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, dsbrown@cs.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. https://adaptive-caregiver.github.io.

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

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

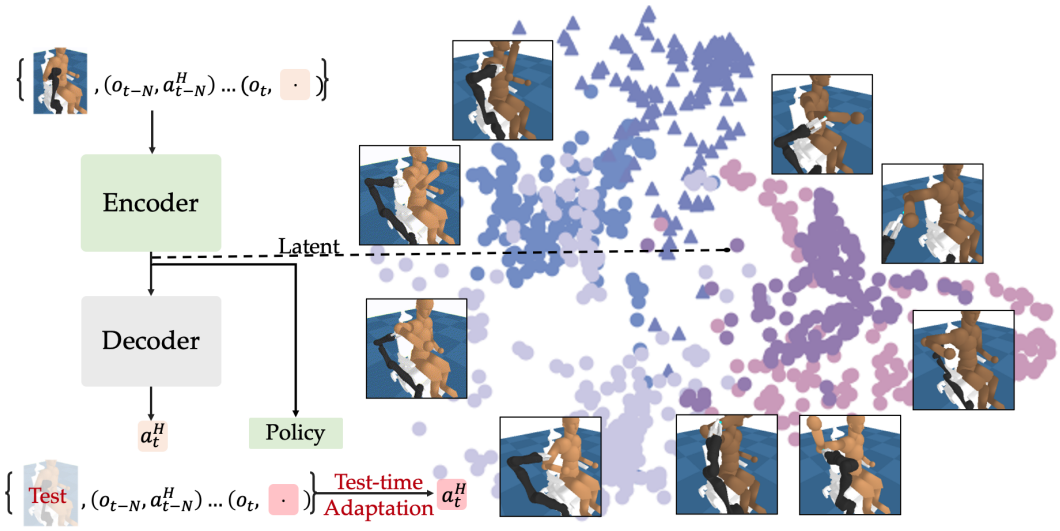

Figure 1: The framework for jointly learning the Personalized Latent Embedding Space and the robot policy. During training time, we train all components end-to-end to optimize for action prediction (orange) and the robot policy (green). At test time, we can further optimize for this objective to perform test-time adaptation (red). The resulting latent space captures the underlying structure of the preferences and strategies of the training humans.

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 latent 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

h𝒮, 𝒜$_R$, 𝒜$_H$, T, Ω$_R$, Ω$_H$, 𝒪, R, 𝛾i. Here $\mathcal{S}$ is the state space and $\mathcal{A}_R$, $\mathcal{A}_H$ are the human's and the robot's action spaces, respectively. The human and the robot share a real-valued reward function $R: \mathcal{S} \times \mathcal{A}_R \times \mathcal{A}_H \to \mathbb{R}$; 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. $T: \mathcal{S} \times \mathcal{A}_R \times \mathcal{A}_H \times \mathcal{S} \to [0, 1]$ 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 $\mathcal{O}: \mathcal{S} \times \mathcal{A}_R \times \mathcal{A}_H \times \Omega_R \times \Omega_H \to [0, 1]$ represents the observation probabilities. We denote the horizon of the MDP by γ.

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 the Dec-POMDP. The robot does not know this a-priori and thus needs to adapt to individual users' capabilities.

The Robotic Caregiving 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 $\pi_H: \Omega_H^* \to [0, 1]$ be the human policy that maps from local histories of observations $\mathbf{o}_H = (o_H^0, \ldots, o_H^t)$ over Ω_H to actions. We define two distributions of human policies $\pi_H \in \mathcal{D}_{train}, \mathcal{D}_{test}$. In the assistive itch scratching, $\mathcal{D}_{train}$ 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*. $\mathcal{D}_{test}$ contains *out-of-distribution* humans whose itch position differ from those in the $\mathcal{D}_{train}$. At training time, the robot has access to $\mathcal{D}_{train}$. 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 $\pi_H \sim \mathcal{D}_{test}$ 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 $\pi_R: \Omega_R^* \to \mathcal{A}_R$ such that we achieve the best performance on test humans. Concretely, we define the performance of the robot and human as

$$J(\pi_R, \pi_H) = \mathbb{E}\left[\sum_{t=0}^{\gamma} R(s^t, a_R^t, a_H^t) \mid a_R^t \sim \pi_R(\mathbf{o}_R^t), a_H^t \sim \pi_H(\mathbf{o}_H^t)\right] \quad (1)$$

Only given access to $\mathcal{D}_{train}$, our objective is to find the robot policy $\pi_R^* = \arg\max_{\pi_R} J(\pi_R, \pi_H) \sim \mathcal{D}_{test}$.

