

# Better Than Diverse Demonstrators: Reward Decomposition from Suboptimal and Heterogeneous Demonstrations

Chunyue Xue<sup>1</sup>, Letian Chen<sup>1</sup>, and Matthew Gombolay<sup>1</sup>

**Abstract**—Inverse Reinforcement Learning (IRL) typically involves inferring a reward function from expert demonstrations to enable agents to imitate the demonstrated behavior. However, real-world settings often provide suboptimal and heterogeneous demonstrations, where human demonstrators use diverse strategies and imperfect actions. Yet, we are unaware of any prior work that simultaneously addresses the challenges of IRL, of which demonstrations are both heterogeneous and suboptimal. In this work, we propose a novel approach, REPRESENT (Reward dEcomposition fRom hETerogeneous Suboptimal dEMoNstration), that disentangles the latent intrinsic task reward and the strategy-specific reward from suboptimal and diverse strategies. Our method learns to identify a shared task reward component that generalizes across varying demonstrator preferences while also modeling distinct strategy-specific rewards. By decomposing the common task reward across varied demonstrations, REPRESENT extracts the core objectives shared by all strategies, enabling the agent to perform better than the demonstrators while preserving individual strategy preferences. We validate our approach on three robotic domains, showing a higher correlation with the true task reward and improved policy performance compared to baselines. These results suggest that REPRESENT can effectively handle suboptimality and heterogeneity, providing a solution for real-world LfD applications to better learn from demonstrations varied in quality and strategy.

**Index Terms**—Reinforcement Learning; Learning from Demonstration

## I. INTRODUCTION

AS robots increasingly enter complex environments, traditional robotics methods requiring expert programming for each task become impractical. Learning from Demonstration (LfD), which enables robots to acquire skills directly from human demonstrations, addresses this scalability issue and has been successfully applied to manufacturing [1], [2], healthcare [3], [4], and autonomous driving [5], [6]. However, demonstrations often come from non-experts with varying skills and preferences [7], [8], making them both *suboptimal* and *heterogeneous*:

- 1) **Suboptimality**: Typical users provide lower-quality demonstrations compared to experts due to limitations in experience or ability [?], [9], hindering policy learning in traditional LfD [7].
- 2) **Heterogeneity**: Demonstrators exhibit varying preferences or habits in performing the same task [10], causing

Manuscript received: December, 6, 2024; Revised March, 13, 2025; Accepted May, 11, 2025.

This paper was recommended for publication by Editor Aleksandra Faust upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by NIH 1R01HL157457 and NSF IIS-2340177

<sup>1</sup>School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA 30332, {cxue43, letian.chen, matthew.gombolay}@gatech.edu

Digital Object Identifier (DOI): see top of this page.

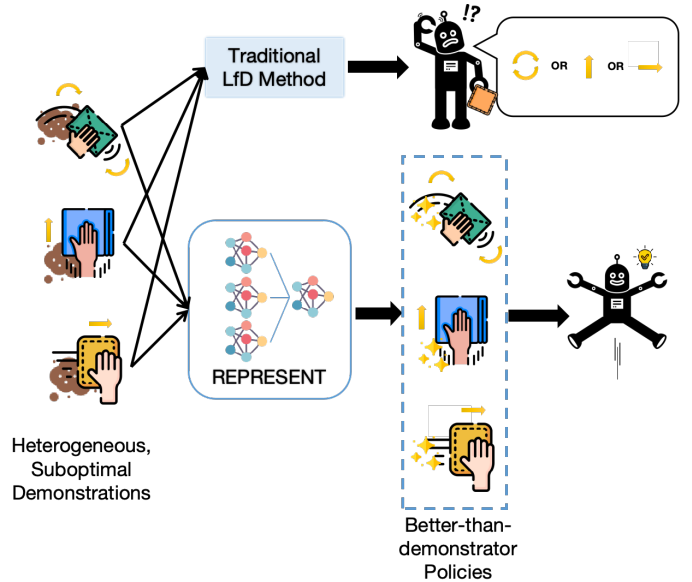


Fig. 1: Traditional LfD methods struggle when demonstrations are heterogeneous and suboptimal, while REPRESENT can learn better policies from these demonstrations.

ambiguity and inconsistency in naive IRL-based policy learning [11].

Collecting higher-quality or more demonstrations is often impractical due to limited expert availability and high costs. For example, household robots usually learn from caregivers or family members who provide imperfect demonstrations with different habits. Thus, demonstrations are inherently suboptimal and heterogeneous.

An effective algorithm addressing both challenges simultaneously is necessary for practical real-world LfD applications. We depict the problem setup with dual need with Fig. 2. While previous work has separately addressed these issues, our experiments indicate they struggle when suboptimality and heterogeneity coexist.

We propose **REPRESENT (Reward dEcomposition fRom hETerogeneous Suboptimal dEMoNstration)**, a novel method to tackle both challenges simultaneously. We consider a scenario with limited, imperfect demonstrations from multiple non-expert users with diverse preferences. Unlike existing methods, REPRESENT extracts a shared latent task reward from suboptimal demonstrations while preserving strategy-specific rewards. This design allows our framework to surpass demonstrator performance by effectively leveraging diverse, imperfect demonstrations. Our contributions include:

- Introducing **REPRESENT**, a novel framework that dis-

entangles shared task rewards from strategy-specific elements, effectively handling suboptimal demonstrations while preserving unique strategies.

- Demonstrating performance improvements (up to 300%) over prior work across three robotic domains.

