Fast Lifelong Adaptive Inverse Reinforcement Learning from Demonstration Supplementary

Letian Chen*, Sravan Jayanthi*, Rohan Paleja
Daniel Martin, Viacheslav Zakharov, Matthew Gombolay
Georgia Institute of Technology
Atlanta, GA 30332
{letian.chen, sjayanthi, rpaleja3, dmartin1, vzakharov3,
matthew.gombolay}@gatech.edu

1 Real-World Table Tennis Robot Experiment

1.1 System Setup

The setup of our table tennis environment consists of a 7-degree of freedom WAM robot arm from Barrett Technology, a 3D printed table tennis paddle holder, three ZED 2 stereo cameras, and a Butterfly Amicus Prime ball feeder. The angle and speed of the ball feeder were calculated empirically prior to collecting human demonstrations and kept constant throughout the experiments. We use Robot Operating System (ROS) as our communication framework for controlling the vision system as well as the arm control system. We record the participant's demonstrations by subscribing to the robot joint state topic. This provides us with joint positions at a rate of 100 Hz. For controlled movements, we use a PID-based positional control loop running at 500 Hz for performing swings. Note that our algorithm sends position control commands at 100 Hz.

1.2 Vision System

A multi-camera vision system is employed to accurately localize the ball. The stereo-cameras' poses were calibrated using an April tag bundle [1] to find their relative world positions. To detect orange table tennis balls, the vision system attempts to find the position of moving orange objects by using classical computer vision techniques including background subtraction and color thresholding. The 720p camera resolution allows the vision system to detect the ball with a frequency of up to 60 Hz.

The vision system then utilizes stereo-geometry to calculate the ping-pong ball coordinates in each stereo-camera's frame of reference. These points are added to the position and orientation derived from the calibration results described above to produce an absolute position of the table tennis ball at the time of image capture. These ball position estimates are fused through an Extended Kalman Filter (EKF) by combining the sensor measurements to produce a single pose estimate. The EKF's state estimate prediction is augmented using ballistic trajectory equations with a table bounce model. An EKF was chosen due to its known robust performance to outlier measurements in presence of a well-defined linear dynamics model. This allows us to use the world position estimate to track the table tennis ball during demonstrations and executions.

1.3 Experiment Details

In this experiment, we test FLAIR on its capability to adapt to users' demonstrations by optimizing *policy mixtures*. We first collect demonstrations for four different table tennis strategies, push, slice, topspin, and lob, from human subjects via kinesthetic teaching. After training the four prototypical strategy policies, we assess how FLAIR's policy mixtures can succeed in new user demonstration modeling in terms of both accomplishing the task and personalizing to the user's preference, and compare FLAIR's performance with a learning-from-scratch approach.

* denotes equal contribution

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

1.3.1 Prototypical Strategy Policy Creation

We recruit one participant (male, college student) for the purpose of providing prototypical strategy demonstrations for push, slice, topspin, and lob strikes. Note this participant is familiar with the WAM robot but does not have past experience providing demonstrations for table tennis strikes. The participant provides kinesthetic teaching to the arm, where we idle the robotic arm with gravity compensation and record joint positions along with ball position estimates. Each demonstration starts from the first frame the cameras detect the ball, and lasts one second (100 timesteps). The participant provides five demonstrations for each of the strategy. After the demonstrations are given, we replay the demonstrations and select one demonstration for each strategy that empirically performs the best for the following training procedure.

For each prototypical strategy policy, we train a two-layer fully connected neural network (with 64 neurons on each layer and ReLU activation functions for hidden layers). The neural network's input is 17-dimensional with seven joints' positions and velocities and the ball's 3D coordinates. The output is 7-dimensional, encoding the difference of the newly desired positions for seven joints relative to the current joint positions. In order to warmstart the policy neural network training, we conduct three phases of warmstart training.

First, we train the policy with behavior cloning loss, minimizing the distance between the policy output action with the ground-truth action taken by the demonstrator.

Second, we augment the behavior cloning data with replays of the human demonstration on the robot and continue behavior cloning with the augmented data, making the policy more robust to robot execution drifts. We repeat this "data-augmented behavior cloning" phase of training for ten environment episodes.

Third, we adopt DAGGER [2] training scheme where we let the policy control the execution and the "expert" (the demonstration) provides corrective feedback for the robot to move back to the demonstrated trajectory, shown in Equation 1, where $s_{t+1}^{\text{demonstration}}$ is the robot position in the next timestep in the demonstration, and s_{current} is the current robot position (not from demonstration). We perform this phase until the policy converges.

$$a_t^{\text{corrected}} = s_{t+1}^{\text{demonstration}} - s_{\text{current}} \tag{1}$$

Through the three phases of warmstarting training, we get fluent, successful robot trajectories for each strategy. This phase essentially represents FLAIR building up a set of prototypical strategy policies with initial demonstrations.

1.3.2 User Demonstration Adaptation by *Policy Mixtures*

We recruit twenty-eight participants from a population of college students to provide demonstrations that FLAIR adapts to. The experiment is split up into two sessions. In the first session, we start by teaching the participants the push, slice, topspin, and lob strikes with participant's practice after each to get them familiar with the setup. Once the participant subjectively judges he/she is comfortable with the setup, he/she starts to practice his/her preferred strike. When the participant is ready, we record five repeats of his/her preferred strikes.

After the first session (personalized demonstration collection), we create 20 policy mixtures with the four prototypical strategy policies, calculate the KL divergence on the state marginals between the demonstrations and the policy mixtures, and obtain the most and the least matching policy mixtures for each participant. Through this approach, FLAIR could reuse the same 20 policy mixtures across participants. To compare with FLAIR's policy mixtures, we train a learning-from-scratch neural network for each of the participant through the approach described in Section 1.3.1 with a budget of 20 episodes to make the comparison fair between FLAIR and learning-from-scratch. Note we also try to train AIRL for each participant with 20 environment episodes, but AIRL fails to produce any meaningful policy on the robot without the warmstart training of the learning-from-scratch approach. Therefore, we compare FLAIR's closest mixtures and least-close mixtures with the learning-from-scratch policy for each participant. For the simplicity of description, we name the FLAIR's closest mixtures as "FLAIR's best mixture" and FLAIR's least-close mixture as "FLAIR's worst mixture".

