, defining  $\eta$  to be a uniform distribution ensures the discriminator is learned effectively across the entire action space. The combined algorithms for training counterfactual predictions with an unknown behavior distribution are given in the Appendix for both the online and offline RL settings.

## V. EXPERIMENTS

Our approach to learning counterfactual predictions for representing the state used in RL to learn a driving policy is applied to two different domains. The first set of experiments is conducted on a Jackal robot in the real-world where we demonstrate the practicality of our approach and its robustness to damaged and distracting lane markings. The second set of experiments is conducted in the TORCS simulator where we conduct an ablation study to understand the effect different counterfactual predictive representations have on performance and comfort. Refer to the Appendix<sup>2</sup> for more details in the experimental setup and training.

### A. Jackal Robot

The proposed solution for learning to drive the Jackal robot in the real-world is called GVF-BCQ since it combines our novel method of learning GVF predictions with BCQ [23]. Two baselines are compared with our method: (1) a classical controller using model predictive control (MPC), and (2) batch-constrained Q-learning that trains end-to-end (E2E-BCQ). The MPC uses a map and 2D laser scanner for localization from pre-existing ROS packages. The E2E-BCQ is the current state-of-the-art in offline deep RL [24]. Comparing to online RL was impractical for safety concerns and the need to recharge the robot's battery every 4 hours.

The training data consisted of 6 training roads in both counter clock-wise (CCW) and clock-wise (CW) directions and 3 test roads where each of the 3 test roads had damaged variants. All training data was flipped to simulate travelling in the reverse direction and balance the data set in terms of direction. The training data was collected using a diverse set of drivers including human drivers by remote control and a pure pursuit controller with safe exploration; thus, the training data was not suitable for imitation learning. The test roads were different from the training data: (1) a rectangle-shaped road with rounded outer corners, (2) an oval-shaped road,

<sup>2</sup>Appendix is at <https://bit.ly/3mIDScp>

TABLE I: Comparison of GVF-BCQ (our method) and E2E-BCQ (baseline) on Rectangle test road with 0.4 m/s target speed in both the CW and CCW directions. GVF-BCQ exceeds performance of E2E-BCQ in all respects with higher overall speed, and far fewer out of lane events. E2E-BCQ was deemed unsafe for further experiments.

Method	Dir.	r/s ↑	Speed ↑	Off-center ↓	Off-angle ↓	Out of Lane ↓
GVF-BCQ	CCW	<b>2.68</b>	<b>0.32</b>	<b>0.14</b>	<b>0.13</b>	<b>0.0%</b>
E2E-BCQ	CCW	1.26	0.18	0.26	0.24	3.8%
GVF-BCQ	CW	<b>2.29</b>	<b>0.31</b>	<b>0.22</b>	<b>0.16</b>	<b>0.0%</b>
E2E-BCQ <sup>3</sup>	CW	-0.13	0.17	0.99	0.30	54.2%

TABLE II: Effect of damaged lanes on GVF-BCQ performance in CCW direction with 0.4 m/s target speed where R, O, and C are the Rectangle, Oval and Complex road shapes respectively. GVF-BCQ demonstrates robustness to damaged and distracting lanes.

	Damage	r/s ↑	Off-center ↓	Off-angle ↓	Out of Lane ↓	Speed Jerk ↓	Steer Jerk ↓
R	No	2.68	0.13	0.13	0.0%	0.036	0.23
	Yes	<b>2.74</b>	0.14	0.14	0.0%	-0.038	0.23
O	No	<b>2.40</b>	<b>0.28</b>	0.21	<b>1.5%</b>	0.035	0.22
	Yes	2.07	0.33	0.21	7.19%	0.033	0.21
C	No	<b>2.35</b>	<b>0.22</b>	<b>0.18</b>	<b>0.0%</b>	<b>0.034</b>	<b>0.23</b>
	Yes	2.11	0.31	0.24	9.42%	0.044	0.29

and (3) a complex road loop with many turns significantly different from anything observed by the agent during training. In addition, the test roads included variants with damaged lane markings. The reward is given by  $r_t = v_t(\cos \beta_t + |\alpha_t|)$  where  $v_t$  is the speed of the vehicle in km/h,  $\beta_t$  is the angle between the road direction and the vehicle direction, and  $\alpha_t$  is the lane centeredness.

A comparison of the learned approaches is given in Table I where GVF-BCQ approach exceeds the performance of E2E-BCQ in all respects demonstrating better performance at nearly double the speed. Both GVF-BCQ and E2E-BCQ were trained with the same data sets and given 10M updates each for a fair comparison. For GVF-BCQ, the first 5M updates were used for learning the counterfactual predictions and the second 5M updates were used for learning the policy from the predictive representation with BCQ. They both received the same observations consisting of two stacked images, current vehicle speed, and last action and produced desired steering angle and speed.

GVF-BCQ was tested on roads with damaged and distracting lane markings as shown in Table III. The damaged and distracting lane markings for the complex test road loop are shown in Figure 1. These results demonstrate robustness because the training data did not include roads with damaged or distracting lane markings.

GVF-BCQ was also compared to MPC in Table III where GVF-BCQ was found to produce superior performance in

<sup>3</sup>E2E-BCQ failed to recover after undershooting the first turn in the clock-wise (CW) direction; it was not safe for testing on the other roads.

TABLE III: Comparison of GVF-BCQ (our method) and MPC (baseline) in CCW direction with 0.4 m/s target speed where R, O, and C are the Rectangle, Oval and Complex road shapes respectively.

	Method	r/s ↑	Off-center ↓	Off-angle ↓	Out of Lane ↓	Speed Jerk ↓	Steer Jerk ↓
R	GVF-BCQ	<b>2.68</b>	<b>0.13</b>	<b>0.13</b>	<b>0.0%</b>	<b>0.036</b>	<b>0.23</b>
	MPC	0.97	0.53	0.19	20.4%	0.083	1.25
O	GVF-BCQ	<b>2.40</b>	<b>0.28</b>	0.21	<b>1.45%</b>	<b>0.035</b>	<b>0.22</b>
	MPC	0.89/s	0.53	<b>0.20</b>	22.7%	0.103	1.41
C	GVF-BCQ	<b>2.35</b>	<b>0.22</b>	<b>0.18</b>	<b>0.0%</b>	<b>0.034</b>	<b>0.23</b>
	MPC	0.72	0.64	0.21	38.9%	-0.063	-1.21

nearly all metrics at a high target speed of 0.4 m/s. The MPC performed poorly since it was difficult to tune for 0.4 m/s; performance was more similar at 0.25 m/s speeds where results are in the Appendix. A clear advantage of GVF-BCQ is the stability and smoothness of control achieved at the higher speeds.

### B. Ablation Study in TORCS

In order to understand the role of counterfactual predictions in representing the state of the agent, we conduct an ablation study in the TORCS simulator. We compare representations consisting of future predictions at multiple time scale, future predictions at a single time scale and predictions with supervised regression of the current (non-future) lane centeredness and relative road angle. These experiments were conducted with online RL using deep deterministic policy gradient (DDPG) [50] in order to more easily understand the impact of the different state representations on the learning process.

Our method is called GVF-DDPG and uses multiple time scales specified by the values  $\gamma = [0.0, 0.5, 0.9, 0.95, 0.97]$ . Two variants of our method called GVF-0.95-DDPG and GVF-0.0-DDPG were defined to investigate the impact of different temporal horizons on performance, where  $\gamma = 0.95$  and  $\gamma = 0.0$  respectively. It is worth pointing out that when  $\gamma = 0$ , the prediction is myopic meaning that it reduces to a standard supervised regression problem equivalent to the predictions learned in [37]. These methods receive a history of two images, velocity and last action and produce desired steering angle and vehicle speed action commands.

Some additional baselines include a kinematic-based steering approach based on [54] and two variants of DDPG with slightly different state representations. The kinematic-based steering approach is treated as a "ground truth" controller since it has access to perfect localization information to steer the vehicle; unlike our approach, the speed is controlled independently. The variants of DDPG are called (1) DDPG-Image and (2) DDPG-LowDim. DDPG-Image is given a history of two images, velocity and last action while DDPG-LowDim is given a history of two images, velocity, last action, current lane centeredness  $\alpha$  and relative road angle  $\beta$  in the observation. Both DDPG-Image and DDPG-LowDim output steering angle and vehicle speed action commands. The performance of DDPG-LowDim serves as an ideal learned

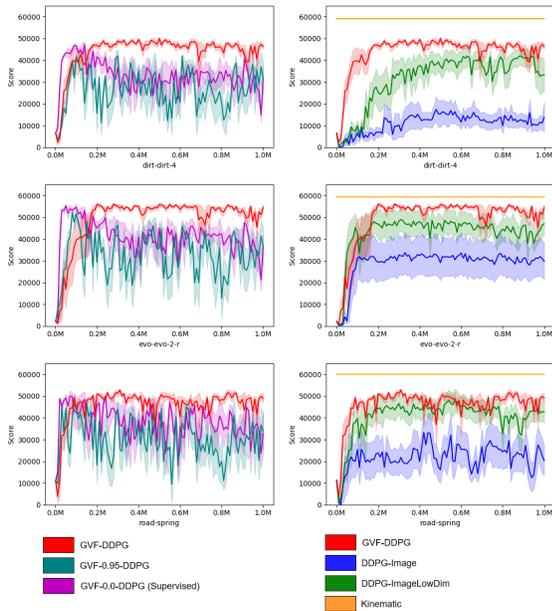


Fig. 5: Ablation study of GVF-DDPG (our method) of test scores (accumulated reward) over different time scale selections (left) and raw image-based state representations (right). Test scores were evaluated every 1000 steps during training for dirt-dirt-4, evo-evo-2 and road-spring which were not part of the training set. Results show our proposed predictive representation with multiple time scales achieves the best performance.

controller since it learns from both images and the perfect localization information.

The learned agents were trained on 85% of 40 tracks available in TORCS. The rest of the tracks were used for testing (6 in total) to measure the generalization performance of the policies. Results are repeated over 5 runs for each method. Only three of the tracks were successfully completed by at least one learned agent and those are reported here. The reward in the TORCS environment is given by  $r_t = 0.0002v_t(\cos \beta_t + |\alpha_t|)$  where  $v_t$  is the speed of the vehicle in km/h,  $\beta_t$  is the angle between the road direction and the vehicle direction, and  $\alpha_t$  is the current lane centeredness. The policies were evaluated on test roads at regular intervals during training as shown in Figures 5 and 6.

The GVF-0.0-DDPG and GVF-0.95-DDPG variations initially learned very good solutions but then diverged indicating that one prediction may not be enough to control both steering angle and vehicle speed. Despite an unfair advantage provided by DDPG-LowDim with the inclusion of lane centeredness and road angle in the observation vector, GVF-DDPG still outperforms both variants of DDPG on many of the test roads. DDPG-Image was challenging to tune and train due to instability in learning; however, the counterfactual predictions in GVF-DDPG stabilized training for more consistent learning even though they were being learned simultaneously. Only GVF-DDPG with multiple time scale predictions is able to achieve extraordinarily smooth control.

