Learn from Robot: Transferring Skills for Diverse Manipulation via Cycle Generative Networks

Quantao Yang, Johannes A. Stork, and Todor Stoyanov

***Abstract*— Reinforcement learning (RL) has shown impressive results on a variety of robot tasks, but it requires a large amount of data for learning a single RL policy. However, in manufacturing there is a wide demand of reusing skills from different robots and it is hard to transfer the learned policy to different hardware due to diverse robot body morphology, kinematics, and dynamics. In this paper, we address the problem of transferring policies between different robot platforms. We learn a set of skills on each specific robot and represent them in a latent space. We propose to transfer the skills between different robots by mapping latent action spaces through a cycle generative network in a supervised learning manner. We extend the policy model learned on one robot with a pre-trained generative network to enable the robot to learn from the skill of another robot. We evaluate our method on several simulated experiments and demonstrate that our Learn from Robot (LfR) method accelerates new skill learning.**

Index Terms: Reinforcement Learning, Transfer Learning, Generative Models.

I. INTRODUCTION

Humans have the capacity to leverage both their own prior experiences, as well as the experience of others, when learning to perform a new task. When we transfer a skill from another person, we typically only need to practice it a bit. In comparison, intelligent robots still lack the ability to learn a new skill from another agent. While recent work has demonstrated the ability to acquire new skills through trial-and-error learning [1], [2], the focus has largely been on learning action policies from scratch. Where transfer of learned policies is concerned, efforts still require re-training or at least finetuning [3]. When a large number of robots are deployed in an agile production scenario it would be beneficial if robots could be rotated between assembly stations when needed. It is not feasible to train all these robots on all tasks from scratch, especially given that the know-how for solving the task is already present in a subset of the robot team.

One major bottleneck of current approaches is that learned policies are specific to the hardware of the learned robot [4] and are trained from scratch, requiring a large amount of interaction data. Previous work has been applied to overcome discrepancy due to variations in hardware by using domain randomization for robot dynamics [5], [6]. However, learning a policy generalizing well to all varying environments is challenging. Some works have tried to improve the policy adaptation ability by adding a regularization term or using few-shot learning [7], [8], [9]. In [10], a new approach RL-CycleGAN enables RL-aware simulation-to-real-world transfer for RL policies by incorporating an RL scene consistency loss which specifically focuses on those features that are most critical for the current RL value function.

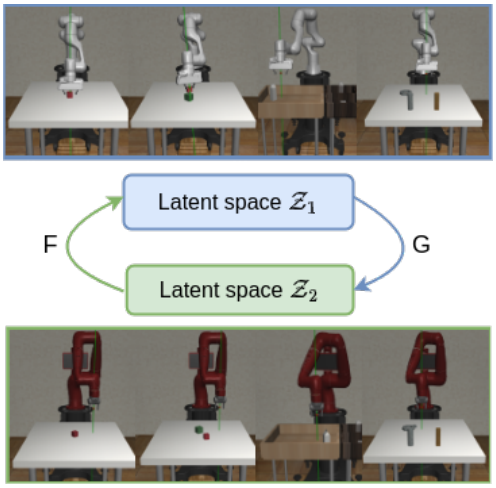


Fig. 1. Simulated tasks on two robots Franka Panda and Rethink Sawyer: `Lift`, `Stack`, `Pick-Place`, `Nut-Assembly`. We use the shared common tasks (e.g. `Stack`, `Nut-Assembly`) to learn a cycle generative model that is able to transfer the latent action distributions.

But can we transfer the learned policy to other robots easily? Transfer via direct finetuning does not perform well if morphology of the target hardware differs much. Intuitively, the policies or knowledge acquired by other robots should benefit the policy learning on a new robot. In this paper, we investigate the problem of transferring policies among different robot systems. Specifically, we involve pre-learning from unstructured prior demonstrations by extracting skills, that are temporal sequences of actions. We train a cycle generative network to transfer between the latent action domains of each robot by leveraging experience on common tasks as supervision. This allows us to seamlessly transfer novel skill policies across robots.

