Using Decision Trees produced by Generative Adversarial Imitation Learning to give insight into black box Reinforcement Learning models

Caspar Meijer
Supervisor(s): Anna Lukina

EEMCS, Delft University of Technology, The Netherlands
c.j.meijer-1@student.tudelft.nl, A.Lukina@tudelft.nl

Abstract

Machine learning models are increasingly being used in fields that have direct impact on the lives of humans. Often these machine learning models are black-box models and they lack transparency and trust which is holding back the implementation. To increase transparency and trust this research investigates whether imitation learning, specifically Generative Adversarial Imitation Learning (GAIL), can be used to give insights into the black-box models by extracting decision trees. To achieve this, an extension of GAIL was made allowing it to extract decision trees. The decision trees were then measured in terms of performance, fidelity, behavior, and interpretability on three different environments. We find that GAIL is able to extract decision trees with high fidelity and can give insightful information into the expert models. Moreover, further research can be done on more complex environments and black-box models, other surrogate models, and possibilities for more specific local insights.

1 Introduction

Machine Learning (ML) models are increasingly being implemented in the following fields: self-driving cars, healthcare, finance, insurance, war, marine logistics, science, construction, and human resource [17, 23].

In these fields, the ML models make decisions that can have a great impact on the lives of humans. The European Union is aware of this and is advocating for more transparency and giving humans the right to request understandable explanations why certain actions have been taken by the models [4]. Besides giving transparency into the models, their correctness and robustness also have to be extensively tested. The combination of transparency, knowing their correctness and robustness improves the trust people will have in the models [17]. This trust fosters the implementation of the ML models into the aforementioned fields [4, 23, 17, 14].

There are three main methods of creating ML models, namely: Supervised Learning, Unsupervised Learning, and Reinforcement Learning (RL) [6]. In this paper, the focus will continue on RL. RL is a method for specific types of problems. Problems where an agent must maximize a numerical reward in a simulated environment [21]. The agent must learn what to do, what actions to take, given the state of the environment, to discover which actions yield the most reward. The actions the agent takes can also affect situations far in the future and thus affect the immediate and all subsequent rewards. Having to explore the environment and delayed reward are defining features of problems that RL algorithms are designed to solve [21].

A core problem with many of the models produced by conventional RL methods is that they are often difficult to interpret. It is almost impossible to understand for a human why the models make certain decisions and thus actions. Such models are referred to as black-box models [14, 17, 23, 22].

The field of Explainable Reinforcement Learning (XRL) addresses this problem [17, 23, 22]. The survey about XRL written by E. Puiutta and E. Veith concludes, first of all, that it is important while analysing the models, to have a common understanding for when a model is understandable, transparent, interpretable, etc. It has been concluded that the term interpretability best represents the intention of what is to be achieved when analysing the models. In their work interpretability is defined as *"the ability to not only extract or generate explanations for the decisions by a model, but also to present this information in a way that is understandable by human (non-expert) users to, ultimately, enable them to predict a model's behavior"* [17]. Secondly, they have mentioned the different approaches for achieving interpretability, depending on the scope (global vs. local) and the moment of information extraction (intrinsic vs. post-hoc). A global post-hoc analysis is most desirable, as people prefer a global explanation over a local explanation and analysis of an already good performing model is more applicable and easier than creating models that are intrinsically more interpretable [17]. With global vs. local is meant whether the entire model or only a specific section of the model is explained.

The field of Imitation Learning (IL) is about creating entirely new models (surrogate models) from existing experts by observing their behavior [8]. IL is mainly focused on mimicking human behavior to train the new ML models [8]. However, instead of using humans, it is also possible to use black box models as experts. This is useful because humans are not always available for their behavior. The current IL algorithms are highly focused on performance by using complex ML to

mimic the expert as good as possible [8]. However, something that is missing in the IL field is how it can be used to create global post-hoc interpretability analysis all the while still guaranteeing the performance.

In this paper, we will research how and if Generative Adversarial Imitation Learning (GAIL) can be used to give insight into black box RL expert models by generating surrogate Decision Trees (DT). We choose DTs as surrogate models because their are inherently interpretable. DTs consist of decision rules which split the data along a set value. To extract these DTs, three expert models trained for each a specific environments will be given as input to GAIL resulting in multiple DTs. These DTs and their actions are analysed by measuring their performance & fidelity, describing their behaviour in the environment, and whether it is possible for us to interpret its decision rules. Their performance and fidelity will be compared to DTs generated via baseline Behavioural Cloning and 2 other IL algorithms: AGGREVATE & VIPER. Respectively researched by J. Wols & O. Kaaij, my research colleges at the TU Delft.

The core contribution are the modifications made to GAIL in order to extract DTs, this is because GAIL is designed to gradually update its surrogate model, which is not possible with DTs. DTs have to be re-created from the root up when new data is presented. Furthermore, we have added features from VIPER into GAIL that help it extract data points where it is important the correct output is given that lead to higher rewards. Moreover, we have also added the option for GAIL to leverage the expert for extra data.

The layout of the paper starts with section 2 where other related papers are mentioned and briefly explained. Then in the preliminaries in section 3 GAIL and the IL algorithm behavioral cloning are explained. Hereafter in section 4 a formal problem description is given. After the problem description we will explain the research methodology and modifications made to GAIL in section 5. Then, in sections 6 and 7 the setup and results are displayed. Finally in sections 8, 9, & 10 the discussion, conclusions and insights about ethical & responsible research can be found respectively.

2 Related Work

There is the algorithm DAGGER[19] which is an IL algorithm that leverages the expert and surrogate model to generate more data points by asking the expert what to do in states the surrogate explores. From DAGGER the algorithms Q-DAGGER[1] & VIPER[1] and AGGREVATE[18] evolved. Q-DAGGER is an extension from DAGGER that also uses the Q-fuction of the expert to evaluate whether given a certain state it is important to take the right action. Then VIPER is a extension on Q-DAGGER allowing it to train DTs. AGGREVATE leverages the cost-to-go function defined in its paper, to better indicate whether actions given a state will yield reward or cost. In the VIPER and AGGREVATE paper, it is mentioned how cost-sensitive learning aka online-learning, which GAIL does originally, is the same as cost-weighted sampling and then learning [1, 19, 2]. cost-weighted sampling is required to train DTs in a online learning matter and is a core component in the modification of GAIL [2].

The algorithm that is investigated in this paper, Generative Adversarial Imitation Learning [7] (GAIL), is an algorithm that works differently then the previously mentioned papers. It does not have access to the expert policy to leverage it for unknown situations, it is made for cost-sensitive learning which makes it unable to train DTs. That is why the cost-weighted sampling from Q-DAGGER has to be implemented. GAIL is an extension of Generative Adversarial Networks [5], which describes how two neural networks, a discriminator and a generator, work together or more against each other to become better, which is further explained in the preliminaries.

None of the paper mention the interpretability of the surrogate models or how this can be used as a global post-hoc analysis [1, 19, 7].

3 Preliminaries

3.1 Terminology

In the appendix in section A the terminology used in this paper are explained. This can be useful for people that are not acquainted with the research field of machine learning, reinforcement learning, etc.

3.2 Original GAIL

The original GAIL algorithm is designed to train a neural network with Trust Region Policy Optimization [20] from expert data, without access to the expert model. The full pseudo-code of GAIL can be seen in Algorithm 2 in the Appendix.

A short description of how to interpret the algorithm: For each iteration, use the generator to generate new trajectories, train the discriminator to distinguish expert trajectories from generator trajectories. Then after training the discriminator, train the generator with the output of the discriminator as its cost function. Over multiple iterations this leads to the discriminator not being able to distinguish between expert or generator data, at this convergent point the generator is like the expert [7]. The key takeaway from the algorithm that is the Generator step (5 6) is done in a gradual fashion, where only a small update to the model is done to prevent the model from changing too much from the previous iteration.

3.3 Behavioral Cloning

Behavioral cloning (BC) is a basic IL algorithm where the states and actions are trained in a supervised learning matter on neural networks or in our case, decision trees. BC suffers from compounding errors when the surrogate models find themselves in situations never seen before [19]. This is what the papers of DAGGER, AGGREVATE, VIPER, & GAIL prevent by including a cost function that is related to the expert. The pseudo-code for BC can be found in Algorithm 3. It will function as the baseline in this research.

3.4 Training expert policies

First, the Deep Q Learning (DQL) [13] algorithm was set up with TensorForce [10] to create expert models for the classical problems presented in OpenAI GYM [3]: MountainCar-v0, Cartpole-v1, and Acrobot-v1.

4 Problem Description

there is the problem of the surrogate models of the original GAIL that are black box like models. The goal is to modify GAIL such that it produces interpretable models, specifically DTs. Furthermore, the GAIL algorithm relies on cost-sensitive learning to train the surrogate model, this is not possible with DTs. DTs are fitted on a data set at once, so the only way to gradually train the DT is to gradually make the data set on which the tree is fitted better. This is called cost-weighted sampling [2].

5 Methodology

5.1 GAIL with decision trees

To tackle the problem of GAIL not being able to train DTs because it relies on cost-sensitive learning. The GAIL paper describes how the discriminator can function as a local cost function for a given state-action pair [7]. Thus the algorithm was changed so that the discriminator output would represent the probability that a state-action pair was going to be used to train the DT. This modification will act as the cost-weighted sampling [1, 19, 2].

5.2 Including expert policy

In the recommendations of the GAIL paper, it is mentioned that GAIL would perform better if it had access to the expert model [7]. They state the algorithm would be better because it could ask the expert what to do in new never seen before situations. This is something that the other algorithms, like VIPER & AGGREVATE already do. Thus the access is added at the end of each iteration. The just fitted DT would generate a defined number of trajectories in the environment. The trajectories are then given to the expert to ask which action it would take at each state. The result of this are new state-action pair expert mappings.

5.3 Extra action cost weighted sampling

In the Viper paper, they use a cost function of how important it is to take a specific action by the agent given a state. This is done by calculating the difference between the probability the expert agent takes an action, and the lowest probability action [1]. $l(s, a) = P(a_{\pi_E}|s) - min(P(a|s))$ This results in removing states where each action has about the same probability.

5.4 Add action cost to train discriminator

This solution of sampling the state-action pairs according to their importance also gave rise to a new idea. Namely, also including this indicator together with the state-action pairs resulting in state-action-cost pairs. With these new pairs, the discriminator has an extra data point to discriminate on.

5.5 Distinct generator trajectories

The generator model is supposed to produce trajectories similar to the expert. Initially, we did not have the generator rollout its own trajectories because that would lead to data sets that are not the same length. So instead of creating a defined number of trajectories, we thought of having the generator rollout trajectories until we would have a list of state-action pairs with the same length as the experts state-action pairs. This way the discriminator was also not just focusing on the mappings of the state-action pairs but also on whether the states were generated by the expert or the generator.

5.6 Follow-The-Leader

Both VIPER [1] and AGGREVATE [18] pick the best performing surrogate model at the end of their iterations. This is called a Follow-The-Leader strategy. The best performing surrogate model is the model that achieves the highest reward over a number rollouts.

5.7 Measuring the decision trees

These modifications to GAIL produced surrogate DT models which had to be measured and analysed in order to say something about their ability to give insight into the expert models. In order extract decision rules from the DTs that give insight into the expert, it is required that the DT has a high functional resembles of the expert, further mentioned as fidelity. The following metrics were defined in order to later conclude something about their ability to give insights.

Performance: In order to measure the performance of the surrogate models their average reward and standard deviation were measured over a defined number of rollouts.

Fidelity: In order to measure how well the surrogate model reflects the expert's logic, we will measure how often the surrogate model takes the same action as the expert.

Trajectory Description: The surrogate trajectories will be compared to the trajectories of the expert to see if they are similar. It is possible for the surrogate model to achieve the same reward but with a different trajectory path. If the trajectories are not similar then the decision rules in the decision tree do not reflect the inner workings of the expert.

Interpretability: In order to measure its interpretability, the model is drawn, is explained by writing the decision rules down in text, the state-action space is plotted to show its distribution.

5.8 Modified version of GAIL

The algorithm is initiated with the Expert policy: π_E, the untrained DT policy θ_0, the discriminator neural network with 2 layers, each 32 nodes: D_0, the number of initial expert trajectories: τ_E: $N\tau_E$, the number of extra trajectories made by the DT but with expert actions at the end of each epoch to extend the expert state-action pairs: N_θ, the number of epochs: N_{Epochs}. At last the booleans indicating whether unique generator trajectories, extra trajectories at the end should be made, the state-action pairs should be sampled with their action aost, and whether to include their action cost when training the discriminator: $ownGeneratorTrajectories$, $hasAccessToExpert$, $sampleWithC$, $discriminateWithC$ The full pusedo-code can be found in Algorithm 1.

Algorithm 1: Modified version of GAIL

Input: π_E, π_{θ_0}, D_0, $N\tau_E$, N_θ, N_{Epochs}, $ownGeneratorTrajectories$, $hasAccessToExpert$, $sampleWithC$, $discriminateWithC$

Output: Surrogate Decision Tree

```
Generate expert trajectories τ_E form π_E
for Epoch: i = 0, 1, 2, ..., N_Epochs do
    if ownGeneratorTrajectories then
        Generate trajectories τ_θi with π_θi
    else
        Create τ_θi with states from τ_E & actions
          mapped with π_θi(s)
    if discriminateWithC then
        Init List_c filled with action cost of each (s, a)
          in τ_E & τ_θi
        for c in List_c do
            Add c to (s, a) in correct τ_E & τ_θi
              resulting in τ_E & τ_θi with (s, a, c).
    Train D with τ_E & τ_θi.
    if sampleWithC then
        if not discriminateWithC then
            Init List_c of action cost: for each (s, a)
              pair in τ_E & τ_θi
        Sample both τ_E & τ_θi weighted with List_c.
    Initialize new train list: T_List(s,a)
    for (s, a, (q)) in τ_E & τ_θi do
        if random < D(s, a, (q)) then
            append (s, a) to T_List(s,a)
    Train the π_θi+1 with T_List(s,a).
    if hasAccessToExpert then
        Use the π_θi+1 to generate new trajectories τ_θ
        Extend the τ_E with states from the τ_θ &
          actions with the π_E(s)
    Save π_θi+1
π_θ = π_θi with highest reward
Return π_θ
```

6 Experimental Setup

6.1 Environments

In this section, the environments are listed in which the experts are trained. See screenshots in Figure 1.