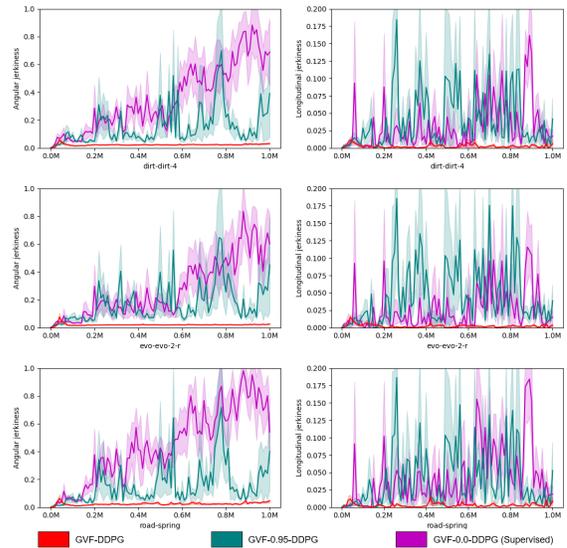


Fig. 6: Ablation study of GVF-DDPG (our method) of jerkiness (lower is better) over different time scale selections. We use angular and longitudinal jerkiness to evaluate the smoothness of the learned policy. The jerkiness is evaluated every 1000 steps during training for dirt-dirt-4, evo-evo-2 and road-spring which were not part of the training set. Results show our proposed multi-time-scale predictions achieves the best performance.

## VI. CONCLUSIONS

We present a new approach to learning to drive through a two step process: (1) learn a limited number of counterfactual predictions about future lane centeredness and road angle under a known policy, and (2) learn an RL policy using the counterfactual predictions as a representation of state. Our novel approach is safe and practical because it learns from real-world driving data without online exploration where the behavior distribution of the driving data is unknown. An experimental investigation into the impact of predictive representations on learning good driving policies shows that they generalize well to new roads, damaged lane markings and even distracting lane markings. We find that our approach improves the performance, smoothness and robustness of the driving decisions from images. We conclude that counterfactual predictions at different time scales is crucial to achieve a good driving policy. To the best of our knowledge, this is the first practical demonstration of deep RL applied to autonomous driving on a real vehicle using only real-world data without any online exploration.

Our approach has the potential to be scaled with large volumes of data captured by human drivers of all skill levels; however, more work is needed to understand how well this approach will scale. In addition, a general framework of learning the right counterfactual predictions for real-world problems is needed where online interaction is prohibitively expensive.

## REFERENCES

- [1] N. Möhler, D. John, and M. Voigtländer, "Lane detection for a situation adaptive lane keeping support system, the safelane system," in *Advanced Microsystems for Automotive Applications 2006*, J. Valldorf and W. Gessner, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 485–500.
- [2] Q. Zou, H. Jiang, Q. Dai, Y. Yue, L. Chen, and Q. Wang, "Robust lane detection from continuous driving scenes using deep neural networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 41–54, 2020.
- [3] T. Ort, L. Paull, and D. Rus, "Autonomous vehicle navigation in rural environments without detailed prior maps," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2040–2047.
- [4] Bing-Fei Wu, Tsu-Tian Lee, Hsin-Han Chang, Jhong-Jie Jiang, Cheng-Nan Lien, Tien-Yu Liao, and Jau-Woei Perng, "Gps navigation based autonomous driving system design for intelligent vehicles," in *IEEE International Conference on Systems, Man and Cybernetics*, 2007, pp. 3294–3299.
- [5] G. Garimella, J. Funke, C. Wang, and M. Kobilarov, "Neural network modeling for steering control of an autonomous vehicle," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2609–2615.
- [6] R. Liu, J. Wang, and B. Zhang, "High definition map for automated driving: Overview and analysis," *Journal of Navigation*, vol. 73, no. 2, p. 324–341, 2020.
- [7] L. Wang, Y. Zhang, and J. Wang, "Map-based localization method for autonomous vehicles using 3d-lidar," *IFAC*, vol. 50, no. 1, pp. 276–281, 2017.
- [8] J. Chen, B. Yuan, and M. Tomizuka, "Deep imitation learning for autonomous driving in generic urban scenarios with enhanced safety," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2884–2890, 2019.
- [9] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016. [Online]. Available: <http://arxiv.org/abs/1604.07316>
- [10] Z. Chen and X. Huang, "End-to-end learning for lane keeping of self-driving cars," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1856–1860.
- [11] A. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, pp. 70–76, 2017.
- [12] L. Chi and Y. Mu, "Deep steering: Learning end-to-end driving model from spatial and temporal visual cues," *CoRR*, vol. abs/1708.03798, 2018. [Online]. Available: <http://arxiv.org/abs/1708.03798>
- [13] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. A. Theodorou, and B. Boots, "Imitation learning for agile autonomous driving," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 286–302, 2020.
- [14] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3389–3396.
- [15] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *CoRR*, vol. abs/2002.00444, 2020. [Online]. Available: <http://arxiv.org/abs/2002.00444>
- [16] G. Dulac-Arnold, D. J. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *International Conference on Machine Learning*, vol. abs/1904.12901, 2019. [Online]. Available: <http://arxiv.org/abs/1904.12901>
- [17] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016.
- [18] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas *et al.*, "Solving rubik's cube with a robot hands," *CoRR*, vol. abs/1910.07113, 2019. [Online]. Available: <http://arxiv.org/abs/1910.07113>
- [19] S. Whiteson, B. Tanner, M. E. Taylor, and P. Stone, "Protecting against evaluation overfitting in empirical reinforcement learning," *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 120–127, 2011.
- [20] C. Zhao, O. Sigaud, F. Stulp, and T. M. Hospedales, "Investigating generalisation in continuous deep reinforcement learning," *CoRR*, vol. abs/1902.07015, 2019. [Online]. Available: <https://arxiv.org/abs/1902.07015>
- [21] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," *CoRR*, vol. abs/1709.06560, 2017. [Online]. Available: <http://arxiv.org/abs/1709.06560>
- [22] J. Farebrother, M. C. Machado, and M. Bowling, "Generalization and regularization in DQN," *CoRR*, vol. abs/1810.00123, 2018. [Online]. Available: <http://arxiv.org/abs/1810.00123>
- [23] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," *CoRR*, vol. abs/1812.02900, 2018. [Online]. Available: <http://arxiv.org/abs/1812.02900>
- [24] S. Fujimoto, E. Conti, M. Ghavamzadeh, and J. Pineau, "Benchmarking batch deep reinforcement learning algorithms," *CoRR*, vol. abs/1910.01708, 2019. [Online]. Available: <http://arxiv.org/abs/1910.01708>
- [25] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: tutorial, review and perspectives on open problems," *CoRR*, vol. abs/2005.01643, 2020. [Online]. Available: <http://arxiv.org/abs/2005.01643>
- [26] R. Sutton, J. Modayil, M. Delp, T. Degris, P. Pilarski, A. White, and D. Precup, "Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction," in *International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '11, vol. 2, 2011, pp. 761–768.
- [27] J. Modayil, A. White, and R. S. Sutton, "Multi-timescale nexting in a reinforcement learning robot," in *From Animals to Animals 12*, T. Ziemke, C. Balkenius, and J. Hallam, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 299–309.
- [28] A. Clark, "Whatever next? predictive brains, situated agents, and the future of cognitive science," *Behavioral and Brain Science*, vol. 36, no. 3, pp. 181–204, 2013.
- [29] E. M. Russek, I. Momennejad, M. M. Botvinick, S. J. Gershman, and N. D. Daw, "Predictive representations can link model-based reinforcement learning to model-free mechanisms," *PLOS Computational Biology*, vol. 13, no. 9, pp. 1–35, 2017.
- [30] D. D. Salvucci and R. Gray, "A two-point visual control model of steering," *Perception*, vol. 33, no. 10, pp. 1233–1248, 2004.
- [31] N. Kapania and J. Gerdes, "Design of a feedback-feedforward steering controller for accurate path tracking and stability at the limits of handling," *Vehicle System Dynamics*, vol. 53, pp. 1–18, 2015.
- [32] C. Beal and J. Gerdes, "Model predictive control for vehicle stabilization at the limits of handling," *Control Systems Technology, IEEE Transactions on*, vol. 21, pp. 1258–1269, 2013.
- [33] M. L. Littman and R. S. Sutton, "Predictive representations of state," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., 2002, pp. 1555–1561.
- [34] E. J. Rafols, M. B. Ring, R. S. Sutton, and B. Tanner, "Using predictive representations to improve generalization in reinforcement learning," in *International Joint Conference on Artificial Intelligence*, ser. IJCAI'05, 2005, pp. 835–840.
- [35] T. Schaul and M. Ring, "Better generalization with forecasts," in *International Joint Conference on Artificial Intelligence*, ser. IJCAI '13, 2013, pp. 1656–1662.
- [36] M. Bansal, A. Krizhevsky, and A. S. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," in *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22–26, 2019*, A. Bicchi, H. Kress-Gazit, and S. Hutchinson, Eds., 2019. [Online]. Available: <https://doi.org/10.15607/RSS.2019.XV.031>
- [37] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2722–2730.
- [38] P. S. Thomas and E. Brunskill, "Data-efficient off-policy policy evaluation for reinforcement learning," in *International Conference on Machine Learning*, ser. ICML'16, vol. 48, 2016, p. 2139–2148.
- [39] J. Cunha, R. Serra, N. Lau, L. Lopes, and A. Neves, "Batch reinforcement learning for robotic soccer using the q-batch update-rule," *Journal of Intelligent & Robotic Systems*, vol. 80, pp. 385–399, 2015.

- [40] J. Günther, P. M. Pilarski, G. Helfrich, H. Shen, and K. Diepold, "Intelligent laser welding through representation, prediction, and control learning: An architecture with deep neural networks and reinforcement learning," *Mechatronics*, vol. 34, pp. 1 – 11, 2016.
- [41] A. L. Edwards, M. R. Dawson, J. S. Hebert, C. Sherstan, R. S. Sutton, K. M. Chan, and P. M. Pilarski, "Application of real-time machine learning to myoelectric prosthesis control: A case series in adaptive switching," *Prosthetics and orthotics international*, vol. 40, no. 5, pp. 573–581, 2016.
- [42] A. White, "Developing a predictive approach to knowledge," Ph.D. dissertation, University of Alberta, 2015.
- [43] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks." *International Conference on Learning Representations*, 2017.
- [44] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. P. van Hasselt, and D. Silver, "Successor features for transfer in reinforcement learning," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., 2017, pp. 4055–4065.
- [45] H. Van Seijen, M. Fatemi, J. Romoff, R. Laroché, T. Barnes, and J. Tsang, "Hybrid reward architecture for reinforcement learning," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., 2017, pp. 5392–5402.
- [46] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: <http://arxiv.org/abs/1312.5602>
- [47] M. Schlegel, W. Chung, D. Graves, J. Qian, and M. White, "Importance resampling off-policy prediction," in *Neural Information Processing Systems*, ser. NeurIPS'19, 2019.
- [48] S. Ghiassian, A. Patterson, M. White, R. S. Sutton, and A. White, "Online off-policy prediction," *CoRR*, vol. abs/1811.02597, 2018. [Online]. Available: <http://arxiv.org/abs/1811.02597>
- [49] D. Graves, K. Rezaee, and S. Scheideman, "Perception as prediction using general value functions in autonomous driving applications," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, ser. IROS 2019, 2019.
- [50] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International Conference on Machine Learning*, ser. ICML'14, vol. 32, 2014, pp. 1–387–1–395.
- [51] M. Sugiyama, T. Suzuki, and T. Kanamori, "Density ratio estimation: A comprehensive review," *RIMS Kokyuroku*, pp. 10–31, 2010.
- [52] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, 1988.
- [53] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *International Conference on Learning Representations*, Puerto Rico, 2016.
- [54] B. Paden, M. Cáp, S. Z. Yong, D. S. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *CoRR*, vol. abs/1604.07446, 2016. [Online]. Available: <http://arxiv.org/abs/1604.07446>
- [55] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, 2011.
- [56] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.
- [57] T. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *International Conference on Learning Representations*, 2016.
- [58] G. Uhlenbeck and L. Ornstein, "On the theory of the brownian motion," *Physical review*, vol. 36, pp. 823–841, 1930.