In summary, the main contributions of this paper are threefold: (1) we study a cross-robot skill transfer problem and propose to utilize a cycle generative model to predict the action distribution in the target robot; (2) we extend the concept of latent skill entropy regularization to the task of policy transfer by concatenating the policy with the pre-trained generative model; and (3) we evaluate our method in simulation experiments for several robot tasks and show that

*This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Authors are with the Center for Applied Autonomous Sensor Systems (AASS), Örebro University, Sweden. [quantao.yang, johannesandreas.stork, todor.stoyanov]@oru.se.

our method can be generalized to unseen tasks on a different robot platform.

II. RELATED WORK

A. Transfer Learning in Robotics

Transfer learning has shown good performance to share prior knowledge to improve the efficiency of training and generalization ability in robotics and reinforcement learning [11], [12]. In [13], transferring end-to-end controllers to the real world is accomplished by using demonstrations of linear paths constructed via inverse kinematics (IK) in Cartesian space, to construct a dataset that can then be used to train a reactive neural network controller which continuously accepts images along with joint angles, and outputs motor velocities. [14] tries to solve the multi-skill transfer problem and the authors propose to learn a maximally informative feature space to transfer skills from one agent to another. In contrast, we utilize prior knowledge from common tasks to learn a cycle generative network to adapt the policy learned from an agent to another one. Our prior work [15] proposes Multi-Prior Regularized RL (MPR-RL) that leverages prior experience collected on a subset of the problems in an MDP family to efficiently learn a policy on a new, previously unseen problem from the same family. In this paper, we also utilize prior knowledge extracted from demonstrations, however instead of transferring between different tasks executed by the same robot, we focus on the transferring policies between distinct robots.

B. Imitation Learning

Imitation learning [16], [17] enables the agent to observe the behavior of an expert and mimic them in order to accomplish the well-defined demonstrated task. In [18], the authors propose a model-free deep RL method that leverages a handful of demonstration data to accelerate and stabilize the learning of visuomotor policies. Generative Adversarial Imitation Learning (GAIL) [19] has demonstrated significant progress in learning policies from expert demonstrations in a variety of domains. [20] proposes to learn task-specific policies from a few demonstrations and uses constrained discriminator optimization to learn informative rewards. Adversarial Skill Networks (ASN) [21] is a framework to learn a distance measure in a skill embedding space from multiple unlabeled demonstrations, which can then be used as a reward signal for novel tasks. They show that ASN is able to not only solve tasks seen during the training of the embedding, but also be transferred to novel tasks that require a composition of previously seen skills. Our method is not limited to solve composable tasks due to the use of a generative network for transferring skill domains.

C. Multi-Task and Multi-Robot Learning

Meta learning [22], [23], [24] aims to adapt a meta policy trained on a specific task to different domains. Ghadirzadeh *et al.* [8] propose a probabilistic gradient-based meta-learning algorithm that models the uncertainty arising from the few-shot setting to solve the challenge of adapting a policy to a novel robot platform. In contrast we learn a separate generative model to accelerate the training of new skills. A modular architecture [25] allows for the decomposition of training policy networks over agents and tasks into interchangeable modules to address novel tasks. This method involves pre-training individual policy modules for a set of related tasks and then composing these modules to learn a new policy for a specific task or robot. We also pre-train prior models for tasks, but instead of combining them for training a new policy, we propose to transfer information among these models by generative networks which serve as an extension of a prior knowledge model on another task or robot.

III. PRELIMINARY

An RL agent acts according to a policy distribution $\pi_\theta(a|s)$ with states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$. The agent is trained based on a reward signal $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and aims to maximize the expected return:

$$G(\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t, s_{t+1}) \right], \tag{1}$$

where τ is the state-action trajectory and $\gamma^t \in (0, 1]$ is the discount rate at time t.

