Learning Representations that Enable Generalization in Assistive Tasks

Zhiyang He
Anca Dragan

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-233
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-233.html

October 25, 2022

Copyright © 2022, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Learning Representations that Enable Generalization in Assistive Tasks

by Zhiyang He

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Anca D. Dragan
Research Advisor

(Date)

* * * * * * *

Professor Zackory Erickson
Second Reader

10/24/2022

(Date)

Learning Representations that Enable Generalization in Assistive Tasks

Jerry Zhi-Yang He[1], Aditi Raghunathan[2], Daniel S. Brown[3],
Zackory Erickson[2], and Anca D. Dragan[1]

[1]University of California Berkeley
[2]Carnegie Mellon University
[3]University of Utah
{hzyjerry, anca}@berkeley.edu, daniel.s.brown@utah.edu, {raditi, zerickso}@cmu.edu

Abstract: Recent work in sim2real has successfully enabled robots to act in physical environments by training in simulation with a diverse "population" of environments (i.e. domain randomization). In this work, we focus on enabling generalization in *assistive tasks*: tasks in which the robot is acting to assist a user (e.g. helping someone with motor impairments with bathing or with scratching an itch). Such tasks are particularly interesting relative to prior sim2real successes because the environment now contains a *human who is also acting*. This complicates the problem because the diversity of human users (instead of merely physical environment parameters) is more difficult to capture in a population, thus increasing the likelihood of encountering out-of-distribution (OOD) human policies at test time. We advocate that generalization to such OOD policies benefits from (1) learning a good latent representation for human policies that test-time humans can accurately be mapped to, and (2) making that representation adaptable with test-time interaction data, instead of relying on it to perfectly capture the space of human policies based on the simulated population only. We study how to best learn such a representation by evaluating on purposefully constructed OOD test policies. We find that sim2real methods that encode environment (or population) parameters and work well in tasks that robots do in isolation, do not work well in *assistance*. In assistance, it seems crucial to train the representation based on the *history of interaction* directly, because that is what the robot will have access to at test time. Further, training these representations to then *predict human actions* not only gives them better structure, but also enables them to be fine-tuned at test-time, when the robot observes the partner act.

Keywords: assistive robots, representation learning, OOD generalization

1 Introduction

Our ultimate goal is to enable robots to assist people with day to day tasks. In the context of patients with motor impairments, this might mean assistance with scratching an itch, bathing, or dressing [1, 2, 3]. These are tasks in which doing reinforcement learning from scratch in the real world is not feasible, and so sim2real transfer is an appealing avenue of research. Sim2real methods for physical robot tasks in isolation typically work by constructing a diverse "population" of environments and training policies that can work with any member of the population (e.g. a range of parameters of a physics simulator or a range of lighting and textures) [4, 5, 6, 7, 8, 9, 10, 11].

Similarly, population-based (self-play) training has proven successful in zero-sum games against humans [12, 13, 14]. But unlike tasks the robot does in isolation, assistance requires coordinating with a human who is also acting. And unlike competitive settings, assuming the human to be optimal when they are not, can result in dramatically poor performance [15]. Thus, in sim2real for assistance, we have to design a population of potential users and strategies to train with, akin to the physical environment parameters in typical sim2real tasks, rather than the standard population-based training approaches used in competitive settings. But designing a population that is diverse and useful enough to enable generalization to test-time humans, each with their own preferences, strategies, and capabilities, remains very challenging, making it likely that test-time partners might lie outside of the distribution the population was drawn from. Therefore, sim2real methods for assistance will need to be ready to *generalize* to *out-of-distribution* partner policies.

In this work, we identify two principles as key to enabling better generalization. First is that we benefit from learning a latent space of partners that distills their policies down to a structure that is useful for the robot's policy *and* that makes it easy to identify partners at test time. Second is that we need to be prepared for this space to not perfectly capture the space of real human policies, and design it so that it is *adaptable* with real test-time interaction data.

We thus propose a framework that *learns a latent space directly from history of interaction by predicting the partner's actions*. Our framework allows a robot to capture the relevant information about the human partner that the robot can actually identify when starting to interact, and also enables test-time adaptation of the latent space itself when observing the partner's actions. When evaluated with partner policies we purposefully design to be out-of-distribution, we find that our approach leads to better generalization than prior methods which either do not learn a latent space at all [16], do not learn a latent space directly based on interaction history [7], or train a latent space based on other observables, like states or rewards [17, 18].

Our contributions are four-fold:

1. We introduce an assistive problem setting where the focus is explicitly on generalization to out-of-distribution partner policies.
2. We introduce a framework for training policies for this problem setting, Prediction-based Assistive Latent eMbedding (PALM). This enables us to study different methods for learning latent representations on how well they enable generalization.
3. We identify that the design choice of training a latent space by *predicting partner actions directly from history* outperforms (1) state-of-the-art sim2real approaches used in non-assistive tasks that are based on embedding environment parameters [7]; as well as (2) human-robot interaction approaches that train representations by predicting observed states or rewards [17, 18].
4. We propose to adapt the learned space at test time, upon observing the partner's actions, and show it leads to generalization performance gains.

2 The Assistive Personalization Problem

In this section, we introduce the personalization problem in an assistive context. In particular, our goal is to learn a robot policy π_R that can assist a novel human partner in zero-shot fashion, or with a small amount of test-time data.

Two-player Dec-POMDP. An assistive task can be modeled as a two-agent, finite horizon decentralized partially-observable Markov decision process (Dec-POMDP) and is defined by a tuple $\langle S, \alpha, A_R, A_H, \mathcal{T}, \Omega_R, \Omega_H, O, R \rangle$. Here S is the state space and A_R, A_H are the human's and the robot's action spaces, respectively. The human and the robot share a real-valued reward function $R : S \times A_R \times A_H \to \mathbb{R}$; however, we assume that the reward function is not necessarily observed by the robot, i.e. its parameters (e.g. the location of the human has an itch) are in the hidden part of the state. $\mathcal{T} : S \times A_R \times A_H \times S \to [0,1]$ is the transition function, which outputs the probability of the next state given the current state and all agents' actions. Ω_R and Ω_H are the sets of observations for the robot and human, respectively, and $O : S \times A_R \times A_H \to \Omega_R \times \Omega_H$ represents the observation probabilities. We denote the horizon of the MDP by T.

Target User. We target users with partial motor functions — a common impairment for individuals with partial arm functions. This is an impairment that can occur in some people with cervical SCI, ALS, MS, and some neurodegenerative diseases — leading to the need for robotic assistance. We model the extent of the impairment as the privileged information in Dec-POMDP. The robot does not know this a-priori and thus needs to adapt to individual users' capabilities.

The Robotic Care-giving Setup. We define the observation space for the robot and the human following [3]: the robot observes its own joint angles, and the human's joint positions in the world coordinate and contact forces; the human observes their joint angles (proprioception) and the end-effector position of the robot. When training with simulated humans, the robot gets a reward signal (which depends on privileged information), and has to use that signal to learn to implicitly identify enough about the human to be useful; at test time, the robot does not observe reward signal and must use what it has learned at training time to identify the human's privileged information and be helpful.