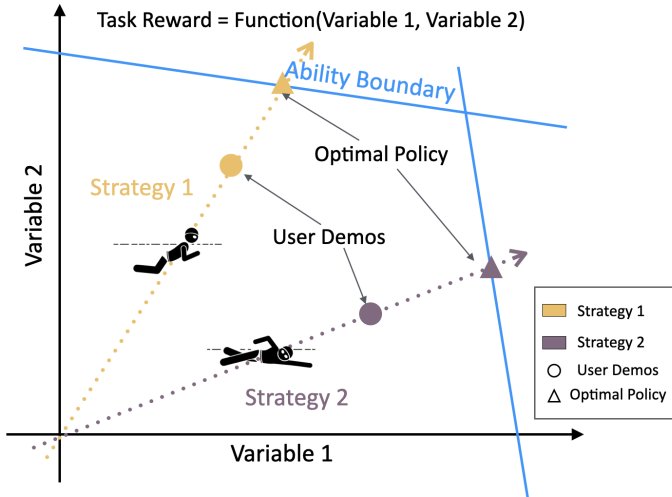


Fig. 2: Human demonstrations (circles) reflect different suboptimal strategies for performing a task, each constrained by individual limitations (ability boundary). The underlying optimization space is *unknown*, and REPRESENT aims to disentangle these distinct strategies and extract a shared, task reward to achieve more-optimal policies (triangles).

## II. RELATED WORK

**LfD and Inverse Reinforcement Learning (IRL)** – LfD [7], [12], [13] has been a widely studied reinforcement learning area in recent years, which allows robots to learn new skills from human teaching without prior knowledge of programming. Behavioral Cloning (BC)[14] learns policies through supervised learning by directly mapping states to actions using expert demonstration data, which can suffer from distribution shift and limited generalization. BC-RNN is a variant of BC with a Recurrent Neural Network (RNN) policy network, which captures temporal correlations in decision-making and becomes useful in multi-modal learning. Within LfD, IRL methods aim to infer the agent’s reward function based on observations of its behavior, given a set of observed trajectories. Adversarial Inverse Reinforcement Learning (AIRL) [15] and Maximum-entropy IRL (MaxEnt-IRL) [16] are two popular probabilistic IRL algorithms. These methods work well when the given demonstrations are heterogeneous and (near-)optimal, but generally struggle to produce significantly better policies when heterogeneous and suboptimal demonstrations are given. In contrast, our method addresses these limitations and enables effective learning even when the demonstrations are suboptimal and have diverse strategies.

**Learning from Suboptimal Demonstrations** – Recently, methods have been introduced to leverage noisy and imperfect data. Some approaches require a mixed dataset of suboptimal and expert demonstrations, where suboptimal data serves as

auxiliary input [17], [18], [19]. However, these methods still depend on the availability of costly expert data. Other methods, such as [20], [21], [22], can work solely with suboptimal data. Among these methods, Self-Supervised Reward Regression (SSRR) [23], and D-REX [24] have shown promise in learning from suboptimal demonstrations. D-REX utilizes BC along with a ranking approach to infer desired reward functions from suboptimal demonstrations. However, D-REX suffers from an inductive bias issue from the ranking formulation and can be brittle to covariate shift, leading to inaccurate reward estimates. In contrast, SSRR leverages a low-pass filter based on the performance-noise relationship. Although SSRR can significantly improve robustness and accuracy from noisy inputs, it cannot model multi-strategy demonstrations.

**Multi-Task and Multi-Strategy Reward Learning** – Learning from heterogeneous demonstrations is another key challenge in LfD, often addressed in multi-task learning. Approaches like Distral [25] and CARE [26] capture shared behaviors across tasks by either distilling common policies or creating task-specific representations. However, these methods focus on handling multiple tasks rather than learning diverse strategies for the same task. In contrast, methods like Multi-Style Reward Distillation (MSRD) [27] aim to disentangle shared task rewards and strategy-specific rewards but require a complex distillation-regularization structure. Fast Lifelong Adaptive IRL (FLAIR) [28] builds on MSRD to enable rapid adaptation by maintaining a library of strategy prototypes. Nevertheless, these approaches are limited by the performance of demonstrations, and the corresponding learned task and strategy rewards will diverge substantially from the true latent reward function when input demonstrations are not optimal.

In contrast to previous approaches that focused only on suboptimality or heterogeneity, our approach, REPRESENT, attempts to take a proactive step in constructing a robust and efficient method that can simultaneously handle both challenges. REPRESENT disentangles the shared task reward and strategy-specific rewards, allowing it to improve learning efficiency by leveraging heterogeneous demonstrations while denoising for suboptimal actions.

## III. PRELIMINARIES

### A. Markov Decision Process

The reinforcement learning problem can be modeled as a Markov Decision Process (MDP), denoted as a 6-tuple  $(S, A, R, T, \gamma, \rho_0)$ , where  $S$  and  $A$  are the state space and action space.  $R(s, a)$  is the reward given that action  $a$  is taken in state  $s$  and sometimes can be simplified as  $R(s)$ .  $T(s, a, s')$  represents the transition probability from state  $s$  to state  $s'$  after taking action  $a$ .  $\rho_0$  is the initial state distribution. A policy,  $\pi(a|s)$ , is the probability of an agent taking action  $a$  in state  $s$ . The aim of RL is to maximize cumulative discounted reward  $\mathbb{E}_\pi [\sum_{t=0}^T \gamma^t R(s_t, a_t)]$ . IRL can also be modeled as an MDP with an unknown reward function. As a type of LfD, the inputs of IRL should be demonstrations from humans or robots, denoted by  $\tau = \{(s_t, a_t)\}$ .

### B. Reward Function Decomposition

We propose a decomposition structure based on MSRD [27] for reward functions. A strategy-combined reward,  $R^i$ , learned from Strategy- $i$  demonstrations is a linear combination of a general task reward,  $R^0$ , and a specified strategy-only reward,  $\tilde{R}^i$ , as given by Eq. 1, where  $\alpha_i$  is a hyperparameter representing the relative weight of the strategy-only reward  $\tilde{R}^i$ .

$$R^i = R^0 + \alpha_i \tilde{R}^i \quad (1)$$

### C. Self-Supervised Reward Regression

SSRR [23] is an IRL method to learn from suboptimal demonstrations we partially adopt and improve upon this work.

