Sample Efficient Imitation Learning via Transfer

Michael Kelly
Department of Computer Science
Stanford University
mkelly2@stanford.ed

Abstract

While interactive imitation learning methods such as DAGGER address the compounding error issues that plague simpler approaches to imitation learning such as behavioral cloning, they require the continual presence of an expert throughout the training of the novice policy. This is a particular limitation in domains where access to the expert is expensive, such as in human-in-the-loop imitation learning. Prior work on this problem has focused on minimizing the total number of expert queries required by DAGGER-like algorithms. This work focuses instead on minimizing the total number of trajectories required by DAGGER to achieve good performance. We explore using reinforcement learning in imperfect simulators of the target environment to learn good initializations for the novice policy. When representing policies as deep neural networks, the straightforward application of this method is found to suffer from catastrophic forgetting. We consider an alternative method that interleaves steps of behavioral cloning and reinforcement learning, showing improvement over the naive approach.

1. Introduction

A. Background and Motivation

Action execution is generally far slower and more expensive in real-world domains than it is in simulation. This makes it difficult for deep reinforcement learning algorithms to acquire the large amounts of data they need to perform well. Imitation learning is a compelling alternative in these types of settings, as the generally richer training signal imitation learning methods provide to the learner allow for significantly lower sample complexities (Sun et al. 2017).

Unlike in reinforcement learning, where the agent learns via a reward signal received during interaction with its environment, in imitation learning the agent or "novice" learns by observing demonstrations - sequences of states and actions - provided by an expert teacher. The simplest approach to imitation learning is behavioral cloning, in which the novice is trained via supervised learning on a dataset of state-action tuples extracted from the provided expert demonstrations (Ross, Gordon, and Bagnell 2011).

Behavioral cloning performs poorly both in theory and practice, however, due to data mismatch and compounding error issues. One method that has been proposed to address these issues is DAGGER, an iterative algorithm which generates training trajectories with a mixed policy constructed from the expert policy and the current instantiation of the novice policy. At each time step in a training trajectory, DAGGER queries both the novice and expert for their actions, appends to the training dataset a tuple consisting of the current state and expert action, and then executes either the expert action with some probability β or the novice action with probability $1-\beta$. The parameter β is generally decayed over training epochs. DAGGER can be shown to produce a stationary deterministic policy with good guarantees on performance under its induced state distribution (Ross, Gordon, and Bagnell 2011).

While interactive imitation learning methods such as DAGGER address the compounding error and data mismatch issues inherent to behavioral cloning, they place a greater burden on the expert, as they require it to be present throughout the training process (Ross, Gordon, and Bagnell 2011). Access to an expert is frequently expensive, however, particularly when working on complex, real world tasks - such as autonomous driving (Zhang and Cho 2017), robotic surgery (Laskey et al. 2016), and autonomous helicopter flight (Abbeel, Coates, and Ng 2010) - where we would like to employ a human as our reference policy.

Previous work has sought to minimize the frequency with which the expert policy is queried during training (Kim and Pineau 2013; Laskey et al. 2016; Zhang and Cho 2017). However, the overall number of training trajectories required to reach good performance is an equally if not more important limiting factor in human-in-the-loop imitation learning. Since prior methods still require the expert to be "on call" at all times during the training process, they do not address this highly relevant minimization.

A.1 Contributions

It is often the case that we have access to an imperfect simulator of our target environment (Cutler, Walsh, and How 2015). In this work, we examine how we might lever such access to reduce the number of trajectories required by DAGGER to reach good performance. Our proposed method is to train a policy in simulation using standard deep reinforcement learning techniques and then to use this trained policy as an initialization for the novice policy in DAGGER.

In order to allow our policies to handle complex tasks in continuous, high-dimensional state and action spaces, we choose to represent them as deep neural networks. However, naively applying our method in this setting results in poor performance, offering no discernible improvements in sample complexity. We examine various potential causes for these results, concluding that catastrophic forgetting is largely responsible. We then outline a potential method for mitigating this issue using multi-task learning.

2. Methods

The DAGGER algorithm requires as an input some initial novice policy. Usually, the novice is initialized randomly, or via simple behavioral cloning. Our method proposes to use deep reinforcement methods to train a good initialization for the policy in simulation. Specifically, this policy will be represented using a deep neural network mapping observations of the environment into actions.

