Fine-Tuning Robot Policies While Maintaining User Privacy

Anonymous Authors

***Abstract*— Recent works introduce general-purpose robot policies. These policies provide a strong prior over how robots should behave — e.g., how a robot arm should manipulate food items. But in order for robots to match an individual person's needs, users typically *fine-tune* these generalized policies — e.g., showing the robot arm how to make their own preferred dinners. Importantly, during the process of personalizing robots, end-users leak data about their preferences, habits, and styles (e.g., the foods they prefer to eat). Other agents can simply roll-out the fine-tuned policy and see these personally-trained behaviors. This leads to a fundamental challenge: how can we develop robots that *personalize* actions while keeping learning *private* from external agents? We here explore this emerging topic in human-robot interaction and develop *PRoP*, a model-agnostic framework for personalized and private robot policies. Our core idea is to equip each user with a unique key; this key is then used to mathematically transform the weights of the robot's network. With the correct key, the robot's policy switches to match that user's preferences — but with incorrect keys, the robot reverts to its baseline behaviors. We show the general applicability of our method across multiple model types in imitation learning, reinforcement learning, and classification tasks. PRoP is practically advantageous because it retains the architecture and behaviors of the original policy, and experimentally outperforms existing encoder-based approaches. See videos and code here: https://prop-icra26.github.io**

I. Introduction

Generalist policies enable robots to learn multiple tasks [1], [2]. So far these methods have traditionally been used in research labs and factories. But we envision a future where robots enter domestic settings for assisting humans [3]. For example, consider a robot that is developed to help in a kitchen. This robot will have some initial policy π_0 that users may want to finetune to match their own preferences and requirements. For instance, perhaps the robot knows how to make a hamburger, but individual users prefer different ingredients, condiments, or even specific ways of stacking the burger. This finetuning raises privacy concerns: the manufacturers can share the users' data collected during finetuning with third-parties. Consequently, there is increasing demand for exploring new avenues to maintain the privacy and transparency of robotic agents [4]. Following this, we come to a fundamental scientific question: how do we make systems that can learn and adapt to individual end-users, while still maintaining those user's privacy?

Privacy in machine learning has traditionally been examined from two perspectives. First is data privacy, which concerns safeguarding the sensitive information of individuals represented in the dataset [5]–[8]. Second is model privacy, which focuses on protecting the learned parameters of a neural network through techniques such as encryption or differentially-private learning [9]–[12]. In this work, we adopt a third perspective with respect to robot learning: ensuring that a trained, personalized robot does not leak user preference information to other users. Returning to our example, privacy in this context means that the robot can be finetuned to learn your preferred way of making a burger while preventing unauthorized users from accessing those preferences *even if they have access to the trained model*. In practice, this can be difficult to achieve because — if someone has access to the finetuned model — they can roll-out this model and infer the previous user's preference by watching the robot actions. So how do we safeguard privacy of user preferences? Our insight is that:

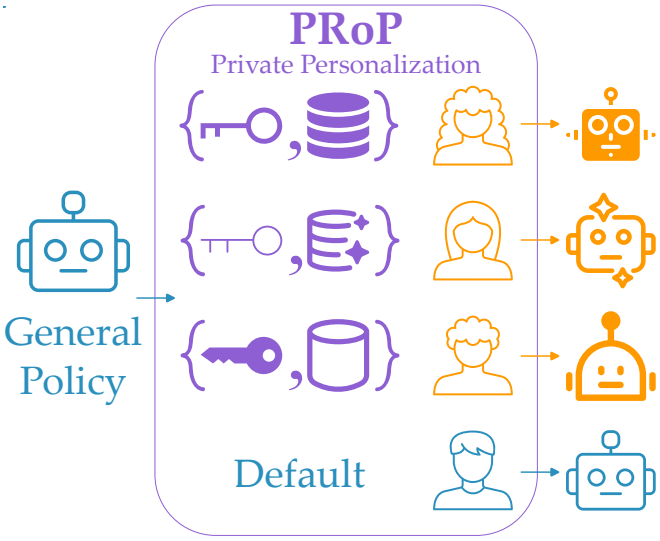

Fig. 1: In human-robot interaction robots are often finetuned to personalize to user-specific needs. The users above have different preferences encoded in their personalized datasets. When the general model is finetuned to the user's personalized dataset, the resulting policy is not *private*. Any user that interacts with the finetuned policy will be able to infer the user's preferences. Instead, we propose **PRoP**: a method that enables private personalization of robot policies to humans. PRoP learns to associate user keys with intermediate transformations of the original policy, causing personalized and private behavior. When users do not provide a key (or provide a key not included in PRoP's training set), they receive the original, pretrained policy.

We can use latent values to transform *intermediate features of a network for enhanced privacy and personalization.*

Concretely, we leverage *keys* (Figure 1). A key is any feature that is unique to the user such as facial structures, vocal patterns, or a textual password. When finetuning the

robot under out approach, a user combines their unique key with the intermediate features of the network and trains it to output their personalized actions. This unique mechanism for personalizing robots safeguards user privacy since preference information remains inaccessible to anyone who does not have the user's key. Without careful design, keys may unintentionally cause the robot to forget its general-purpose policy. But our technical approach avoids this pitfall — and preserves the initial model architecture — by leveraging the key to perform mathematical operations on the intermediate weights. Our proposed mechanism is not tied to a specific network architecture or application as we later demonstrate in our experiments with visual data, imitation learning, MLP classifiers, and reinforcement learning. Indeed, as shown in our experiments on robot arms, users can finetune the robot to make their desired hamburger without losing the robot's previously learned behaviors, and without exposing their preference to other agents. We see this work as a step towards safe and personalized human-robot interaction.

Overall, we make the following contributions:

Key-based Personalization of Robot Policies. We present a formulation for key-based personalization of robot control policies. Under this formalism, the robot learns to personalize to new users' specifications while retaining its original, general behavior. This formalism is nontrivial to implement in a learning algorithm, since the original and conditional policies operate in different domains, i.e., adding a key as the input requires changing the size of the pre-trained architecture. Instead, we use keys to transform the intermediate features of the pre-trained policy, circumventing the need for changing the architecture size.