- MountainCar-v0, with a minimal and maximal reward of -200 & -110±3 respectively. The goal is to reach the finish on top of the right hill as fast as possible. The cart has to build up momentum before it can move up the right hill. The State-space is the cart's position and velocity. The Action space is powering the cart to move left, right, or not at all.
- CartPole-v1, with a minimal and maximal reward of 9±0.5 & 500 respectively. The goal is to balance the pole on top of the cart for as long as possible. The State-space is the pole angle relative to the cart, the pole angular velocity, the cart's position, and velocity. The Action space is powering the cart to move the cart left or right.
- Acrobot-v1, with a minimal and maximal reward of -500 & -60±2 respectively. The goal is to move the part of the pendulum above the top line as fast as possible. The State-space is the angles of the first and second links and the angular velocities of the links. The Action space is providing +1, 0, or 1 torque to the joint between the two links.

(a) MountainCar-v0 (b) CartPole-v1 (c) Acrobot-v1

Figure 1: Screenshots of Environments.

6.2 Environment Experts

The experts are trained with the use of TensorForce [10]. In Table 7 in the appendix, the environment-specific settings for each environment are displayed. All of the other settings are default. In Table 1 the results of 100 rollouts are displayed, containing the average reward, the standard deviation of the reward, and the minimum and maximum reward.

Description of Expert behavior

In this section, short descriptions of the expert rollouts are given to later compare with the surrogate models extracted with GAIL. Screenshots of the rollouts can be found in the appendix in Figures 5, 6, & 7

MountainCar: The expert first moves right up the finish hill a little bit, then moves left up till around 60% to 80% of the hill and then accelerates right to the finish.

CartPole: The expert lets the pole fall to the left but is able to put the cart underneath the pole just before it reaches the end of the map. Then it lets the pole slowly fall to the right till the end of the map and then lets the pole fall to the left again. The rollout often ends before it can reach the right side of the map.

Acrobot: The expert swings the pendulum left and right to build up momentum until it goes over the line and finishes. The expert does not seem to have a clear trajectory it follows every time, making it difficult to see whether it does something else than move left and right to build up momentum.

Table 1: Expert Performance

Environment	Average Reward	SD	Minimum	Maximum
MountainCar-v0	-115.1	2.02	-125	-113
CartPole-v1	500	0.00	500	500
Acrobot-v1	-92.73	14.24	-130	-76

6.3 Discriminator Surrogate model settings

The following settings were gathered from the GAIL paper and slightly modified to prevent overfitting [7]. The discriminator is a multi-layer neural network created with Tensorflow [11] & Keras [11]. It had 2 layers with each 32 nodes and they are sequentially placed dense layers with activation function LeakyReLU with alpha=0.05 and a dropout rate of 0.2. this alpha and dropout rate are used to avoid overfitting the data. The output layer has a sigmoid activation function so that the output is between 0 and 1. The input shape is so that it fits state-action(-cost) pairs. The output via sigmoid is 1 single scalar. The optimizer is Adam the loss function is cross-entropy.

The surrogate model is a DT that is implemented using the Scikit-learn [16] library. The hyper-parameters which are specified are the max depth of a tree and the Cost-Complexity Pruning[16] alpha (ccp_alpha) set to 0.013, which helps avoid overfitting the data.

6.4 Hyper-Parameters

In this section, we will describe all of the settings for the modified version of GAIL.

Set parameters:

These parameters were set and were not changed while gathering the results.

- Starting expert trajectories: 1
- If access to an expert: Trajectories to be added after training the surrogate DT: 1
- Epochs to run GAIL: 10
- Epochs to train the discriminator each GAIL epoch: 500
- Rollouts to test performance for Expert & BC: 100
- Rollouts to test performance for Surrogate: 50

Variable parameters:

These parameters were not set and every combination was tried while gathering the results.

$Environment$**:** The openAI GYM Environment names: MountainCar-v0, CartPole-v1, Acrobot-v1

$Depth$**:** The Max depth of the DT: 1, 2, 3

$ownGeneratorTrajectories$**:** Create unique Generator trajectories instead of using the expert states: True / False

$hasAccessToExpert$**:** Use Generator to create extra trajectories with actions of expert: True / False

$sampleWithC$**:** Sample with the action cost for each state-action pair: True / False

$discriminateWithC$**:** Include the action cost with the state-action pairs for the Discriminator: True / False

7 The Results:

In this section, we will take a closer look at the results. generated by running the modified version of GAIL with all combinations of the variable parameters two times. So for there are 2 data points for each variable parameter combination. All of the data points are located in Tables 8, 9, & 10 in the appendix.

7.1 Expert Initial Trajectory Reward

At the beginning before starting the process of extracting the DTs, 1 rollout was done given the expert, resulting in the performance seen in Table 2.

Table 2: Expert Initial Trajectory Reward

Environment	Reward run 1	Reward run 2
MountainCar-v0	-114	-124
CartPole-v1	500	500
Acrobot-v1	-78	-87

7.2 Baseline BC Results:

Each of the values in table 3 are the average of 3 baselines fitted on 1 expert trajectory. The reward of the expert trajectory is located in table 2. The run number refers to which expert trajectory reward the baselines were fitted on. These results can also be found in the tables 8, 9, & 10 in the appendix.

Table 3: Baseline Results

Environment	Depth	BC Reward		BC SD		BC Fidelity	
		Run 1	Run 2	Run 1	Run 2	Run 1	Run 2
MountainCar-v0	1	-127.5	-128.6	20.779	21.316	90.5%	91.2%
MountainCar-v0	2	-122.9	-129.7	17.993	27.427	93.2%	92.2%
MountainCar-v0	3	-121.9	-132.3	15.323	29.229	93.8%	92.8%
CartPole-v1	1	9.31	9.24	0.666	0.705	86.4%	86.0%
CartPole-v1	2	9.263	9.247	0.773	0.706	86.4%	86.0%
CartPole-v1	3	9.217	9.37	0.755	0.73	86.4%	88.9%
Acrobot-v1	1	-92.427	-92.987	21.99	22.21	81.6%	81.8%
Acrobot-v1	2	-92.363	-88.57	29.359	25.592	84.6%	83.3%
Acrobot-v1	3	-102.22	-86.53	31.931	19.775	84.3%	83.0%

7.3 GAIL Results

We analysed the data from the tables 8, 9, & 10 in the appendix by sorting them on GAIL reward from high to low and then sorting on GAIL fidelity from high to low, also further refered to as the sort rule. This way the DT at the top of the table has the highest reward given the highest fidelity. The distributions of the states and actions can be seen in Figures: 2, 3, 4 or in higher resolution in the appendix. The distributions are expert trajectories of 100 rollouts put into the DT, that way it is possible to tell the DT matches the experts behavior.

MountainCar-v0:

For MountainCar-v0 the highest fidelity was 95.2% with a reward of -116.88 and an SD of 1.267 with a DT with depth 3. This DT was gathered with $ownGeneratorTrajectories$, $hasAccessToExpert$ $discriminateWithC$ all set to true. The DT can be seen in Figure 2(a) and in Figure 8 in the appendix.

The DT with the 3rd highest-fidelity of 94.7% has a reward of -117.0 and SD of 1.523 also had a depth of 3 but was trained on only 1 expert trajectory worth of data. Indicating that the GAIL is able to gather DTs with high performance & fidelity using little data for MountainCar-v0. This DT was gathered with $ownGeneratorTrajectories$ & $discriminateWithC$ set to True. This DT can be seen in Figure 2(b) and in the large Figure 9 in the appendix.

The DT with the 4th highest fidelity of 93.9% has a reward of -116.52 and an SD of 1.873 but has a depth of 2. This DT was gathered with $ownGeneratorTrajectories$ & $hasAccessToExpert$ set to True. This DT can be seen in Figure 2(c) and in Figure 10 in the appendix.

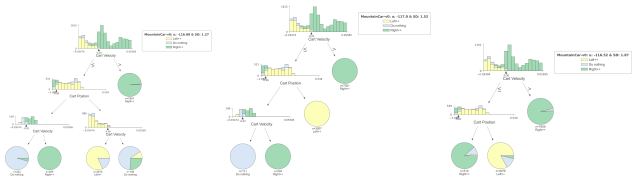
Something worth noting is that the setting of $sampleWithC$ has a negative effect on results. With the sorting rule defined above the first 24 DTs, all have $sampleWithC$ set to False.

Performance: The performance of the DTs all match that of the expert with only 1 or 2 reward points lower. The Standard Deviation is also similar. The performance is significantly higher than the baselines.

Fidelity: The Fidelity is in each of the DTs higher than the baselines with at least 2 percentage points.

Trajectory Description: The trajectory is very similar to that of the expert, first moving slightly to the right and then going up the hill to the left and then powering right again to the finish.

Interpretability: All three of the DTs take the same actions but respectively with fewer nodes. How we would interpret the DT is as followed: Power the cart in the direction it moving but not if it is near the left edge of the valley, then power the cart back down to the right giving it the momentum to get to the top of the hill on the right and finishing.



(a) 95.2% (b) 94.7% (c) 93.9%

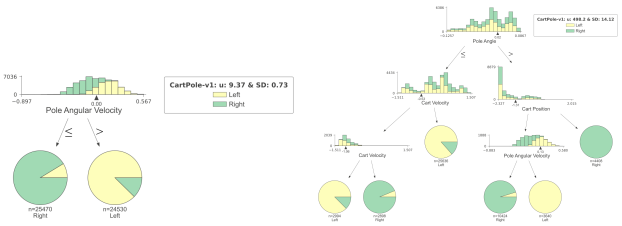
Figure 2: Small Screenshots of MountainCar-v0 DTs.

CartPole-v1:

For CartPole-v1 the highest fidelity was 86.4% but with a reward of 9.37 and an SD of 0.73 with a DT of depth 1. The DT was gathered with $ownGeneratorTrajectories$ set to True. The DT can be seen in Figure 3(a) and in Figure 11 in the appendix.

The first DT that resembles the reward of the expert has a fidelity of 61.7% which is number 45 from the top with the defined sort rule. It has a reward of 498.2, an SD of 14.116 and a depth of 3. This DT was gathered with $ownGeneratorTrajectories$, $hasAccessToExpert$, $sampleOnC$, and $discriminateWithC$ all set to True. The DT can be seen in Figure 3(b) and in Figure 12 in the appendix.

Something worth noting is that the setting of $hasAccessToExpert$ set to False has a negative effect on the reward, but a positive result on Fidelity. The first 21 DTs with the highest fidelity all had $hasAccessToExpert$ set to False.



(a) 86.4% (b) 61.7%

Figure 3: Small Screenshots of CartPole-v1 DTs.

Performance: The performance of the DTs often does not match that of the expert, only 3 times did GAIL come close to the expert reward. The performance is however often higher than the baselines.

Fidelity: GAIL can generate DTs with a fidelity that looks high, but the reward is in these cases close to the minimum reward possible. With a higher reward, the fidelity lowers.

Trajectory Description: The Trajectory of the DT with high fidelity is similar to the expert in that it lets the pole fall, however it is not able to recover from this. The DT with a similar reward does something different from the expert, namely, it balances the pole close to the middle of the map and does not reach the edges at all.

Interpretability: The main point that can be made up from the DT with the highest fidelity is that the actions of the expert seem to force the pole to always be falling a bit, either to the left or right. As the pole falls to the right the DT matches the expert by moving the cart to the left, making the pole fall faster. The expert is however able to restore the pole from falling, which the DT is not able to do. The DT with the high reward does not have nodes that describe a clear split of the data, making it difficult to describe how it makes the choices.

Acrobot-v1:

For Acrobot-v1 the highest fidelity was 84.7% and a reward of -95.89 and an SD of 27.36 with a DT of depth 3. The DT was gathered with $ownGeneratorTrajectories$ & $discriminateWithC$ set to True. The DT can be seen in Figure 4(a) and in Figure 13 in the appendix.

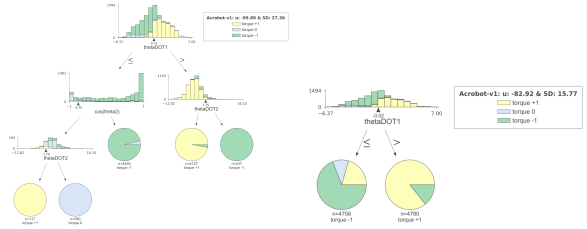
There are DTs with an average reward higher than the expert and with the same SD. The DT with the second-best reward has a depth of 1. It has an average reward of -82.92, an SD of 15.77, and a fidelity of 80.2%. This DT was gathered with only $hasAccessToExpert$ set to true. This DT can be seen in Figure 4(b) and in Figure 14 in the appendix. The first 6 DTs all had $hasAccessToExpert$ & $sampleWithC$ set to False, so they were all trained with only 1 expert trajectory worth of data.

Performance: The reward resembles the expert but the SD is almost 2 times higher. The performance is often higher than the baselines.

Fidelity: The fidelity of the DT makes it seem it resembles the expert quite well. The fidelity is often the same as the baselines.

Trajectory Description: The trajectory is similar to the expert in that it is building up momentum until it moves over the line and finishes. However, it is difficult to compare it in depth as it moves the pendulum fast.

Interpretability: The DT with the highest fidelity gives torque in the same direction as the first link is moving, making it build up momentum. Then the other nodes look at the angle & velocity of the second link, but it is not clear what it tries to do here. The DT with the high reward only looks at the velocity of the first link, also indicating that it is used for the build-up of momentum.



(a) 84.7% (b) 80.2%

Figure 4: Small Screenshots of Acrobot-v1 DTs.

7.4 GAIL vs BC vs AGGREVATE vs VIPER

In this section we will be comparing the results extracted from GAIL with the baseline BC, AGGREVATE, & VIPER. Due to the fact that AGGREVATE & VIPER both had their own expert, their trajectories do not match, so we cannot compare whether their DTs extract the same decision rules. We can, however, compare the DTs on how well they resemble their experts in terms of fidelity and thus whether they can give insights into their experts. We can also compare how well they can resemble their experts in terms of performance, but with this, we have to keep in mind that the experts are different and can thus have different trajectories to achieve their reward.

MountainCar-v0 comparison

For MountainCar-v0 every algorithm was able to closely resemble their expert in terms of performance and fidelity and each of them did better than their baseline. The expert of VIPER performed best and VIPER was able to create a DT with almost 100% fidelity. AggreVate was only able to obtain a fidelity of 80% which is much lower than that of GAIL and VIPER.