This method assumes access to a simulator with observation and action spaces identical to those of the target environment, so that the policy trained in simulation can be directly ported over to the target environment. The observation and transition models of the simulator and target environment may differ in nontrivial ways, however.

The rationale behind this method is that we are only requiring DAGGER to fine-tune the novice policy rather than to train it from scratch. Provided that the simulator captures salient features of the target environment, some of the knowledge gained via training in simulation will likely be useful in the target domain. For example, in the domain tested in this work, the agent has access to noisy lidar scans of its environment rather than its true state. Learning to parse these lidar measurements is a potentially transferable skill that would be highly useful in the target domain. Acquiring such knowledge in simulation could shift much of the learning burden onto the reinforcement learning algorithm and away from DAGGER, allowing DAGGER to achieve good performance in fewer iterations and thereby lessening the supervisory burden placed on the expert.

3. Experiments

A. Environment

A variant of the Dubins Car Lidar environment used in (Menda, Driggs-Campbell, and Kochenderfer 2017) was constructed to test the methods developed in this work. This "bottleneck" environment, illustrated in Figure 1, consists of a pair of rooms connected by a narrow tunnel. The agent is a small Dubins car with constant velocity. It starts with a random pose within the first room and must find its way out to the exit in the second room. An episode terminates if the novice collides with a wall or successfully manages to exit.

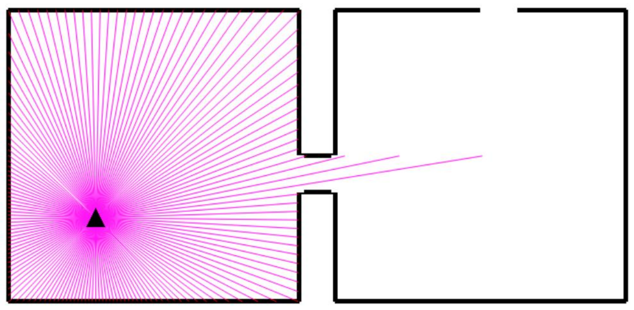

Figure 1: The bottleneck environment, rendered without sensor noise.

The agent has access not to its true state but rather to noisy "lidar" measurements of its surroundings: range measurements to the nearest obstacle along 100 equally spaced rays propagating out from the car. The action returned by the policy is an angular velocity, which is clipped at 1.0 rad/s.

Zero-mean Gaussian noise is injected into the positional component of the state transitions. Additionally, there is a persistent drift force acting on the the agent throughout the environment causing an additional displacement in the positive y-direction (upwards in Figure 1).

The simulator environments differ from the target environment in two ways. The first is that they have a noiseless observation model. The various simulator environments tested also differ from each other and from the target environment in that they only capture varying fractions of the full strength of the drift force used in the target environment.

B. Policies

All policies - both the expert and the novices - are represented as deep neural networks. The input to the networks is the 100-dimensional lidar scan of the environment. The output is a single real number representing the desired angular velocity for the agent.

All policy networks have three densely-connected hidden layers of size 64 units, 32 units, and 32 units. We used hyperbolic tangent activations for all layers, including the output unit. All of the network weights are initialized using Xavier initialization, and all of the biases were initialized to zero.

During the supervised learning phase of imitation learning, we employed a learning rate of 0.001 and L1-regularization with a regularization parameter of 0.001. We used a minibatch size of 8.

All policies were initially trained with rllab's implementation of trust-region policy optimization (Duan et al. 2016; Schulman et al. 2015), using a discount rate of 0.995 and a learning rate of 0.01. Rollouts were capped at 750 timesteps. We also defined a reward function for the environment in order to apply TRPO: the agent receives a penalty of -10,000 for colliding with a wall, a reward of 100,000 for successfully exiting the second room, and a one-time reward of 1,000 both for entering the tunnel and for reaching the second room. We ran TRPO for 100 epochs in all cases, although it usually found good policies much sooner.

C. *Experimental Setup*[1]

We evaluated the performance of DAGGER in the bottleneck environment in six cases. The baseline case evaluated the performance of standard DAGGER, using a randomly initialized novice policy. The other five cases examined the performance of DAGGER when the novice policy had undergone RL pretraining in simulation. The five pretrained novice policies were trained in different simulators with varying degrees of fidelity. Performance was measured by the probability that the agent could successfully reach the exit within 750 time steps without colliding with a wall.