*Phase I* – SSRR generates a set of noisy trajectories from the initial policy learned using suboptimal demonstrations. On each set of the suboptimal demonstrations with Strategy  $i$ , we train an IRL model, Noisy Adversarial Inverse Reinforcement Learning (Noisy-AIRL), to obtain an initial reward function  $R^i$  and an initial policy  $\pi^i$  via a GAN(Generative Adversarial Network)-like structure, serving as the starting point for generating the self-supervised data. Different levels of uniform random noise are injected into the initial policy  $\pi^i$  to generate noisy policies  $\{\pi^i\}$  and create diverse performance level behaviors. For each noisy policy, a corresponding set of trajectories  $\{\tau^i\} = \{\eta^i, \{s_t^i\}, \{a_t^i\}, \{r_t^i\}\}$  is generated, where each trajectory  $\tau^i$  consists of the noise parameter  $\eta^i$ , a sequence of states and actions  $\{s_t^i\}, \{a_t^i\}$ , and the corresponding initial reward  $r_t^i = R^i(s_t^i, a_t^i)$ . This process results in a self-supervised dataset that spans different levels of policy performance, providing a rich set of training examples for learning the noise-performance relationship.

*Phase II* – SSRR then focuses on characterizing the relationship between the injected noise and the performance of the corresponding trajectories. We fit a sigmoid function to capture the relationship between the noise level and the performance of each noisy policy. The fitting uses the cumulative reward estimates from the initial reward function  $R^i$ . The sigmoid function is defined as:  $\sigma(\eta^i) = \frac{c}{1 + e^{-k(\eta^i - x_0)}} + y_0$ , where  $\eta^i$  is the noise parameter, and  $c, k, x_0, y_0$  are the sigmoid parameters. This curve serves as a low-pass filter, yielding a smooth, noise-performance curve, which is shown in Fig. 3 in the “Noise-reward curve fitting” block. By using the noise-performance curve, SSRR smooths the estimated rewards from the initial policy and corrects for the biases induced by noisy trajectories. This regularized reward estimate can help to capture the overall reward more accurately, even under varying levels of suboptimality.

To denoise and learn from the suboptimal demonstrations, we adopt the above two rephases before our novel decomposition reward regression architecture.

## IV. METHOD

We present **REPRESENT (Reward dEcomposition fRom hEterogeneous Suboptimal dEmoNstraTion)**, a novel IRL framework (Fig. 3) that learns reward functions by disentangling shared task rewards from strategy-specific components in

suboptimal, heterogeneous demonstrations. REPRESENT introduces (1) a reward network that separates task and strategy-specific rewards, and (2) a custom loss function that isolates task from strategy rewards. Our approach effectively captures complex demonstration data, enabling policies to outperform diverse, non-expert demonstrators. In the following sections, we provide details on the problem setup, reward network design, the custom loss function, and how our framework learns from diverse data.

### A. Problem Setup

In our settings, assuming the given demonstrations are distinguished by their providers, and each provider is associated with one specific strategy, we denote the demonstrations from User  $i$  with Strategy  $i$  as  $\mathcal{D}^i = \{\tau_1^i, \tau_2^i, \dots, \tau_N^i\}$ , and the combined full dataset with  $M$  varying strategies is  $\mathcal{D} = \{\mathcal{D}^1, \mathcal{D}^2, \dots, \mathcal{D}^M\}$ , where  $i \in \{1, 2, \dots, M\}$  is the strategy index,  $N$  is the number of trajectories for one strategy, and  $M$  is the number of different demonstrators/strategies.

Our objective is to learn strategy-combined reward function  $R^i$  for each strategy given this set of demonstration trajectories and distill a disentangled task reward  $R^0$  that can extract the common points of distinct strategy-combined reward functions and mimic the environment reward, together with each strategy-only reward functions  $\tilde{R}^i$ . The policies acquired using these task reward functions or strategy-combined functions can achieve a performance exceeding that of the demonstrator.

### B. Reward Regression with Distillation

In our method, we first have two phases to handle the suboptimality for each set of demonstrations with the same strategy by estimating noise-performance curves. Phase 1 generates self-supervised data through noisy policies, starting with an initial reward learned from suboptimal demonstrations for each strategy. In Phase 2, we characterize the relationship between noise and performance by fitting a noise-performance curve for each strategy, using a sigmoid function to smooth the reward estimates.

The key innovation in our approach is Phase 3: a multi-component reward structure that separately models task and strategy rewards. This structure is necessary to learn from both suboptimal and heterogeneous demonstrations, as it enables us to capture the shared task objectives and the distinct behaviors generated by diverse strategies, consolidating common task reward information into the shared  $R^0$ , while preserving strategy-specific nuances. Jointly employing all the noise-performance curves from the first two phrases, we regress the combined reward functions of each strategy parameterized by trajectory states and actions, each having two distinct components: Shared Task Reward Network and Strategy-Specific Reward Networks.

For **Shared Task Reward Network**, we introduce a shared reward network  $R^0$  that captures the intrinsic task reward common to all demonstrations. This network is designed to learn the core task-related rewards that are invariant to individual strategies. And for **Strategy-Specific Reward Networks**, in parallel, we use separate strategy reward networks  $\tilde{R}^i$  for

