Fast Lifelong Adaptive Inverse Reinforcement Learning from Demonstrations

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

Abstract: Learning from Demonstration (LfD) approaches empower end-users to teach robots novel tasks via demonstrations of the desired behaviors, democratizing access to robotics. However, current LfD frameworks are not capable of fast adaptation to heterogeneous human demonstrations nor the large-scale deployment in ubiquitous robotics applications. In this paper, we propose a novel LfD framework, Fast Lifelong Adaptive Inverse Reinforcement learning (FLAIR). Our approach (1) leverages learned strategies to construct policy mixtures for fast adaptation to new demonstrations, allowing for quick end-user personalization, (2) distills common knowledge across demonstrations, achieving accurate task inference; and (3) expands its model only when needed in lifelong deployments, maintaining a concise set of prototypical strategies that can approximate all behaviors via policy mixtures. We empirically validate that FLAIR achieves *adaptability* (i.e., the robot adapts to heterogeneous, user-specific task preferences), *efficiency* (i.e., the robot achieves sample-efficient adaptation), and *scalability* (i.e., the model grows sublinearly with the number of demonstrations while maintaining high performance). FLAIR surpasses benchmarks across three control tasks with an average 57% improvement in policy returns and an average 78% fewer episodes required for demonstration modeling using policy mixtures. Finally, we demonstrate the success of FLAIR in a table tennis task and find users rate FLAIR as having higher task ($p < .05$) and personalization ($p < .05$) performance.

Keywords: Personalized Learning, Learning from Heterogeneous Demonstration, Inverse Reinforcement Learning

1 Introduction

Robots are becoming increasingly ubiquitous with recent advancements in Artificial Intelligence (AI), largely due to the success of Deep Reinforcement Learning (DRL) techniques in generating high-performance continuous control behaviors [1, 2, 3, 4, 5, 6, 7, 8]. However, DRL's success heavily relies on sophisticated reward functions designed for each task. These hand-crafted reward functions typically require iterations of fine-tuning and consultation with domain experts to be effective [9]. Instead, Learning from Demonstration (LfD) approaches democratize access to robotics by having users demonstrate the desired behavior to the robot [10], removing the need for per-task reward engineering. While LfD research strives to empower end-users with the ability to program novel behaviors onto robots, we must consider that end-users may adopt varying preferences and strategies in how they complete the same task [11]. An LfD framework that assumes homogeneity across the set of provided demonstrations could cause the robot to fail to infer the accurate intention, resulting in unwanted or even unsafe behavior [12, 13]. On the other hand, embracing individual preferences can help robots achieve better performance and long-term acceptance from humans [14].

While personalization is important for accurate recovery of the demonstrator's behavior, personalization can also prove inefficient if each individual policy must be inferred separately. To avoid

* denotes equal contribution

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

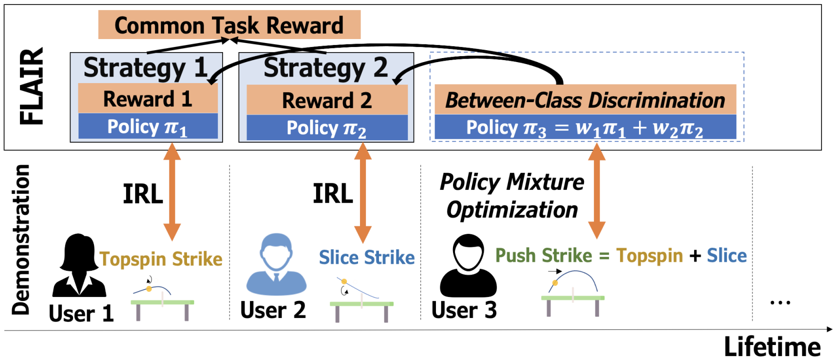


Figure 1: This figure shows an illustration of the lifelong learning process with our proposed method, FLAIR. As each demonstrator performs their strike, FLAIR determines whether the demonstration is novel. If a demonstration can be explained by a *policy mixture* of previously learned strategies, FLAIR accepts the policy mixture without training a new strategy. If the policy mixture is not close to the demonstration, FLAIR creates a new strategy and a prototype policy for the demonstration.

this issue, prior work, MSRD [15], decomposed shared and individual-specific reward information across heterogeneous demonstrations (i.e., demonstrations seeking to accomplish the same task with different styles). While MSRD significantly improves the accuracy and efficiency in personalized policy modeling, the framework must be trained all-at-once and is unable to handle *incremental/lifelong learning*, a more realistic paradigm for LfD real-world applications.

In this work, we develop FLAIR: Fast Lifelong Adaptive Inverse Reinforcement learning. As a running example, consider a series of humans teaching a robot how to play table tennis, a compelling robot learning platform [16, 17, 18]. Users of the robot may have their own preferences for table tennis strikes. As shown in Figure 1, the first user demonstrates a topspin strike, while the second user demonstrates a slice strike. The third user demonstrates a push strike, which could be explained by a composition of known behaviors of the previously seen topspin and slice prototypical behaviors.

Unlike prior LfD algorithms, FLAIR is capable of continually learning and refining a set of prototypical strategies either to (1) efficiently model new demonstrations as mixtures of the acquired prototypes (e.g., the third user in our example) or (2) incorporate a new strategy as a prototype if the strategy is sufficiently unique (e.g., the second user). Consider a real-world example where household robots are delivered to users' homes and the users want to teach those robots skills over the course of the deployment. User demonstrations from different end-users form a demonstration sequence the robots personalize to. In such a lifelong learning scenario, FLAIR autonomously identifies prototypical strategies, distills common knowledge across strategies, and precisely models each demonstration as prototypical strategies or policy mixtures. We show FLAIR accomplishes *adaptivity*, *efficiency*, and *scalability* in LfD tasks in simulated and real robot experiments:

1. **Adaptive Learning**: We display the *adaptivity* of FLAIR by successfully personalizing to heterogeneous demonstrations on three simulated continuous control tasks. FLAIR models demonstrations better than best benchmarks and achieves an average of 57% higher returns on the task.
2. **Efficient Adaptation**: FLAIR is more *efficient*, empirically needing an average of 78% fewer samples to model demonstrations compared to training a new policy.
3. **Lifelong Scalability**: We showcase the *scalability* of FLAIR in a simulated experiment obtaining 100 demonstrations sequentially. FLAIR identifies on average eleven strategies and utilizes *policy mixtures* to achieve a precise representation of each demonstration, providing empirical evidence for FLAIR's ability to learn a compact set of prototypical strategies in lifelong learning.
4. **Robot Demonstration**: We demonstrate FLAIR's ability to successfully leverage *policy mixtures* to achieve stronger task and personalization performance than learning from scratch in a real-world table tennis robot experiment.

2 Related Work

Two common approaches in LfD are to either directly learn a policy, i.e., Imitation Learning (IL), or infer a reward to train a policy, i.e., Inverse Reinforcement Learning (IRL) [19]. IL learns a direct mapping from states to the actions demonstrated [20, 21]. Although a straightforward approach, IL suffers from correspondence matching issues and is not robust to changes in environment dynamics due to its mimicry of the demonstrated behaviors [22, 23]. IRL, on the other hand, infers the demonstrator's latent intent in a more robust and transferable form of a reward function [24].

