LEARNING AN OFF-POLICY PREDICTIVE STATE REPRESENTATION FOR DEEP REINFORCEMENT LEARNING FOR VISION-BASED STEERING IN AUTONOMOUS DRIVING

Anonymous authors
Paper under double-blind review

ABSTRACT

An algorithm is introduced for learning a predictive state representation with off-policy temporal difference (TD) learning that is then used to learn to steer a vehicle with reinforcement learning. There are three components being learned simultaneously: (1) the off-policy predictions as a compact representation of state, (2) the behavior policy distribution for estimating the off-policy predictions, and (3) the deterministic policy gradient for learning to act. A behavior policy discriminator is learned and used for estimating the important sampling ratios needed to learn the predictive representation off-policy with general value functions (GVFs). A linear deterministic policy gradient method is used to train the agent with only the predictive representations while the predictions are being learned. All three components are combined, demonstrated and evaluated on the problem of steering the vehicle from images in the TORCS racing simulator environment. Steering from only images is a challenging problem where evaluation is completed on a held-out set of tracks that were never seen during training in order to measure the generalization of the predictions and controller. Experiments show the proposed method is able to steer smoothly and navigate many but not all of the tracks available in TORCS with performance that exceeds DDPG using only images as input and approaches the performance of an ideal non-vision based kinematics model.

1 INTRODUCTION

Predicting the future is an important topic in machine learning and is believed to be an important part of how humans process and interact with the world, cf Clark (2013). Study of the brain shows that it is highly predictive of future events and outcomes. Despite these advances, there is still much work needed to bridge the worlds of predictive learning and control. Most predictive control approaches learn either a forward model or a backward model Lesort et al. (2018) however these next-step models suffer from compounding errors Sutton (1988). This paper introduces a predictive control architecture using one kind of off-policy predictive learning, called general value functions (GVFs) Sutton et al. (2011)White (2015)Modayil et al. (2012)Daniel Graves (2019)Schaul & Ring (2013), that learns to predict the relevant aspects of the environment, decided by an expert, from raw sensor data such as pixel data captured from a camera. GVFs answer the predictive question, "if I follow policy τ, how much total cumulant will I see in the future?" The value of the GVF framework is not yet fully understood and realized despite the connections to neuroscience; but some early work has investigated its advantages for predictive representations and found that the representations are compact and general Schaul & Ring (2013). An objective of this research is to better understand the value that GVFs have to offer in real-world applications. Our work is based on the hypothesis that predictive representations are good for generalization Rafols et al. (2005) Schaul & Ring (2013). We are motivated by the belief that GVFs, like RL, could allow for behavior that is anticipative of future consequences rather than reactive to the current state.

General value functions (GVFs) are an understudied topic of interest in AI research fields and applications. There is a considerable focus on understanding how to learn these predictions but limited efforts on understanding how to use them in real applications. This is unfortunate, as to-date, research into applications of GVFs suggest they have potential in real world robotics and its applications Günther et al. (2016)Pilarski et al. (2011)Pilarski et al. (2013)Sutton et al. (2011)White (2015)Modayil et al. (2012). However, several elements have been missing to apply these predictions to a larger scale problem such as autonomous driving: (1) how to characterize the behavior policy to achieve off-policy learning when it is unknown, (2) what predictions are useful, and (3) how to use those predictions to control the vehicle. Our objective is two-fold: (1) introduce a novel architecture combining elements of predictive learning, adversarial learning and reinforcement learning, and (2) demonstrate how this architecture can be used to steer a vehicle in a racing simulator.

1.1 Related Works

Steering a vehicle is a challenging problem where the bicycle model is the classical approach Paden et al. (2016). However, the bicycle model requires knowing the angle of the vehicle with respect to the road direction in order to compute the desired steering angle. Steering directly from images has been a long desired goal in autonomous driving where approaches like Salvucci & Gray (2004) advocate for a two point model which inspired the multi-point predictive representation proposed in this paper.

In comparison, learning to regress an image directly to the steering angle in an end-to-end manner has been a recent hot topic Bojarski et al. (2016)Garimella et al. (2017)Chen & Huang (2017)Sallab et al. (2017). However, a serious challenge is ensuring robustness of the controller when learning end-to-end Bojarski et al. (2016). In particular, the agent is not typically trained on recovery mode scenarios and so there are generalization and data coverage issues; for this reason, authors Bojarski et al. (2016) introduced augmented images in training by artificially shifting and rotating them to help the network learn to recover with some limited success.

The approach in Chen et al. (2015) learns to predict the current road angle directly from images and then uses a classical steering controller to control the vehicle. The proposed approach is similar except we predict *future* road angles and lane centeredness at different temporal horizons which is then passed to a controller module to choose steering angles. Policy gradient with the predictive state representation is the approach used in this paper but this can also be replaced with other controllers. This architecture allows for a degree of interpretability in the controller that is not easily achieved with end-to-end approaches Bojarski et al. (2016) despite work on understanding and improving its robustness.