Personalized and Private Robot Policies. We present our implementation of the aforementioned key-based personalization with privacy guarantees. Our method, **PRoP** (Personalized and Private Robot Policies) retains the original network architecture, exhibits behavior of the original robot policy for unprivileged users, and personalizes to specific users through a privacy-oriented mechanism. Importantly, PRoP extends to arbitrary learning rules and architectures that enables simple, end-to-end training of the model.

Real-world Validation and Empirically Verified Robustness. We empirically test the performance of PRoP in a collection of controlled simulations and real-world studies, including Imitation Learning, Reinforcement Learning, Image Classification, and Task Allocation. We further extend PRoP to more complex settings, such as language prose personalization and key-based obfuscation.

II. Related Works

In recent years there has been a variety of research that analyzes robot-human personalization. This is especially true in Human-Robot Interaction (HRI): humans can provide additional information to robots in order to personalize future interactions. This information may be demonstrations [13], [14], corrections [15], preferences [16], or any combination of the three [17], [18].

Within these settings the robot adjusts its parameters to align with the user's desired behaviors. However, most existing research on human-to-robot personalization ignore the dimension of privacy. While some works exist that outline approaches to protecting user privacy in HRI [19], [20], these primarily focus on protecting the data gathered from users. But even if the robot's training set is secure, once a robot's policy has been personalized, any human with access to the policy will be able to infer the previous operator's preferences. For this reason we focus on privacy not at the data level — which has been explored — but at the *interaction* level, ensuring that user information remains protected during human-robot interaction.

Personalizing Policies. Robots often learn policies instantiated as neural networks. Once trained on a general dataset, neural networks can then be finetuned (i.e., personalized) to new distributions through data aggregation. As previously mentioned, methods can use diverse data types to personalize to new users: learning from preferences, corrections, and demonstrations is standard practice [13]–[16], [18]. The personalized network is learned end-to-end and will affect *all* future users: it is not gated to that particular user. Put simply, subsequent users that have the fine-tuned model can potentially infer the preferences of previous users by directly interacting with the robot.

Large-scale foundational models such as π_0 [1] and Cogact [2] enable researchers to leverage vision and language for out-of-the-box robot control. These foundational models can then be finetuned to researcher-specific datasets: such as for shared-autonomy in robotic assembly [21]. Like their small-scale counterparts, foundational models are *architecture-specific* and need to be fine-tuned end-to-end. Consequently, they are also vulnerable to leaking user information to unauthorized future users. Recent works in low-rank adaptation (LoRA) [22], [23] offer small-scale personalization of large-scale models via intermediate transformations (i.e., without retraining the original policy's weights), but they offer no privacy guarantees. If a third-party acquired the LoRA weights, then the finetuned behavior would be freely available and user privacy is compromised. In this work, we accordingly propose a method that safeguards user privacy by associating each user with a unique key before applying similar transformations. We recognize that foundational models have a fixed architecture and cannot be easily transferred to a new domain with more inputs. For example, the domains of these vision-language-action models (VLA) cannot be trivially transferred from vision-language to vision-language-*password*, where the *password* gates the personalization of the VLA. However, our method does not alter the architecture — instead it works by transforming intermediate features of the network. Hence, our method can be easily applied to pretrained models to ensure private personalization while retaining the original learned behaviors.

Human-Robot Privacy. Existing works on user privacy in HRI typically deal with securing the user's underlying training data: whether that be their demonstrations or demo-

graphic information. [19] presents methods for securing user datasets for robot learning, and [20] addresses how people *perceive* the way their data is utilized in HRI research. On a lower level, works such as [5], [8], [12] employ (ϵ, δ)-differentially private learning to assert privacy guarantees on user-data. Unfortunately, approaches that use differentially-private learning still struggle with budgeting ϵ for complex interaction tasks. For example, the state-of-the-art presented in [12] still exhibits a probability of privacy-failure of approximately 67.2% on the MNIST classification task. Furthermore, these methods are not gated: they attempt to secure *training data* instead of securing the policy's *output*.

Methods that leverage fully homomorphically-encrypted neural networks [9], [10] may solve our problem, but they are computationally infeasible. From our testing, fully-homomorphically encrypted inference can be up to 10^7 times slower than the baseline, which is infeasible for real-time applications in robotics. Instead, we propose that methods that *mimic* encryption by obfuscating the intermediate features of the policy network can be used for private human-robot personalization. Overall, our proposed method will lock the user's preferences behind their key, and even trying mathematically similar keys will not cause the network to output the user's fine-tuned behaviors.

III. Problem Statement

We are interested in human-robot interaction settings where a pre-trained robot policy — represented as a neural network — must be personalized to specific users and remain unchanged for others. In this manuscript, we interchange "user" and "human collaborator," but in practice the "user" could be another robotic agent.

Dynamics. The robot takes actions $u \in \mathcal{U}$ according to its policy π:

$$u \sim \pi\left(\circ \mid x\right) \tag{1}$$

where $x \in \mathcal{X}$ is the system state. The system state transitions in discrete-time with deterministic dynamics according to the state-transition function $f : \mathcal{X} \times \mathcal{U} \mapsto \mathcal{X}$:

$$x^{t+1} = f(x^t, u^t) = f\left(x^t, \{u_i^t\}_{i=0}^N\right) \tag{2}$$

Here $t \in [0, T)$ represents the current timestep. More generally, the system may transition according to the actions of multiple agents. In this general case u in Equation (2) becomes the combined actions of N robots and M humans, i.e., $u = u_{R_1} \cup u_{R_2} \cdots \cup u_{R_N} \cup u_{H_1} \cdots \cup u_{H_M}$, where u_{R_i} is the action of robot i and u_{H_j} is the action of human j.