Although traditional IRL approaches often overlook heterogeneity within demonstrations, there has been recent work that models heterogeneous demonstrations [25, 26, 27, 28, 29, 30]. One intuitive way is to classify demonstrations into homogeneous clusters before applying IRL [11]. The Expectation-Maximization (EM) algorithm also operates on a similar idea and iterates between E-step and M-step, where E-step clusters demonstrations and M-step solves the IRL problem on each cluster [31, 32]. When the number of strategies is unknown, a Dirichlet Process prior [33, 34, 35] or non-parametric methods [36] could be used. In these approaches, each reward function only learns from a portion of the demonstrations, making them prone to the issue of reward ambiguity [15]. Furthermore, these methods assume access to all demonstrations beforehand, which is not realistic for LfD algorithm deployment. We instead consider the more realistic setting of lifelong learning [37], where an agent adapts to new demos through its lifetime and continually builds its knowledge base. One instance to generate such demonstration sequences is through crowd-sourcing (seeking knowledge from a large set of people) [38, 39, 40].

Despite the abundance of previous approaches, few consider the relationship between the policies learned to represent each demonstration. Our method, FLAIR, exploits these relationships to not only model heterogeneous demonstrations (*adaptability*), but do so by creating expressive policy mixtures from previously extracted strategies (*efficiency*), and can scale to model large number of demonstrations utilizing a compact set of strategies (*scalability*).

3 Preliminaries

In this section, we introduce preliminaries on Markov Decision Processes (MDP), Inverse Reinforcement Learning (IRL), and Multi-Strategy Reward Distillation (MSRD).

Markov Decision Process – A MDP, M, is a 6-tuple, $\langle \mathbb{S}, \mathbb{A}, R, T, \gamma, \rho_0 \rangle$. $\mathbb{S}$ and $\mathbb{A}$ are the state and action space, respectively. R is the reward function, meaning the agent is rewarded $R(s)$ in state s. $T(s'|s,a)$ is the probability of transitioning into state s' after taking action a in state s. $\gamma \in (0,1)$ is the temporal discount factor. ρ_0 denotes the initial state probability. A policy, $\pi(a|s)$, represents the probability of choosing an action given the state and is trained to maximize the expected cumulative reward, $\pi^* = \arg\max_\pi \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R(s_t)\right]$, where $\tau = \{s_1, a_1, s_2, a_2, \cdots\}$ is a trajectory.

Inverse Reinforcement Learning – IRL considers an MDP sans reward function (MDP\R) and infers the reward function R based on a set of demonstration trajectories $\mathcal{U} = \{\tau_1, \tau_2, \cdots, \tau_N\}$, where N is the number of demonstrations. Our method is based on Adversarial Inverse Reinforcement Learning (AIRL) [23], which solves the IRL problem with a generative-adversarial setup. The discriminator, D_θ, predicts whether the transition, (s_t, s_{t+1}), belongs to a demonstrator vs. the generator, $\pi_\phi(a|s)$. π_ϕ is trained to maximize the pseudo-reward given by the discriminator.

Multi-Strategy Reward Distillation – MSRD [15] assumes access to the strategy label, $c_{\tau_i} \in \{1, 2, \cdots, M\}$ (M is the number of strategies), for each demonstration, τ_i, and decomposes the per-strategy reward, R_i, for strategy i as a linear combination of a common task reward, R_{Task}, and a strategy-only reward, $R_{\text{S-}i}$. MSRD parameterizes the task reward by θ_{Task} and strategy-only reward by $\theta_{\text{S}-i}$. MSRD takes AIRL as its backbone IRL algorithm, and adds a regularization loss which distills common knowledge into θ_{Task} and only keeps personalized information in $\theta_{\text{S}-i}$. The MSRD loss for the discriminator (the reward) is shown in Equation 1.

$$
\begin{aligned}
L_D = &- \mathbb{E}_{(\tau,c_\tau)\sim\mathcal{U}}\left[\log D_{\theta_{\text{Task}},\theta_{\text{S-}c_\tau}}(s_t, s_{t+1})\right] - \mathbb{E}_{(\tau,c_\tau)\sim\pi_\phi}\left[\log\left(1 - D_{\theta_{\text{Task}},\theta_{\text{S-}c_\tau}}(s_t, s_{t+1})\right)\right] \\
&+ \alpha\mathbb{E}_{(\tau,c_\tau)\sim\pi_\phi}\left[||R_{\text{S-}c_\tau}(s_t)||_2\right]
\end{aligned}
\tag{1}
$$

4 Method

In this section, we start by introducing the problem setup and notations. We then provide an overview of FLAIR, and its two key components: *policy mixture* and *between-class discrimination*.

We consider a lifelong learning from heterogeneous demonstration process where demonstrations arrive in sequence, as illustrated in Figure 1. We denote the i-th arrived demonstration as τ_i. Unlike prior work, FLAIR does not assume access to the strategy label, c_{τ_i}. Similar to MSRD, FLAIR learns a shared task reward $R_{\theta_{\text{Task}}}$, strategy rewards $R_{\theta_{\text{S-}j}}$, and policies corresponding to each strategy π_{ϕ_j}. We define the number of prototype strategies created by FLAIR till demonstration τ_i as M_i, and $\eta_R(\tau) = \sum_{t=1}^{\infty} \gamma^{t-1} R_\theta(s_t)$ as trajectory τ's discounted cumulative reward with the reward function R_θ. The goal of the problem is to accurately model each demonstration sequentially with as few environment samples as possible. Note that learning from sequential demonstrations is not a requirement of FLAIR but rather a feature in comparison to batch-based methods where all demonstrations must be available before the learning could start.

4.1 Fast Lifelong Adaptive Inverse Reinforcement Learning (FLAIR)

In our lifelong learning problem setup, when a new demonstration τ_i becomes available, we seek to accomplish two goals: a) design a policy that solves the task while personalizing to the demonstration (i.e., the objective in personalized LfD), and b) incorporate knowledge from the demonstration to facilitate efficient and scalable adaptation to future users (i.e., the characteristics required for a lifelong LfD framework). We present our method, FLAIR, in pseudocode in Algorithm 1.