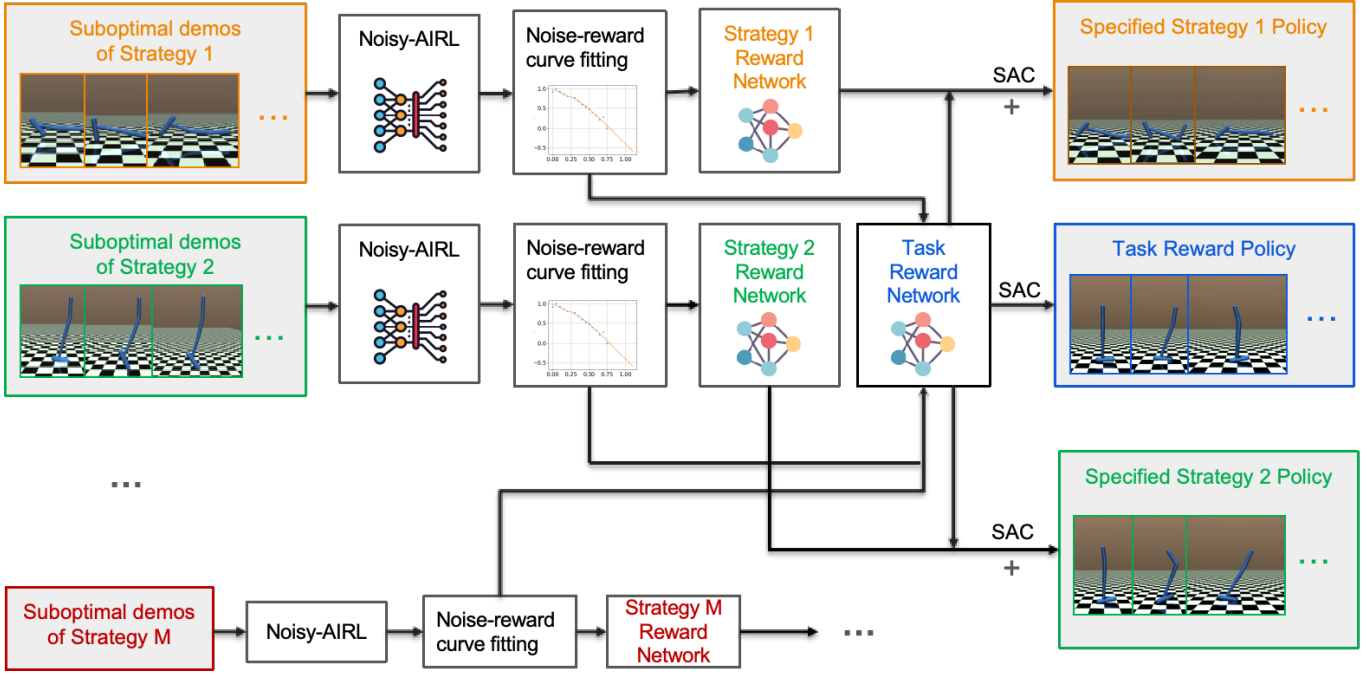


Fig. 3: Using the Hopper domain as an example, for  $M$  different strategies, we have  $M$  substructures to get the noise-reward curve, which are jointly trained to learn the task rewards and strategy-specified rewards. The learned rewards are then used for task and strategy policy training.

each distinct strategy  $i$ . These networks model the individual preferences or heuristics exhibited by different demonstrators. Unlike prior work that assumes a single reward structure, our method allows for multiple strategy rewards, which capture variations in human performance.

### C. Loss Function

To ensure the disentanglement of task and strategy rewards, we design a custom loss function. The overall loss function is given in Equation 2, where  $\lambda_{SSRR}$ ,  $\lambda_{reg}$ ,  $\lambda_{BCD}$  are weighting parameters that balance the different components,  $\theta^i = \{\theta^0, \tilde{\theta}^i\}$  are the full parameters of strategy-combined reward neural networks for each strategy  $i$ ,  $\theta^0$  are the parameters of task reward network, and  $\tilde{\theta}^i$  are parameters of the reward-specified reward network.

$$\mathcal{L} = \sum_{i=1}^M (\lambda_{SSRR} \cdot \mathcal{L}_{SSRR}(\theta^i) + \lambda_{reg} \cdot \mathcal{L}_{reg}(\theta^i) + \lambda_{BCD} \cdot \mathcal{L}_{BCD}(\theta^i)) \quad (2)$$

$$\mathcal{L}_{SSRR}(\theta^i) = E_{\tau^i} [((\sum_{t=0}^T (R_{\theta^0}(s_t^i, a_t^i) + R_{\tilde{\theta}^i}(s_t^i, a_t^i)) - \sigma(\eta^i)^2)] \quad (3)$$

$$\mathcal{L}_{reg}(\theta^i) = E_{\tau^i} \| R_{\tilde{\theta}^i}(s^i, a^i) \|_2 \quad (4)$$

$$\mathcal{L}_{BCD}(\theta^i) = \sum_{m=1}^M (e^{\sum_{t=0}^T R_{\theta^i}(s_t^i, a_t^i)} - e^{\sum_{t=0}^T R_{\theta^i}(s_t^m, a_t^m)})^2 \quad (5)$$

The objective includes 1) a performance loss adapted from SSRR  $\mathcal{L}_{SSRR}$ , which aligns the complete strategy-combined reward network with the regularized reward estimates obtained in Phase 2; 2) an L2-regularization loss  $\mathcal{L}_{reg}$  on strategy-only reward's output to make  $R_{\tilde{\theta}^i}$  close to 0 and  $R^i$  close to  $R^0$ , i.e., prioritizing optimizing the intrinsic task reward, according to

Eq. 1; 3) a Between-Class Discrimination (BCD) loss from FLAIR [28] to increase the strategy reward's discriminability between different strategies. The BCD loss encourages Strategy  $i$ 's reward to give a lower reward to other strategy demonstration  $\tau_m$ , which encourages the strategy rewards to learn nonsimilar, archetypal strategies that can cover a diverse range of behaviors and more easily discriminate and assign strategy labels among those behaviors. We note that there exists a trade-off between L2-regularization loss and BCD loss, which we tune empirically.

Once the task and strategy rewards are learned, we apply a standard reinforcement learning algorithm (e.g., Soft Actor-Critic (SAC) [29]) to obtain a policy  $\pi^*$  that maximizes the strategy-combined reward  $R^i$  and the task reward  $R^0$ .

### D. Algorithm

REPRESENT is summarized in Algorithm 1. Starting with a heterogeneous and suboptimal demonstration dataset  $\mathcal{D}$ , we initialize the task network  $\theta^0$  and strategy-specific networks  $\tilde{\theta}^i$ . Phase 1 (lines 3-6) applies Noisy-AIRL to learn the distinct and suboptimal demonstrations. For each strategy's dataset  $\mathcal{D}_i$ , we infer the initial latent reward function  $r^i$ . Phase 2 (lines 7-11) involves sampling noisy trajectories at various noise levels to infer a noise-performance curve.