### A. Jackal Robot Experiments

The Jackal robot is equipped with a 5MP camera using a wide angle lens, an IMU sensor and an indoor Hokuyo UTM-30LX LIDAR sensor with a  $270^\circ$  scanning range and 0.1 to 10 meters scanning distance. The objective is to drive the robot using a camera in the center of a lane marked with tape using only data collected in the real-world. Training directly from real world experience requires addressing multiple issues such as minimizing wear and tear on the robot, and the need of human supervision during training in order to prevent or resolve robot crashes and recharge the battery.

There are two learned controllers, called GVF-BCQ and E2E-BCQ respectively, and one classical baseline called MPC (model predictive control). The learned controllers output a steering angle  $a_t^{steer}$  and target speed  $a_t^{speed}$  based on the image taken by the camera in order drive centered in a closed loop road outlined with tape on a carpeted floor. The MPC outputs a steering angle  $a_t^{steer}$  and target speed  $a_t^{speed}$  based on localization of the robot on a prior constructed map of the environment used to follow a sequence of waypoints supplied to the robot beforehand. Localization, map and waypoints are needed to train the GVF-BCQ controller; however, this information is no longer used during testing. GVF-BCQ tolerates noisy, low-accuracy localization methods that might otherwise be not accurate enough for smooth control with MPC or other methods.

1) *Training and testing environment:* The environment used for collecting data and evaluating the agents included two types of carpet floor - each with a different amount of friction. The evaluation roads were done on one carpet only which was included in only about 20% of the training data; the rest of the training data was on another type of carpet flooring to provide a generalization challenge for the learned controllers. The friction was quite high on both carpets and it caused the agent to shake the camera violently while turning since the robot employs differential steering; tape on the wheels helped reduce the friction a bit. Nevertheless, localization techniques using wheel odometry was deemed unsuitable and LIDAR-based localization was used instead. LIDAR localization was not highly accurate but was sufficient; our tests showed that it was repeatable to within roughly 5 centimeters which is deviation of upto about 13% from the center of the road.

Nine closed loop roads were created by placing left and right lanes markings on the carpeted floor separated to form a consistent lane width of roughly 76 centimeters for all the roads; some error in lane width measurements were tolerated when creating our roads. 6 of the roads were selected for collecting data to train our learned agents. Data was collected in both directions. The remaining 3 roads were reserved for testing the performance of the policies. In addition, each test road included damaged variants for a total of 6 test roads.

The poses and orientations of a sequence of waypoints were established to denote the center of lane which is the desired path of the agent; this was used to train our agents

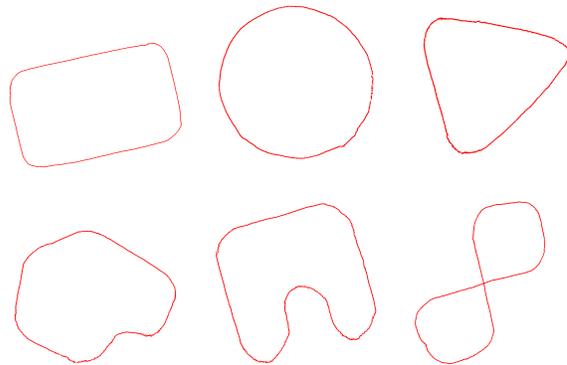


Fig. 7: Six roads for training. The second row shows more complex road structure. The rectangle road has rectangular edges at all corners.

on the training roads and evaluate them on the test roads. The center waypoints were collected by an expert manually and carefully navigating the Jackal robot on the road in the approximate center of lane; while this was inaccurate, it was not found to harm learning since the GVF-BCQ approach was able to generalize and learn features to drive the vehicle centered in the lane. The LIDAR-based localization produced poses periodically to form the center waypoints; these were cleaned up by removing overlapping waypoints to form a closed loop path. However, it did not produce poses at a consistent sampling frequency and thus a linear interpolation method was used to fill in the gaps and provide localization and center waypoint information at every time step. The purpose of the center waypoints was to compute the road angle and lane centeredness of the robot in the road at any given time which is needed to train the GVF predictions and evaluate all of our controllers.

The center waypoints for the training and testing roads are depicted in Figure 7 and Figure 8, respectively. The first three training roads were simple geometric shapes while the other three were a bit more complex. The first test road was the most similar to the training data where the outer edge of the four corners were rounded. The second test road was an oval shape to evaluate how well the agent maintained a turn that, unlike the circle training road, requires the steering angle to be modulated rather than remain constant. The third test road was a complex shape with multiple sudden turns that was very different from any of the roads in the training data set. This tests generalization to new roads and confirms that the agent is not memorizing an action sequence to remain centered in the path. All methods were evaluated at 0.25 m/s and 0.4 m/s maximum speeds and clock-wise (CW) and counter clock-wise (CCW) directions. In order to test robustness, all three test roads were altered by degrading or damaging the lane markings. The complex test road also included distracting markings as shown in Figure 1. An example of an image received from the robot with and without damage to the lane markers is shown in Figure 9.

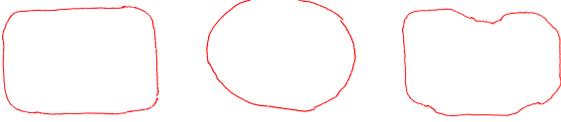


Fig. 8: Three roads for testing. Left to right: (1) rectangle with rounded corners; (2) oval; (3) complex.

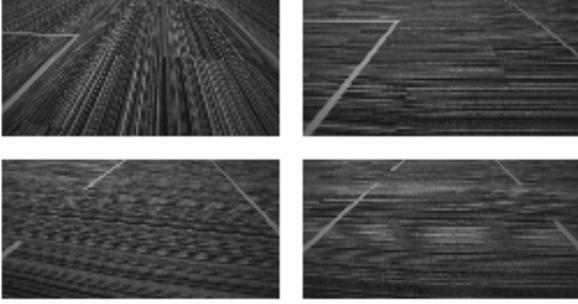


Fig. 9: Input images of normal lane markings (top row) and damaged lane markers (bottom row).

2) *Data collection*: Both learned agents – GVF-BCQ and E2E-BCQ – were trained with batch RL [23] where the GVF-BCQ method learned a predictive representation. Rosbags were collected with the Jackal robot each containing around 30 minutes to 2 hours of data for a total of 40 hours of data containing approximately 1.5 million images. The camera image, localization pose, IMU measurement, and action taken by the controller was recorded at each time step. The Jackal robot was controlled using a random walk controller that was confined to the road area to provide sufficient and safe exploration. The map was created with Hector SLAM [55] and the localization produced by Adaptive Monte-Carlo Localization [56].

The random walk controller was based on a pure pursuit controller, where action taken at each time step is defined by

$$\begin{aligned} a_t^{steer} &= \text{clip}(\text{angle}(p_t, p_{k(t)}^*) - \theta_t^z, -\pi/2, \pi/2) \\ a_t^{speed} &= \text{clip}(v_{k(t)}^*, 0.2, 0.5) \end{aligned} \quad (9)$$

where  $\theta_t^z$  and  $p_t$  were the yaw angle and the 2-dimensional position of the robot in the real world (obtained from localization),  $p_{k(t)}^*$  and  $v_{k(t)}^*$  were the target position and linear velocity at time  $t$ ,  $\text{clip}(x, \text{MinVal}, \text{MaxVal})$  is the clip function that operates in scalar and vector element-wise, and  $\text{angle}(p_t, p_{k(t)}^*)$  is the function that returns the yaw angle in the real world of a vector that points from  $p_t$  to  $p_{k(t)}^*$ . The target position  $p_{k(t)}^*$  and linear velocity  $v_{k(t)}^*$  were encapsulated in the next target pose at index  $k(t)$  in the

sequence of target poses:

$$\begin{aligned} k(1) &= 1 \\ k(t+1) &= \begin{cases} k(t) + 1 & \text{if } \|p_t - p_{k(t)}^*\|_2 < 0.025 \\ k(t) & \text{otherwise} \end{cases} \end{aligned} \quad (10)$$

Thus, the robot advanced to the next target position and linear velocity in the target pose sequence once it arrived within 2.5 centimeters of the current target position. In order to provide efficient and safe exploration that can be confined to the road area, the target position  $p_j^*$  was based on the position  $\tilde{p}_j$  of the center waypoints collected earlier with some noise added to provide the necessary exploration to learn the predictions and policy:

$$\begin{aligned} p_j^* &= \tilde{p}_{j \% N} + \varepsilon_j^p \\ v_j^* &= \begin{cases} v_{j-1}^* + \varepsilon_j^v & \text{if } j > 1 \\ 0.35 & \text{if } j = 1 \end{cases} \end{aligned} \quad (11)$$

where  $N$  is the number of points that define the center waypoints of the closed loop road.  $\varepsilon_j^p$  and  $\varepsilon_j^v$  were the noises added at each time step:

$$\begin{aligned} \varepsilon_j^p &= \begin{cases} \text{clip}(\varepsilon_{j-1}^p + \mathcal{N}(0, 0.02 * \mathbb{1}), -0.3, 0.3) & \text{if } j > 1 \\ [0, 0]^T & \text{if } j = 1 \end{cases} \\ \varepsilon_j^v &= \mathcal{N}(0, 0.02) \end{aligned} \quad (12)$$

The noises for the poses were clipped so that the robot would not explore too far outside the road area.

The rosbags were processed to synchronize the sensor data streams at a fixed sample frequency of 10Hz and compute the lane centeredness  $\alpha_t$ , road angle  $\beta_t$ , and speed  $\nu_t$  of the robot at each time step:

$$\begin{aligned} \nu_t &= \text{knn}(p_t, S_t) \\ \alpha_t &= \text{clip}\left(\frac{\|p_t - \nu_t\|_2}{H}, -1.0, 1.0\right) \\ \beta_t &= \text{clip}(\text{angle}(p_t, \nu_t) - \theta_t^z, -\pi/2, \pi/2) \end{aligned} \quad (13)$$

where  $\text{knn}(x, S)$  returns  $\nu$  as the closest point to  $x$  in  $S$  using k-nearest neighbor and  $H = 38$  centimeters as the half lane width.  $S_t$  is a pruned set of center waypoints where  $S_t = \{\tilde{p}_t\}$  for all roads, except for the figure 8 road in the lower right of Figure 7 where  $S_t$  was based on a sliding window to prevent issues with k-nn at the intersection:

$$S_t = \begin{cases} \{\tilde{p}_t\} & \text{if } j = 1 \\ \{\tilde{p}_{t=I_{t-1}-w \rightarrow I_{t-1}+w}\} & \text{if } j > 1 \end{cases} \quad (14)$$

where  $I_{j-1}$  is the index of  $\nu_{t-1}$  in  $S_{t-1}$  at the previous time step and  $w = 10$  is the size of the sliding window. Negative indices are wrapped to the beginning of the waypoint list. The speed was estimated using the change in position over a single time step which was quite noisy but more reliable than speed returned from the robot's odometry. Due to computation constraints on the robot, the localization messages were output at less than 10Hz; thus, a linear interpolation was used to fill in missing poses and orientations in the data and synchronize the data streams.

