USING LEARNING FROM DEMONSTRATIONS TO WORK WITH SUBOPTIMAL DEMONSTRATIONS

Bryan Cochran*
School of Mechanical Engineering
Georgia Institute of Technology
Georgia, GA 30332
bcochran3@gatech.edu

Yosuke Yajima*
Institute for Robotics and Intelligent Machines
Georgia Institute of Technology
Georgia, GA 30332
yyajima@gatech.edu

Zulfiqar Zadi*
School of Mechanical Engineering
Georgia Institute of Technology
Georgia, GA 30332
zzaidi8@gatech.edu
* Equal Contribution

August 2, 2021

ABSTRACT

Reinforcement learning often faces a challenge that the real-world application may contain data impurity and sub-optimal demonstration due to non-expert demonstrators. These data impurity can limit the ability of the agent to learn from expert demonstrations. Additionally, most learning from demonstration (LfD) algorithms assume that the demonstration is optimal, and sub-optimal demonstration can decrease the performance of the LfD algorithms. While several research focus on how to train agent and learn from sub-optimal demonstration, most work does not consider how to retrain agent with sub-optimal demonstrations. In this work, we analyzed how human supervision can improve pre-trained agent that is trained with sub-optimal demonstrations using Convergent Actor-Critic by Humans (COACH). In the first step, we used two behavior cloning agent to learn from both optimal and sub-optimal demonstrations. Then we analyzed how the COACH can improve the agent with sub-optimal demonstrations compared to the optimal agent. Furthermore, we also study the effect of heterogeneity caused by different human supervisions. Our result shows that retraining agent improves driving scores by an average of 95 % compared to suboptimal behavior cloning agent and achieves an average of 108 % of optimal behavior cloning agent score. This work can show a case study that how retraining agent can improve the performance and detailed analysis of heterogeneity effects.

1 Introduction

As machine learning techniques continue to develop, the testing environment of these systems are starting to transition out of the laboratory. The majority of LfD algorithms rely on the assumption that the expert demonstrations are optimal. This assumption however does not reflect the real-world applications where the human experts may contain bias and data impurity that creates a sub-optimal demonstration. Sub-optimal demonstrations can cause most LfD method to limit their agent from learning from demonstrations. Additionally, the variety of demonstrators that are now training these algorithms coupled with the difficulty and complexity of tasks increases the probability of heterogeneity amongst demonstration sets. Heterogeneity occurs when trajectories are uniquely different but still achieve the same goal. In complex tasks, it is often very difficult to differentiate between suboptimal and heterogeneous demonstrations.

Current mitigation techniques involve manually filtering the demonstrations or by putting additional controls in how the demonstrations are made. These techniques work well in a controlled environment such as a laboratory but do not work well out in the real-world. While several LfD method tries to optimize their learning approach using either Inverse reinforcement learning or other approaches, none of the approaches consider fixing the agent that is trained with sub-optimal demonstrations. In this work, we propose a novel framework that retrains an agent that is previously trained with sub-optimal demonstration. Using our modified COACH method and pre-trained behavior cloning agent, we show that the pre-trained agent can improve their learning together with COACH. In the next sections, we will discuss our pre-trained agent using behavior cloning model, modified coach method, and results of our performance against base line model. The key contributions of our work include the following items.

- We developed a framework using COACH that retrain the agent using human supervision.
- We show our result and performance of retrained agents against agent that is trained with optimal demonstration using behavior cloning agent.
- We analyze the correlation between the heterogeneity in retraining our agent based on different human experts.

2 Related Works

Suboptimal demonstration with reinforcement learning Several bodies of research focus on inverse reinforcement learning to learn from sub-optimal demonstrations. The inverse reinforcement learning aims for the agent to learn from demonstrations without reward signals. T-REX[1] is one of examples that uses IRL with rank demonstrations. The benefit of this algorithm are robust to data impurity, no human supervision during policy learning, and no action labels required. Additionally, this approach solves typical IRL issues that the agent cannot learn better than demonstrators and IRL is difficult to scale to complex problems. Chen et al. developed a novel framework using AIRL to learn from sub-optimal demonstration by synthesizing optimality parameterized data and training the reward function. Their approach uses a self-supervised reward regression (SSRR) to learn from noise and performance relationship. Other approaches include a relative entropy Q learning method [2] to learn from sub-optimal demonstrations. However, these approaches do not focus on fixing the agent trained with sub-optimal demonstrations. In our paper, we will discuss how to retrain the agent trained with suboptimal demonstrations.

Reinforcement learning with human feedback In the field of reinforcement learning, several methods incorporate a human feedback to teach how the agent is performing during the training phase. Teaching an Agent Manually via Evaluative reinforcement (TAMER) [3] is one approach that relies human reward feedback and learns a predicted model of human reward. This method allows user to input their reward given the action and state, and the agent chooses the action that has a highest reward out of possible state-action pairs. Celemin et al. develops a Corrective Advice Communicated by Humans (COACH) [4] that relies on non-expert humans to update a policy using corrective feedback and a binary signal. Compared to TAMER, this approach adjusts the amount of human feedback during the training phase and takes a consideration of past feedback. Additionally, the COACH shows a strong performance over the TAMER approach and this method is easy to use and compatible with continuous space actions. Besides the two approaches, Convergent Actor-Critic by Humans (COACH) [5] is another approach that agent learns from policy dependent human feedback. In this work, we are using the COACH method that allows the behavior cloning agent to learn from the non-expert human's feedback.