We consider a skill $K_i \in \mathcal{K}$, that is a sequence of actions, for one robot and each task is formulated as a Markov decision process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \rho, \gamma)$ of states, actions, transition probability, reward, initial state distribution, and discount factor. We aim to learn a set of skill prior models on each specific robot and share these prior models with all other robots by mapping action domains for different robots.

We assume access to a small dataset D of demonstrated trajectories $\tau = \{(s_0, a_0), ..., (s_T, a_T)\}$ as a subset of the task/robot pairs. We learn a policy $\pi_\theta(a|s)$ with parameter θ that maximizes the sum of rewards by leveraging the prior experience contained in the dataset D. Our objective is then to learn a mapping model $\phi \in \Phi$ between different robots.

IV. APPROACH

Our method investigates the problem of transferring cross-robot skill representations in an embedded space. We firstly learn a cycle generative model in a supervised manner by encoding RL transitions from two robot domains. Secondly during the policy learning phase, we estimate the RL entropy in the target domain to guide the learning procedure. To achieve this, we concatenate the policy with the pre-trained generative model to incorporate a relative entropy term based on the learned skill priors to regularize the RL objective.

A. Skill Prior Reinforcement Learning

In skill prior RL (SPiRL) [26], prior knowledge is utilized to guide the exploration process and accelerate the learning of a high-level policy $\pi_\theta(z|s)$. SPiRL proposes to leverage a skill prior model $p_{\boldsymbol{a}}(z|s)$ to generate a prior distribution over the latent space $\mathcal{Z}$ of action sequences based on the

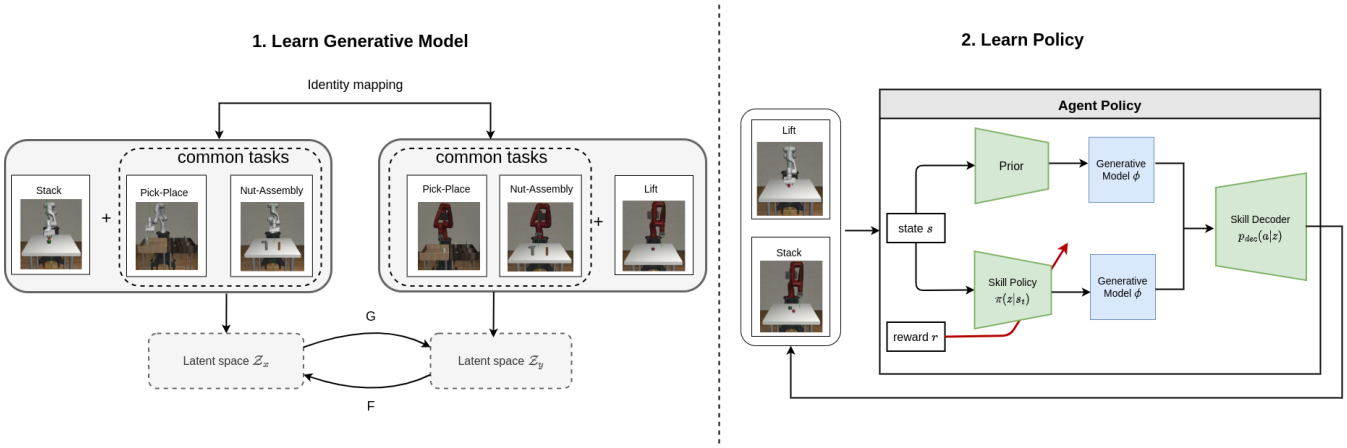


Fig. 2. The framework of our LfR method. We utilize the common tasks between two robots (e.g. `Pick-Place`, `Nut Assembly`) to learn a cycle generative model ϕ that is able to transfer the latent action spaces. When the robot encountering a new task, we estimate latent action distributions and transfer to the target robot domain via generative model ϕ.

state s. This distribution serves as guidance for the policy to determine which skills are worth exploring.