Phase 3 (lines 12-16) performs reward regression and decomposition jointly by updating  $\theta^0$  and  $\tilde{\theta}^i$  iteratively using the curve and the loss in Eq. 2. This step ensures  $\theta^0$  generalizes across strategies while each  $\tilde{\theta}^i$  captures strategy-specific details. Upon convergence, the final output is the task network  $\theta^0$  and each strategy-specific network  $\tilde{\theta}^i$ , representing the distilled reward functions across strategies.

**Algorithm 1** REPRESENT

- 
- 1: **Input:** Heterogeneous and suboptimal demonstration dataset  $\mathcal{D} = \{\mathcal{D}^1, \mathcal{D}^2, \dots, \mathcal{D}^M\}$ , where  $\mathcal{D}^i = \{\tau_1^i, \tau_2^i, \dots, \tau_N^i\}$ ,  $i \in \{1, 2, \dots, M\}$
  - 2: **Initialize:**  $\theta^0, \tilde{\theta}^i$  for each strategy  $i, \forall i \in \{1, 2, \dots, M\}$
  - 3: **Phase 1: Apply Noisy-AIRL to handle heterogeneous and suboptimal demonstrations**
  - 4: **for** each strategy demonstration set  $\mathcal{D}_i \in \mathcal{D}$  **do**
  - 5:     Infer latent reward function  $R$  using Noisy-AIRL
  - 6: **end for**
  - 7: **Phase 2: Sample noisy trajectories and construct noise-performance curve**
  - 8: **for** each noise level  $i$  **do**
  - 9:     Sample noisy trajectories  $\{\pi^i\}$  with noise level  $\eta_j$
  - 10:     Generate noise-performance curve based on  $\{\pi^i\}$
  - 11: **end for**
  - 12: **Phase 3: Joint reward regression and decomposition using strategy-specified and task networks**
  - 13: **while** not converged **do**
  - 14:     Compute reward regression using  $\tilde{\theta}^i$  and  $\theta^0$
  - 15:     Update parameters  $\tilde{\theta}^i$  and  $\theta^0$  with Eq. 2
  - 16: **end while**
  - 17: **Output:**  $\theta^0, \tilde{\theta}^i$  for each strategy  $i$
- 

## V. EMPIRIMENTAL SETUP

In this section, we describe the experiment settings, including benchmarks and baselines we choose and the demonstration generation process.

## A. Benchmark and Baseline

We test our approach on four robotic tasks based on the OpenAI Gym [30], Gymnasium-Robotics [31], and MuJoCo [32]: Hopper-v3, HalfCheetah-v3, DoubleInvertedPendulum-v2, and Franka Kitchen. To fit our setting, we remove termination judgments of environments to gain flexibility in behaviors such as the crawling actions of Hopper. In our experiments, we only utilize demonstrations with two different strategies. We collect demonstrations for Franka Kitchen using human teleoperation, and generate synthetic demonstrations for the other three tasks.

As for the baselines, we choose SSRR, MSRD, D-REX, BC, and BC-RNN, and analyze them with different sets of input demonstrations.

## B. Synthetic Demonstration Generation

To generate suboptimal, heterogeneous demonstrations, we leverage Diversity is All You Need (DIAYN) [33] with an early stopping mechanism, and generate demonstrations with two different strategies. DIAYN trains a discriminator to differentiate policies based on latent variables,  $z$ , and utilizes the pseudo-reward defined in Eq. 6, where  $q_\phi(z|s)$  is the posterior probability of  $z$  given state  $s$ , and  $p(z)$  is the prior.

$$r_z(s, a) = \log q_\phi(z|s) - \log p(z), \quad (6)$$

This model encourages learning distinct behaviors but disregards task objectives. Thus, we integrate a linear combination

of task rewards like Eq. 1 and diversity rewards to balance task performance and behavior diversity.

We employ early stopping while training DIAYN with environment rewards before policies display the perfect performances to create suboptimality. This method follows a standard practice [23], producing a set of suboptimal policies. This method generates diverse user-like demonstrations at non-expert levels.

Moreover, to create a test set for reward correlation analysis, we utilize different trained DIAYN agents with varying training steps to produce new test datasets, which provide trajectories with performance ranging from low to high. Additional information about demonstrations we use for experiments is shown in Table II together with the experiment results.

## C. Human Demonstration Collection

We modify the original Gymnasium-Robotics environment FrankaKitchen-v1 as one of our experiment domains and collect the human demonstrations from experimenters by keyboard teleoperation. The goal of the domain is to use the Franka robot arm to move the kettle to the top left burner. We use two strategies to push the kettle - directly pushing the body part or holding the handle. The reward per step is the negative norm between the kettle position and its goal position.

## VI. RESULT

In this section, we present the advances of our method in learning from heterogeneous and suboptimal demonstrations compared to the previous works focusing on only one side.

## A. Research Questions

As for results, we want to test two main hypotheses: **H1**: the reward learned by our method has a higher correlation with the ground-truth reward than baselines; **H2**: using the reward to train an RL policy, this learned policy can achieve better than demonstrator’s performance and outperform the learned policy generating from baseline method reward functions. Besides the main hypotheses, additional studies on data efficiency and loss function ablation are conducted.

TABLE I: **Learned Reward Correlation Coefficients with Ground-Truth Reward Comparison:** Reported results are mean±standard deviation with five different seeds.