3 Method

We implement a strategy that is intended on retraining an agent that is already trained for a performance driving task. The retraining agent is intended to reproduce optimal results regardless of the optimally of the original demonstrations used to train the initial agent. Our overall framework is visualized in Figure 1.

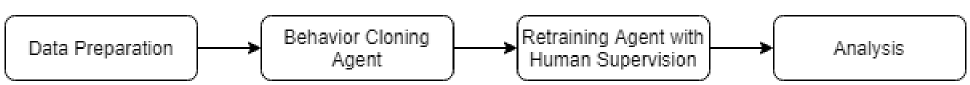

Figure 1: Overall framework

3.1 Game Environment

We developed and tested our framework using a modified Open AI Gym Box 2D environment [6]. It is a top-down racing environment. State consists of 96x96 RGB pixels and the action space is 3x1 vector. The 3x1 action vector, action label “a”, corresponds to steering, throttle, and braking actions. The action labeled as "0" corresponds to the steering actions and ranges from -1 to 1, the action labeled as "1” corresponds to the throttle action and ranges from 0 to 1, and the action labeled as "2" corresponds to the braking action and also ranges from 0 to 1.

3.2 Providing Demonstrations

The environment was modified extensively to allow for demonstrations to be recorded, to facilitate the training of the driving performance agent, and to make the simulation easier to drive for demonstrators. The environment modifications were outlined [7]. This publication was very useful in setting both the environment as well as the driving behavioral cloning agent.

To extract the demonstrations from the expert, a script was written that establishes functions to map the actions that correspond to the press and release of the four arrow keys as well as the space key. These five user inputs map to the 3x1 action array. Another function details the directory where the demonstrations will be stored as well as the format in which the data will be stored in. The final function defines how the data from the demonstrations are stored and then written to the output file.

The main body of the script calls the Box2D Car Racing environment and defines the initial states of the environment. The initial states select which driving track is used during the simulation as well as initializing the rewards. Additionally, the body of the script outlines a reset procedure so the demonstrator can restart an individual demonstration and have the data from that demonstration disregarded.

The data was gathered from 3 different experts. Each expert was asked to provide a dataset of demonstrations that they would consider optimal, and a dataset of demonstrations that they would consider suboptimal. For a single dataset, the experts were asked to lap the racing track about 10-15 times.

3.3 Behavior Cloning Agent

The driving agent was trained using behavior cloning due to Behavior Cloning’s sensitivity to demonstration variation. Some prior work was done on creating an agent that learns to drive around the Box2D Car Racing environment. The architecture of the agent used in this investigation was inspired from the work of Y. Zhang et. al [8][7]. For training the behavior cloning agent, the pipeline shown in Figure 2 was used:

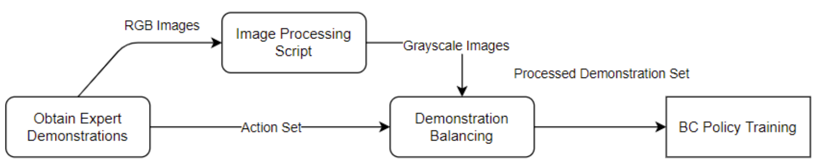

Figure 2: Flowchart for behavior cloning agent [5]

3.3.1 Pre-processing the recorded data

The state of the game is a 96x96 RGB image. Figure 2 showed that the behavior cloning agent performs better if we convert the RGB image to a greyscale image and use that as an input to our agent. We pre-processed the state image with the aim of removing unnecessary details out of the image, so the neural network can learn from binary images with less computation cost. Following steps were performed:

- Recolored road and grass with single color of gray and green respectively.
- Recolored the red and white curbs with the color of the road.
- Hid the reward counter at the bottom of the screen.
- Converted the image to a greyscale image.

The workflow of the demonstration preprocessing is shown in Figure 2 and the result of the preprocessing can be seen in Figure 3

One other preprocessing step was done on the training data. Before training the BC agent, the training dataset was balanced to make sure that no one action accounted for more than 50% of the entirety of the dataset. In most of our

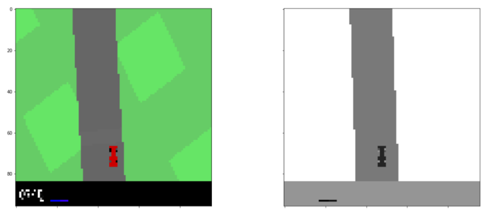

Figure 3: Flowchart for behavior cloning agent

recorded demonstrations, the most common action was driving straight with no steering input, no throttle input, and no brake input. Excess actions were omitted from the dataset and effectively mitigated the dataset bias issue.