Algorithm 1: FLAIR

```
Input  : Demonstration modeling quality threshold $\epsilon$
1  $M_0 = 0$, MixtureWeights=[], $m$=[]
2  while lifetime learning from heterogeneous demonstration do
3      Obtain demonstration $\tau_i$
4      $\vec{w}_i, D^{\text{mix}}_{\text{KL}} \leftarrow$ PolicyMixtureOptimization($\tau_i, \{\pi_{\phi_j}\}_{j=1}^{M_i}$)
5      if $D^{mix}_{KL} < \epsilon$ then
6          MixtureWeights[i]$\leftarrow \vec{w}_i$, $M_{i+1} \leftarrow M_i$
7      else
8          $\pi_{\text{new}}, R_{\theta_{\text{S-}(M_i+1)}} \leftarrow$ AIRL($\tau_i$)
9          $D^{\text{new}}_{\text{KL}} \leftarrow \mathbb{E}_{\tau\sim\pi_{\text{new}}} D_{\text{KL}}(\tau_i, \tau)$
10         if $D^{mix}_{KL} < D^{new}_{KL}$ then
11             MixtureWeights[i]$\leftarrow \vec{w}_i$, $M_{i+1} \leftarrow M_i$
12         else
13             $M_{i+1} \leftarrow M_i + 1$
14             $m_{M_{i+1}} \leftarrow i$
15             MixtureWeights[i]$\leftarrow [\underbrace{0, 0, \cdots, 0}_{M_i \text{ zeros}}, 1]$
16     Update $R_{\theta_{\text{Task}}}, R_{\theta_{\text{S-}j}}, \pi_{\phi_j}$ by Between-Class Discrimination and MSRD
```

To accomplish these goals, FLAIR decides whether to explain a new demonstration with previously learned policies (a highly efficient approach), or create a new strategy from scratch (a fallback technique). In the first case, FLAIR attempts to explain the new demonstration, τ_i, by constructing *policy mixtures* with previously learned strategies according to the demonstration recovery objective (line 4). If the trajectory generated by the mixture is close to the demonstration (evidenced by the KL-divergence between the *policy mixture* trajectory and the demonstration state distributions falling under a threshold, ϵ), FLAIR adopts the mixture without considering creating a new strategy (line 6). Since the *policy mixture* optimization (details in Section 4.2) is more sample efficient than the AIRL training-from-scratch, FLAIR can bypass the computationally expensive new-strategy training (line 8) if the mixture provides a high-quality recovery of the demonstrated behavior. This procedure results in an *efficient* policy inference.

If the mixture does not meet the quality threshold, ϵ, FLAIR trains a new strategy by AIRL with τ_i and compares the quality of the new policy to the *policy mixture* (Lines 8-10). If the mixture performs better, we accept the mixture weights (line 11). If the new strategy performs better, we accept the new strategy as a new prototype and update our reward and policy models (accordingly, in Line 13, we increment the number of strategies by one). Further, we call the demonstration, τ_i, the “pure” demonstration for strategy M_{i+1}, meaning strategy M_{i+1} represents demonstration τ_i (line 14). As such, the mixture weight for τ_i is a one-hot vector on strategy M_{i+1} (line 15).

To effectively maintain a knowledge base, we propose a novel training signal named *Between-Class Discrimination* (BCD). BCD trains each strategy reward to capture the fact that each demonstration has a certain percentage of the strategy. In the table tennis example (Figure 1), the third user's behavior is a mixture of the topspin and the slice, indicating topspin and slice strategy rewards should be apparent in the third demonstration. BCD encourages the two strategy rewards to give partial rewards to the third demonstration. In addition to BCD, FLAIR also optimizes MSRD loss (Equation 1) for all strategies with their corresponding pure demonstrations, and updates the generator policies based on the learned reward (line 16).

4.2 Policy Mixture Optimization

To achieve efficient personalization for a new demonstration τ_i (Line 4 of Algorithm 1), we construct a *policy mixture* with a linear geometric combination of existing policies $\pi_1, \pi_2, \cdots, \pi_{M_i}$ (Equation 2), where $w_{i,j} \geq 0$ are learned weights such that: $\sum_{j=1}^{M_i} w_{i,j} = 1$.

$$\pi_{\vec{w}_i}(s) = \sum_{j=1}^{M_i} w_{i,j} a_j, \quad a_j \sim \pi_j(s) \tag{2}$$

As the ultimate goal of demonstration modeling is to recover the demonstrated behavior, we optimize the linear weights, $\vec{w}_i$, to minimize the divergence between the trajectory induced by the mixture policy and the demonstration, shown in Equation 3.

$$\underset{\vec{w}_i}{\text{minimize}}\, \mathbb{E}_{\tau \sim \pi_{\vec{w}_i}} \left[D_{\text{KL}}(\tau_i, \tau) \right] \tag{3}$$

Specifically, we choose Kullback-Leibler divergence (KL-divergence) [41] on the state marginal distributions of trajectories in our implementation. We estimate the state distribution within a trajectory by the kernel density estimator [42]. More details can be found in supplementary.

Since the trajectory generation process is non-differentiable, we seek a non-gradient-based optimizer to solve Equation 3. Specifically, FLAIR utilizes a naïve, random optimization method; it generates random weight vectors $\vec{w}_i$, evaluates Equation 3, and chooses the weight that achieves the minimization. Empirically, we find random optimization outperforms various other optimization methods for FLAIR. Please see the supplementary for a detailed comparison.

4.3 Between-Class Discrimination

Although MSRD distills the task reward from heterogeneous demonstrations, it does not encourage the strategy rewards to encode distinct strategic preferences. MSRD also requires access to ground-truth strategy labels for all demonstrations, which limits scalability. In order to increase the strategy reward's discriminability between different strategies, we propose a novel learning objective named *Between-Class Discrimination* (BCD). BCD enforces the strategy reward to correctly discriminate mixture demonstrations from the pure demonstration: 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. A proof is provided in the supplementary.

Lemma 1. *Under the maximum entropy principal,*

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

Thus, we enforce the relationship of strategy rewards, S-j, evaluated on pure strategy demonstration, τ_{m_j}, and mixture strategy demonstration, τ_i with mixture weight $w_{i,j}$, as shown in Equation 4.

$$L_{\text{BCD}}(\theta^{\text{S-}j}) = \sum_{i=1}^{n} \left(e^{\eta \theta_{\text{S-}j}(\tau_i)} - w_{i,j} e^{\eta \theta_{\text{S-}j}(\tau_{m_j})} \right)^2 \tag{4}$$

An extreme case of BCD loss is when τ_i is the pure demonstration for another strategy, k (i.e., $m_k = i$). In this case, $w_{i,j} = 0$ (as τ_i is purely on strategy k), and Equation 4 encourages the strategy j's reward to give as low as possible reward to τ_i. In turn, strategy rewards gain better discrimination between different strategies, facilitating more robust strategy reward learning, and contributing to the success in lifelong learning.

Table 1: This table shows learned policy metrics between AIRL, MSRD, and FLAIR. The higher environment returns / lower estimated KL divergence / higher strategy rewards, the better.

Domains	Inverted Pendulum			Lunar Lander			Bipedal Walker		
Methods	AIRL	MSRD	FLAIR	AIRL	MSRD	FLAIR	AIRL	MSRD	FLAIR
Environment Returns	-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	$\mathbf{4.01}^{**}$	72.0	70.9	$\mathbf{67.2}^{**}$	13.0	32.6	$\mathbf{12.1}^{**}$
Strategy Rewards	-5.73	-6.22	$-\mathbf{1.23}$	-12.67	-20.26	$-\mathbf{4.19}^{*}$	-5.31	-29.82	$-\mathbf{4.22}^{**}$

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

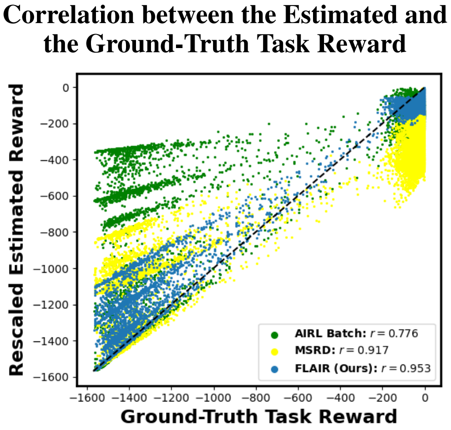


Figure 2: This figure shows the correlation between the estimated task reward with the ground truth task reward for Inverted Pendulum. Each dot is a trajectory. FLAIR achieves a higher task reward correlation.