2 Predictive Learning

We consider an environment described by a set of states S, a set of actions A, and Markov transition dynamics with probability $P(s'|s,a)$ of transitioning to next state s' after taking action a from state s. This setting is nearly identical to a Markov Decision Process (MDP) where the only difference is the absence of a reward signal to maximize. The goal is to learn an estimator that predicts the return G_t of a cumulant c_t defined by

$$G_t \equiv \sum_{k=0}^{\infty} (\prod_{j=0}^{k} \gamma_{t+j+1}) c_{t+k+1} \tag{1}$$

where c_t is a cumulant signal to be predicted, and $0 \leq \gamma_t < 1$ is the continuation function. The general value function is defined as

$$V^{\tau}(s) = \mathbb{E}_{\tau}[G_t | s_t = s, a_t = a, a_{t+1:T-1} \sim \tau, T \sim \gamma] \tag{2}$$

where $\tau(a|s)$, $\gamma(s,a,s')$, and $c(s,a,s')$ are the policy, continuation and cumulant functions, respectively, that make up the predictive question Sutton et al. (2011) where $V^{\tau}(s)$ represents the total discounted cumulant starting from state s and acting under policy τ. Unfortuantely, there are currently no algorithms to learn the predictive question through interaction with the environment; thus, τ, γ, and c are typically defined by an expert. Cumulants are commonly scaled by a factor of $1-\gamma$ when γ is a constant in non-episodic predictions.

A GVF can be approximated with a function approximator, such as a neural network, parameterized by θ to predict equation 1. The agent usually collects experience under a different behavior policy

$\mu(a|s)$ where off-policy policy evaluation methods are needed to learn the GVF. The parameters θ are optimized with gradient descent minimizing the following loss function

$$L(\theta) = \mathbb{E}_{s \sim d_\mu, a \sim \mu}[\rho \delta^2] = \mathbb{E}_{s \sim d_\mu, a \sim \tau}[\delta^2] \tag{3}$$

where $\delta = \mathbb{E}[y - \hat{v}^\tau(s;\theta)|s,a]$ 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 target policy distribution τ and behavior distribution μ. Note that only the behavior policy distribution is corrected rather than the state distribution d_μ. The target y is produced by bootstrapping a prediction Sutton (1988) of the value of the next state following target policy τ given by

$$y = \mathbb{E}_{s' \sim P}[c + \gamma \hat{v}^\tau(s';\theta)|s,a] \tag{4}$$

where y is a bootstrap prediction using recent parameters θ that are assumed constant in the gradient computation. Some approaches use older parameters θ of the network to make a bootstrapped prediction to improve stability in the learning Mnih et al. (2013). However, this was not found to be necessary when learning GVFs since the target policy is fixed and the learning is simply off-policy policy evaluation. d_μ is the state distribution of the behavior policy μ and the time subscript on c and γ has been dropped to simplify notation.

The gradient of the loss function equation 3 is given by

$$\nabla_\theta L(\theta) = \mathbb{E}_{s \sim d_\mu, a \sim \mu}[\rho \delta \nabla_\theta \hat{v}^\tau(s;\theta)] \tag{5}$$

An alternative approach to using importance sampling ratios ρ is to apply importance resampling Schlegel et al. (2019). With importance resampling, a replay buffer D of size N is required and the gradient is multiplied with the average importance sampling ratio of the samples in the buffer $\bar{\rho} = \frac{\sum_{i=1}^N \rho_i}{N}$. The importance resampling gradient is given by

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

where the transitions in the replay buffer are sampled according to $D(a_i, s_i) = \frac{\rho(a_i|s_i)}{\sum_{j=1}^N \rho(a_j|s_j)}$ for transition with state s_i and action a_i in replay buffer D. This approach is proven to have lower variance than equation 5 with linear function approximation Schlegel et al. (2019). An efficient data structure for the replay buffer is the SumTree used in prioritized experience replay Schaul et al. (2016). This is a natural approach to learning predictions with deep reinforcement learning since sampling a mini-batch from the replay buffer helps to decorrelate sample updates in deep function approximation Mnih et al. (2013).

A behavior policy needs to be defined to adequately explore the environment when learning GVFs. This may be an evolving policy that is learned by RL, a random policy for exploring the environment, or a human driver collecting data safely. It is common, especially in the case of human drivers, for the behavior policy distribution $\mu(a|s)$ of the agent to be unknown. We propose an algorithm using the density ratio trick to learn the behavior policy distribution in an adversarial way. It is well suited for problems with low dimensional action spaces like autonomous driving.