In all experiments, the dimensions of the rooms were set to be 70m by 70m, while the tunnel was 15m long and 10m wide. The exit in the second room was also made 10m wide. The agent's fixed velocity was set at 10 m/s, and time was discretized into 0.1 second steps. In all environments, we used transition noise drawn from a Gaussian with σ = 1m. The upwards drift in the target domain was set such that it resulted in an additional displacement in the positive y direction of 0.5m per time step if not corrected by the agent.

For each experiment, we performed 20 DAGGER runs. The parameter β, which determines the probability of executing the expert action in any given time step during a DAGGER rollout, was initialized to 1.0 and then decayed by a factor of 0.75 after each training epoch. The expected performance of the novice at a given epoch during a given DAGGER run was evaluated via 40 rollouts in the target environment.

4. Results

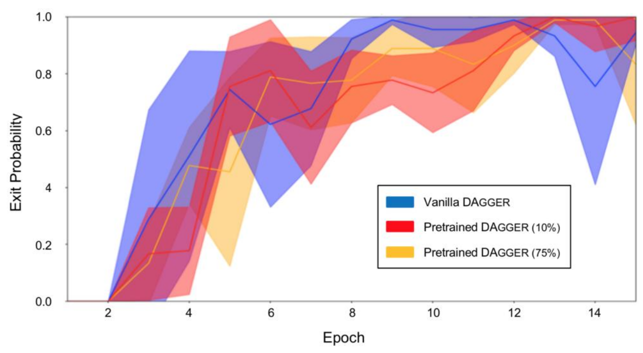

Figure 2: Performance of DAGGER given a randomly initialized policy (blue) and pretrained policies (orange and red). Error bars indicate standard deviation.

Figure 2 shows the baseline performance of DAGGER, as well as its performance when the novice policy was pretrained in simulators where the drift force was 10% and 75% of that of the target environment. Since the performance of DAGGER was essentially identical across the five pretraining regimes, only two of the five were plotted. The error bars (for this chart and for all charts in this paper) show standard deviation and indicate significant noise in the learning process.

Figure 2 appears to show no benefit to pretraining in simulation. Not only is there no additional benefit to pretraining in the higher fidelity simulator relative to pretraining in the lower fidelity simulator, there seems to be no benefit to pretraining of any kind over the baseline approach, where the novice is initialized at random. We examine potential causes for this result in the following section.

[1]Code written for this project can be found at `https://github.com/MichaelKevinKelly/Final_Project_CS230/tree/master/ImitationLearning`.

5. Evaluation

A. *Weight Initialization*

The baseline method tested in our experiments used a novice network initialized with Xavier initialization, a method of weight initialization that allows for more efficient training of deep neural networks by addressing the vanishing gradient problem (Glorot and Bengio 2010). Our method, on the other hand, employed a novice network with weights that were pretrained using TRPO. We suspected that this initialization might result in slower learning, counteracting any benefits from starting out with knowledge gained from the simulator. In other words, we thought that the supervised learning routine used in imitation learning might have been tuning the novice initialized with Xavier initialization much more efficiently than the novice initialized using TRPO pretraining in simulation.

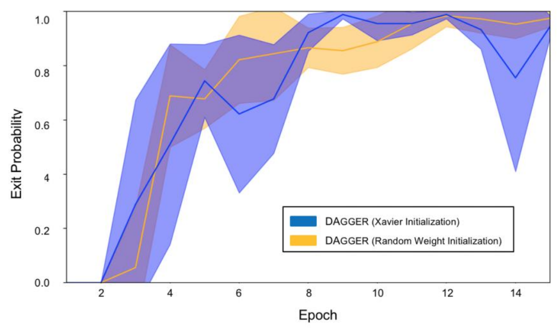

Figure 3: Performance of DAGGER given a novice policy initialized with Xavier initialization (blue - same as in Figure 2) and with weights drawn at random from a distribution over weights similar to those seen in novice networks pretrained with TRPO (orange)

To test this theory, we compared the performance of DAGGER using two random novice initializations. The first used Xavier initialization (i.e. the baseline approach from the experiments described in Section 3.C), while the second was initialized with weights drawn at random from a normal distribution with mean and variance equal to the mean and variance of the weights in the corresponding layer of the TRPO-pretrained network. Experiments were performed in the same manner as described in Section 3.C, and the results are plotted in Figure 3.