		Hopper	HalfCheetah	DoubleInverted Pendulum
Task Reward Function	Ours	<b>0.949</b> ±0.002	<b>0.929</b> ±0.003	<b>0.999</b> ±0.000
	SSRR	0.578±0.057	0.821±0.058	-0.269±0.789
	MSRD	0.551±0.071	0.709±0.069	-0.303±0.750
	D-REX	0.649±0.058	0.779±0.067	-0.393±0.782
Strategy 1 Reward Function	Ours	<b>0.938</b> ±0.003	<b>0.930</b> ±0.003	<b>0.999</b> ±0.000
	SSRR	0.933±0.017	0.913±0.038	0.434±0.825
	MSRD	0.653±0.045	0.627±0.018	-0.361±0.668
Strategy 2 Reward Function	Ours	<b>0.957</b> ±0.004	<b>0.928</b> ±0.004	<b>0.999</b> ±0.000
	SSRR	0.916±0.031	0.912±0.024	0.946±0.074
	MSRD	0.704±0.142	0.656±0.107	-0.269±0.703

## B. Metrics

We select two evaluation metrics with different data sets. First, we use the learned reward correlation coefficients with ground truth reward on training and test sets. Note that for SSRR and D-REX, we do not have the task reward as a component, therefore we directly use the overall reward function for correlation evaluation. Second, we measure the ground-truth rewards of RL policies trained with the learned reward functions. High correlation for strategy reward functions indicates that each function not only captures the feature of demonstrators but is also aligned with the underlying task reward and does not drift from the task goal.

## C. Correlation Result

Table I reports the correlation between learned and ground-truth rewards across three domains. Our method consistently achieves higher correlations than baseline approaches. Because our method disentangles latent and strategy-specific rewards from suboptimal, diverse data, it produces three distinct reward functions. For ours and MSRD which both model a task reward explicitly, we report task reward correlation along with the two combined-strategy reward correlations. For SSRR and D-REX, which lack explicit task reward modeling, we train on the combined strategy demonstrations and treat the result as the task reward prediction to ensure a fair comparison across methods handling multi-strategy data.

As indicated by the results in Table I, together with Fig. 4, the correlation coefficients of our method with ground-truth rewards are consistently higher across all evaluation settings compared to the two baseline methods, which supports our first hypothesis **H1**. The results highlight that even when confronted with imperfect and heterogeneous demonstrations, REPRESENT achieves stronger alignment with ground-truth rewards compared to existing methods such as SSRR, MSRD and D-REX, demonstrating the ability to accurately infer the reward function. The results support the hypothesis that decomposed rewards lead to more accurate learning outcomes, thus validating its effectiveness in complex, multi-strategy, and suboptimal environments.

Notably, combining data from multiple strategies yields lower correlation in SSRR than treating each strategy separately. This suggests that increasing dataset size with diverse strategies can introduce ambiguity that disrupts suboptimality-based methods, ultimately degrading reward modeling.

## D. Policy Performance Result

Table II presents the highest ground-truth reward achieved by the final policy, trained using different reward functions in three benchmark domains averaged over five trials. The results show the advancement of our method in learning higher-performance policies.

The RL algorithm we utilize to train policies with learned reward functions for REPRESENT, SSRR, and D-REX is Soft Actor-Critic (SAC) [29]. For each domain, we report the results for two distinct strategies and the learned policy result for our method’s task reward. The goal is to evaluate whether the learned reward functions can guide policies to

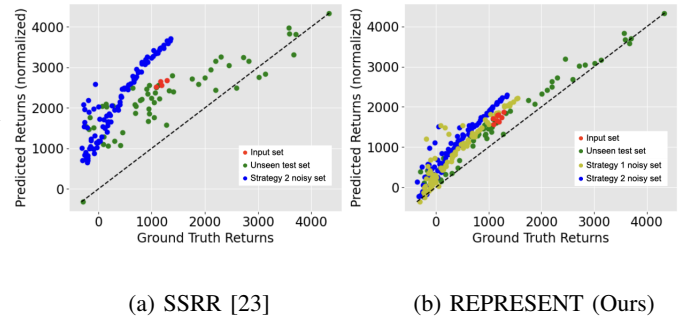


Fig. 4: SSRR vs. REPRESENT’s rewards correlating with ground truth. Red, green, yellow, and blue dots represent input demonstrations, unseen test data, noisy Strategy 1, and noisy Strategy 2 demonstrations, respectively.

achieve higher ground-truth performance, thereby validating the effectiveness of the reward inference approach. To display the enhancement more intuitively, we include the percentage of improvement of the final policy rewards relative to the rewards of the initial demonstrations. For the policy with task reward, we choose the average reward of all strategy demonstrations as the denominator. Since none of SSRR, MSRD, or D-REX can be directly trained for a task-reward policy, we only compare our method with the input demonstrations for the results of the policy learned from task reward.

Our approach consistently performs better than the baseline methods and can attain higher performance than the demonstrator for all testing domains. In the Hopper-v3 environment, REPRESENT achieves 10.8% better task reward than SSRR and 14.5% better than MSRD for Strategy 1, 85.1% better than SSRR and 100.5% better than MSRD for Strategy 2. For the policy with overall task reward, our method reaches 1692.14, exceeding the baselines including D-REX. The results are similar for the HalfCheetah-v3 domain. Our approach still shows superior performance, while SSRR cannot even gain positive rewards for one strategy.

In the DoubleInvertedPendulum-v2 domain, which is relatively simple and easy to learn, one of the baseline, SSRR, can get a slightly lower but similar optimal-level performance compared to our method. However, REPRESENT displays advantages in terms of efficiency during learning. As shown in Fig. 5, our method is able to converge to the same or higher reward level with fewer training steps for each strategy, showing a faster learning process compared to SSRR. For Strategy 1, our method is more stable during the training process. For Strategy 2, both our method and SSRR exhibit significant instability around 19,000 and 26,000 iterations, leading to the temporary drop in performance.

Additionally, we train BC and BC-RNN policies directly using combined datasets of two strategies’ demonstrations. As shown in Table II, BC only achieves performance similar to the inputs. In contrast, BC-RNN, despite its ability to handle multimodality, results in lower performance. This is likely due to the increased model complexity and additional parameters of RNNs when demonstration data is diverse and limited. Therefore, for suboptimal and heterogeneous data, BC and its

TABLE II: The average cumulative rewards and percentage improvement over input demonstrations for baselines.