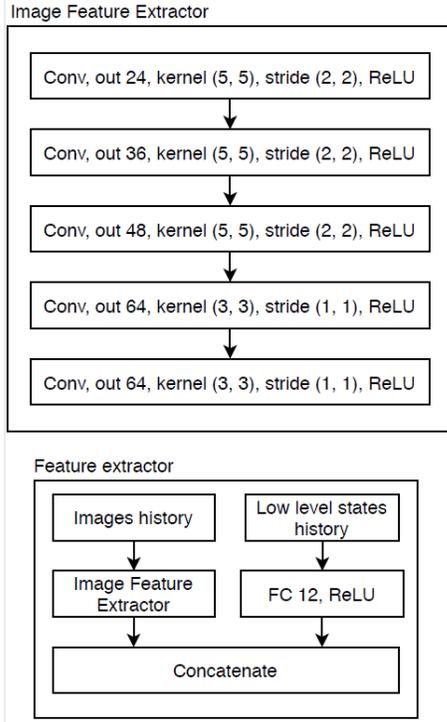


Fig. 10: Model of the feature extractor of the robot’s state concatenates features from the image with low-dimensional state information from the robot like speed and last action.

Images from camera with original size  $1920 \times 1080$  were cropped to  $960 \times 540$  region in the center and then additionally top-cropped by 60. The images were then downsampled by a factor of 8 in both spatial dimensions to give a final size of  $120 \times 60$  and then converted to gray scale. To improve generalization in deep learning and balance the left-right biases in the data, augmented data was created with horizontally flipped images along with the corresponding signs of the lane centeredness  $\alpha_t$ , road angle  $\beta_t$ , and steering action  $a_t^{steer}$  flipped.

3) *GVF-BCQ Training*: The predictive neural network used was trained using the offline version of the predictive learning algorithm 2. The transitions in the data were loaded into a replay buffer in the same order that the transitions were observed in the rosbag where mini-batches of size 128 were sampled from the growing replay buffer and used to update the GVFs. The GVFs were updated for 5 million steps followed by BCQ for an additional 5 million steps for a total of 10 million steps. The order that the rosbags were loaded into the replay buffer was randomized. The replay buffer had a maximum capacity of 0.5 million samples; once the replay buffer was filled, the oldest samples were removed. Training began once the replay buffer reached 0.1 million samples. While an alternative approach would have been to sample mini-batches from the entire data set from the beginning, our approach was found to be effective and required minimal changes to the data loader of the online version of the algorithm.

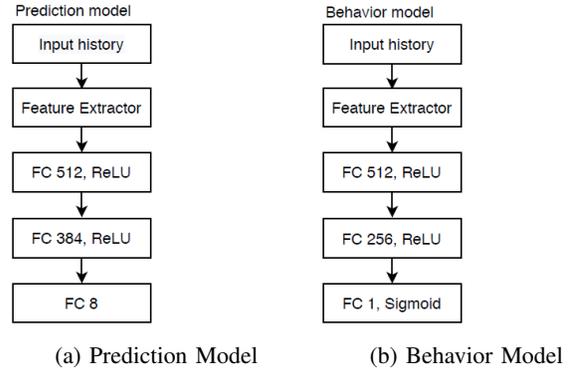


Fig. 11: Neural network models for (a) Prediction Model that produces  $\psi(s)$ , and (b) Behavior Model that estimates  $\mu(a|s)$

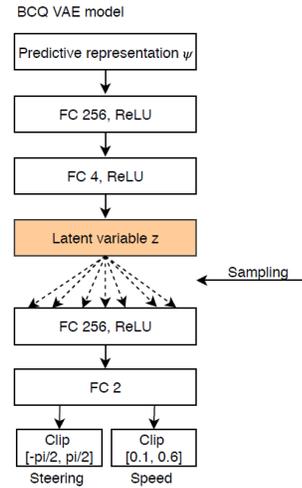


Fig. 12: Model of the GVF-BVQ variational auto-encoder (VAE).

The feature extractor model of the robot state is depicted in Figure 10. Estimating the behavior distribution for the GVF predictions was done with  $\eta(a|s)$  as a uniform distribution defined on the interval  $[-\pi/2, \pi/2]$  for the steering action and a uniform distribution defined on the interval  $[0, 1]$  for the target speed action. The neural network model used to learn the GVFs predictions that produce  $\phi$  are depicted in Figure 11a. The model used to estimate the behavior distribution  $\mu(a|s)$  is shown in Figure 11b. The BCQ network models that are used to learn the policy of the agent were all relatively small fully connected networks with hidden layer of size 256 as shown in Figures 12, 13a, and 13b.

The predictive representation  $\psi$  is a vector of length 11 consisting of  $\phi$  (vector of predictions of size 8), the last steering action, the last target speed action and the current robot speed. The latent vector dimension was 4 which was a Normal distribution parameterised by mean and log standard deviation. All networks used ReLU activation for the hidden layers and linear activation for the outputs. The action output from the actor and VAE were clipped to  $[-\pi/2, \pi/2]$  for steering and  $[0.1, 0.6]$  for the target speed.

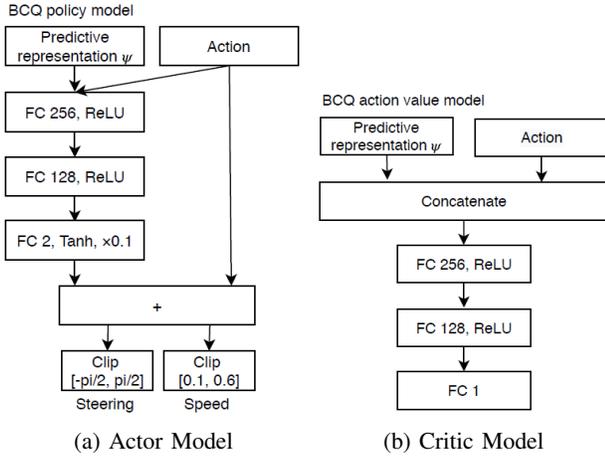


Fig. 13: Neural network models for GVF-BCQ (a) Actor, and (b) Critic

The weight of the KL divergence loss used in BCQ was 0.5. The learning rate was  $10^{-4}$  for both GVF and BCQ model training. Figure 14 shows the training curves for the predictive representation (GVFs) including the temporal-difference loss, behavior model loss and mean importance sampling ratio.

4) *End-to-end BCQ Baseline Training*: Nearly the same training setup used for GVF-BCQ was also applied to the E2E-BCQ method. The hyperparameters, training settings, activation functions for the output and action clipping are exactly the same as GVF-BCQ unless noted otherwise. All the networks in E2E-BCQ including the VAE, actor and critic shared the same feature extractor as the GVF-BCQ shown in Figure 10. The neural network models are given in Figures 15b, and 15a.

For a fair comparison, E2E-BCQ was trained for 10 millions update steps - the same number as GVF-BCQ method which divides the update budget evenly where 5 million updates are applied to the GVF predictions and 5 million updates are applied to BCQ networks. The agent was then tested on the rectangle test road and it was found that E2E-BCQ performed very poorly. The agent was too slow reaching an average speed of about 0.18 m/s whereas the GVF-BCQ method was able to reach double that speed. In addition, E2E-BCQ steered very poorly and was often not centered in the lane; unlike the GVF-BCQ method which was observed to be quite robust, the E2E-BCQ method sometimes drove out of the lane where an emergency stop was needed to prevent collision. For this reason, E2E-BCQ was only compared to GVF-BCQ on the rectangle test road; and we focused on comparisons against the MPC controller which was more robust than E2E-BCQ. A detailed evaluation of E2E-BCQ is shown in Figure 18 and Table I.

5) *MPC Baseline*: An MPC baseline using standard ros nodes available for the Jackal robot were used for controlling the Jackal robot. The baseline was tuned for 0.4 m/s; however, it was challenging to achieve good performance due to limited computation power for the look ahead, noisy localization with LIDAR scan matching and inaccurate modeling of the steering

characteristics on carpet floors. The best performance was achieved for 0.25 m/s but significant oscillation was observed for 0.4 m/s that was very challenging to completely eliminate. The center waypoints provided as input to the MPC controller were processed so that the minimum distance between two consecutive waypoints was 2.5cm; the waypoints were then downsampled by a factor of 16 in order to increase their separation and increase the look ahead distance; different downsampling factors was tested but oscillation was never completely eliminated. The MPC had an optimization window of 5 steps into the future; this was limited by computation available on the Jackal robot’s on-board computer. This look ahead ensured the MPC was far enough into the future for real-time control.

6) *Test Results*: This section provides a detailed comparison of GVF-BCQ and the baselines at different speeds and directions. For both GVF-BCQ and E2E-BCQ methods, the actor produced the steering and desired speed and thus the agent was able to modulate its own speed and slow down as necessary in advance of sharp turns. The speed command was clipped at the maximum target speed. The controllers started at the same position and heading angle and they were allowed to run for exactly 300 seconds. The agents were evaluated based on the following criteria:

- Reward per second:  $\frac{1}{N} \sum_{t=1}^N r_t$
- Average speed:  $\frac{1}{N} \sum_{t=1}^N v_t$
- Average absolute lane centeredness:  $\frac{1}{N} \sum_{t=1}^N |\alpha_t|$
- Average absolute road angle:  $\frac{1}{N} \sum_{t=1}^N |\beta_t|$
- Near out of lane<sup>4</sup>:  $\frac{1}{N} \sum_{t=1}^N \mathbb{1}_{|\alpha_t| > 0.75}$ <sup>5</sup>.
- First Order Jerk<sup>6</sup>:  $\frac{1}{N-1} \sum_{t=1}^{N-1} |a_{t+1} - a_t|$
- Second Order jerk<sup>7</sup>:  $\frac{1}{N-2} \sum_{t=1}^{N-2} |(a_{t+2} - a_{t+1}) - (a_{t+1} - a_t)|$

A comparison of GVF-BCQ and E2E-BCQ is given in Tables IV and V. Experiments are named according to the method used, the selected target speed and the direction of the road loop (i.e. counter-clock-wise versus clock-wise). For example, GVF-BCQ-0.4-CCW points to the test of the GVF-BCQ controller with 0.4 m/s target speed in the counter-clock-wise direction.

Finally, the GVF-BCQ method generalized well to damaged lane markings and distractions in the visual images as shown in the similar scores and similar distributions in Figure 17 and Table VI and VII. Experiments on roads with damaged lane markings are denoted with suffix -D.

Details evaluations against the MPC baseline are shown in Tables VIII and IX. In our evaluation at 0.25 m/s in the counterclockwise direction, the gap between the controllers narrowed but GVF-BCQ still out-performed MPC overall. A

<sup>4</sup>ratio of time steps where the agent’s absolute lane centeredness is greater than 0.75

<sup>5</sup>Where  $\mathbb{1}$  is the indicator function.

