Autonomous Navigation from Visual Demonstrations

Apurba Bose
Department of Electrical and Computer Science Engineering
University of California, San Diego

Sai Hanisha Kilari
Department of Mechanical and Aerospace Engineering
University of California, San Diego

***Abstract*—Autonomous Driving has received a lot of attention in recent years with advancement in Deep Learning. This has been further accelerated by incorporating Reinforcement Learning (RL) algorithms. This project aims to train an agent in the TORCs simulator environment using RL algorithms. Starting with naive algorithm using only available state space variables, Deep Deterministic Policy Gradients (DDPG) is trained. We further implement Behavior Cloning (BC) and our variation of Generative Adversarial Imitation Learning (GAIL). In GAIL, we train a multi-modal based deep RL algorithm including visual inputs and causal information encoding to train the agent. We also present the utilization of the causality of the latent code in an expert trajectory consisting of different sub-tasks and leverage this in the model.**

***Index Terms*—Imitation Learning, Latent codes**

I. Introduction

A key limitation of reinforcement learning (RL) is that it involves the optimization of a predefined reward function or reinforcement signal. Designing an appropriate reward function can be difficult in complex and less well-specified environments, e.g., for autonomous driving where there is a need to balance safety, comfort, and efficiency. Imitation learning methods have the potential to close this gap by learning how to perform tasks directly from expert demonstrations, and has succeeded in a wide range of problems. Among them, Generative Adversarial Imitation Learning (GAIL) [1], is a model-free imitation learning method that is highly effective and scales to relatively high dimensional environments. The training process of GAIL can be thought of as building a generative model, which is a stochastic policy. when coupled with a fixed simulation environment, produces similar behaviors to the expert demonstrations. Similarity is achieved by jointly training a discriminator to distinguish expert trajectories from ones produced by the learned policy, as in GANs.

In Imitation learning, example demonstrations are typically provided by human experts. These demonstrations can show significant variability. External latent factors of variation that are not explicitly captured by the simulation environment can also significantly affect the observed behavior. Info-GAIL develops an imitation learning framework that is able to automatically discover and disentangle the latent factors of variation underlying expert demonstrations.

In a dynamic environment, i.e., sequences of state-actions pairs in a Markov Decision Process, the model can accurately reproduce expert behavior, and also empirically learn a latent space of the observations that is semantically meaningful. This approach suits best for learning each mode, but is not efficient in transition from one mode to the other. This effects the learning process of end to end trajectories. Its extension Directed Info Gail uses a causal latent code model where latent code at time t is dependant on time $t-1$.

Our approach is an extension of Info-GAIL, where the objective is augmented with a mutual information term between the latent variables and the observed state-action pairs. we demonstrate an application in autonomous driving, where we learn to imitate complex driving behaviors while recovering semantically meaningful structure, without any supervision beyond the expert trajectories.

II. Background

A. Imitation Learning

Typically, there are two approaches to imitation learning: 1) Behavior Cloning (BC): Learns a policy through supervised learning over the state-action pairs from the expert trajectories. 2) Apprenticeship Learning (AL): Assumes the expert policy is optimal under some unknown reward and learns a policy by recovering the reward and solving the corresponding planning problem. BC tends to have poor generalization properties due to compounding errors and covariate shift. AL, on the other hand, is typically expensive to run and yet fails to directly yield actions. In search for an imitation learning algorithm that both bypasses an intermediate IRL step and is suitable for large environments, GAIL had been introduced.

B. Generative Adversarial Imitation Learning

Generative Adversarial Imitation Learning is a recent AL method inspired by Generative Adversarial Networks. In the GAIL framework, the optimum is achieved when the distance between state-action distribution and the expert's distribution, is minimized as measured by Jensen-Shannon divergence. GAIL doesn't include the variability of expert trajectories, as in different ways(left and Right) to pass a vehicle. If it has to be considered, they need to be solved as different problems. It's extension, InfoGAIL can disentangle different behaviour modes and can be solved as a single problem.

C. InfoGAIL

Using visual inputs as the only external perceptual information InfoGAIL can discover and disentangle salient latent factors of variation underlying expert demonstrations without supervision, and learn policies that produce trajectories which correspond to these latent factors [2].

If the expert performs sub-optimally, the learned policy will also be sub-optimal. This motivates the introduction of **reward augmentation**, a general framework to incorporate prior knowledge in imitation learning by providing additional incentives to the agent without interfering with the learning process. We achieve this by specifying a surrogate state-based reward $\eta(\pi_\theta) = E_{(s\sim\pi_\theta)}[r(s)]$ that reflects our bias over the desired agent's behavior. In our case,by providing the agent with a penalty if it collides with other cars or drives off the road, the average rollout distance of the learned policy can be significantly improved.