In the second human-subject session, we invite the participants back to show them the robot striking a ping pong ball based upon what the robot learned from their demonstration. We show twelve

Table 1: This table depicts policy metrics between FLAIR's best mixtures, FLAIR's worst mixtures, and learning-from-scratch with AIRL. The scores are shown as averages $\pm$ standard deviations across twenty-eight participants. Bold denotes the highest scores.

Metrics	FLAIR's Best Mixture	FLAIR's Worst Mixture	Learning-from-Scratch
Task Score	$\mathbf{66.9 \pm 10.3}$	59.5 ± 12.8	56.6 ± 12.3
Strategy Score	$\mathbf{96.6 \pm 17.4}$	70.3 ± 23.7	90.0 ± 18.0

strikes (i.e., trajectories) consisting of three sets of four replications each of 1) FLAIR's best mixture for that subject, 2) FLAIR's worst mixture for that subject, and 3) a policy that was learned from scratch on that subject's demonstrations. The order of the trajectories shown is randomized.

After each robot trajectory, we administer a 10-item Likert Scale on a 5-point response scale (Strongly Disagree to Strongly Agree). The questionnaire includes ten questions, where four questions pertain to whether the robot successfully accomplishes the task, and six questions are about whether the performed trajectory fits the participant's style. The participants are invited to express their comments about the strikes performed by the robot after the questionnaire is completed.

The ten questions in the questionnaire are listed below. Questions 1-6 are for assessing the strategy component and questions 7-10 are for the task component of the robot's performance.

1. The robot did a good job performing the task using the style I demonstrated.
2. The robot understands my style in performing the task.
3. The robot paid attention to the style I demonstrated.
4. The robot failed to imitate the style I demonstrated for the task.
5. The robot tried to perform the task using its own style.
6. The robot ignored my preferences for how to do the task.
7. The robot attempted to accomplish the task.
8. The robot knows how to hit the targeted spot.
9. The robot did not work on assigned task.
10. The robot performed the task poorly.

1.4 Experimental Results

In this section, we quantitatively compare FLAIR's performance on accomplishing the task and personalizing to user preference with the learning-from-scratch approach, and qualitatively show policy mixtures FLAIR creates.

Based on the questionnaire each participant fills, we calculate a strategy score and a task score by summing the corresponding Likert items. As such, the strategy score ranges from 24-120 (four repeated trajectories for one policy by 6 strategy questions with each question having a score of 1-5). Similarly, the task score ranges from 16-80.

The strategy and task scores results are summarized in Table 1. Levene's test shows the homoscedasticity assumption holds for comparisons of strategy score ($W = 1.24, p = 0.296$) and task score ($W = 0.78, p = 0.464$), and therefore we carry out ANOVA tests when comparing the three policies. The one-way repeated measure ANOVA shows significant differences in both the task score ($F(2, 54) = 9.88, p < .001$) and the strategy score ($F(2, 54) = 22.55, p < .001$). The posthoc paired t-test shows the FLAIR best mixture has significantly higher task scores than both learning-from-scratch ($t(27) = 5.06, p < .001$) and the FLAIR worst mixture ($t(27) = 2.88, p = 0.004$). The posthoc paired t-test on strategy score shows that the FLAIR best mixture has significantly higher strategy scores than both learning-from-scratch ($t(27) = 1.93, p = 0.032$) and the FLAIR worst mixture ($t(27) = 5.88, p < .001$). Learning-from-scratch has a higher strategy score than the FLAIR worst mixture ($t(27) = 4.62, p < .001$). We applied the Holm–Bonferroni method to counteract the problem of multiple comparisons. After ranking the three p-values, we observe the lowest p-value (the FLAIR best mixture vs. the FLAIR worst mixture, $p < .001$) is less than $0.05/n = 0.05/3 \approx 0.017$, thus being significant. The second lowest p-value (learning-from-scratch vs. the FLAIR worst mixture, $p < .001$) is less than $0.05/(n-1) = 0.05/2 = 0.025$, thus

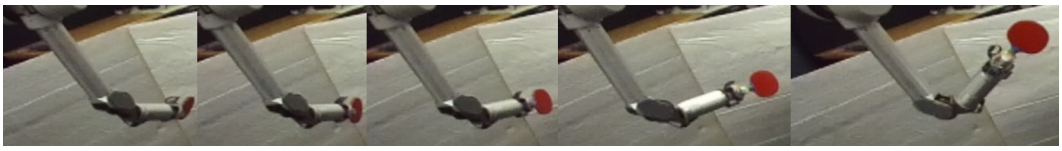

Figure 1: Frames of a 90% **Topspin +** 10% **Lob** mixture (full video available in the supplementary video). It is seen in the motion that the paddle starts in a tilted motion (standard for a topspin strike) and curves upward to return a ball with high curvature (typically of a lob strike). The location of the paddle also moves from low to high, which fits the characteristics of a lob strike.

being significant. The third lowest p-value (the FLAIR best mixture vs. the FLAIR worst mixture, $p = 0.032$) is less than $0.05/(n-2) = 0.05$, thus also being significant. These results indicate the success of FLAIR's mixture optimization in identifying a policy mixture that accomplishes the task and fulfills the user's preference in the table tennis real-robot setup.

The mean task and strategy scores FLAIR best mixture achieves are higher than the ones of the learning-from-scratch policy. Moreover, FLAIR best mixture has smaller standard deviation on both scores, showing FLAIR's best mixture not only achieve higher scores in general but also has a more stable performance. In fact, multiple participants note the learning-from-scratch policies are "jerky" and "noisy". On the contrary, three participants provided compliments for FLAIR best mixture trajectories including "these four trajectories (from FLAIR best mixture) performs much better than others and are very similar to what I did", "this trajectory is amazing", "this trajectory performs my strategy even better than I did", and "this trajectory is exactly what I want. Can I give it a star on top of the score 5?"

We further qualitatively provide several mixture videos in the supplementary video. Policy mixtures are versatile, creating novel behaviors that successfully accomplish the task of hitting the ping pong ball over the net. We illustrate one such mixture (90% Topspin + 10% Lob) in Figure 1, displaying a frame-by-frame example of a mixture producing an interesting motion.