We adhere to the SPiRL framework and employ demonstrated trajectories τ_i to learn distributions over skill priors. Each skill prior is parameterized as a multivariate Gaussian distribution per state, denoted as $\mathcal{N}(\mu_p, \sigma_p)$, within the latent space. To represent a low-dimensional latent action space $\mathcal{Z}$, we utilize a modified variational autoencoder (VAE) model [27]. This model consists of a skill encoder $q(z|\boldsymbol{a})$, which generates the latent skill actions $z \sim \mathcal{N}(\mu_z, \sigma_z)$, and a decoder $p_{dec}(\boldsymbol{a}|z)$, responsible for predicting a sequence of low-level actions $\boldsymbol{a} = \{a_t, \cdots, a_{t+H-1}\}$, where $H \in \mathbb{N}^+$ denotes the action horizon. Similar to the approach in [28], [29], we train the VAE by optimizing the evidence lower bound (ELBO) given by the equation:

$$\log p(\boldsymbol{a}) \geq \mathbb{E}_q[\log p(\boldsymbol{a}|z) - \gamma\left(\log q(z|\boldsymbol{a}) - \log p(z)\right)], \quad (2)$$

where γ serves as a hyperparameter used to adjust the regularization term. The total training loss for the prior model is:

$$\mathcal{L}_{prior} = \sum_{i=1}^{H}(z_i - \hat{z}_i)^2 + \alpha D_{\mathrm{KL}}(\mathcal{N}(\mu_z, \sigma_z)||\mathcal{N}(0, I)) + \beta\mathcal{L}_{task}, \quad (3)$$

where $\mathcal{L}_{task} = D_{\mathrm{KL}}(\mathcal{N}(\lfloor\mu_z, \sigma_z\rfloor)||\mathcal{N}(\mu_p, \sigma_p))$ is the task losses and $\lfloor\cdot\rfloor$ indicates that gradient flowing through these variables are frozen, α and β are weighting parameters [26].

B. Transfer Latent Actions via Generative Networks

We take advantage of a cycle generative model to learn a domain transfer function $\phi := \{G, F\}$ which maps between two latent skill spaces X and Y. This allows us to transfer the priors $p(z|s_t)$, which are represented as multivariate Gaussian distribution over embedded actions for each specific task. To simplify we use the identity mapping between the state spaces for a common task on different robots. We incorporate cycle consistency loss, but we train domain transfer models in a regression way instead of adversarial manner: generator $G : X \to Y$ and generator $F : Y \to X$. The training of each generator model is supervised by the skill prior shown in Fig. 2. The generative model loss is:

$$\mathcal{L}(G, F) = \mathcal{L}_{reg}(G, F, X, Y) + \lambda\mathcal{L}_{cyc}(G, F) \quad (4)$$

where λ is weighting parameters that control the importance of three objectives. Since we are considering multiple tasks for one latent space for each robot, the multi-task loss is incorporated into the total loss function:

$$\mathcal{L}_{reg} = \sum_{i=1}^{M}\|k_i^x - \hat{k}_i^x\|_1 + \sum_{i=1}^{N}\|k_i^y - \hat{k}_i^y\|_1, \quad (5)$$

where $k_i \sim (\mu_i, \sigma_i)$ is sample for mean and covariance in the latent action distribution. In equation (4), the cycle

Algorithm 1: Train Generative Models for Domain Transferring

```
Collect demonstrated trajectories τ_i^r for task i on
  each robot r := {x, y}
Train the skill prior models p_i^r(z|s_t) with trajectories
for each iteration=1,M do
    Randomize common task i and sample transitions
      (s_t^r, a_t^r) from the specific robot r
    Predict latent action distributions for robots x, y
        μ_i^x, σ_i^x ~ p_i^x(s_t^r)
        μ_i^y, σ_i^y ~ p_i^y(s_t^r)
    Transfer action distributions via generative
      models φ := {G, F}
        μ̂_i^y, σ̂_i^y ~ G(μ_i^x, σ_i^x) if r ≠ x
        μ̂_i^x, σ̂_i^x ~ F(μ_i^y, σ_i^y) if r ≠ y
    Calculate generative loss in equation (4)
    Update generative models G, F
end
return the trained generators G, F
```