The results show no significant difference between the two initializations, perhaps because this network is not sufficiently deep to see a significant benefit from Xavier initialization. In any case, Figure 3 strongly suggests that we must look elsewhere to understand the failure of our RL pretraining method to outperform the baseline.

B. *Catastrophic Forgetting*

Catastrophic forgetting refers to the tendency of artificial neural networks to abruptly lose previously learned knowledge about a task (e.g. task A) when trained on new data relevant to a different task (e.g. task B). Catastrophic forgetting occurs specifically in the continual learning domain - when the network is trained sequentially on multiple tasks - because the network weights that are important for task A are changed to meet the objectives of task B (Kirkpatrick et al. 2017).

To see how catastrophic learning poses a challenge in our domain, we note that there likely exist multiple good policies in a given environment that nevertheless display significantly different behavioral styles. In the bottleneck environment, for example, it is often the case that, as the agent approaches the tunnel, the transition noise perturbs the agent in such a way that it is no longer safe to attempt to enter the tunnel. Rather than continuing ahead, most policies instead learn to loop back and then realign for another attempt at accessing the tunnel. However, some policies learned to perform a clockwise loop, while others learned to perform a counter-clockwise loop, and there does not seem to be a strong reason to prefer one approach over the other. These maneuvers are demonstrated in Figure 4.

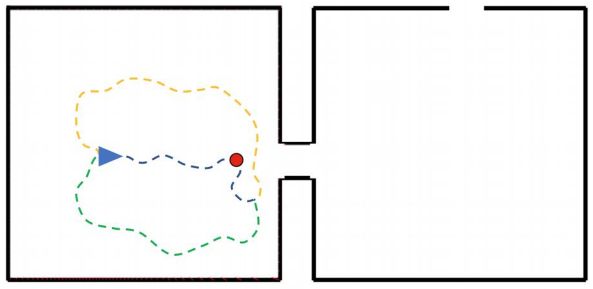

Figure 4: Illustration of different "resetting" behaviors learned by TRPO. The novice's original approach to the exit (the blue line) is perturbed at the red point, and as it is unsafe to enter the tunnel at a sharp angle, the agent loops back for another attempt. The yellow and green lines represent two distinct and yet essentially utility-equivalent ways of performing this recovery maneuver.

In addition to the multiple equally good approaches to this specific subtask, it also seems likely that there are multiple equally good approaches to many other subtasks as well. If we view higher level control policies as amalgamations of these various lower level methods, then we can expect that there will exist a number of distinct high-level policies that can achieve similar levels of performance on the higher-level task. In this domain, it appears that TRPO can converge to multiple such policies.

We suspect that this diversity of good policies is the underlying cause for the poor performance of our method because, when performing imitation learning, it means that we will often be switching between highly distinct behavioral modes. The novice will thus be training on a new and sufficiently distinct task, resulting in catastrophic forgetting. In other words, rather than simply fine-tuning the policy to account for the differences between the simulator and the target environment, the novice agent is instead learning a completely new policy, and in so doing it is overwriting knowledge learned from pretraining in simulation.

To test this hypothesis, we used TRPO to train two policies in an identical environment, with an upwards drift of 0.9m per time step. One of the policies was randomly selected to be the expert while the other was selected to be the novice. We then ran DAGGER using the two policies, performing the same number of runs and evaluating performance in the same manner as the experiments discussed in the previous section.

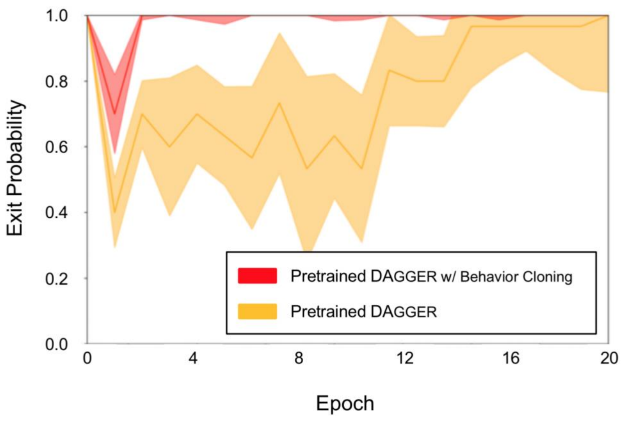

Figure 5: Performance of DAGGER when the novice is pretrained in the target environment with TRPO (orange) and with both TRPO and behavioral cloning (red).