3.3.2 Architecture of Behavior Cloning Agent

The architecture of the network is shown in Table 1 where the input to the agent is a 96x96x1 greyscale image, and at the output we have a 3x1 action vector. For training the agent, ADAM optimizer was used with a learning rate of 5 x 10-4. The Mean-Squared Error (MSE) was used as the loss function. The agent was trained on the dataset collected from the expert demonstrators. A total of 6 agents were trained (agent trained on optimal dataset, and agent trained on suboptimal dataset obtain from each of the three experts). For all the agents, the network converged within 250 episodes.

Table 1: Neural network architecture used in behavioral cloning

Layer #	Type	Size	Stride	Activation
1	Conv2D	5x5x16 filters	4x4	ReLU
2	Dropout	50% Drop	-	-
3	Conv2D	3x3x32 filters	2x2	ReLU
4	Dropout	50% Drop	-	-
5	Dense	128 units	-	Linear
6	Dense	3 units	-	Linear

3.4 Retraining Agent

The Convergent Actor-Critic by Humans (COACH) algorithm [5] is used to retrain the agent trained with demonstrations from our team shown in Figure 5. This method allows the agent to learn from policy dependent human feedback. The benefit of using the COACH method includes an advantage function that updates a policy with a use of the critic's TD error shown. Figure 4 shows the overall architecture used to retrain the agent.

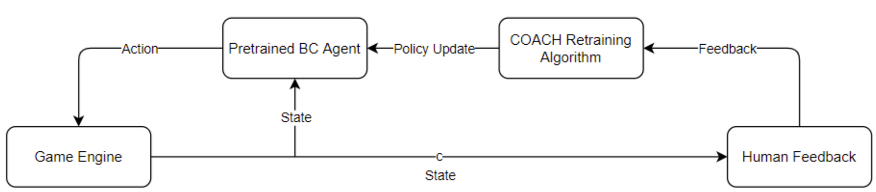

Figure 4: Flowchart for behavior cloning agent

The COACH algorithm starts with inputs of policy, trace set, delay, and learning rate. The trace set is then initialized, and the initial state is observed by the agent. In the next step, the behavior cloning agent makes an action based on the given state. The human input is available to provide a feedback with reward for the agent. For instance, there are feedback values of –1, 0, 1 for steering angles (left and right) and acceleration/braking actions. The feedback value of 0 implies that there is no human input available, and the agent keeps their original action. Subsequently, e_λ represents a mini gradients term and this term is updated in the next using the gradient of policy, state, and actions. Finally, the policy is updated using the terms calculated above.

Algorithm 1 Real-time COACH

Input: policy $\pi_{\boldsymbol{\theta}_0}$, trace set $\boldsymbol{\lambda}$, delay d, learning rate α

Initialize traces $\boldsymbol{e}_\lambda \leftarrow \mathbf{0}\ \forall \lambda \in \boldsymbol{\lambda}$

observe initial state s_0

for $t = 0$ to ∞ **do**

  select and execute action $a_t \sim \pi_{\boldsymbol{\theta}_t}(s_t, \cdot)$

  observe next state s_{t+1}, sum feedback f_{t+1}, and λ

  for $\lambda' \in \boldsymbol{\lambda}$ **do**

    $\boldsymbol{e}_{\lambda'} \leftarrow \lambda' \boldsymbol{e}_{\lambda'} + \frac{1}{\pi_{\boldsymbol{\theta}_t}(s_{t-d}, a_{t-d})} \nabla_{\boldsymbol{\theta}_t} \pi_{\boldsymbol{\theta}_t}(s_{t-d}, a_{t-d})$

  end for

  $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha f_{t+1} \boldsymbol{e}_\lambda$

end for

Figure 5: Flowchart for COACH [4]

Our approach modifies the original coach implementation by incorporating the pretrained behavior cloning agent inside the COACH algorithm. During the retraining phase, we provide feedback to the agent which is then used to calculate gradients that are used to update the policy of the pretrained model. During the retraining phase, we provide feedback to the model in the same manner that a driving instructor instructs a driving student. Our feedback inputs to the agent are shown below in Table 2.

Table 2: Neural network architecture used in behavioral cloning

Key #	a[0]	a[1]	a[2]
Arrow Left	a[0] - 1	0	0
Arrow Right	a[0] + 1	0	0
Arrow Up	0	a[1] + 1	a[2] - 1
Arrow Down	0	a[1] - 1	a[2] + 1
Space Bar		Low Pass filter on a[0]	

We tell the model to slow down or speed up at different sections of the track if it is performing incorrectly. We also tell the model when it should turn left or right if it is turning in the wrong direction or at the wrong time. Our first implementation of the retraining using the COACH algorithm was executed using this feedback only but there were issues with the performance of the BC agent that could not be retrained using the base level feedback. The suboptimal BC agents were prone to high frequency oscillations in the steering after making a turn and base level feedback had negligible effect in eliminating the response. Steering feedback would increase the amplitude of the steering oscillations and accelerator feedback would increase the period of the oscillations.

To reduce this response, a new feedback was implemented. This feedback implements a conditional low-pass filter that is applied to the steering actions output by the trained agent. These filtered actions are then used to update the policy and effectively decay the high frequency steering response. In the next section, we will discuss our results of behavior cloning agent and effect of our modified coach algorithm.