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 a probability density function $\eta(a|s)$ for the distribution to compare to the behavior distribution $\mu(a|s)$ and class labels $y = +1$ and $y = -1$ that denote the class of the distribution that the state action pair was sampled from: $\mu(a|s)$ or $\eta(a|s)$ respectively. A discriminator $g(a,s)$ is learned that distinguishes state action pairs from these two distributions using the cross-entropy loss. The ratio of the densities can be computed using only the discriminator $g(a,s)$.

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

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

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

where $\eta(a|s)$ is a known distribution over action conditioned on state. The uniform distribution over the action is independent of state and has the advantage of being effective and easy to implement.

The algorithm for training a GVF off-policy with an unknown behavior distribution is given by

Algorithm 1 Off-policy GVF training algorithm with unknown $\mu(a|s)$

```
1: Initialize $\hat{v}^\tau$, $g(a,s)$, and replay memory $D$,
2: Observe initial state $s_0$
3: for t=0,T do
4:     Sample action $a_t$ from unknown $\mu(a|s)$
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:     Compute behavior density value $\hat{\mu}(a_t|s_t)$ according to equation 8
9:     Compute 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:    Sample random minibatch $A$ of transitions $(s_i, a_i, c_{i+1}, \gamma_{i+1}, s_{i+1})$ from $D$ according to
       probability $\frac{\rho_i}{\sum_{j=1}^{n} \rho_j}$
12:    Compute $y_i = c_{i+1} + \gamma_{i+1}\hat{v}^\tau(s_{i+1};\theta)$ for minibatch $A$
13:    Perform gradient descent step on $(y_i - \hat{v}^\tau(s_i;\theta))^2$ according to equation 6 for minibatch $A$
14:    Sample random minibatch $B$ of state action pairs $(s_i, a_i)$ from $D$ according to a uniform
       probability and assign label $y = +1$ to each pair
15:    Randomly select half the samples in the minibatch $B$ and temporarily replace the label with
       $y = -1$ and action with $a_t \sim \eta(a|s)$
16:    Update behavior discriminator $g(a,s)$ with modified minibatch $B$
```

3 Predictive Control

Let us consider an MDP with a predictive representation $\phi(s)$ mapping state s to predictions $\phi(s)$. The reward for the problem is denoted as r. The problem is to find a policy $\pi(a|\phi(s))$ that maximizes future return or accumulated discounted reward. We hypothesize that this approach should be easier to train than learning $\pi(a|s)$ directly for the following reasons:

- the target policy of the predictions τ is fixed making for faster learning
- the compact abstraction $\phi(s)$ allows for simple (possibly even linear) policy and action-value functions
- the cumulant signal c may only be available during training or is expensive to obtain

The last advantage is particularly important in autonomous driving where localization techniques often require a collection of expensive sensors and high definition map data that is not always available or easily scalable to a fleet of autonomous vehicles. In this way, one can train a neural network to map images captured by inexpensive cameras to predictions of lane centeredness and road angle captured by any number of highly accurate but expensive localization approaches with the hopes of generalizing features for lane control.

The agent learns to steer with deterministic policy gradient (DPG) Silver et al. (2014) using the predictions as the state of the agent. When linear policy function approximation is used, the controller learned is essentially equivalent to a prediction-based PID controller only where there is a deep mapping from images captured by a camera to predictions of the future used to control the vehicle. Using predictions for PID control is not new; this approach can be used to tackle problems with high temporal delay between the error signal and the corrective actions. One can also add integral and derivative terms of the predictions to the state space representation of the agent.

Action-value $Q^\pi(\phi(s), a)$ and policy $\pi(\phi(s))$ networks are trained according to DPG Silver et al. (2014) where the action value approximates the expected discounted return, i.e. $Q^\pi(\phi(s), a) = \mathbb{E}_\pi[\sum_{i=0}^{\infty} \gamma^i r_{t+i+1}]$. In addition, a policy network $\pi(\phi(s))$ produces an action according to the current predictive state representation $\phi(s)$ that maximizes the expected discounted return. Because of the interesting connection between DPG and PID control when linear function approximation is used, the policy network is parameterized as

$$a_t = \pi(\phi(s_t)) = \psi^\intercal \phi(s_t) \tag{9}$$

where ψ is a matrix denoting parameters to be learned by DPG. They also represent the gain coefficients for a proportional controller which allows for interpretability of the learned parameters. The action-value network $Q^\pi(\phi(s), a)$ is given by a small neural network that maps the predictive state representations $\phi(s)$ and action a to an estimate of the action-value in the current state s if a is taken and the optimal policy π followed thereafter.