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. We then highlight the advantage of action prediction in contrast to prior works. Finally, we describe how we can optimize PALM at test time to work with out-of-distribution humans.

3.1 Learning an Assistive Latent Space

Given a training distribution of humans $\mathcal{D}_{train}$ [1], we would like to learn a robot policy that can 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 *learn a 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 infers this latent embedding and uses it to generate personalized assistance.

We denote the latent space as $\mathcal{Z}$. $E_\phi(z \mid o_H^{1:t})$, where E_ϕ encodes the trajectory $o_H^{1:t}$ so far and outputs latent vector $z \in \mathcal{Z}$. The robot uses this latent space to compute its actions $a_R^t = \pi_R(o_R^t, z)$. We train

[1] we describe how we generate this distribution in Sec. 4.1

the base policy π_a and the latent space encoder E_ϕ jointly as they are interdependent [19] — better robot policy leads to different trajectories across humans, which in turn leads to distinguishing z. See Appendix D on training details. Ideally, we would like z to capture sufficient information to differentiate the humans, similar to performing a "dimensionality reduction" on human policies π_h. We hereby introduce different objectives for learning such latent space, and how our method — learning by action prediction — makes a good fit for the assistive personalization problem.

3.2 How to Construct the Latent Space

Prior work and limitations LILI [17] and RILI [18] learn a latent embedding of the humans by predicting the next observations and rewards. While they have been shown to work in predicting and influencing human behaviours, both methods assume access to the reward function at test time, which we do not have access to in the assistive setting — we don't know a-priori the preference and needs of a new user. RMA [7] enables fast robot adaptation by learning a latent space of environment parameters, such as friction, payload, terrain type, etc. While it works for a single robot, it is unclear in human-robot settings, how we can encode human motions and preferences as environment parameters.

Learning by action prediction. Given history $\tau_{0:t} = (s_0, a_0^R, a_0^H, \ldots, s_t)$ of t robot observation and human action[2] pairs, as outlined in Fig. 1, we embed this trajectory to a low-dimensional manifold and use it to predict a_t^H. The intuition is that if we are able to predict the next action by this human, we extract the sufficient information about the human's policy π_h. The latent vector z is representative of the trajectory so far and indicative of the person's future actions. We do this by training a decoder D_ϕ parameterized by ϕ to predict the next action from the encoder's output $z = E_\theta(s_0, a_0^R, a_0^H, \ldots, s_t)$.

$$L_{\text{pred}} = \min_{\theta,\phi} \| D_\phi(z, s_t) - a_t^H \|^2 + \beta_{KL} D_{KL}(E_\theta(s_0, a_0^R, a_0^H, \ldots, s_t) \| \mathcal{N}(0, I)) \quad (2)$$

The encoder E is a recurrent neural network parameterized by θ. Here the second term is a regularization term motivated by Variational Autoencoder [20, 21], that enforces the latent space to follow a normal prior distribution. This encourages nearby terms in the latent space to encode 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.

3.3 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 $\|E(\ldots) - z\|^2$. 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*.

Since 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. (2) 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, $\mathcal{D}$, and perform a few gradient steps to optimize both the encoder and decoder for Eq. (2): $(\theta, \phi) \leftarrow (\theta, \phi) - \nabla_{(\theta,\phi)} L_{\text{pred}}(E_\theta, D_\phi, \mathcal{D})$.

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

[2] We do not assume access to the person's sensorimotor action (e.g. joint torques). We define human action as change in the person's Cartesian pose, which can be tracked externally.

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 examples meant for demonstrating the effectiveness of the algorithm, and we do not claim to solve the full robotic caregiving problem.

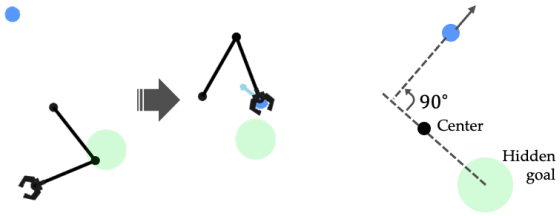

Figure 2: The assistive reacher environment. Left: the robot's goal is to move the human agent towards the hidden target. Right: the hidden goal's position can be inferred 90 degrees from the humans force output.