4 Result

The first step of the evaluation was to provide the demonstrations that were going to be used to train our BC agents. Each of the three members of the project group provided 8 to 15 optimal demonstrations and suboptimal demonstrations each. Though the same simulation was used, different experts used different strategies to generate the optimal and suboptimal demonstrations.

Figure 6 and Table 3 below shows the average and standard deviations in the driving score and actions taken by each driver in their optimal and suboptimal demonstration sets. From Figure 6 and Table 3 we can see the heterogeneity caused by different demonstrators. Details of each run is given in Table 5 attached in the Appendix.

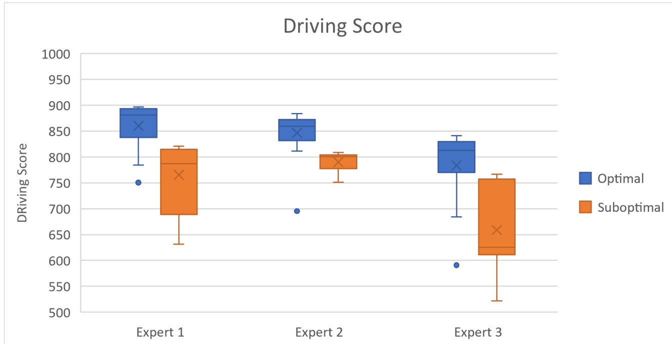

Figure 6: Box Plot showing the driving scores for the optimal and suboptimal demonstrations from each expert

			Score	Straight	Accelerate	Left	Right	Brake
Expert 1	Optimal	μ	860.15	79.7%	6.0%	10.5%	3.8%	0.0%
		σ	47.02	83.6%	3.6%	9.1%	3.6%	0.0%
	Suboptimal	μ	765.29	75.9%	1.8%	13.9%	8.5%	0.0%
		σ	65.18	68.0%	2.9%	16.7%	12.4%	0.0%
Expert 2	Optimal	μ	846.66	73.5%	6.3%	13.3%	7.0%	0.0%
		σ	45.24	65.5%	7.5%	16.3%	10.7%	0.0%
	Suboptimal	μ	790.24	85.8%	1.0%	9.2%	4.1%	0.0%
		σ	19.36	80.8%	0.0%	12.2%	7.0%	0.0%
Expert 3	Optimal	μ	783.58	86.8%	1.3%	8.6%	3.3%	0.0%
		σ	74.62	87.4%	1.0%	8.7%	2.9%	0.0%
	Suboptimal	μ	658.88	68.0%	5.2%	16.0%	10.9%	0.0%
		σ	89.31	70.7%	5.5%	13.7%	10.2%	0.0%

Table 3: Average and Standard Deviation for the driving scores and actions taken by the experts while recording the optimal and suboptimal demonstrations

Each expert arrived at these strategies themselves after doing a few practices. Based on the data we can describe the style of each expert that leads to the heterogeneity between the demonstration sets. The optimal demonstrations of Expert 1 followed tight racing lines that had little acceleration other than starting the demonstration. The suboptimal demonstrations of Expert 1 were driven slowly and were prone to over compensations in the steering. The optimal demonstrations of Expert 2 were driven very quickly but there were additional turns to prevent the car from drifting off the track. The suboptimal demonstrations of Expert 2 were driven very slowly but with minimal steering inputs. The optimal demonstrations of Expert 3 were very slow to ensure that turns were executed correctly and the suboptimal demonstrations were driven much more quickly and Expert 3 had trouble keeping the car on the track.

4.1 Heterogeneity Testing

The first focus of this study was to investigate how the implementation of the COACH retraining could affect the heterogeneity of the demonstrations. The set of optimal demonstrations provided by our individual team members showed substantial differences in driving styles between each other due to different driving styles and abilities. The optimal demonstration set from each team member was then used to train a BC agent. The driving scores and actions taken by each of these three agents were recorded as they completed 15 laps of the track. The summary of the results is shown in Figure 7 and Table 4. It can be seen the behavior cloning agents mimic the driving style of the demonstrator.

The agent trained on optimal dataset from each team member was then retrained using our modified implementation of the COACH algorithm. The BC agents were each retrained for about 7 to 10 episodes. The retrained agents then executed the driving simulation 15 times to generate the dataset used for the performance comparison. Figure 7 and Table 4 shows the average results of the retrained agents compared to the BC baseline.

The driving actions breakdown in Table 4 indicate that the retraining decreased the heterogeneity in the driving styles of the agent. We evaluate the heterogeneity by comparing the consistency in the performance between the three models. Though improved, the heterogeneity in the actions of the agent could not be removed completely. Additionally, it can also be seen that the retraining resulted in an improvement in the performance of each agent.