In autonomous driving, knowing the road curvature ahead can be informative for making decisions to maintain lane centeredness in a tight turn. In Figure 1, it is demonstrated how future off-policy predictions of lane centeredness can be used to predict the deviation (or error) between the true center of lane and the projected lane centeredness along the current direction of the vehicle. These predictions must be off-policy because if they were on-policy they would tell us no information to inform the agent how much adjustment is needed to make corrective actions to stay in the center of the lane.

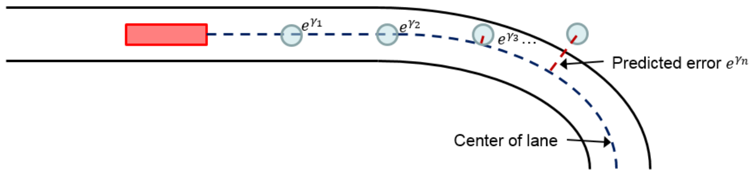
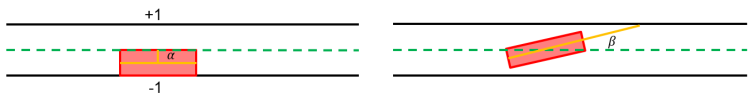


Figure 1: Future predictions capturing information about the shape of the road ahead



(a) Lane centeredness (b) Road angle

Figure 2: (a) the lane centeredness position α is the distance from the center of the lane to the center of the vehicle. (b) the road angle β is the angle between the direction of the vehicle and the direction of the road.

The lane centeredness α and road angle β are two kinds of predictions that are useful in steering the vehicle as depicted in Figure 2. We represent the road curvature as a set of predictions of future lane centeredness α and road angles β at different time horizons. The predictive state representation is given by the feature vector $\phi(s)$

$$\phi(s_t) = [V^\tau_\alpha(s_t; \gamma = \gamma_0), ...V^\tau_\alpha(s; \gamma = \gamma_{m_\alpha}), V^\tau_\beta(s; \gamma = \gamma_0), ...V^\tau_\beta(s; \gamma = \gamma_{m_\beta})] \tag{10}$$

where $V^\tau_\alpha(s_t; \gamma = \gamma_0)$ is a GVF prediction under target policy τ, cumulant α (lane centeredness) and continuation function γ_0 while $V^\tau_\beta(s_t; \gamma = \gamma_0)$ is a GVF prediction of cumulant β (road angle). There are m_α number of γ functions for predicting lane centeredness and m_β number of γ functions for predicting road angle at different temporal horizons. Because the predictions represent deviations from the desired lane centeredness and road angle, the policy network of DPG can be linear.

4 Experiments in TORCS

The predictive learning approach is applied to the challenging problem of learning to steer a vehicle in the TORCS Wymann et al. (2013) racing environment. A kinematic-based steering approach Paden et al. (2016) is used as a baseline for all the experiments. The reward in the TORCS environment is given by $r_t = v_t \cos \beta_t$ where v_t is the speed of the vehicle in km/h, β_t is the angle between the road direction and the vehicle direction. A simple scaling factor of 0.01 was applied to the reward in order to reduce the variance of the action-value. Notice that this reward doesn't force the agent to stay in the center of the lane; however, this strategy is likely a good idea to achieve high total reward on all the test tracks. The target speed of all the agents is 50 km/h where vehicle speed was controlled by a separate manually tuned PID controller.

The agents were trained on 85% of the 40 tracks available in TORCS. The rest of the tracks were used for testing (6 in total); all of the results in this section are on the testing tracks which were never presented to the agents during training. This was done to measure the generalization performance of the policies learned by the agents which can be a serious problem in RL, cf. Zhao et al. (2019)Farebrother et al. (2018). Tracks where the road was not level were excluded since they proved to be challenging likely because a non-zero action was required to keep the vehicle centered.

4.1 Test Results

In all the figures, blue corresponds to DDPG-Image with only image and speed as input, green corresponds to DDPG-ImageLowDim with images, current speed, α and β provided as input, orange corresponds to the new GVF-DPG approach with image, current speed and last two actions since the target policy of the prediction depends on the last action taken, and red corresponds to the classical front wheel steering model. A history of two images were provided to each method. A supervised method of predicting the current lane centeredness and road angle directly from the image was attempted with negative results: the controller learned was consistently unstable. Results were repeated over 5 runs. The total score achieved on the test tracks by the agents is plotted in Figure 3 over 1M training iterations.

It is clear that DDPG-ImageLowDim performs best of the learned methods on all test tracks. This low dimensional information provides ground truth to the agent which may not always be available in all locations; however, it makes a good baseline target for what we hope our proposed method could achieve through generalization. With DDPG-Image, it is clear that it does not learn to steer from images very well.