Assistive Reacher (Fig. 2) is 2D collaborative environment where a two-link robot arm assists a point human agent to get to the target position. This target is located at $\ell\cos\theta_H - \ell\sin\theta_H$, where ℓ is a fixed value, and $\theta_H \sim U(-\pi, \pi)$ is known to the human, but not the robot. The human agent is initialized randomly in the 2D plane with random hidden parameters $\theta_H \sim U(-\pi, \pi)$. The robot can only identify the target position by physical interactions — once the robot initiates contact, the human applies a force $F_H = \cos\theta_H, \frac{\pi}{2} - \sin\theta_H, \frac{\pi}{2}$. Only by recognizing the human in terms of $\theta_H - \theta_H$ can the robot compensate the force, and successfully move the human to the hidden target. Each episode has 40 timesteps.

The Scope of Generalization. We define D_{train} as 36 samples uniformly sampled from $\theta_H \sim U(-\frac{\pi}{2}, -\theta_H \sim U(0.5, 1.5)$ and D_{test} as 12 samples uniformly sampled from $\theta_H \sim U(\frac{\pi}{2}, -\theta_H \sim U(0.5, 1.5)$.

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. The human has limited mobility — they can only move the 10 joints on the right arm and upper chest, and needs the robot's assistance to scratch the itch. An itch spot is randomly generated on the human's right arm. The robot does not directly observe the itch spot, and relies on interaction with the human to infer its location. Each episode has 100 timesteps.

We use co-optimization to create D_{train} and D_{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 $r_p = \alpha_p \|a_H\|^2$ where α_p is a constant controlling the penalty. The overall objective becomes

$$\max_{\pi_R, \pi_H} E\left[\sum_t r_t(s_t, a_H, a_R) - \alpha_p \|a_H\|^2\right] \quad (3)$$

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 $\alpha_p = 3 - 3.5 - 4$, 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.

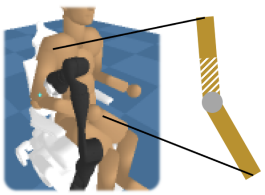

Figure 3: Definition of D_{train} and D_{test}.

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 D_{train} and D_{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. D_{train} consists of three of the four portions and D_{test} consists of the remaining one. We then construct three distribution sets in increasing order of difficulties. In the first distribution D^1, we confine itch positions from a

line-shaped region. In D^2, we sample from all the arm areas. Note that D^1 and D^2 are constructed by setting action penalty $\square_p = 3$. In D^3, we combine humans of $\square_p = 3-3\bullet5-4$, 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.

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, 19, 23]. 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-Param** and **RMA-Onehot**, where we assign each training human a one-hot vector. For RMA-Param, we use a three-dimensional vector that includes the $\square$–$\square$ position of the itch position and the arm index. When 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 $\square_p$.

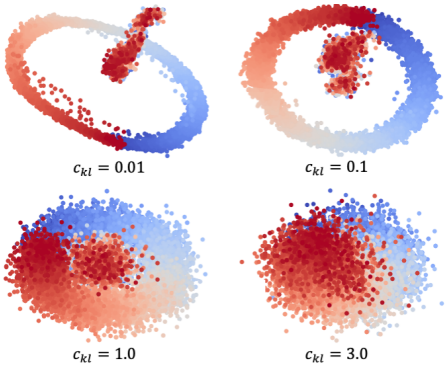

Figure 4: Latent space of PALM in the assistive reacher environment when we can sample humans $\square_H$ 2 » $\square$ (red)–$\square$ (blue)... continuously.

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 L_{pred} and policy loss L_{pol}. We also use the KL term in Eq. (2) 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 L_{pred}: the model shares the same encoder and policy network architecture, yet we don't optimize for L_{pred}. By removing the prediction loss in Fig. 1, the latent space is not explicitly trained to contain human information. (2) No $\square_{KL}$: 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 $\square_H$ 2 » $\square$–$\square$ ..continuously. We train PALM with different amount of prior regularization, $\square_{\square\square}$ from Eq. (2). 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.