Table 4: Algorithm Results of MountainCar-v0

		GAIL			AggreVaTe			Viper	
	Expert	BC	GAIL	Expert	BC	AggreVaTe	Expert	BC	Viper
Depth	N/A	3	3	N/A	1	1	N/A	3	3
Reward	-115.1	-121.9	-116.88	-120.6	-124.35	-120.35	-112.1	-114.17	111.2
SD	2.02	15.323	1.267	19.9	3.9	4.3	1.8	3.3	2.7
Fidelity	N/A	93.8%	95.2%	N/A	79.0%	80.0%	N/A	95%	97.1%

CartPole-v1 comparison

For CartPole the algorithms show a large diversity of results. GAIL was able to extract a DT with the highest fidelity but was not able to match the performance of the expert. GAIL is also the only algorithm that performs better than the baseline in terms of fidelity. The other algorithms outperform their baseline in terms of performance. The VIPER DT has a fidelity of 62.8% making it resemble the expert far less than the AGGREVATE & GAIL.

Table 5: Algorithm Results of CartPole-v1

		GAIL			AggreVaTe			Viper	
	Expert	BC	GAIL	Expert	BC	AggreVaTe	Expert	BC	Viper
Depth	N/A	1	1	N/A	2	2	N/A	2	2
Reward	500	9.31	9.37	470.5	227.5	474.95	500	466.5	500
SD	0	0.666	0.73	71.5	46.2	52.4	0	33.5	0
Fidelity	N/A	86.0%	86.4%	N/A	78.0%	70.0%	N/A	84.1%	62.8%

Acrobot-v1 comparison

For Acrobot the algorithms were trained using experts with significantly different performance. Namely, the performance of the expert used in AGGREVATE indicates that it has not been optimized. Other than that, GAIL is the only algorithm that outperformed its baseline in terms of fidelity. Furthermore, the fidelity of GAIL was higher than those of AGGREVATE & VIPER. Even though the fidelity of VIPER is lower than that of GAIL, the performance resembles its expert better.

Table 6: Algorithm Results of Acrobot-v1

		GAIL			AggreVaTe			Viper	
	Expert	BC	GAIL	Expert	BC	AggreVaTe	Expert	BC	Viper
Depth	N/A	3	3	N/A	3	3	N/A	2	2
Reward	-92.73	-102.22	-95.89	-179.95	93.4	-91.95	-85	-114.2	-84.3
SD	14.24	31.931	27.36	50.1	34.1	34	17.1	93.8	20.81
Fidelity	N/A	84.3%	84.7%	N/A	53.0%	53.0%	N/A	84.3%	82.1%

8 Discussion

In this section, we will first discuss the results of the extracted DTs of GAIL and then discuss the comparison between the other IL algorithms.

From the experiment mentioned in section 6, we can see that GAIL is able to perform better than its baseline in extracting DTs with higher fidelity. This higher fidelity allows us to better use the decision rules in the DT to gain insights into the workings of the expert.

For example, with the extracted decision trees for MountainCar-v0 we can indicate the location on the map where the expert starts to accelerate towards the finish. Another thing that can be seen in the results is that GAIL is consistent with extracting decision rules. The first 2 decision lines in each of the DTs are the same. What can be said about the changes is that using $sampleWithCost$ seems to have a negative effect on fidelity as the first 24 results set this setting to false. An explanation for this could be that this cost function focuses on reward and not on fidelity. In addition, the use of $ownGeneratorTrajectories$ is present in the first 5 highest-fidelity results, indicating that it helps extract those last few action maps.

Moreover, from the results of CartPole-v1 we can extract that the expert does not use the angular velocity of the pole, the most valuable feature of the environment, to decide on his actions. If this analysis would have been used in practice, a re-evaluation of the expert is recommended. On the other hand, GAIL was unable to generate a DT that matched the

expert's performance and behavior. The DT that most closely matched the expert's performance only had a confidence level of about 62%, which was visible in his trajectory description, which was different from the expert. This raises the question of whether GAIL or DTs are suitable for extracting behavior from complex experts. Additionally, there is the question of whether it is better than the baseline as there is a baseline with higher fidelity. But this could also be the result of the complex expert that are require larger DTs to describe. GAIL does perform better than the baseline in terms of extracting a DT with a reward that resembles the expert. Furthermore, the negative effect of $hasAccessToExpert$ can be explained by the fact that the expert may not be able to recover from the new trajectories generated by the generator. These trajectories are then nevertheless classified as expert trajectories, resulting in a bias towards unfavorable states. Besides, $ownGeneratorTrajectories$ was also present in the DT with the highest fidelity, but not consistent thereafter.

The results of Acrobot-v1 indicate that the expert uses the angular velocity of the first link to build up momentum, which is the most valuable feature of the environment. The DT extracted with GAIL has a slightly higher fidelity compared to the baseline. This might indicate that the decision-making of the experts does not require complex data sampling. The top results also had $ownGeneratorTrajectories$ enabled but not consistent thereafter. The top half of the results mostly had $sampleWithCost$ set to false, indicating it has a negative effect on the fidelity.

And finally, the comparison with GAIL, AGGREVATE, and VIPER indicates that GAIL outperforms the other algorithms in terms of fidelity. Making it the most suited algorithm to give insights into the expert. Something to keep in mind is that all 3 the algorithms were examined with different expert models, thus it is not possible to conclude whether the extracted DTs would differ in their decision rules. It can also be said that both GAIL, AGGREVATE and VIPER are able to extract DTs with high reward alone.

9 Conclusions and Future Work

In this paper, we have demonstrated that GAIL is able to produce insight into black-box reinforcement learning models for small environments by extracting decision trees (DT) that are functionally ver similar these black-box models. GAIL is able to produce DTs that better resemble the black-box models than the baseline behavioral cloning, AGGREVATE, & VIPER. With this high similarity in behavior, it was possible to use the decision rules in the DTs to interpret the decision-making of the black-box models. It is more important to focus on the similarity with the black-box than the reward from the environment. Because only with high fidelity can something truly be said about the black-box model. Furthermore, it can be stated that the DTs generated with GAIL are interpretable. They are small and do not have a lot of decision rules. However, this is a setting that is adjusted in the algorithm that fits the DT, not in GAIL that provides the data for fitting the DT. The DTs generated by GAIL is slightly less interpretable than those generated by AGGREVATE, & VIPER but they do not resemble the expert with which they are used as much.

In order to make GAIL able to extract DTs certain modifications had to be made. The core adjustment is using the discriminator output, which functions as a loss function to the generator, as the probability that a data point is to be used to train the DT. Moreover, allowing the generator to generate its own trajectories to be distinguished from the expert by the discriminator has proven to be a viable modification. It was used in all of the DTs with the highest fidelity. Furthermore we have included features found in VIPER & AGGREVATE into the algorithm. Namely, leveraging the expert to generate more data for data points not yet seen, the action cost weighted sampling and the Follow-The-Leader DT selection feature.

While conducting the research limitations and ideas for future work came to mind. For instance: Leveraging the surrogate model and expert to generate more trajectories can work counterproductive if the expert is not optimized for every possible state. The surrogate model might make mistakes in which the expert would be asked what it would do, but if the expert has not been trained to recover from such mistakes its action is of no use. Like asking a professional skydiver what it would do in a situation 10 meters above ground after jumping from an airplane. Maybe using the discriminator to first select, with probability, only those states it classifies as the expert, to prevent the bias of unfavorable states.

Furthermore, interpretability is subjective to the complexity of the surrogate model, which in turn depends on the expert model which is optimized for the environment. Thus indirectly the interpretability of the surrogate model will depend on the complexity of the environment. Extending on this, for complex environments, it might be an idea to analyze the trajectories in more depth to see where in the trajectory the expert and surrogate model actions differ from each other. Allowing to use the insights in those parts where they resemble each other a lot.

Moreover, DTs are limited to splitting the data on fixed data values. For instance, they are not able to compare features with each other, $x < y$. To extend on this, there are different types of Tree training algorithms like Gradient Boosting which do incrementally train the DT in an online learning matter. Maybe this is a better solution to be used for GAIL as it originally relies on small step improvements. With the current DT training algorithm, it rebuilds the tree each time it receives new data.

The code that was used to conduct this research can be found on GitHub [12]

.

10 Responsible Research

10.1 Ethical aspects

This research was conducted without the use of any human subjects. But still something can be said about the ethical aspects of this research, because as the paper starts it stats that machine learning models are increasingly being implemented into fields that have impact on human lives. The use of black box models in situations of war, insurance, healthcare, etc would raise the question of how ethically correct their inner workings and training data are. We all know the examples of machine learning models that become racist if the data that they are trained on is not carefully unbiased. On top of that it is the European Union that is advocating for more transparency into the machine learning models to increase the trust of the people [4].

10.2 Reproducibility

The article written by M. Baker and D. Penny [15] indicates that there is a reproducibility crisis. However, this paper does not mention statistics about the computer science field. But eitherway it is important to assess the reproducibility of the research conducted in this paper. The research was done with software that is freely available on the internet, the used libraries are cited in the text [10, 11, 16, 3] and the papers of algorithms used are also cited in the text. The experts that are used in the paper can be learned by copying the settings into TensorForce [10] and running the algorithms. Furthermore, the code that was created during this research is publicly available on Github at time of release [12].

10.3 Research Integrity

In 2018, the Netherlands Code of Conduct for Research Integrity has been published by KNAW and NFU and NWO and TO2-Federatie and Vereniging Hogescholen and VSNU [9]. This code of conduct mentions five principles that form the basis of integrity in research. They are mentioned below.

Honesty

The research in this paper is done as honestly as possible, by reporting the research process, settings and results as accurately as possible. All of the data used in the research can be found in the appendix in large data tables. In order to create honest results the whole process was done 2 times after which the data analysed. All of the data points show their standard deviation where applicable so to indicate the values are not highly accurate.

Scrupulousness

The research and methods in this paper are taken from other research papers and carefully curated that they do indeed reflect their original intend. On top of that the research was peer-review multiple times by a scholarly supervisor which agreed with the methods undertaken.

Transparency

All of the data used in the paper can be found in the appendix and the code can be found on the github [12]. There were no significant external stakeholders that had any financial motive for supporting this research. Apart from the TU Delft themselves, but that is in the broader sense and not specific for this research. No data used in the paper is withheld from the public. Something that did happen was that while developing, after having converted the decision tree vector files in this paper to png, the other the decision trees of each of the rows in the large result tables in the appendix got deleted. However, it is possible to regenerate the new data by running the publicly available code. The changes to the code can be followed via the git branches system.

Independence

The research was done with no commercial or political nature involved. The choice of method was influenced by our course supervisor which had setup the initial guidelines for the research. These were located on a project forum website of the TU Delft and contained a brief description about how ai was increasingly being used to control safety critical systems and that interpretability is a problem holding back the implementation of such machine learning algorithms. This supervisor also helped searching for related papers to the topic.

Responsability

The research did not happen in isolation and no human or animal test subjects were used. The research is societally relevant because of the previously mentioned reasons. About how AI should be interpretable and it increases trust and transparency.

Ethically this research is giving insights into how black box models are making decisions. This could bring a lot more transparency into the field. The project code will be published so that the reproducibility is very high.

References

[1] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable Reinforcement Learning via Policy Extraction. *arXiv:1805.08328 [cs, stat]*, January 2019. arXiv: 1805.08328.

[2] Alina Beygelzimer, John Langford, and Bianca Zadrozny. Machine Learning Techniques—Reductions Between Prediction Quality Metrics. In Zhen Liu and Cathy H. Xia, editors, *Performance Modeling and Engineering*, pages 3–28. Springer US, Boston, MA, 2008.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016. original-date: 2016-04-27T14:59:16Z.

[4] European Parliament. Directorate General for Parliamentary Research Services. *A governance framework for algorithmic accountability and transparency.* Publications Office, LU, 2019.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron

Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[6] Hunter Heidenreich. What are the types of machine learning?, December 2018.

[7] Jonathan Ho and Stefano Ermon. Generative Adversarial Imitation Learning. *arXiv:1606.03476 [cs]*, June 2016. arXiv: 1606.03476.

[8] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation Learning: A Survey of Learning Methods. *ACM Computing Surveys*, 50(2):1–35, March 2018.

[9] KNAW, NFU, NWO, TO2-Federatie, Vereniging Hogescholen, and VSNU. Nederlandse gedragscode wetenschappelijke integriteit, 2018. Medium: application/pdf Type: dataset.

[10] Kuhnle (last), Alexander (last), Schaarschmidt (last), Michael (last), Fricke (last), and Kai (last). Tensorforce: a TensorFlow library for applied reinforcement learning, January 2022. original-date: 2017-03-19T16:24:22Z.

[11] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015.

[12] Caspar Meijer. interpretable-gail: Repository for research project about using decision trees produced by generative adversarial imitation learning to give insight into black box reinforcement learning models. at https://github.com/capsar/interpretable-gail, 2022.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv:1312.5602 [cs]*, December 2013. arXiv: 1312.5602.

[14] Christoph Molnar. Interpretable Machine Learning. page 312, February 2020.

[15] Open Science Collaboration. Estimating the reproducibility of psychological science. *Science*, 349(6251):aac4716, August 2015.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python, 2011.

[17] Erika Puiutta and Eric MSP Veith. Explainable Reinforcement Learning: A Survey. *arXiv:2005.06247 [cs, stat]*, May 2020. arXiv: 2005.06247.

[18] Stephane Ross and J. Andrew Bagnell. Reinforcement and Imitation Learning via Interactive No-Regret Learning. *arXiv:1406.5979 [cs, stat]*, June 2014. arXiv: 1406.5979.

[19] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, June 2011. ISSN: 1938-7228.

[20] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust Region Policy Optimization. *arXiv:1502.05477 [cs]*, April 2017. arXiv: 1502.05477.

[21] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning series. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018.

[22] Lindsay Wells and Tomasz Bednarz. Explainable AI and Reinforcement Learning—A Systematic Review of Current Approaches and Trends. *Frontiers in Artificial Intelligence*, 4:48, 2021.

[23] Amber E. Zelvelder, Marcus Westberg, and Kary Främling. Assessing Explainability in Reinforcement Learning. In Davide Calvaresi, Amro Najjar, Michael Winikoff, and Kary Främling, editors, *Explainable and Transparent AI and Multi-Agent Systems*, volume 12688, pages 223–240. Springer International Publishing, Cham, 2021. Series Title: Lecture Notes in Computer Science.

A Terminology

Environment (Env): The simulation, in our case, OpenAI GYM simulation. If given an Action it changes its state.