The results of this experiment are plotted on the orange line in Figure 5. Since the novice was pretrained in the target environment, it initially demonstrates perfect performance. However, the initial imitation learning epochs rapidly degrade its performance, demonstrating the effects of catastrophic forgetting. Only once training has progressed for a while and the novice begins to gets better at mimicking the expert does its performance begin to climb back up to its previous level.

Clearly, catastrophic forgetting is a fundamental challenge complicating the use of RL pretraining to speed up imitation learning methods. The following section outlines a method to mitigate this problem.

C. *Overcoming Catastrophic Forgetting*

One of the classic methods used to address catastrophic forgetting is to move the problem from the sequential learning setting to the multitask learning setting. In multitask learning, data from all of the tasks of interest are made available to the neural network during a single training period. This prevents forgetting because the weights of the network can be jointly optimized for all of the tasks simultaneously (Kirkpatrick et al. 2017).

We would like to find a good policy in simulation that is fairly similar to the expert policy, so that when we perform imitation learning we do not completely retrain the novice network. Adapting the multitask learning paradigm to this setting, we decided to interleave rounds of behavioral cloning with TRPO in the pretraining phase. In particular, we divided the 100 TRPO epochs into ten blocks of ten epochs and preceded each block with a single round of behavioral cloning. The inclusion of behavioral cloning in the pretraining phase is intended to guide the reinforcement learning process towards a good policy with similar behavioral characteristics to that of the expert. Since behavioral cloning does not require the presence of the expert at training time, this method does not create any additional supervisory burden.

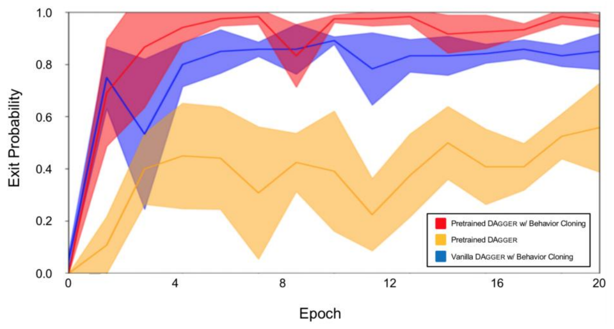

Figure 6: Performance of DAGGER when the novice is pretrained with behavioral cloning (blue), with TRPO in a simulator (orange), and with both behavioral cloning and TRPO in a simulator (red).

We initially tested this approach in the setting where pretraining is performed in the target environment (that is, in the setting where our "simulator" has perfect fidelity). The results are displayed on the red line in Figure 5. As before, the novice initially displays perfect performance and, as before, it suffers from forgetting during the imitation learning process. However, this forgetting is milder than that experienced by the novice pretrained using pure RL, and the novice returns to perfect performance much more quickly. Thus, it appears that combining behavioral cloning and reinforcement learning in the pretraining phase does indeed help us find a good policy that can be quickly fine-tuned to match the expert.

We also tested this method on the main problem of interest: the transfer task where pretraining occurs in an imperfect simulator of the target environment (which again had an upwards drift of 0.9m per time step). The results of this experiment are shown in Figure 6.

The red line again shows the performance of DAGGER when the novice has been pretrained using the combined method consisting of both TRPO and behavioral cloning. This approach clearly outperforms the simpler approach of pretraining using reinforcement learning only, which is represented by the orange line. This suggests that the combined method does indeed find a better novice initialization than the naive, RL-only pretraining method. However, the pure-RL pretraining method is likely an overly forgiving baseline, as it involves training on less data than the combined method.

A fairer comparison can be made against a method that pretrains the novice with behavioral cloning only, using the same number of behavioral cloning rounds as employed by the combined pretraining method. The performance of this approach is shown in the blue line in Figure 6. Unsurprisingly, this approach also appears to avoid the issue of catastrophic forgetting and also demonstrates significantly faster learning than the pure RL pretraining method. The combined method does however appear to slightly outperform the pure behavioral cloning approach, reaching a slightly higher level of performance somewhat faster. If this performance gap is genuine, and not a result of the small number of initializations tested (i.e. three per method), then it may be the case that the novice is employing knowledge about regions of the state space that were explored during RL pretraining in the simulator, but not encountered in the expert trajectories used in behavioral cloning.