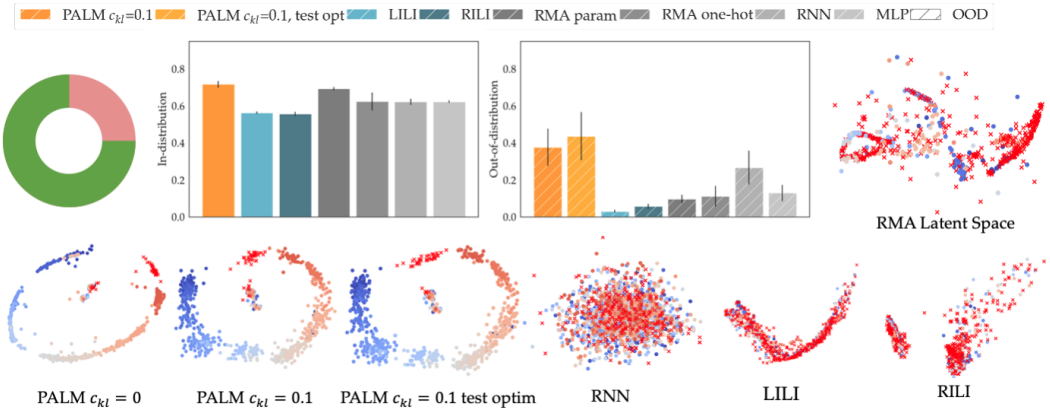

Figure 5: Top left: evaluation of PALM and baselines on in-distribution (green) and out-of-distribution (pink) humans. Right and bottom: visualization of the latent embeddings of different methods. OOD humans are highlighted in red crosses. Best viewed electronically.

We average test results using 100 episodes and visualize the results in Fig. 4.3. Given that humans are parameterized by $\phi_H - \phi_\pi$, the ideal embedding space looks like a ring with a small blob in the center. The ring corresponds to the 2D projection of ϕ_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 ϕ_H, it can learn a latent space that characterizes ϕ_H. Interestingly, varying the amount of regularization qualitatively affects the shape of the latent space. Setting the VAE regularization $\phi_{KL} = 0.1$ recovers a latent space that most resembles to the ideal latent space.

4.4 Assistive Reacher Main Experiment

Experiment Setting. We use the finite D_{train} described in Sec. 4.1 and train all baselines for 200 epochs with 512 batch size. We then evaluate the trained policies on D_{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.

The visualization also offers some insight into why having ϕ_{KL} regularization is helpful for generalization. Compare the latent space of PALM $\phi_{KL} = 0$ and PALM $\phi_{KL} = 0.1$, 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

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 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.

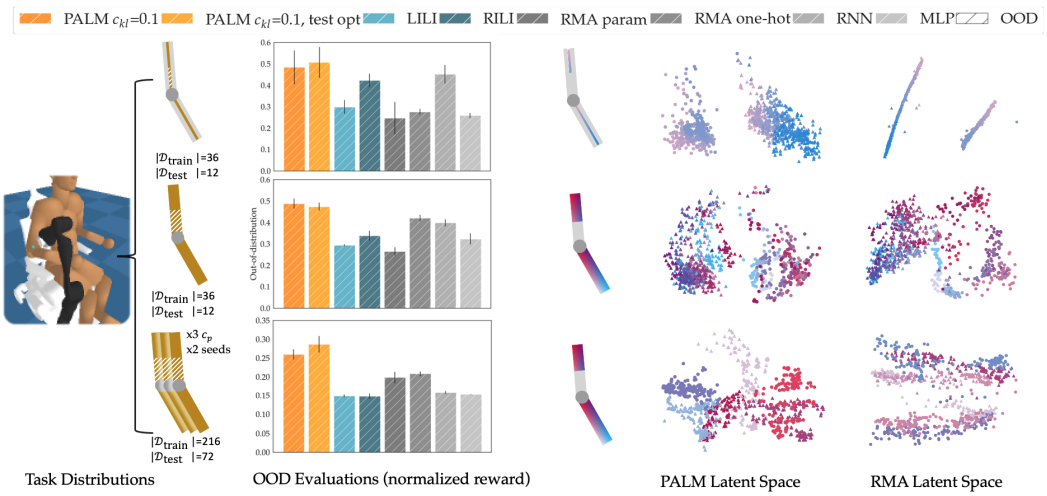

Figure 6: In assistive itch scratching, we sample humans by different itch positions and activity levels (varying action penalty λ_p). We visualize the in- and out-of-distribution humans on the two-link arm figures. We also visualize the embedding space of PALM and RMA, where we color-code the embeddings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

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.