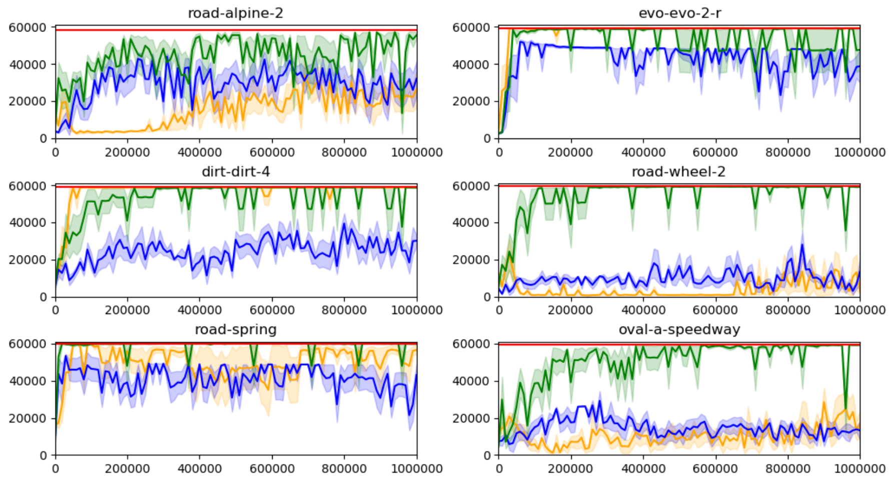


Figure 3: Test scores (accumulated reward) during training

It could be argued that DDPG-Image may improve with more iterations; however this is likely not the case. The reason is shown in Figure 4 where DDPG-Image consistently converges to a solution that oscillates between the extreme left and right steering actions very rapidly. This oscillation is so extreme that the agent is unable to achieve the target speed of 50 km/h; instead it travels at 40 km/h on average for all the test tracks. The performance gap also suggests that DDPG-ImageLowDim may be relying more on the low dimensional lane information rather than the image.

To highlight how uncomfortable the two DDPG agents drive compared to the GVF-DPG agent, Figure 4 shows the standard deviation of the change in action during 1M iterations of training. On most tracks, it is apparent that the GVF-DPG approach controls the vehicle more smoothly than the other learned methods.

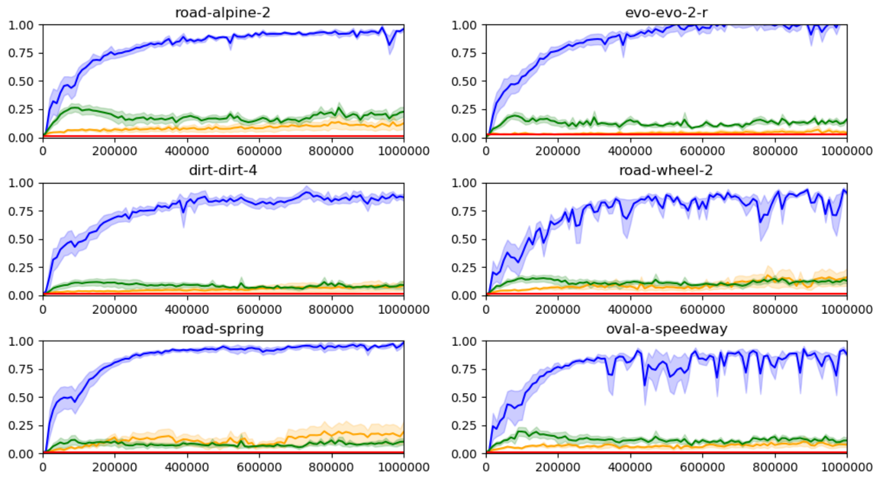


Figure 4: Standard deviation of the change in action during training

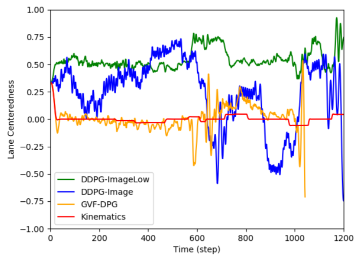
The performance of the individual test tracks is given in the following Figure 5. The GVF-DPG approach does not steer successfully on all test tracks: it fails immediately on wheel-2, part-way through on a-wheelway, and drives well on most of alpine-2. The DDPG-Image fails to complete dirt-4, wheel-2, spring, and a-speedway. Finally, DDPG-ImageLowDim successfully completes all the test tracks; however, the agent has a strong bias to the left side of the track. The GVF-DPG agent often follows the classical controller relatively well except on the tracks where the agent fails. This suggests that using a predictive representation of lane centeredness α and road angle β achieves closer performance to a classical controller than an end-to-end learned approach. However, more work is needed to improve the generalization abilities of the approach.