Distributions of Humans. Let function $\pi_H : \Omega_H^* \times A_H \to [0,1]$ be the human policy that maps from local histories of observations $\mathbf{o}_t^H = (o_1^H, \dots, o_t^H)$ over Ω_H to actions. We define two distributions of human policies $\pi_H \in \mathscr{D}_{\text{train}}, \mathscr{D}_{\text{test}}$. In the assistive itch scratching, $\mathscr{D}_{\text{train}}$ can be a set of humans with different itch positions on their arms, which lead to their different movements. We refer to them as *in-distribution humans*. $\mathscr{D}_{\text{test}}$ contains *out-of-distribution* humans whose itch position differ from those in the $\mathscr{D}_{\text{train}}$. At training time, the robot has access to $\mathscr{D}_{\text{train}}$. Thus, it has ground-truth knowledge about the training human's privileged information, such as each human's itch position. At test time, we evaluate the robot policy by sampling humans $\pi_H \sim \mathscr{D}_{\text{test}}$ and directly pairing them with the robot policy. We evaluate the zero-shot and few-shot adaptation performance of the robot policy.

Objective. The main problem we study is how to leverage the training distribution to learn a robot policy $\pi_R : \Omega_R^* \to A_R$ such that we achieve the best performance on test humans. Concretely, we define the performance of the robot and human as

$$J(\pi_R, \pi_H) = \mathbb{E}\left[\sum_{t=0}^{T} R(s_t, \pi_R(\mathbf{o}_t^R), \pi_H(\mathbf{o}_t^H))\right], \tag{1}$$

Thus, only given access to $\mathscr{D}_{\text{train}}$, our objective is to find the robot policy π_R such that

$$\pi_R = \arg\max_{\pi} J(\pi, \pi_H), \pi_H \sim \mathscr{D}_{\text{test}}. \tag{2}$$

3 Learning Personalized Embeddings for Assistance with PALM

In this section, we present Prediction-based Assistive Latent eMbedding (PALM). We introduce the general framework of using a latent space to perform personalization in an assistive context. Finally, we describe how we can optimize PALM at test time to handle out-of-distribution humans.

3.1 Learning an Assistive Latent Space

Given a training distribution of humans $\mathscr{D}_{\text{train}}$, we would like to learn a robot policy that can successfully adapt to assist new users. To achieve that, a robot must learn to solve the task, while efficiently inferring the hidden component that differs across humans. One natural way to do so is to use a *learned latent space* that succinctly captures what differs across humans in a way that affects the robot's policy. When deployed on a test human, the robot must learn to infer this latent embedding. How do we learn such a latent space? We assume access to a population of training humans sampled from $\mathscr{D}_{\text{train}}$. We describe how we generate this distribution in Sec. 4.1.

Learning by prediction. We outline our method in Fig. 1. Given a history, $\tau_{1:t} = \big((o_1^R, a_1^H), \dots (o_t^R, a_t^H)\big)$, of robot observation and human action[1] pairs up to time t, we embed this trajectory to a low-dimensional manifold. The goal is to extract information about the human policy π_H. Achieving this means that the vector z is representative of the trajectory so far and

[1] We do not assume access to the person's raw action (e.g. joint torques), but track the change in the person's Cartesian pose.

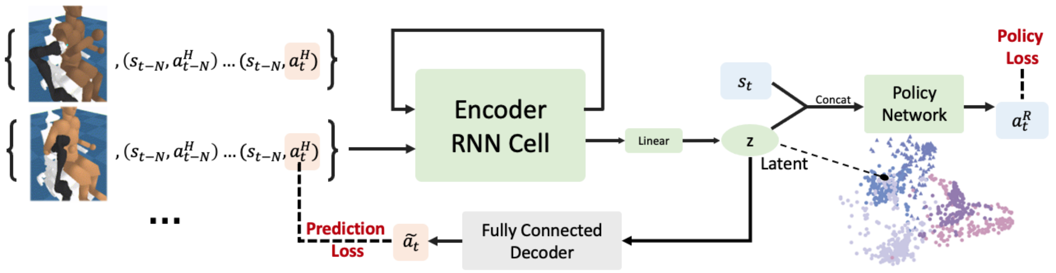

Figure 1: The framework for jointly learning the Personalized Latent Embedding Space and the robot policy. During training time, we train all components end-to-end to optimize the prediction loss and the policy loss. At test time, we execute the robot policy, which is highlighted in green.

indicative of the person's future actions. In other words, we can use z to predict the person's action at $t+1$. We do this by training a decoder $\mathcal{D}_\phi$ parameterized by ϕ to predict the next action from the encoder's output $z \sim \mathcal{E}_\theta(z;\tau_{1:t})$.

$$\mathcal{L}_{\text{pred}} = \min_{\theta,\phi} ||\mathcal{D}_\phi(z) - a^H_{t+1}||^2 + c_{\text{KL}} \cdot KL(\mathcal{E}_\theta(z;\tau_{1:t})||\mathcal{N}(z)) \tag{3}$$

The encoder $\mathcal{E}$ is a recurrent neural network as an encoder parameterized by θ. Here the second term is a regularization term motivated by Variational Autoencoder [19, 20], that enforces the latent space to follow a normal distribution. This encourages nearby terms in the latent space to follow similar semantic meanings. In the context of assistive tasks, this helps us better cluster similar humans closer in the latent space, and we show a didactic example in Sec. 4.3.

Other forms of Prediction-based Human Embeddings Besides predicting actions, LILI [17] and RILI [18] are two prior methods that fit within the PALM framework and that learn to predict the next observations and rewards. These methods have been shown to work when the human is operating under a discrete number of dynamics and can be used for influencing human behaviours. Note, however, that both LILI and RILI assume access to the reward function at test time, and in the assistive setting, we do not have access to humans reward function — we don't know a-priori the preference and needs of a new user.

End-to-end training of latent space We apply standard policy optimization methods to learn π_R. Note that because base policy π_R and the latent space z are interdependent on each other — z is the input to π_R, and π_R decides the data distribution which leads to z, we need to train them jointly. We follow prior work [17, 18] and jointly learning the latent space and the robot policy by using the PPO [21] algorithm. In addition to the prediction loss $\mathcal{L}_{\text{pred}}$, we can optimize for the policy loss $\mathcal{L}_{\text{pol}}$ simultaneously. Because training with a population of humans with hidden information can be unstable, we also leverage Behaviour Cloning to stabilize our policy training. To achieve this, we use expert policies obtained via co-optimization with the human population, which we describe in the appendix. We then query the expert for actions on on-policy data, and optimize the standard behavioral cloning loss to minimize deviation from it: $\mathcal{L}_{\text{BC}} = \sum_t ||a^{\text{exp}}_t - a^R_t||^2$. The ensures that we encounter no distribution shift at test time. The overall policy optimization loss include three terms: latent prediction loss, PPO loss and the Behaviour Cloning loss:

$$\mathcal{L}_{\text{PALM}} = \mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{PPO}} + \mathcal{L}_{\text{BC}} \tag{4}$$

3.2 Latent Space Adaptation at Test Time

At test time, as we work with a new user, we would like our encoding of the new user to match the true latent information, z^* of that user. In other words, we would like to minimize $||\mathcal{E}(\tau) - z^*||^2$. Because we do not know about the new users a-priori, we can only optimize for this objective via proxy, which we refer to as *test time adaptation*.