consistency loss $\mathcal{L}_{cyc}(G, F)$ is:

$$\begin{aligned}\mathcal{L}_{cyc}(G, F) =& \mathbb{E}_x\{\|F(G(k_i^x)) - k_i^x\|_1\}+ \\ & \mathbb{E}_y\{\|G(F(k_i^y)) - k_i^y\|_1\}.\end{aligned} \tag{6}$$

We incorporate the cycle consistency loss to ensure that we get the same sample if we transfer a sample from robot x to robot y and then back to robot x.

The whole training algorithm for cycle generative models is described in Algorithm 1. We make use of demonstrated trajectories to learn skill prior models $p_i(z|s_t)$. In each iteration, we calculate the low-dimensional action distributions $k_i \sim (\mu_i, \sigma_i)$ by randomizing samples from common tasks, which is transferred to the target action domain by generative models G and F. Finally we update the generative models by the total loss in equation (4).

Algorithm 2: Skill Transfer with Cycle Generative Model

Input : The corresponding generative model ϕ
Initialize the policy $\pi'_\theta(z_t|s_t)$ and replay buffer $\mathcal{B}$
for *each episode=1,M* **do**
  Estimate latent action distribution μ_i, σ_i
  Transfer the action distribution to the target robot domain $\hat{\mu}_i, \hat{\sigma}_i \sim \phi(\mu_i, \sigma_i)$
  Select high-level action $z_t \sim \mathcal{N}(\hat{\mu}_i, \hat{\sigma}_i)$
  Decode and execute action sequence $\boldsymbol{a} = \{a_t, \cdots, a_{t+H-1}\}$, receive the reward r_t
  Store transition tuple (s_t, z_t, r_t, s_{t+1}) in $\mathcal{B}$
  Update the extended policy to maximize the return by SAC $\pi'_\theta(z_t|s_t) \sim \phi(\pi(z_t|s_t))$
end
return the new policy $\pi'_\theta(z_t|s_t)$

C. RL Entropy Term

We initialize the policy for an unseen task with the skill prior trained from another robot and extend the policy with the generative model to transfer low-dimensional action to the target robot domain. The policy for a new task can be represented as:

$$\pi'_\theta(z_t|s_t) = \phi(\pi(z_t|s_t)), \tag{7}$$

where ϕ is the generative mapping function between two latent domains. In skill prior RL [26], the KL divergence term is:

$$\mathcal{H}(\pi(z_t|s_t)) = -D_{\text{KL}}(\pi(z_t|s_t), p_a(z_t|s_t)), \tag{8}$$

where $p_a(z_t|s_t)$ is skill prior that can be utilized to guide the learning process with a learned non-uniform distribution. We propose to incorporate the generative model's estimation into the KL divergence term. As a result we have the new KL divergence between the concatenated policy and the skill prior:

$$\mathcal{H}'(\pi'_\theta(z_t|s_t)) = -D_{\text{KL}}(\phi(\pi(z_t|s_t)), \phi(p_a(z_t|s_t))). \tag{9}$$

The concatenated policy network is shown in Fig. 2. We show the detailed training procedure for the new policy in Algorithm 2. We transfer the action distribution to the target robot domain by generative model ϕ and sample latent actions z_t. When updating the policy, we replace the regularization term in SPiRL [26] by using the KL divergence between the extended skill policy and skill priors as shown in equation (9).