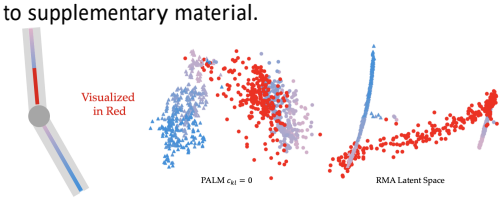

Figure 7: To better under extrapolations to OOD human, which have new itch locations highlighted in red on the left, we visualize the embeddings of both the IND (colored) and OOD (red) humans on the right.

4.6 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.

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) learning a joint or separate latent space for human patient and caregiver, (2) modeling a population of human caregivers for training in simulation, and (3) modeling 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.

Acknowledgments

We would like to thank Ashish Kumar for discussions on the RMA method and Annie Xie for providing the implementation for LILI. We would also like to thank Charlie C. Kemp for feedbacks and insights on assistive tasks. This research was supported by the NSF National Robotics Initiative.

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] 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.

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

[21] 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.

[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] 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.

[24] 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.

[25] A. Gupta, A. Pacchiano, Y. Zhai, S. M. Kakade, and S. Levine. Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity. *arXiv preprint arXiv:2210.09579*, 2022.

[26] Y. Zhai, C. Baek, Z. Zhou, J. Jiao, and Y. Ma. Computational benefits of intermediate rewards for goal-reaching policy learning. *Journal of Artificial Intelligence Research*, 73:847–896, 2022.

[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.

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

[30] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

Supplementary Material

In this supplementary material, We first present details and visualizations on how we generated D_{train}, the training distribution of human agents in assistive itch scratching in Sec. A. We then present the main algorithm pseudo-code for PALM in Sec. C. Next we provide implementation details of different methods in our main experiments in Sec. D and the ablation study of PALM in Sec. E and an evaluation of PALM with a second task, bed bathing assistance, in Sec. F.

A 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.

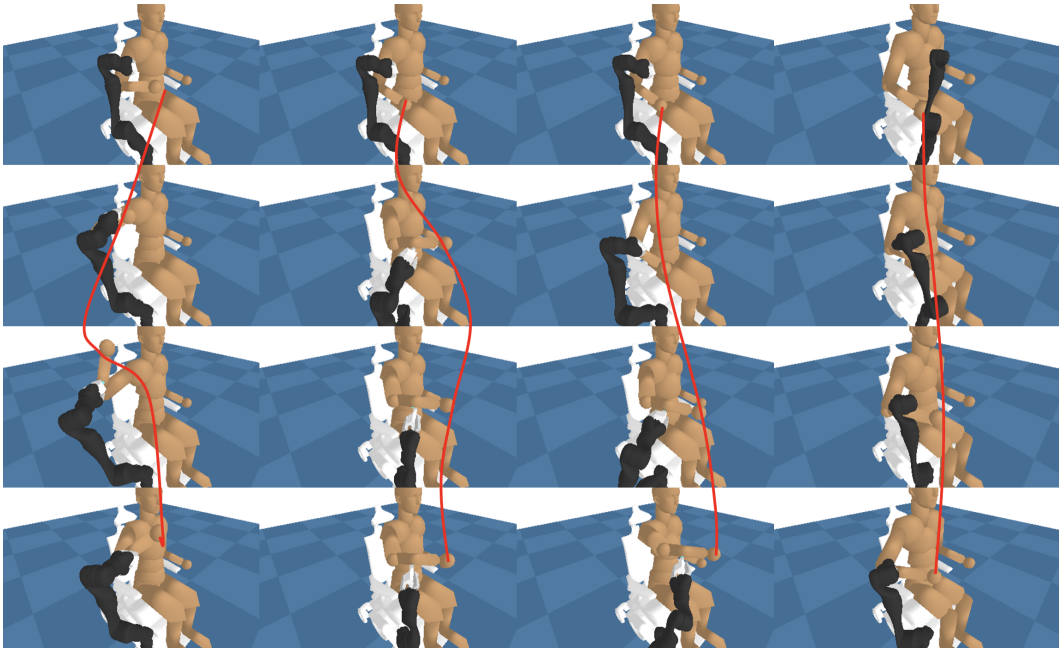

Figure 8: Visualization of humans generated of different activity levels. From left to right we apply action penalties $\alpha_p = 0{-}10{-}30{-}60$. Qualitatively, increasing the penalty results in the human taking more steady actions with less swinging motions. This results in the human being more likely to expose the itch spot for the robot to scratch, as opposed to scratching themselves.