Because the PALM latent space is based on action prediction, we can adapt it to a new user by further optimizing the latent space. Note that because Eq. (3) requires only observation-action data, we do not need any additional label to perform test time adaptation. More formally, we collect a small dataset of test-time interaction trajectories, τ, and perform a few gradient steps to optimize both the encoder and decoder for Eq. (3): $(\theta,\phi) \to (\theta,\phi) - \delta\nabla_{(\theta,\phi)}\mathcal{L}_{\text{pred}}(\mathcal{E}_\theta, \mathcal{D}_\phi, \tau)$.

The idea of test-time optimization has been shown to improve perceptual robustness for grasping in sim2real research [22]. We follow a similar pipeline, where we can improve the latent encoding by collecting unsupervised action data from test users. Here the main difference is instead of perceptual differences, our goal is to reduce the domain gap on test users.

4 Experiments

In this section, we evaluate our method PALM (Prediction-based Assistive Latent eMbedding) in collaborative human-robot environments of varying tasks and varying populations of human models. In particular, we focus on the out-of-distribution generalization by constructing different forms of out-of-distribution populations. We focus on empirically investigating the benefits of learning a latent space, the effect of different kinds of prediction on learning a useful latent space, the properties of learned latent spaces, and the gains from test-time adaptation to humans.

4.1 Environments

Here we introduce two environments where we study assistive personalization. In both environments, the robot has to infer some hidden information from the human in order to successfully solve the task. Note that these are two example meant for demonstrating the effectiveness of the algorithm, and we do not claim to solve the full robotic caregiving problem.

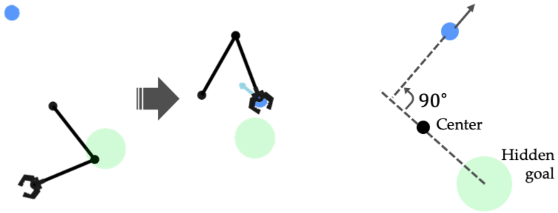

Figure 2: The assistive reacher environment. Left: the robot's goal is to move the human agent towards the hidden target. Right: the hidden goal's position can be inferred 90 degrees from the humans force output.

Assistive Reacher (Fig. 2) is 2D collaborative environment where a two-link robot arm assisting a human agent, represented as a point particle, to get to the target position. This target is located at $(d\cos\alpha_{\mathrm{H}}, d\sin\alpha_{\mathrm{H}})$, where d is a fixed value, and $\alpha_{\mathrm{H}} \in [-\pi,\pi]$ is known to the human, but not the robot. The human agent is initialized randomly in the 2D plane with random hidden parameters α_{H}, k_H, where $\alpha_{\mathrm{H}} \in [-\pi,\pi], k_{\mathrm{H}} \in [0.5, 1.5]$. The robot can only identify the target position by physically interacting with the human — once the robot initiates contact, the human applies a force $k_{\mathrm{H}} \cdot (\cos\alpha_{\mathrm{H}} + \frac{\pi}{2}, \sin\alpha_{\mathrm{H}} + \frac{\pi}{2})$. Only by recognizing the human in terms of $\alpha_{\mathrm{H}}, k_{\mathrm{H}}$ can the robot compensate the force, and successfully move the human to the hidden target.

The Scope of Generalization. We define $\mathscr{D}_{\text{train}}$ as 36 samples uniformly sampled from $\alpha_{\mathrm{H}} \in [-\pi, \frac{\pi}{2}], k_{\mathrm{H}} \in [0.5, 1.5]$ and $\mathscr{D}_{\text{test}}$ as 12 samples uniformly sampled from $\alpha_{\mathrm{H}} \in [\frac{\pi}{2}, \pi], k_{\mathrm{H}} \in [0.5, 1.5]$. Each episode has 40 timesteps.

Assistive Itch Scratching (Fig. 1) is adapted from assistive gym [3]. It consists of a human and a wheelchair-mounted 7-dof Jaco robot arm. An itch spot is randomly generated on the human's right arm. The human has limited mobility — they can only move the 10 joints on the right arm and upper chest, and needs to collaborate with the robot to scratch the itch spot. The robot does not directly observe the itch spot, and has to rely on interaction with the human to infer the itch spot. Each episode has 100 timesteps.

We use co-optimization to create $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{test}}$ for Assistive Itch Scratching. A benefit of the co-optimization framework is that it naturally induces reward-seeking behaviour from the human and the robot, which simulates assistance scenarios. For instance, to generate more inactive human policies, we can introduce a weighting term in the reward function for human action penalties

$R_{\mathrm{p}} = c_{\mathrm{p}} \cdot ||\pi_{\mathrm{H}}(s_t)||^2$ where c_{p} is a constant controlling the penalty. The overall objective becomes

$$\max_{\pi_{\mathrm{H}}, \pi_{\mathrm{R}}} \mathbb{E}\Big[\sum_t R\big(s_t, \pi_{\mathrm{H}}(s_t), \pi_{\mathrm{R}}(s_t)\big)\Big] + c_{\mathrm{p}} \cdot ||\pi_{\mathrm{H}}(s_t)||^2 \tag{5}$$

The Scope of Generalization. We are motivated by real world applications where users tend to have different levels of mobility limitations, or itch locations in different body parts. To generate a synthetic population to capture such diversity in itch scratching task, we explore different co-optimization settings (1) we assign different human action penalty to be $c_{\mathrm{p}} = 3, 3.5, 4$, where larger penalties lead to the human agent exerting less effort. (2) We simulate different itch positions on the human's arm and train co-optimized human and robot policies conditioned on them. This leads to qualitatively different strategies for the human and the robot. Note that this serves first step to understanding how different methods generalize, since we never expect to be able to capture the diversity in humans perfectly. For training, we use Proximal Policy Optimization (PPO) to optimize human and robot policies in an interleaving fashion. Note that we also keep the co-optimized robot policy and use it to obtain expert actions for assistive policy training (see Sec. 3.1 and supplement for full details).

To construct $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{test}}$, we divide the two arms' areas into four equal portions, as shown in Fig. 3, and generate human policies conditioned on itch positions in these areas. $\mathscr{D}_{\text{train}}$ consists of three of the four portions and $\mathscr{D}_{\text{test}}$ consists of the remaining one. We then construct three distribution sets in increasing order of difficulties. In the first distribution $\mathscr{D}^1$, we confine itch positions from a line-shaped region. In $\mathscr{D}^2$, we sample from all the arm areas. Note that $\mathscr{D}^1$ and $\mathscr{D}^2$ are constructed by setting action penalty $c_{\mathrm{p}} = 3$. In $\mathscr{D}^3$, we combine humans of $c_{\mathrm{p}} = 3, 3.5, 4$, each trained with two different random seeds. This adds the extra complexity of human activity levels. We simulate 12 in-distribution humans from each of the three training portions under each action penalty.

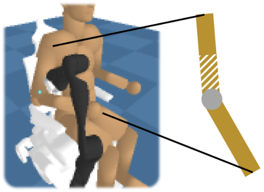

Figure 3: Definition of $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{test}}$.