The learning curves for the predictors and the policy gradient agents is given in the following Figure 6. It is interesting that the learning curves of the action-value function estimator of the GVF-DPG agent is much smaller than the other agents and quite smooth. The reason is believed to be because the predictive state representation is constrained to values between $[-1, +1]$ acting as a sort of regularizer to the state representation of the agent. The learning curve of the DDPG-ImageLowDim however eventually approaches the low error of the GVF-DPG agent. The predictors converge relatively quickly as shown in Figure 6(b). The behavior estimator in Figure 6(c) stabilizes relatively quickly during learning as well; it is postulated that the error does not decrease further since the behavior policy is changing slowly over time and the behavior estimator must track this change.

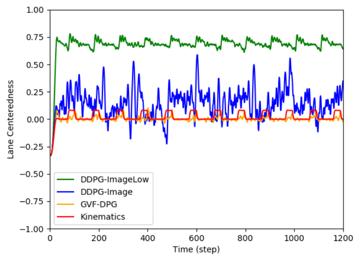
5 Conclusions

A method of learning a predictive representation off-policy is presented where the behavior policy distribution is estimated via an adversarial method employing the density ratio trick. It is demonstrated that deep off-policy predictions can be learned with a deep behavior policy estimation to predict future lane centeredness and road angles from images. The predictive representation is learned with linear deterministic policy gradient. All of these components are combined together in a framework called GVF-DPG and learned simultaneously on the challenging problem of steering a vehicle in TORCS from only images. The results show that the GVF-DPG is able to steer smoothly with less change in action and achieve better performance than DDPG from only images and similar performance to the kinematics model in several but not all of the test tracks. This work is also a demonstration that we can learn off-policy predictions, characterize the behavior policy and learn the controller all at the same time despite the challenges of the behavior policy evolving with the agent and the predictive state representation changing over time.

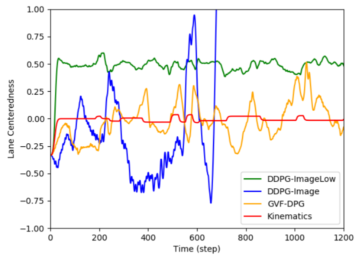
Our work demonstrates that a learned prediction-based vision-only steering controller could potentially be viable with more work on improving the generalizability of the off-policy predictions.



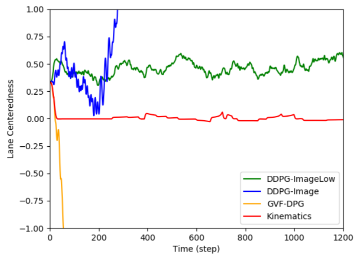
(a) Alpine-2



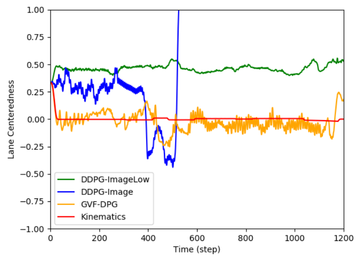
(b) Evo-2-r



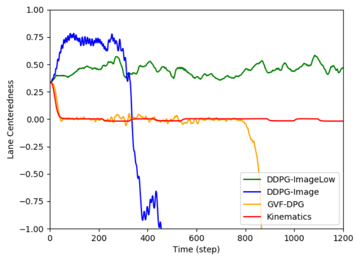
(c) Dirt-4



(d) Wheel-2

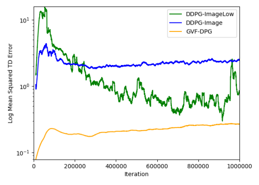


(e) Spring

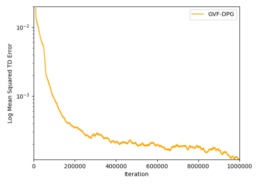


(f) A-Speedway

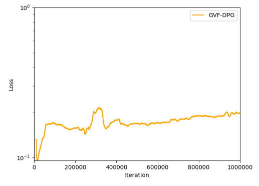
Figure 5: The lane centeredness position on the (a) alpine-2, (b) evo-2-r, (c) dirt-4, (d) wheel-2, (e) spring, and (f) a-speedway tracks in TORCS.



(a) Action-Value



(b) Predictors



(c) Behavior

Figure 6: Log-loss learning curves for the (a) Q-values of the DPG agents, (b) mean squared TD (temporal difference) errors of the GVF predictors, and (c) MSE of the behavior model estimator

This work supports the predictive state representation hypothesis in Rafols et al. (2005) that deep predictions can improve the generalization of RL to new road environments when using only images as input. For future work, we hope to study how to learn the question for the predictive state representation: τ, γ, and c. Moreover, because the behavior policy is unknown and estimated, our results suggest that collecting real-world human driving to train predictions off-policy without the need for a simulator could be a viable approach to steering a vehicle from images. This is potentially advantageous since the human driver can explore the road safely.