2 Method Details

2.1 State Marginal Distribution KL-Divergence Estimation

Why utilize the KL Divergence? The KL-divergence is a common choice to measure the distance between two distributions in the machine learning community and within the imitation learning field [3]. A recent study of different f-divergence measures shows Forward KL is a well-performing f-divergence metric to compare expert trajectories with policy-generated trajectories in an IRL setting [4]. Our method, FLAIR, utilizes the forward KL divergence between the demonstration's and mixture policy's estimated state distributions as the minimization objective in mixture optimization and a goodness-of-fit metric.

How do we estimate the KL-Divergence between two state marginal distributions with their samples? To calculate the estimated KL-Divergence between two state marginal distributions, we utilize the identity in Equation 2, where p and q are two probability distributions, $\mathbb{H}(p, q)$ denotes cross entropy, and $\mathbb{H}(p)$ denotes entropy.

$$D_{\text{KL}}(p, q) = \mathbb{H}(p, q) - \mathbb{H}(p) \tag{2}$$

We adopt the Kozachenko-Leonenko estimator, a non-parametric entropy estimator that uses k-nearest-neighbors distances of n i.i.d random vectors, to estimate the entropy and cross entropy [5]. The states in demonstrations and policy rollouts are pooled to serve as i.i.d. random samples for the two state marginal distributions. FLAIR performs k-nearest-neighbors to estimate KL-divergence between the true state marginal distribution (expert demonstration) and the generated state marginal distribution (rollouts from mixture policy), which follows [4].

Table 2: This table shows the results of multiple, non-differentiable optimization methods in minimizing the estimated KL Divergence in Inverted Pendulum in one trial.

Estimated KL Divergence	Random Search	PSO	GPO	CMAES
Demonstration 1	**2.39124**	6.420129	6.489698	17.45635
Demonstration 2	-0.91501	**-1.03243**	-0.78408	22.73914
Demonstration 3	2.676944	**2.492234**	2.705531	14.91208
Demonstration 4	-0.7321	-0.7271	**-0.70971**	18.07601
Demonstration 5	0.348959	**0.272621**	0.6422	14.0717
Demonstration 6	**18.8059**	18.91321	19.37367	21.19286
Demonstration 7	**-2.85413**	-2.93581	0.558	19.046
Demonstration 8	2.5174	**-1.35019**	12.76616	22.05171
Average	2.779906	**2.756573**	5.130184	18.69326

2.2 Policy Mixture Optimization Method

We study different approaches for performing the non-differentiable policy mixture optimization. We consider approaches including Particle Swarm Optimization (PSO) [6], Bayesian Optimization (GPO) [7], and Covariance Matrix Adaptation Evolution Strategy (CMAES) [8]. We examine how these approaches perform during a trial of FLAIR and find that random search proves to be the most effective method. As shown in Table 2.2, we find that, despite the marginally lower average estimated KL divergence of 1% (≈ 2.78 for Random Search compared to ≈ 2.76 for PSO), random search is a reliable method for exploring the space of possible mixture policies and selecting a well-performing one. Other approaches such as GPO or CMAES fail to meet similar performance as Random Search and PSO (GPO is 85% higher than Random Search, CMAES is 572% higher than Random Search).

2.3 Proof of Lemma 1

If demonstration τ_i has weight $w_{i,j}$ on strategy j (as identified in *Policy Mixture*), we could view the probability that τ_i happens under the strategy reward, $R_{\text{S-}i}$, should be $w_{i,j}$ proportion of the probability of the pure demonstration, τ_{m_j}. This property can be exploited to enforce a structure on the reward given to the pure-demonstration, τ_{m_j}, and mixture-demonstration τ_i, as per Lemma 1.

Lemma 1. *Under the maximum entropy principal,*

$$w_{i,j} = \frac{P(\tau_i; \textit{S-}j)}{P(\tau_{m_j}; \textit{S-}j)} = \frac{e^{\eta R_{\textit{S-}j}(\tau_i)}}{e^{\eta R_{\textit{S-}j}(\tau_{m_j})}}$$

Proof. According to the maximum entropy principle,

$$P(\tau; R) = \frac{e^{\eta R(\tau)}}{\int_{\tau'} e^{\eta R(\tau')}}$$

Therefore,

$$\begin{aligned}\frac{P(\tau_i; \text{S-}j)}{P(\tau_{m_j}; \text{S-}j)} &= \frac{\frac{e^{\eta R_{\text{S-}j}(\tau_i)}}{\int_{\tau'} e^{\eta R(\tau')}}}{\frac{e^{\eta R_{\text{S-}j}(\tau_{m_j})}}{\int_{\tau'} e^{\eta R(\tau')}}} \\ &= \frac{e^{\eta R_{\text{S-}j}(\tau_i)}}{e^{\eta R_{\text{S-}j}(\tau_{m_j})}}\end{aligned}$$

Combined on our assumption, $w_{i,j} = \frac{P(\tau_i; \text{S-}j)}{P(\tau_{m_j}; \text{S-}j)}$, we prove Lemma 1. □

3 Simulation Experiment Details

3.1 Environment Details

We test FLAIR on three simulated continuous control environments in OpenAI Gym [9]: Inverted Pendulum [10], Lunar Lander, and Bipedal Walker [11]. The goal in Inverted Pendulum (IP) is to

balance a pendulum by the cart's horizontal movements, where reward is given by the negative angle between the pendulum and the upright position. The objective in Lunar Lander (LL) is to achieve a controlled landing of a spacecraft. The agent receives a reward of 100 if it successfully lands, -100 if it crashes, 10 for each leg-ground contact, and -0.3 for firing its engine. The goal in Bipedal Walker (BW) is to teach a robot to walk using the hull speed, joints' angular speed, and 10 LiDAR rangefinder measurements. BW receives a reward based on forward-moving speed.

Generally, various strategies in IP include sliding or swinging along the rail. In LL, the spacecraft takes unique flight paths to approach the landing pad. In BW, the robot limps, runs, or hops to propel itself forward[1].