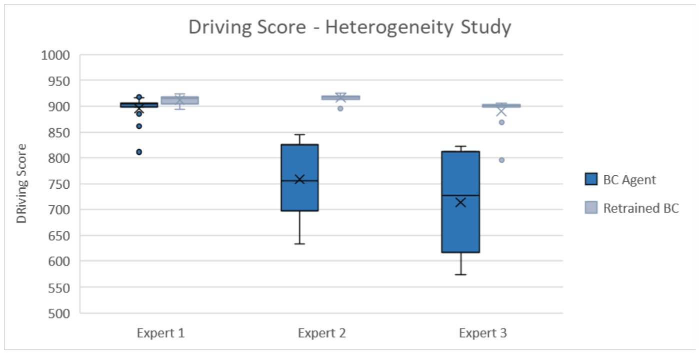

Figure 7: Comparison of BC agent trained on optimal data with the retrained BC agent for each expert

			Score	Straight	Accelerate	Left	Right	Brake
Expert 1	BC Agent	μ	895.47	81.3%	5.4%	10.5%	2.8%	0.0%
		σ	26.98	82.1%	3.1%	9.4%	5.3%	0.0%
	Retrained BC	μ	911.83	54.3%	29.0%	11.9%	3.7%	0.6%
		σ	8.03	51.7%	35.8%	5.5%	4.9%	2.1%
Expert 2	BC Agent	μ	758.33	75.5%	5.1%	12.4%	6.9%	0.0%
		σ	67.67	71.0%	6.5%	13.4%	9.1%	0.0%
	Retrained BC	μ	916.44	38.4%	42.2%	10.3%	5.7%	1.6%
		σ	6.48	35.6%	16.7%	17.1%	24.5%	5.7%
Expert 3	BC Agent	μ	711.43	88.3%	0.7%	8.0%	3.0%	0.0%
		σ	90.80	87.5%	0.0%	8.2%	4.3%	0.0%
	Retrained BC	μ	881.55	80.9%	5.2%	10.2%	3.0%	0.3%
		σ	40.54	84.8%	2.4%	9.2%	2.6%	0.9%

Table 4: Average and Standard Deviation for the driving scores and actions taken by the agent trained on optimal demonstrations from each expert and the retrained agents using COACH

4.2 Suboptimality Study

The second focus of this study was to investigate whether the implementation of the COACH retraining could overcome demonstration suboptimality. Like the heterogeneity demonstration, the suboptimal demonstrations from each team member were used to train three BC agents. The performance of these agents can be seen in Figure 8.

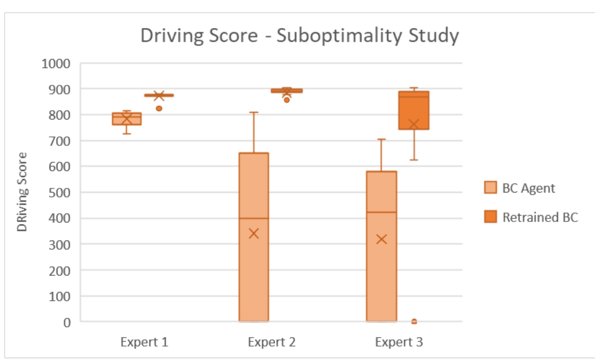

Figure 8: Comparison of BC agent trained on optimal data with the retrained BC agent for each expert

Each of these agents had varying degrees of suboptimality and this investigation intended to determine if these agents could be retrained to produce optimal results or if there was a degree of suboptimality in which retraining could not be successful. Each of the BC agent trained on the suboptimal demonstration obtained from the experts were retrained using the COACH algorithm for about 7 to 10 episodes. The retrained agents then executed the driving simulation 15 times to generate the dataset used for the performance comparison. Seen in Figure (Suboptimality Study), the retrained agents from Experts 1 and 2 perform significantly better after the retraining and exceed the performance of the Optimal BC Agent, as seen in Figure (Heterogeneity Study). Moreover, the variation in the driving score is also reduced significantly. The retrained agent from Expert 3 does show significant improvement but does not reach the performance or consistency of the models from Experts 1 and 2.

5 Discussion

The purpose of our testing was to determine if our framework can decrease the heterogeneity in the actions taken by the behavior cloned agents trained on different datasets, and if our framework is able to improve the performance of the agents trained on suboptimal datasets. From heterogeneity testing results, we can see that the retraining the agent leads to a decrease in heterogeneity of the actions taken by the different agents. From our tests, we can see that after retraining each agent tries to accelerate much more. However, based on the breakdown of each agent's actions we can conclude that heterogeneity was not completely removed. This is due to the retrained agent having no history with certain actions or responses. For example, Expert 1 never applied the brakes in his demonstrations, so the retrained agent uses the brakes the least amount of any of the retrained agent. Additionally, during retraining for Expert 2, one of the episodes had an instance where the agent drove off the track, but since he had not driven off the track in a previous demonstration, the agent did not know how to respond. Following this line of though, we feel that the heterogeneity could be further reduced if a few suboptimal demonstrations were incorporated into the dataset. These suboptimal demonstrations may provide responses that the agent can draw from during retraining that may increase the consistency in the response between agents. From the results of the suboptimality testing, we can see that the retrained agents trained on suboptimal datasets are able to significantly improve their driving scores. There is an 11.4%, a 160%, and a 114% improvement in the driving score of the suboptimal BC agents from Experts 1, 2, and 3, respectively. These retrained agents also matched or outperformed the driving scores of the agents that were trained on optimal dataset obtained from each demonstrator but were not retrained. Lastly, the agents trained on suboptimal datasets were able to reach 97%, 116%, and 110% of the driving score obtained by the corresponding Expert's BC agents trained on the optimal datasets obtained from experts 1, 2, and 3 respectively.