REFERENCES

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016. URL http://arxiv.org/abs/1604.07316.

C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2722–2730, Dec 2015. doi: 10.1109/ICCV.2015.312.

Z. Chen and X. Huang. End-to-end learning for lane keeping of self-driving cars. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1856–1860, June 2017. doi: 10.1109/IVS.2017.7995975.

A. Clark. Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Science*, 36(3):181–204, 2013. doi: 10.1017/S0140525X12000477.

Sean Scheideman Daniel Graves, Kasra Rezaee. Perception as prediction using general value functions in autonomous driving applications. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS 2019, 2019.

Jesse Farebrother, Marlos C. Machado, and Michael Bowling. Generalization and regularization in DQN. *CoRR*, abs/1810.00123, 2018. URL http://arxiv.org/abs/1810.00123.

G. Garimella, J. Funke, C. Wang, and M. Kobilarov. Neural network modeling for steering control of an autonomous vehicle. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2609–2615, Sep. 2017. doi: 10.1109/IROS.2017.8206084.

Johannes Günther, Patrick M. Pilarski, Gerhard Helfrich, Hao Shen, and Klaus Diepold. Intelligent laser welding through representation, prediction, and control learning: An architecture with deep neural networks and reinforcement learning. *Mechatronics*, 34:1 – 11, 2016. ISSN 0957-4158. doi: https://doi.org/10.1016/j.mechatronics.2015.09.004. URL http://www.sciencedirect.com/science/article/pii/S0957415815001555. System-Integrated Intelligence: New Challenges for Product and Production Engineering.

Timothée Lesort, Natalia Díaz Rodríguez, Jean-François Goudou, and David Filliat. State representation learning for control: An overview. *CoRR*, abs/1802.04181, 2018. URL http://arxiv.org/abs/1802.04181.

Timothy Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *ICLR*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

Joseph Modayil, Adam White, and Richard S. Sutton. Multi-timescale nexting in a reinforcement learning robot. In Tom Ziemke, Christian Balkenius, and John Hallam (eds.), *From Animals to Animats 12*, pp. 299–309, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-33093-3.

Brian Paden, Michal Cáp, Sze Zheng Yong, Dmitry S. Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *CoRR*, abs/1604.07446, 2016. URL http://arxiv.org/abs/1604.07446.

P. M. Pilarski, M. R. Dawson, T. Degris, F. Fahimi, J. P. Carey, and R. S. Sutton. Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. In *2011 IEEE International Conference on Rehabilitation Robotics*, pp. 1–7, June 2011. doi: 10.1109/ICORR.2011.5975338.

P. M. Pilarski, M. R. Dawson, T. Degris, J. P. Carey, K. M. Chan, J. S. Hebert, and R. S. Sutton. Adaptive artificial limbs: a real-time approach to prediction and anticipation. *IEEE Robotics Automation Magazine*, 20(1):53–64, March 2013. ISSN 1070-9932. doi: 10.1109/MRA.2012.2229948.

Eddie J. Rafols, Mark B. Ring, Richard S. Sutton, and Brian Tanner. Using predictive representations to improve generalization in reinforcement learning. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pp. 835–840, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc. URL `http://dl.acm.org/citation.cfm?id=1642293.1642427`.

Ahmad Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017:70–76, 01 2017. doi: 10.2352/ISSN.2470-1173.2017.19.AVM-023.

Dario D Salvucci and Rob Gray. A two-point visual control model of steering. *Perception*, 33(10): 1233–1248, 2004. doi: 10.1068/p5343. URL `https://doi.org/10.1068/p5343`. PMID: 15693668.

Tom Schaul and Mark Ring. Better generalization with forecasts. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pp. 1656–1662. AAAI Press, 2013. ISBN 978-1-57735-633-2. URL `http://dl.acm.org/citation.cfm?id=2540128.2540366`.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *ICLR*, Puerto Rico, 2016.

Matthew Schlegel, Wesley Chung, Daniel Graves Jian Qian, and Martha White. Importance resampling off-policy prediction. *CoRR*, abs/1906.04328, 2019. URL `http://arxiv.org/abs/1906.04328`.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pp. I–387–I–395, 2014.

Richard Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '11, pp. 761–768, 2011.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3 (1):9–44, Aug 1988. ISSN 1573-0565. doi: 10.1023/A:1022633531479. URL `https://doi.org/10.1023/A:1022633531479`.

George Uhlenbeck and Leonard Ornstein. On the theory of the brownian motion. *Physical review*, 36:823–841, 1930.

Adam White. *Developing a predictive approach to knowledge*. PhD thesis, University of Alberta, 2015.

Bernhard Wymann, Eric Espie, Christophe Guionneau, Christos Dimitrakakis, Remi Coulom, and Andrew Sumner. Torcs, the open racing car simulator, v1.3.5, 2013. URL `http://www.torcs.org/`.