We note that we modified InvertedPendulum-v2, LunarLanderContinuous-v2, and BipedalWalker-v3 environments to disable the option "terminate_when_unhealthy" and set a fixed horizon of 1000, 500, and 1000, respectively. In turn, when calculating the task reward correlation, it is more difficult to achieve a high correlation with the ground truth task reward due to large cumulative negative rewards that punish failing behaviors and modest rewards reinforcing successful behaviors.

3.2 Experiment Details

We generate a dataset of 10 heterogeneous demonstrations by jointly optimizing an environment and diversity reward with DIAYN [12] for each domain, Inverted Pendulum (IP), Lunar Lander (LL), and Bipedal Walker (BW). We utilize this dataset of ten heterogeneous demonstrations for all experiments except the scalability experiment. We provide these same demonstrations in our experiments to each of the baseline methods and FLAIR. We also generate an additional test dataset of 10,000 demonstrations and record the cumulative reward of trajectories during training to evaluate the correlation between our learned task reward and the ground truth environment reward.

On policy evaluation metrics, we train AIRL on each individual demonstration (named AIRL Single) in favor of its personalization. On reward metrics, we train AIRL on the entire set of demonstrations (named AIRL Batch) to improve its reward robustness.

3.3 Mixture Optimization Details

In order to accelerate the mixture optimization process, we parallelize the mixture policy rollouts in fixed batches of thirty for Inverted Pendulum and nine for Lunar Lander and Bipedal Walker trajectories. This difference is due to computational constraints to parallel rollouts. After each batch, we check whether any of the mixtures has an estimated KL Divergence below our KL divergence threshold, ϵ, and, if so, we end the search. If not, we continue the search until we reach the cap of random search samples, which is 2000. If we cannot find a suitable mixture by this limit of 2000 samples, we train a new policy and compare its performance with the best mixture found. We include the new demonstration as a mixture if the mixture policy has a lower estimated KL divergence with the demonstration. Otherwise, the demonstration is introduced as a new strategy with the newly trained policy, as shown in the pseudocode of the main paper.

3.4 Implementation Details and Hyperparameters

We utilized the rllab [13], garage [14], AIRL [15], MSRD [16] implementations of TRPO, AIRL, and MSRD to develop FLAIR[2].

For Inverted Pendulum, FLAIR trains 600 iterations of AIRL if it is needed for a new strategy and 400 iterations of MSRD when each demonstration is introduced. We train AIRL for 600 iterations since it is empirically the number of iterations it takes for AIRL to converge in Inverted Pendulum and the additional iterations of MSRD improve the learned task reward. The mixture optimization threshold, ϵ, is 1.0, empirically tuned to encourage the best performance by evaluating how closely the mixture videos align with the demonstrations. FLAIR starts the MSRD and Between Class Discrimination training once three strategies are introduced. The maximum number of samples used for policy mixture optimization is 2000.

[1] Link for the videos of the demonstrations and learned policies of FLAIR: https://tinyurl.com/FLAIRVIDS

[2] All code/data will be open sourced.

Table 3: This table shows the hyperparameters used in the benchmark experiments for all methods studied (AIRL, MSRD, and FLAIR).

Hyperparameter	Method	Value
Discriminator Update Step In Each Iteration	All	10
Batch Size	All	10000
Episode for Rollouts per Iteration	All	10
γ	All	0.99
Entropy Weight	All	0.0
Fusion Size	All	10000
L2 Regularization: Strategy Reward	MSRD & FLAIR	0.01
L2 Regularization: Task Reward	MSRD & FLAIR	0.0001

Table 4: This table shows learned policy metrics between AIRL, MSRD, and FLAIR.

Domains	Inverted Pendulum			Lunar Lander			Bipedal Walker		
Methods	AIRL	MSRD	FLAIR	AIRL	MSRD	FLAIR	AIRL	MSRD	FLAIR
Demonstration Log Likelihood	-29216.5	-40870.5	$\mathbf{-6525.0}$	-14835.5	$\mathbf{-11124.2}$	-14550.8	$\mathbf{-48162.6}$	-88557.8	-59406.6
Environment Return	-172.7	-166.4	$\mathbf{-38.5}$	-7418.1	-9895.3	$\mathbf{-6346.6}$	-30637.2	-74166.0	$\mathbf{-7064.0}$
Estimated KL Divergence*	4.08	7.67	**4.01**	72.0	70.9	**67.2**	13.0	32.6	**12.1**
Strategy Rewards	-5.73	-6.22	$\mathbf{-1.23}$	-12.67	-20.26	$\mathbf{-4.19}$	-5.31	-29.82	$\mathbf{-4.22}$

* Lower is better

For Lunar Lander, FLAIR trains 1000 iterations of AIRL if it is needed for a new strategy and 100 iterations of MSRD when each demonstration is introduced. Likewise for Bipedal Walker, FLAIR trains 1800 iterations of AIRL and 100 iterations of MSRD. The mixture optimization threshold is 40.0 for Lunar Lander and 8.0 for Bipedal Walker. FLAIR starts MSRD and Between Class Discrimination when three strategies are introduced. The maximum number of samples for random search is 900, with three repeats for each mixture weight, meaning each mixture policy is rolled out three times and all three trajectories are used in the KL divergence estimation. The KL divergence is calculated with respect to three repeats of the policy rollouts, thus ensuring a more robust estimation of the state marginal distributions.

In all domains, Flair has a learning rate of 0.0001 for Between Class Discrimination (BCD) and each iteration it samples 10 trajectories for training. For a fair comparison against AIRL and MSRD, we calculate the number of environment episodes used by FLAIR, and train both AIRL and MSRD to the same number of samples in each domain. All methods use a replay buffer (named fusion) in reward training which keeps a record of generated trajectories for reward training, similar to [15]. For AIRL, MSRD, and FLAIR, the hyperparameters for policy and reward training are shown in Table 3.

3.5 Result Details

3.5.1 Policy Performance

This section corresponds to the Q1&Q2 in the main paper.

Table 4 summarizes our results for comparing policy metrics, and Table 5 provides results of associated statistical tests. Tests for normality and homoscedasticity indicate that the data does not satisfy the assumptions of a parametric ANOVA test when comparing FLAIR with benchmarks. Thus, we instead perform a non-parametric Friedman test followed by a posthoc Nemenyi–Damico–Wolfe (Nemenyi) test [17].