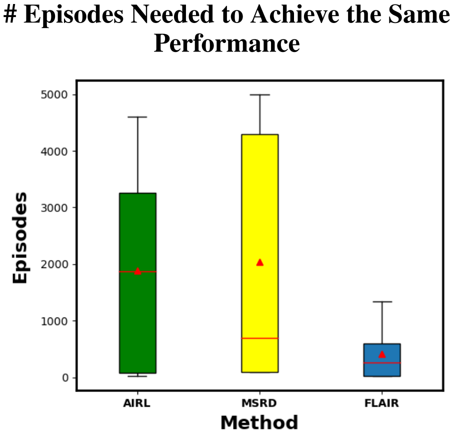


Figure 3: This figure compares the number of episodes needed for AIRL and MSRD to achieve the same Log Likelihood as FLAIR's mixture optimization. The red bar is the median and the red triangle represents the mean.

5 Results

In this section, we show that FLAIR achieves *adaptability*, *efficiency*, and *scalability* in modeling heterogeneous demonstrations. We test FLAIR on three simulated continuous control environments in OpenAI Gym [43]: Inverted Pendulum (IP) [44], Lunar Lander (LL), and Bipedal Walker (BW) [45]. We generate a collection of heterogeneous demonstrations by jointly optimizing an environment and diversity reward with DIAYN [46]. For all experiments excluding the scalability study, we use ten demonstrations. We compare FLAIR with AIRL and MSRD by running three trials of each method. More experiment details and statistical test results are provided in the supplementary.

5.1 Adaptability

Q1: ***Can FLAIR's policy mixtures perform well at the task?*** From ten demonstrations, FLAIR created 6.3 ± 0.5 strategies (average and standard deviation across three trials) in IP, 5.3 ± 1.2 in LL, and 3.3 ± 0.5 in BW. FLAIR's learned policies including *policy mixtures* are significantly more successful at the task (row "Environment Returns" in Table 1), outperforming benchmarks in task performance with 77% higher returns in IP, 14% in LL, and 80% in BW than best baselines.

Q2: ***How closely does the policy recover the strategic preference?*** Qualitatively, we find that FLAIR learns policies and policy mixtures that closely resemble their respective strategies, visualized in policy renderings (videos available in supplementary). We further show that FLAIR is statistically significantly better in estimated KL divergence than AIRL (average 4% better) and MSRD (average 18% better), shown in row "Estimated KL Divergence" in Table 1, where KL divergence is evaluated between policy rollouts and demonstration state distributions. We further tested the learned policies' performance on ground-truth strategy reward functions given by DIAYN. The results on row "Strategy Rewards" illustrate FLAIR's better adherence to the demonstrated strategies.

Q3. ***How well does the task reward model the ground truth environment reward?*** We evaluate the learned task reward functions by calculating the correlation between estimated task rewards

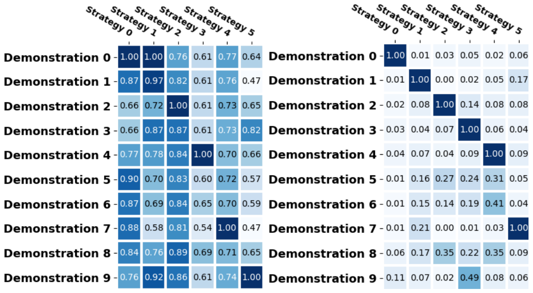


Figure 4: This figure depicts the normalized strategy rewards on demonstrations in IP for FLAIR without BCD (left) and with BCD (right).

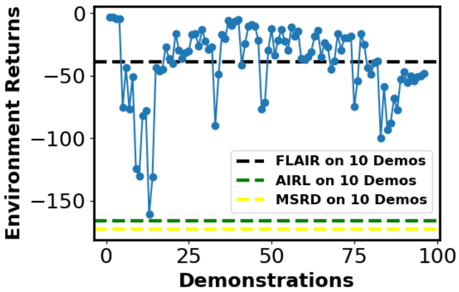


Figure 5: This figure plots the returns of FLAIR policies in a 100 demonstration experiment in Inverted Pendulum.

with ground-truth environment rewards. We construct a test dataset of 10,000 trajectories with multiple policies obtained during the "DIAYN+env reward" training. FLAIR's task reward achieves $r = 0.953$ in IP (shown in Figure 2), $r = 0.614$ in LP, and $r = 0.582$ in BW, with an average 18% higher correlation than best baselines and statistical significance compared with AIRL and MSRD.

Q4. ***Can the learned strategy rewards discriminate between different strategies?*** We analyze the learned strategy rewards on heterogeneous demonstrations (shown in Figure 4 right). We find that each strategy reward of FLAIR identifies the corresponding pure demonstration (Demonstrations 0-4,7) alongside the mixtures (Demonstrations 5-6, 8-9). In contrast, the strategy rewards learned without BCD (Figure 4 left) do not distinguish between different strategies. This ablation study also finds that FLAIR with BCD achieves 70% better environment returns and 10% better KL divergence than FLAIR without BCD (additional metrics available in supplementary). The qualitative results in Figure 4 and quantitative results in supplementary together provide empirical evidence that FLAIR with BCD can train strategy rewards to better identify different strategies.

5.2 Efficiency & Scalability

Q5. ***Can FLAIR's mixture optimization model demonstrations more efficiently than learning a new policy?*** We study the number of episodes needed by FLAIR's mixture optimization and AIRL/MSRD policy training to achieve the same modeling performance of demonstrations. The result in Figure 3 demonstrates FLAIR requires 77% fewer episodes to achieve a high log likelihood of the demonstration relative to AIRL and 79% fewer episodes than MSRD. Three (out of ten) of AIRL's learned policies and four of MSRD's learned policies failed to reach the same performance as FLAIR even given 10,000 episodes, and are thus left out in Figure 3. By reusing learned policies through *policy mixtures*, FLAIR explains the demonstration in an efficient manner.

Q6. ***Can FLAIR's success continue in a larger-scale LfD problem?*** We generate 95 mixtures with randomized weights from 5 prototypical policies for a total of 100 demonstrations to test how well FLAIR scales. We train FLAIR sequentially on the 100 demonstrations and observe FLAIR learns a concise set of 17 strategies in IP, 10 in LL, and 6 in BW that capture the scope of behaviors while also achieving a consistently strong task performance (Figure 5 and supplementary). We find FLAIR maintains or even exceeds its 10-demonstration performance when scaling up to 100 demonstrations.

5.3 Sensitivity Analysis

Q7. ***How sensitive is FLAIR's mixture optimization threshold?*** We study the classification skill of the mixture optimization threshold and find it has a strong ability to classify whether a demonstration should be included as a mixture or a new strategy. A Receiver Operating Characteristic (ROC) Analysis suggests FLAIR with thresholding achieves a high Area Under Curve (0.92) in the ROC Curve for IP; the specific choice of the threshold depends on the performance/efficiency tradeoff the user/application demands (see the ROC Curve and threshold selection methodology in the supplementary).