To deal with high dimensional inputs such as images in our experiments with 110 x 200 x 3 pixels, and also to alleviate the problem of vanishing gradient and mode collapse in traditional GAN Objective, Wasserstein GAN technique has been used. Other variance reduction techniques, such as baselines and replay buffers have been used.

Implementing InfoGAIL for a complex task that has multiple modes or hierarchical structure can be challenging. It's extension Directed InfoGAIL talks about the interaction between sub-tasks from their resulting state action trajectory sequences using a directed graphical model.

D. Directed Info-GAIL

Directed Info-GAIL deals with intra-trajectory variations and sub-tasks (variations) within a demonstration [3].It can learn sub-task policies from unsegmented demonstrations automatically by maximizing the directed information flow between sub-task latent variables and their generated trajectories. In InfoGAIL, we trained different networks for the **pass** and **turn** cases with two dimensional one hot encoding of the latent variable in each. We extend it here to learn a encoder-decoder based causal model, in which the posterior network is an encoded model with $P(c|s, a, c_1)$

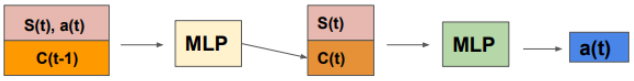

Fig. 1. Directed Info-GAIL. The encoder MLP uses previous c_{t-1}, a_t and S_t to produce current latent variable c_t The decoder reconstructs the action using c_t

III. METHODOLOGY

A. Visual Inputs via Transfer Learning

Due to the high dimensional nature of the visual inputs, the policy will have to simultaneously learn how to identify meaningful visual features and leverage them to achieve the desired behavior using only a small number of expert trajectories. we use a Deep Residual Network ResNet50(activation layer-40) pre-trained on the ImageNet classification task to obtain the visual features used as inputs for the policy network demonstrations.

B. Network Architecture

The auxiliary information(10x1) can be passed along with the raw visual inputs., such as velocity at time t (3x1), actions at time $t-1$ and $t-2$ (3x1,3x1), damage of the car(1x1).For the policy network, input visual features are passed through two convolutional layers (3x3x256, LeakyReLU), and then combined with the auxiliary information vector and the latent code c(2x1). We parameterize the baseline as a network with the same architecture except for the final layer, which is just a scalar output that indicates the expected accumulated future rewards.

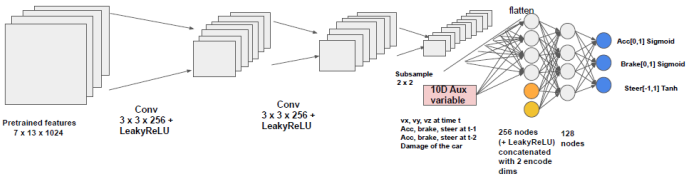

Fig. 2. Generator Network

The discriminator D accepts three elements as input: the input image, the auxiliary information, and the current action. The output is a score for the WGAN training objective, which is supposed to be higher for expert state-action pairs, and lower for generated ones.

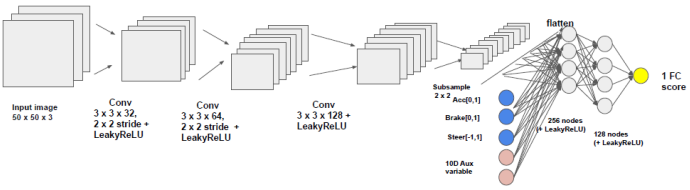

Fig. 3. Descriminator Network

The posterior approximation network Q adopts the same architecture as the discriminator, except that the output is a softmax over the discrete latent variables or a factored Gaussian over continuous latent variables. In the causal model we train this network to be an encoding of the previous latent variable, action and state to the present latent variable which then reproduces the action.

Algorithm 1 Causal Info-GAIL Model

Data: Input: Expert Trajectories Π_E, initialize generator-posterior (θ_0), discriminator(ω_0), Replay buffer - null
Output: Learn the policy Π_θ

for *i in epochs* **do**
  for *i in rollouts* **do**
    Sample batch of trajectories in each rollout Update the replay buffer B $\leftarrow$ samples
  end
  Sample same batch size of expert trajetories for the discriminator network
  Update (ω_i) to (ω_{i+1}) with gradients

$$\nabla_{\omega_i} = \mathbb{E}_{\chi_i}\left[\nabla_{\omega_i} D_{\omega_i}(s,a)\right] + \mathbb{E}_{\chi_E}\left[\nabla_{\omega_i}(1 - D_{\omega_i}(s,a))\right]$$

  Update policy (θ_i) to (θ_{i+1}) using TRPO update rule

$$\nabla_{\theta_i} = \mathbb{E}_{\chi_i}\left[D_{\omega_{i+1}}(s,a)\right] - \lambda_1 L_1(\pi_{\theta_i}, \nabla(Q_i(c_{i-1})) - \lambda_2 H(\pi_{\theta_i})$$

end

IV. Results

A. Problem Setup

TORCs [4] is a simulation platform used in research for Machine Learning, with the flexibility to play games and collect expert trajectories smoothly. We train the car in different tracks using Deep Deterministic Policy gradient, Behavior Cloning, GAIL and Directed Info-GAIL. TORCs is an effective tool to test the trained model in a simulated environment. We use 10 expert trajectories in each problem setting, with 60 frames in each. Some of the sample trajectories generated are shown below.