Demonstration Log Likelihood FLAIR is able to model the personal preferences demonstrated by the users similarly to AIRL and MSRD, shown as "Demonstration Log Likelihood" in Table 4. Note that AIRL trains a separate policy for each demonstration from scratch, and MSRD has access to the ground-truth strategy labels. A Friedman test is significant for IP and BW ($p < .01$) and only AIRL significantly outperforms MSRD in the posthoc Nemenyi test ($p < .01$). AIRL models each demonstration individually while MSRD builds a static joint model of all demonstrations. We note that AIRL has advantage on the metric by creating a separate model for each of the demonstrations but are notoriously inefficient as shown in Figure 3 of the main paper. In contrast, FLAIR auto-

Table 5: This table shows statistical tests for policy metrics comparing AIRL, MSRD, and FLAIR.

Domains	Inverted Pendulum			Lunar Lander			Bipedal Walker		
Tests	Friedman	Nemenyi	Nemenyi	Friedman	Nemenyi	Nemenyi	Friedman	Nemenyi	Nemenyi
FLAIR vs.		AIRL	MSRD		AIRL	MSRD		AIRL	MSRD
	$Q_2 =$	$q_{87} =$	$q_{87} =$	$Q_2 =$	$q_{87} =$	$q_{87} =$	$Q_2 =$	$q_{87} =$	$q_{87} =$
Demonstration Log Likelihood	16.8^{**}	0.77	3.87^{**}	2.4			31.2^{**}	2.32	3.49^{**}
Environment Return	29.4^{**}	4.26^{**}	5.03^{**}	6.6^{*}	1.16	2.52^{*}	$34.2**$	2.32	5.81^{**}
Estimated KL Divergence	22.2^{**}	0.39	4.26^{**}	12.6^{**}	2.53^{*}	3.49^{**}	37.8^{**}	1.16	5.81^{**}
Strategy Rewards	1.40	0.77	0.39	6.87^{*}	0.90	2.58^{*}	26.87^{**}	0.65	4.78^{**}

* Significance of $p < 0.05$
** Significance of $p < 0.01$

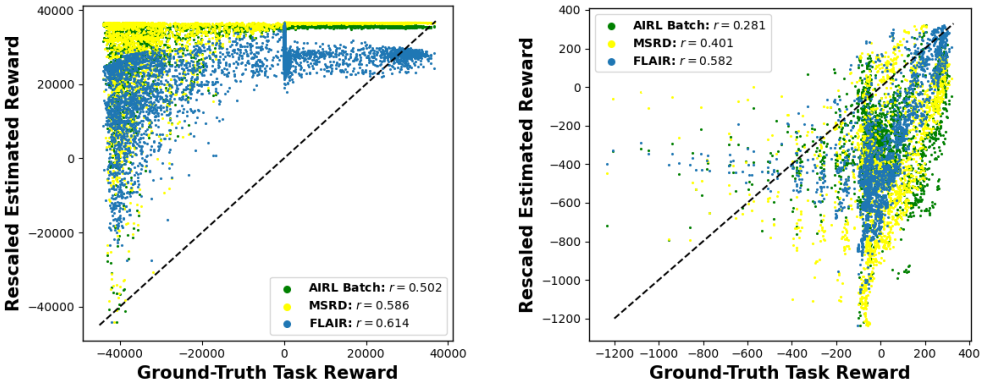

Figure 2: This figure shows the correlation between the estimated task reward with the ground truth task reward for Lunar Lander and Bipedal Walker respectively. Each dot is a trajectory. FLAIR achieves a higher task reward correlation than baselines. The diagonal dashed lines denote perfect correlation.

matically constructs policy mixtures to more efficiently adapt to each demonstration and achieves a similar performance.

Environment Return FLAIR succeeds at learning policies that perform better at the ground truth task. A Friedman test is significant in all three domains ($p < .01$ in IP/BW, $p < .05$ in LL) and FLAIR outperforms both AIRL and MSRD in IP ($p < .01$) and BW ($p < .05$ for AIRL, $p < .01$ for MSRD). Additionally, FLAIR outperforms MSRD in LL ($p < 0.05$), showing that FLAIR is able to adeptly tease out the latent task goal and leverage it to train highly successful policies.

Estimated KL Divergence Qualitatively, we find that FLAIR learns policies and policy mixtures that closely resemble their respective strategies, visualized in policy renderings[1]. Quantitative evidence that FLAIR generates trajectories that are closer to the demonstration than both AIRL and MSRD, shown as row "Estimated KL Divergence" in Table 4, which is evaluated between the policy rollout and the demonstration state marginal distributions. A Friedman test is significant in all domains ($p < .01$). In IP and BW, FLAIR significantly outperforms MSRD ($p < .01$) while in LL, FLAIR significantly outperforms both AIRL ($p < .05$) and MSRD ($p < .01$).

3.5.2 Task Reward Correlation

This section corresponds to the Q3 in the main paper.

We evaluate the learned task reward functions by calculating the correlation between estimated task rewards and ground-truth environment rewards. We construct a test dataset of 10,000 trajectories with multiple policies obtained during the "DIAYN+env reward" training by collecting ten trajectories for each policy every 100 training iterations. Through this approach, the test dataset has different strategies with varying success. We compare correlations using a z-test after Fischer r-to-z-transformation [18]. For IP, FLAIR has a better correlation than AIRL ($z = 58.56, p < .01$), and than MSRD ($z = 20.76, p < .01$). Likewise in LL, as shown in Figure 2, FLAIR has a correlation

Table 6: This table shows the performance of FLAIR in an Ablation experiment with Between Class Discrimination (BCD) for Inverted Pendulum.