Policy Learned from Strategy 1 Reward Function				
	Demo	Ours	SSRR	MSRD
Hopper-v3	1197.93	1316.01 (110%)	1187.50 (99%)	1149.11 (96%)
HalfCheetah-v3	1243.13	<b>3930.14 (316%)</b>	-256.09 (-21%)	1201.08 (97%)
DoubleInvertedPendulum-v2	861.84	<b>9350.52 (1085%)</b>	9290.24 (1078%)	29.67 (3%)

Policy Learned from Strategy 2 Reward Function				
	Demo	Ours	SSRR	MSRD
Hopper-v3	1587.98	<b>2893.47 (182%)</b>	1563.45 (98%)	1443.17 (91%)
HalfCheetah-v3	1172.71	3860.52 (329%)	2013.79 (172%)	1410.34 (120%)
DoubleInvertedPendulum-v2	1217.65	9350.47 (768%)	9315.36 (765%)	29.82(2%)

Policy Learned from Task Reward Function		Policy Learned with Merged Data			
	Demo	Ours	D-REX	BC	BC-RNN
Hopper-v3	1392.96	1692.14 (121%)	0.91 (0%)	1102.19 (79%)	187.20 (13%)
HalfCheetah-v3	1207.92	3540.06(293%)	3148.64 (261%)	960.12 (79%)	799.44 (66%)
DoubleInvertedPendulum-v2	1039.75	9350.50 (899%)	83.00 (7%)	722.88 (70%)	129.96 (12%)

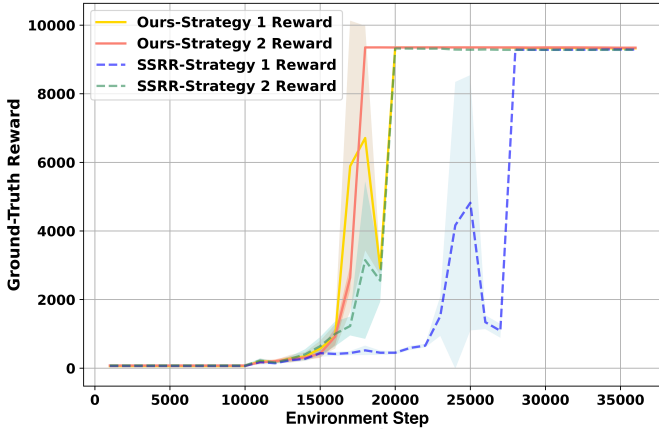


Fig. 5: Learned Policy Performance vs. Training Steps in DoubleInvertedPendulum-v2. Lines show the average cumulative ground-truth rewards across five trials; the shaded region depicts the standard deviation.

variant BC-RNN will be limited by the input performance and difficult to have an outperformed policy.

Our results demonstrate REPRESENT outperforms baseline methods in final policy performance using learned reward functions, supporting hypothesis **H2**. Its multi-component architecture more effectively captures reward structures and integrates information across diverse strategies – critical when demonstrators exhibit varied, non-expert behavior. Additionally, REPRESENT reaches comparable or superior performance earlier in training, indicating enhanced efficiency.

### E. Ablation Study

We conduct an ablation study to assess the contribution of the SSRR, regularization, and BCD losses in Eq. 2 in Hopper-v3 (Table III). Our full model achieves the highest task performance and produces distinct behavioral strategies. Removing the regularization term improves task reward optimization but sacrifices diversity. Removing the BCD loss degrades optimality and diversity.

TABLE III: Ablation Study in Hopper-v3

Function	Strategy 1	Strategy 2	Task
Ours	<b>1316.01</b>	<b>2893.47</b>	1692.14
- Regularization Loss	1090.03	2269.25	<b>1785.19</b>
- BCD Loss	1214.00	2000.38	1633.13
SSRR	1187.5	1563.45	N/A

### F. Human Demonstration Experiment: Franka Kitchen

Utilizing the demonstrations from Section V-C, we benchmark REPRESENT and BC with the results presented in Table IV. REPRESENT framework outperforms BC and the original demonstrations, although some strategies still show room for improvement due to the challenges posed by larger observation and action spaces.

TABLE IV: Results for FrankaKitchen. Higher is better.

	Demo	REPRESENT		BC
		Strategy	Task	
Strat. 1	-122.86	<b>-121.94</b>	<b>-106.48</b>	-467.95
Strat. 2	-174.46	<b>-169.20</b>	<b>-106.48</b>	-528.68

## VII. LIMITATIONS AND FUTURE WORK

While REPRESENT shows promising results, two key limitations offer opportunities for future work:

- *Real-World Deployment*– Evaluations were conducted only in simulations. Future efforts should deploy REPRESENT on physical robots to evaluate its robustness against real-world challenges like noise, sensor errors, and hardware constraints.
- *Scalability*– Our current approach assumes a finite number of distinct strategies. As a first step, we extend our analysis of two-strategy models from Section VI to now consider how REPRESENT could leverage three distinct strategies. Results are shown in Table V, indicating promising results for learning strategy and task rewards in HalfCheetah-v3. Future research should further investigate how to scale to larger diversity settings. Furthermore, future work should also investigate how mechanisms to

relax the assumption of known strategy labels, e.g. by inferring those labels [28].

TABLE V: Average cumulative rewards of policies learning from three different strategies in HalfCheetah-v3.

	Demo.	REPRESENT
Strat. 1	1243.13	<b>2279.29</b>
Strat. 2	1172.71	<b>3220.52</b>
Strat. 3	1181.35	<b>2664.76</b>
Task	1199.06	<b>2901.24</b>

## VIII. CONCLUSION

We propose a novel framework that infers reward functions from suboptimal, heterogeneous demonstrations by separating shared task rewards from strategy-specific elements. By intelligently integrating methods for learning from suboptimal (i.e., SSRR) and heterogeneous (i.e., MSRD) demonstrations, our approach efficiently learns from imperfect demonstrations, outperforming existing methods in virtual environments, achieving better-than-demonstrator performance.