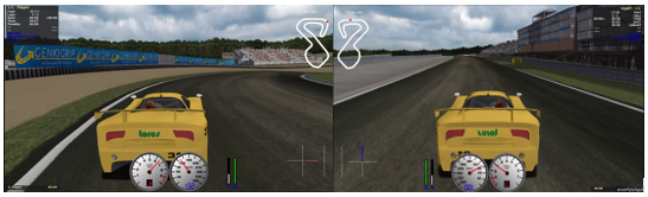

Fig. 4. Expert trajectories collected for turn in the Brondehach track

Here we consider two subsets of human driving behaviors: turn, where the expert takes a turn using either the inside lane$[0,1]$ or the outside lane $[1,0]$; and pass, where the expert passes another vehicle from either the left$[0,1]$ or the right$[1,0]$. c is a discrete latent code, which is a one-hot encoded vector with two possible states.

We use SnakeOil Python library to interface our model with the TORCs environment. We have access to the following 29 state variables- angle, curl lap time, Damage, distance from the start, distance raced, focus, fuel, gear, last lap time, opponents, race position, engine RPM, speed - X,Y,Z, track, track position, wheel spin velocity. Till DDPG, we use this setting, but from BC we add camera image extracted visual features (ResNet) for smaller compressed representation of the images as an input to the RL network.

BC, pass and Causal Info-GAIL are trained on the Chenyi-Street track while the turns are trained on Brondehach and modified Chenyi-Brondehach track. For the end to end causal model training, we use the Dirt track. Reward is assigned a negative -200 value for collision, damage, out of tracks and car stalling. The aim is to maximise the reward while minimizing the policy divergence.

B. Behavior Cloning

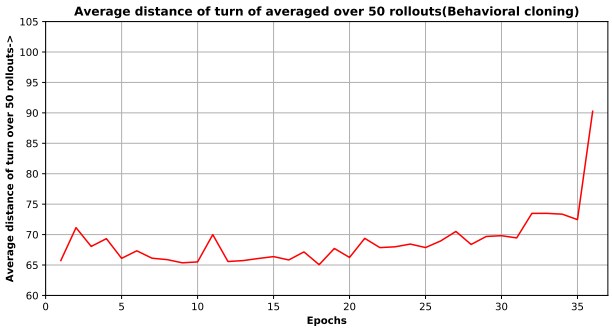

Fig. 5. Trained with 30 rollouts per epoch, Batch Size = 50

C. GAIL with image state inputs and Latent Codes

Graphs for the pass case:

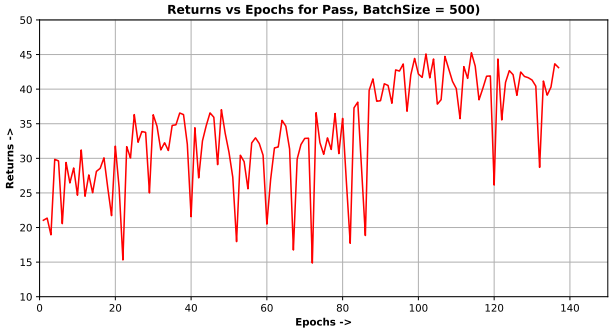

Fig. 6. Discounted returns of rollout episodes in each epoch. Return is given by $output_d * 0.1 + log(output_p) * encode * 0.2$, Batch Size = 500

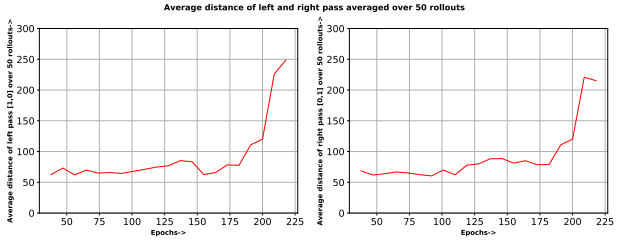

Fig. 7. Average distance of left and right pass taken over 50 rollouts per epoch, Batch Size = 100

Graphs for the turn case: The following results have been trained for 12 epochs only since the GPU ran out of memory for further rollouts and updates.