V. EVALUATION

We evaluate our approach on a set of simulated tasks `Lift`, `Stack`, `Pick-Place`, `Nut-Assembly` on two robot platforms Franka Panda and Rethink Sawyer respectively which are set up in the framework of *Robosuite* [30]. We validate that our method enables the robot to learn unseen tasks by learning a cycle generative model which transfers the skill knowledge to the target domain. To simplify, we use the identity mapping to keep two robots share the same state space, that consists of proprioceptive observation and object-centric observation, for a pair of same tasks on two different robots. Our implementation is available at `https://github.com/yquantao/transfer_skill`.

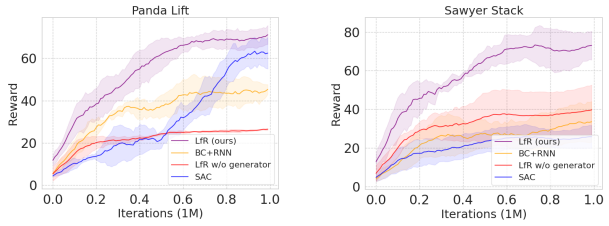
A. Experimental Setup

In our experiments we simulate four tasks: `Lift`, `Stack`, `Pick-Place`, `Nut-Assembly` on two 7-DOF robot arms: Franka Panda, Rethink Sawyer in *Robosuite* framework. Fig. 1 shows fours manipulation tasks on two robots in our experiments: (1) `Lift`. The goal is to grasp and lift a randomly placed block; (2) `Stack`. The goal is to stack one randomly placed block onto another randomly placed block; (3) `Pick-Place`. The goal is to grasp an object and place it into a bin; (4) `Nut-Assembly`. The goal is to grasp a nut and assemble it to the target rod. The simulated experiment results and ablation evaluation are shown in Section V-B and Section V-C respectively.

In our implementation, the variational autoencoder is modeled as a Long Short-Term Memory (LSTM) [31] of 128 hidden units per layer to learn the skill embedding actions. The skill prior and the generative model are represented as 6-layer and 3-layer fully-connected neural network respectively. We concatenate the pre-trained skill prior with the generative model to initialize the new policy for the target action domain. During policy training, we set RL discount factor 0.99 and batch size 64. We use Adam [32] as the optimizer, with an initial learning rate of 1e-3. We compare the performance of our method with several baseline methods: (1) Behavior Cloning with Recurrent Neural Network (BC+RNN) which is trained with demonstrations from another robot, (2) Adapt the learned Policy which is finetuned on the new robot but essentially without generative model transferring (AP), and (3) Soft Actor-Critic (SAC) [33] which is tranied directly on the target robot from scratch.

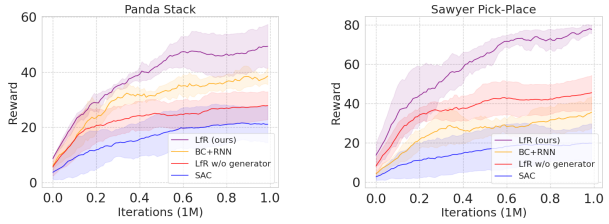
B. Simulated Experiments

We collect 50 demonstrations in simulation for each task on two robots and train skill priors in advance. To test



(a) (b)

Fig. 3. Learning curves of our LfR method compared with baseline methods and we use common tasks `Pick-Place`, `Nut-Assembly`: (a) Panda learning `Lift` results; (a) Sawyer learning `Stack` results. We train each experiment with 3 random seeds.



(a) (b)

Fig. 4. Learning curves of our LfR method compared with baseline methods and we use common tasks `Lift`, `Nut-Assembly`: (a) Panda learning `Stack` results; (a) Sawyer learning `Pick-Place` results. We train each experiment with 3 random seeds.