State (s): A state revers to the state of the Environment.

Agent (π): The actor inside an Environment. Makes decisions and takes actions given the state of the environment.

Action (a): A action refers to an action that the agent takes given a state.

Rollout R(π, Env): Running an environment given an agent from beginning to end. Returns a trajectory cumulative reward.

Trajectory (τ): This is a recording of a rollout containing the states and actions. List of state-action pairs.

State-action pair (s, a): This is a combination of a state and the action the agent would decide on given the state.

Action mapping: The action an agent takes given a state. The state can be from a rollout given an agent different from the agent making the decision.

Behavior: Is the resulting trajectory from the action mappings.

Reward (r): A numerical resemblance of the goodness of a state an Agent in an Environment.

Cumulative Reward: The cumulative reward at the end of a rollout. In the environments mentioned in this paper the total number of actions taken during a rollout.

Expert (π_E): A human or ML model which performs its tasks perfectly. The behaviour that we want to imitate. Makes decisions in an Agent.

Surrogate (DT or π_θ): The model we get from imitating the Expert. Makes decisions in an Agent.

Generator (DT or π_θ): The model in GAIL which is being trained to be like the expert, at the end of all r will become the surrogate.

Iteration: Is an iteration within GAIL, each iteration a new list of state-action pairs is used to train the Generator.

B Algorithms

Algorithm 2: GAIL [7]

Input: Expert trajectories τ_E, initial policy θ_0, and discriminator parameters w_0
Output: Surrogate model

for *iteration: i = 0, 1, 2, ...* **do**
- Sample trajectories: τ_i from Generator π_{θ_i}
- Update the Discriminator parameters from w_i to w_{i+1} with the gradient:
  $\hat{E}_{\tau_i}\left[\nabla_w \log\left(D_w(s,a)\right)\right] + \hat{E}_{\tau_E}\left[\nabla_w \log\left(1 - D_w(s,a)\right)\right]$
- Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with $\hat{E}_{\tau_i}\left[\nabla_\theta \log \pi_\theta(a \mid s) Q(s,a)\right] - \lambda \nabla_\theta H\left(\pi_\theta\right)$ where $Q(\bar{s},\bar{a}) = \hat{E}_{\tau_i}\left[\log\left(D_{w_{i+1}}(s,a)\right) \mid s_0 = \bar{s}, a_0 = \bar{a}\right]$

Algorithm 3: Behavioural Cloning

Input: N Expert trajectories τ_E
Output: Decision Tree π_{DT}

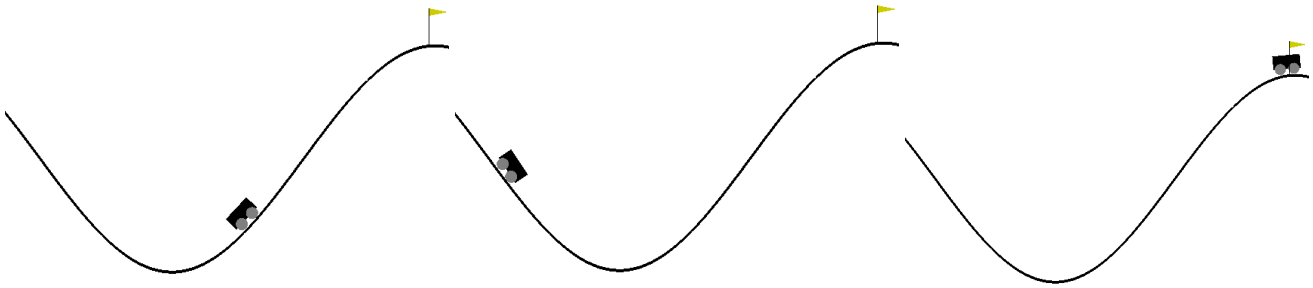
Function *Train BC*:
- Initialize Decision Tree.
- Train π_{DT} on τ_E
- Return: π_{DT}

C Information related to the experts

Table 7 and Figures 5, 6, & 7 are related to the experts.

Table 7: Expert TensorForce Settings

Environment	Agent	Memory size	Layer Types	Layer Sizes
MountainCar-v0	DQN	100000	[Dense, Dense, Dense]	[64, 64, 64]
CartPole-v1	DQN	50000	[Dense, Dense]	[64, 64]
Acrobot-v1	DQN	200000	[Dense, Dense, Dense]	[64, 64, 64]

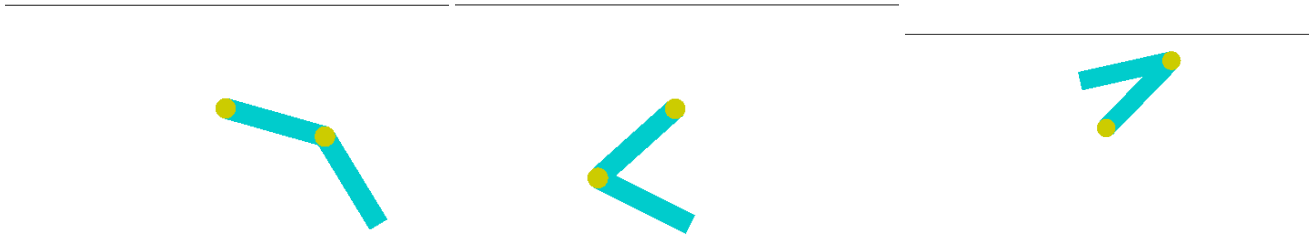


(a) Move right (b) Left 60% / 80% (c) Right to finish

Figure 5: Screenshots of MountainCar Expert rollout.

(a) Fall to the left (b) Left edge of map (c) Fall to the right

Figure 6: Screenshots of CartPole Expert rollout.



(a) Momentum right (b) Momentum left (c) Almost finished

Figure 7: Screenshots of Acrobot Expert rollout.