Personalization of Pretrained Policies. We assume that pretrained policies are represented as neural networks. These neural networks are parameterized by weights θ which follow the gradient descent learning rule shown below:

$$\theta^{\tau+1} = \theta^{\tau} - \alpha \nabla_\theta \mathcal{L}(\theta^{\tau}) \tag{3}$$

where $\mathcal{L}$ is the loss function to be minimized by θ. Traditionally, personalization or finetuning of existing policies occurs by retraining the policy on a new dataset or loss function [14], [17], [24], [25]. For example, on a user-specific level, finetuning the original weights θ^0 can cause the general model to better fit the user's preferences [17], objectives [24], [25], or perceptions [14]. Hereafter we refer to this pretrained robot policy with weights θ^0 as $\pi^\star$.

Private Personalization. Above we outlined a simple dynamical system where the robot's action u is sampled from a distribution conditioned on the system state x. However, for the robot to personalize to multiple users, the robot policy should instead condition on the system state *and* some personal user information. Hence, the personalized policy is a distribution conditioned on the state and user information:

$$u \sim \pi(\circ \mid x, k) \tag{4}$$

where $k \in \mathcal{K}$ is the personal information that the robot should condition their behavior on. This personal information is assumed to be non-fungible. In other words, two separate users should not be able to imitate each other's k. This information can take many forms, such as biographical information, facial features, fingerprints, or a password. In this work we treat k as the bit-representation of a user password. Note that representing the policy as a conditional distribution on k is critical from a privacy perspective. A naive implementation of a privately personalized policy is to have a separate robot policy $\pi_0(x), \ldots, \pi_N(x)$ for each user $\mathcal{H}_0, \ldots, \mathcal{H}_N$. If the developer has access to policy π_i, then they can directly infer the private preferences of the user $\mathcal{H}_i$: this becomes a clear breach of user privacy.

We seek to learn a robot policy that follows the form of Equation (4) while leveraging the architecture and weights of the pretrained policy $\pi^\star$. Satisfying both of these requirements is non-trivial: we assume that the pretrained policy has a mapping $\pi^\star : \mathcal{X} \mapsto \mathcal{U}$ while the personalized robot policy has a mapping $\pi^p : \mathcal{X} \times \mathcal{K} \mapsto \mathcal{U}$. Furthermore, the amended domain should not interfere with the model performance. For example, performance of the personalized model with an incorrect key should be as close as possible to the general policy $\pi^\star$. In what follows, we will discuss our method for privately personalizing a pretrained policy without affecting its architecture or base behavior.

IV. Personalized and Private Robot Policies

In this section we introduce **PRoP**: **P**rivate **Ro**bot **P**olicies. PRoP implements Equation (4), integrating user keys into the original robot policy $\pi^\star$ without affecting its architecture. Towards this end, we leverage encoders that map the human's keys into a latent space, and then use the latent encodings to transform the intermediate features of the robot policy architecture. Our method is summarized in Figure 2.

Concretely, we train the key encoders $\Delta^i_{\varphi_i} : \mathcal{K} \mapsto \mathcal{Z}^i \in \Delta^{0:N}_{\varphi}$ corresponding to each of the intermediate layers of the robot policy network. The encoders are parameterized by φ_i, and since the latent encoding transforms the intermediate features of the policy architecture, the size of the latent space $|\mathcal{Z}^i|$ is determined by the pretrained policy's intermediate architecture at layer i. These encoders are coupled with

the robot's policy $\mathcal{R}_\phi : \mathcal{X} \mapsto \mathcal{U}_\mathcal{R}$ to perform private personalization using keys $k \in \mathcal{K}$. To ensure personalization without affecting the architecture of $\mathcal{R}_\phi$, we perform latent augmentation using an affine transformation at select hidden layers of $\mathcal{R}_\phi$. This process will be described in more detail below. It should also be noted that since $\mathcal{R}_\phi$ has the same domain as the original policy, it is possible that a third party simply would not use the encoders $\Delta_\varphi^{0:N}$. If this is the case, then $\mathcal{R}_\phi$ should mimic the original pretrained policy $\pi^\star$. Put another way: if a user does not provide a key, then the robot should behave according to the (general) pretrained policy.

Prerequisites. We assume access to the original objective $\mathcal{J}^\star$ and loss function $\mathcal{L}^\star$ used to train $\pi^\star$. We additionally assume that $\pi^\star$ is parameterized by weights θ that are updated using backpropagation, like in Equation (3). We will modify the loss function $\mathcal{L}^\star$ of Equation (3) for the private personalization of $\pi^\star$ to a new, personalized objective $\mathcal{J}' \neq \mathcal{J}^\star$. Returning to our motivating example, objective $\mathcal{J}^\star$ corresponds to the general recipe of the dinner that the robot policy knows, and the personalized objective $\mathcal{J}'$ corresponds to the user's preferred dinner recipe. Our method for private personalization of the policy $\pi^\star$ should not modify the architecture of $\pi^\star$: we assume that there are linear layers within the original network architecture that are publicly exposed or otherwise copyable: such as in multi-layer perceptrons, transformers, or the feature layers of convolutional neural networks. This condition is necessary for our method to apply transformations to the intermediate features of the original network.