Average Metrics	FLAIR without BCD	**FLAIR with BCD**
Environment Return	-129.593	**-38.5**
Log Likelihood	-16947.5	**-6525.0**
Estimated KL Divergence*	4.44	**4.01**
Task Reward Correlation	**0.954**	0.953
Cosine Distance*	0.43	**0.03**

* Lower is better

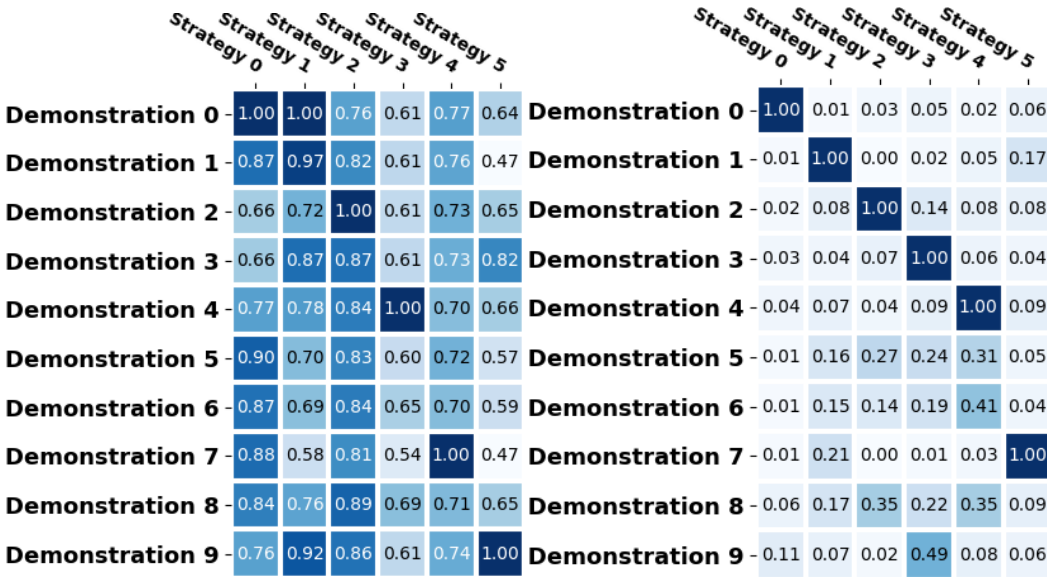

Figure 3: This figure depicts the normalized rewards on demonstrations based on the strategy reward output in Inverted Pendulum for FLAIR without BCD (left) and with BCD (right).

$r = 0.614$ which is better than AIRL $r = 0.502$ ($z = 11.55, p < .01$) and MSRD $r = 0.586$ ($z = 3.09, p < .01$). In BW, as shown in Figure 2, FLAIR has a correlation $r = 0.582$ which is better than AIRL $r = 0.281$ ($z = 26.6, p < .01$) and MSRD $r = 0.401$ ($z = 17.0, p < .01$). AIRL underperforms since it treats heterogeneous demonstrations as homogeneous and does not distinguish between the strategic preference and the underlying task objective.

3.5.3 Strategy Reward Learning & BCD Ablation

This section corresponds to the Q4 in the main paper.

We evaluate learned strategy rewards on the demonstrations, "predict" the strategy mixture weights for each demonstration via strategy rewards, and compare the predicted strategy labels to the strategy weights obtained from *mixture optimization*. More specifically, we normalize the strategy reward outputs with a demonstration to obtain the predicted strategy reward label by $C_{i,j} = \frac{e^{R_{\theta_{\text{S-}i}}(\tau_j)}}{\max_{k=1}^{n} e^{R_{\theta_{\text{S-}i}}(\tau_k)}}$. We compute the cosine distance between the true mixture weights (obtained via *mixture optimization*) and the predicted label (calculated with strategy rewards) as in Equation 3.

$$\text{Cosine Distance} = 1 - \frac{\vec{w} \cdot \vec{C}_i}{||\vec{w}|| \times ||\vec{C}_i||} \tag{3}$$

With this metric, we compare how successful the strategy reward is in discriminating strategic preferences at the end of training with and without BCD. We calculate the predicted class labels by the learned strategy rewards for each demonstration (shown in Figure 3, Figure 4, and Figure 5 for the three domains) and compare them to the ground-truth strategy weights (estimated by mix-

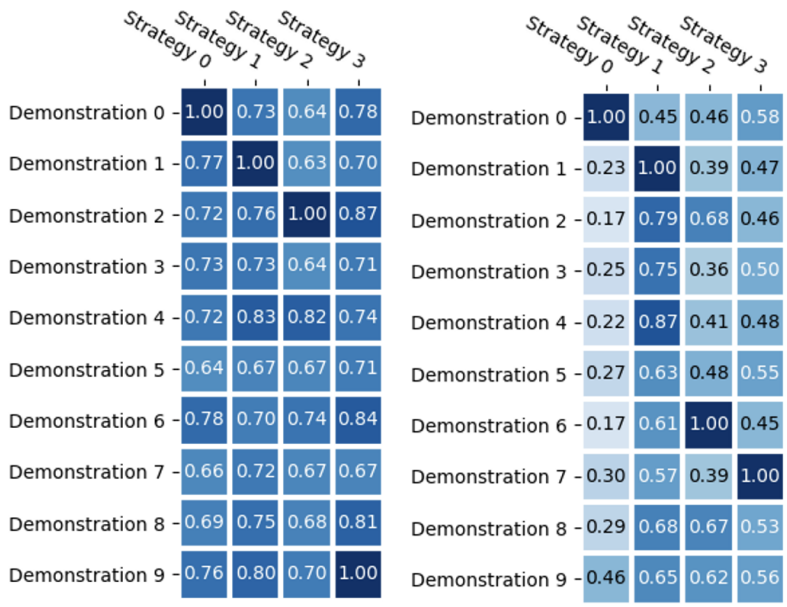

Figure 4: This figure depicts the normalized rewards on demonstrations based on the strategy reward output in Lunar Lander for FLAIR without BCD (left) and with BCD (right).

ture optimization). The performance improvement with BCD is shown in Table 6. Along with all key policy metrics such as Environment Return and Log Likelihood, FLAIR with Between Class Discrimination shows a lower cosine distance of 0.03, compared to 0.43 of FLAIR without BCD, between the true strategy labels and the strategy reward predictions. The results show FLAIR with Between Class Discrimination can train the strategy reward to better recognize the class labels of each demonstration. In contrast, the strategy rewards in FLAIR without Between Class Discrimination do not clearly distinguish between different strategies hence cannot identify the strategic preference for each strategy.