6 Conclusion

In conclusion, the implementation of a human-assisted retraining algorithm was able to greatly improve the performance of Behavior Cloning agents and minimize the effects of demonstration suboptimality and heterogeneity. The heterogeneity of the retrained agents was significantly reduced when compared to the untrained agents. The heterogeneity was not fully eliminated, and elements of the expert's signature persisted after retraining, which led to each agent having unique driving response, but the performance of each agent was very similar. In future work it may be worthwhile investigating the response of retraining an agent trained on a mixed set of optimal and suboptimal demonstrations. The variation seen in the suboptimal demonstrations may generate more randomized response in the trained agent that could reduce or eliminate the stylistic differences between retrained agents. This could further reduce the heterogeneity between the trained agents. Additionally, the retraining algorithm proved to be significantly effective in producing an agent that exhibited near-optimal response though its initial training demonstrations were suboptimal. Based on information given in class and through supplemental readings, understood that suboptimality was a critical problem that needed to be addressed when training an effective agent. Even in the homework assignments, it was very difficult to generate a good agent if even one suboptimal demonstration was involved in the training set. We were curious to see if it were possible use suboptimal demonstrations to create a usable agent and we were quite surprised at how successful a retrained suboptimal agent could be after only a few episodes of retraining. In future work the

References

[1] Daniel S. Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:1232–1241, 2019.

[2] Rae Jeong, Jost Tobias Springenberg, Jackie Kay, Daniel Zheng, Yuxiang Zhou, Alexandre Galashov, Nicolas Heess, and Francesco Nori. Learning Dexterous Manipulation from Suboptimal Experts. pages 1–20, 2020.

[3] W. Bradley Knox and Peter Stone. TAMER: Training an agent manually via evaluative reinforcement. *2008 IEEE 7th International Conference on Development and Learning, ICDL*, pages 292–297, 2008.

[4] Carlos Celemin and Javier Ruiz-del solar. COrrective Advice Communicated by Humans. *2015 International Conference on Advanced Robotics (ICAR)*, pages 581–586, 2015.

[5] James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, David Roberts, Matthew E Taylor, and Michael L Littman. Interactive Learning from Policy-Dependent Human Feedback.

[6] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv*, pages 1–16, 2018.

[7] Guilherme Miotto. DeepLearningLab/DL Lab - Assignment 03 - Guilherme Miotto.pdf at master · guimiotto/DeepLearningLab · GitHub.

[8] Yanyu Zhang and Hairuo Sun. Deep Reinforcement Learning with Mixed Convolutional Network. Technical report, 2020.

A Appendix

The driving scores and the action breakdowns for the optimal and suboptimal demonstrations provided by each expert is given in the following tables.

A.1 Appendix A

	Expert 1 - Optimal Demonstration				
	Score	Straight	Accelerate	Left	Right
1	881.6	967	56	113	48
2	750.7	2091	94	232	76
3	784.6	1783	88	210	73
4	880.3	1001	81	96	19
5	892.9	855	73	112	31
6	890.1	859	82	118	40
7	849.6	1155	121	171	57
8	873	954	104	146	66
9	864	1027	104	169	60
10	801.3	1604	92	216	75
11	893.5	821	70	124	50
12	893.6	804	64	130	66
13	890.3	885	67	107	38
14	896.6	791	83	119	41

Table 5: Expert 1 Optimal Demonstration

	Expert 2 - Optimal Demonstration				
	Score	Straight	Accelerate	Left	Right
1	811.7	1461	83	223	116
2	875.6	926	74	160	84
3	847	1122	98	213	97
4	874	858	100	199	103
5	864.8	942	109	200	101
6	695.3	2134	153	474	286
7	863.4	1058	77	157	74
8	855.6	1038	91	193	122
9	880.4	920	53	140	83
10	865	1030	68	164	88
11	883.8	875	59	149	79
12	826.4	1289	91	237	119
13	856.1	1108	95	169	67
14	855.3	1005	108	225	109
15	825.3	1240	198	198	111
16	866.9	1023	81	159	68

Table 6: Expert 2 Optimal Demonstration

	Expert 3 - Optimal Demonstration				
	Score	Straight	Accelerate	Left	Right
1	834	1429	39	139	56
2	841	1404	27	125	38
3	591	3593	20	357	118
4	830	1454	27	158	65
5	815	1620	20	154	59
6	828	1522	25	129	42
7	797	1711	38	193	83
8	774	1936	36	200	91
9	811	1636	20	169	67
10	769	2016	29	191	78
11	829	1458	27	157	69
12	684	2770	38	264	87

Table 7: Expert 3 Optimal Demonstration

	Expert 1 - Suboptimal Demonstration				
	Score	Straight	Accelerate	Left	Right
1	780.4	1607	25	332	232
2	631.2	2567	81	622	418
3	820.8	1342	42	250	158
4	817	1246	52	318	214
5	685.3	2416	68	405	258
6	689	2468	57	389	196
7	808.7	1485	30	246	152
8	814.8	1467	33	237	115
9	787.2	1694	20	260	154
10	786.2	1634	30	298	176
11	797.6	1673	20	222	109