5.4 Discussion

The above findings show that our algorithm, FLAIR, sets a new state-of-the-art in personalized LfD. Across several domains, FLAIR achieves better demonstration recovery compared to the baselines.

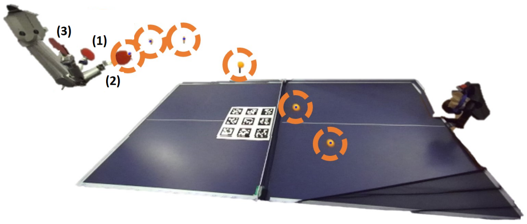


Figure 6: This figure illustrates a topspin and slice mixture policy (a push-like behavior). The robot moves from location (1) to (2) and (3).

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

* Significance of $p < 0.05$

Table 2: This table depicts policy metrics between FLAIR's best mixtures, FLAIR's worst mixtures, and learning-from-scratch policies. The scores are shown as averages ± standard deviations across 28 participants. Bold denotes the highest scores.

Not only can FLAIR more accurately infer the task reward and associated policies, but FLAIR is also able to perform policy inference with much fewer environmental interactions. These characteristics make FLAIR amenable to lifelong LfD, resulting in one of the first LfD frameworks that can handle sequential demonstrations without requiring retraining the entire model.

6 Real-World Robot Case Study: Table Tennis

We perform a real-world robot table tennis experiment where we leverage FLAIR's *policy mixtures* to model user demonstrations. An illustration of an example policy mixture is shown in Figure 6 (more videos are available in supplementary).

We first collect demonstrations of four different table tennis strategies (i.e. push, slice, topspin, and lob) via kinesthetic teaching from one human participant who is familiar with the WAM robot but does not have prior experience providing demonstrations for table tennis strikes. After training the four prototypical strategy policies, we assess how well FLAIR can use policy mixtures to model new user demonstrations. To do so, we collected demonstrations from 28 participants by instructing them to demonstrate five repeats of their preferred PingPong strike. We utilize this data and compare three LfD approaches for learning a robot policy: 1) the best policy mixture identified by FLAIR, 2) a learning-from-scratch approach, and 3) an adversarially optimized policy mixture (i.e., minimize the KL divergence between the rollout and the demonstration). We then have users/participants observe the robot executing these policies in a random order. Using ad hoc Likert scale questionnaires (see supplementary), participants evaluate the robot's performance in (i) accomplishing the task and (ii) doing so according to the user's preferences. Table 2 shows that FLAIR's best mixture outperforms both the worst mixture (task score: $p < .01$, strategy score: $p < .001$) and the learning-from-scratch policy (task score: $p < .001$, strategy score: $p < .05$), demonstrating FLAIR's ability to optimize policy mixtures that succeed in the task and fit user's preferences. Full statistical testing results are available in the supplementary.

7 Conclusion, Limitations, & Future Work

In this paper, we present FLAIR, a fast lifelong adaptive LfD framework. In benchmarks against AIRL and MSRD, we demonstrate FLAIR's *adaptability* to novel personal preferences and *efficiency* by utilizing policy mixtures. We also illustrate FLAIR's *scalability* in how it learns a concise set of strategies to solve the problem of modeling a large number of demonstrations.

Some limitations of FLAIR are 1) if the initial demonstrations are not representative of a diverse set of strategies, the ability to effectively model a large number of demonstrations may be impacted due to the biased task reward and non-diverse prototypical policies; 2) FLAIR's learned rewards are non-stationary (the learned reward function changes due to the adversarial training paradigm), a property inherited from AIRL, and hence could suffer from catastrophic forgetting. For the first limitation, we could pre-train FLAIR with representative demonstrations before deployment to avoid biasing the task reward and to provide diverse prototypical policies. Another potential direction is to adopt a "smoothing"-based approach over a "filtering" method. The smoothing-based approach would allow new prototypical policies to model previous demonstrations, relaxing the diversity assumptions on initial policies. We are also interested in studying how to recover a minimally spanning strategy set that could explain all demonstrations. For the second limitation, we seek to leverage IRL techniques that yield stationary reward for the FLAIR framework. f-IRL [47] could be a potential candidate, but is notoriously slow due to the iterative reward training and policy training.

Acknowledgments

We wish to thank our reviewers for their valuable feedback in revising our manuscript. This work was sponsored by NSF grant IIS-2112633, MIT Lincoln Laboratory grant FA8702-15-D-0001, and Office of Naval Research grant N00014-19-1-2076.

References

[1] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[3] R. Paleja, Y. Niu, A. Silva, C. Ritchie, S. Choi, and M. Gombolay. Learning interpretable, high-performing policies for continuous control problems. *arXiv preprint arXiv:2202.02352*, 2022.

[4] E. Seraj, Z. Wang, R. Paleja, D. Martin, M. Sklar, A. Patel, and M. Gombolay. Learning efficient diverse communication for cooperative heterogeneous teaming. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 1173–1182, 2022.

[5] A. Silva, N. Moorman, W. Silva, Z. Zaidi, N. Gopalan, and M. Gombolay. Lancon-learn: Learning with language to enable generalization in multi-task manipulation. *IEEE Robotics and Automation Letters*, 7(2):1635–1642, 2022. doi:10.1109/LRA.2021.3139667.

[6] E. Seraj, L. Chen, and M. C. Gombolay. A hierarchical coordination framework for joint perception-action tasks in composite robot teams. *IEEE Transactions on Robotics*, 38(1):139–158, 2021.

[7] S. Konan, E. Seraj, and M. Gombolay. Iterated reasoning with mutual information in cooperative and byzantine decentralized teaming. *arXiv preprint arXiv:2201.08484*, 2022.

[8] S. G. Konan, E. Seraj, and M. Gombolay. Contrastive decision transformers. In *6th Annual Conference on Robot Learning*, 2022.

[9] L. Matignon, G. J. Laurent, and N. Le Fort-Piat. Reward function and initial values: Better choices for accelerated goal-directed reinforcement learning. In *Artificial Neural Networks – ICANN 2006*, pages 840–849, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[10] S. Schaal. Learning from demonstration. In M. C. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1997. URL https://proceedings.neurips.cc/paper/1996/file/68d13cf26c4b4f4f932e3eff990093ba-Paper.pdf.

[11] S. Nikolaidis, R. Ramakrishnan, K. Gu, and J. Shah. Efficient model learning from joint-action demonstrations for human-robot collaborative tasks. In *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 189–196. IEEE, 2015.

[12] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120, Dec. 2014. doi:10.1609/aimag.v35i4.2513. URL https://ojs.aaai.org/index.php/aimagazine/article/view/2513.

[13] E. F. Morales and C. Sammut. Learning to fly by combining reinforcement learning with behavioural cloning. In *Proceedings of the International Conference on Machine Learning (ICML)*, page 76, 2004.

[14] I. Leite, C. Martinho, and A. Paiva. Social robots for long-term interaction: A survey. *International Journal of Social Robotics*, 5(2):291–308, Apr 2013. ISSN 1875-4805. doi:10.1007/s12369-013-0178-y. URL https://doi.org/10.1007/s12369-013-0178-y.