3.5.4 Scalability

This section provides more details to Q6 of the main paper.

As described in our larger scale LfD experiment, we generate 95 mixtures with randomized weights from 5 base policies for a total of 100 demonstrations to test how well FLAIR scales. Our goal is to study the success of FLAIR in a lifelong learning setting by evaluating how it scales to the challenge of modeling a large number of demonstrations. We perform this large scale experiment in all 3 domains and compare the results to the baseline metrics of AIRL, MSRD, and FLAIR from the ten demonstration experiment in Section 3.5.1. We include results of the environment returns as each demonstration is introduced in the experiment along with additional results for other key metrics, including log likelihood, KL divergences, and task reward correlation in Figure 8.

FLAIR demonstrates consistently strong performance in environment returns as it is able to mitigate capacity saturation by dynamically expanding the model if presented with new strategies. However, FLAIR inherits a shortcoming in AIRL (unstationary reward learning [19]) that makes it prone to catastrophic forgetting [20, 4], reflected in the marginal decline in performance with the task reward correlation and KL divergence as the number of demonstrations increases. Yet, through reward distillation and BCD, FLAIR is able to extract key knowledge from an excess of information to learn well-performing policies that explain each expert's preferences effectively.

3.5.5 Sensitivity Analysis

This section corresponds to the Q7 in the main paper.

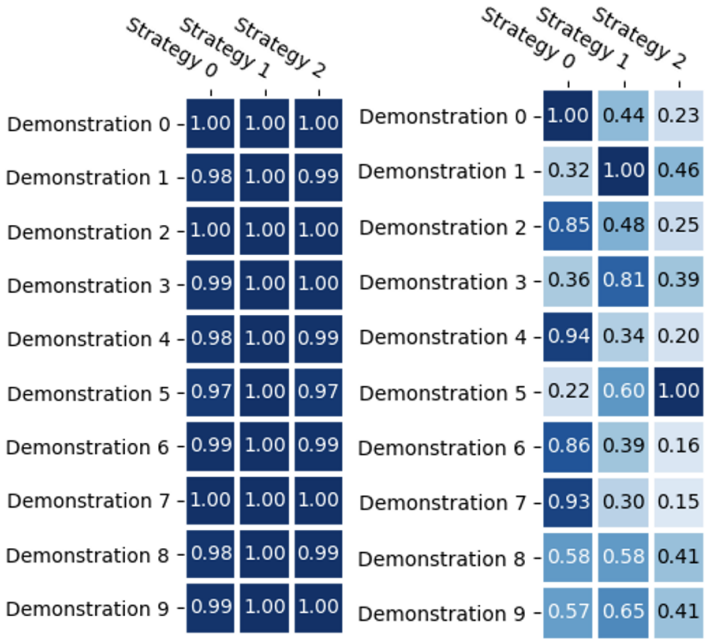

Figure 5: This figure depicts the normalized rewards on demonstrations based on the strategy reward output in Bipedal Walker for FLAIR without BCD (left) and with BCD (right).

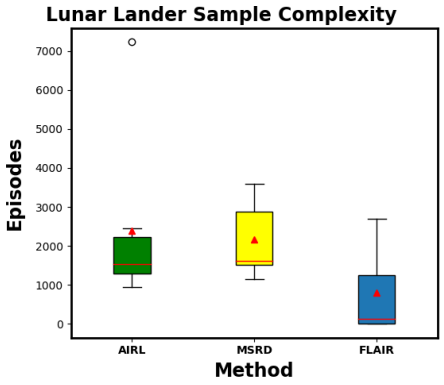

Figure 6: This figure shows the correlation between the estimated task reward with the ground truth task reward for Inverted Pendulum. Each dot is a trajectory. FLAIR achieves a higher task reward correlation.

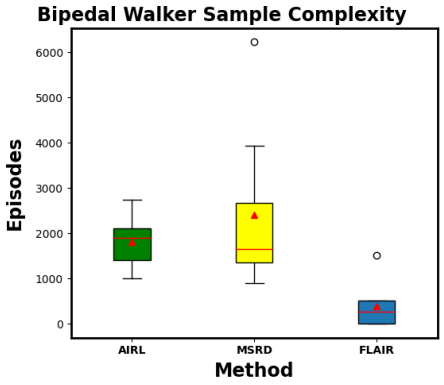

Figure 7: This figure compares the number of episodes needed for AIRL and MSRD to achieve the same Log Likelihood as FLAIR's mixture optimization. The red bar is the median and the red triangle represents the mean.

We generate a Receiver Operating Characteristic (ROC) Analysis by treating the classification of the threshold as our estimation, and the KL divergence comparison between the best mixture policy and the new-strategy as the true signal. FLAIR with a mixture optimization threshold has a high Area Under Curve (0.92) in the ROC Curve for IP, suggesting that there is a wide range of acceptable threshold values that can determine whether to accept the policy mixture without considering training a new policy by AIRL.

Tuning the threshold parameter trades off computational efficiency and modeling accuracy: Creating more strategies would be correlated with higher accuracy but with increased computational costs. As such, there is not a one-size-fits-all. In our experiments, we empirically tune this threshold to maximize the modeling accuracy of FLAIR. For deployment, one could collect a subset of the data, tune the threshold for application-specific criteria, and continue running with this tuned parameter.