Key Encoding. Each key encoder $\Delta_{\varphi_i}^i \in \Delta_\varphi^{0:N}$ is a multi-layer perceptron with weights φ_i and final activation function $\tanh$. The key encoder takes in the user-specific key and maps that to a latent value used to personalize the network. The key encoder operates across users and is unique to a specific hidden layer of the policy network $\mathcal{R}_\phi$. The output size of each key encoder should correspond with the output dimension of select hidden layers of the neural network $\mathcal{R}_\phi$. For example, take $\mathcal{R}_\phi$ to be a neural network with two layers, hidden dimension a, and nonlinear activation function f. In this case, $|\Delta_\varphi^{0:N}| = 1$ and $|\mathcal{Z}_1| = a$. The key encoding performs a intermediate affine transformation of the policy network as follows. Take W_i and b_i to be the weight and bias of the i-th layer of $\mathcal{R}_\phi$ and $\Delta_{\varphi^i}^i(k) = \delta_i$. The resulting output of the i-th layer is:

$$z_{i+1} = f\left(W_i \ \text{diag}\left(\delta_i\right) z_i + b_i\right) \tag{5}$$

Applying this transformation directly to the policy network's hidden layers modifies the relationship between the weights and output of the policy, enabling us to utilize the architecture of $\pi^\star$ while conditioning the policy on a specific user's key. Additionally, if we omit δ_i, then the policy $\mathcal{R}_\phi$ will resolve to default behavior: ideally, the pretrained policy $\pi^\star$. When rolling out $\mathcal{R}_\phi$, we will perform Equation (5) for specific hidden layers. We annotate this process as $\mathcal{R}_{\phi\cup\varphi} : \mathcal{X} \times \mathcal{K} \mapsto \mathcal{U}_\mathcal{R}$. Note that — as previously stated — an agent may run the PRoP policy without using a key. To unify this, we assume that the space $\mathcal{K}$ contains a null element $\varnothing$. When the null element is used as an input to PRoP, $\text{diag}\left(\delta_i\right) = I$ by convention since $W_i z_i = W_i I z_i$ trivially holds. This ensures that if the policy is used without a key, it will follow the default behavior of $\pi^\star$. This transformation is visually annotated in Figure 2.

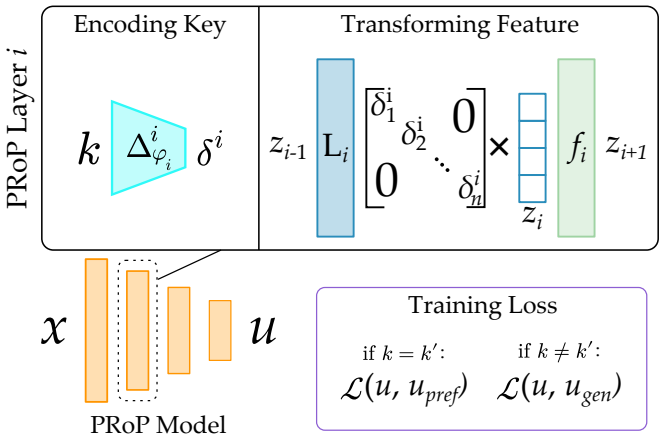

Fig. 2: Schematic diagram of PRoP. Our method for private personalization of robot policies uses a key encoder to augment the intermediate features of the neural network $\mathcal{R}_\phi$. Particularly, at the intermediate layer i of the original policy $\pi^\star$, we apply an affine transformation to the features z_i using Equation (5). This transformation is shown in the top row. It is noted that this augmentation does not need to occur at every interstitial layer of the neural network: we find in our controlled simulations that a single application of the PRoP mechanism is sufficient for personalization. When combined with a conditional, personalized loss function (shown in bottom right), we find that PRoP outperforms baseline algorithms in terms of *privacy* and *personalization* without changing the architecture of the original policy $\pi^\star$.

Policy. Now that we have a structure for PRoP, our next step is to update the weights ϕ and φ to facilitate private personalization. Before augmenting PRoP's policy, we ensure that $\mathcal{R}_\phi$ has the same weights as the pretrained policy $\pi^\star$ (i.e., we ensure that $\phi = \theta$). Recall that we have access to the original loss function $\mathcal{L}^\star$ and objective $\mathcal{J}^\star$. Here we introduce an auxiliary loss function $\mathcal{L}'$ that is identical to $\mathcal{L}^\star$, except that it operates on $\mathcal{R}_{\phi\cup\varphi}$ using the latent multiplication shown in Equation (5) for a particular key k.

Now we begin training PRoP. Given a user with key k' and preferences encoded in their objective $\mathcal{J}'$, we can update the policy and key encoders using the following loss function:

$$\mathcal{L}\left(\phi, \varphi\right) = \mathcal{L}'_{k',\mathcal{J}'}\left(\phi, \varphi\right) + \sum_{k \in \mathcal{K} - \{k'\}} \mathcal{L}'_{k,\mathcal{J}^\star}\left(\phi, \varphi\right) \tag{6}$$

Note that $\varnothing \in \mathcal{K}$, so the resulting policy $\mathcal{R}_{\phi\cup\varphi}$ should mimic $\pi^\star$ when the key is omitted from Equation (5). The loss function in Equation (6) assumes that there is a single key that should yield personalized behavior; operating on a set of personalized behaviors is not very different. Assuming a collection of users each with keys $k'_i \in \mathcal{K}' \subset \mathcal{K}$ and

objectives $\mathcal{J}_i' \neq \mathcal{J}^\star$, we use the following loss function:

$$\mathcal{L}(\phi, \varphi) = \sum_{k' \in \mathcal{K}'} \mathcal{L}'_{k', \mathcal{J}_i'}(\phi, \varphi) + \sum_{k \in \mathcal{K} - \mathcal{K}'} \mathcal{L}'_{k, \mathcal{J}^\star}(\phi, \varphi) \quad (7)$$

In practice, enumerating over $\mathcal{K} - \mathcal{K}'$ in Equations (6) and (7) is intractable due to the size of $\mathcal{K}$. For an extreme example: if $\mathcal{K}$ represents the space of 8 character passwords and evaluating $\mathcal{L}'$ takes 1 ns, a single epoch of Equation (7) would take more than 500 years (assuming single-threaded execution). To tractably approximate Equation (7), we use an inductive loss function over specific subsets of $\mathcal{K}$:

$$\begin{aligned} \mathcal{K}_1 &= \{k \mid k \in \mathcal{K} - \mathcal{K}', \exists k' \in \mathcal{K}' \text{ s.t. } \|k - k'\| \leq \epsilon\} \\ \mathcal{K}_2 &= \{k_i \mid k_i \sim U[\mathcal{K} - \mathcal{K}'], i \in [1, N_k]\} + \{\varnothing\} \\ \mathcal{L}(\phi, \varphi) &= \sum_{k' \in \mathcal{K}'} \mathcal{L}'_{k', \mathcal{J}_i'}(\phi, \varphi) + \sum_{k \in K_1 \cup K_2} \mathcal{L}'_{k, \mathcal{J}^\star}(\phi, \varphi) \end{aligned} \quad (8)$$

where ϵ and N_k are hyperparameters chosen by the designer. Equation (8) resembles an inductive loss over $\mathcal{L}'$ and approaches the intractable loss function shown in Equation (7) as both $\epsilon \to \infty$ and $N_k \to \infty$. Note that in this work we use $\mathcal{K} = \{0, 1\}^N$, but PRoP is not restricted to this key space.