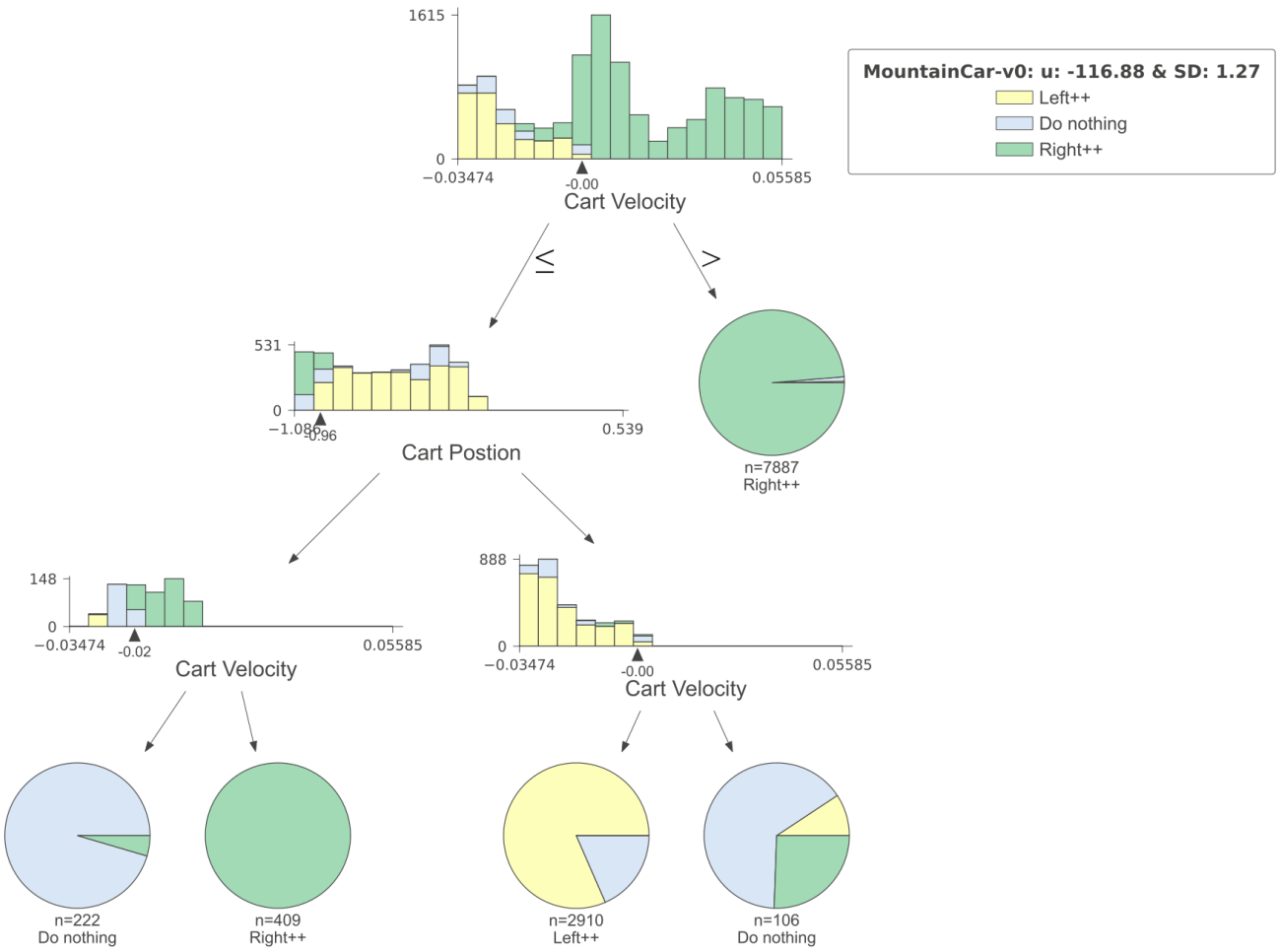


Figure 8: MountainCar Reward: -116.88, SD: 1.267, & Fidelity: 0.952

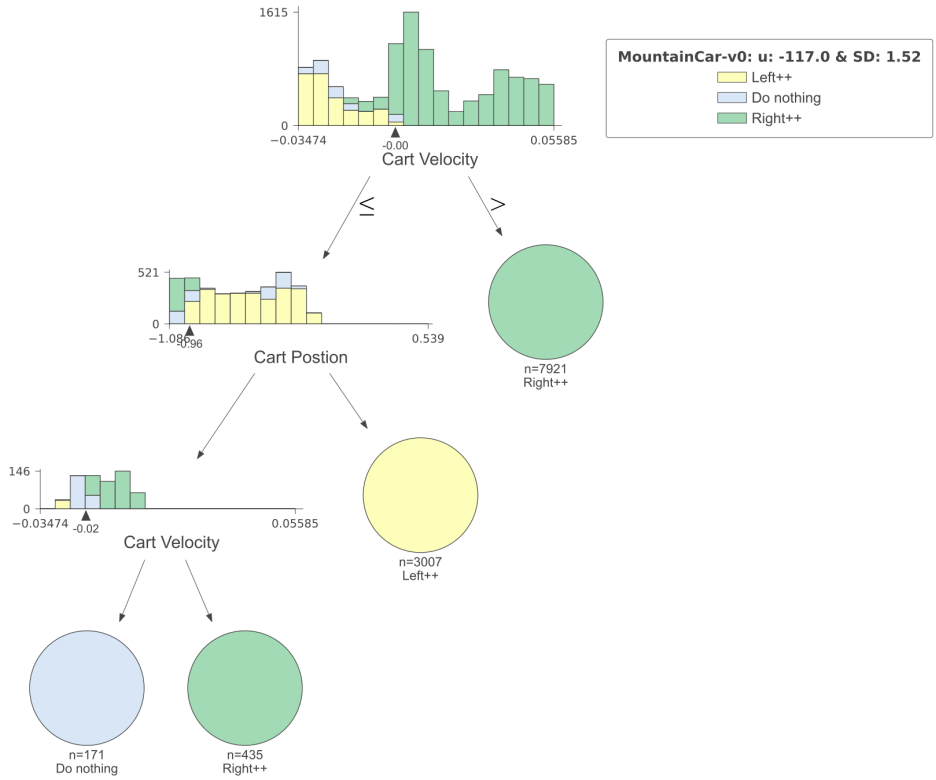


Figure 9: MountainCar Reward: -117.0, SD: 2.112, & Fidelity: 0.939

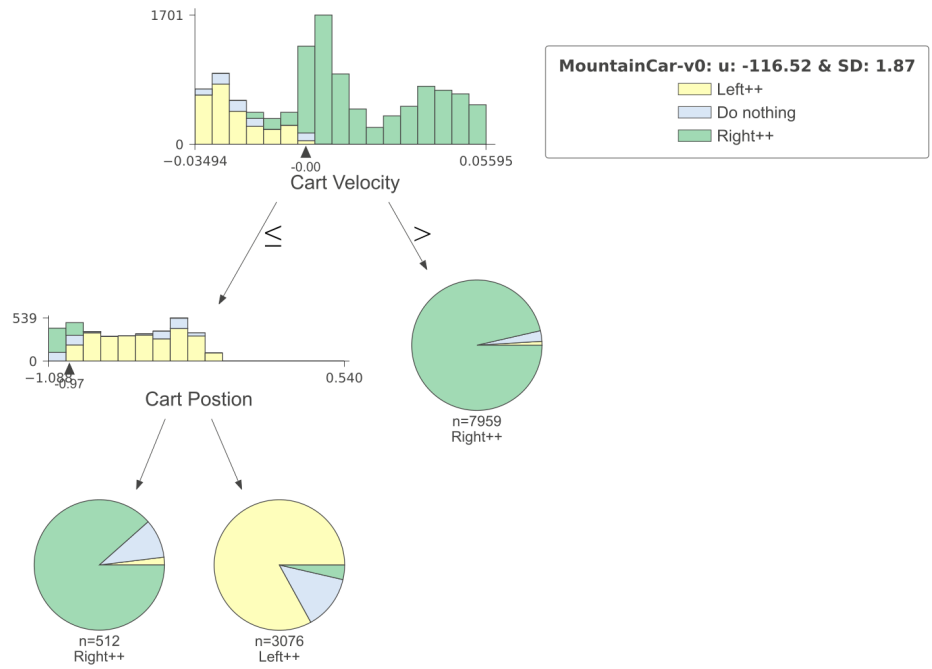


Figure 10: MountainCar Reward: -116.52, SD: 1873, & Fidelity: 0.939

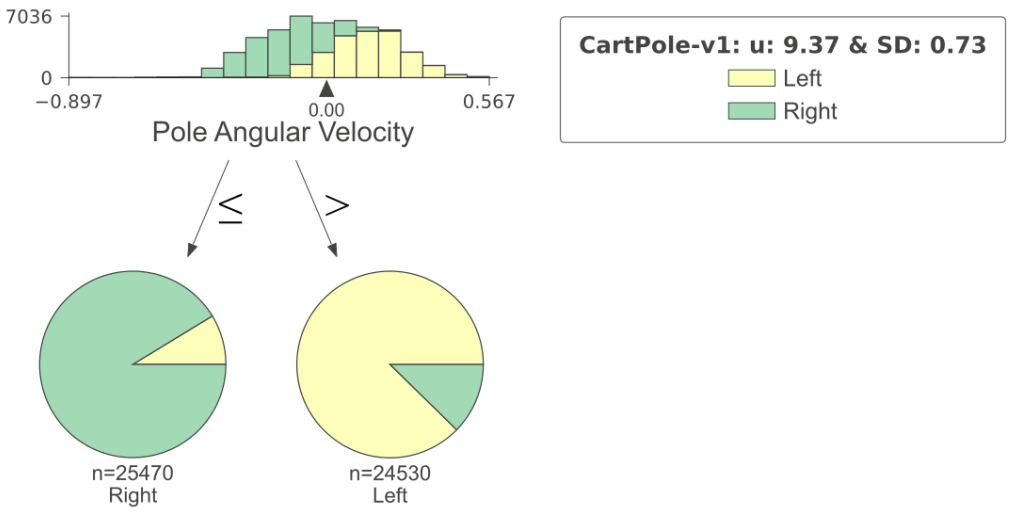


Figure 11: CartPole Reward: 9.37, SD: 0.73, & Fidelity: 0.864

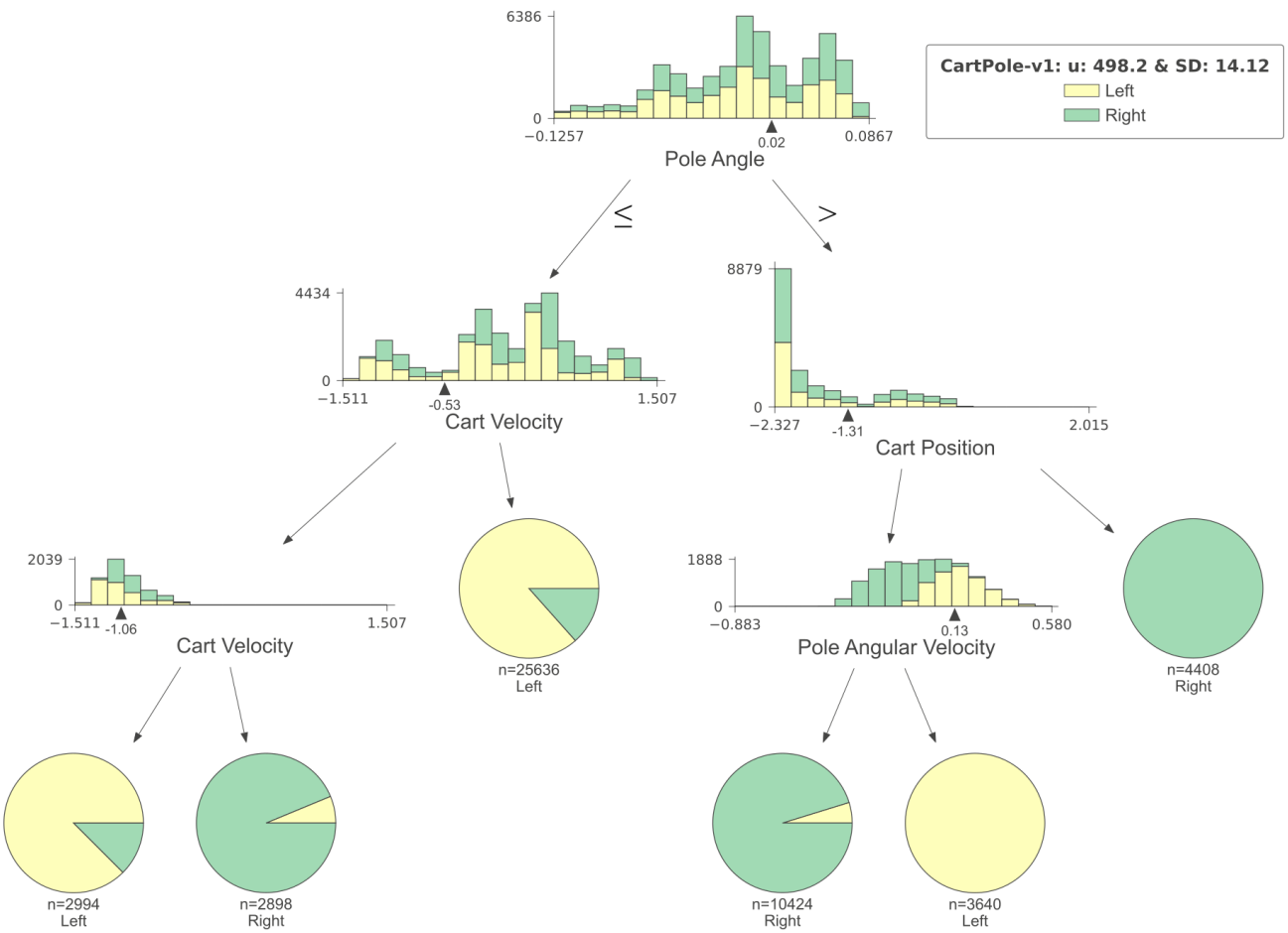


Figure 12: CartPole Reward: 498.2, SD: 14.12, & Fidelity: 0.617

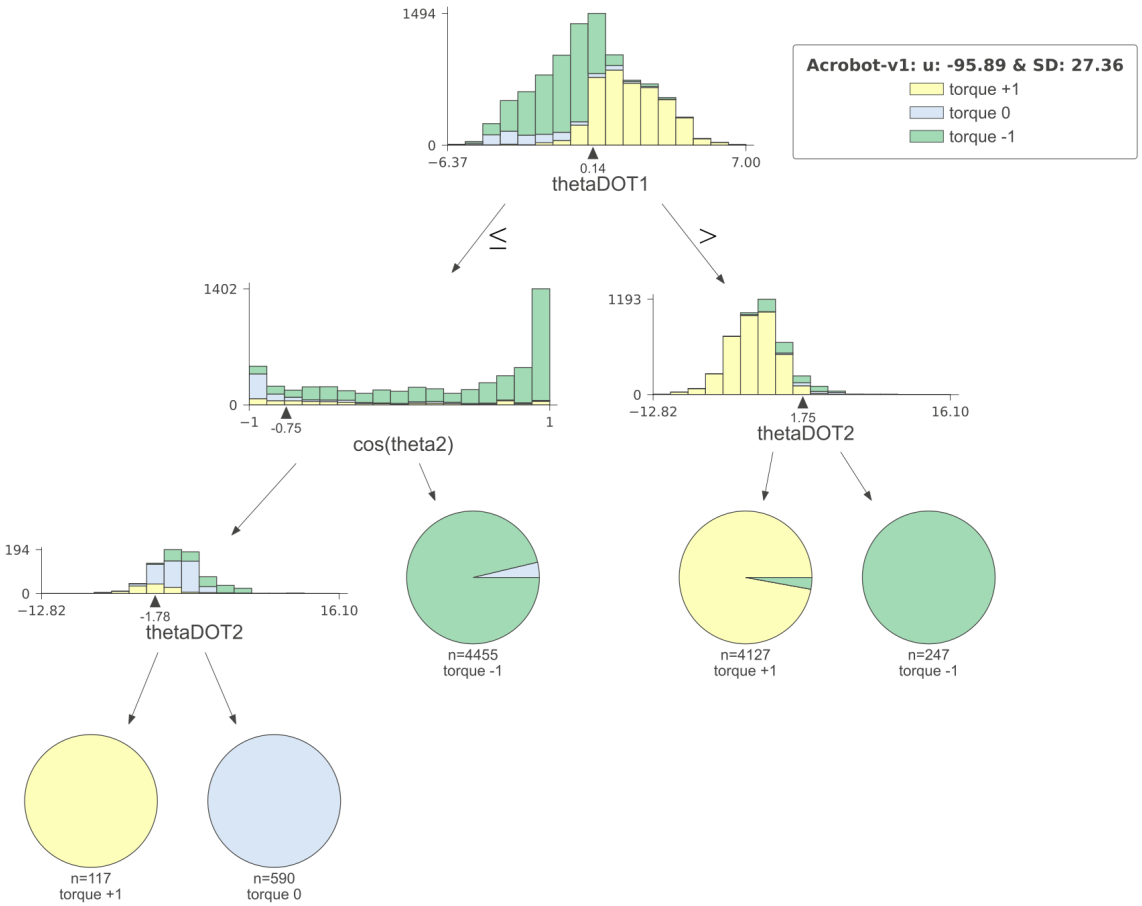


Figure 13: Acrobot Reward: -95.89, SD: 27.36, & Fidelity: 0.847

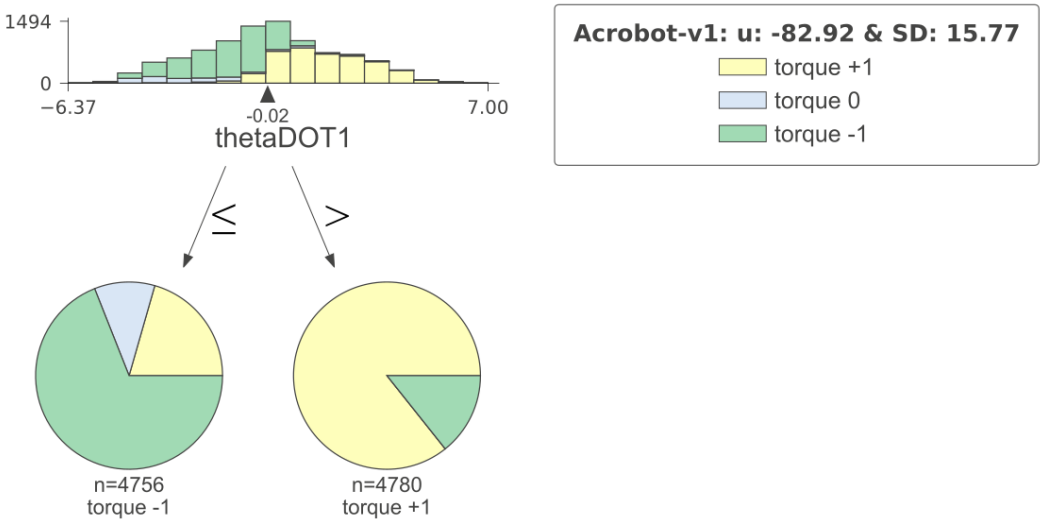


Figure 14: Acrobot Reward: -82.92, SD: 15.77, & Fidelity: 0.802

D GAIL Result Decision Trees

Figures 8 to 14 are made by GAIL.

E GAIL Results Tables

Column Names:

Index: The index in the table, for reference.

A: Environment Name

B: Decision Tree Depth, for both BC and GAIL

C: $ownGeneratorTrajectories$: Generate own Generator Trajectories (not using expert states with generator action mappings)

D: $hasAccessToExpert$: Use the Expert to create new trajectories and replace expert actions with actions of the Generator, and add this list to the Expert Trajectories.

E: $sampleWithC$: Use the sampling technique described in Viper. Re-sample the training data weighted with the loss function: the difference between maximum probability of an action given a state and minimum probability of an action given a state, by the Expert.

F: $discriminateWithC$ Add this loss function to the state-action pairs to train the discriminator.

G: The Total Number of Expert State-Action pairs at the end of running GAIL. This will increase if $hasAccessToExpert$ is True.