4.2 Baselines

We compare with baselines that do not learn an explicit latent space as well as existing methods for adaptation via learning latent embeddings.

MLP and RNN. We follow [16] that trains sequential models to enable adaptation to simulated humans. We explore using a recurrent neural network or a feed-forward network on concatenated state-action histories. The models directly output robot action, and there is no latent space modeling.

ID-based Human Embeddings. In contrast to learning latent space from history, another class of method studies encodes human-designed environment parameters [7, 23, 24]. We focus specifically on RMA [7], a two-phased method that first learns to encode task-ID (phase I) and then trains a recurrent network to regress to the embeddings from observation history (phase II). For training a quadruped robot, RMA encodes the physical parameters (friction, payload, etc) of the environment. The first stage trains a policy with ground-truth information, and the second phase performs environment identification. While RMA is shown to be effective for learning policy for in-distribution environments, it is unclear how well it generalizes to out-of-distribution environments. Furthermore, in assistive tasks, it is unclear how to construct the "ground-truth ID" for phase I that quantifies the user characteristics. We study **RMA-Vanilla** and **RMA-Onehot**, where we assign each training human a one-hot vector. For RMA-Vanilla, we use a three-dimensional vector that includes the x, y position of the itch position and the arm index. When

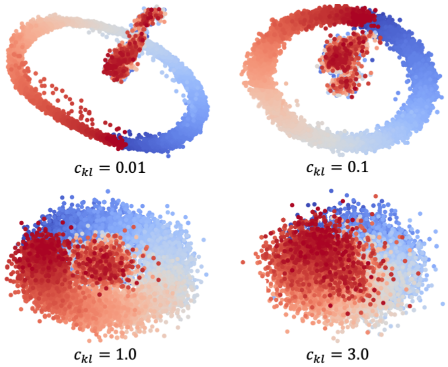

Figure 4: Latent space of PALM in the assistive reacher environment when we can sample humans $\alpha_{\mathrm{H}} \in [-\pi \text{ (red)}, \pi \text{ (blue)}]$ continuously.

there are multiple human activity levels as mentioned in Sec. 4.1, we introduce a fourth dimension with an integer to indicate the action penalty c_{p}.

LILI and RILI. We consider two other methods of the PALM framework: LILI [17] and RILI [18]. As mentioned in Sec. 3.1, LILI jointly predicts future observation and reward, and RILI predicts reward. Given that reward is only available at training time, we cannot perform test time optimization for LILI and RILI.

Ablations of PALM. Our method has several components: we use a recurrent neural network to encode interaction history, and use its output to minimize prediction loss $\mathcal{L}_{\mathrm{pred}}$ and policy loss $\mathcal{L}_{\mathrm{pol}}$. We also use the KL term in Eq. (3) to regularize the human embeddings. To test the effectiveness of our method, we separate different parts and create a set of baselines. We hereby describe them in detail: (1) No $\mathcal{L}_{\mathrm{pred}}$: the model shares the same encoder and policy network architecture, yet we don't optimize for $\mathcal{L}_{\mathrm{pred}}$. By removing the prediction loss in Fig. 1, the latent space is not explicitly trained to contain human information. (2) No c_{KL}: no regularization in the latent space. (3) Frozen embedding: instead of jointly training embeddings and the policy network, we first train the encoder on expert data, freeze it, and then train robot policy. We include an ablation study of our main experiments in the appendix.

4.3 Didactic Experiment in Assistive Reacher Environment

Can PALM learn a meaningful distribution from the interaction? Unlike other ID-based methods like RMA, PALM does not have access to the human parameters at training time. We study whether PALM can learn a meaningful latent space without explicitly knowing this information. We sample training humans from $\alpha_{\mathrm{H}} \in [-\pi, \pi]$ continuously. We train PALM with different amount of prior regularization, c_{KL} from Eq. (3). We train using a recurrent window of length 4 and a batch size of 512 episodes. Additional training details can be found in the supplementary material.

We average test results using 100 episodes and visualize the results in Fig. 4.3. Given that humans are parameterized by α_{H}, K_H, the ideal embedding space looks like a ring with a small blob in the center. The ring corresponds to the 2D projection of α_{H} and the blob denotes the initial part of interaction before contact, which is indistinguishable. We find that while PALM never observes the underlying parameter α_{H}, it can learn a latent space that characterizes α_{H}. Interestingly, varying the amount of regularization qualitatively affects the shape of the latent space. Setting the VAE regularization $c_{\mathrm{KL}} = 0.1$ recovers a latent space that most resembles to the ideal latent space.

4.4 Assistive Reacher Main Experiment

Experiment Setting. We use the finite $\mathscr{D}_{\mathrm{train}}$ described in Sec. 4.1 and train all baselines using Eq. (4) for 200 epochs with 512 batch size. We then evaluate the trained policies on $\mathscr{D}_{\mathrm{test}}$. We normalize the resulting reward with respect to oracle reward.

Results. We average test results using 100 episodes. On in-distribution humans, we find that all methods successfully follow the right policy that assists the human to reach their goal. This shows that they all successfully predict the human latent information explicitly or implicitly. On out-of-distribution humans, the methods are no longer guaranteed to predict the correct embedding. PALM with action prediction significantly outperforms other methods. With test-time adaptation, PALM further improves.

Visualizing the latent space. We qualitatively study generalization by visualizing the latent space as well as the mapped embeddings of both in-distribution and out-of-distribution humans (in red crosses) in Fig. 5. Interestingly, only PALM with action prediction can infer the "ring" structure. RMA, RNN, LILI and RILI fail to do so. We hypothesize that because hand-crafted human IDs do not convey the information about human policy, RMA warped the IDs in arbitrary what that are harmful for generalization. The same happens with RILI and LILI. We hypothesize this is due to the inherent ambiguity in reward prediction: a low reward does not necessarily recover the human policy structure.

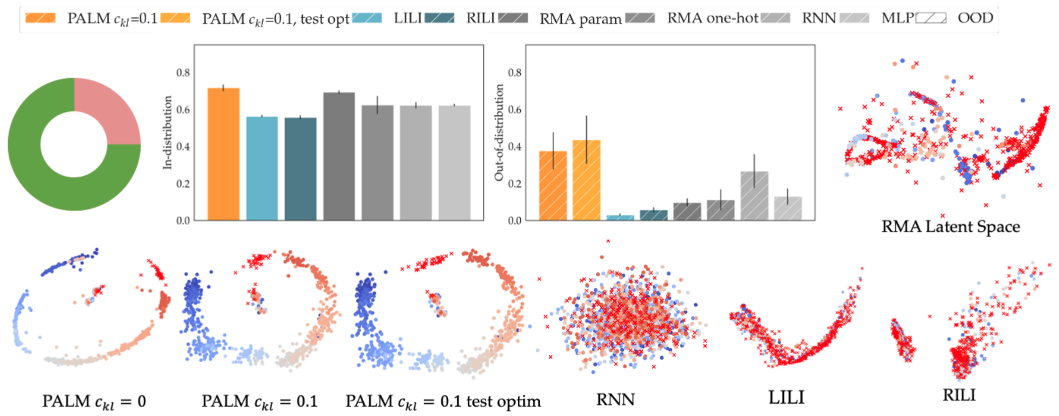

Figure 5: Top left: evaluation of PALM and baselines on in-distribution (green) and out-of-distribution (pink) humans. Right and bottom: visualization of the latent embeddings of different methods. OOD humans are highlighted in red crosses. Best viewed electronically.