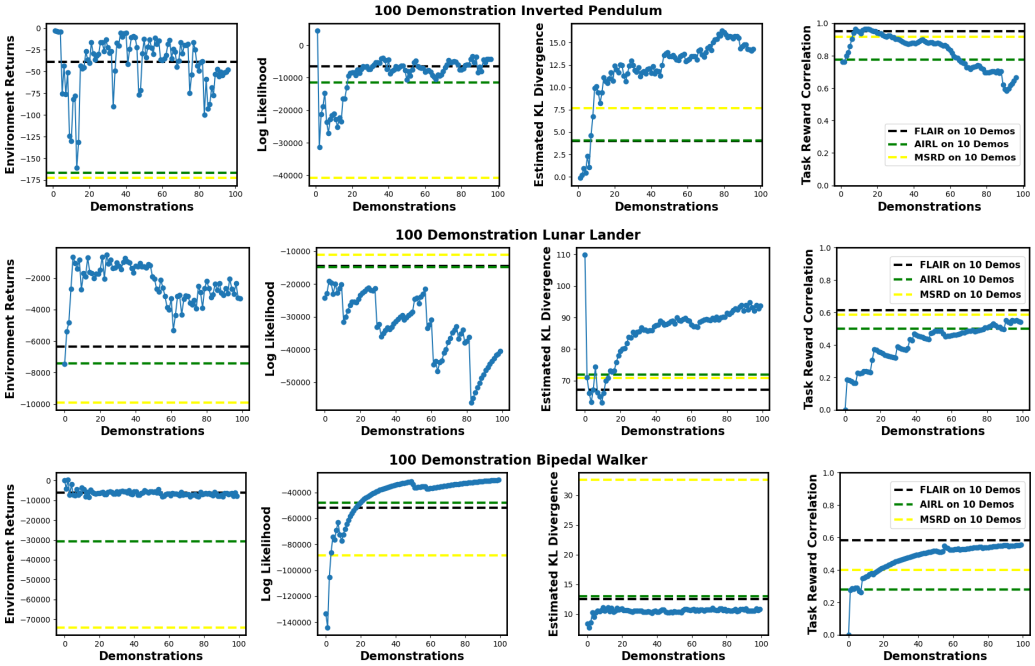

Figure 8: This figure shows the key metrics for FLAIR during the lifelong learning experiment in all three domains. The blue lines show the performance of FLAIR in the scalability experiment for each demonstration. We see FLAIR demonstrates consistently strong performance in environment returns as it is able to mitigate capacity saturation by dynamically expanding the model if presented with new strategies. Note: Estimated KL Divergence is better when it's lower, while for all other metrics, the higher the better.

3.5.6 More Benchmarks

We compare our method, FLAIR, with InfoGAIL [21], a state-of-the-art method for learning from heterogeneous (i.e., diverse) demonstrators. Unlike FLAIR, InfoGAIL is unable to perform incremental learning and must be retrained given any new demonstrations. Averaged over our dataset of ten demonstrations in Inverted Pendulum, FLAIR outperforms InfoGAIL in terms of the log-likelihood of the demonstrators' actions (Infogail: -24504, 276% worse than FLAIR), the policies' rewards (InfoGAIL: -511 reward; 1227% worse than FLAIR), and forward KL divergence (InfoGAIL: 12.32, 207% worse than FLAIR).

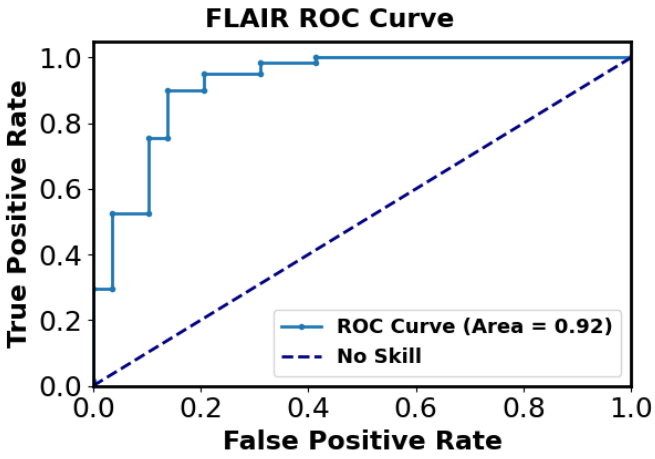

Figure 9: This figure depicts the ROC curve with FLAIR's ability to predict the mixture- vs. new-strategy as the threshold is varied. We see FLAIR with a mixture optimization threshold has a high Area Under Curve (0.92) in the ROC Curve for IP.

References

[1] D. Malyuta. Guidance, Navigation, Control and Mission Logic for Quadrotor Full-cycle Autonomy. Master thesis, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109, USA, Dec. 2017.

[2] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2011.

[3] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[4] T. Ni, H. S. Sikchi, Y. Wang, T. Gupta, L. Lee, and B. Eysenbach. f-irl: Inverse reinforcement learning via state marginal matching. *CoRR*, abs/2011.04709, 2020. URL https://arxiv.org/abs/2011.04709.

[5] L. F. Kozachenko and N. N. Leonenko. Sample estimate of the entropy of a random vector. *Probl. Inf. Transm.*, 23(1-2):95–101, 1987.

[6] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[7] P. I. Frazier. Bayesian optimization. In *Recent Advances in Optimization and Modeling of Contemporary Problems*, pages 255–278. INFORMS, 2018.

[8] N. Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation*, pages 75–102, 2006.

[9] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. URL http://arxiv.org/abs/1606.01540.

[10] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2012.

[11] C. Ericson. *Real-Time Collision Detection*. CRC Press, Inc., USA, 2004.

[12] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

[13] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1329–1338. JMLR.org, 2016.

[14] T. garage contributors. Garage: A toolkit for reproducible reinforcement learning research. https://github.com/rlworkgroup/garage, 2019.

[15] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[16] L. Chen, R. R. Paleja, M. Ghuy, and M. C. Gombolay. Joint goal and strategy inference across heterogeneous demonstrators via reward network distillation. In *Proceedings of the International Conference on Human-Robot Interaction (HRI)*, 2020.

[17] J. A. Damico and D. A. Wolfe. Extended tables of the exact distribution of a rank statistic for all treatments multiple comparisons in one-way layout designs. *Communications in Statistics-Theory and Methods*, 16(8):2343–2360, 1987.

[18] M. Eid, M. Gollwitzer, M. Gollwitzer, and M. Schmitt. *Statistik und Forschungsmethoden*. Beltz (Weinheim [ua]), 2015.

[19] T. Ni, H. Sikchi, Y. Wang, T. Gupta, L. Lee, and B. Eysenbach. f-irl: Inverse reinforcement learning via state marginal matching. *arXiv preprint arXiv:2011.04709*, 2020.

[20] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[21] Y. Li, J. Song, and S. Ermon. Infogail: Interpretable imitation learning from visual demonstrations. *Advances in Neural Information Processing Systems*, 30, 2017.