H: Expert Reward

I: Expert Standard Deviation

J: BC Reward

K: BC Standard Deviation

L: GAIL Reward

M: GAIL Standard Deviation

N: BC Fidelity

O: GAIL Fidelity

Table 8: MountainCar Sorted Results - 2 Full Runs

Index	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	MountainCar-v0	3	TRUE	TRUE	FALSE	TRUE	1369	-114	0	-121.9	15.323	-116.88	1.267	0.938	0.952
2	MountainCar-v0	3	TRUE	TRUE	FALSE	TRUE	1383	-124	0	-132.32	29.229	-116.7	1.825	0.928	0.949
3	MountainCar-v0	3	TRUE	FALSE	FALSE	TRUE	114	-114	0	-121.9	15.323	-117	1.523	0.938	0.947
4	MountainCar-v0	2	TRUE	TRUE	FALSE	FALSE	1367	-124	0	-129.71	27.427	-116.52	1.873	0.922	0.939
5	MountainCar-v0	3	TRUE	TRUE	FALSE	FALSE	1396	-124	0	-132.32	29.229	-117	2.112	0.928	0.939
6	MountainCar-v0	3	FALSE	TRUE	FALSE	FALSE	1374	-124	0	-132.32	29.229	-116.35	1.664	0.928	0.937
7	MountainCar-v0	2	TRUE	TRUE	FALSE	TRUE	1470	-124	0	-129.71	27.427	-116.41	2.122	0.922	0.936
8	MountainCar-v0	2	FALSE	TRUE	FALSE	TRUE	1373	-124	0	-129.71	27.427	-116.53	2.012	0.922	0.936
9	MountainCar-v0	2	FALSE	TRUE	FALSE	FALSE	1486	-114	0	-122.89	17.993	-116.31	1.573	0.932	0.935
10	MountainCar-v0	3	TRUE	FALSE	FALSE	FALSE	114	-114	0	-121.9	15.323	-121.75	14.218	0.938	0.933
11	MountainCar-v0	2	TRUE	TRUE	FALSE	TRUE	1352	-114	0	-122.89	17.993	-115.69	1.412	0.932	0.932
12	MountainCar-v0	2	FALSE	TRUE	FALSE	FALSE	1385	-124	0	-129.71	27.427	-116.98	2.839	0.922	0.932
13	MountainCar-v0	2	FALSE	TRUE	FALSE	TRUE	1424	-114	0	-122.89	17.993	-117.53	3.351	0.932	0.932
14	MountainCar-v0	3	FALSE	FALSE	FALSE	FALSE	124	-124	0	-132.32	29.229	-122.06	17.056	0.928	0.932
15	MountainCar-v0	2	TRUE	FALSE	FALSE	FALSE	114	-114	0	-122.89	17.993	-122.31	16.286	0.932	0.932
16	MountainCar-v0	2	TRUE	FALSE	FALSE	TRUE	114	-114	0	-122.89	17.993	-123.54	19.586	0.932	0.932
17	MountainCar-v0	3	FALSE	FALSE	FALSE	FALSE	114	-114	0	-121.9	15.323	-124.81	20.996	0.938	0.932
18	MountainCar-v0	3	TRUE	TRUE	FALSE	FALSE	1266	-114	0	-121.9	15.323	-116.56	1.711	0.938	0.931
19	MountainCar-v0	2	FALSE	FALSE	FALSE	TRUE	114	-114	0	-122.89	17.993	-122.1	16.296	0.932	0.93
20	MountainCar-v0	3	FALSE	TRUE	FALSE	FALSE	1361	-114	0	-121.9	15.323	-120.09	13.363	0.938	0.929
21	MountainCar-v0	3	FALSE	TRUE	FALSE	TRUE	545	-114	0	-121.9	15.323	-115.2	0.529	0.938	0.928
22	MountainCar-v0	3	FALSE	FALSE	FALSE	TRUE	124	-124	0	-132.32	29.229	-117.06	2.445	0.928	0.926
23	MountainCar-v0	2	FALSE	FALSE	FALSE	TRUE	124	-124	0	-129.71	27.427	-126.32	23.493	0.922	0.922
24	MountainCar-v0	3	FALSE	FALSE	FALSE	TRUE	114	-114	0	-121.9	15.323	-124.96	19.148	0.938	0.918
25	MountainCar-v0	3	TRUE	TRUE	TRUE	TRUE	1874	-114	0	-121.9	15.323	-118.57	2.783	0.938	0.917
26	MountainCar-v0	2	FALSE	FALSE	FALSE	FALSE	114	-114	0	-122.89	17.993	-126.26	20.514	0.932	0.914
27	MountainCar-v0	1	TRUE	FALSE	TRUE	FALSE	124	-124	0	-128.56	21.316	-128.26	21.677	0.912	0.912
28	MountainCar-v0	1	TRUE	FALSE	TRUE	TRUE	124	-124	0	-128.56	21.316	-128.79	21.446	0.912	0.912
29	MountainCar-v0	3	TRUE	TRUE	TRUE	TRUE	1607	-124	0	-132.32	29.229	-128.84	21.456	0.928	0.912
30	MountainCar-v0	2	TRUE	FALSE	TRUE	TRUE	124	-124	0	-129.71	27.427	-129.3	22.854	0.922	0.912
31	MountainCar-v0	1	FALSE	FALSE	FALSE	TRUE	124	-124	0	-128.56	21.316	-129.66	22.565	0.912	0.912
32	MountainCar-v0	2	TRUE	FALSE	TRUE	FALSE	124	-124	0	-129.71	27.427	-129.82	23.897	0.922	0.912
33	MountainCar-v0	2	FALSE	FALSE	TRUE	FALSE	124	-124	0	-129.71	27.427	-129.98	23.909	0.922	0.912
34	MountainCar-v0	1	FALSE	FALSE	FALSE	FALSE	124	-124	0	-128.56	21.316	-131.02	24.695	0.912	0.912
35	MountainCar-v0	2	FALSE	TRUE	TRUE	TRUE	1483	-124	0	-129.71	27.427	-116.4	1.954	0.922	0.911
36	MountainCar-v0	1	FALSE	TRUE	TRUE	TRUE	1518	-124	0	-128.56	21.316	-119.27	4.259	0.912	0.911
37	MountainCar-v0	1	TRUE	FALSE	FALSE	TRUE	124	-124	0	-128.56	21.316	-120.29	4.073	0.912	0.911
38	MountainCar-v0	1	TRUE	TRUE	TRUE	FALSE	1670	-124	0	-128.56	21.316	-120.4	4.366	0.912	0.911
39	MountainCar-v0	1	TRUE	FALSE	FALSE	FALSE	124	-124	0	-128.56	21.316	-120.47	4.478	0.912	0.911
40	MountainCar-v0	3	FALSE	TRUE	TRUE	FALSE	1577	-124	0	-132.32	29.229	-120.93	4.174	0.928	0.911
41	MountainCar-v0	2	TRUE	TRUE	TRUE	TRUE	1677	-124	0	-129.71	27.427	-121.9	4.446	0.922	0.911
42	MountainCar-v0	2	TRUE	TRUE	TRUE	FALSE	1719	-124	0	-129.71	27.427	-122.31	2.253	0.922	0.911
43	MountainCar-v0	3	TRUE	FALSE	TRUE	FALSE	124	-124	0	-132.32	29.229	-130.88	24.805	0.928	0.911

44	MountainCar-v0	3	FALSE	TRUE	FALSE	TRUE	1328	-124	0	-132.32	29.229	-119.37	9.023	0.928	0.91
45	MountainCar-v0	1	FALSE	TRUE	FALSE	TRUE	1380	-124	0	-128.56	21.316	-119.75	4.038	0.912	0.91
46	MountainCar-v0	1	FALSE	TRUE	FALSE	FALSE	1336	-124	0	-128.56	21.316	-120.04	3.58	0.912	0.91
47	MountainCar-v0	1	FALSE	FALSE	TRUE	TRUE	124	-124	0	-128.56	21.316	-136.46	30.206	0.912	0.91
48	MountainCar-v0	1	FALSE	FALSE	TRUE	FALSE	124	-124	0	-128.56	21.316	-140.9	33.468	0.912	0.91
49	MountainCar-v0	1	TRUE	TRUE	TRUE	TRUE	1577	-124	0	-128.56	21.316	-119.78	3.862	0.912	0.909
50	MountainCar-v0	1	TRUE	TRUE	FALSE	TRUE	1405	-124	0	-128.56	21.316	-119.83	3.829	0.912	0.909
51	MountainCar-v0	2	FALSE	FALSE	FALSE	FALSE	124	-124	0	-129.71	27.427	-128.86	23.92	0.922	0.909
52	MountainCar-v0	3	TRUE	FALSE	FALSE	FALSE	124	-124	0	-132.32	29.229	-130.37	27.215	0.928	0.909
53	MountainCar-v0	3	FALSE	FALSE	TRUE	FALSE	124	-124	0	-132.32	29.229	-145.29	35.184	0.928	0.909
54	MountainCar-v0	2	FALSE	FALSE	TRUE	TRUE	124	-124	0	-129.71	27.427	-147.64	36.169	0.922	0.909
55	MountainCar-v0	3	TRUE	FALSE	TRUE	TRUE	124	-124	0	-132.32	29.229	-149.68	36.399	0.928	0.909
56	MountainCar-v0	2	TRUE	FALSE	FALSE	FALSE	124	-124	0	-129.71	27.427	-117.88	3.269	0.922	0.908
57	MountainCar-v0	2	TRUE	FALSE	FALSE	TRUE	124	-124	0	-129.71	27.427	-118.54	3.984	0.922	0.907
58	MountainCar-v0	3	TRUE	TRUE	TRUE	FALSE	1476	-124	0	-132.32	29.229	-118.87	3.825	0.928	0.907
59	MountainCar-v0	1	TRUE	TRUE	FALSE	FALSE	1371	-124	0	-128.56	21.316	-119.35	3.825	0.912	0.907
60	MountainCar-v0	1	FALSE	TRUE	TRUE	FALSE	1656	-124	0	-128.56	21.316	-119.28	3.707	0.912	0.906
61	MountainCar-v0	2	FALSE	TRUE	TRUE	TRUE	1557	-114	0	-122.89	17.993	-120.99	5.084	0.932	0.906
62	MountainCar-v0	3	FALSE	FALSE	TRUE	FALSE	114	-114	0	-121.9	15.323	-121.14	5.453	0.938	0.906
63	MountainCar-v0	1	FALSE	FALSE	FALSE	TRUE	114	-114	0	-127.493	20.779	-121.27	4.181	0.905	0.906
64	MountainCar-v0	2	TRUE	TRUE	TRUE	TRUE	1601	-114	0	-122.89	17.993	-121.41	4.233	0.932	0.906
65	MountainCar-v0	2	TRUE	TRUE	TRUE	FALSE	1801	-114	0	-122.89	17.993	-121.45	4.84	0.932	0.906
66	MountainCar-v0	1	FALSE	FALSE	TRUE	FALSE	114	-114	0	-127.493	20.779	-121.54	4.428	0.905	0.906
67	MountainCar-v0	1	FALSE	TRUE	TRUE	TRUE	1649	-114	0	-127.493	20.779	-121.87	4.983	0.905	0.906
68	MountainCar-v0	2	FALSE	FALSE	TRUE	TRUE	114	-114	0	-122.89	17.993	-129.44	22.635	0.932	0.906
69	MountainCar-v0	3	FALSE	TRUE	TRUE	TRUE	1674	-114	0	-121.9	15.323	-131.36	24.496	0.938	0.906
70	MountainCar-v0	2	FALSE	TRUE	TRUE	FALSE	1652	-124	0	-129.71	27.427	-119.36	3.543	0.922	0.905
71	MountainCar-v0	1	TRUE	TRUE	TRUE	TRUE	1650	-114	0	-127.493	20.779	-120.45	3.74	0.905	0.905
72	MountainCar-v0	1	FALSE	FALSE	FALSE	FALSE	114	-114	0	-127.493	20.779	-122.55	8.925	0.905	0.905
73	MountainCar-v0	1	TRUE	FALSE	FALSE	TRUE	114	-114	0	-127.493	20.779	-127.81	20.372	0.905	0.905
74	MountainCar-v0	3	FALSE	FALSE	TRUE	TRUE	114	-114	0	-121.9	15.323	-128.12	21.69	0.938	0.905
75	MountainCar-v0	1	TRUE	TRUE	TRUE	FALSE	1419	-114	0	-127.493	20.779	-119.61	3.834	0.905	0.904
76	MountainCar-v0	1	TRUE	TRUE	FALSE	TRUE	1363	-114	0	-127.493	20.779	-120.05	3.756	0.905	0.904
77	MountainCar-v0	2	FALSE	TRUE	TRUE	FALSE	1479	-114	0	-122.89	17.993	-120.24	3.645	0.932	0.904
78	MountainCar-v0	2	TRUE	FALSE	TRUE	FALSE	114	-114	0	-122.89	17.993	-135.56	28.536	0.932	0.904
79	MountainCar-v0	2	TRUE	FALSE	TRUE	TRUE	114	-114	0	-122.89	17.993	-138.69	30.031	0.932	0.904
80	MountainCar-v0	3	TRUE	FALSE	TRUE	FALSE	114	-114	0	-121.9	15.323	-141.29	32.391	0.938	0.904
81	MountainCar-v0	1	FALSE	TRUE	FALSE	FALSE	1409	-114	0	-127.493	20.779	-119.56	3.514	0.905	0.903
82	MountainCar-v0	3	FALSE	TRUE	TRUE	FALSE	1466	-114	0	-121.9	15.323	-119.79	3.92	0.938	0.903
83	MountainCar-v0	3	TRUE	TRUE	TRUE	FALSE	1495	-114	0	-121.9	15.323	-120.16	3.627	0.938	0.903
84	MountainCar-v0	3	TRUE	FALSE	FALSE	TRUE	124	-124	0	-132.32	29.229	-120.88	23.935	0.928	0.903
85	MountainCar-v0	1	FALSE	TRUE	TRUE	FALSE	1652	-114	0	-127.493	20.779	-119.24	3.798	0.905	0.902
86	MountainCar-v0	1	TRUE	FALSE	TRUE	TRUE	114	-114	0	-127.493	20.779	-149.14	35.88	0.905	0.902
87	MountainCar-v0	3	TRUE	FALSE	TRUE	TRUE	114	-114	0	-121.9	15.323	-154.19	36.756	0.938	0.902
88	MountainCar-v0	2	FALSE	FALSE	TRUE	FALSE	114	-114	0	-122.89	17.993	-159.39	37.556	0.932	0.902
89	MountainCar-v0	1	TRUE	TRUE	FALSE	FALSE	1670	-114	0	-127.493	20.779	-119.64	3.738	0.905	0.901

90	MountainCar-v0	2	TRUE	TRUE	FALSE	FALSE	1606	-114	0	-122.89	17.993	-119.7	3.39	0.932	0.901
91	MountainCar-v0	1	FALSE	FALSE	TRUE	TRUE	114	-114	0	-127.493	20.779	-126.11	33.306	0.905	0.885
92	MountainCar-v0	1	FALSE	TRUE	FALSE	TRUE	1613	-114	0	-127.493	20.779	-132.87	30.872	0.905	0.88
93	MountainCar-v0	1	TRUE	FALSE	FALSE	FALSE	114	-114	0	-127.493	20.779	-134.34	32.416	0.905	0.868
94	MountainCar-v0	3	FALSE	FALSE	TRUE	TRUE	124	-124	0	-132.32	29.229	-147.77	35.471	0.928	0.838
95	MountainCar-v0	3	FALSE	TRUE	TRUE	TRUE	1950	-124	0	-132.32	29.229	-138.46	9.515	0.928	0.816
96	MountainCar-v0	1	TRUE	FALSE	TRUE	FALSE	114	-114	0	-127.493	20.779	-200	0	0.905	0.117