[15] 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.

[16] K. Mülling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *Proceedings of the International Journal of Robotics Research (IJRR)*, 32(3):263–279, 2013.

[17] K. Muelling, A. Boularias, B. Mohler, B. Schölkopf, and J. Peters. Learning strategies in table tennis using inverse reinforcement learning. *Biological cybernetics*, 108(5):603–619, 2014.

[18] L. Chen, R. Paleja, and M. Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. In *Proceedings of Conference on Robot Learning (CoRL)*, 2020.

[19] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3, 2020.

[20] A. Chella, H. Dindo, and I. Infantino. A cognitive framework for imitation learning. *Robotics and Autonomous Systems*, 54(5):403–408, 2006. ISSN 0921-8890. doi:https://doi.org/10.1016/j.robot.2006.01.008. URL https://www.sciencedirect.com/science/article/pii/S0921889006000200. The Social Mechanisms of Robot Programming from Demonstration.

[21] 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.

[22] P. de Haan, D. Jayaraman, and S. Levine. Causal confusion in imitation learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/947018640bf36a2bb609d3557a285329-Paper.pdf.

[23] 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.

[24] N. D. Daw and P. Dayan. The algorithmic anatomy of model-based evaluation. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1655):20130478, 2014.

[25] A. Y. Ng, S. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[26] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*. ACM, 2004.

[27] D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, page 2586–2591. Morgan Kaufmann Publishers Inc., 2007.

[28] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the National Conference on Artificial intelligence (AAAI)*, pages 1433–1438, 2008.

[29] B. D. Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, Carnegie Mellon University, 2010.

[30] R. Paleja, A. Silva, L. Chen, and M. Gombolay. Interpretable and personalized apprenticeship scheduling: Learning interpretable scheduling policies from heterogeneous user demonstrations. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6417–6428. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/477bdb55b231264bb53a7942fd84254d-Paper.pdf.

[31] M. Babes-Vroman, V. Marivate, K. Subramanian, and M. Littman. Apprenticeship learning about multiple intentions. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 897–904, 01 2011.

[32] G. Ramponi, A. Likmeta, A. M. Metelli, A. Tirinzoni, and M. Restelli. Truly batch model-free inverse reinforcement learning about multiple intentions. In S. Chiappa and R. Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2359–2369. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/ramponi20a.html.

[33] J. Almingol, L. Montesano, and M. Lopes. Learning multiple behaviors from unlabeled demonstrations in a latent controller space. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 136–144, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[34] A. Bighashdel, P. Meletis, P. Jancura, and G. Dubbelman. Deep adaptive multi-intention inverse reinforcement learning. In N. Oliver, F. Pérez-Cruz, S. Kramer, J. Read, and J. A. Lozano, editors, *Machine Learning and Knowledge Discovery in Databases. Research Track*, pages 206–221, Cham, 2021. Springer International Publishing.

[35] J. Choi and K.-e. Kim. Nonparametric bayesian inverse reinforcement learning for multiple reward functions. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/140f6969d5213fd0ece03148e62e461e-Paper.pdf.

[36] S. Rajasekaran, J. Zhang, and J. Fu. Inverse reinforce learning with nonparametric behavior clustering. *arXiv preprint arXiv:1712.05514*, 2017.

[37] J. A. Mendez, S. Shivkumar, and E. Eaton. Lifelong inverse reinforcement learning. In *NeurIPS*, pages 4507–4518, 2018.

[38] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.

[39] A. Mandlekar, J. Booher, M. Spero, A. Tung, A. Gupta, Y. Zhu, A. Garg, S. Savarese, and L. Fei-Fei. Scaling robot supervision to hundreds of hours with roboturk: Robotic manipulation dataset through human reasoning and dexterity. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1048–1055. IEEE, 2019.

[40] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[41] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.

[42] 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.

[43] 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.

[44] 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.

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

[46] 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.

[47] 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.

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 ± 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

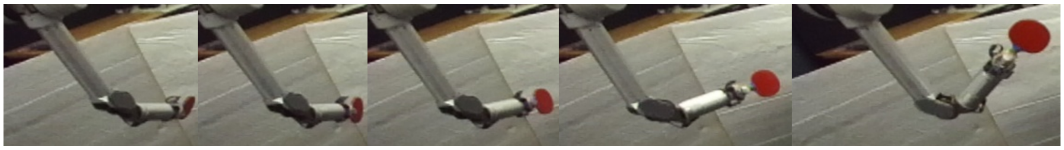


Figure 1: Frames of a 90% **Topspin +** 10% **Lob** mixture (full video available in the supplementary video). It is seen in the motion that the paddle starts in a tilted motion (standard for a topspin strike) and curves upward to return a ball with high curvature (typically of a lob strike). The location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

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

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	$\mathbf{4.01}$	72.0	70.9	$\mathbf{67.2}$	13.0	32.6	$\mathbf{12.1}$
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 FLAIR vs.	Friedman	Nemenyi AIRL	Nemenyi MSRD	Friedman	Nemenyi AIRL	Nemenyi MSRD	Friedman	Nemenyi AIRL	Nemenyi 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 $p < 0.01$

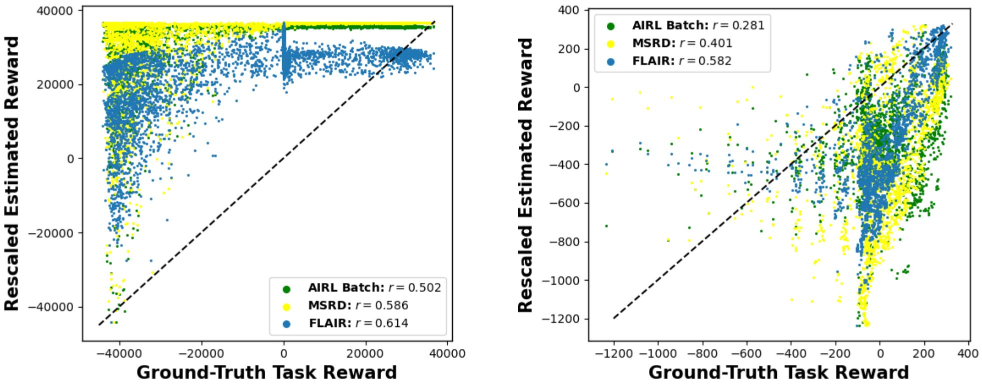


Figure 2: This figure shows the correlation between the estimated task reward with the ground truth task reward for Lunar Lander and Bipedal Walker respectively. Each dot is a trajectory. FLAIR achieves a higher task reward correlation than baselines. The diagonal dashed lines denote perfect correlation.