<sup>6</sup>First order jerk is the absolute change of action taken by the agent in one time step. Lower jerk scores are better. Both steering and speed actions considered separately.

<sup>7</sup>Second order jerk is the absolute change of the first order jerk in one time step. Lower jerk scores are better. Both steering and speed actions considered separately.

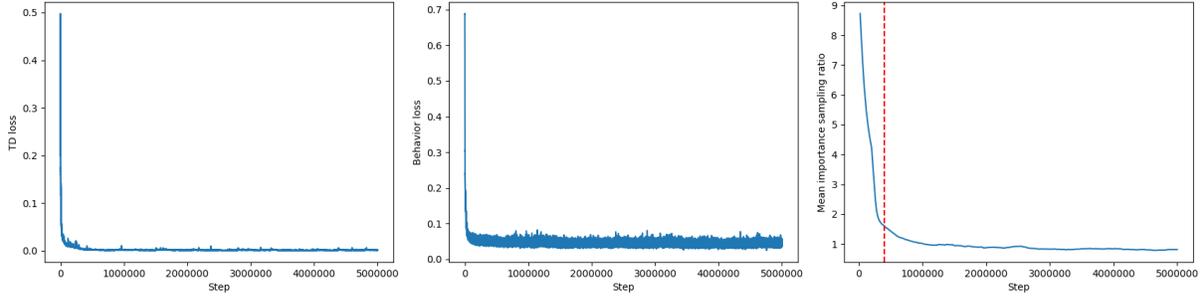


Fig. 14: TD loss, behavior loss and mean importance sampling ratio in the buffer over training steps. Red vertical dash line is the point when the buffer is full.

TABLE IV: Comparison of GVF-BCQ and E2E-BCQ on the Rectangle test road at 0.4 m/s

	Experiment	Reward per second $\uparrow$	Average speed $\uparrow$	Average off-center (normalized) $\downarrow$	Average off-angle $\downarrow$	Out of lane $\downarrow$
Rectangle shape	GVF-BCQ-0.4-CCW	<b>2.6835</b>	<b>0.3205</b>	<b>0.1345</b>	<b>0.1315</b>	<b>0.0%</b>
	E2E-BCQ-0.4-CCW	1.2578	0.1816	0.2558	0.2414	3.76%
	GVF-BCQ-0.4-CW	<b>2.2915</b>	<b>0.3140</b>	<b>0.2217</b>	<b>0.1586</b>	<b>0.0%</b>
	E2E-BCQ-0.4-CW	-0.1302	0.1710	0.9927	0.3034	54.18%

TABLE V: Comparison of GVF-BCQ and E2E-BCQ jerk levels on the Rectangle test road at 0.4 m/s

	Experiment	First order speed jerk $\downarrow$	Second order speed jerk $\downarrow$	First order steering jerk $\downarrow$	Second order steering jerk $\downarrow$
Rectangle shape	GVF-BCQ-0.4-CCW	<b>0.0356</b>	<b>0.2532</b>	<b>0.2251</b>	<b>1.3403</b>
	E2E-BCQ-0.4-CCW	0.0154	0.2266	0.1109	1.4240
	GVF-BCQ-0.4-CW	<b>0.0311</b>	<b>0.2149</b>	<b>0.1995</b>	<b>1.1850</b>
	E2E-BCQ-0.4-CW	0.0148	0.1937	0.1174	1.3514

TABLE VI: Evaluation of the robustness of GVF-BCQ method on damaged lane markings on all the test roads

	Experiment	Reward per second $\uparrow$	Average speed $\uparrow$	Average off-center (normalized) $\downarrow$	Average off-angle $\downarrow$	Out of lane $\downarrow$
Rectangle shape	GVF-BCQ-0.4-CCW	2.6835	0.3205	<b>0.1345</b>	<b>0.1315</b>	0.0%
	GVF-BCQ-0.4-CCW-D	<b>2.7407</b>	<b>0.3261</b>	0.1358	0.1351	0.0%
Oval shape	GVF-BCQ-0.4-CCW	<b>2.4046</b>	<b>0.3501</b>	<b>0.2754</b>	0.2125	<b>1.45%</b>
	GVF-BCQ-0.4-CCW-D	2.0728	0.3279	0.3285	<b>0.2089</b>	7.19%
Complex shape	GVF-BCQ-0.4-CCW	<b>2.3501</b>	0.3129	<b>0.2221</b>	<b>0.1817</b>	<b>0.0%</b>
	GVF-BCQ-0.4-CCW-D	2.1059	<b>0.3284</b>	0.3125	0.2365	9.42%

clear advantage of GVF-BCQ is the stability and smoothness of control achieved at the higher speeds. Our proposed GVF-BCQ method beats the MPC in reward and was better on all tracks at both speed values without access to localization information during testing. The reason for this can be explained by looking at the average lane centeredness of the agent. The MPC performed as well as the GVF-BCQ method while maintaining good speed; however, it fails at keeping the vehicle in the center of the lane. The reason may be due to a number of different factors including possible inaccuracies in the MPC forward model resulting from the friction between the wheels and the carpet in the test runs, especially at higher speeds. The MPC suffered from many near out of lane events and had trouble staying within the lane markings of the oval road. The GVF-BCQ method was

better at controlling steering, leading to much higher average reward even though it had lower average speed at 0.4 m/s max speed. Additionally, the GVF-BCQ method was much better in achieving smooth control. These points are reflected in Figure 19 on the rectangle test track where the MPC lane centeredness distribution is skewed to one side and its steering action distribution has two modes that are far from zero while the GVF-BCQ method distributions are more concentrated around zero.

In order to provide more insight into the performance of the controllers, we also investigated the distributions of  $\alpha_t$ ,  $\beta_t$ ,  $v_t$  and  $a_t$  in Figures 17, 18, and 19.

<sup>7</sup>Note that measured vehicle speed might not be equal to speed action from the agent due to physical constraints of the environment and noises in measurement.

TABLE VII: Evaluation of the jerk of GVF-BCQ method on damaged lane markings on all the test roads

	Experiment	First order speed jerk ↓	Second order speed jerk ↓	First order steering jerk ↓	Second order steering jerk ↓
Rectangle shape	GVF-BCQ-0.4-CCW	<b>0.0356</b>	<b>0.2532</b>	<b>0.2251</b>	<b>1.3403</b>
	GVF-BCQ-0.4-CCW-D	0.0383	0.2715	0.2303	1.4620
Oval shape	GVF-BCQ-0.4-CCW	0.0348	<b>0.2423</b>	0.2191	<b>1.4632</b>
	GVF-BCQ-0.4-CCW-D	<b>0.0334</b>	0.2953	<b>0.2094</b>	1.6612
Complex shape	GVF-BCQ-0.4-CCW	<b>0.0341</b>	<b>0.2540</b>	<b>0.2272</b>	<b>1.5306</b>
	GVF-BCQ-0.4-CCW-D	0.0437	0.3608	0.2897	2.0946

TABLE VIII: Comparison of GVF-BCQ method and MPC on all the test roads at different speeds and directions

	Experiment	Reward per second ↑	Average speed ↑	Average off-center (normalized) ↓	Average off-angle ↓	Out of lane ↓
Rectangle shape	GVF-BCQ-0.4-CCW	<b>2.6835</b>	0.3205	<b>0.1345</b>	<b>0.1315</b>	<b>0.0%</b>
	MPC-0.4-CCW	0.9700	<b>0.3833</b>	0.5252	0.1943	20.42%
	GVF-BCQ-0.4-CW	<b>2.2915</b>	0.3140	<b>0.2217</b>	<b>0.1586</b>	<b>0.0%</b>
	MPC-0.4-CW	0.1282	<b>0.3836</b>	0.9086	0.1916	67.86%
Oval shape	GVF-BCQ-0.25-CCW	<b>2.1442</b>	<b>0.2467</b>	<b>0.1098</b>	<b>0.1181</b>	<b>0.0%</b>
	MPC-0.25-CCW	1.1971	0.2412	0.1218	0.1308	0.0%
	GVF-BCQ-0.4-CCW	<b>2.4046</b>	0.3501	<b>0.2754</b>	0.2125	<b>1.45%</b>
	MPC-0.4-CCW	0.8928	<b>0.3825</b>	0.5293	<b>0.1963</b>	22.75%
Complex shape	GVF-BCQ-0.4-CW	<b>2.4848</b>	0.3658	<b>0.2953</b>	<b>0.1922</b>	<b>0.0%</b>
	MPC-0.4-CW	-0.7168	<b>0.3836</b>	1.3182	0.2095	91.22%
	GVF-BCQ-0.25-CCW	<b>1.5112</b>	<b>0.2473</b>	<b>0.3645</b>	0.1466	<b>3.32%</b>
	MPC-0.25-CCW	0.0225	0.2296	0.9565	<b>0.1381</b>	87.92%
Complex shape	GVF-BCQ-0.4-CCW	<b>2.3501</b>	0.3129	<b>0.2221</b>	<b>0.1817</b>	<b>0.0%</b>
	MPC-0.4-CCW	0.7172	<b>0.3845</b>	0.6407	0.2131	38.94%
	GVF-BCQ-0.4-CW	<b>2.3182</b>	0.3168	<b>0.2317</b>	<b>0.2150</b>	<b>0.06%</b>
	MPC-0.4-CW	0.4324	<b>0.3905</b>	0.7662	0.2264	52.23%
Complex shape	GVF-BCQ-0.25-CCW	<b>1.9326</b>	<b>0.2472</b>	0.1890	<b>0.1509</b>	<b>0.0%</b>
	MPC-0.25-CCW	1.1559	0.2435	<b>0.1664</b>	0.1720	0.0%

TABLE IX: Comparison of jerk of GVF-BCQ method and MPC on all the test roads at different speeds and directions

	Experiment	First order speed jerk ↓	Second order speed jerk ↓	First order steering jerk ↓	Second order steering jerk ↓
Rectangle shape	GVF-BCQ-0.4-CCW	<b>0.0356</b>	<b>0.2532</b>	<b>0.2251</b>	<b>1.3403</b>
	MPC-0.4-CCW	0.0832	0.7605	1.2542	8.1963
	GVF-BCQ-0.4-CW	<b>0.0311</b>	<b>0.2149</b>	<b>0.1995</b>	<b>1.1850</b>
	MPC-0.4-CW	0.0944	0.8916	1.4328	10.9570
Oval shape	GVF-BCQ-0.25-CCW	<b>0.0009</b>	<b>0.0112</b>	<b>0.1466</b>	<b>0.8890</b>
	MPC-0.25-CCW	0.0570	0.5272	0.6384	3.5208
	GVF-BCQ-0.4-CCW	<b>0.0348</b>	<b>0.2423</b>	<b>0.2191</b>	<b>1.4632</b>
	MPC-0.4-CCW	0.1026	0.9301	1.4119	8.9051
Complex shape	GVF-BCQ-0.4-CW	<b>0.0241</b>	<b>0.1638</b>	<b>0.1674</b>	<b>1.1451</b>
	MPC-0.4-CW	0.0847	0.7534	1.3957	9.0432
	GVF-BCQ-0.25-CCW	<b>0.0005</b>	<b>0.0061</b>	<b>0.0969</b>	<b>0.7614</b>
	MPC-0.25-CCW	0.0657	0.6273	0.4830	3.0566
Complex shape	GVF-BCQ-0.4-CCW	<b>0.0341</b>	<b>0.2540</b>	<b>0.2272</b>	<b>1.5306</b>
	MPC-0.4-CCW	0.0625	0.5846	1.2133	8.1747
	GVF-BCQ-0.4-CW	<b>0.0348</b>	<b>0.2339</b>	<b>0.2240</b>	<b>1.3911</b>
	MPC-0.4-CW	0.0809	0.7521	1.2861	8.7905
Complex shape	GVF-BCQ-0.25-CCW	<b>0.0006</b>	<b>0.0082</b>	<b>0.1696</b>	<b>1.0394</b>
	MPC-0.25-CCW	0.0525	0.4932	0.6457	3.6786