Table 8: Expert 1 suboptimal Demonstration

	Expert 2 - Suboptimal Demonstration				
	Score	Straight	Accelerate	Left	Right
1	809	1649	20	175	66
2	801.1	1725	20	170	74
3	806.5	1678	20	165	72
4	751.4	2152	20	219	95
5	768.4	1978	20	224	94
6	786.3	1824	20	198	95
7	802	1701	20	175	84
8	800.8	1714	20	179	79
9	786.7	1775	20	227	111

Table 9: Expert 2 suboptimal Demonstration

	Expert 3 - Suboptimal Demonstration				
	Score	Straight	Accelerate	Left	Right
1	522	3545	131	660	444
2	624	2543	165	635	418
3	750	1700	113	406	280
4	626	2719	181	512	332
5	611	2457	270	678	480
6	760	1500	149	443	313
7	767	1592	168	349	218
8	611	2491	241	676	486

Table 10: Expert 3 suboptimal Demonstration

A.2 Appendix B

The driving scores and the action breakdown for the behavior cloned agents trained on optimal and suboptimal datasets is given in the following tables.

	BC Agent (Optimal Demonstrations from Expert 1)				
	Reward	Straight	Accelerate	Left	Right
1	904.5	773	51	104	27
2	898.7	829	47	103	34
3	902.3	817	49	91	20
4	903.7	807	51	88	17
5	811.5	1609	57	178	41
6	905.7	787	53	87	16
7	913.7	691	61	94	17
8	904.6	776	51	100	27
9	916.1	661	66	95	17
10	905.9	766	53	96	26
11	884.9	855	76	144	76
12	898.5	842	46	103	24
13	903.9	796	51	92	22
14	860.7	1130	64	157	42
15	917.3	608	71	112	36

Table 11: BC Agent Expert 1 Optimal Demonstration

	BC Agent (Optimal Demonstrations from Expert 2)				
	Reward	Straight	Accelerate	Left	Right
1	825.3	1520	21	149	57
2	698	2422	146	308	144
3	826	1191	172	239	138
4	633.5	2588	212	535	330
5	682.9	2435	144	368	224
6	709.2	2143	175	386	204
7	744.4	1930	151	314	161
8	754.8	1877	123	291	161
9	805	1419	119	269	143
10	692.7	2398	65	388	222
11	803.5	1503	79	232	151
12	707.1	2230	113	361	225
13	812	1344	135	263	138
14	845.2	1136	100	201	111
15	835.4	1245	105	203	93

Table 12: BC Agent Expert 2 Optimal Demonstration

	BC Agent (Optimal Demonstrations from Expert 3)				
	Reward	Straight	Accelerate	Left	Right
1	574.5	3663	21	382	189
2	611.3	3473	21	300	93
3	633.3	3334	21	257	55
4	810.6	1628	21	167	78
5	823.1	1582	21	127	39
6	818.3	1596	21	146	54
7	747.8	2192	21	222	87
8	612.5	3527	21	257	70
9	656.6	3000	21	291	122
10	707.5	2621	21	218	65
11	812.9	1615	21	168	67
12	755.3	2168	21	187	71
13	815.2	1621	21	154	52
14	621.7	3266	21	347	149
15	670.9	2928	21	252	90

Table 13: BC Agent Expert 3 Optimal Demonstration

	BC Agent (Suboptimal Demonstrations from Expert 1)				
	Reward	Straight	Accelerate	Left	Right
1	784	1732	21	255	152
2	792.4	1731	21	217	107
3	807.3	1636	21	180	90
4	815.3	1583	21	168	75
5	810.5	1626	21	172	76
6	800.2	1674	21	202	101
7	761.6	1920	21	279	164
8	802.6	1597	21	235	121
9	752.2	2019	21	273	165
10	773.3	1871	21	233	142
11	751.4	1999	21	294	172
12	808.3	1512	21	232	152
13	766.5	1884	21	273	157
14	805.7	1512	21	261	149
15	726.6	2119	21	352	242

Table 14: BC Agent Expert 1 suboptimal Demonstration

	BC Agent (Suboptimal Demonstrations from Expert 2)				
	Reward	Straight	Accelerate	Left	Right
1	810.2	1601	21	179	97
2	0	1436	21	85	42
3	454.5	4536	21	458	440
4	0	1039	21	14	0
5	0	1850	21	104	53
6	399.9	4966	21	693	321
7	0	1416	21	84	37
8	0	1391	21	87	44
9	650.9	2950	21	351	169
10	0	1970	21	129	70
11	805.8	1664	21	177	80
12	549.5	3759	21	467	258
13	0	4200	21	379	217
14	788.4	1785	21	207	103
15	651.9	3047	21	287	126

Table 15: BC Agent Expert 2 suboptimal Demonstration

	BC Agent (Suboptimal Demonstrations from Expert 3)				
	Reward	Straight	Accelerate	Left	Right
1	400.2	4011	340	882	765
2	445.7	3644	291	725	883
3	657.6	2240	214	546	424
4	497.1	3543	222	523	741
5	0	3004	242	628	454
6	599.1	2697	225	433	654
7	706.4	1989	207	440	300
8	0	3823	396	1002	769
9	0	4083	301	863	749
10	520.9	3121	228	748	694
11	0	4063	303	932	700
12	0	4182	314	767	734
13	682.1	2075	207	511	386
14	573.1	2939	200	656	474
15	432.3	3760	298	945	674

Table 16: BC Agent Expert 3 suboptimal Demonstration

A.3 Appendix C

The driving scores and the action breakdown for the retrained behavior cloning agents using COACH is given in the following tables.