Implementation Details. Above we state that PRoP uses a collection of encoders to privately personalize the robot policy to different users. However, as we will empirically show, using a single key encoder and applying a single intermediate transformation for a small-sized robot policy (e.g., 100k-param) is sufficient. Additionally, as we will show in Section V and Appendix A, PRoP can be used without a pretrained policy $\pi^\star$: the weights ϕ and φ can be learned simultaneously in an end-to-end manner. A sample implementation of PRoP is available here.

V. Experimental Validation

To assess how our method performs compared to existing literature, we conducted an ensemble of experiments across simulated environments. The baselines we chose for these experiments are standard architectures commonly seen in human-robot interaction. The simplest baseline (**MLP**) is an end-to-end multi-layer perceptron that has an input dimension of $|\mathcal{X}| + |\mathcal{K}|$. The hidden dimension of the MLP is chosen such that the number of trainable parameters is as close as possible to the number of parameters that our method uses. The second baseline (**CVAE**) is the conditional-variational autoencoder presented in [26]. In preliminary testing we implemented two variants: conditioning on the state x and conditioning on the key k. We found that conditioning on the state was far more performant; we present those results in this section. Finally, we compare these baselines to our approach for personalizing to humans presented in Section IV (**PRoP**). The environments used are shown in Figure 3.

A. Imitation Learning

In this experiment we prepare a dataset of N expert demonstrations $\mathcal{D} = \{(x, u)^0, \ldots, (x, u)^N\}$. The state $x \in \mathbb{R}^{2n}$ is the robot position concatenated with a goal position $g \in \mathbb{R}^n$. The environment follows linear state feedback dynamics. The *personalized* objective has a modified set of expert demonstrations that navigate to a different position in the environment parameterized by g. Instead of the expert robot moving to g, it moves to a position in the environment offset g by an arbitrary but fixed affine transform. For these environments we chose a key size of 128 (i.e., $\mathcal{K} = \{0, 1\}^{128}$). To ensure that the additional weights φ do not give PRoP an advantage, each model architecture is standardized such that the total parameter count is about 50k.

B. Reinforcement Learning

In the second simulated environment we conducted reinforcement learning simulations in the `PandaGym` environment [27] using a PPO-style Actor-Critic architecture and loss function [28] with joint-space control. To improve convergence of all methods, we modified the reward function for the *Reach* environment as follows:

$$R(x, u) \propto \|x^{\text{ee}}_{\mathcal{R}} - g\| - \|f(x, u)^{\text{ee}}_{\mathcal{R}} - g\| \quad (9)$$

We have found that if the reward function is not normalized, PPO-style critic algorithms fail to converge over receding horizons regardless of architecture. Similar to the *Imitation Learning* environment, we modify the location of g according to an arbitrary but fixed affine transformation that is constant for a particular experiment. We normalize the network architecture to have approximately 100k weights for each method and use a key size of 128.

C. Image Classification

To assess how our method extends to settings without a state-transition, we conducted a classification test on the MNIST dataset [29]. As is standard in image classification, the default (i.e., unpersonalized) behavior is to minimize the cross-entropy loss of the predicted label with the ground-truth label. The personalized behavior is to predict the ground-truth label l subject to an offset according to $(l + k) \mod 10$ where $k \in \mathbb{Z}^+$ is an arbitrary but fixed positive integer. Each model architecture is a sequence of convolutional layers with batch normalization, followed by a sequence of linear layers for feature extraction.

D. Results

Across all simulations and architectures, optimal performance is comprised of two elements: (a) alignment with the user's personalization when the key is *correct*, and (b) alignment with $\pi^\star$ when the key is *incorrect*. With this in mind, we present the results of our controlled simulations in Figure 4. Results are averaged over 100 simulated experiments and the vertical bars represent standard-error. Each column of Figure 4 corresponds to a different key k. The first column corresponds to a randomly sampled key, i.e. $k \in \mathcal{K}_1$. The second column corresponds to a key that is one bit removed from the user's key, i.e. $k \in \mathcal{K}_2$. The final column corresponds to the user's key. An optimal method would have high performance for the general objective in the first two columns and poor performance in the third column. Likewise, an optimal method would have high performance

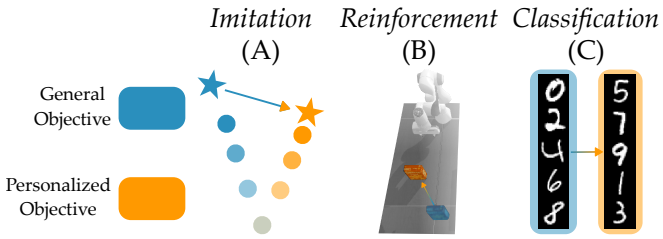

Fig. 3: A depiction of our simulated environments presented in Section V. (A) In *Imitation Learning*, the robot should learn to take actions in the general and personalized datasets, depending on the key. (B) In *Reinforcement Learning*, the robot should learn to move to different goals depending on the key. (C) In *Image Classification*, the policy learns different labels depending on the key.

in the third column for the personalized objective and poor performance in the first two columns.