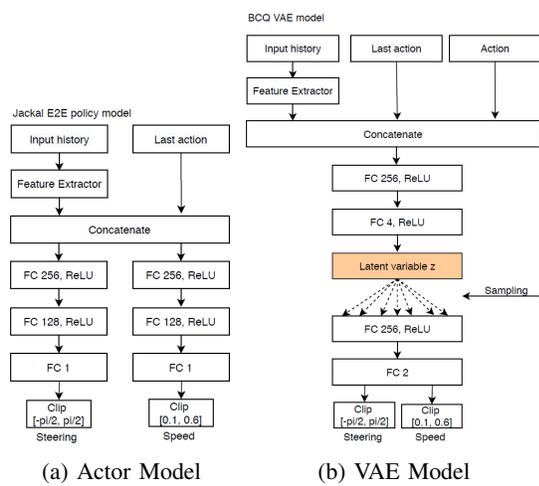


Fig. 15: Neural network models for E2E-BCQ (a) Actor, and (b) VAE. The Critic model is the same as DDPG in Figure 21

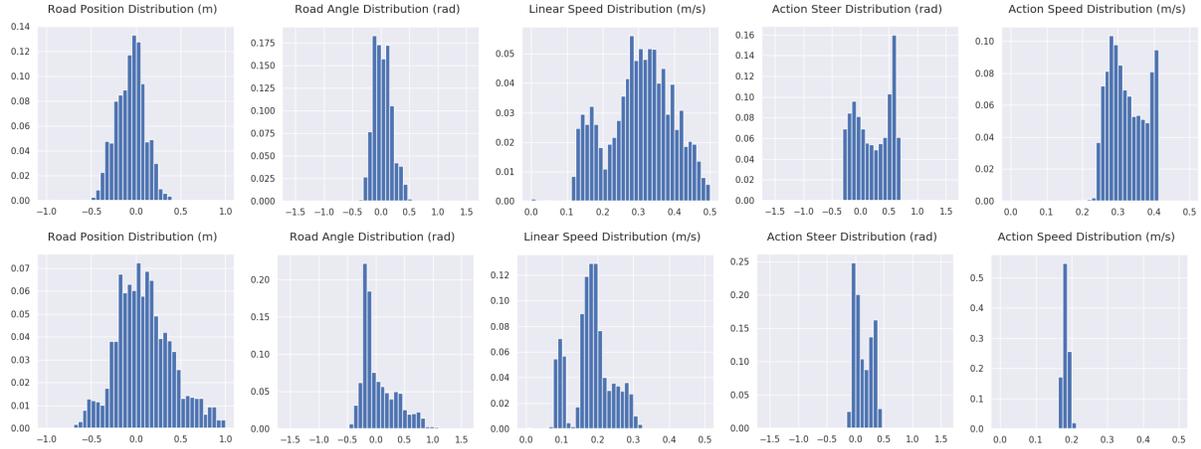


Fig. 16: Distribution of lane centeredness, road angle, speed and action distribution of GVF-BCQ and E2E-BCQ on the rectangle test track at 0.4 speed, counterclockwise direction.

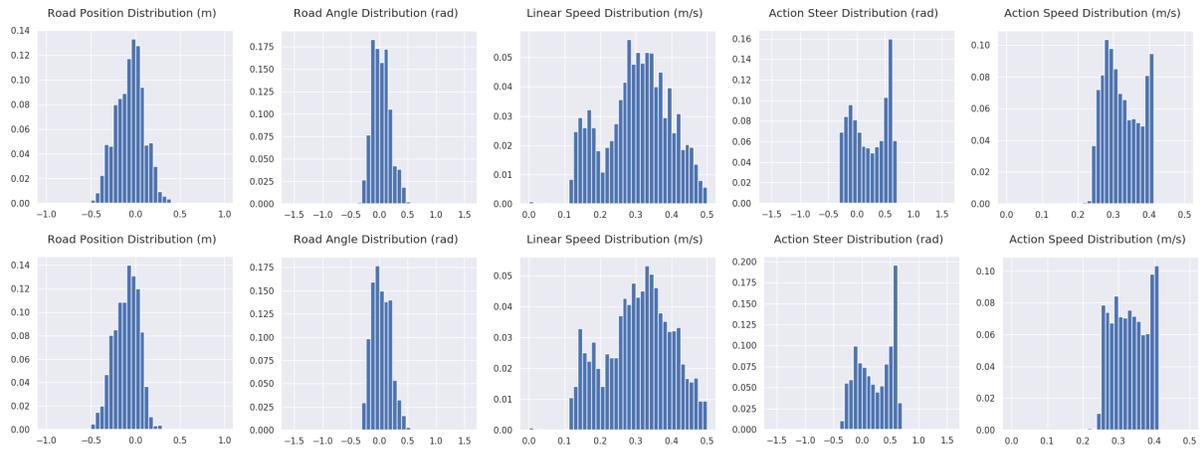


Fig. 17: Distribution of lane centeredness, road angle, speed and action distribution on the rectangle test road. From top to bottom: GVF-BCQ-0.4, GVF-BCQ-0.4 with lane marking damage on the rectangle road. The similarities highlight the robustness of GVF-BCQ to the introduction of damaged lanes.

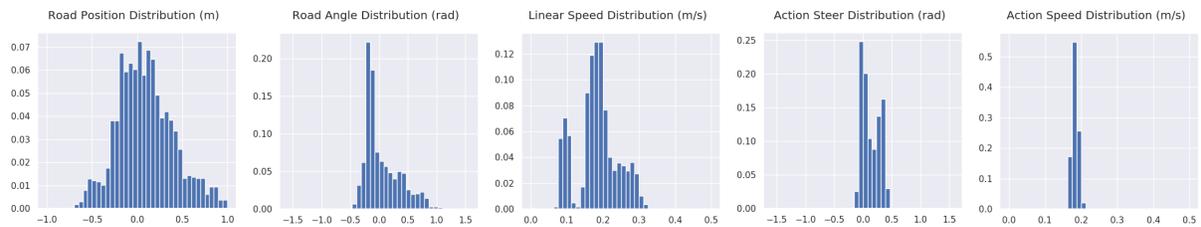


Fig. 18: Distribution of lane centeredness, road angle, speed and action distribution of E2E-BCQ on the rectangle test road at 0.4 speed, counterclockwise direction

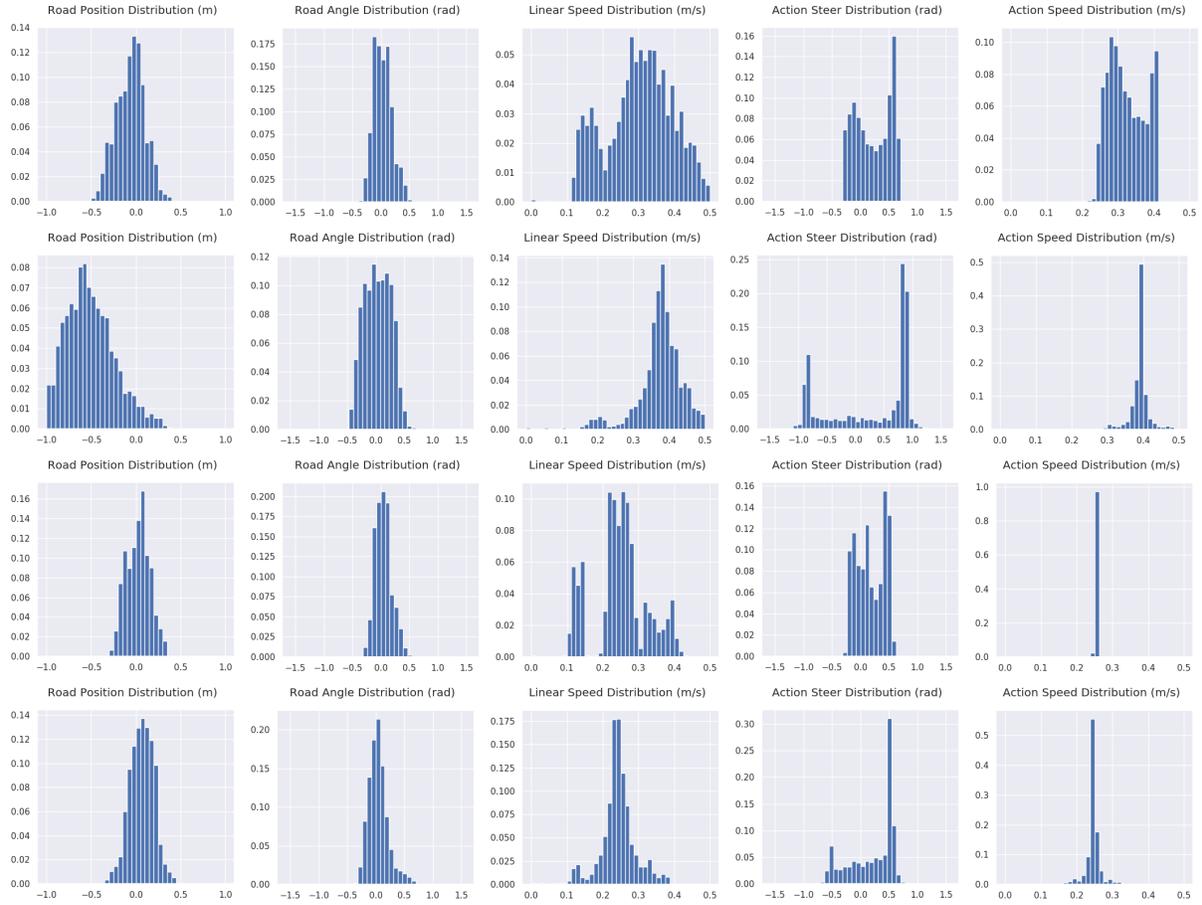


Fig. 19: Distribution of lane centeredness, road angle, speed and action distribution of GVF-BCQ and MPC at 0.4 m/s and 0.25 m/s on the rectangle test track. From top to bottom: GVF-BCQ-0.4-CCW, MPC-0.4-CCW, GVF-BCQ-0.25-CCW, MPC-0.4-CCW<sup>8</sup>

## B. TORCS Experiments

TORCS is a racing simulator used for learning to drive. All opponent vehicles were removed for these experiments as well as roads that were sloped. The goal of the agent is to maximize the future accumulation of the following reward:  $r_t = 0.0002v_t(\cos\beta_t + |\alpha_t|)$  where  $v_t$  is the speed of the vehicle in km/h,  $\beta_t$  is the angle between the road direction and the vehicle direction, and  $\alpha_t$  is the current lane centeredness. Termination occurs when either the agent leaves the lane or the maximum number of steps has been reached (1200 steps = 120 seconds) triggering a reset of the environment. Upon reset, a priority sampling method is used during training to select the next road to train on. The probability of sampling road  $i$  during a reset is given by

$$\frac{e^{-\frac{n_i}{\kappa}}}{\sum_{j=1}^N e^{-\frac{n_j}{\kappa}}} \quad (15)$$

where  $n_i$  is the number of steps that the agent was able to achieve the last time the road was sampled and  $\kappa$  controls the spread of the distribution. A value of  $\kappa = \frac{1}{N} \sum_{j=1}^N n_j$  was found to perform well. The initial probabilities are equal for all roads. This improved the efficiency in learning for all learned methods.