	Retrained Optimal BC Agent (Expert 1)					
	Reward	Straight	Accelerate	Left	Right	Brake
1	906.9	560	230	106	28	7
2	904.1	612	205	96	33	13
3	914.9	534	200	88	14	15
4	910.8	504	219	115	40	14
5	917.1	388	268	120	40	13
6	916.9	405	275	106	34	11
7	921	384	281	96	27	2
8	918.5	386	288	100	31	10
9	914.2	391	333	93	30	11
10	894.1	483	427	115	23	11
11	902.8	595	225	105	34	13
12	910.5	536	213	107	29	10
13	924.4	359	219	117	48	13
14	916.2	469	222	103	40	4
15	905.1	570	231	101	39	8

Table 17: Retrained Expert 1 Optimal Demonstration

	Retrained Optimal BC Agent (Expert 2)					
	Reward	Straight	Accelerate	Left	Right	Brake
1	918.8	295	359	78	46	34
2	919.6	311	361	77	25	30
3	915.8	348	349	89	32	24
4	924.8	288	320	81	40	23
5	919.5	325	346	74	38	22
6	918.4	325	364	60	39	28
7	916.4	305	368	83	49	31
8	895.9	386	368	130	120	37
9	917.8	337	333	87	35	30
10	913.8	331	343	97	62	29
11	913.7	333	356	97	60	17
12	919.4	282	359	88	45	32
13	912.8	360	345	95	46	26
14	921.4	258	378	76	37	37
15	918.5	323	340	83	42	27

Table 18: Retrained Expert 2 Optimal Demonstration

	Retrained Optimal BC Agent (Expert 3)					
	Reward	Straight	Accelerate	Left	Right	Brake
1	902.2	789	52	101	31	5
2	795.2	1733	74	183	52	6
3	901.3	789	56	103	38	1
4	898.2	807	69	103	27	12
5	901.4	779	54	105	40	8
6	901	779	63	104	31	13
7	903.2	780	58	98	28	4
8	868.6	1009	76	173	48	8
9	901.5	785	55	107	35	3
10	903.2	794	52	90	25	7
11	904.1	774	54	100	29	2
12	905.8	741	70	94	28	9
13	852.6	1212	66	147	41	8
14	782.5	1808	82	211	60	14
15	902.4	801	51	97	25	2

Table 19: Retrained Agent Expert 3 Optimal Demonstration

	Retrained Suboptimal BC Agent (Expert 1)					
	Reward	Straight	Accelerate	Left	Right	Brake
1	876.7	693	182	222	136	0
2	879.9	583	181	242	195	0
3	876.4	622	179	245	190	0
4	878	669	162	232	157	0
5	872.3	665	171	261	180	0
6	878.3	639	193	231	154	0
7	881.9	549	183	251	198	0
8	877.1	648	208	223	150	0
9	871.6	678	184	245	177	0
10	872.7	676	192	243	162	0
11	823.3	808	244	408	307	0
12	876.3	667	174	234	162	0
13	876.4	614	167	264	191	0
14	875.2	630	162	267	189	0
15	879.1	560	197	258	194	0

Table 20: Retrained Expert 1 suboptimal Demonstration

	Retrained Suboptimal BC Agent (Expert 2)					
	Reward	Straight	Accelerate	Left	Right	Brake
1	899.2	513	320	94	57	24
2	893.2	631	306	82	23	26
3	904	494	310	87	49	20
4	899.8	524	321	83	49	25
5	899.4	589	285	66	42	24
6	889.3	659	284	74	61	29
7	889.3	640	298	98	42	29
8	888.4	578	359	94	51	34
9	886.9	680	284	82	57	28
10	877.8	700	333	99	56	34
11	890.4	703	270	73	28	22
12	891.5	606	348	82	23	26
13	813	935	573	185	124	53
14	891	610	345	79	31	25
15	855.7	795	422	118	71	37

Table 21: Retrained Expert 2 suboptimal Demonstration

	Retrained Suboptimal BC Agent (Expert 3)					
	Reward	Straight	Accelerate	Left	Right	Brake
1	889.8	376	300	267	112	47
2	903.6	367	226	232	97	42
3	793.8	651	509	385	406	111
4	865.2	420	336	252	280	60
5	874.4	385	289	332	214	36
6	889.4	544	265	162	95	40
7	0	711	476	4096	384	334
8	813.7	662	388	378	350	85
9	879.6	377	312	231	230	54
10	899.2	350	349	178	101	30
11	623.6	793	786	1516	555	114
12	726.2	703	758	634	525	118
13	816.5	641	486	328	266	114
14	898.8	304	312	226	125	45
15	920.6	249	274	155	82	34

Table 22: Retrained Expert 3 suboptimal Demonstration