Table 9: CartPole Sorted Results - 2 Full Runs

Index	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	CartPole-v1	1	TRUE	FALSE	FALSE	FALSE	500	500	0	9.31	0.666	9.37	0.73	0.864	0.864
2	CartPole-v1	1	FALSE	FALSE	FALSE	TRUE	500	500	0	9.31	0.666	9.26	0.716	0.864	0.864
3	CartPole-v1	2	TRUE	FALSE	FALSE	TRUE	500	500	0	9.263	0.773	9.2	0.787	0.864	0.864
4	CartPole-v1	3	FALSE	FALSE	TRUE	TRUE	500	500	0	9.217	0.755	9.18	0.779	0.864	0.864
5	CartPole-v1	3	FALSE	FALSE	TRUE	FALSE	500	500	0	9.37	0.73	11.17	4.539	0.889	0.861
6	CartPole-v1	2	TRUE	FALSE	FALSE	FALSE	500	500	0	9.247	0.706	9.35	0.726	0.86	0.861
7	CartPole-v1	1	TRUE	FALSE	FALSE	FALSE	500	500	0	9.24	0.705	9.13	0.688	0.86	0.861
8	CartPole-v1	3	FALSE	FALSE	FALSE	FALSE	500	500	0	9.37	0.73	9.28	0.763	0.889	0.86
9	CartPole-v1	2	FALSE	FALSE	FALSE	FALSE	500	500	0	9.247	0.706	9.22	0.729	0.86	0.86
10	CartPole-v1	1	FALSE	FALSE	FALSE	FALSE	500	500	0	9.24	0.705	9.21	0.725	0.86	0.86
11	CartPole-v1	3	FALSE	FALSE	FALSE	TRUE	500	500	0	9.37	0.73	9.13	0.688	0.889	0.859
12	CartPole-v1	3	FALSE	FALSE	TRUE	TRUE	500	500	0	9.37	0.73	10.96	3.947	0.889	0.857
13	CartPole-v1	1	FALSE	FALSE	TRUE	TRUE	500	500	0	9.24	0.705	9.25	0.817	0.86	0.852
14	CartPole-v1	2	FALSE	FALSE	TRUE	FALSE	500	500	0	9.263	0.773	9.29	0.791	0.864	0.843
15	CartPole-v1	3	FALSE	FALSE	TRUE	FALSE	500	500	0	9.217	0.755	9.49	0.685	0.864	0.842
16	CartPole-v1	3	TRUE	FALSE	TRUE	FALSE	500	500	0	9.217	0.755	9.9	0.412	0.864	0.834
17	CartPole-v1	1	TRUE	FALSE	TRUE	TRUE	500	500	0	9.31	0.666	9.35	0.841	0.864	0.833
18	CartPole-v1	1	FALSE	FALSE	TRUE	TRUE	500	500	0	9.31	0.666	9.36	0.794	0.864	0.832
19	CartPole-v1	1	TRUE	FALSE	TRUE	FALSE	500	500	0	9.31	0.666	9.37	0.716	0.864	0.82
20	CartPole-v1	2	TRUE	FALSE	TRUE	FALSE	500	500	0	9.247	0.706	18.86	5.641	0.86	0.81
21	CartPole-v1	3	FALSE	FALSE	FALSE	TRUE	500	500	0	9.217	0.755	9.24	0.709	0.864	0.81
22	CartPole-v1	3	FALSE	TRUE	TRUE	FALSE	712	500	0	9.217	0.755	28.63	12.169	0.864	0.803
23	CartPole-v1	3	TRUE	FALSE	TRUE	TRUE	500	500	0	9.217	0.755	20.35	5.432	0.864	0.802
24	CartPole-v1	2	FALSE	FALSE	TRUE	TRUE	500	500	0	9.263	0.773	14.38	6.043	0.864	0.801
25	CartPole-v1	1	TRUE	FALSE	TRUE	FALSE	500	500	0	9.24	0.705	9.37	0.73	0.86	0.79
26	CartPole-v1	1	TRUE	FALSE	TRUE	TRUE	500	500	0	9.24	0.705	9.36	0.755	0.86	0.774
27	CartPole-v1	2	FALSE	FALSE	FALSE	FALSE	500	500	0	9.263	0.773	9.33	0.694	0.864	0.765
28	CartPole-v1	3	TRUE	TRUE	FALSE	TRUE	759	500	0	9.217	0.755	93.95	57.967	0.864	0.754
29	CartPole-v1	3	TRUE	TRUE	FALSE	FALSE	1024	500	0	9.217	0.755	101.48	66.116	0.864	0.741
30	CartPole-v1	2	TRUE	TRUE	FALSE	FALSE	807	500	0	9.263	0.773	62.77	53.232	0.864	0.736
31	CartPole-v1	2	FALSE	TRUE	FALSE	TRUE	786	500	0	9.263	0.773	55.62	57.231	0.864	0.736
32	CartPole-v1	2	TRUE	TRUE	FALSE	TRUE	678	500	0	9.263	0.773	103.91	58.5	0.864	0.733
33	CartPole-v1	2	FALSE	TRUE	FALSE	FALSE	924	500	0	9.263	0.773	129.4	71.288	0.864	0.727
34	CartPole-v1	2	TRUE	TRUE	FALSE	FALSE	785	500	0	9.247	0.706	71.58	53.934	0.86	0.724
35	CartPole-v1	3	FALSE	TRUE	FALSE	TRUE	989	500	0	9.217	0.755	307.31	187.514	0.864	0.71
36	CartPole-v1	3	FALSE	TRUE	FALSE	FALSE	998	500	0	9.217	0.755	160.69	63.672	0.864	0.703
37	CartPole-v1	2	TRUE	TRUE	FALSE	TRUE	897	500	0	9.247	0.706	132.41	69.365	0.86	0.703
38	CartPole-v1	3	TRUE	TRUE	FALSE	TRUE	859	500	0	9.37	0.73	126.6	60.965	0.889	0.698
39	CartPole-v1	3	FALSE	TRUE	FALSE	TRUE	1221	500	0	9.37	0.73	139.78	62.084	0.889	0.693
40	CartPole-v1	3	FALSE	TRUE	FALSE	FALSE	898	500	0	9.37	0.73	60.62	54.685	0.889	0.678
41	CartPole-v1	3	TRUE	TRUE	TRUE	TRUE	744	500	0	9.217	0.755	102.97	48.074	0.864	0.651
42	CartPole-v1	2	FALSE	TRUE	FALSE	FALSE	763	500	0	9.247	0.706	96.2	61.71	0.86	0.651
43	CartPole-v1	3	TRUE	TRUE	FALSE	FALSE	1466	500	0	9.37	0.73	309.28	169.249	0.889	0.648

44	CartPole-v1	3	TRUE	TRUE	TRUE	FALSE	1188	500	0	9.217	0.755	261.17	64.364	0.864	0.637
45	CartPole-v1	3	TRUE	TRUE	TRUE	TRUE	1120	500	0	9.37	0.73	498.2	14.116	0.889	0.617
46	CartPole-v1	3	FALSE	TRUE	TRUE	TRUE	1111	500	0	9.217	0.755	410.72	44.652	0.864	0.611
47	CartPole-v1	2	FALSE	FALSE	FALSE	TRUE	500	500	0	9.263	0.773	9.42	0.695	0.864	0.554
48	CartPole-v1	1	FALSE	FALSE	TRUE	FALSE	500	500	0	9.31	0.666	43.75	9.104	0.864	0.551
49	CartPole-v1	1	FALSE	FALSE	TRUE	FALSE	500	500	0	9.24	0.705	43.27	9.229	0.86	0.55
50	CartPole-v1	1	FALSE	TRUE	TRUE	FALSE	727	500	0	9.31	0.666	41.3	7.436	0.864	0.548
51	CartPole-v1	1	TRUE	TRUE	TRUE	TRUE	736	500	0	9.31	0.666	42.25	9.068	0.864	0.546
52	CartPole-v1	1	FALSE	TRUE	TRUE	TRUE	725	500	0	9.31	0.666	41.59	7.561	0.864	0.545
53	CartPole-v1	3	TRUE	FALSE	FALSE	TRUE	500	500	0	9.217	0.755	41.37	8.162	0.864	0.544
54	CartPole-v1	2	FALSE	TRUE	TRUE	TRUE	1387	500	0	9.247	0.706	210.86	14.466	0.86	0.54
55	CartPole-v1	1	FALSE	FALSE	FALSE	FALSE	500	500	0	9.31	0.666	40.65	8.85	0.864	0.54
56	CartPole-v1	1	TRUE	TRUE	TRUE	FALSE	741	500	0	9.31	0.666	42.81	8.222	0.864	0.538
57	CartPole-v1	1	FALSE	FALSE	FALSE	TRUE	500	500	0	9.24	0.705	42.01	9.429	0.86	0.538
58	CartPole-v1	2	FALSE	FALSE	TRUE	TRUE	500	500	0	9.247	0.706	41.56	8.975	0.86	0.538
59	CartPole-v1	2	TRUE	FALSE	TRUE	FALSE	500	500	0	9.263	0.773	41.34	8.687	0.864	0.537
60	CartPole-v1	1	TRUE	TRUE	FALSE	TRUE	592	500	0	9.31	0.666	40.51	7.619	0.864	0.537
61	CartPole-v1	3	FALSE	TRUE	TRUE	FALSE	1010	500	0	9.37	0.73	151.26	27.814	0.889	0.536
62	CartPole-v1	2	TRUE	TRUE	TRUE	FALSE	744	500	0	9.247	0.706	41.66	8.995	0.86	0.536
63	CartPole-v1	1	TRUE	TRUE	TRUE	FALSE	757	500	0	9.24	0.705	42.07	8.1	0.86	0.535
64	CartPole-v1	3	FALSE	TRUE	TRUE	TRUE	1426	500	0	9.37	0.73	354.75	39.62	0.889	0.534
65	CartPole-v1	1	FALSE	TRUE	TRUE	TRUE	759	500	0	9.24	0.705	42.44	7.697	0.86	0.533
66	CartPole-v1	2	FALSE	TRUE	TRUE	FALSE	711	500	0	9.263	0.773	30.44	12.624	0.864	0.53
67	CartPole-v1	2	TRUE	TRUE	TRUE	TRUE	825	500	0	9.247	0.706	131.77	25.902	0.86	0.528
68	CartPole-v1	2	TRUE	FALSE	FALSE	TRUE	500	500	0	9.247	0.706	41.56	7.843	0.86	0.528
69	CartPole-v1	2	TRUE	FALSE	FALSE	FALSE	500	500	0	9.263	0.773	39.7	7.721	0.864	0.528
70	CartPole-v1	3	TRUE	TRUE	TRUE	FALSE	1804	500	0	9.37	0.73	497.64	10.768	0.889	0.515
71	CartPole-v1	2	TRUE	TRUE	TRUE	TRUE	831	500	0	9.263	0.773	210.24	55.044	0.864	0.515
72	CartPole-v1	2	FALSE	TRUE	TRUE	FALSE	1460	500	0	9.247	0.706	498.7	7.383	0.86	0.514
73	CartPole-v1	2	TRUE	TRUE	TRUE	FALSE	686	500	0	9.263	0.773	39.51	18.724	0.864	0.51
74	CartPole-v1	2	FALSE	FALSE	TRUE	FALSE	500	500	0	9.247	0.706	31.79	11.534	0.86	0.51
75	CartPole-v1	2	TRUE	FALSE	TRUE	TRUE	500	500	0	9.247	0.706	28.83	11.387	0.86	0.51
76	CartPole-v1	3	TRUE	FALSE	TRUE	TRUE	500	500	0	9.37	0.73	26.8	13.574	0.889	0.51
77	CartPole-v1	2	FALSE	TRUE	TRUE	TRUE	673	500	0	9.263	0.773	35.62	14.733	0.864	0.509
78	CartPole-v1	1	TRUE	TRUE	FALSE	TRUE	594	500	0	9.24	0.705	28.74	12.73	0.86	0.509
79	CartPole-v1	1	TRUE	TRUE	FALSE	FALSE	597	500	0	9.24	0.705	27.94	11.645	0.86	0.509
80	CartPole-v1	1	FALSE	TRUE	FALSE	TRUE	593	500	0	9.31	0.666	28.08	14.199	0.864	0.508
81	CartPole-v1	1	TRUE	FALSE	FALSE	TRUE	500	500	0	9.24	0.705	9.3	0.768	0.86	0.49
82	CartPole-v1	3	TRUE	FALSE	FALSE	FALSE	500	500	0	9.217	0.755	42.67	21.966	0.864	0.448
83	CartPole-v1	2	FALSE	FALSE	FALSE	TRUE	500	500	0	9.247	0.706	41.02	18.455	0.86	0.448
84	CartPole-v1	1	TRUE	FALSE	FALSE	TRUE	500	500	0	9.31	0.666	43.78	21.587	0.864	0.447
85	CartPole-v1	1	TRUE	TRUE	FALSE	FALSE	592	500	0	9.31	0.666	40	13.548	0.864	0.446
86	CartPole-v1	3	FALSE	FALSE	FALSE	FALSE	500	500	0	9.217	0.755	39.88	15.377	0.864	0.445
87	CartPole-v1	3	TRUE	FALSE	FALSE	TRUE	500	500	0	9.37	0.73	39.2	15.768	0.889	0.445
88	CartPole-v1	1	FALSE	TRUE	TRUE	FALSE	749	500	0	9.24	0.705	34.28	4.633	0.86	0.429
89	CartPole-v1	3	TRUE	FALSE	TRUE	FALSE	500	500	0	9.37	0.73	152.98	32.617	0.889	0.148

90	CartPole-v1	1	TRUE	TRUE	TRUE	TRUE	791	500	0	9.24	0.705	159.21	33.667	0.86	0.146
91	CartPole-v1	1	FALSE	TRUE	FALSE	FALSE	594	500	0	9.31	0.666	162.44	35.828	0.864	0.144
92	CartPole-v1	1	FALSE	TRUE	FALSE	FALSE	595	500	0	9.24	0.705	189.46	51.038	0.86	0.142
93	CartPole-v1	1	FALSE	TRUE	FALSE	TRUE	591	500	0	9.24	0.705	189.21	45.342	0.86	0.141
94	CartPole-v1	3	TRUE	FALSE	FALSE	FALSE	500	500	0	9.37	0.73	178.33	47.208	0.889	0.141
95	CartPole-v1	2	FALSE	TRUE	FALSE	TRUE	787	500	0	9.247	0.706	198.49	46.719	0.86	0.14
96	CartPole-v1	2	TRUE	FALSE	TRUE	TRUE	500	500	0	9.263	0.773	203.11	43.316	0.864	0.137