Chenyang Zhao, Olivier Sigaud, Freek Stulp, and Timothy M. Hospedales. Investigating generalisation in continuous deep reinforcement learning. *CoRR*, abs/1902.07015, 2019.

6 Appendix

6.1 GVF-DPG Training Setup

The predictive learning approach presented in this paper is called GVF-DPG (general value function deterministic policy gradient). Exploration followed the same approach as Lillicrap et al. (2016) where an Ornstein Uhlenbeck process Uhlenbeck & Ornstein (1930) is used to explore the track where the parameters of the process ($\theta = 0.01$, $\sigma = 0.01$) were tuned to provide a gradual wandering behavior on the track without excessive oscillations in the action. The reason is to improve the learning of the off-policy predictions for GVF-DPG since the behavior policy $\mu(a|s)$ is closer to the target policy $\tau(a|s)$ of the predictions.

The GVF-DPG approach learned 8 predictions: 4 predictions of lane centeredness α, and 4 predictions of road angle β. Each of the 4 predictions had different values of γ for different temporal horizons: 0.5, 0.9, 0.95, 0.97. This allowed the agent to make short-term, medium term and long term predictions of lane centeredness α and road angle β. The GVF predictors all share the same deep convolutional neural network where the convolutional layers are identical to the architecture in Bojarski et al. (2016) followed by three fully connected layers of 512, 384 and 8 outputs, respectively. The behavior estimator $\mu(a|s)$ also uses a very similar neural network with identical convolutional layers followed by three fully connected layers of 512, 256, and 1 output, respectively. The GVF predictors and behavior estimator are both given the current image, previous image, current speed and the last two actions taken.

The policy network of the DPG agent is linear with respect to the predictions. The action-value network is a small 4 layer network of 64x64x32x1 and outputs the change in the action from the previous action. The predictions are supplied to the agent and the last action is needed to compute the next action. All networks use rectified linear unit activations except for the last layers which are linear. The learning rates for the linear policy network, action-value network, predictors and behavior policy network were $1e^{-4}$, $1e^{-8}$, $1e^{-4}$, and $1e^{-4}$ respectively. It was noted that the action-value learning rate needed to be small in order to faciliate two time-scale learning of the predictive representation and the action-values.

6.2 DDPG Training Setup

The GVF-DPG approach is compared to two DDPG (deep deterministic policy gradient) Lillicrap et al. (2016) baselines where the differences are given by only the information provided in the state. The first method is a vision-based approach where the image and current speed is provided to the agent; the only information available to the agent about the road is supplied via images. This proved challenging for the DDPG agent. Thus, a second agent was trained with the image and the lane centeredness α and road angle β that the GVF-DPG agent had access to during training. This is cheating in a sense because the agent must have the lane centeredness and road angle available at all times including at test time, whereas the GVF-DPG agent is hopefully learning features that are generalizable to unseen roads where this information may not be always available. However, it provides a good target for evaluating the GVF-DPG agent. An Ornstein Uhlenbeck process Uhlenbeck & Ornstein (1930) is used to explore the track ($\theta = 0.05$, $\sigma = 0.05$).

In previous works with DDPG, target networks were utilized for both the action-value and the policy networks. The target networks are copies of the action-value and policy networks that are updated more slowly in order make the bootstrapped prediction of the action-values more stable Lillicrap et al. (2016). However, in our experiments it was found that removing the target networks improved learning and so no target networks were deployed in any of the following experiments. Identical network architectures were used for the policy and action-value networks of the DDPG agents as was used for the behavior network for GVF-DPG. The low dimensional state information was feed through a separate branch of 12 neurons that were then merged with the fully connected layer of 512 neurons. This method of merging can present challenges due to mismatching statistical properties of the branches but that did not seem to present a significant challenge in learning. Future work would be to find better ways to bridge these two different pieces of information. The hyperbolic tangent activation was used for output layer of the policy network, the linear activation was used for the output layer of the action-value network and rectified linear activation was used everywhere else.

6.3 TORCS Environment

During training, a track is selected randomly according to a priority sampling method, since the tracks were not balanced, and the agent is allowed to interact with the environment and learn until termination. Termination occurs when either the agent leaves the lane or the maximum number of steps has been reached (1200 steps = 120 seconds). A priority sampling method was chosen so that tracks that were more difficult were selected more often. The probability of sampling a track i is given by

$$\frac{e^{-\frac{n_i}{\kappa}}}{\sum_{j=1}^{N} e^{-\frac{n_j}{\kappa}}} \tag{11}$$

where n_i is the number of steps that the agent was able to achieve in the last episode for that track and κ 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 tracks.

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 which is beneficial to both policy gradient methods and predictive learning.