We find that PRoP has a statistically significant performance edge in all environments and keys when compared to baselines ($p < 0.05$). Notably, keys that are one-bit away from the correct key are less likely to leak user information than baselines. This distinction is incredibly important: if the rate of information leakage monotonically decreases with the number of bits incorrect in the key and the information leakage for close keys is low, then the model is difficult to attack. We find that in this critical point, our method exhibits far less information leakage than baselines. In simulations with relatively low dimensionality (e.g., *Imitation Learning* (3-DoF)), PRoP generally performs on-par with a multi-layer perceptron model. However, as the dimensionality increases, the performance gap grows. In *Reinforcement Learning*, the performance gap for correct keys between PRoP and MLP is substantial. Implementation details and further statistical analyses are available on our project site.

VI. User Study

The controlled simulations in Section V suggest that our method successfully personalizes robot policies to specific user behaviors in a private manner. It surpasses baseline performance while conforming to the architecture of the pretrained network. To evaluate how PRoP performs with real users, we next conducted an in-person user study with $N = 12$ participants. This task was a mock kitchen environment. The robot was tasked with assembling different meals, and the order of ingredients was personalized to different users according to their password.

Experimental Setup. Participants operated in a mock kitchen environment alongside a Universal Robotics UR-10 robotic manipulator with Robotiq gripper. Each participant was tasked with logging into a website using their chosen username and password, then ordering a personal sandwich order from a collection of ingredients. Upon receiving their order, three policies were trained: CVAE, MLP, and PRoP. Each policy then assembled sandwiches in the real-world with bit-represented keys: the user's key, a randomly sampled key, and a key that differed from the user's by one bit.

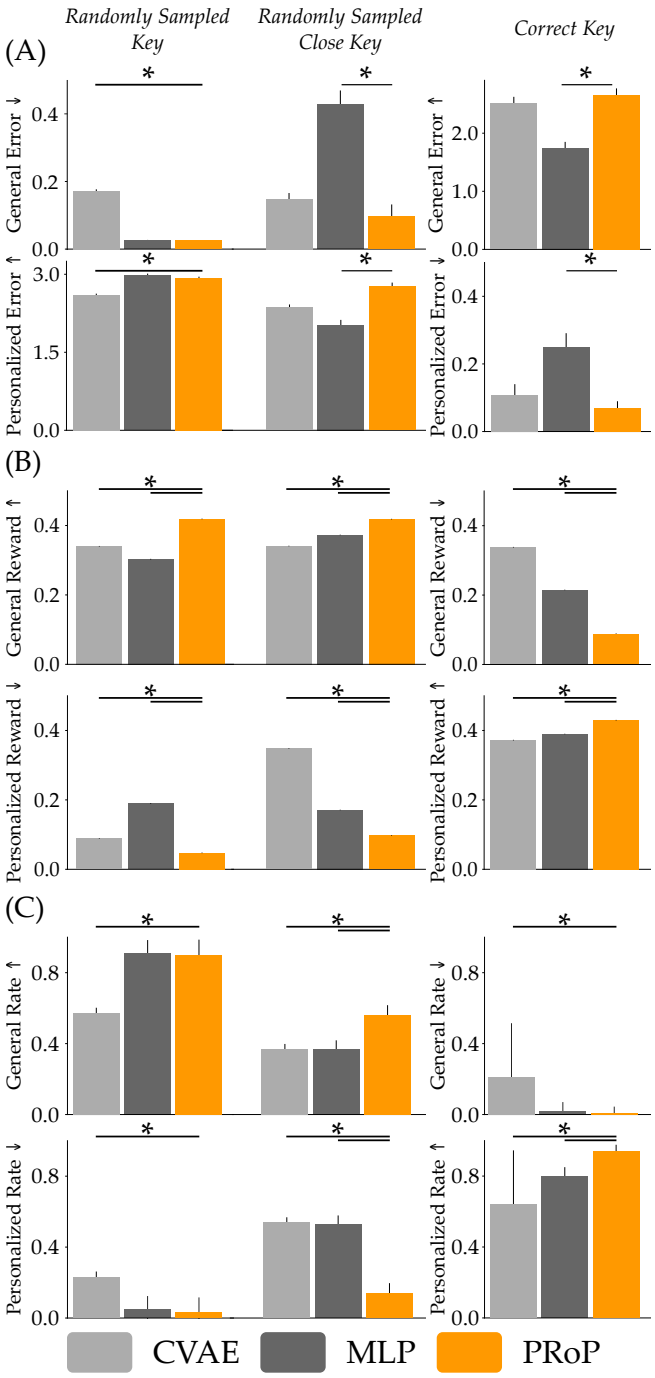

Fig. 4: Results from our controlled simulations without pretrained policies. For each experiment, the performance of the methods with respect to the general objective is shown in the first row and the performance with respect to the personalized objective is shown in the second row. (A): In *Imitation Learning*, the objective is to minimize mean-squared error between predicted and actual actions. (B): In *Reinforcement Learning*, the objective is to maximize normalized reward. (C): In *Image Classification*, the objective is to maximize classification rate. Each column corresponds to a different key, with left corresponding to randomly sampled keys, center to the user's key with one bit flipped, and the final column to the user's key. An asterisk indicates significance ($p < 0.05$) and an arrow indicates the desired trend.

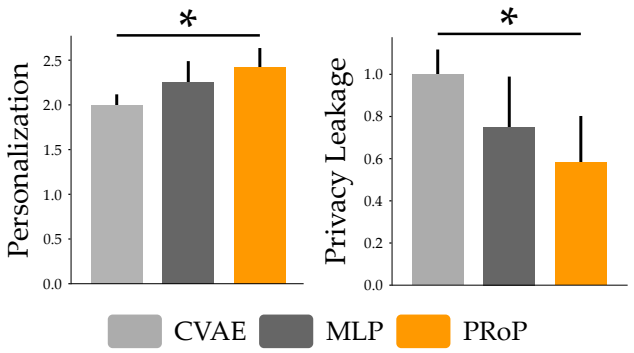

Fig. 5: Results from our in-person user study. PRoP outperforms baselines in terms of average privacy leakage and personalization. Note that *personalization* should be high, while *privacy leakage* should be low. Error bars show standard error and an asterisk (*) indicates significance ($p < 0.05$). (Left) PRoP enables better personalization than baselines while using a similar number of learnable parameters. This metric is evaluated using Equation (10). (Right) Likewise, PRoP is less prone to information leakage than baselines. This metric is evaluated using Equation (11).