Table 10: Acrobot Sorted Results - 2 Full Runs

Index	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Acrobot-v1	3	TRUE	FALSE	FALSE	TRUE	79	-78	0	-102.22	31.931	-95.89	27.36	0.843	0.847
2	Acrobot-v1	2	TRUE	FALSE	FALSE	FALSE	79	-78	0	-92.363	29.359	-92.56	29.304	0.846	0.846
3	Acrobot-v1	2	TRUE	FALSE	FALSE	TRUE	79	-78	0	-92.363	29.359	-95.08	30.989	0.846	0.846
4	Acrobot-v1	3	FALSE	FALSE	FALSE	TRUE	79	-78	0	-102.22	31.931	-96.24	44.92	0.843	0.844
5	Acrobot-v1	3	TRUE	FALSE	FALSE	FALSE	79	-78	0	-102.22	31.931	-102.8	30.25	0.843	0.842
6	Acrobot-v1	3	FALSE	FALSE	FALSE	FALSE	88	-87	0	-86.53	19.775	-88.75	21.843	0.83	0.84
7	Acrobot-v1	2	FALSE	TRUE	TRUE	TRUE	3406	-87	0	-88.57	25.592	-103.27	79.092	0.833	0.839
8	Acrobot-v1	2	FALSE	FALSE	FALSE	TRUE	79	-78	0	-92.363	29.359	-95.31	24.636	0.846	0.835
9	Acrobot-v1	2	FALSE	FALSE	FALSE	TRUE	88	-87	0	-88.57	25.592	-93.42	49.799	0.833	0.834
10	Acrobot-v1	3	FALSE	FALSE	FALSE	TRUE	88	-87	0	-86.53	19.775	-88.88	24.575	0.83	0.831
11	Acrobot-v1	3	TRUE	TRUE	TRUE	TRUE	1910	-87	0	-86.53	19.775	-92.24	36.061	0.83	0.831
12	Acrobot-v1	2	TRUE	FALSE	FALSE	FALSE	88	-87	0	-88.57	25.592	-343.46	196.288	0.833	0.831
13	Acrobot-v1	3	TRUE	FALSE	FALSE	FALSE	88	-87	0	-86.53	19.775	-89.06	19.831	0.83	0.83
14	Acrobot-v1	3	FALSE	TRUE	FALSE	TRUE	444	-78	0	-102.22	31.931	-93.9	16.856	0.843	0.826
15	Acrobot-v1	3	FALSE	FALSE	TRUE	FALSE	79	-78	0	-102.22	31.931	-96.26	21.011	0.843	0.825
16	Acrobot-v1	2	FALSE	FALSE	FALSE	FALSE	88	-87	0	-88.57	25.592	-91.35	18.709	0.833	0.824
17	Acrobot-v1	1	FALSE	FALSE	FALSE	FALSE	88	-87	0	-92.987	22.21	-109.55	81.349	0.818	0.824
18	Acrobot-v1	3	FALSE	TRUE	TRUE	TRUE	2773	-78	0	-102.22	31.931	-129.69	98.345	0.843	0.823
19	Acrobot-v1	3	FALSE	TRUE	FALSE	FALSE	1636	-87	0	-86.53	19.775	-95.07	50.264	0.83	0.82
20	Acrobot-v1	3	TRUE	TRUE	FALSE	FALSE	2363	-87	0	-86.53	19.775	-95.68	37.138	0.83	0.82
21	Acrobot-v1	1	TRUE	TRUE	TRUE	FALSE	3464	-87	0	-92.987	22.21	-199.64	176.498	0.818	0.82
22	Acrobot-v1	2	TRUE	FALSE	TRUE	TRUE	88	-87	0	-88.57	25.592	-90.92	28.922	0.833	0.818
23	Acrobot-v1	1	TRUE	FALSE	FALSE	FALSE	88	-87	0	-92.987	22.21	-92	18.326	0.818	0.818
24	Acrobot-v1	1	FALSE	FALSE	FALSE	TRUE	88	-87	0	-92.987	22.21	-94.56	25.239	0.818	0.818
25	Acrobot-v1	1	FALSE	TRUE	TRUE	TRUE	1921	-87	0	-92.987	22.21	-273.29	195.698	0.818	0.818
26	Acrobot-v1	2	FALSE	TRUE	FALSE	TRUE	1239	-78	0	-92.363	29.359	-102.84	50.719	0.846	0.817
27	Acrobot-v1	2	FALSE	TRUE	FALSE	FALSE	1643	-78	0	-92.363	29.359	-88.99	19.673	0.846	0.816
28	Acrobot-v1	1	FALSE	FALSE	FALSE	TRUE	79	-78	0	-92.427	21.99	-91.82	19.781	0.816	0.816
29	Acrobot-v1	1	TRUE	FALSE	FALSE	FALSE	79	-78	0	-92.427	21.99	-92.56	28.019	0.816	0.816
30	Acrobot-v1	1	FALSE	TRUE	FALSE	TRUE	1071	-78	0	-92.427	21.99	-93.63	19.486	0.816	0.816
31	Acrobot-v1	1	TRUE	FALSE	FALSE	TRUE	79	-78	0	-92.427	21.99	-95.35	28.653	0.816	0.816
32	Acrobot-v1	1	TRUE	TRUE	FALSE	TRUE	157	-78	0	-92.427	21.99	-95.65	32.816	0.816	0.816
33	Acrobot-v1	1	FALSE	FALSE	FALSE	FALSE	79	-78	0	-92.427	21.99	-97.73	26.357	0.816	0.816
34	Acrobot-v1	3	FALSE	FALSE	FALSE	FALSE	79	-78	0	-102.22	31.931	-88.46	19.464	0.843	0.815
35	Acrobot-v1	1	FALSE	TRUE	FALSE	TRUE	1582	-87	0	-92.987	22.21	-90.25	18.015	0.818	0.814
36	Acrobot-v1	1	TRUE	TRUE	FALSE	TRUE	970	-87	0	-92.987	22.21	-91.8	26.847	0.818	0.814
37	Acrobot-v1	2	FALSE	FALSE	FALSE	FALSE	79	-78	0	-92.363	29.359	-92.09	20.077	0.846	0.812
38	Acrobot-v1	3	TRUE	FALSE	TRUE	FALSE	79	-78	0	-102.22	31.931	-266.98	197.014	0.843	0.811
39	Acrobot-v1	3	TRUE	FALSE	FALSE	TRUE	88	-87	0	-86.53	19.775	-88.33	21.239	0.83	0.81
40	Acrobot-v1	3	FALSE	TRUE	FALSE	FALSE	1090	-78	0	-102.22	31.931	-90.16	17.07	0.843	0.809
41	Acrobot-v1	3	TRUE	FALSE	TRUE	TRUE	88	-87	0	-86.53	19.775	-88.12	28.162	0.83	0.808
42	Acrobot-v1	2	TRUE	FALSE	TRUE	TRUE	79	-78	0	-92.363	29.359	-398.8	171.175	0.846	0.808
43	Acrobot-v1	2	FALSE	FALSE	TRUE	TRUE	79	-78	0	-92.363	29.359	-135.36	109.361	0.846	0.807

44	Acrobot-v1	2	FALSE	TRUE	TRUE	FALSE	965	-87	0	-88.57	25.592	-85.17	20.29	0.833	0.806
45	Acrobot-v1	1	FALSE	TRUE	TRUE	FALSE	4278	-78	0	-92.427	21.99	-95.65	32.311	0.816	0.806
46	Acrobot-v1	2	TRUE	TRUE	TRUE	TRUE	2506	-78	0	-92.363	29.359	-96.55	19.735	0.846	0.805
47	Acrobot-v1	1	FALSE	TRUE	TRUE	FALSE	2576	-87	0	-92.987	22.21	-94.13	32.47	0.818	0.804
48	Acrobot-v1	1	TRUE	TRUE	TRUE	FALSE	2553	-78	0	-92.427	21.99	-114.12	96.671	0.816	0.804
49	Acrobot-v1	3	FALSE	TRUE	TRUE	TRUE	2150	-87	0	-86.53	19.775	-95.96	29.044	0.83	0.803
50	Acrobot-v1	1	TRUE	TRUE	TRUE	TRUE	2413	-87	0	-92.987	22.21	-114.44	85.031	0.818	0.803
51	Acrobot-v1	1	FALSE	TRUE	FALSE	FALSE	585	-78	0	-92.427	21.99	-82.92	15.771	0.816	0.802
52	Acrobot-v1	2	FALSE	TRUE	FALSE	FALSE	689	-87	0	-88.57	25.592	-88.46	22.782	0.833	0.801
53	Acrobot-v1	3	TRUE	TRUE	TRUE	FALSE	1327	-87	0	-86.53	19.775	-88.79	24.6	0.83	0.801
54	Acrobot-v1	1	FALSE	TRUE	TRUE	TRUE	2766	-78	0	-92.427	21.99	-136.31	126.671	0.816	0.801
55	Acrobot-v1	3	TRUE	FALSE	TRUE	TRUE	79	-78	0	-102.22	31.931	-388.75	178.569	0.843	0.801
56	Acrobot-v1	1	TRUE	TRUE	FALSE	FALSE	1747	-78	0	-92.427	21.99	-84.55	16.19	0.816	0.8
57	Acrobot-v1	2	TRUE	TRUE	TRUE	TRUE	2706	-87	0	-88.57	25.592	-86.9	30.712	0.833	0.8
58	Acrobot-v1	3	FALSE	FALSE	TRUE	FALSE	88	-87	0	-86.53	19.775	-113.63	82.544	0.83	0.8
59	Acrobot-v1	2	TRUE	TRUE	FALSE	TRUE	606	-87	0	-88.57	25.592	-82.97	14.929	0.833	0.799
60	Acrobot-v1	1	TRUE	TRUE	FALSE	FALSE	1472	-87	0	-92.987	22.21	-93.15	42.023	0.818	0.796
61	Acrobot-v1	2	TRUE	TRUE	FALSE	FALSE	1849	-87	0	-88.57	25.592	-91.17	50.014	0.833	0.795
62	Acrobot-v1	2	FALSE	TRUE	FALSE	TRUE	757	-87	0	-88.57	25.592	-85.56	16.915	0.833	0.794
63	Acrobot-v1	2	TRUE	FALSE	FALSE	TRUE	88	-87	0	-88.57	25.592	-93.18	51.511	0.833	0.794
64	Acrobot-v1	1	FALSE	TRUE	FALSE	FALSE	1383	-87	0	-92.987	22.21	-87.4	28.862	0.818	0.793
65	Acrobot-v1	2	TRUE	TRUE	FALSE	TRUE	253	-78	0	-92.363	29.359	-97.52	36.108	0.846	0.792
66	Acrobot-v1	2	FALSE	TRUE	TRUE	TRUE	3575	-78	0	-92.363	29.359	-106.63	50.433	0.846	0.792
67	Acrobot-v1	2	TRUE	FALSE	TRUE	FALSE	79	-78	0	-92.363	29.359	-111.6	56.497	0.846	0.792
68	Acrobot-v1	3	TRUE	TRUE	FALSE	TRUE	1497	-87	0	-86.53	19.775	-79.46	18.535	0.83	0.791
69	Acrobot-v1	3	FALSE	TRUE	FALSE	TRUE	1099	-87	0	-86.53	19.775	-92.59	20.876	0.83	0.791
70	Acrobot-v1	3	TRUE	TRUE	TRUE	TRUE	2793	-78	0	-102.22	31.931	-93.55	24.967	0.843	0.79
71	Acrobot-v1	3	FALSE	TRUE	TRUE	FALSE	694	-78	0	-102.22	31.931	-104.42	52.272	0.843	0.786
72	Acrobot-v1	2	FALSE	FALSE	TRUE	FALSE	79	-78	0	-92.363	29.359	-103.5	33.194	0.846	0.785
73	Acrobot-v1	2	FALSE	FALSE	TRUE	FALSE	88	-87	0	-88.57	25.592	-101.48	53.732	0.833	0.783
74	Acrobot-v1	3	TRUE	TRUE	TRUE	FALSE	1091	-78	0	-102.22	31.931	-91.34	17.467	0.843	0.777
75	Acrobot-v1	1	FALSE	FALSE	TRUE	FALSE	88	-87	0	-92.987	22.21	-111.19	80.777	0.818	0.777
76	Acrobot-v1	3	TRUE	TRUE	FALSE	FALSE	2046	-78	0	-102.22	31.931	-101.13	24.231	0.843	0.776
77	Acrobot-v1	3	TRUE	TRUE	FALSE	TRUE	1539	-78	0	-102.22	31.931	-85.3	17.46	0.843	0.762
78	Acrobot-v1	3	FALSE	FALSE	TRUE	TRUE	79	-78	0	-102.22	31.931	-150.86	130.435	0.843	0.759
79	Acrobot-v1	1	TRUE	FALSE	TRUE	TRUE	88	-87	0	-92.987	22.21	-239.47	188.777	0.818	0.759
80	Acrobot-v1	1	FALSE	FALSE	TRUE	TRUE	88	-87	0	-92.987	22.21	-139.03	124.208	0.818	0.755
81	Acrobot-v1	1	TRUE	FALSE	FALSE	TRUE	88	-87	0	-92.987	22.21	-301.9	199.786	0.818	0.755
82	Acrobot-v1	3	FALSE	TRUE	TRUE	FALSE	1272	-87	0	-86.53	19.775	-95.41	14.91	0.83	0.743
83	Acrobot-v1	3	TRUE	FALSE	TRUE	FALSE	88	-87	0	-86.53	19.775	-105.73	37.972	0.83	0.741
84	Acrobot-v1	2	TRUE	FALSE	TRUE	FALSE	88	-87	0	-88.57	25.592	-95.88	19.07	0.833	0.737
85	Acrobot-v1	2	FALSE	FALSE	TRUE	TRUE	88	-87	0	-88.57	25.592	-117.35	98.757	0.833	0.722
86	Acrobot-v1	2	TRUE	TRUE	FALSE	FALSE	1492	-78	0	-92.363	29.359	-103.42	20.322	0.846	0.695
87	Acrobot-v1	3	FALSE	FALSE	TRUE	TRUE	88	-87	0	-86.53	19.775	-117.11	48.264	0.83	0.686
88	Acrobot-v1	2	FALSE	TRUE	TRUE	FALSE	2877	-78	0	-92.363	29.359	-136.92	26.069	0.846	0.672
89	Acrobot-v1	2	TRUE	TRUE	TRUE	FALSE	2857	-87	0	-88.57	25.592	-137.19	28.175	0.833	0.657

90	Acrobot-v1	2	TRUE	TRUE	TRUE	FALSE	3194	-78	0	-92.363	29.359	-137.08	30.122	0.846	0.652
91	Acrobot-v1	1	TRUE	TRUE	TRUE	TRUE	4228	-78	0	-92.427	21.99	-270.39	44.425	0.816	0.521
92	Acrobot-v1	1	TRUE	FALSE	TRUE	TRUE	79	-78	0	-92.427	21.99	-160.83	41.458	0.816	0.449
93	Acrobot-v1	1	TRUE	FALSE	TRUE	FALSE	88	-87	0	-92.987	22.21	-161.33	47.17	0.818	0.446
94	Acrobot-v1	1	FALSE	FALSE	TRUE	TRUE	79	-78	0	-92.427	21.99	-500	0	0.816	0.206
95	Acrobot-v1	1	TRUE	FALSE	TRUE	FALSE	79	-78	0	-92.427	21.99	-500	0	0.816	0.106
96	Acrobot-v1	1	FALSE	FALSE	TRUE	FALSE	79	-78	0	-92.427	21.99	-500	0	0.816	0.101