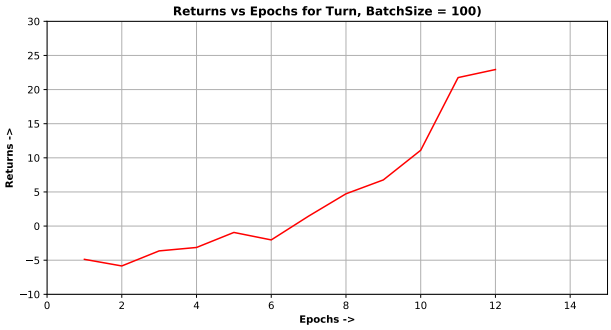

Fig. 8. Discounted returns of rollout episodes in each epoch. Return is given by $output_d * 0.1 + log(output_p) * encode * 0.2$, Batch Size = 100

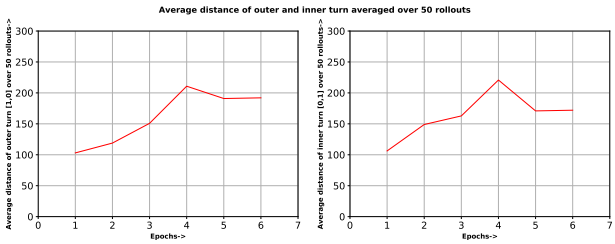

Fig. 9. Average distance of inside and outside turn taken over 50 rollouts per epoch, Batch Size = 100

D. Causal Info-GAIL Model

We collect mixed trajectories in this case, and train the encode decoder model by combining the posterior and generator network. It has been trained for 25 epochs only since the GPU ran out of memory.

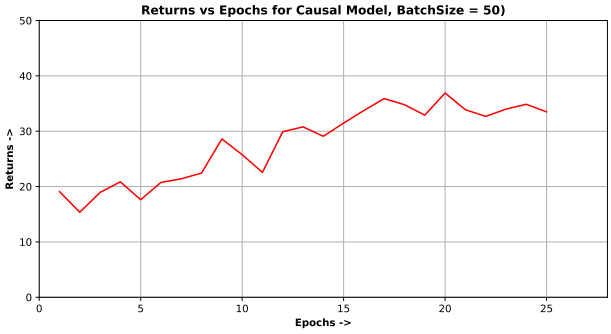

Fig. 10. Discounted returns of rollout episodes in each epoch. Return is given by $output_d$

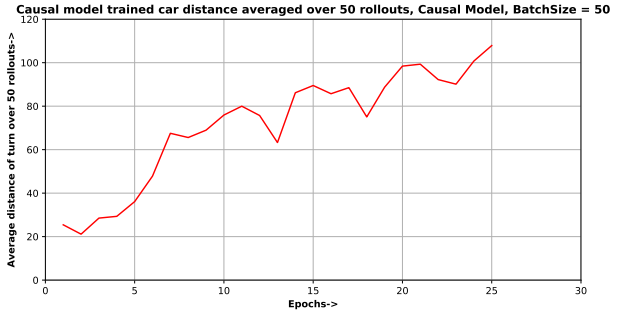

Fig. 11. Average distance traversed by the car over 50 rollouts. Batch Size = 50

V. Conclusion

In this project we explore different Deep Reinforcement Learning techniques to navigate the car in the TORCs environment. The model is trained on Nvidia GeForce GTX 1080 GPU. Our results show that embedding the latent code information in the posterior network while training showed significant improvement in performance. Further we present a method to learn end to end trajectories by claiming that the trajectories can be divided into subtasks and we can exploit the causality of the latent code by training an encoder decoder model. We saw good performance after training the model, however due to resource limitations we could train for only 25 epochs with 50 rollouts in each epoch. The graphs show that the rollout distance in BC reached 90, with the car getting confused in the trajectories and colliding sooner. Training with latent information and further extending it to a causal model raises the average distance traversed to 200.

VI. Limitations and Future Work

We would like to further explore on the following points

- One limitation in the pass case noticed was that the car slowed down when near to other cars in the race arena. This possibly required further manipulation to the reward function in the environment.
- DDPG without the use of visual features was not giving a robust performance in our case. Use of pre-Trained visual features and a more robust reward function encompassing the velocity and track position would have given better results.

References

[1] Ho, J., and Ermon, S. Generative adversarial imitation learning. *CoRR abs/1606.03476* (2016).

[2] Li, Y., Song, J., and Ermon, S. Infogail: Interpretable imitation learning from visual demonstrations. *arXiv preprint arXiv:1703.08840* (2017).

[3] Sharma, A., Sharma, M., Rhinehart, N., and Kitani, K. M. Directed-info GAIL: learning hierarchical policies from unsegmented demonstrations using directed information. *CoRR abs/1810.01266* (2018).

[4] Wymann, B., Espi, E., Guionneau, C., Dimitrakakis, C., Coulom, R., and Sumner, A. Torcs, the open racing car simulator.