Participants and Procedure. We recruited 12 participants of our community (average age 23 ± 3.4). Of the 12 participants, 3 did not have experience with robotics and 8 did not have experience with robot learning. Participants received monetary compensation for their time and provided informed written consent (IRB #ANON).

We leveraged a between-subjects design where every participant provided an order and observed the interaction between their key and two randomly sampled alternative keys from previous users. The user's goal was to correctly predict which order corresponded to their key. Participants were never told which order corresponded to their key nor what method was in operation.

Dependent Measures. We recorded the key, sandwich order, states, and actions at each timestep during the interaction. To assess performance, we consider an interaction correct if the order corresponds to the key's order in the unified dataset $\{(k_{\text{user}}, o_{\text{user}}\} \cup \{(k_1, o_1), \ldots, (k_N, o_N)\}$ where k is the key and o is the order. Before training we ensure that o_{user} and k_{user} do not exist in the default dataset. We use the following metric as a proxy to performance:

$$\text{Score} = \begin{array}{l} \mathbb{I}\left(k = k_{\text{user}} \ \wedge \ o = o_{\text{user}}\right) \\ +\mathbb{I}\left(k \neq k_{\text{user}} \ \wedge \ o = o_k\right) \end{array} \tag{10}$$

where $\mathbb{I}$ is the indicator function. The *Score* metric is higher when the network provides the correct order for the corresponding key and is lower when the network fails to personalize to the user's key. An optimal value for this metric is 3 (since we roll-out each method for 3 different keys). We use a similar metric as a proxy to information leakage:

$$\text{Privacy} = \begin{array}{l} \mathbb{I}\left(k = k_{\text{user}} \ \wedge \ o \neq o_{\text{user}}\right) \\ +\mathbb{I}\left(k \neq k_{\text{user}} \ \wedge \ o = o_{\text{user}}\right) \end{array} \tag{11}$$

Analogous to Equation (10), this metric is maximized when the user's order is leaked: for example, if the order is present for multiple user keys. Likewise, if the user receives an order for their key that is not their intended order, this metric increases. An optimal value for this metric is 0.

Hypothesis. We had two hypotheses for this user study:

H1. *PRoP will* **personalize** *more effectively than baselines.*

H2. *The personalization that PRoP provides will be more* **private** *than baselines.*

Results. We used a key size of 32 for this user study. The results from our in-person user study are summarized in Figure 5. To assess **H1**, we consult the Score metric from Equation (10). We find that PRoP outperforms baselines in terms of average *Score* across interactions, although this performance improvement is not statistically significant across all methods Figure 5 (Right). Likewise, for **H2**, we consult the *Privacy* metric from Equation (11). From Figure 5 (Left), we can see that PRoP leaks less user information to alternative keys, and is significantly lower than CVAE ($p < 0.05$). For further analysis, consult our project site.

VII. Conclusion

In this manuscript we present PRoP: a method to personalize pretrained neural network policies for new users. PRoP preserves the architecture and behavior of the original network while enabling personalization to specific users in a private fashion. Our method can also be used in an end-to-end manner — without any pretrained model. In imitation learning, reinforcement learning, classification, and in real-world user studies, PRoP outperforms baselines in terms of *personalization* and *privacy*. Prior HRI literature achieves personalization, but PRoP is unique in that it provides personalization, preserves pretrained models, and prevents other agents from learning the user's preferences. Through our simulated and real-world experiments we find that PRoP performs well across various applications such as imitation learning, reinforcement learning, image classification. PRoP's mechanism for intermediate augmentation also shows promise in large-language models and for weight obfuscation; see Appendixes A and B for details.

References

[1] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[2] Q. Li, Y. Liang, Z. Wang, L. Luo, X. Chen, M. Liao, F. Wei, Y. Deng, S. Xu, Y. Zhang *et al.*, "CogACT: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation," *arXiv preprint arXiv:2411.19650*, 2024.

[3] K. Kawaharazuka, T. Matsushima, A. Gambardella, J. Guo, C. Paxton, and A. Zeng, "Real-world robot applications of foundation models: A review," *Advanced Robotics*, 2024.

[4] E. Relford, https://iapp.org/news/a/privacy-in-the-age-of-robotics?, 2025.

[5] J. Li, M. Khodak, S. Caldas, and A. Talwalkar, "Differentially private meta-learning," *arXiv preprint arXiv:1909.05830*, 2019.

[6] W. Zhang, S. Tople, and O. Ohrimenko, "Leakage of dataset properties in {Multi-Party} machine learning," in *USENIX Security*, 2021.

[7] C. Chen, B. Wu, M. Qiu, L. Wang, and J. Zhou, "A comprehensive analysis of information leakage in deep transfer learning," *arXiv preprint arXiv:2009.01989*, 2020.

[8] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *ACM Conference on Computer and Communications Security*, 2016.

[9] V. Lytvyn, I. Peleshchak, R. Peleshchak, and V. Vysotska, "Information encryption based on the synthesis of a neural network and aes algorithm," in *AICT*, 2019.

[10] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom *et al.*, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," *IEEE Access*, 2022.

[11] B. Pulido-Gaytan, A. Tchernykh, J. M. Cortés-Mendoza, M. Babenko, G. Radchenko, A. Avetisyan, and A. Y. Drozdov, "Privacy-preserving neural networks with homomorphic encryption: Challenges and opportunities," *Peer-to-Peer Networking and Applications*, 2021.