## ACKNOWLEDGMENT

This work was supported by the National Institutes of Health (NIH) under Grant 1R01HL157457 and National Science Foundation (NSF) under Grant IIS-2340177. We thank Megan Langwasser for her help in editing this manuscript.

## REFERENCES

- [1] W. K. H. Ko, Y. Wu, K. P. Tee, and J. Buchli, "Towards industrial robot learning from demonstration," in *Proceedings of the 3rd international conference on human-agent interaction*, 2015, pp. 235–238.
- [2] D. A. Duque, F. A. Prieto, and J. G. Hoyos, "Trajectory generation for robotic assembly operations using learning by demonstration," *Robotics and Computer-Integrated Manufacturing*, vol. 57, pp. 292–302, 2019.
- [3] C. Lauretti, F. Cordella, E. Guglielmelli, and L. Zollo, "Learning by demonstration for planning activities of daily living in rehabilitation and assistive robotics," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1375–1382, 2017.
- [4] L. Wang, W. Yu, X. He, W. Cheng, M. R. Ren, W. Wang, B. Zong, H. Chen, and H. Zha, "Adversarial cooperative imitation learning for dynamic treatment regimes," in *Proceedings of The Web Conference 2020*, ser. WWW '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1785–1795.
- [5] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2641–2646.
- [6] O. Scheel, L. Bergamini, M. Wolczyk, B. Osiński, and P. Ondruska, "Urban driver: Learning to drive from real-world demonstrations using policy gradients," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 718–728.
- [7] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.
- [8] 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)*. IEEE, 2019, pp. 1048–1055.
- [9] J. Drugowitsch, V. Wyart, A.-D. Devauchelle, and E. Koechlin, "Computational precision of mental inference as critical source of human choice suboptimality," *Neuron*, vol. 92, no. 6, pp. 1398–1411, 2016.
- [10] R. Paleja, A. Silva, L. Chen, and M. Gombolay, "Interpretable and personalized apprenticeship scheduling: Learning interpretable scheduling policies from heterogeneous user demonstrations," *Advances in neural information processing systems*, vol. 33, pp. 6417–6428, 2020.
- [11] S. Nikolaidis, R. Ramakrishnan, K. Gu, and J. Shah, "Efficient model learning from joint-action demonstrations for human-robot collaborative tasks," in *Proceedings of the tenth annual ACM/IEEE international conference on human-robot interaction*, 2015, pp. 189–196.
- [12] S. Schaal, "Learning from demonstration," *Advances in neural information processing systems*, vol. 9, 1996.
- [13] 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*, vol. 3, no. 1, pp. 297–330, 2020.
- [14] M. Bain and C. Sammut, "A framework for behavioural cloning," in *Machine intelligence 15*, 1999, pp. 103–129.
- [15] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," *arXiv preprint arXiv:1710.11248*, 2017.
- [16] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, et al., "Maximum entropy inverse reinforcement learning," in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.
- [17] S. Choi, K. Lee, and S. Oh, "Robust learning from demonstrations with mixed qualities using leveraged gaussian processes," *IEEE Transactions on Robotics*, vol. 35, no. 3, pp. 564–576, 2019.
- [18] M. Yang, S. Levine, and O. Nachum, "Trail: Near-optimal imitation learning with suboptimal data," 2021.
- [19] L. Yu, T. Yu, J. Song, W. Neiswanger, and S. Ermon, "Offline imitation learning with suboptimal demonstrations via relaxed distribution matching," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 9, 2023, pp. 11 016–11 024.
- [20] Y. Gao, H. Xu, J. Lin, F. Yu, S. Levine, and T. Darrell, "Reinforcement learning from imperfect demonstrations," 2019.
- [21] D. Brown, W. Goo, P. Nagarajan, and S. Niekum, "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations," in *International conference on machine learning*. PMLR, 2019, pp. 783–792.
- [22] Z. Zhu, K. Lin, B. Dai, and J. Zhou, "Self-adaptive imitation learning: Learning tasks with delayed rewards from sub-optimal demonstrations," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, pp. 9269–9277, Jun. 2022.
- [23] L. Chen, R. Paleja, and M. Gombolay, "Learning from suboptimal demonstration via self-supervised reward regression," in *Conference on robot learning*. PMLR, 2021, pp. 1262–1277.
- [24] D. S. Brown, W. Goo, and S. Niekum, "Better-than-demonstrator imitation learning via automatically-ranked demonstrations," in *Conference on robot learning*. PMLR, 2020, pp. 330–359.
- [25] Y. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu, "Distral: Robust multitask reinforcement learning," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, 2017.
- [26] S. Sodhani, A. Zhang, and J. Pineau, "Multi-task reinforcement learning with context-based representations," in *International Conference on Machine Learning*. PMLR, 2021, pp. 9767–9779.
- [27] L. Chen, R. Paleja, M. Ghuy, and M. Gombolay, "Joint goal and strategy inference across heterogeneous demonstrators via reward network distillation," in *Proceedings of the 2020 ACM/IEEE international conference on human-robot interaction*, 2020, pp. 659–668.
- [28] L. Chen, S. Jayanthi, R. R. Paleja, D. Martin, V. Zakharov, and M. Gombolay, "Fast lifelong adaptive inverse reinforcement learning from demonstrations," in *Conference on Robot Learning*. PMLR, 2023, pp. 2083–2094.
- [29] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. Pmlr, 2018, pp. 1861–1870.
- [30] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.
- [31] R. de Laza, K. Andreas, J. J. Tai, S. R. Lee, and J. Terry, "Gymnasium robotics," 2024. [Online]. Available: <http://github.com/Farama-Foundation/Gymnasium-Robotics>
- [32] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.
- [33] B. Eysenbach, J. Ibarz, A. Gupta, and S. Levine, "Diversity is all you need: Learning skills without a reward function," in *7th International Conference on Learning Representations, ICLR 2019*, 2019.