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

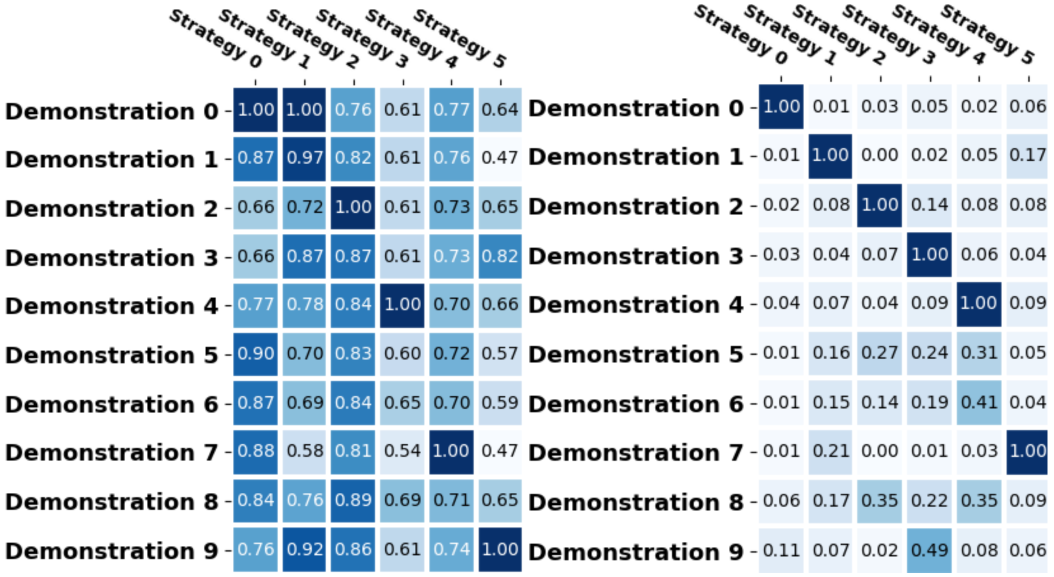
This section corresponds to the Q3 in the main paper.

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



	Strategy 0	Strategy 1	Strategy 2	Strategy 3	Strategy 4	Strategy 5
Demonstration 0	1.00	1.00	0.76	0.61	0.77	0.64
Demonstration 1	0.87	0.97	0.82	0.61	0.76	0.47
Demonstration 2	0.66	0.72	1.00	0.61	0.73	0.65
Demonstration 3	0.66	0.87	0.87	0.61	0.73	0.82
Demonstration 4	0.77	0.78	0.84	1.00	0.70	0.66
Demonstration 5	0.90	0.70	0.83	0.60	0.72	0.57
Demonstration 6	0.87	0.69	0.84	0.65	0.70	0.59
Demonstration 7	0.88	0.58	0.81	0.54	1.00	0.47
Demonstration 8	0.84	0.76	0.89	0.69	0.71	0.65
Demonstration 9	0.76	0.92	0.86	0.61	0.74	1.00

	Strategy 0	Strategy 1	Strategy 2	Strategy 3	Strategy 4	Strategy 5
Demonstration 0	1.00	0.01	0.03	0.05	0.02	0.06
Demonstration 1	0.01	1.00	0.00	0.02	0.05	0.17
Demonstration 2	0.02	0.08	1.00	0.14	0.08	0.08
Demonstration 3	0.03	0.04	0.07	1.00	0.06	0.04
Demonstration 4	0.04	0.07	0.04	0.09	1.00	0.09
Demonstration 5	0.01	0.16	0.27	0.24	0.31	0.05
Demonstration 6	0.01	0.15	0.14	0.19	0.41	0.04
Demonstration 7	0.01	0.21	0.00	0.01	0.03	1.00
Demonstration 8	0.06	0.17	0.35	0.22	0.35	0.09
Demonstration 9	0.11	0.07	0.02	0.49	0.08	0.06

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.

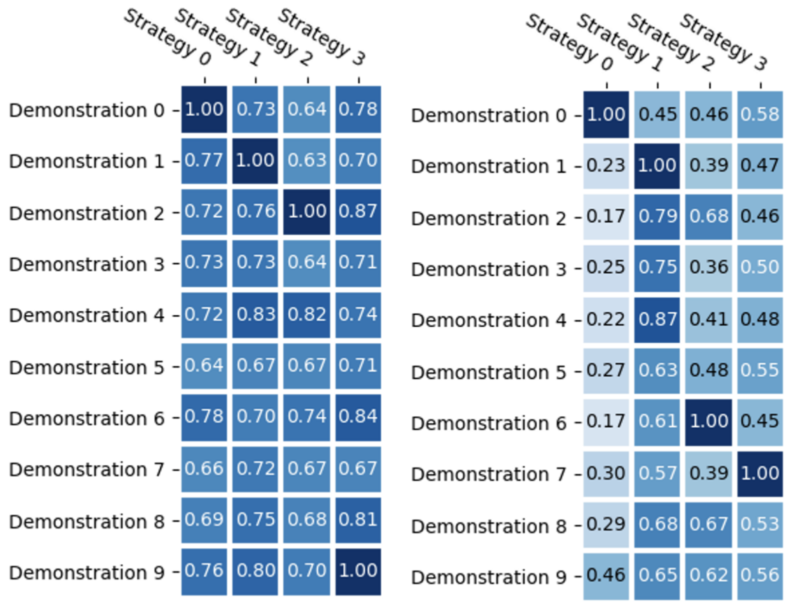
3.5.3 Strategy Reward Learning & BCD Ablation

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-



	Strategy 0	Strategy 1	Strategy 2	Strategy 3
Demonstration 0	1.00	0.73	0.64	0.78
Demonstration 1	0.77	1.00	0.63	0.70
Demonstration 2	0.72	0.76	1.00	0.87
Demonstration 3	0.73	0.73	0.64	0.71
Demonstration 4	0.72	0.83	0.82	0.74
Demonstration 5	0.64	0.67	0.67	0.71
Demonstration 6	0.78	0.70	0.74	0.84
Demonstration 7	0.66	0.72	0.67	0.67
Demonstration 8	0.69	0.75	0.68	0.81
Demonstration 9	0.76	0.80	0.70	1.00

	Strategy 0	Strategy 1	Strategy 2	Strategy 3
Demonstration 0	1.00	0.45	0.46	0.58
Demonstration 1	0.23	1.00	0.39	0.47
Demonstration 2	0.17	0.79	0.68	0.46
Demonstration 3	0.25	0.75	0.36	0.50
Demonstration 4	0.22	0.87	0.41	0.48
Demonstration 5	0.27	0.63	0.48	0.55
Demonstration 6	0.17	0.61	1.00	0.45
Demonstration 7	0.30	0.57	0.39	1.00
Demonstration 8	0.29	0.68	0.67	0.53
Demonstration 9	0.46	0.65	0.62	0.56

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

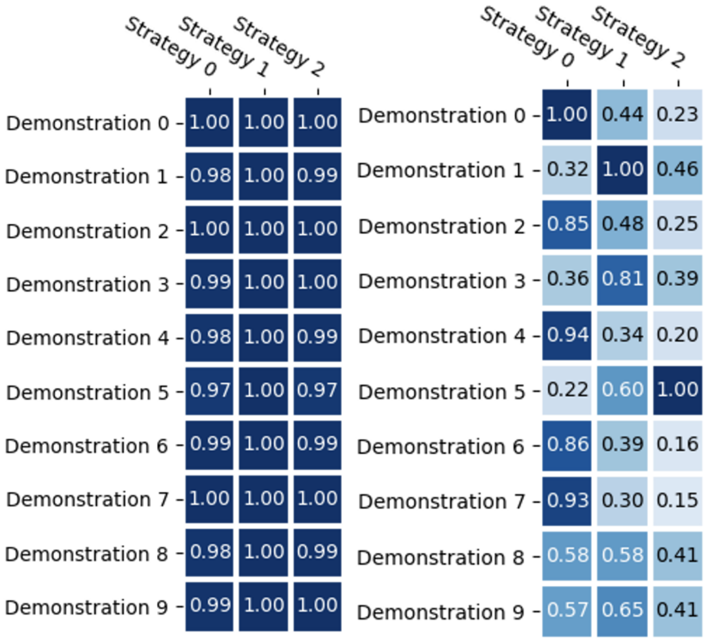
This section provides more details to Q6 of the main paper.