The TORCS environment was modified to provide higher resolution images in grayscale rather than RGB with most of the image above the horizon cropped out of the image. The grayscale images were 128 pixels wide by 64 pixels high. This allowed the agent to see more detail farther away which is very helpful in making long term predictions and is beneficial to both policy gradient methods and predictive learning.

1) *Training*: Two main learning algorithms are compared, along with their variants: our proposed GVF-DDPG (general value functions with deep deterministic policy gradient) and end-to-end DDPG. The parameters used to train the methods will be described in more detail here.

**GVF-DDPG Training:** Exploration followed the same approach as [57] where an Ornstein Uhlenbeck process [58] was used to explore the road; the parameters of the process ( $\theta = 1.0$ ,  $\sigma = 0.1$ ,  $dt = 0.01$ ) were tuned to provide a gradual wandering behavior on the road without excessive oscillations in the action. This improved the learning of the off-policy predictions for GVF-DDPG since the behavior policy  $\mu(a|s)$  was closer to the target policy  $\tau(a|s)$  of the predictions.

The GVF-DDPG approach learned 8 predictions: 4 predictions of lane centeredness  $\alpha$ , and 4 predictions of road angle  $\beta$ . Each of the 5 predictions had different values of  $\gamma$  for different temporal horizons: 0.0, 0.5, 0.9, 0.95, 0.97. This allowed the agent to predict how the road will turn in the future providing the necessary look ahead information for the agent to control effectively. The GVF predictors shared the same deep convolutional neural network as used on the Jackal robot where the convolutional layers were identical to the architecture in Figure 10 followed by three fully connected layers of 512, 384 and 8 outputs, respectively as shown in

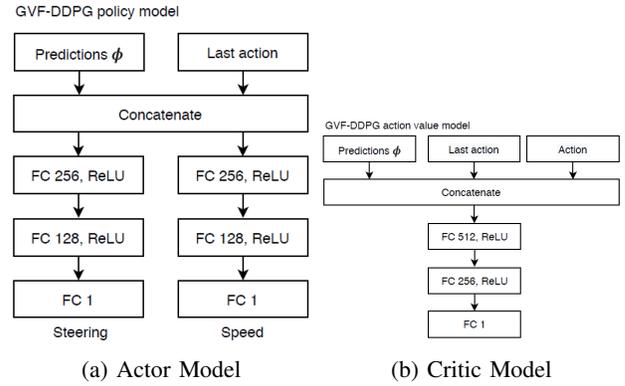


Fig. 20: Neural network models for GVF-DDPG (a) Actor, and (b) Critic

Figure 11a. The behavior estimator  $\mu(a|s)$  was identical with the one used on the Jackal robot in Figure 11b. The models for actor and critic models for GVF-DDPG are given in Figures 20a and 20b. Actions produced by the actor network was clipped to the range  $[-1.0, 1.0]$ . A linear transformation was applied to the target speed action to change the range of values from  $[-1.0, 1.0]$  to  $[0.5, 1.0]$ . The steering and target speed input into both policy and critic networks were normalized to range  $[-1.0, 1.0]$ .

A replay buffer of size 100,000 was used with a warmup of 10,000 samples. In order to not bias the replay buffer, the last layers of the actor network were initialized with a uniform distribution  $[-1e^{-3}, 1e^{-3}]$  for the weight and 0 for the bias. The learning rates for the actor network, critic network, predictor network and behavior policy network were  $1e^{-6}$ ,  $1e^{-4}$ ,  $1e^{-4}$ , and  $1e^{-4}$  respectively. Target networks [57] were used for the critic and actor networks with  $\tau = 0.001$  in order to make the bootstrapped prediction of the action-values more stable. However, target networks were not necessary for the GVF predictions or the behavior policy estimation. The reward for was scaled by 0.0002 to scale the action-values to fall within the range  $[-1.0, 1.0]$ .

**Baseline DDPG Training:** The two DDPG (deep deterministic policy gradient) [57] baselines were trained nearly identically where the only difference was the information provided in the observation. The first method called DDPG-Image is a vision-based approach where the image, current speed, and last action are provided to the agent; the only information available to the agent about the road is supplied via images. The second agent called DDPG-LowDim includes lane centeredness  $\alpha$  and road angle  $\beta$  as part of the agent's state. The purpose was to understand the value of this information in learning when supplied as a cumulant in GVF-DDPG during training only or supplied an input to the actor and critic during training *and* testing. It should be noted that DDG-LowDim was the only learned approach that used  $\alpha$  and  $\beta$  as inputs to the actor and critic networks during testing, whereas GVF-DDPG and DDPG-Image did not have access to this information during testing.

The network architectures for the DDPG actor and critic

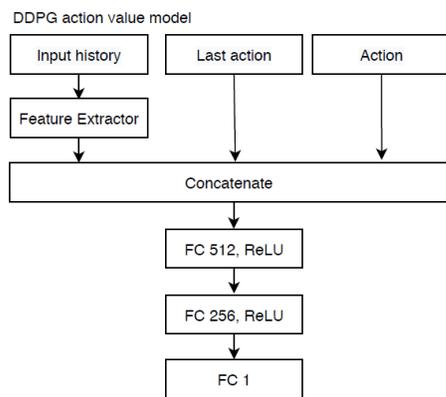


Fig. 21: Model of the DDPG critic network  $Q(s, a)$ .

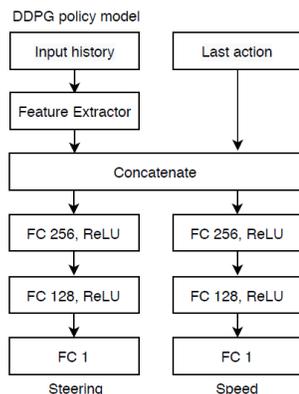


Fig. 22: Model of the DDPG actor network  $\pi(s)$ .

are given in Figures 22 and 21 respectively; they share the architecture for the feature extractor given in Figure 10.

The training setup for DDPG was the same as GVF-DDPG unless noted otherwise. An Ornstein Uhlenbeck process [58] was used to explore the road ( $\theta = 1.0$ ,  $\sigma = 0.1$ , and  $dt = 0.01$ ). Experimentally, it was found that the exploration parameters did not affect the learning performance of DDPG that much. Target networks were used with  $\tau = 0.001$ . The learning rates of the critic and actor networks were the same as those of GVF-DDPG.

2) *Experimental Results:* The experimental results were averaged over 5 runs and mean and standard deviations plotted based on performance measured on the test roads during training. The learning curves for the critic networks for each of the DDPG agents, as well as learning curves for GVF predictions and behavior estimation for GVF-DDPG are shown in Figure 23. The average episode length is shown in Figure 24. The average lane centeredness and road angle during each episode are plotted in Figures 25 and 26. We can see how the GVF-DDPG with predictions over multiple time scales is able to maintain lane centeredness and road angle better than GVF-DDPG with myopic prediction ( $\gamma = 0.0$ ) and future predictions with only  $\gamma = 0.95$ . The average lane centeredness and road angle is not substantially different though among the learned methods; however, DDPG-Image struggles a bit largely due to instability in learning as the high

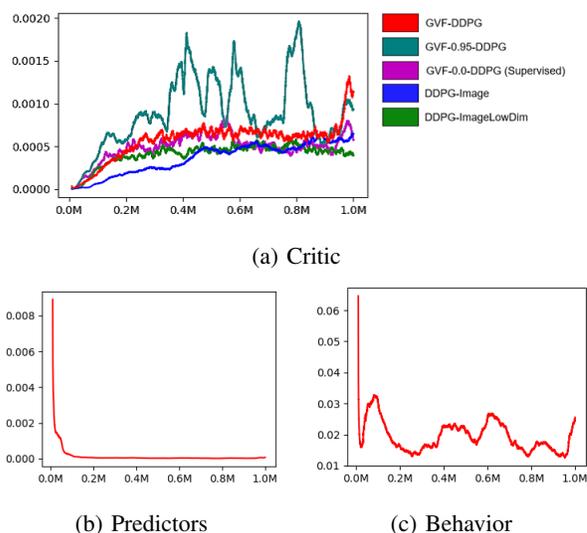


Fig. 23: Learning curves for (a) Q-values of the DDPG agents, (b) mean squared TD (temporal difference) errors of the GVF predictors, and (c) MSE of the behavior model estimator

variance is due to some failed runs where no learning occurs. Figure 27 shows the standard deviation in the change in the target speed action at each time step across an episode on each test road; this measures the jerkiness of the speed controller. GVF-DDPG and DDPG-LowDim are both able to control speed comfortably since the jerkiness is low. Finally, Figure 28 shows the lane centeredness on all six of the test roads during a final evaluation after training was completed. All six roads in the test set were challenging but the test roads a-speedway, alpine-2, and wheel-2 were especially challenging because the image of the roads were too different from the training roads. Nevertheless, on the dirt-4, evo-2-r, and spring roads, the lane centeredness of the the methods was quite good for all learned methods except for DDPG-Image. Note that while DDPG-LowDim performs well in learning to steer and control speed with lane centeredness and road angle, it required that information to learn to steer the vehicle which may be expensive or prohibitive to obtain in all situations such as in GPS-denied locations or locations where there is no map or it is out of date.