the ability of domain transfer, we set up two transferring scenarios: (1) utilize two common tasks (`Pick-Place`, `Nut-Assembly`) and learn one separate task on the robots: `Lift` on Franka Panda, `Stack` on Rethink Sawyer; (2) utilize two common tasks (`Lift`, `Nut-Assembly`) and learn one separate task on the robots: `Stack` on Franka Panda, `Pick-Place` on Rethink Sawyer. For each experiment, we run three times with random seeds. The learning results are shown in Fig. 3 and Fig. 4 respectively in which shaded areas show standard deviation for three seeds. In both scenarios, our LfR method outperforms the other three baseline methods. For the first scenario, SAC succeeds in learning the `Lift` task on Panda due to the simplicity of the `Lift` task, but the learning progress is very slow compared with our LfR method. In contrast, BC+RNN and AP fails to learn any task in both experimental scenarios. It shows that transfer via direct finetuning does not perform well if morphology of the target hardware differs much. We conclude that in our experiments it is essential to transfer skills with cycle generative models on a new robot system.

We evaluate the performance of the trained policies by testing 3 times and each test has 10 independent trials. We calculate the mean accuracy and the standard deviation. The evaluation results for two experiment scenarios are shown in Table I and Table II respectively. For the first scenario (`Pick-Place`, `Nut-Assembly` as common tasks), LfR enables Panda and Sawyer to achieve `Lift` task with 98.0% success rate and `Stack` tasks with 92.0% success rate respectively. And when deploying the trained policy with finetuning on another robot platform, we get 56.6% and 58.0% which means the finetuned policy achieves occasional success for the new task. BC+RNN performs worst in both tasks compared other methods. We also notice that SAC achieves 89% success rate for the `Lift` task, but only obtains 40.5% for the `Stack` in the first scenario. If we consider `Pick-Place` task in the second scenario, all baseline methods' performances drop substantially. We can see that SAC fails to learn this task from scratch and cannot accomplish a single `Pick-Place` trial. BC+RNN and AP both obtain occasional success in the `Stack` experiments and their performances decrease noticeably. In comparison, our LfR shows stable accomplishment although there is a marginal performance degradation for the `Pick-Place` task.

TABLE I

SUCCESS RATE FOR TRANSFERRING SCENARIO 1

Task	BC+RNN	AP	SAC	Ours
Lift	42.0 ± 2.2	56.6 ± 1.4	89.0 ± 0.2	$\mathbf{98.0 \pm 0.6}$
Stack	36.0 ± 1.0	58.0 ± 0.6	40.5 ± 3.2	$\mathbf{92.0 \pm 0.7}$

TABLE II

SUCCESS RATE FOR TRANSFERRING SCENARIO 2

Task	BC+RNN	AP	SAC	Ours
Stack	42.8 ± 1.5	37.0 ± 3.5	30.2 ± 2.5	$\mathbf{90.2 \pm 1.7}$
Pick-Place	16.3 ± 4.4	10.6 ± 5.7	0.0 ± 0.0	$\mathbf{82.5 \pm 3.2}$

C. Ablation Evaluation

TABLE III

ABLATION RESUTLS

Task	No Generator	Dataset Lift	Dataset Nut-Assembly
Stack	5.0 ± 1.6	67.8 ± 2.2	87.6 ± 1.8
Pick-Place	3.5 ± 1.2	42.4 ± 3.2	76.0 ± 2.5

To validate the efficacy of our LfR method, we compare its performance without generative models and only using one common task to learn the cycle generative model. In this ablation experiment, we consider the test scenario 2 in which Panda learns the task `Stack` and Sawyer learns the task `Pick-Place`. We evaluate the impact of training the cycle generative model by using different common task datasets. The results are shown in Table III. It can be seen that without using the generative model, both robots struggle with the new task. On average Panda achieves 5.0% with task `Stack` and Sawyer only finishes 3.5% with task `Pick-Place`. When only using common task `Lift` to train the generative model, we get success rates 67.8% and 42.4% respectively. While with common task `Nut-Assembly`, the success rates are 87.6% and 76.0% that are marginally less favorable than our previous results. We conclude that task `Nut-Assembly`

encompasses a wide range of the latent action spaces for the new tasks.