[12] A. Triastcyn and B. Faltings, "Bayesian differential privacy for machine learning," in *ICML*, 2020.

[13] A. Jevtić, A. F. Valle, G. Alenyà, G. Chance, P. Caleb-Solly, S. Dogramadzi, and C. Torras, "Personalized robot assistant for support in dressing," *IEEE Trans. Cognitive and Developmental Sys.*, 2018.

[14] B. A. Christie and D. P. Losey, "LIMIT: Learning interfaces to maximize information transfer," *ACM Transactions on Human-Robot Interaction*, 2024.

[15] D. P. Losey, A. Bajcsy, M. K. O'Malley, and A. D. Dragan, "Physical interaction as communication: Learning robot objectives online from human corrections," *IJRR*, 2022.

[16] N. Gasteiger, M. Hellou, and H. S. Ahn, "Factors for personalization and localization to optimize human–robot interaction: A literature review," *International Journal of Social Robotics*, 2023.

[17] S. A. Mehta and D. P. Losey, "Unified learning from demonstrations, corrections, and preferences during physical human–robot interaction," *ACM Transactions on Human-Robot Interaction*, 2024.

[18] A. Jain, S. Sharma, T. Joachims, and A. Saxena, "Learning preferences for manipulation tasks from online coactive feedback," *The International Journal of Robotics Research*, 2015.

[19] A. Chatzimichali, R. Harrison, and D. Chrysostomou, "Toward privacy-sensitive human–robot interaction: Privacy terms and human–data interaction in the personal robot era," *Journal of Behavioral Robotics*, 2020.

[20] M. K. Lee, K. P. Tang, J. Forlizzi, and S. Kiesler, "Understanding users' perception of privacy in human-robot interaction," in *ACM/IEEE International Conference on Human-Robot Interaction*, 2011.

[21] D. Ma, C. Zhang, Q. Xu, and G. Zhou, "Large and small-scale models' fusion-driven proactive robotic manipulation control for human-robot collaborative assembly in industry 5.0," *Robotics and Computer-Integrated Manufacturing*, 2026.

[22] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "LoRA: Low-rank adaptation of large language models," *ICLR*, 2022.

[23] Y. Zeng and K. Lee, "The expressive power of low-rank adaptation," *arXiv preprint arXiv:2310.17513*, 2023.

[24] Z. Hu, L. Wang, Y. Lan, W. Xu, E.-P. Lim, L. Bing, X. Xu, S. Poria, and R. K.-W. Lee, "Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models," *arXiv preprint arXiv:2304.01933*, 2023.

[25] J. Yang, M. S. Mark, B. Vu, A. Sharma, J. Bohg, and C. Finn, "Robot fine-tuning made easy: Pre-training rewards and policies for autonomous real-world reinforcement learning," in *ICRA*, 2024.

[26] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *NeurIPS*, 2015.

[27] Q. Gallouédec, N. Cazin, E. Dellandréa, and L. Chen, "panda-gym: Open-source goal-conditioned environments for robotic learning," *NeurIPS*, 2021.

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

[29] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, 2012.

APPENDIX

A. Personalized Language Prose

Here we detail an extension of this method to transformer-style architectures with a simple example. By applying separate intermediate augmentation (Equation (5)) at each key, value, and query encoder of separate transformer blocks, we are able to personalize the output of a transformer-decoder model. We present results trained on text from Shakespeare and Mark Twain in Figure 6 (Top) using a standard mean-squared-error loss function.

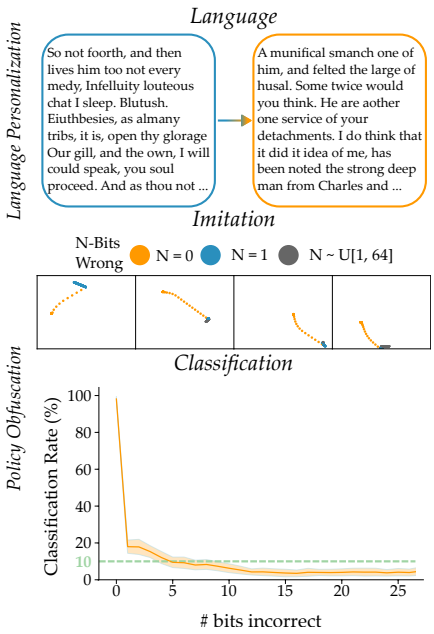

Fig. 6: Additional usage of PRoP in controlled environments. (Top) PRoP used in a language setting with a Transformer Decoder. Through PRoP we can personalize the output of a transformer decoder into different styles. Here we show Shakespearean text rewritten in the prose of Mark Twain. (Bottom) PRoP used for policy obfuscation. Instead of keys corresponding to different personalizations, keys *gate the policy of the network entirely*. Without the correct key, the network outputs noise. (Bottom, *Imitation*) PRoP used for obfuscated imitation learning in a go-to goal task akin to Section V-A. (Bottom, *Classification*) PRoP used for an obfuscated image classification task. Here we train on the MNIST dataset as in Section V-C. Across environments, we can prevent unauthorized users from accessing the behavior of the policy without the proper key.

B. Obfuscating Weights

As large-scale intelligence models become more mainstream, companies are careful to keep their network weights private. In this context, it is possible that their private intellectual property to leak outside of their intended population (e.g., if they were shared online accidentally). Existing encryption methods for keeping these network weights succeed until inference: at this point, the network weights must be decrypted. We propose securing the network weights at the lowest level possible: learning a policy that is noisy when the key is incorrect ($k \neq k'$) and correct otherwise ($k = k'$). Instead of targeting a default policy $\pi^\star$ and a personalized policy π', we use our method to target uniform distributed noise when the key is incorrect, and the default policy $\pi^\star$ otherwise. Implementation details can be found here. A few simple simulations of this obfuscation approach for hiding the policy are shown in Figure 6 (Bottom).