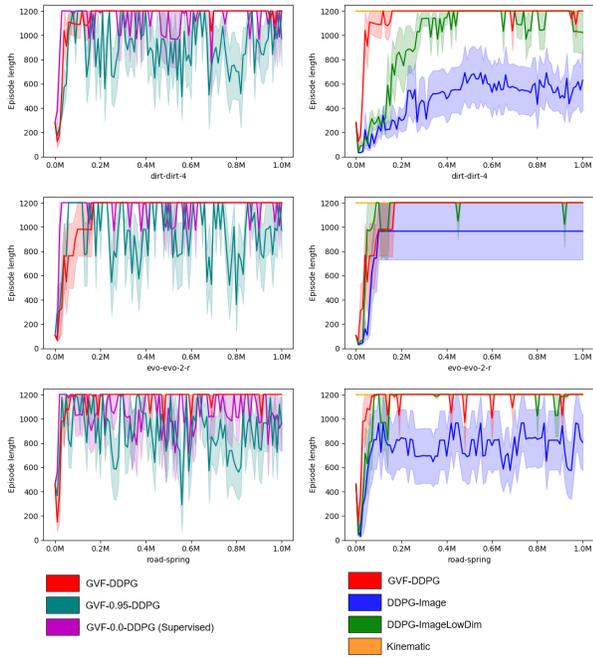


Fig. 24: Mean episode length during training for dirt-dirt-4, evo-evo-2 and road-spring

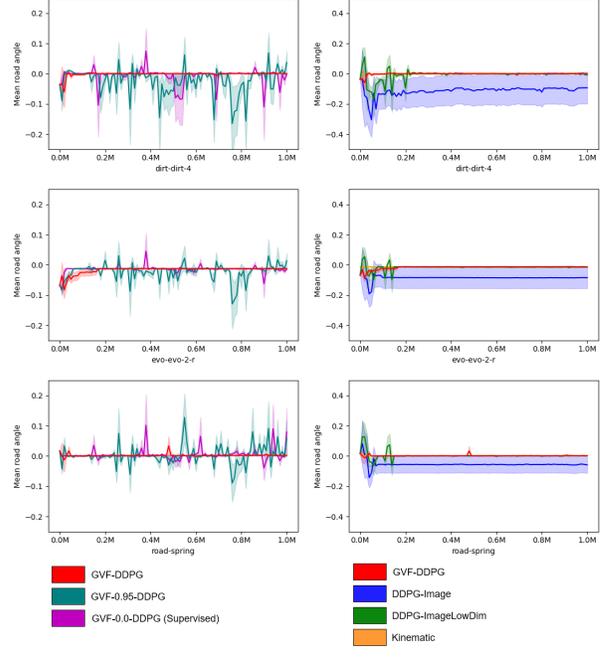


Fig. 26: Mean road angle during training for dirt-dirt-4, evo-evo-2 and road-spring

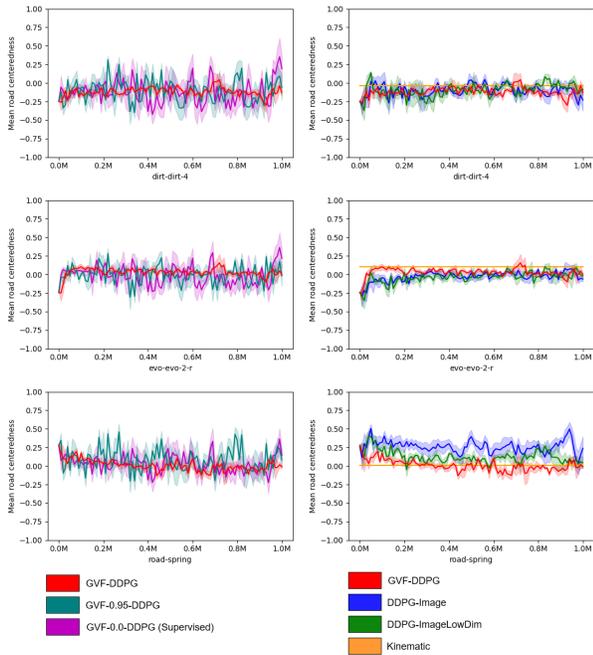


Fig. 25: Mean lane centeredness during training for dirt-dirt-4, evo-evo-2 and road-spring

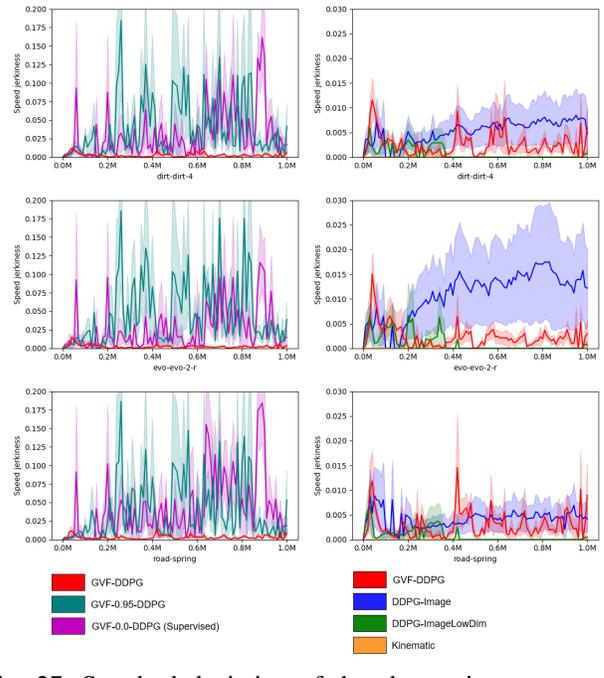


Fig. 27: Standard deviation of the change in target speed action during training for dirt-dirt-4, evo-evo-2 and road-spring

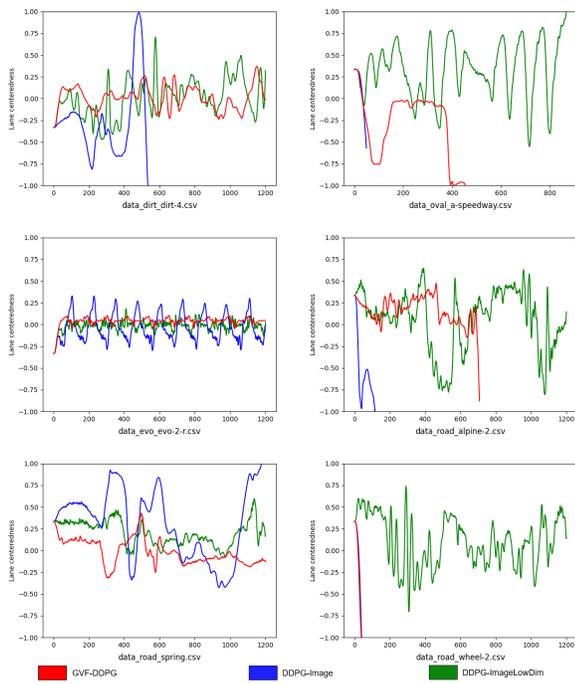


Fig. 28: The lane centeredness position on the (a) alpine-2, (b) evo-2-r, (c) dirt-4, (d) wheel-2, (e) spring, and (f) a-speedway roads in TORCS.

### C. Predictive Learning Algorithms

The algorithm used for learning counterfactual predictions (GVFs) online through interaction with an environment is given in Algorithm 1. In important distinction of this algorithm is that the distribution of the behavior policy used to collect the data does not need to be known.

---

**Algorithm 1** Online Counterfactual GVF training algorithm with unknown  $\mu(a|s)$

---

- 1: Initialize  $\phi^\tau(s)$ ,  $g(a, s)$ ,  $\eta(a|s)$ , and replay memory  $D$
  - 2: Observe initial state  $s_0$
  - 3: **for**  $t = 0, T$  **do**
  - 4:   Sample action  $a_t$  from unknown  $\mu(a_t|s_t)$
  - 5:   Execute action  $a_t$  and observe state  $s_{t+1}$
  - 6:   Compute cumulant  $c_{t+1} = c(s_t, a_t, s_{t+1})$
  - 7:   Compute continuation  $\gamma_{t+1} = \gamma(s_t, a_t, s_{t+1})$
  - 8:   Estimate behavior density value  $\hat{\mu}(a_t|s_t) = \frac{g(a_t, s_t)}{1-g(a_t, s_t)}\eta(a_t|s_t)$
  - 9:   Estimate importance sampling ratio  $\rho_t = \frac{\tau(a_t|s_t)}{\hat{\mu}(a_t|s_t)}$
  - 10:   Store transition  $(s_t, a_t, c_{t+1}, \gamma_{t+1}, s_{t+1}, \rho_t)$  in  $D$
  - 11:   Compute average importance sampling ratio in replay buffer  $D$  of size  $n$  with  $\bar{\rho} = \frac{1}{n} \sum_{j=1}^n \rho_j$
  - 12:   Sample random minibatch  $A$  of transitions  $(s_i, a_i, c_{i+1}, \gamma_{i+1}, s_{i+1})$  from  $D$  according to probability  $\frac{\rho_i}{\sum_{j=1}^n \rho_j}$
  - 13:   Compute  $y_i = c_{i+1} + \gamma_{i+1}\phi^\tau(s_{i+1}; \hat{\theta})$  for minibatch  $A$  for most recent parameters  $\hat{\theta}$
  - 14:   Update parameters  $\theta$  using gradient descent on (3) with gradient (6) over the minibatch  $A$
  - 15:   Sample random minibatch  $B$  of state action pairs  $(s_i, a_i)$  from  $D$  according to a uniform probability and assign label  $z = 1$  to each pair
  - 16:   Randomly select half the samples in the minibatch  $B$  replacing the action with  $a_t \sim \eta(a|s)$  and label with  $z = 0$  and storing the updated samples in  $\hat{B}$
  - 17:   Update behavior discriminator  $g(a, s)$  with labels  $z$  in the modified minibatch  $\hat{B}$  using binary cross-entropy loss
- 

With minor modifications, an offline version of the algorithm can be derived. This algorithm learns by reading the data in sequence and populating a replay buffer just as it would in online learning; the only difference is that the offline algorithm returns the action taken in the data. This allows the same algorithm and code for learning counterfactual predictions (GVF) to be used in either online or offline learning settings.

---

**Algorithm 2** Offline Counterfactual GVF training algorithm with unknown  $\mu(a|s)$

---

- 1: Initialize  $\phi^\tau(s)$ ,  $g(a, s)$ ,  $\eta(a|s)$ , and replay memory  $D$ ,
  - 2: Obtain the first state in the data file  $s_0$
  - 3: **for**  $t = 0, T$  **do**
  - 4:   Obtain action  $a_t$  recorded in the data file that sampled from an unknown  $\mu(a_t|s_t)$
  - 5:   Obtain next state  $s_{t+1}$  from the data file
  - 6:   Compute cumulant  $c_{t+1} = c(s_t, a_t, s_{t+1})$
  - 7:   Compute continuation  $\gamma_{t+1} = \gamma(s_t, a_t, s_{t+1})$
  - 8:   Estimate behavior density value  $\hat{\mu}(a_t|s_t) = \frac{g(a_t, s_t)}{1-g(a_t, s_t)}\eta(a_t|s_t)$
  - 9:   Estimate importance sampling ratio  $\rho_t = \frac{\tau(a_t|s_t)}{\hat{\mu}(a_t|s_t)}$
  - 10:   Store transition  $(s_t, a_t, c_{t+1}, \gamma_{t+1}, s_{t+1}, \rho_t)$  in  $D$
  - 11:   Compute average importance sampling ratio in replay buffer  $D$  of size  $n$  with  $\bar{\rho} = \frac{1}{n} \sum_{j=1}^n \rho_j$
  - 12:   Sample random minibatch  $A$  of transitions  $(s_i, a_i, c_{i+1}, \gamma_{i+1}, s_{i+1})$  from  $D$  according to probability  $\frac{\rho_i}{\sum_{j=1}^n \rho_j}$
  - 13:   Compute  $y_i = c_{i+1} + \gamma_{i+1}\phi^\tau(s_{i+1}; \hat{\theta})$  for minibatch  $A$  for most recent parameters  $\hat{\theta}$
  - 14:   Update parameters  $\theta$  using gradient descent on (3) with gradient (6) over the minibatch  $A$
  - 15:   Sample random minibatch  $B$  of state action pairs  $(s_i, a_i)$  from  $D$  according to a uniform probability and assign label  $z = 1$  to each pair
  - 16:   Randomly select half the samples in the minibatch  $B$  replacing the action with  $a_t \sim \eta(a|s)$  and label with  $z = 0$  and storing the updated samples in  $\hat{B}$
  - 17:   Update behavior discriminator  $g(a, s)$  with labels  $z$  in the modified minibatch  $\hat{B}$  using binary cross-entropy loss
-