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.



	Strategy 0	Strategy 1	Strategy 2
Demonstration 0	1.00	1.00	1.00
Demonstration 1	0.98	1.00	0.99
Demonstration 2	1.00	1.00	1.00
Demonstration 3	0.99	1.00	1.00
Demonstration 4	0.98	1.00	0.99
Demonstration 5	0.97	1.00	0.97
Demonstration 6	0.99	1.00	0.99
Demonstration 7	1.00	1.00	1.00
Demonstration 8	0.98	1.00	0.99
Demonstration 9	0.99	1.00	1.00

	Strategy 0	Strategy 1	Strategy 2
Demonstration 0	1.00	0.44	0.23
Demonstration 1	0.32	1.00	0.46
Demonstration 2	0.85	0.48	0.25
Demonstration 3	0.36	0.81	0.39
Demonstration 4	0.94	0.34	0.20
Demonstration 5	0.22	0.60	1.00
Demonstration 6	0.86	0.39	0.16
Demonstration 7	0.93	0.30	0.15
Demonstration 8	0.58	0.58	0.41
Demonstration 9	0.57	0.65	0.41

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).

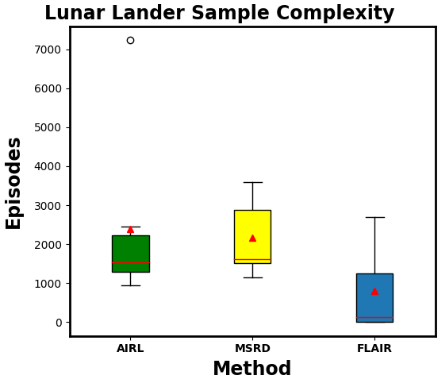


Figure 6: This figure shows the correlation between the estimated task reward with the ground truth task reward for Inverted Pendulum. Each dot is a trajectory. FLAIR achieves a higher task reward correlation.

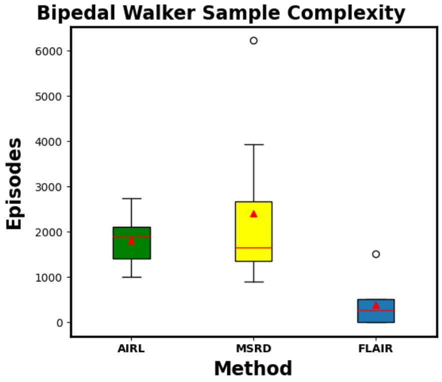


Figure 7: This figure compares the number of episodes needed for AIRL and MSRD to achieve the same Log Likelihood as FLAIR's mixture optimization. The red bar is the median and the red triangle represents the mean.

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.

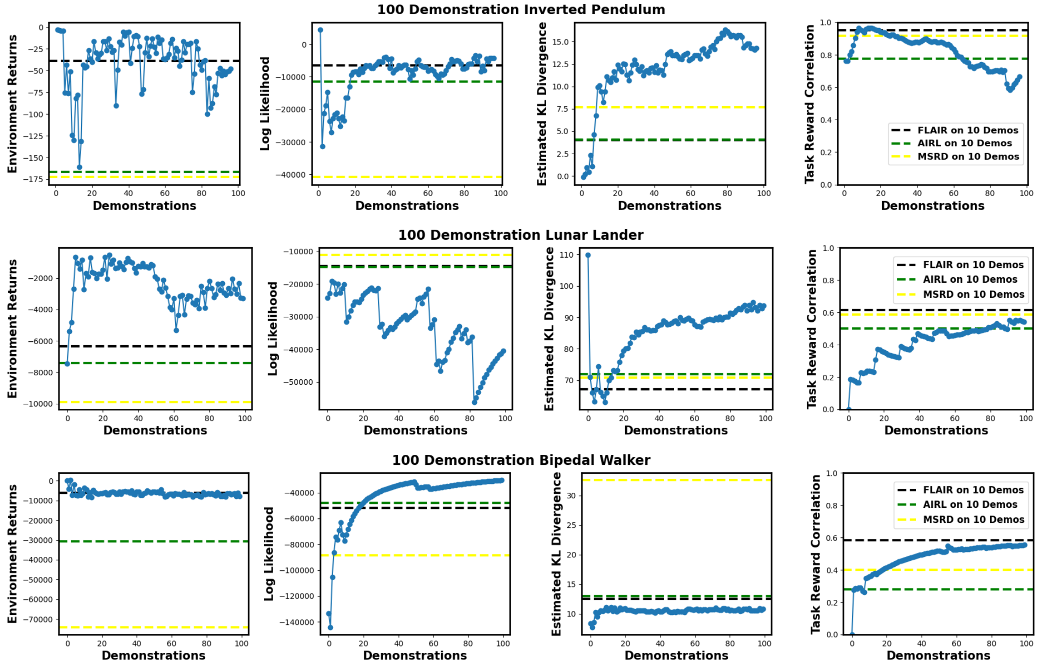


Figure 8: This figure shows the key metrics for FLAIR during the lifelong learning experiment in all three domains. The blue lines show the performance of FLAIR in the scalability experiment for each demonstration. We see FLAIR demonstrates consistently strong performance in environment returns as it 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).

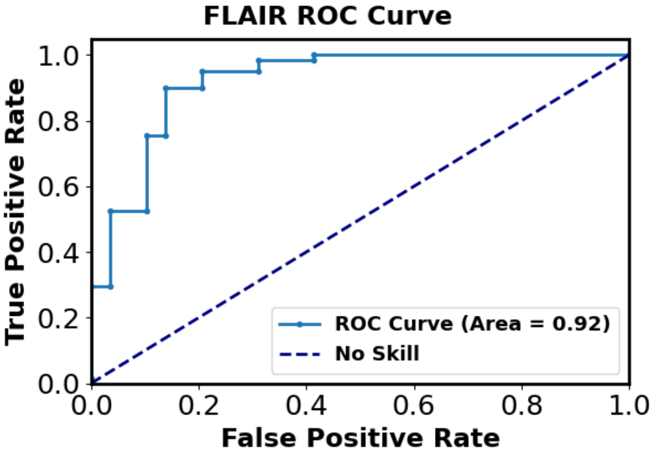


Figure 9: This figure depicts the ROC curve with FLAIR's ability to predict the mixture- vs. new-strategy as the threshold is varied. We see FLAIR with a mixture optimization threshold has a high Area Under Curve (0.92) in the ROC Curve for IP.

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.