The visualization also offers some insight into why having c_{KL} regularization is helpful for generalization. Compare the latent space of PALM $c_{\text{KL}} = 0$ and PALM $c_{\text{KL}} = 0.1$, the latter induces a smoother distribution where test humans are better fitting in the "missing arc" of the "ring". Further more, we see that with test time optimization, the PALM latent space embeds the OOD human better, by filling in more of the arc.

4.5 Assistive Itch Scratch Main Experiment

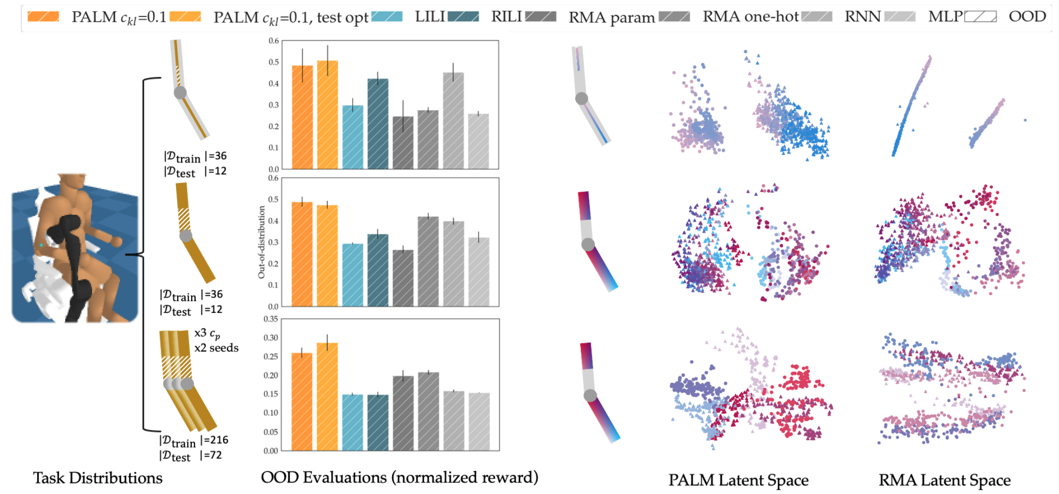

Figure 6: In assistive itch scratching, we sample humans by different itch positions and activity levels (varying action penalty c_{p}). We visualize the in- and out-of-distribution humans on the two-link arm figures. We also visualize the embedding space of PALM and RMA, where we color-code the embeddings of in-distribution humans. Here we leave the embeddings of other baselines to supplementary material.

Results. We follow a similar procedure as the reach environment to train itch scratching policy, and train for 240 epochs with 192 trajectories for batch size. As shown in Fig. 6, we observe PALM with action prediction has better generalization performance than other baselines. We see that in the simplified distribution $\mathscr{D}^1$, RILI and MLP have the best generalization performance among baselines, yet as the complexity of the training human distribution increases, they deteriorate. Detailed results of the ablation study are included in the appendix.

Visualizing the Latent Space To further investigate why PALM generalize better to OOD human than RMA baselines, We visualize the latent space of the "straight-line" distribution. As we see in Fig. 7, PALM can capture the structure in human training distribution as two clusters, and also correctly embed the OOD humans distribution as a part of the upper arm distribution. RMA-based methods, on the other hand, can discover the structure of training humans. Yet qualitatively, they fail the correctly embed the OOD humans in proximity to the upper arm distribution.

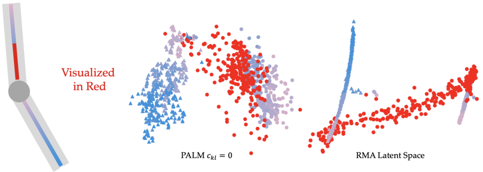

Figure 7: To better under extrapolations to OOD human, which have new itch locations highlighted in red on the left, we visualize the embeddings of both the IND (colored) and OOD (red) humans on the right.

4.6 Real World User Study

We conduct a user study with 9 users from age 22 to 29 to evaluate our methods on real users. [2] In the study, we inform the user that they have a marker indicating their itch location, which the robot does not know. We then advise the user to form their strategies in interacting with the robot. Real users are almost always different from synthetic humans, and to work well with real users, the methods need to generalize beyond their training distributions. For each user, we test with two itch spots, one on the upper arm and one on the lower arm. We use randomized orderings of different methods. Each interaction is 150 simulation time steps and roughly six seconds of wall clock time. Within each method, we let the user interact for a total of five trials. We also find that the robot can be initialized in poses that collide with the human at that moment. We discard trials when such collisions are detected.

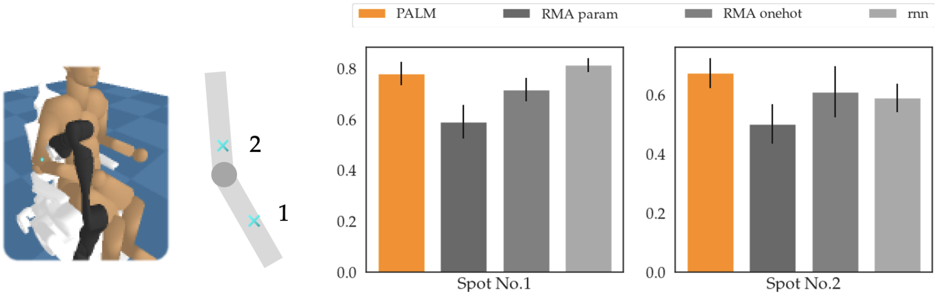

Figure 8: Left: visualization of the two itch spots in user study. Right: relative performance of different methods. The error bar indicates standard error.

For evaluation, we compute *success count*: the total number of time steps where the robot's tool is within 10 centimeters of the itch location. To account for different user dexterity levels, we then normalize the result against the highest and the lowest *success count* on that itch spot for that user. We show the results in Fig. 8. Note that PALM yields high performance in both itch spots and outperforms the RMA baselines. We also find the performance of the RNN baseline to be comparable with PALM for the first task. We hypothesize that this might be due to the real users being able to quickly adapt to the robot's behaviour. This causes the performance of all the methods to be more similar than shown in the simulation experiments.

4.7 Limitations and Failure Cases

Although PALM achieves good average-case performance, it works best with humans sampled near the training distribution. If we pair the robot with an adversarial human, PALM is likely to fail as it lacks a fall-back safety policy.

[2] Please refer to the appendix for more details on the setup and the training distributions.

The major limitation of PALM is the requirement of generating a human population. While we provide one way to generate human populations based on weighted human-robot co-optimization, we lack ways to systematically generate diverse and realistic human motions. One important direction for future work is to incorporate real user data to create training populations. Improving the realism of the training human population is likely a crucial step to supporting transfer to real partners.