Co-Optimization Prior works in robotic assistance [3, 24] 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. Further more, we can leverage reward engineering [25, 26] to generate a diverse set of motions.

To generate diverse population in itch scratching task, we explore two sources of diversities: (1) we assign different human action penalty α_p, where larger penalties lead to the human agent exerting less effort. In the simulation experiment we use $\alpha_p = 3{-}3{\bullet}5{-}4$.

(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. 9. Note that higher penalties result in the human taking more steady actions of smaller magnitude.

B 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 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 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 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]

C Algorithm

We present the main algorithm for PALM assume we have access to training and test distributions D_{train} and D_{test}.

D 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^H_{t-1} and robot actions a^R_{t-1}. For the current time step t, we use zero vector for a^H_t and a^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.

D.1 PPO and Behaviour Cloning

We need to train the base policy π_θ and the latent space encoder $E_{\psi_1}(z|o;\psi_2)$ jointly because they are interdependent — o is the input to π_θ and ψ_2 decides the data distribution which leads to o. We simultaneously optimize the prediction loss in Eq. (2) and the policy loss using PPO [29] algorithm. See Appendix D for more training details. To amend for the instability of training with a population of humans, we leverage Behaviour Cloning, where we use expert robot policies obtained via co-optimization (see Appendix A) to supervise π_θ on on-policy data. More specifically, we query the expert actions a^{exp}_t in a DAgger fashion[30] [3] during training, and optimize π^a_θ to minimize deviation from it: $L_{BC} = \mathbb{E}_t[||\pi^a_\theta - a^{exp}_t||^2]$. The overall policy optimization loss include three terms: latent prediction loss, PPO loss and the Behaviour Cloning loss: $L_{PALM} = L_{pred}, L_{PPO}, L_{BC}$.

[3] This ensures that we encounter no distribution shift at deployment time.

Algorithm 1 Prediction-based Assistive Latent eMbedding Training

Randomly initialize base policy π, encoder E_ϕ parameterized by ϕ, decoder D_ψ parameterized by ψ. Empty replay buffer $\mathcal{D}$, window size k
for itr = 1, ..., N_{itr} **do**
  for t = 1, ..., N_{batch} **do**
    Sample $s_H \sim D_{\text{train}}$, optionally find pre-trained expert robot policy π_E
    o_0 env.reset()
    Initialize history h
    for t = 0, ..., T **do**
      Get latest k steps from h: $h_t = [o_{t-k+1}, \ldots, o_t]$
      $z_t \sim E_\phi(h_t)$
      $a_H \sim \pi_H(o_0)$
      $a_R \sim \pi(o_t, a_H, z_t)$
      o_{t+1} env.step(a_H, a_R)
      Store $(o_t, a_H, a_R, a_e, h_t)$ in $\mathcal{D}$
    end
  end
  for t = 1, ..., N_{opt} **do**
    Sample a batch of (o_t, a_H, a_R, h_t) from $\mathcal{D}$
    Compute L_{pred} using Eq. (2), L_{PPO} using [29] and $L_{BC} = \|a_H^{\text{exp}} - a_H\|^2$
    Optimize ϕ, ψ, π for $L_{PALM} = \alpha_{\text{pred}} L_{\text{pred}} + \alpha_{\text{PPO}} L_{\text{PPO}} + \alpha_{BC} L_{BC}$
  end
end

Algorithm 2 Prediction-based Assistive Latent eMbedding Test Time Adaptation

Sample $s_H \sim D_{\text{test}}$, Initialize history h, Empty trajectory data $\mathcal{D}$
for t = 0, ..., N_{adapt} **do**
  o_0 env.reset()
  for t = 0, ..., T **do**
    Get latest k steps from h: $h_t = [o_{t-k+1}, \ldots, o_t]$
    $z_t \sim E_\phi(h_t)$
    $a_R \sim \pi(o_t, z_t)$
    o_{t+1} env.step(a_H, a_R)
    Store (o_t, a_H, a_R, h_t) in $\mathcal{D}$
  end
  Compute L_{pred} using Eq. (2) on $\mathcal{D}$, $\phi \leftarrow \phi - \nabla_\phi L_{\text{pred}}(E_\phi)$
end