6. Conclusion and Future Work

In this paper, we described a method intended to reduce the sample complexity of DAGGER-like algorithms by pretraining the novice with reinforcement learning in an imperfect simulator of the target environment. This approach was shown to perform poorly, likely as a consequence of catastrophic forgetting. We introduced an extension that addresses forgetting by interleaving rounds of behavioral cloning into the RL pretraining process. This method was found to drastically reduce forgetting, and may also improve learning performance in the original transfer task.

Future work on this topic should include more extensive testing using a larger number of novice initializations, environments, and simulators. Exploring more sophisticated ways of addressing catastrophic forgetting might also be a useful extension of this work. One such extension could be to augment the supervised learning procedure used during imitation learning with elastic weight consolidation (Kirkpatrick et al. 2017), which selectively slows down learning on weights that are particularly important for previously learnt tasks. Examining ways to interleave imitation and reinforcement learning at a finer level during pretraining might also be productive. For example, demonstration trajectories provided by the expert could be used to shape the reward function used by the reinforcement learning algorithm, in an inverse reinforcement learning (IRL)-style approach.

Acknowledgement of shared project:

This project was shared with CS 239 (Advanced Topics in Sequential Decision Making). The CS 230 components of the project consisted of implementing the neural network representations of the policies (along with the supervised learning routine used to train them during imitation learning) and the analysis of the effects of catastrophic forgetting, and of using different weight initializations, on the learning performance of the methods presented in the paper. The CS 239 components of the project included the overall design of the methods tested and the implementation of the environment and of the DAgger algorithm. The CS 239 paper can be found at `https://github.com/MichaelKevinKelly/CS239_Final_Project/blob/master/Sequential_Decision_Making_Project.pdf`.

References

[Abbeel, Coates, and Ng 2010] Abbeel, P.; Coates, A.; and Ng, A. Y. 2010. Autonomous helicopter aerobatics through apprenticeship learning. In *International Journal of Robotics Research*.

[Cutler, Walsh, and How 2015] Cutler, M.; Walsh, T. J.; and How, J. P. 2015. Real-world reinforcement learning via multi-fidelity simulators. In *IEEE Transactions on Robotics*, volume 31, 655–671.

[Duan et al. 2016] Duan, Y.; Chen, X.; Houthooft, R.; Schulman, J.; and Abbeel, P. 2016. Benchmarking deep reinforcement learning for continuous control. In *ICML*.

[Glorot and Bengio 2010] Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, volume 9.

[Kim and Pineau 2013] Kim, B., and Pineau, J. 2013. Maximum mean discrepancy imitation learning. In *Robotics:Science and Systems*.

[Kirkpatrick et al. 2017] Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; Hassabis, D.; Clopath, C.; Kumaran, D.; and Hadsell, R. 2017. Overcoming catastrophic forgetting in neural networks. In *PNAS*.

[Laskey et al. 2016] Laskey, M.; Staszak, S.; Hsieh, W. Y.-S.; Mahler, J.; Pokorny, F. T.; Dragan, A. D.; and Goldberg, K. 2016. Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces. In *ICRA*, 462–469.

[Menda, Driggs-Campbell, and Kochenderfer 2017] Menda, K.; Driggs-Campbell, K.; and Kochenderfer, M. 2017. DropoutDAgger: A bayesian approach to safe imitation learning. *arxiv* 1709.06166.

[Ross, Gordon, and Bagnell 2011] Ross, S.; Gordon, G. J.; and Bagnell, J. A. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 627–635.

[Schulman et al. 2015] Schulman, J.; Levine, S.; Moritz, P.; Jordan, M.; and Abbeel, P. 2015. Trust region policy optimization. In *ICML*.

[Sun et al. 2017] Sun, W.; Venkatraman, A.; Gordon, G. J.; Boots, B.; and Bagnell, J. A. 2017. Deeply AggreVaTeD: Differentiable imitation learning for sequential prediction. *arxiv* 1703.01030.

[Zhang and Cho 2017] Zhang, J., and Cho, K. 2017. Query-efficient imitation learning for end-to-end autonomous driving. In *AAAI Conference on Artificial Intelligence*.