One future direction is extending to settings with one patient and one human caregiver. While our framework still applies, this leads to new challenges including (1) whether to learn a joint or separate latent space for human patient and caregiver, (2) how to model a population of human caregivers for training in simulation, and (3) how to model communication between the human caregiver and patient.

5 Conclusion

Generalization is an important task for assistive robotics, and in this paper, we formulate a problem setting that focuses on Out-Of-Distribution users. To that end, we contribute a framework PALM for learning a robot policy that can quickly adapt to new partners at test time. PALM assumes a distribution of training humans and constructs an embedding space for them by learning to predict partner actions. We can further adapt this embedding at test time for new partners. Experiments show that PALM outperforms state-of-the-art approaches. We are excited by the potential of using PALM to enable robotic assistance in the future.

References

[1] D. P. Miller. Assistive robotics: an overview. *Assistive technology and artificial intelligence*, pages 126–136, 1998.

[2] S. W. Brose, D. J. Weber, B. A. Salatin, G. G. Grindle, H. Wang, J. J. Vazquez, and R. A. Cooper. The role of assistive robotics in the lives of persons with disability. *American Journal of Physical Medicine & Rehabilitation*, 89(6):509–521, 2010.

[3] Z. Erickson, V. Gangaram, A. Kapusta, C. K. Liu, and C. C. Kemp. Assistive gym: A physics simulation framework for assistive robotics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10169–10176. IEEE, 2020.

[4] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[5] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

[6] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu. Sim-to-real transfer for biped locomotion. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3503–3510. IEEE, 2019.

[7] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[8] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

[9] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel. Learning to manipulate deformable objects without demonstrations. *arXiv preprint arXiv:1910.13439*, 2019.

[10] D. Seita, A. Ganapathi, R. Hoque, M. Hwang, E. Cen, A. K. Tanwani, A. Balakrishna, B. Thananjeyan, J. Ichnowski, N. Jamali, K. Yamane, S. Iba, J. Canny, and K. Goldberg. Deep Imitation Learning of Sequential Fabric Smoothing From an Algorithmic Supervisor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[11] J. Hietala, D. Blanco-Mulero, G. Alcan, and V. Kyrki. Closing the sim2real gap in dynamic cloth manipulation. *arXiv preprint arXiv:2109.04771*, 2021.

[12] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[13] D. Balduzzi, M. Garnelo, Y. Bachrach, W. Czarnecki, J. Perolat, M. Jaderberg, and T. Graepel. Open-ended learning in symmetric zero-sum games. In *International Conference on Machine Learning*, pages 434–443. PMLR, 2019.

[14] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

[15] M. Carroll, R. Shah, M. K. Ho, T. Griffiths, S. Seshia, P. Abbeel, and A. Dragan. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019.

[16] D. Strouse, K. McKee, M. Botvinick, E. Hughes, and R. Everett. Collaborating with humans without human data. *Advances in Neural Information Processing Systems*, 34, 2021.

[17] A. Xie, D. P. Losey, R. Tolsma, C. Finn, and D. Sadigh. Learning latent representations to influence multi-agent interaction. *arXiv preprint arXiv:2011.06619*, 2020.

[18] S. Parekh, S. Habibian, and D. P. Losey. Rili: Robustly influencing latent intent. *arXiv preprint arXiv:2203.12705*, 2022.

[19] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2013. URL https://arxiv.org/abs/1312.6114.

[20] M. Watter, J. T. Springenberg, J. Boedecker, and M. A. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *CoRR*, abs/1506.07365, 2015. URL http://arxiv.org/abs/1506.07365.

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

[22] T. Yoneda, G. Yang, M. R. Walter, and B. C. Stadie. Invariance through latent alignment. *CoRR*, abs/2112.08526, 2021. URL https://arxiv.org/abs/2112.08526.

[23] E. Z. Liu, A. Raghunathan, P. Liang, and C. Finn. Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices. In *International Conference on Machine Learning*, pages 6925–6935. PMLR, 2021.

[24] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *CoRR*, abs/2104.08212, 2021. URL https://arxiv.org/abs/2104.08212.

[25] Z. Erickson, Y. Gu, and C. C. Kemp. Assistive VR gym: Interactions with real people to improve virtual assistive robots. *CoRR*, abs/2007.04959, 2020. URL https://arxiv.org/abs/2007.04959.

[26] A. Clegg, Z. Erickson, P. Grady, G. Turk, C. C. Kemp, and C. K. Liu. Learning to collaborate from simulation for robot-assisted dressing. *IEEE Robotics and Automation Letters*, 5(2): 2746–2753, 2020.

[27] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Józefowicz, and S. Bengio. Generating sentences from a continuous space. *CoRR*, abs/1511.06349, 2015. URL http://arxiv.org/abs/1511.06349.

[28] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. *CoRR*, abs/1506.02216, 2015. URL http://arxiv.org/abs/1506.02216.

Supplementary Material

In this supplementary material, We first provide details of a pilot user study in Sec. A. We then present details and visualizations on how we generated $\mathscr{D}_{\text{train}}$, the training distribution of human agents in assistive itch scratching in Sec. B. We then present the main algorithm pseudo-code for PALM in Sec. D. Next we provide implementation details of different methods in our main experiments in Sec. E and the ablation study of PALM in Sec. F.

A User Study

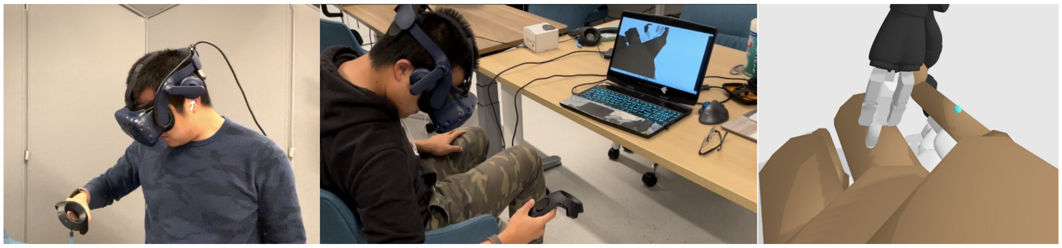

Figure 9: The user study setup. Left and center are the user interacting with virtual robots through the HTC VIVE headset and the hand controller. The right is the first-person view in VR[25].

We conduct a user study with 9 users from age 22 to 29. We simplify the VR environment such that the user has four degrees of freedom: three on the shoulder and one on the elbow. We do not explicitly track the elbow and use the controller rotation and orientation to track the arm motions. We then convert the human pose deltas into actions in the environment. Note that the real user can exceed the action limit in assistive gym (`env.action_space_human.high`). This can be challenging for the robot because it has not seen such action magnitude during training. We use a red marker to remind the users if they have exceeded the action limit.

For each user, we test with two itch spots, one on the upper arm and one on the lower arm. We use randomized orderings of different methods. Each interaction is 150 simulation time steps and roughly six seconds of wall clock time. Within each method, we let the user interact for a total of five trials. We also find that the robot can be initialized in poses that collide with the human at that moment. We discard trials when such collisions are detected.

For training the robot controller, we synthesize humans of six different activity levels ($c_{\text{p}} = 0, 3, 10, 30, 60, 100$), each with two random seeds to simulate users of different activity levels. We then sample 16 itch spots from each human, which results in a total of 192 training humans. The training itch spots do not include the two spots that we test on.