VI. CONCLUSION AND FUTURE WORK

In this paper we learn to transfer prior knowledge between robots, which enables us to quickly learn policies for novel manipulation tasks. Each policy is expressed as actions in a latent skill space. We study the cross-domain skill transfer problem and propose to utilize a cycle generative model to predict the action distribution in the target robot domain. We extend entropy regularization for learning new policies by concatenating the policy with the pre-trained generative model. By concatenating the policy model learned on one robot with a pre-trained generative network, our method allows the robot to learn from the skills of another robot. We evaluate our method in simulation experiments for several robot tasks and show that our method can be generalized to unseen tasks on different robot platforms.

In our work, we utilize the identity mapping of the observation space for two agents which requires that the robots share the same input dimension and similar kinematics. A interesting future work is to investigate how to transfer the observation space for the robots more generally and effectively. Moreover, in this work we evaluate our method in simulation. It will be valuable to show the efficacy of transferring skills using our LfR method for robots from simulation to reality.

REFERENCES

[1] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[2] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[3] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4243–4250.

[4] T. Chen, A. Murali, and A. Gupta, "Hardware conditioned policies for multi-robot transfer learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[5] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[6] M. Hazara and V. Kyrki, "Transferring generalizable motor primitives from simulation to real world," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2172–2179, 2019.

[7] A. A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," in *Conference on robot learning*. PMLR, 2017, pp. 262–270.

[8] A. Ghadirzadeh, X. Chen, P. Poklukar, C. Finn, M. Björkman, and D. Kragic, "Bayesian meta-learning for few-shot policy adaptation across robotic platforms," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1274–1280.

[9] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," in *International Conference on Learning Representations*.

[10] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari, "Rl-cyclegan: Reinforcement learning aware simulation-to-real," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 157–11 166.

[11] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey." *Journal of Machine Learning Research*, vol. 10, no. 7, 2009.

[12] Z.-H. Yin, L. Sun, H. Ma, M. Tomizuka, and W.-J. Li, "Cross domain robot imitation with invariant representation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 455–461.

[13] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Conference on Robot Learning*. PMLR, 2017, pp. 334–343.

[14] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," *arXiv preprint arXiv:1703.02949*, 2017.

[15] Q. Yang, J. A. Stork, and T. Stoyanov, "Mpr-rl: Multi-prior regularized reinforcement learning for knowledge transfer," *IEEE Robotics and Automation Letters*, 2022.

[16] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv:2108.03298*, 2021.

[17] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[18] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills," *arXiv preprint arXiv:1802.09564*, 2018.

[19] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[20] K. Zolna, S. Reed, A. Novikov, S. G. Colmenarejo, D. Budden, S. Cabi, M. Denil, N. de Freitas, and Z. Wang, "Task-relevant adversarial imitation learning," in *Conference on Robot Learning*. PMLR, 2021, pp. 247–263.

[21] O. Mees, M. Merklinger, G. Kalweit, and W. Burgard, "Adversarial skill networks: Unsupervised robot skill learning from video," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4188–4194.

[22] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.

[23] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *International conference on machine learning*. PMLR, 2019, pp. 5331–5340.

[24] K. Arndt, M. Hazara, A. Ghadirzadeh, and V. Kyrki, "Meta reinforcement learning for sim-to-real domain adaptation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2725–2731.

[25] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 2169–2176.

[26] K. Pertsch, Y. Lee, and J. Lim, "Accelerating reinforcement learning with learned skill priors," in *Conference on robot learning*. PMLR, 2021, pp. 188–204.

[27] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *CoRR*, vol. abs/1312.6114, 2014.

[28] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *International conference on machine learning*. PMLR, 2014, pp. 1278–1286.

[29] Q. Yang, A. Dürr, E. A. Topp, J. A. Stork, and T. Stoyanov, "Variable impedance skill learning for contact-rich manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8391–8398, 2022.

[30] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín, "robosuite: A modular simulation framework and benchmark for robot learning," *arXiv preprint arXiv:2009.12293*, 2020.

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[33] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.