D.2 Hyperparameters

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

E Ablation Studies

PALM Baselines	Reach	Itch D^1	Itch D^2	Itch 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 L_{pred}	0.30 0.05	0.50 0.08	0.45 0.02	0.25 0.01
$_{KL} = 0$	0.32 0.08	0.47 0.04	0.46 0.01	0.23 0.02
Frozen E	0.24 0.04	0.21 0.06	0.15 0.07	0.11 0.05

Table 2: Normalized Reward on D_{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 E and policy $\square$.

In the assistive reaching experiment, we observe that test-time optimization, L_{pred}, KL regularization, and joint training all contribute to the OOD performance.

In the assistive itch scratching experiment, test time optimization and L_{pred} improve experiment results in all the settings. Applying KL regularization provides some gain in the complex distribution D_3, but does not lead to improvement in simpler distribution D_1 and D_2.

F Additional Experiment: Bed Bathing Task

We further evaluate the performance of PALM in another assistive robotics task: robot-assisted bathing. This task is a modified version of the bathing task introduced in Assistive Gym [3]. In this task, we have a human lying on a tilted bed, with a robot mounted on the nightstand. The human can move their right arm, and there is an identified patch of skin to be cleaned. Unlike the original bed bathing task, where the entire right arm is covered in target points to be cleaned, we instead initialize a fixed size region of points to be cleaned at a uniform randomly selected location along the surface of the right forearm. The patch spans 10 centimeter along and 150 degrees around the forearm. Only the human knows the center position of the patch, and the robot must infer the location of the patch based on observations of the human motion. The points along the body are cleaned whenever the robot initiates contact with the spot using its end-effector and applies a positive normal force. The task reward is based on how many points are cleaned.

We generate synthetic humans by co-optimizing humans and robots conditioned on the centroid position of the region to be cleaned. During co-optimization, we adopt the formulation in [3] where both the human and the robot observe the centroid position along the human forearm. This helps induce collaborative behaviour for the human and robot policies, where we then use the resulting human as the synthetic population. During training, we blind of robot policy of the centroid position. We use the co-optimized robot (observes the centroid position) as the oracle for querying expert actions for Behaviour Cloning. We sample 18 humans from the training distribution, and save 6 humans from the held-out distribution as the out-of-distribution evaluation. We use the same hyperparameters as the itch scratching experiment.

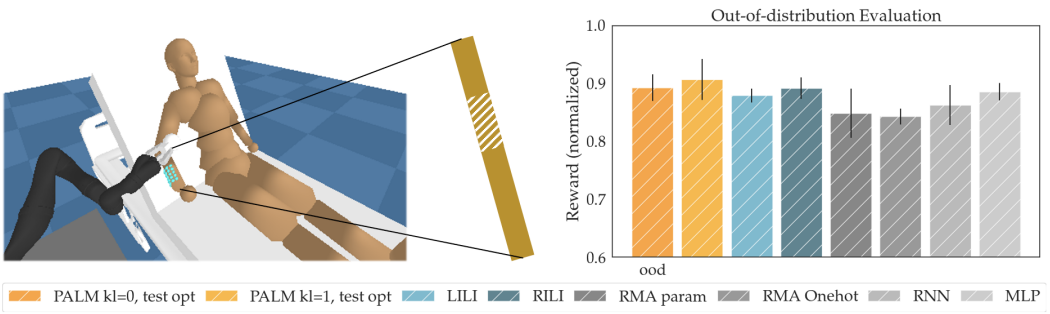

Figure 9: Visualization of the bed bathing task (left), where the human lies on a tilted bed with a table-mounted robot to clean an area on their arm. The area is random initialized (middle) on the forearm, where we hold out a quarter of the length as the held out distribution. The results of the held-out distribution is visualized on the right.

As we see in the results in Fig. 9, PALM achieves better out-of-distribution results compared to the baseline methods. We also observe that this performance gap is smaller than the itch scratching task. We believe this is due to the nature of the bed bathing task, where a robot controller that maintains contact with the human's forearm can be sufficient for solving the task if the human policy learns to move and rotate their forearm accordingly to help the robot.