For evaluation, we compute *success count*: the total number of time steps where the robot's tool is within 10 centimeters of the itch location. To account for different user dexterity levels, we then normalize the result against the highest and the lowest *success count* on that itch spot for that user. We show the results in Fig. 8. Note that PALM yields high performance in both itch spots and outperforms the RMA baselines. We also find the performance of the RNN baseline to be comparable with PALM for the first task. We hypothesize that this might be due to the real users being able to quickly adapt to the robot's behaviour. This causes the performance of all the methods to be more similar than shown in the simulation experiments.

B Generating Human Populations

To train our robot using sim2real, we would like to have a set of diverse environments. However, unlike single-agent domain randomization where we can vary environment parameters such as friction, in assistive tasks the environment entails a changing user policy. It is not obvious how to best generate a diverse population that captures user preferences, levels of disabilities, or movement characteristics.

Generating human motions that realistically capture the variation observed in physical human-robot interaction has remained an unsolved challenge in robotics.

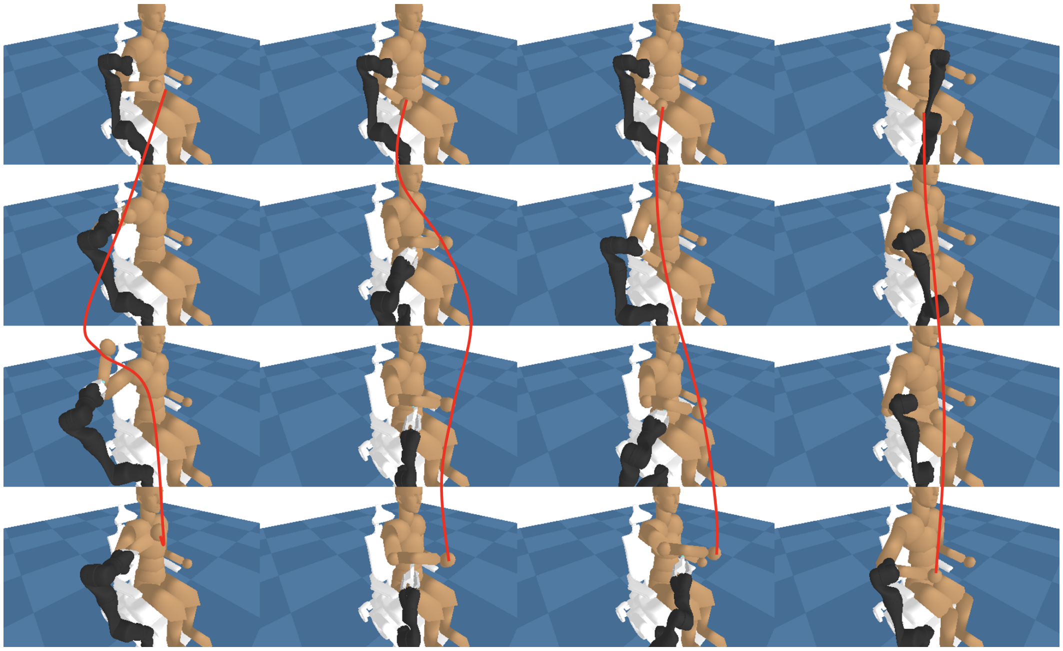

Figure 10: Visualization of humans generated of different activity levels. From left to right we apply action penalties $c_{\mathrm{p}} = 0, 10, 30, 60$. Qualitatively, increasing the penalty results in the human taking more steady actions with less swinging motions. This results in the human being more likely to expose the itch spot for the robot to scratch, as opposed to scratching themselves.

Co-Optimization Prior works in robotic assistance [3, 26] have demonstrated that by optimizing for the same task objective, we can generate human and robot motions that coordinate towards the same goal, such as robot-assisted dressing.

To generate diverse population in itch scratching task, we explore two sources of diversities: (1) we assign different human action penalty c_{p}, where larger penalties lead to the human agent exerting less effort. In the simulation experiment we use $c_{\mathrm{p}} = 3, 3.5, 4$ and in the user study we use $c_{\mathrm{p}} = 0, 3, 10, 30, 60, 100$. (2) We simulate different itch positions on the human's arm and train co-optimized human and robot policies conditioned on them. This leads to qualitatively different strategies for the human and the robot. Note that this serves as the first step to understanding how different methods generalize since we never expect to be able to capture the diversity in humans perfectly. For training, we use Proximal Policy Optimization (PPO) to optimize human and robot policies in an interleaving fashion. Note that we also keep the co-optimized robot policy and use it to obtain expert actions for assistive policy training (see Sec. 3.1 and supplement for full details).

Visualization Here we visualize trajectories from humans with different action penalties in Fig. 10. Note that higher penalties result in the human taking more steady actions of smaller magnitude.

C Comparison with other VAE baselines for sequential data

Note that our method relies on embedding human trajectories as sequential data into a latent space. Our implementation uses the final hidden state of RNN as the input to variational autoencoder. This is based on [27], which has been shown to be effective in embedding and generating sentences. Given that there are other different generative models for sequential data, our framework can be easily combined with them. In fact, wee believe coming up with a better model for human embedding is a future direction.

We hereby provide comparison with another different generative sequential model [28]. Different from [27] that uses only the final hidden state, they construct a latent space for every intermediate step in the sequential model. We keep all the experiment hyper-parameters the same, and concatenate the final hidden state with observation as input to the robot policy. We show the results in

Normalized Reward	Distribution	Our Method [27]	Baseline [28]	RNN
Assistive Reacher	IND	**0.72 ± 0.02**	0.66 ± 0.02	0.62 ± 0.02
	OOD	**0.38 ± 0.10**	0.11 ± 0.01	0.27 ± 0.09
Itch Scratching $\mathscr{D}^1$	IND	0.75 ± 0.01	0.45 ± 0.04	**0.79 ± 0.01**
	OOD	**0.48 ± 0.08**	0.32 ± 0.01	0.25 ± 0.08
Itch Scratching $\mathscr{D}^2$	IND	0.58 ± 0.03	**0.70 ± 0.03**	0.44 ± 0.01
	OOD	**0.49 ±.0.02**	0.31 ± 0.05	0.40 ± 0.02
Itch Scratching $\mathscr{D}^3$	IND	**0.34 ± 0.02**	0.23 ± 0.22	0.18 ± 0.01
	OOD	**0.25 ± 0.01**	0.13 ± 0.01	0.16 ± 0.01

Table 1: Comparison with RNN-VAE baseline [28]

D Algorithm

We present the main algorithm for PALM assume we have access to training and test distributions $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{test}}$.

Algorithm 1 Prediction-based Assistive Latent eMbedding Training

```
Randomly initialize base policy $\pi$, encoder $\mathcal{E}_\phi$ parameterized by $\phi$, decoder $\mathcal{D}_\theta$ parameterized by $\theta$. Empty replay buffer $D_1$, window size $w$
for itr = 1, .., $N_{\text{itr}}$ do
    for $i = 1, .., N_{\text{batch}}$ do
        Sample $\pi_{\text{H}_i} \sim \mathscr{D}_{\text{train}}$, optionally find pre-trained expert robot policy $\pi_{\text{E}i}$
        $o_0 \leftarrow$ env.reset()
        Initialize history $H \leftarrow \phi$
        for $t = 0, ..., T$ do
            Get latest $w$ steps from $H$: $H_{-w} \leftarrow H[-w:]$.
            $z_t \leftarrow \mathcal{E}_\phi(o_t, H_{-w})$
            $a_{\text{H}t} \leftarrow \pi_{\text{H}_i}(o_0)$
            $a_{\text{R}t} \leftarrow \pi(o_t, z_t), a_{\text{e}t} \leftarrow \pi_{\text{E}i}(o_t)$
            $o_{t+1} \leftarrow$ env.step($a_{\text{H}t}, a_{\text{R}t}$)
            Store $(o_t, a_{\text{H}t}, a_{\text{R}t}, a_{\text{e}t}, H_{-w})$ in $D_1$
        end
    end
    for $j = 1, ..., N_{\text{opt}}$ do
        Sample a batch of $(o_t, a_{\text{H}t}, a_{\text{R}t}, H_{-\tau})$ from $D_1$
        Compute $\mathcal{L}_{\text{pred}}$ using Eq. (3), $\mathcal{L}_{\text{PPO}}$ using [21] and $\mathcal{L}_{\text{BC}} = \sum_t ||a_t^{\text{exp}} - a_t^R||^2$
        Optimize $\theta, \phi, \pi$ for $\mathcal{L}_{\text{PALM}} = \lambda_{\text{pred}}\mathcal{L}_{\text{pred}} + \lambda_{\text{PPO}}\mathcal{L}_{\text{PPO}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}}$
    end
end
```

E Additional Training and Implementation Details

In our experiments in both the assistive reaching and assistive itch scratching, we use a recurrent network over a sliding window of 4-time steps, each of which is a concatenated vector of observation o_t, human action $a_{\text{H}t-i}$ and robot actions $a_{\text{R}t-i}$. For the current time step t, we use zero vector for $a_{\text{H}t}$ and $a_{\text{R}t}$. We set the latent space dimension to be four, and use a recurrent network with six layers. Our base policy network has four dimensions and hidden size of 100.

PALM Training We use $\lambda_{\text{PPO}} = 0.1, \lambda_{\text{BC}} = 1, \lambda_{\text{pred}} = 0.1$ in our experiments. We find that the behaviour cloning loss is essential for the Assistive Itch Scratching task. Under this hyperparameter

Algorithm 2 Prediction-based Assistive Latent eMbedding Test Time Adaptation

```
Sample $\pi_{\mathrm{H}} \sim \mathscr{D}_{\text{test}}$, Initialize history $H \leftarrow \phi$, Empty trajectory data $\tau$
for $i = 0, .., N_{\text{adapt}}$ do
    $o_0 \leftarrow$ env.reset()
    for $t = 0, ..., T$ do
        Get latest $w$ steps from $H$: $H_{-w} \leftarrow H[-w:]$.
        $z_t \leftarrow \mathcal{E}_\phi(o_t, H_{-w})$
        $a_{\mathrm{R}t} \leftarrow \pi(o_t, z_t)$
        $o_{t+1} \leftarrow$ env.step($a_{\mathrm{H}t}, a_{\mathrm{R}t}$)
        Store $(o_t, a_{\mathrm{H}t}, H_{-w})$ in $\tau$
    end
    Compute $\mathcal{L}_{\text{pred}}$ using Eq. (3) on $\tau$, $\theta \rightarrow \theta - \delta \nabla_\theta \mathcal{L}_{\text{pred}}(\mathcal{E}_\theta, \tau)$.
end
```

setting, we train for 200 iterations. During each iteration, we collect 19,200 state-action transitions, which is evenly divided into 20 mini-batches. Each mini-batch is fed to the base policy and encoder for 30 rounds to compute the loss and error for back-propagation. We set the learning rate to be 0.00005.

For PALM prediction training, we use a three-layer decoder with hidden size 12 to predict the next human action from the hidden state from the encoder. To implement the KL regularization, follow the standard VAE approach. We use two linear networks to transform the encoder hidden state into μ and σ, which denote the mean and the standard deviation of the latent space. We then compute approximate KL divergence to normal distribution on this latent distribution.

PALM Test Time Optimization At test time, we roll out the trained robot policy and collect data with the same user for 25 iterations, or 2,500 time steps. This amounts to 150 seconds of wall clock time. We then optimize for the prediction loss (including KL regularization term) using learning rate of 0.0001 for one to five steps, and use the one with the lowest loss. We empirically find the hyperparameters by doing the same process with humans from the training distribution, where we collect a mini training set and mini evaluation set both of 25 iterations. We use the mini training set to find the learning rate and use the evaluation set to ensure there is no over-fitting.

RILI/LILI Training We follow a similar approach to PALM, except that we learn to predict the next state o_{t+1} and scalar reward.

RMA Training We follow the two-phase training procedure in [7]. Note that we find it crucial in phase 2 to train the encoder with on-policy data, meaning that the regression data is collected by rolling out actions output by the "recurrent learner", not the trained network from phase 1. The phase 1 network is used simply for generating labels.

RNN/MLP Training. For RNN, we directly feed the hidden state of the recurrent encoder to the policy network. The architectural difference between RNN and PALM is that we do not concatenate the current observation o_t to the encoded output. To ensure that the policy has at least the same capacity as PALM, we use a base policy with the same number of parameters as in PALM.

F Ablation Studies

PALM Baselines	Reach	Itch $\mathscr{D}^1$	Itch $\mathscr{D}^2$	Itch $\mathscr{D}^3$
PALM test optim	**0.43 ± 0.13**	**0.51 ± 0.08**	**0.50 ± 0.02**	**0.29 ± 0.02**
PALM w/o test optim	0.38 ± 0.10	0.48 ± 0.08	0.49 ± 0.02	0.26 ± 0.01
No $\mathcal{L}_{\text{pred}}$	0.30 ± 0.05	0.50 ± 0.08	0.45 ± 0.02	0.25 ± 0.01
$c_{\text{KL}} = 0$	0.32 ± 0.08	0.47 ± 0.04	0.46 ± 0.01	0.23 ± 0.02
Frozen $\mathcal{E}$	0.24 ± 0.04	0.21 ± 0.06	0.15 ± 0.07	0.11 ± 0.05

Table 2: Normalized Reward on $\mathscr{D}_{\text{test}}$, standard deviation over 3 seeds.

We include ablation studies of PALM in the main experiment in Sec. 4, where we study the effect of test-time optimization, prediction loss, KL regularization and jointly training encoder $\mathcal{E}$ and policy π.

In the assistive reaching experiment, we observe that test-time optimization, $\mathcal{L}_{\text{pred}}$, KL regularization, and joint training all contribute to the OOD performance.

In the assistive itch scratching experiment, test time optimization and $\mathcal{L}_{\text{pred}}$ improve experiment results in all the settings. Applying KL regularization provides some gain in the complex distribution $\mathcal{D}_3$, but does not lead to improvement in simpler distribution $\mathcal{D}_1$ and $\mathcal{D}_2$.